

MX



macromedia®  
**FLASH**<sup>TM</sup>MX  
2004

Utilización de componentes

## Marcas comerciales

Add Life to the Web, Afterburner, Aftershock, Andromedia, Allaire, Animation PowerPack, Aria, Attain, Authorware, Authorware Star, Backstage, Bright Tiger, Clustercats, ColdFusion, Contribute, Design In Motion, Director, Dream Templates, Dreamweaver, Drumbeat 2000, EDJE, EJIPT, Extreme 3D, Fireworks, Flash, Fontographer, FreeHand, Generator, HomeSite, JFusion, JRun, Kawa, Know Your Site, Knowledge Objects, Knowledge Stream, Knowledge Track, LikeMinds, Lingo, Live Effects, el logotipo y el diseño de MacRecorder, Macromedia, Macromedia Action!, Macromedia Flash, el logotipo y el diseño de Macromedia M, Macromedia Spectra, el logotipo y el diseño de Macromedia xRes, MacroModel, Made with Macromedia, el logotipo y el diseño de Made with Macromedia, el logotipo y el diseño de MAGIC, Mediamaker, Movie Critic, Open Sesame!, Roundtrip, Roundtrip HTML, Shockwave, Sitespring, SoundEdit, Titlemaker, UltraDev, Web Design 101, what the web can be y Xtra son marcas registradas o marcas comerciales de Macromedia, Inc. y pueden estar registradas en Estados Unidos o en otras jurisdicciones, incluidas las internacionales. Otros nombres de productos, logotipos, diseños, títulos, palabras o frases mencionados en esta publicación pueden ser marcas comerciales, marcas de servicio o nombres registrados de Macromedia, Inc. o de otras entidades y pueden estar registrados en ciertas jurisdicciones, incluidas las internacionales.

## Información de terceros

Esta guía contiene vínculos a sitios Web de terceros que no están bajo el control de Macromedia y, por consiguiente, Macromedia no se hace responsable del contenido de dichos sitios Web. El acceso a uno de los sitios Web de terceros mencionados en esta guía será a cuenta y riesgo del usuario. Macromedia proporciona estos vínculos únicamente como ayuda y su inclusión no implica que Macromedia se haga responsable del contenido de dichos sitios Web.

La tecnología de compresión y descompresión de voz tiene licencia de Nellymoser, Inc. ([www.nellymoser.com](http://www.nellymoser.com)).



La tecnología de compresión y descompresión de vídeo Sorenson™ Spark™ tiene licencia de Sorenson Media, Inc.

Navegador Opera® Copyright © 1995-2002 Opera Software ASA y sus proveedores. Todos los derechos reservados.

## Limitación de garantías de Apple

**APPLE COMPUTER, INC. NO GARANTIZA, DE FORMA EXPRESA NI IMPLÍCITA, LA COMERCIABILIDAD O IDONEIDAD PARA UN FIN DETERMINADO DEL PAQUETE DE SOFTWARE INFORMÁTICO INCLUIDO. LA EXCLUSIÓN DE GARANTÍAS IMPLÍCITAS NO ESTÁ PERMITIDA EN ALGUNOS ESTADOS. LA RESTRICCIÓN ANTERIOR PUEDE NO AFECTARLE. ESTA GARANTÍA LE PROPORCIONA DERECHOS LEGALES ESPECÍFICOS. PUEDE TENER OTROS DERECHOS QUE VARÍAN SEGÚN LA LEGISLACIÓN LOCAL.**

**Copyright © 2003 Macromedia, Inc. Todos los derechos reservados. No se permite la copia, fotocopia, reproducción, traducción ni la conversión en formato electrónico o legible por equipos, ya sea de forma total o parcial de este manual, sin la autorización previa por escrito de Macromedia, Inc. Número de referencia ZFL70M500SP**

## Agradecimientos

Director: Erick Vera

Dirección del proyecto: Stephanie Gowin, Barbara Nelson

Redacción: Jody Bleye, Mary Burger, Kim Diezel, Stephanie Gowin, Dan Harris, Barbara Herbert, Barbara Nelson, Shirley Ong, Tim Statler

Directora de edición: Rosana Francescato

Edición: Mary Ferguson, Mary Kraemer, Noreen Maher, Antonio Padial, Lisa Stanziano, Anne Szabla

Administración de la producción: Patrice O'Neill

Producción y diseño multimedia: Adam Barnett, Christopher Basmajian, Aaron Begley, John Francis, Jeff Harmon

Localización: Tim Hussey, Seungmin Lee, Masayo Noda, Simone Pux, Yuko Yagi, Jorge G. Villanueva

Primera edición: octubre de 2003

Macromedia, Inc.  
600 Townsend St.  
San Francisco, CA 94103, EE UU

# CONTENIDO

|                                                                                            |    |
|--------------------------------------------------------------------------------------------|----|
| <b>INTRODUCCIÓN:</b> Primeros pasos con los componentes. . . . .                           | 7  |
| Audiencia . . . . .                                                                        | 7  |
| Requisitos del sistema . . . . .                                                           | 8  |
| Instalación de componentes. . . . .                                                        | 8  |
| Documentación. . . . .                                                                     | 9  |
| Convenciones tipográficas . . . . .                                                        | 10 |
| Términos utilizados en este manual . . . . .                                               | 10 |
| Recursos adicionales . . . . .                                                             | 10 |
| <br><b>CAPÍTULO 1:</b> Los componentes . . . . .                                           | 11 |
| Ventajas de los componentes de la v2. . . . .                                              | 12 |
| Categorías de componentes . . . . .                                                        | 12 |
| Arquitectura de componentes . . . . .                                                      | 13 |
| Novedades en los componentes de la v2 . . . . .                                            | 14 |
| Clips compilados y archivos SWC. . . . .                                                   | 14 |
| Accesibilidad y componentes . . . . .                                                      | 15 |
| <br><b>CAPÍTULO 2:</b> Trabajo con componentes . . . . .                                   | 17 |
| Panel Componentes . . . . .                                                                | 17 |
| Componentes del panel Biblioteca. . . . .                                                  | 18 |
| Componentes del panel Inspector de componentes y del inspector<br>de propiedades . . . . . | 18 |
| Componentes en vista previa dinámica . . . . .                                             | 20 |
| Trabajo con archivos SWC y clips compilados. . . . .                                       | 20 |
| Adición de componentes a documentos de Flash. . . . .                                      | 21 |
| Definición de parámetros de componentes . . . . .                                          | 24 |
| Eliminación de componentes de documentos de Flash . . . . .                                | 24 |
| Utilización de las sugerencias para el código . . . . .                                    | 24 |
| Eventos de componentes . . . . .                                                           | 25 |
| Creación de un desplazamiento personalizado de la selección . . . . .                      | 27 |
| Administración de la profundidad del componente en un documento. . . . .                   | 28 |
| Utilización de preloader con componentes . . . . .                                         | 29 |
| Actualización de componentes de la versión 1 a la arquitectura de la versión 2 . . . .     | 29 |

|                                                                                         |        |
|-----------------------------------------------------------------------------------------|--------|
| <b>CAPÍTULO 3: Personalización de componentes</b> . . . . .                             | 31     |
| Utilización de estilos para personalizar el texto y el color de un componente . . . . . | 31     |
| Temas . . . . .                                                                         | 39     |
| Aplicación de aspectos a los componentes . . . . .                                      | 41     |
| <br><b>CAPÍTULO 4: Diccionario de componentes</b> . . . . .                             | <br>49 |
| Componentes de la interfaz de usuario (IU) . . . . .                                    | 49     |
| Componentes de datos . . . . .                                                          | 50     |
| Componentes multimedia . . . . .                                                        | 51     |
| Administradores . . . . .                                                               | 51     |
| Pantallas . . . . .                                                                     | 52     |
| Componente Accordion (sólo para Flash Professional) . . . . .                           | 52     |
| Componente Alert (sólo Flash Professional) . . . . .                                    | 66     |
| Componente Button . . . . .                                                             | 75     |
| API CellRenderer . . . . .                                                              | 87     |
| Componente CheckBox . . . . .                                                           | 94     |
| Componente ComboBox . . . . .                                                           | 102    |
| Clases de vinculación de datos (sólo para Flash Professional) . . . . .                 | 131    |
| Componente DataGrid (sólo para Flash Professional) . . . . .                            | 166    |
| Componente DataHolder (sólo para Flash Professional) . . . . .                          | 198    |
| Interfaz API de DataProvider . . . . .                                                  | 200    |
| Componente DataSet (sólo para Flash Professional) . . . . .                             | 211    |
| Componente DateChooser (sólo para Flash Professional) . . . . .                         | 257    |
| Componente DateField (sólo para Flash Professional) . . . . .                           | 269    |
| Clase DepthManager . . . . .                                                            | 286    |
| Clase FocusManager . . . . .                                                            | 292    |
| Clase Form (sólo para Flash Professional) . . . . .                                     | 299    |
| Componente Label . . . . .                                                              | 305    |
| Componente List . . . . .                                                               | 310    |
| Componente Loader . . . . .                                                             | 339    |
| Componentes multimedia (sólo para Flash Professional) . . . . .                         | 350    |
| Componente Menu (sólo para Flash Professional) . . . . .                                | 392    |
| Componente MenuBar (sólo para Flash Professional) . . . . .                             | 423    |
| Componente NumericStepper . . . . .                                                     | 434    |
| Clase PopUpManager . . . . .                                                            | 443    |
| Componente ProgressBar . . . . .                                                        | 445    |
| Componente RadioButton . . . . .                                                        | 460    |
| Componente RDBMSResolver (sólo para Flash Professional) . . . . .                       | 470    |
| Interfaz API del componente Remote Procedure Call (RPC) . . . . .                       | 481    |
| Clase Screen (sólo para Flash Professional) . . . . .                                   | 486    |
| Componente ScrollPane . . . . .                                                         | 499    |
| Clase Slide (sólo para Flash Professional) . . . . .                                    | 514    |
| Clase StyleManager . . . . .                                                            | 540    |
| Componente TextArea . . . . .                                                           | 542    |
| Componente TextInput . . . . .                                                          | 554    |
| Interfaz TransferObject . . . . .                                                       | 566    |
| Componente Tree (sólo para Flash Professional) . . . . .                                | 568    |
| Interfaz TreeDataProvider (sólo para Flash Professional) . . . . .                      | 588    |
| UIComponent . . . . .                                                                   | 593    |



|                                                                     |         |
|---------------------------------------------------------------------|---------|
| UIEventDispatcher . . . . .                                         | 600     |
| UIObject. . . . .                                                   | 603     |
| Clases de servicios Web (sólo para Flash Professional) . . . . .    | 622     |
| WebServiceConnector (sólo para Flash Professional) . . . . .        | 646     |
| Componente Window. . . . .                                          | 657     |
| Componente XMLConnector (sólo para Flash Professional) . . . . .    | 668     |
| Componente XUpdateResolver (sólo para Flash Professional) . . . . . | 677     |
| <br><b>CAPÍTULO 5: Creación de componentes . . . . .</b>            | <br>683 |
| Novedades. . . . .                                                  | 683     |
| Trabajo con el entorno de Flash. . . . .                            | 683     |
| Creación de componentes . . . . .                                   | 686     |
| Escritura de ActionScript del componente. . . . .                   | 688     |
| Importación de clases . . . . .                                     | 690     |
| Selección de una clase principal. . . . .                           | 690     |
| Escritura del constructor . . . . .                                 | 692     |
| Control de versiones . . . . .                                      | 692     |
| Nombres de clase, símbolo y propietario . . . . .                   | 693     |
| Definición de captadores y definidores . . . . .                    | 693     |
| Metadatos de componente. . . . .                                    | 694     |
| Definición de parámetros de componentes . . . . .                   | 700     |
| Implementación de métodos principales . . . . .                     | 700     |
| Gestión de eventos . . . . .                                        | 701     |
| Aplicación de aspectos . . . . .                                    | 705     |
| Adición de estilos . . . . .                                        | 706     |
| Cómo facilitar el acceso a los componentes . . . . .                | 706     |
| Exportación del componente. . . . .                                 | 707     |
| Cómo facilitar la utilización del componente. . . . .               | 709     |
| Recomendaciones para el diseño de un componente . . . . .           | 710     |
| <br><b>ÍNDICE ALFABÉTICO . . . . .</b>                              | <br>711 |



# INTRODUCCIÓN

## Primeros pasos con los componentes

Macromedia Flash MX 2004 y Macromedia Flash MX Professional 2004 son las herramientas estándar de edición profesional para la creación de publicaciones Web de gran impacto. Los componentes son bloques de creación para aplicaciones de Internet ricas en contenido que proporcionan dicho impacto. Un componente es un clip de película con parámetros que se definen durante la edición en Macromedia Flash. El código ActionScript es una interfaz API que permite a los desarrolladores volver a utilizar y compartir código, así como encapsular complejas funciones que los diseñadores pueden utilizar y personalizar sin necesidad de utilizar ActionScript.

Los componentes están integrados en la versión 2 (v2) de la arquitectura de componentes de Macromedia, lo que permite crear con facilidad y rapidez aplicaciones sólidas con apariencia y comportamiento uniformes. En este manual se explica la forma de crear aplicaciones con componentes de la v2 y se describen las interfaces API (interfaz de programación de aplicaciones) de los componentes. Incluye ejemplos de escenarios y procedimientos que utilizar con los componentes de la v2 de Flash MX 2004 o Flash MX Professional 2004, así como descripciones de las interfaces API de los componentes, en orden alfabético.

Puede utilizar los componentes creados por Macromedia, descargar componentes creados por otros desarrolladores o crear los suyos propios.

## Audiencia

Este manual está dirigido a desarrolladores que crean aplicaciones de Flash MX 2004 o Flash MX Professional 2004 y desean utilizar componentes para agilizar el desarrollo. Deberá estar familiarizado con el desarrollo de aplicaciones en Macromedia Flash, con el código ActionScript y con Macromedia Flash Player.

En el manual se presupone que el usuario ya ha instalado Flash MX 2004 o Flash MX Professional 2004 y sabe cómo utilizarlo. Antes de utilizar los componentes, debe completar la lección “Crear una interfaz de usuario con componentes” (seleccione Ayuda > Cómo > Inicio rápido > Crear una interfaz de usuario con componentes).

Si desea escribir el menor código ActionScript posible, puede arrastrar los componentes a un documento, establecer sus parámetros en el inspector de propiedades o en el panel Inspector de componentes y adjuntar un controlador `on()` directamente a un componente en el panel Acciones para controlar los eventos de componentes.

Si es un programador que desea crear aplicaciones más sólidas, puede crear componentes de forma dinámica, utilizar sus interfaces API para establecer propiedades y llamar a métodos en tiempo de ejecución, así como utilizar el modelo de eventos detector para controlar los eventos.

Para más información, consulte el [Capítulo 2, “Trabajo con componentes”](#), en la [página 17](#).

## Requisitos del sistema

Los componentes de Macromedia no tienen ningún requisito del sistema adicional a los de Flash MX 2004 o Flash MX Professional 2004.

## Instalación de componentes

Al iniciar por primera vez Flash MX 2004 o Flash MX Professional 2004, ya hay un conjunto de componentes de Macromedia instalado. Pueden verse en el panel Componentes.

Flash MX 2004 incluye los componentes siguientes:

- [Componente Button](#)
- [Componente CheckBox](#)
- [Componente ComboBox](#)
- [Componente Label](#)
- [Componente List](#)
- [Componente Loader](#)
- [Componente NumericStepper](#)
- [Componente ProgressBar](#)
- [Componente RadioButton](#)
- [Componente ScrollPane](#)
- [Componente TextArea](#)
- [Componente TextInput](#)
- [Componente Window](#)

Flash MX Professional 2004 incluye los componentes de Flash MX 2004 y los siguientes componentes y clases adicionales:

- [Componente Accordion](#) (sólo para Flash Professional)
- [Componente Alert](#) (sólo Flash Professional)
- [Clases de vinculación de datos](#) (sólo para Flash Professional)
- [Componente DateField](#) (sólo para Flash Professional)
- [Componente DataGrid](#) (sólo para Flash Professional)
- [Componente DataHolder](#) (sólo para Flash Professional)
- [Componente DataSet](#) (sólo para Flash Professional)
- [Componente DateChooser](#) (sólo para Flash Professional)
- [Clase Form](#) (sólo para Flash Professional)
- [Componentes multimedia](#) (sólo para Flash Professional)
- [Componente Menu](#) (sólo para Flash Professional)

- Componente MenuBar (sólo para Flash Professional)
- Componente RDBMSResolver (sólo para Flash Professional)
- Clase Screen (sólo para Flash Professional)
- Clase Slide (sólo para Flash Professional)
- Componente Tree (sólo para Flash Professional)
- Clase WebServiceConnector (sólo para Flash Professional)
- Componente XMLConnector (sólo para Flash Professional)
- Componente XUpdateResolver (sólo para Flash Professional)

**Para verificar la instalación de los componentes de Flash MX 2004 o Flash MX Professional 2004:**

- 1 Inicie Flash.
- 2 Seleccione Ventana > Paneles de desarrollo > Componentes para abrir el panel Componentes, si aún no está abierto.
- 3 Seleccione UI Components para expandir el árbol y ver los componentes instalados.

También puede descargar componentes de [Macromedia Exchange](#). Para instalar componentes descargados de Exchange, descargue e instale [Macromedia Extensions Manager](#).

Todos los componentes, ya se trate de un archivo SWC o de un archivo FLA (véase “[Clips compilados y archivos SWC](#)” en la página 14), pueden aparecer en el panel Componentes de Flash. Realice estos pasos para instalar componentes en un equipo con Windows o Macintosh.

**Para instalar componentes en un equipo con Windows o Macintosh:**

- 1 Salga de Flash.
- 2 Coloque el archivo SWC o FLA que contiene el componente en la siguiente carpeta del disco duro:
  - \Archivos de programa\Macromedia\Flex MX 2004\<idioma>\First Run\Components (Windows)
  - Disco duro/Applications/Macromedia Flash MX 2004/First Run/Components (Macintosh)
- 3 Abra Flash.
- 4 Seleccione Ventana > Paneles de desarrollo > Componentes para ver los componentes del panel Componentes, si aún no está abierto.

## Documentación

En este documento se explican los detalles de la utilización de componentes para desarrollar aplicaciones de Flash. Se presupone que el lector tiene conocimientos generales de Macromedia Flash y ActionScript. Hay disponible por separado documentación específica sobre Flash y productos relacionados.

- Para información sobre Macromedia Flash, consulte *Primeros pasos con Flash* (o Primeros pasos de la Ayuda), el apartado Utilización de Flash de la Ayuda, el apartado Guía de referencia de ActionScript de la Ayuda y Diccionario de ActionScript de la Ayuda.
- Para obtener información sobre el acceso a los servicios Web con aplicaciones de Flash, consulte *Utilización de convirtiendo a remoto de Flash*.

## Convenciones tipográficas

En este manual se utilizan las siguientes convenciones tipográficas:

- *Fuente en cursiva* indica un valor que se debe sustituir (por ejemplo, en una ruta de carpeta).
- La fuente para código indica que se trata de código de ActionScript.
- *Fuente para código en cursiva* indica un parámetro de ActionScript.
- **Fuente en negrita** indica una entrada de caracteres.

**Nota:** la fuente negrita no es la misma que la fuente utilizada en los encabezados run-in, es decir, los encabezados que aparecen en la misma línea que el texto al que introducen. La fuente de los encabezados run-in se utiliza como una alternativa a las viñetas.

## Términos utilizados en este manual

En este manual se utilizan los términos siguientes:

**en tiempo de ejecución** Cuando el código se ejecuta en Flash Player.

**durante la edición** Mientras se trabaja en el entorno de edición de Flash.

## Recursos adicionales

Para obtener información reciente sobre Flash, además de las sugerencias de usuarios expertos, temas avanzados, ejemplos, consejos y otras actualizaciones, consulte el sitio Web [Macromedia DevNet](#), que se actualiza con regularidad. Visite el sitio Web regularmente para conocer las últimas noticias sobre Flash y cómo sacar el máximo partido del programa.

Para información sobre las notas técnicas, actualizaciones de documentación y vínculos a otros recursos de la comunidad de Flash, consulte Centro de soporte de Macromedia Flash en [www.macromedia.com/go/flash\\_support\\_es](http://www.macromedia.com/go/flash_support_es).

Para información detallada sobre los términos, la sintaxis y la utilización de ActionScript, consulte los apartados Guía de referencia de ActionScript y Diccionario de ActionScript de la Ayuda.

Para obtener una introducción sobre cómo utilizar los componentes, acceda a Macromedia On Demand Seminar, Flash MX 2004 Family, y consulte el apartado Using UI Components en [www.macromedia.com/macromedia/events/online/ondemand/index.html](http://www.macromedia.com/macromedia/events/online/ondemand/index.html).

# CAPÍTULO 1

## Los componentes

Los componentes son clips de película con parámetros que permiten modificar la apariencia y el comportamiento. Un componente puede proporcionar cualquier función que su creador pueda imaginar. Puede ser desde un sencillo control de interfaz de usuario como un botón de opción o una casilla de verificación, hasta un elemento con contenido, como un panel de desplazamiento; por otra parte, un componente también puede ser un elemento que no se ve, como FocusManager, que permite controlar qué objeto pasa a estar seleccionado en una aplicación.

Los componentes permiten a cualquier usuario crear complejas aplicaciones de Macromedia Flash MX 2004 y Macromedia Flash MX Professional 2004, aunque no tenga conocimientos avanzados de ActionScript. En lugar de crear botones personalizados, cuadros combinados y listas, basta con arrastrar dichos componentes desde el panel Componentes para añadir funciones a las aplicaciones. Asimismo, la apariencia de los componentes se puede personalizar fácilmente y conseguir, de esta manera, una mayor adaptación a las necesidades de diseño propias.

Los componentes están integrados en la versión 2 (v2) de la arquitectura de componentes de Macromedia, lo que permite crear con facilidad y rapidez aplicaciones sólidas con apariencia y comportamiento uniformes. La arquitectura de la v2 incluye clases en las que se basan todos los componentes, mecanismos de estilo y de aspecto para personalizar fácilmente la apariencia de los componentes, un modelo de eventos difusor/detector, gestión de profundidad y selección, implementación de accesibilidad, etc.

Cada componente dispone de parámetros predefinidos que se pueden establecer mientras se edita en Flash. Asimismo, cada uno dispone de un conjunto único de métodos, propiedades y eventos de ActionScript, llamado también *API* (interfaz de programación de aplicaciones), que permite definir parámetros y opciones adicionales en tiempo de ejecución.

Flash MX 2004 y Flash MX Professional 2004 incluyen muchos componentes de Flash nuevos y varias versiones nuevas de componentes que se incluyeron en Flash MX. Para obtener una lista completa de los componentes que se incluyen con Flash MX 2004 y Flash MX Professional 2004, consulte [“Instalación de componentes” en la página 8](#). También puede descargar componentes creados por miembros de la comunidad de Flash en [Macromedia Exchange](#).

## Ventajas de los componentes de la v2

Los componentes permiten establecer la separación entre codificación y diseño. Asimismo, permiten volver a utilizar código, ya sea en componentes que se creen, o bien descargando e instalando componentes creados por otros desarrolladores.

Los componentes permiten a los programadores crear funciones que los diseñadores pueden utilizar en aplicaciones. Los desarrolladores pueden encapsular en componentes las funciones utilizadas con mayor frecuencia, y los diseñadores pueden personalizar el aspecto y el comportamiento de los componentes cambiando los parámetros en el inspector de propiedades o en el panel Inspector de componentes.

Los miembros de la comunidad de Flash pueden utilizar [Macromedia Exchange](#) para intercambiar componentes. Si utiliza componentes, ya no tendrá que crear cada uno de los elementos de una aplicación Web compleja desde cero. Simplemente, busque los componentes que necesite y póngalos en un documento de Flash para crear una aplicación nueva.

Los componentes que se basan en la arquitectura de componentes de la v2 comparten una funcionalidad básica, como estilos, gestión de eventos, aplicación de aspectos, gestión de selección y de profundidad. Cuando se añade el primer componente de la v2 a una aplicación, se añaden alrededor de 25 K al documento, que incluyen dicha funcionalidad básica. Si añade más componentes, esos mismos 25 K se volverán a utilizar también para ellos, por lo que el aumento de tamaño del documento será inferior al esperado. Para más información sobre cómo actualizar los componentes de la v1 a la v2, consulte [“Actualización de componentes de la versión 1 a la arquitectura de la versión 2” en la página 29](#).

## Categorías de componentes

Los componentes incluidos en Flash MX 2004 y Flash MX Professional 2004 se dividen en cinco categorías: componentes de la interfaz de usuario, componentes de datos, componentes de medios, administradores y pantallas. Los componentes de la interfaz de usuario permiten interactuar con una aplicación. Por ejemplo, los componentes `RadioButton`, `CheckBox` y `TextInput` son controles de la interfaz de usuario. Los componentes de datos permiten cargar y manipular la información de las fuentes de datos; los componentes `WebServiceConnector` y `XMLConnector` son componentes de datos. Los componentes de medios permiten reproducir y controlar flujos de medios. `MediaController`, `MediaPlayback` y `MediaDisplay` son los componentes de medios. Los administradores son componentes que no se ven y que permiten gestionar una función (por ejemplo, selección o profundidad) en una aplicación. `FocusManager`, `DepthManager`, `PopUpManager` y `StyleManager` son los componentes de administrador que se incluyen con Flash MX 2004 y Flash MX Professional 2004. La categoría de pantallas incluye las clases de `ActionScript` que permiten controlar formularios y diapositivas de Flash MX Professional 2004. Para obtener una lista completa de cada categoría, consulte el [Capítulo 4, “Diccionario de componentes”](#), en la página 49.



## Arquitectura de componentes

Puede utilizar el inspector de propiedades o el panel Inspector de componentes para cambiar los parámetros del componente y utilizar las funciones básicas de los componentes. No obstante, si desea tener un mayor control sobre los componentes, tendrá que utilizar las interfaces API de los mismos y poseer conocimientos sobre cómo se crearon.

Los componentes de Flash MX 2004 y Flash MX Professional 2004 se crean utilizando la versión 2 (v2) de la arquitectura de componentes de Macromedia. Flash Player 6 y Flash Player 7 admiten los componentes de la versión 2. Dichos componentes no siempre son compatibles con componentes creados mediante la arquitectura de la versión 1 (v1) (todos los componentes anteriores a Flash MX 2004). Asimismo, Flash Player 7 no admite los componentes de la versión 1. Para más información, consulte [“Actualización de componentes de la versión 1 a la arquitectura de la versión 2” en la página 29](#).

Los componentes de la v2 están incluidos en el panel Componentes como símbolos de clip compilado (SWC). Un clip compilado es un clip de película de componente cuyo código se ha compilado. Los clips compilados tienen vistas previas dinámicas incorporadas y no se pueden editar, aunque se pueden cambiar los parámetros en el inspector de propiedades o en el panel Inspector de componentes, al igual que con cualquier otro componente. Para más información, consulte [“Clips compilados y archivos SWC” en la página 14](#).

Los componentes V2 se escriben en ActionScript 2.0. Cada componente es una clase y cada clase se encuentra en un paquete de ActionScript. Por ejemplo, un componente de botón de opción es una instancia de la clase `RadioButton` cuyo nombre de paquete es `mx.controls`. Para más información sobre paquetes, consulte [“Utilización de paquetes” en el apartado Guía de referencia de ActionScript de la Ayuda](#).

Todos los componentes creados con la versión 2 de la arquitectura de componentes de Macromedia son subclases de las clases `UIObject` y `UIComponent`, y heredan todos los métodos, eventos y propiedades de dichas clases. Asimismo, muchos componentes son subclases de otros componentes. La ruta de herencia de cada componente se muestra en el [Capítulo 4, “Diccionario de componentes”, en la página 49](#).

Todos los componentes utilizan también el mismo modelo de evento, los mismos estilos basados en CSS y el mismo mecanismo de aplicación de aspectos incorporados. Para más información sobre los estilos y la aplicación de aspectos, consulte el [Capítulo 3, “Personalización de componentes”, en la página 31](#). Para más información sobre la gestión de eventos, consulte el [Capítulo 2, “Trabajo con componentes”, en la página 17](#).

## Novedades en los componentes de la v2

El **panel Inspector de componentes** le permite cambiar los parámetros de los componentes al mismo tiempo que edita en Macromedia Flash y en Macromedia Dreamweaver. (Véase [“Componentes del panel Inspector de componentes y del inspector de propiedades” en la página 18.](#))

**Modelo de eventos del detector** permite a los objetos detectores de funciones gestionar eventos. (Véase [“Eventos de componentes” en la página 25.](#))

Las **propiedades de aspecto** permiten cargar estados sólo cuando es necesario. (Véase [“Aplicación de aspectos a los componentes” en la página 41.](#))

**Estilos basados en CSS** permite crear una apariencia uniforme en todas las aplicaciones. (Véase [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31.](#))

**Temas** permite arrastrar una apariencia nueva a un conjunto de componentes. (Véase [“Temas” en la página 39.](#))

**Tema Halo** proporciona a las aplicaciones una interfaz de usuario lista, útil y flexible.

**Clases de administrador** proporciona una manera fácil de gestionar la selección y profundidad de una aplicación. (Véase [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) y [“Administración de la profundidad del componente en un documento” en la página 28.](#))

Las **clases básicas UIObject y UIComponent** proporcionan funciones básicas a todos los componentes. (Véase [“UIComponent” en la página 593](#) y [“UIObject” en la página 603.](#))

**Empaquetado como archivo SWC** permite una fácil distribución y ocultación de código. Véase el [Capítulo 5, “Creación de componentes”, en la página 683.](#)

**Vinculación de datos incorporados** está disponible mediante el panel Inspector de componentes. Para más información sobre esta función, presione el botón de actualización de la Ayuda.

**Jerarquía de clases ampliable fácilmente** mediante ActionScript 2.0 le permite crear namespaces exclusivos, importar clases cuando sea necesario y poner fácilmente en subclases para ampliar componentes. Véase el [Capítulo 5, “Creación de componentes”, en la página 683](#) y el apartado Guía de referencia de ActionScript de la Ayuda.

## Clips compilados y archivos SWC

Los clips compilados se utilizan para compilar previamente símbolos complejos de un documento de Flash. Por ejemplo, un clip de película con gran cantidad de código ActionScript que no cambie a menudo puede convertirse en un clip compilado. Como resultado, tanto el comando Probar película como Publicar necesitarían menos tiempo para ejecutarse.

Un archivo SWC es el tipo de archivo utilizado para guardar y distribuir componentes. Si coloca un archivo SWC en la carpeta First Run\Components, el componente aparecerá en el panel Componentes. Si añade un componente al escenario desde el panel Componentes, se añadirá un símbolo de clip compilado a la biblioteca.

Para más información sobre archivos SWC, consulte el [Capítulo 5, “Creación de componentes”, en la página 683.](#)

## Accesibilidad y componentes

Cada vez resulta más importante que el contenido Web sea accesible, es decir, que lo puedan utilizar usuarios con diferentes discapacidades. Los invidentes pueden acceder al contenido visual de las aplicaciones de Flash mediante la utilización de un software de lectura de pantalla que proporciona una descripción hablada del contenido de la pantalla.

Cuando se crea un componente, el autor puede escribir código ActionScript que permite la comunicación entre el componente y el lector de pantalla. A continuación, cuando el desarrollador utiliza componentes para crear una aplicación en Flash, recurre al panel Accesibilidad para configurar la instancia de cada componente.

La mayoría de los componentes creados por Macromedia están diseñados teniendo en cuenta la accesibilidad. Para saber si un componente es accesible, consulte su entrada en el [Capítulo 4, “Diccionario de componentes”](#), en la [página 49](#). Cuando cree una aplicación en Flash, tendrá que añadir una línea de código por cada componente

`(mx.accessibility.ComponentNameAccImpl.enableAccessibility());` y definir los parámetros de accesibilidad en el panel Accesibilidad. La accesibilidad de los componentes funciona igual que para todos los clips de película de Flash. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda.

Existe la posibilidad de navegar con el teclado por la mayoría de los componentes creados por Macromedia. Cada entrada del componente del [Capítulo 4, “Diccionario de componentes”](#), en la [página 49](#) indica si puede controlar o no el componente con el teclado.



# CAPÍTULO 2

## Trabajo con componentes

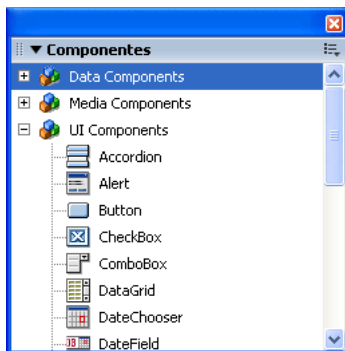
Existen varias formas de trabajar con componentes en Macromedia Flash MX 2004 y Macromedia Flash MX Professional 2004. El panel Componentes sirve para ver componentes y añadirlos a un documento durante su edición. Una vez añadido el componente al documento, las propiedades de éste pueden verse en el inspector de propiedades o en el panel Inspector de componentes. Los componentes pueden comunicarse entre sí escuchando sus eventos y gestionándolos mediante ActionScript. También puede gestionar la profundidad del componente en un documento y controlar cuándo se selecciona el componente.

### Panel Componentes

En el panel Componentes se almacenan todos los componentes. Cuando se instala Flash MX 2004 o Flash MX Professional 2004 y se inicia por primera vez, los componentes de la carpeta Macromedia\Flash 2004\es\First Run\Components (Windows) o Macromedia Flash 2004/es/First Run/Components (Macintosh) se muestran en el panel Componentes.

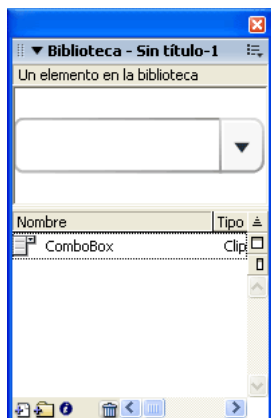
**Para mostrar el panel Componentes:**

- Seleccione Ventana > Paneles de desarrollo > Componentes.



## Componentes del panel Biblioteca

Cuando se añade un componente a un documento, el componente aparece como un símbolo de clip compilado (archivo SWC) en el panel Biblioteca.



*Componente ComboBox del panel Biblioteca.*

Puede agregar más instancias de un componente arrastrando el icono del componente desde la biblioteca al escenario.

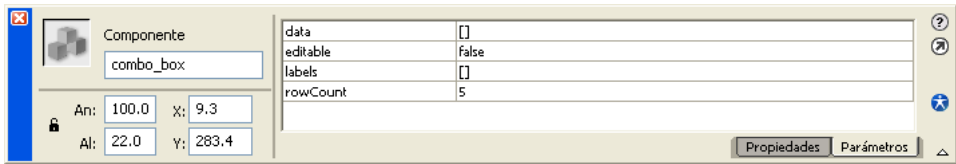
Para más información sobre clips compilados, consulte [“Trabajo con archivos SWC y clips compilados” en la página 20](#).

## Componentes del panel Inspector de componentes y del inspector de propiedades

Una vez que haya añadido una instancia de un componente a un documento de Flash, utilice el inspector de propiedades para definir y ver información de la instancia. Para crear una instancia de un componente, arrástrelo del panel Componentes al escenario; asigne un nombre a la instancia en el inspector de propiedades y, a continuación, especifique los parámetros de la instancia mediante los campos de la ficha Parámetros. También puede definir parámetros para una instancia de componente mediante el panel Inspector de componentes. No importa qué panel utiliza para definir los parámetros; es sólo cuestión de preferencias personales. Para más información sobre cómo definir parámetros, consulte [“Definición de parámetros de componentes” en la página 24](#).

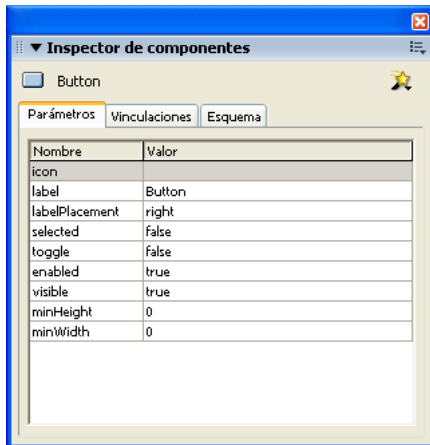
**Para ver información sobre una instancia de componente en el inspector de propiedades:**

- 1 Seleccione Ventana > Propiedades.
- 2 Seleccione una instancia de componente en el escenario.
- 3 Para ver los parámetros, haga clic en la ficha Parámetros.



**Para ver los parámetros de una instancia de componente en el panel Inspector de componentes:**

- 1 Seleccione Ventana > Paneles de desarrollo > Inspector de componentes.
- 2 Seleccione una instancia de componente en el escenario.
- 3 Para ver los parámetros, haga clic en la ficha Parámetros.

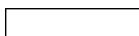


## Componentes en vista previa dinámica

La función Vista previa dinámica, que se activa de forma predeterminada, permite ver los componentes del escenario tal como aparecerán en el contenido de Flash publicado, incluido el tamaño aproximado. La vista previa dinámica refleja parámetros diferentes para componentes diferentes. Para obtener información sobre qué parámetros de componente se reflejan en la vista previa dinámica, consulte la entrada de cada componente en el [Capítulo 4, “Diccionario de componentes”, en la página 49](#). Los componentes en vista previa dinámica no son funcionales. Para probar la funcionalidad de los componentes, utilice el comando Control > Probar película.



*Componente Button con la vista previa dinámica activada.*



*Componente Button con la vista previa dinámica desactivada.*

**Para activar o desactivar la función Vista previa dinámica:**

- Seleccione Control > Activar vista previa dinámica. Si aparece una marca de verificación junto a la opción, ésta está activada.

Para más información, consulte el [Capítulo 5, “Creación de componentes”, en la página 683](#).

## Trabajo con archivos SWC y clips compilados

Los componentes que se incluyen con Flash MX 2004 o Flash MX Professional 2004 no son archivos FLA, sino archivos SWC. SWC es el formato de archivo de Macromedia para componentes. Si añade un componente al escenario desde el panel Componentes, se añadirá un símbolo de clip compilado a la biblioteca. Un SWC es un clip compilado que se ha exportado para su distribución.

Un clip de película también se puede “compilar” en Flash y convertir en un símbolo de clip compilado. El símbolo de clip compilado se comporta como el símbolo de clip de película desde el que se compiló, aunque los clips compilados se muestran y se publican más rápido que los símbolos de clip de película normales. Los clips compilados no se pueden editar, aunque tienen propiedades que aparecen en el inspector de propiedades y en el panel Inspector de componentes e incluyen una vista previa dinámica.

Los componentes que se incluyen con Flash MX 2004 o Flash MX Professional 2004 ya están convertidos en clips compilados. Si crea un componente, puede exportarlo como SWC para su distribución. Para más información, consulte el [Capítulo 5, “Creación de componentes”, en la página 683](#).

**Para compilar un símbolo de clip de película:**

- Seleccione el clip de película en la biblioteca, haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) y seleccione Convertir en clip compilado.



### Para exportar un archivo SWC:

- Seleccione el clip de película en la biblioteca, haga clic con el botón derecho del ratón (Windows) o con la tecla Control presionada (Macintosh) y seleccione Exportar archivo SWC.

**Nota:** Flash MX 2004 y Flash MX Professional 2004 siguen admitiendo los componentes FLA.

## Adición de componentes a documentos de Flash

Si arrastra un componente desde el panel Componentes al escenario, se añadirá un símbolo de clip compilado en el panel Biblioteca. Una vez que el símbolo de clip compilado está en la biblioteca, también puede añadir el componente a un documento en tiempo de ejecución mediante el método `UIObject.createClassObject()` de ActionScript.

- Los usuarios que empiezan a utilizar Flash pueden usar el panel Componentes para añadir componentes a documentos de Flash, especificar parámetros básicos mediante el inspector de propiedades o la ficha Parámetros del panel Inspector de componentes y utilizar el controlador de eventos `on()` para controlar los componentes.
- Los usuarios de nivel intermedio pueden utilizar el panel Componentes para añadir componentes a documentos de Flash y, a continuación, utilizar el inspector de propiedades, los métodos ActionScript o ambos para especificar parámetros. También pueden utilizar el controlador de eventos `on()` o los detectores de eventos para gestionar los eventos del componente.
- Los programadores de Flash de nivel avanzado pueden utilizar una combinación del panel Componentes y de ActionScript para añadir componentes y especificar propiedades, o bien pueden implementar instancias de componente en tiempo de ejecución usando sólo ActionScript. También pueden utilizar detectores de eventos para controlar los componentes.

Si edita los aspectos de un componente y, a continuación, añade otra versión del componente o un componente que comparte los mismos aspectos, puede utilizar los aspectos editados o reemplazarlos por un nuevo conjunto de aspectos predeterminados. Si reemplaza los aspectos editados, todos los componentes que utilizan estos aspectos se actualizan con versiones predeterminadas de los aspectos. Para más información sobre cómo editar los aspectos, consulte el [Capítulo 3, “Personalización de componentes”, en la página 31](#).

## Adición de componentes mediante el panel Componentes

Una vez que haya añadido un componente a un documento mediante el panel Componentes, puede añadir instancias adicionales del componente al documento arrastrando el componente desde el panel Biblioteca al escenario. Puede definir propiedades para instancias adicionales en la ficha Parámetros del inspector de propiedades o en la ficha Parámetros del panel Inspector de componentes.

### Para añadir un componente a un documento de Flash mediante el panel Componentes:

- 1 Seleccione Ventana > Paneles de desarrollo > Componentes.
- 2 Siga uno de estos procedimientos:
  - Arrastre un componente del panel Componentes al escenario.
  - Haga doble clic en un componente en el panel Componentes.
- 3 Si el componente es un FLA (todos los componentes instalados de la v2 son SWC) y si ha editado aspectos para otra instancia del mismo componente, o para un componente que comparte aspectos con el componente que se está añadiendo, siga uno de estos procedimientos:
  - Seleccione No reemplazar elementos ya existentes para mantener los aspectos editados y aplicarlos al nuevo componente.
  - Seleccione Reemplazar elementos ya existentes para reemplazar todos los aspectos por aspectos predeterminados. El nuevo componente y todas sus versiones anteriores, o las de los componentes que comparten sus aspectos, utilizarán los aspectos predeterminados.
- 4 Seleccione el componente en el escenario.
- 5 Seleccione Ventana > Propiedades.
- 6 En el inspector de propiedades, introduzca un nombre para la instancia de componente.
- 7 Haga clic en la ficha Parámetros y especifique parámetros para la instancia.

Para más información, consulte [“Definición de parámetros de componentes” en la página 24](#).
- 8 Cambie el tamaño del componente como desee.

Para más información sobre cómo cambiar el tamaño de tipos de componente específicos, consulte las entradas referentes al componente en cuestión en el [Capítulo 4, “Diccionario de componentes”, en la página 49](#).
- 9 Cambie el color y el formato de texto de un componente como desee mediante uno o varios de estos procedimientos:
  - Defina o cambie un valor de propiedad de estilo específico para una instancia de componente mediante el método `setStyle()` disponible para todos los componentes. Para más información, consulte [UIObject.setStyle\(\)](#).
  - Edite varias propiedades en la declaración de estilo `_global` asignada a todos los componentes de la v2.
  - Si lo desea, cree una declaración de estilo personalizada para instancias de componente específicas.

Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).
- 10 Personalice la apariencia del componente como desee mediante uno de estos procedimientos:
  - Aplique un tema (véase [“Temas” en la página 39](#)).
  - Edite los aspectos de un componente (véase [“Aplicación de aspectos a los componentes” en la página 41](#)).

## Adición de componentes mediante ActionScript

Para añadir un componente a un documento mediante ActionScript, primero debe añadirlo a la biblioteca.

Puede utilizar métodos ActionScript para definir otros parámetros para componentes añadidos dinámicamente. Para más información, consulte el [Capítulo 4, “Diccionario de componentes”](#), en la [página 49](#).

**Nota:** en las instrucciones de esta sección se presupone que el usuario posee un conocimiento intermedio o avanzado de ActionScript.

### Para añadir un componente a un documento de Flash mediante ActionScript:

- 1 Arrastre un componente del panel Componentes al escenario y elimínelo.  
De esta manera, añadirá el componente a la biblioteca.
- 2 Seleccione el fotograma en la línea de tiempo donde desea colocar el componente.
- 3 Abra el panel Acciones si aún no está abierto.
- 4 Llame al método `createClassObject()` para crear la instancia de componente en tiempo de ejecución.

Este método puede llamarse por sí solo, o bien a partir de una instancia de componente. Como parámetros tiene un nombre de clase de componente, un nombre de instancia para la instancia nueva, una profundidad y un objeto de inicialización opcional. En el parámetro `className`, puede especificar el paquete de clase tal como se indica a continuación:

```
createClassObject(mx.controls.CheckBox, "cb", 5, {label:"Check Me"});
```

O puede importar el paquete de clase tal como se indica a continuación:

```
import mx.controls.CheckBox;  
createClassObject(CheckBox, "cb", 5, {label:"Check Me"});
```

Para más información, consulte [UIObject.createClassObject\(\)](#).

- 5 Utilice las propiedades y métodos ActionScript del componente para especificar opciones adicionales o sustituir parámetros establecidos durante la edición.

Para obtener información detallada sobre las propiedades y métodos ActionScript disponibles para cada componente, consulte sus respectivas entradas en el [Capítulo 4, “Diccionario de componentes”](#), en la [página 49](#).

## Acerca del tamaño de la etiqueta del componente y la anchura y altura del componente

Si una instancia de componente que se ha añadido a un documento no es lo suficientemente grande para mostrar la etiqueta, el texto de la etiqueta aparece recortado. Si una instancia de componente es más grande que el texto, la zona activa sobrepasa la etiqueta.

Utilice la herramienta Transformación libre o el método `setSize()` para cambiar el tamaño de las instancias de componente. Puede llamar al método `setSize()` desde cualquiera de las instancias de componente (véase [UIObject.setSize\(\)](#)). Si utiliza las propiedades de ActionScript `_width` y `_height` para ajustar la anchura y altura de un componente, se cambia el tamaño del componente pero el diseño del contenido no varía. Esto puede provocar la distorsión del componente al reproducir la película. Para más información sobre el tamaño de los componentes, consulte sus respectivas entradas en el [Capítulo 4, “Diccionario de componentes”](#), en la [página 49](#).

## Definición de parámetros de componentes

Todos los componentes tienen parámetros que se pueden definir para cambiar la apariencia y el comportamiento. Un parámetro es una propiedad o método que aparece en el inspector de propiedades o en el panel Inspector de componentes. Las propiedades y métodos que se utilizan con más frecuencia aparecen como parámetros de edición; los demás deben definirse mediante ActionScript. Todos los parámetros que se pueden definir editando, también se pueden definir mediante ActionScript. Los valores de los parámetros definidos mediante ActionScript sustituyen cualquier otro valor que se haya definido durante la edición.

Todos los componentes de la v2 heredan las propiedades y los métodos de la clase `UIObject` y la clase `UIComponent`; se trata de propiedades y métodos que utilizan todos los componentes, como por ejemplo, `UIObject.setSize()`, `UIObject.setStyle()`, `UIObject.x` y `UIObject.y`. Asimismo, cada componente tiene propiedades y métodos exclusivos, algunos de los cuales están disponibles como parámetros de edición. Por ejemplo, el componente `ProgressBar` tiene una propiedad `percentComplete` (`ProgressBar.percentComplete`), mientras que el componente `NumericStepper` dispone de las propiedades `nextValue` y `previousValue` (`NumericStepper.nextValue`, `NumericStepper.previousValue`).

## Eliminación de componentes de documentos de Flash

Para eliminar instancias de un componente de un documento de Flash, debe borrar el componente de la biblioteca eliminando el icono del clip compilado.

**Para eliminar un componente de un documento:**

- 1 En el panel Biblioteca, seleccione el símbolo de clip compilado (SWC).
- 2 Haga clic en el botón Eliminar situado en la parte inferior del panel Biblioteca, o seleccione Eliminar en el menú de opciones del panel Biblioteca.
- 3 En el cuadro de diálogo Eliminar, haga clic en Eliminar para confirmar la operación.

## Utilización de las sugerencias para el código

Cuando utiliza ActionScript2, puede escribir estrictamente una variable basada en una clase incorporada, incluidas las clases de componente. Si lo hace así, el editor ActionScript muestra sugerencias para el código de la variable. Por ejemplo, supongamos que escribe lo siguiente:

```
import mx.controls.CheckBox;
var myCheckBox:CheckBox;
myCheckBox.
```

En cuanto escriba el punto, Flash muestra una lista de métodos y propiedades disponibles para los componentes `CheckBox`, porque la variable se ha escrito como `CheckBox`. Para más información sobre la escritura de datos, consulte “Strict data typing” en el apartado Guía de referencia de ActionScript de la Ayuda. Para obtener información sobre cómo utilizar las sugerencias para el código cuando aparecen, consulte “Utilización de las sugerencias para el código” en el apartado Guía de referencia de ActionScript de la Ayuda.

## Eventos de componentes

Todos los componentes tienen eventos que se difunden cuando el usuario interactúa con un componente o cuando ocurre algo significativo al componente. Para gestionar un evento, escriba código ActionScript que se ejecute cuando se activa el evento.

Los eventos de componentes pueden gestionarse de las siguientes maneras:

- Utilice el controlador de eventos de componentes `on()`.
- Utilice detectores de eventos.

### Utilización del controlador de eventos `on()`

La forma más fácil de gestionar un evento de un componente es mediante el controlador de eventos de componentes `on()`. El controlador de eventos `on()` puede asignarse a una instancia de componente igual que se asigna un controlador a un botón o a un clip de película.

Cuando se utiliza un controlador de eventos `on()`, se genera automáticamente un objeto de evento, `eventObj`, cuando el evento se desencadena y se pasa al controlador. El objeto `Event` tiene propiedades que contienen información sobre el evento. El objeto `Event` que se pasa al controlador `on()` siempre es `eventObj`. Para más información, consulte [“UIEventDispatcher” en la página 600](#).

La palabra clave `this`, utilizada dentro de un controlador `on()` asociado con un componente, se refiere a la instancia de componente. Por ejemplo, el código siguiente, asociado a la instancia del componente `Button myButtonComponent`, envía `“_level0.myButtonComponent”` al panel Salida:

```
on(click){
    trace(this);
}
```

#### Para utilizar el controlador de eventos `on()`:

- 1 Arrastre un componente `CheckBox` desde el panel Componentes al escenario.
- 2 Seleccione el componente y elija `Ventana > Acciones`.
- 3 En este panel, introduzca los códigos siguientes:

```
on(click){
    trace("The " + eventObj.type + " event was broadcast");
}
```

Puede introducir cualquier código que desee entre las llaves `{ }`.

- 4 Seleccione `Control > Probar película` y haga clic en la casilla de verificación para ver el seguimiento en el panel Salida.

Para más información, consulte la entrada correspondiente a cada evento en el [Capítulo 4, “Diccionario de componentes”](#), en la página 49.

## Utilización de detectores de eventos de componente

La forma más eficaz de gestionar eventos de componente consiste en utilizar detectores. Los eventos se difunden mediante componentes y cualquier objeto que esté registrado como detector en el difusor de eventos (instancia de componente) puede recibir notificación del evento. Al detector se le asigna una función que gestiona el evento. Se pueden registrar varios detectores en una instancia de componente. Asimismo, se puede registrar un detector en varias instancias de componente.

Para utilizar el modelo de detector de eventos, debe crear un objeto detector con una propiedad que sea el nombre del evento. La propiedad se asigna a una función callback. A continuación, llama al método `UIEventDispatcher.addEventListener()` en la instancia de componente que difunde el evento y le pasa el nombre del evento y el del objeto detector. Llamar al método `UIEventDispatcher.addEventListener()` también recibe el nombre de “registrarse” o “suscribirse” a un detector, como se muestra a continuación:

```
listenerObject.eventName = function(evtObj){
    // aquí código propio
};
componentInstance.addEventListener("eventName", listenerObject);
```

En el código anterior, la palabra clave `this`, si se utiliza en la función callback, tiene su ámbito en `listenerObject`.

El parámetro `evtObj` es un objeto `Event` que se genera automáticamente cuando se activa un evento y se pasa a la función callback del objeto detector. El objeto `Event` tiene propiedades que contienen información sobre el evento. Para más información, consulte [“UIEventDispatcher” en la página 600](#).

Para información sobre los eventos como difusiones de componentes, consulte la entrada correspondiente de cada componente en el [Capítulo 4, “Diccionario de componentes”, en la página 49](#).

**Para registrar un detector de eventos, utilice el procedimiento siguiente:**

- 1 Arrastre un componente `Button` desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia **button**.
- 3 Arrastre un componente `TextInput` desde el panel Componentes al escenario.
- 4 En el inspector de propiedades, introduzca el nombre de instancia **myText**.
- 5 Seleccione el fotograma 1 de la línea de tiempo.
- 6 Seleccione `Ventana > Acciones`.
- 7 En este panel, introduzca los códigos siguientes:

```
form = new Object();
form.click = function(evt){
    myText.text = evt.target;
}
button.addEventListener("click", form);
```

La propiedad `target` del objeto `Event` es una referencia a la instancia que difunde el evento. Este código muestra el valor de la propiedad `target` en el campo de entrada de texto.

## Sintaxis adicional de eventos

Además de utilizar un objeto detector, también puede utilizar una función como detector. Un detector es una función si no pertenece a ningún objeto. Por ejemplo, el código siguiente crea la función de detección `myHandler` y la registra en `buttonInstance`:

```
function myHandler(eventObj){
    if (eventObj.type == "click"){
        // el código se escribe aquí
    }
}
buttonInstance.addEventListener("click", myHandler);
```

**Nota:** en un detector de función, la palabra clave `this` es `buttonInstance`, no la línea de tiempo en la que se define la función.

También puede utilizar objetos detectores que admitan el método `handleEvent`. Con independencia del nombre del evento, se llama al método `handleEvent` del objeto detector. Para gestionar varios eventos, es preciso utilizar una sentencia `if else` o `switch`, por lo que la sintaxis es complicada. Por ejemplo, el código siguiente utiliza una sentencia `if else` para gestionar los eventos `click` y `enter`:

```
myObj.handleEvent = function(o){
    if (o.type == "click"){
        // el código se escribe aquí
    } else if (o.type == "enter"){
        // el código se escribe aquí
    }
}
target.addEventListener("click", myObj);
target2.addEventListener("enter", myObj);
```

Existe otro estilo de sintaxis de evento que sólo debe utilizarse cuando se está editando un componente y se sabe que un objeto determinado es el único detector de un evento. En dicho caso, puede aprovechar que el modelo de evento de la v2 siempre llama a un método de la instancia de componente que es el nombre del evento más “Handler”. Por ejemplo, si desea gestionar el evento `click`, escriba el código siguiente:

```
componentInstance.clickHandler = function(o){
    // introduzca aquí el código propio
}
```

En el código anterior, la palabra clave `this`, si se utiliza en la función callback, tiene el ámbito de `componentInstance`.

Para más información, consulte el [Capítulo 5, “Creación de componentes”, en la página 683](#).

## Creación de un desplazamiento personalizado de la selección

Cuando un usuario presiona la tecla Tabulador para navegar por una aplicación de Flash o hace clic en una aplicación, la [Clase FocusManager](#) determina qué componente se selecciona. No es preciso añadir una instancia de `FocusManager` a una aplicación, ni escribir código para activarlo.

Si un objeto `RadioButton` se selecciona, `FocusManager` examina el objeto, así como todos los objetos con el mismo valor `groupName`, y define la selección del objeto estableciendo el valor `true` en la propiedad `selected`.

Cada componente `Window` modal contiene una instancia de `FocusManager` para que los controles de dicha ventana se conviertan en su propio conjunto de tabuladores, lo que evita que el usuario acceda sin darse cuenta a los componentes de otras ventanas al presionar la tecla `Tabulador`.

Para que la selección pueda desplazarse por la aplicación, defina la propiedad `tabIndex` en los componentes que deban seleccionarse (botones incluidos). Cuando un usuario presiona la tecla `Tabulador`, la [Clase `FocusManager`](#) busca un objeto activado cuya propiedad `tabIndex` sea superior al valor actual de `tabIndex`. Una vez que la [Clase `FocusManager`](#) llegue a la propiedad `tabIndex` más alta, vuelve a cero. Por ejemplo, en este caso, el objeto `comment` (probablemente un componente `TextArea`) es el primero en seleccionarse y después se selecciona el objeto `okButton`:

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
```

También se puede utilizar el panel `Accesibilidad` para asignar un valor de índice de tabulación.

Si ningún elemento del escenario tiene un valor de índice de tabulación, `FocusManager` utiliza el orden `z`. El orden `z` se configura principalmente mediante el orden en el que los componentes se arrastran al escenario. Sin embargo, también se pueden utilizar los comandos `Modificar/Organizar/Traer al frente/Atrás` para determinar el orden `z` final.

Para seleccionar un componente de una aplicación, llame al método `FocusManager.setFocus()`.

Para crear un botón que se seleccione cuando el usuario presione la tecla `Intro` (Windows) o `Retorno` (Macintosh), defina la propiedad `FocusManager.defaultPushButton` en el nombre de instancia del botón deseado, como en el ejemplo siguiente:

```
FocusManager.defaultPushButton = okButton;
```

La [Clase `FocusManager`](#) sustituye el rectángulo de selección de `Flash Player` predeterminado y dibuja un rectángulo de selección personalizado con esquinas redondeadas.

## Administración de la profundidad del componente en un documento

Si desea colocar un componente por encima o por debajo de otro objeto de la aplicación, deberá utilizar la [Clase `DepthManager`](#). La interfaz de programación de aplicaciones (API) `DepthManager` permite colocar los componentes de la interfaz de usuario en un orden `z` apropiado (por ejemplo, un cuadro combinado se despliega delante de otros componentes, los puntos de inserción aparecen delante del resto de los elementos, las ventanas de diálogo aparecen sobre el contenido, etc.).

`DepthManager` tiene dos objetivos principales: administrar las asignaciones de profundidad relativa en cualquier documento y administrar las profundidades reservadas en la línea de tiempo raíz para servicios del sistema como, por ejemplo, el cursor y las sugerencias.

Para utilizar `DepthManager`, llame a sus métodos (véase [“Clase `DepthManager`” en la página 286](#)).

El código siguiente coloca la instancia del componente `loader` bajo el componente `button`:

```
loader.setDepthBelow(button);
```



## Utilización de preloader con componentes

Los componentes están definidos de forma predeterminada en Exportar en primer fotograma. Esto hace que los componentes se carguen antes de que aparezca el primer fotograma de una aplicación. Si desea crear un preloader para una aplicación, debe anular la selección de Exportar en primer fotograma para cualquier símbolo de clip compilado de la biblioteca.

**Nota:** si está utilizando el componente ProgressBar para mostrar el progreso de carga, deje seleccionado Exportar en primer fotograma para ProgressBar.

## Actualización de componentes de la versión 1 a la arquitectura de la versión 2

Los componentes de la v2 se escribieron de modo que cumplieran varios estándares de Web (relacionados con [eventos](#), estilos, políticas captador/definidor, etc.) y son muy distintos de sus equivalentes de la v1 publicados con Macromedia Flash MX y en los DRK publicados antes de Macromedia Flash MX 2004. Los componentes de la v2 tienen API diferentes y se han escrito con ActionScript 2.0. Por lo tanto, utilizar los componentes de la v1 y la v2 juntos en una misma aplicación puede provocar un comportamiento imprevisible. Para obtener información sobre cómo actualizar los componentes de la v1 para utilizar la gestión de eventos, los estilos y el acceso captador/definidor a las propiedades en lugar de métodos de la versión 2, consulte el [Capítulo 5, “Creación de componentes”](#), en la [página 683](#).

Las aplicaciones de Flash que contienen componentes de la v1 funcionan correctamente en Flash Player 6 y Flash Player 7 cuando se publican para Flash Player 6 o Flash Player 6 versión 65. Si desea actualizar las aplicaciones para que funcionen cuando se publiquen para Flash Player 7, debe convertir el código para que utilice strict data-typing. Para más información, consulte “Creación de clases con ActionScript 2.0” en el apartado Guía de referencia de ActionScript de la Ayuda.



# CAPÍTULO 3

## Personalización de componentes

Puede cambiar la apariencia de los componentes cuando los utiliza en diferentes aplicaciones. Para ello, Macromedia Flash MX 2004 y Macromedia Flash MX Professional 2004 le ofrecen tres formas distintas:

- Utilizar la interfaz API de estilos.
- Aplicar un tema.
- Modificar o sustituir los aspectos de un componente.

La interfaz API (interfaz de programación de aplicaciones) de estilos dispone de métodos y propiedades que permiten cambiar el color y el formato de texto de un componente.

Un tema es un conjunto de estilos y aspectos que forman la apariencia de un componente.

Los aspectos son símbolos que se utilizan para visualizar componentes. La *aplicación de aspectos* es el proceso mediante el cual se cambia la apariencia de un componente, modificando o sustituyendo el gráfico de origen. Un aspecto puede ser una parte pequeña, como un borde o esquina de un borde, o bien una parte compuesta como la imagen completa de un botón sin oprimir (estado en el que se encuentra cuando no se ha presionado). Asimismo, puede ser un símbolo sin gráfico que contiene código que dibuja una parte del componente.

### Utilización de estilos para personalizar el texto y el color de un componente

Todas las instancias de componente tienen propiedades de estilo y un método `setStyle()` y `getStyle()` (véase `UIObject.setStyle()` y `UIObject.getStyle()`) que se pueden utilizar para modificar las propiedades de estilo y tener acceso a las mismas. Puede utilizar los estilos para personalizar un componente de varias formas:

- Defina los estilos en una instancia de componente.  
Puede cambiar las propiedades de texto y de color de una única instancia de componente. Este recurso es efectivo en determinadas situaciones, aunque puede tardar mucho tiempo si necesita definir propiedades individuales en todos los componentes de un documento.
- Utilice la declaración de estilo `_global` que define los estilos de todos los componentes de un documento.

Si desea que el documento completo tenga un aspecto uniforme, puede crear estilos en la declaración de estilo `_global`.

- Cree declaraciones de estilo personalizadas y aplíquelas a instancias de componente específicas. También puede hacer que grupos de componentes de un documento compartan un mismo estilo. Para ello, puede crear declaraciones de estilo personalizadas para aplicarlas a componentes específicos.
- Cree declaraciones de estilo de clase predeterminadas. También puede definir una declaración de estilo de clase predeterminada para que todas las instancias de una misma clase compartan una apariencia predeterminada.

Los cambios efectuados en las propiedades de estilo no aparecen al ver los componentes en el escenario mediante la función Vista previa dinámica. Para más información, consulte [“Componentes en vista previa dinámica” en la página 20](#).

## Definición de los estilos en una instancia de un componente

Puede escribir código ActionScript para definir y obtener las propiedades de estilo de cualquier instancia de componente. Los métodos `UIObject.setStyle()` y `UIObject.getStyle()` pueden llamarse directamente desde cualquier componente. Por ejemplo, el código siguiente define el color de texto de una instancia de botón denominada `myButton`:

```
myButton.setStyle("color", "0xFF00FF");
```

Aunque puede tener acceso directo a los estilos como propiedades (por ejemplo, `myButton.color = 0xFF00FF`), es preferible utilizar los métodos `setStyle()` y `getStyle()` para que los estilos funcionen correctamente. Para más información, consulte [“Definición de los valores de la propiedad de estilo” en la página 37](#).

**Nota:** para definir más de una propiedad, no llame varias veces al método `UIObject.setStyle()`. Si desea cambiar varias propiedades o cambiar propiedades de varias instancias de componente, debe crear un formato de estilo personalizado. Para más información, consulte [“Definición de los estilos de componentes específicos” en la página 33](#).

### Para definir o cambiar una propiedad de una sola instancia de componente:

- 1 Seleccione la instancia de componente en el escenario.
- 2 En el inspector de propiedades, asígnele el nombre de instancia **myComp**.
- 3 Abra el panel Acciones y seleccione Escena 1 y, a continuación, Capa 1: Fotograma 1.
- 4 Introduzca el código siguiente para cambiar el color de la instancia a azul:
 

```
myComp.setStyle("themeColor", "haloBlue");
```

La sintaxis siguiente especifica una propiedad y un valor para una instancia de componente:

```
instanceName.setStyle("property", value);
```
- 5 Seleccione Control > Probar película para ver los cambios.
 

Para obtener una lista de los estilos admitidos, consulte [“Estilos admitidos” en la página 38](#).

## Definición de estilos globales

La declaración de estilo `_global` se asigna a todos los componentes de Flash incorporados en la versión 2 de la arquitectura de componentes de Macromedia (componentes de la v2). El objeto `_global` tiene una propiedad llamada `style` (`_global.style`) que es una instancia de `CSSStyleDeclaration`. La propiedad `style` actúa como declaración de estilo `_global`. Si cambia el valor de una propiedad en la declaración de estilo `_global`, el cambio se aplica a todos los componentes del documento de Flash.

Algunos estilos se definen en la `CSSStyleDeclaration` de la clase de componente (por ejemplo, el estilo `backgroundColor` de los componentes `TextArea` y `TextInput`). La declaración de estilo de clase tiene prioridad sobre la declaración de estilo `_global` a la hora de determinar los valores de estilo, por lo que la definición de `backgroundColor` en la declaración de estilo `_global` no tendría efecto alguno sobre `TextArea` o `TextInput`. Para más información, consulte [“Utilización de los estilos global, personalizado y de clase en el mismo documento” en la página 35](#).

### Para cambiar una o más propiedades en la declaración de estilo global:

- 1 Asegúrese de que el documento contiene como mínimo una instancia de componente.  
Para más información, consulte [“Adición de componentes a documentos de Flash” en la página 21](#).
- 2 Cree una nueva capa en la línea de tiempo y asígnele un nombre.
- 3 Seleccione un fotograma en la capa nueva donde aparece el componente (o antes de que aparezca).
- 4 Abra el panel Acciones.
- 5 Utilice la sintaxis siguiente para cambiar las propiedades de la declaración de estilo `_global`. Sólo es necesario enumerar las propiedades cuyos valores desea cambiar, como se muestra a continuación:  

```
_global.style.setStyle("color", 0xCC6699);  
_global.style.setStyle("themeColor", "haloBlue")  
_global.style.setStyle("fontSize", 16);  
_global.style.setStyle("fontFamily" , "_serif");
```

  
Para obtener una lista de estilos, consulte [“Estilos admitidos” en la página 38](#).
- 6 Seleccione Control > Probar película para ver los cambios.

## Definición de los estilos de componentes específicos

Puede crear declaraciones de estilo personalizadas para especificar un conjunto exclusivo de propiedades para componentes específicos de un documento de Flash. Cree una instancia nueva del objeto `CSSStyleDeclaration`, cree un nombre de estilo personalizado y póngalo en la lista `_global.styles` (`_global.styles.newStyle`), especifique las propiedades y valores del estilo y asígnelo a una instancia. Se puede acceder al objeto `CSSStyleDeclaration` si se ha colocado como mínimo una instancia de componente en el escenario.

Para modificar un formato de estilo personalizado se sigue el mismo procedimiento que para editar las propiedades de la declaración de estilo `_global`. En lugar de utilizar el nombre de la declaración de estilo `_global`, utilice la instancia `CSSStyleDeclaration`. Para más información sobre la declaración de estilo `_global`, consulte [“Definición de estilos globales” en la página 33](#).

Para obtener información sobre las propiedades del objeto `CSSStyleDeclaration`, consulte [“Estilos admitidos” en la página 38](#). Para obtener una lista de los estilos que los componentes admiten, consulte sus entradas individuales en el [Capítulo 4, “Diccionario de componentes”](#), en la [página 49](#).

**Para crear una declaración de estilo personalizada para componentes específicos:**

- 1 Asegúrese de que el documento contiene como mínimo una instancia de componente.

Para más información, consulte [“Adición de componentes a documentos de Flash” en la página 21](#).

En este ejemplo se utilizan tres componentes de botón con los nombres de instancia `a`, `b` y `c`. Si utiliza otros componentes, asígneles nombres de instancia en el inspector de propiedades y utilice dichos nombres en el paso 9.

- 2 Cree una nueva capa en la línea de tiempo y asígnele un nombre.
- 3 Seleccione un fotograma en la capa nueva donde aparece el componente (o antes de que aparezca).
- 4 Abra el panel Acciones en modo Experto.
- 5 Utilice la sintaxis siguiente para crear una instancia del objeto `CSSStyleDeclaration` con el fin de definir el nuevo estilo de formato personalizado:

```
var styleObj = new mx.styles.CSSStyleDeclaration;
```

- 6 Defina la propiedad `styleName` de la declaración de estilo para asignar un nombre al estilo:

```
styleObj.styleName = "newStyle";
```

- 7 Ponga el estilo en la lista global de estilos:

```
_global.styles.newStyle = styleObj;
```

**Nota:** también puede crear un objeto `CSSStyleDeclaration` y asignarlo a una declaración de estilo nueva mediante la sintaxis siguiente:

```
var styleObj = _global.styles.newStyle = new  
mx.styles.CSSStyleDeclaration();
```

- 8 Utilice la sintaxis siguiente para especificar las propiedades que desea definir para la declaración de estilo `myStyle`:

```
styleObj.fontFamily = "_sans";  
styleObj.fontSize = 14;  
styleObj.fontWeight = "bold";  
styleObj.textDecoration = "underline";  
styleObj.color = 0x336699;  
styleObj.setStyle("themeColor", "haloBlue");
```

- 9 En el mismo panel Script, utilice la sintaxis siguiente para definir la propiedad `styleName` de dos componentes específicos en la declaración de estilo personalizada:

```
a.setStyle("styleName", "newStyle");  
b.setStyle("styleName", "newStyle");
```

También puede tener acceso a los estilos de una declaración de estilo personalizada mediante los métodos `setStyle()` y `getStyle()`. El código siguiente define el estilo `backgroundColor` en la declaración de estilo `newStyle`:

```
_global.styles.newStyle.setStyle("backgroundColor", "0xFFCCFF");
```

## Definición de los estilos de una clase de componente

Puede definir una declaración de estilo de clase para cualquier clase de componente (Button, CheckBox, etc.) que defina los estilos predeterminados de cada instancia de dicha clase. Antes de crear las instancias, es preciso crear la declaración de estilo. Algunos componentes, como TextArea y TextInput, tienen declaraciones de estilo de clase predefinidas de forma predeterminada, ya que las propiedades `borderStyle` y `backgroundColor` deben personalizarse.

El código siguiente crea una declaración de estilo de clase para CheckBox y define el color azul para la casilla de verificación:

```
var o = _global.styles.CheckBox = new mx.styles.CSSStyleDeclaration();  
o.color = 0x0000FF;
```

También puede tener acceso a los estilos de una declaración de estilo de clase mediante los métodos `setStyle()` y `getStyle()`. El código siguiente define el estilo de color en la declaración de estilo RadioButton:

```
_global.styles.RadioButton.setStyle("color", "blue");
```

Para más información sobre los estilos admitidos, consulte [“Estilos admitidos” en la página 38](#).

## Utilización de los estilos global, personalizado y de clase en el mismo documento

Si define un estilo sólo en un sitio del documento, Flash utilizará dicha definición cuando necesite saber el valor de una propiedad. No obstante, un único documento de Flash puede tener una declaración de estilo `_global`, declaraciones de estilo personalizadas, propiedades de estilo definidas directamente en instancias de componente y declaraciones de estilo de clase predeterminadas. En dicha situación, Flash determina el valor de una propiedad buscando su definición en todos los lugares mencionados, siguiendo un orden específico.

En primer lugar, Flash buscará una propiedad de estilo en la instancia de componente. Si el estilo no está directamente definido en la instancia, Flash buscará la propiedad `styleName` de la instancia para ver si tiene asignada una declaración de estilo.

Si no se ha asignado la propiedad `styleName` a ninguna declaración de estilo, Flash buscará la propiedad en la declaración de estilo de clase predeterminada. Si no hay declaración de estilo de clase y la propiedad no hereda su valor, se activará la declaración de estilo `_global`. Si la propiedad no está definida en la declaración de estilo `_global`, la propiedad es `undefined`.

Si no hay declaración de estilo de clase y la propiedad hereda su valor, Flash buscará la propiedad en la instancia principal. Si la propiedad no está definida en la instancia principal, Flash comprobará la propiedad `styleName` de la instancia principal; si no está definida, Flash seguirá buscando en instancias principales hasta que llegue al nivel `_global`. Si la propiedad no está definida en la declaración de estilo `_global`, la propiedad es `undefined`.

StyleManager indica a Flash si un estilo hereda su valor o no. Para más información, consulte [“Clase StyleManager” en la página 540](#).

**Nota:** no se admite el valor `inherit` de CSS.

## Propiedades del estilo del color

Las propiedades del estilo de color tienen un comportamiento diferente de las propiedades que no son de color. Todas las propiedades tienen un nombre que termina por “Color”; por ejemplo, `backgroundColor`, `disabledColor` y `color`. Cuando se cambian las propiedades del estilo de color, inmediatamente se cambia también el color de la instancia y de todas las instancias secundarias correspondientes. Los cambios restantes en la propiedad de estilo sólo indican que el objeto se debe volver a dibujar y que los cambios se producirán a partir del fotograma siguiente.

El valor de una propiedad de estilo de color puede ser un número, una cadena o un objeto. Si se trata de un número, representa el valor RVA del color como número hexadecimal (0xRRGGBB). Si el valor es una cadena, debe ser un nombre de color.

Los nombres de color son cadenas que se asignan a los colores que se utilizan con mayor frecuencia. Se pueden añadir nombres de color nuevos mediante `StyleManager` (véase [“Clase StyleManager” en la página 540](#)). En la tabla siguiente se enumeran los nombres de color predeterminados:

| Nombre del color | Valor      |
|------------------|------------|
| black            | 0x000000   |
| white            | 0xFFFFFFFF |
| red              | 0xFF0000   |
| green            | 0x00FF00   |
| blue             | 0x0000FF   |
| magenta          | 0xFF00FF   |
| yellow           | 0xFFFF00   |
| cyan             | 0x00FFFF   |

**Nota:** si el nombre del color no está definido, es posible que el componente no se dibuje correctamente.

Puede utilizar cualquier identificador de `ActionScript` válido para crear nombres de color propios (por ejemplo, `WindowText` o `ButtonText`). Utilice `StyleManager` para definir colores nuevos, como en el ejemplo siguiente:

```
mx.styles.StyleManager.registerColorName("special_blue", 0x0066ff);
```

La mayoría de los componentes no pueden gestionar un objeto como valor de la propiedad de estilo de color. No obstante, determinados componentes pueden gestionar objetos de color que representan degradados u otras combinaciones de color. Para más información, consulte la sección “Utilización de estilos” de cada componente en el [Capítulo 4, “Diccionario de componentes”, en la página 49](#).



Puede utilizar las declaraciones de estilo de clase y los nombres de color para controlar fácilmente los colores del texto y los símbolos que se muestran en la pantalla. Por ejemplo, si desea una pantalla de configuración de la visualización que tenga la apariencia de Microsoft Windows, defina nombres de color como `ButtonText` y `WindowText`, y declaraciones de estilo de clase como `Button`, `CheckBox` y `Window`. Si define las propiedades del estilo de color de las declaraciones de estilo en `ButtonText` y `WindowText`, y proporciona una interfaz de usuario para que el usuario pueda cambiar los valores de `ButtonText` y `WindowText`, podrá proporcionar los mismos esquemas de color que Microsoft Windows, Macintosh OS o cualquier otro sistema operativo.

## Definición de los valores de la propiedad de estilo

Utilice el método `UIObject.setStyle()` para definir una propiedad de estilo en una instancia de componente, en la declaración de estilo global, en una declaración de estilo personalizada o en una declaración de estilo de clase. El código siguiente define el estilo de `color` de la instancia de un botón de opción en rojo:

```
myRadioButton.setStyle("color", "red");
```

El código siguiente define el estilo de `color` de la declaración de estilo personalizada `CheckBox`:

```
_global.styles.CheckBox.setStyle("color", "white");
```

El método `UIObject.setStyle()` sabe si un estilo hereda y notifica a los secundarios de la instancia si su estilo cambia. Asimismo, notifica a la instancia del componente que debe volver a dibujarse para reflejar el estilo nuevo. Por consiguiente, debe utilizar `setStyle()` para definir o cambiar estilos. No obstante, como optimización a la hora de crear declaraciones de estilo, puede definir directamente las propiedades en un objeto. Para más información, consulte [“Definición de estilos globales” en la página 33](#), [“Definición de los estilos de componentes específicos” en la página 33](#) y [“Definición de los estilos de una clase de componente” en la página 35](#).

Utilice el método `UIObject.getStyle()` para recuperar un estilo de una instancia de componente, la declaración de estilo `_global`, una declaración de estilo personalizada o una declaración de estilo de clase. El código siguiente obtiene el valor de la propiedad de `color` y lo asigna a la variable `o`:

```
var o = myRadioButton.getStyle("color");
```

El código siguiente obtiene el valor de una propiedad de estilo definida en la declaración de estilo `_global`:

```
var r = _global.style.getValue("marginRight");
```

Si el estilo no está definido, `getStyle()` puede devolver el valor `undefined`. No obstante, `getStyle()` comprende cómo se heredan las propiedades de estilo. Por consiguiente, aunque los estilos son propiedades, debe utilizar `UIObject.getStyle()` para tener acceso a los mismos sin tener que saber si el estilo se hereda.

Para más información, consulte `UIObject.getStyle()` y `UIObject.setStyle()`.

## Estilos admitidos

Flash MX 2004 y Flash MX Professional 2004 se entregan con dos temas: *Halo* (HaloTheme fla) y *Sample* (SampleTheme fla). Cada tema admite un conjunto de estilos diferente. El tema Sample utiliza todos los estilos del mecanismo de estilos v2 y se proporciona para que pueda ver una muestra de dichos estilos en un documento. El tema Halo admite un subconjunto de los estilos del tema Sample.

La mayoría de los componentes de la v2 del estilo Sample admiten las propiedades de estilo siguientes. Para obtener información sobre qué estilos de Halo admiten los componentes individuales, consulte el [Capítulo 4, “Diccionario de componentes”, en la página 49](#).

Si se introducen valores no permitidos, se utilizará el valor predeterminado. Esto resulta importante si se vuelven a utilizar declaraciones de estilo CSS que utilizan valores que no pertenecen al subconjunto de valores de Macromedia.

Los componentes pueden admitir los estilos siguientes:

| Estilo          | Descripción                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------|
| backgroundColor | Fondo de un componente. Éste es el único estilo de color que no hereda su valor. El valor predeterminado es transparente.                |
| borderColor     | Sección negra de un borde tridimensional o sección de color de un borde bidimensional. El valor predeterminado es 0x000000 (negro).      |
| borderStyle     | Borde del componente: puede ser “none”, “inset”, “outset” o “solid”. Este estilo no hereda su valor. El valor predeterminado es “solid”. |
| buttonColor     | La cara de un botón y una sección del borde tridimensional. El valor predeterminado es 0xEFEFEF (gris claro).                            |
| color           | Texto de la etiqueta de un componente. El valor predeterminado es 0x000000 (negro).                                                      |
| disabledColor   | Color desactivado del texto. El color predeterminado es 0x848384 (gris oscuro).                                                          |
| fontFamily      | Nombre de fuente del texto. El valor predeterminado es _sans.                                                                            |
| fontSize        | Tamaño en puntos de la fuente. El valor predeterminado es 10.                                                                            |
| fontStyle       | Estilo de la fuente: puede ser “normal” o “italic”. El valor predeterminado es “normal”.                                                 |
| fontWeight      | Grosor de la fuente: puede ser “normal” o “bold”. El valor predeterminado es “normal”.                                                   |
| highlightColor  | Sección del borde tridimensional. El valor predeterminado es 0xFFFFFFFF (blanco).                                                        |
| marginLeft      | Número que indica el margen izquierdo del texto. El valor predeterminado es 0.                                                           |
| marginRight     | Número que indica el margen derecho del texto. El valor predeterminado es 0.                                                             |

| Estilo                        | Descripción                                                                                                                                            |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| scrollTrackColor              | Guía de desplazamiento de la barra de desplazamiento. El valor predeterminado es 0xEFEEEF (gris claro).                                                |
| shadowColor                   | Sección del borde tridimensional. El valor predeterminado es 0x848384 (gris oscuro).                                                                   |
| symbolBackgroundColor         | Color de fondo de las casillas de verificación y de los botones de opción. El valor predeterminado es 0xFFFFFF (blanco).                               |
| symbolBackgroundDisabledColor | Color de fondo de las casillas de verificación y de los botones de opción cuando están desactivados. El valor predeterminado es 0xEFEEEF (gris claro). |
| symbolBackgroundPressedColor  | Color de fondo de las casillas de verificación y de los botones de opción cuando se presionan. El valor predeterminado es 0xFFFFFF (blanco).           |
| symbolColor                   | Marca de verificación de la casilla de verificación o punto del botón de opción. El valor predeterminado es 0x000000 (negro).                          |
| symbolDisabledColor           | Color de la marca de verificación o del punto del botón de opción desactivado. El valor predeterminado es 0x848384 (gris oscuro).                      |
| textAlign                     | Alineación del texto: puede ser "left", "right" o "center". El valor predeterminado es "left".                                                         |
| textDecoration                | Decoración del texto: puede ser "none" o "underline". El valor predeterminado es "none".                                                               |
| textIndent                    | Número que indica la sangría del texto. El valor predeterminado es 0.                                                                                  |

## Temas

Los temas son conjuntos de estilos y de aspectos. El tema predeterminado de Flash MX 2004 y Flash MX Professional 2004 se llama Halo (HaloTheme fla). El tema Halo se ha desarrollado para permitir que los usuarios saquen el máximo rendimiento posible. Flash MX 2004 y Flash MX Professional 2004 incluye otro tema llamado Sample (SampleTheme fla). El tema Sample le permite experimentar con el conjunto completo de estilos disponibles para los componentes de la v2. El tema Halo sólo utiliza un subconjunto de los estilos disponibles. Los archivos del tema se encuentran en las carpetas siguientes:

- First Run\ComponentFLA (Windows)
- First Run\ComponentFLA (Macintosh)

Puede crear temas nuevos y aplicarlos a una aplicación para cambiar la apariencia de todos los componentes. Por ejemplo, puede crear un tema bidimensional y otro tridimensional.

Los componentes de v2 utilizan aspectos (símbolos de clip de película o gráficos) para visualizar las apariencias visuales. El archivo .as que define cada componente contiene código que carga aspectos específicos del componente. Puede crear fácilmente un tema nuevo haciendo una copia del tema Halo o el tema Sample y modificando los gráficos de los aspectos.

Un tema también puede contener un conjunto de estilos nuevo. Para crear una declaración de estilo global o cualquier otra declaración de estilo adicional, es preciso escribir código ActionScript. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31.](#)

## Aplicación de un tema a un documento

Para aplicar un tema nuevo a un documento, abra un archivo FLA del tema como biblioteca externa y arrastre la carpeta Theme desde la biblioteca externa hasta la biblioteca del documento. Los pasos siguientes explican el proceso en detalle.

### Para aplicar un tema a un documento:

- 1 Seleccione Archivo > Abrir y abra el documento que utiliza componentes de v2 en Flash o seleccione Archivo > Nuevo y cree un documento nuevo que utilice componentes de v2.
- 2 Seleccione Archivo > Guardar y elija un nombre exclusivo como **ThemeApply fla**.
- 3 Seleccione Archivo > Importar > Abrir biblioteca externa y seleccione el archivo FLA del tema que desee aplicar al documento.  
Si no ha creado un tema nuevo, puede utilizar el tema Sample de la carpeta Flash 2004/en/Configuration/SampleFLA.
- 4 En el panel Biblioteca del tema, seleccione Flash UI Components 2 > Themes > MMDefault y arrastre la carpeta Assets de cualquier componente del documento a la biblioteca ThemeApply fla.  
Si no está seguro de qué componentes se encuentran en los documentos, puede arrastrar la carpeta Themes completa al escenario. Los aspectos que se encuentran en la carpeta Themes de la biblioteca se asignan automáticamente a componentes del documento.  
**Nota:** la vista previa dinámica de los componentes del escenario no reflejará el tema nuevo.
- 5 Seleccione Control > Probar película para ver el documento con el tema nuevo aplicado.

## Creación de un tema nuevo

Si no desea utilizar el tema Halo ni el tema Sample, puede modificar cualquiera de ellos para crear un tema nuevo.

Algunos aspectos de los temas tienen un tamaño fijo. Puede agrandarlos o reducirlos y los componentes cambiarán de tamaño automáticamente para coincidir con ellos. Otros aspectos están formados por varias piezas, algunas de ellas estáticas y otras no.

Algunos aspectos (por ejemplo RectBorder y ButtonSkin) utilizan la interfaz API de dibujo de ActionScript para dibujar sus gráficos, ya que es más eficiente desde el punto de vista del tamaño y del rendimiento. En dichos aspectos, puede utilizar código ActionScript como plantilla para ajustar los aspectos a sus necesidades.

### Para crear un tema nuevo:

- 1 Seleccione el archivo FLA del tema que desee utilizar como plantilla y haga una copia.  
Asigne a la copia un nombre exclusivo como **MyTheme fla**.
- 2 Seleccione Archivo > Abrir MyTheme fla en Flash.
- 3 Seleccione Ventana > Biblioteca para abrir la biblioteca si aún no está abierta.
- 4 Haga doble clic en el símbolo de aspecto que desee modificar para abrirlo en el modo de edición de símbolos.  
Los aspectos se encuentran en la carpeta Themes > MMDefault > *Componente Assets* (en este ejemplo, Themes > MMDefault > RadioButton Assets).
- 5 Modifique el símbolo o elimine los gráficos y cree gráficos nuevos.  
Puede que necesite seleccionar Ver > Acercar para subir el grado de aumento. Si se edita un aspecto, es preciso mantener el punto de registro para que el aspecto se vea correctamente. La esquina superior izquierda de todos los símbolos editados debe estar en (0,0).
- 6 Cuando haya terminado de editar el símbolo de aspecto, haga clic en el botón Atrás situado en la parte izquierda de la barra de información en el área superior del escenario para volver al modo de edición de documentos.
- 7 Repita los pasos 4, 5 y 6 hasta que haya terminado de editar todos los aspectos que desee cambiar.
- 8 Aplique MyTheme fla a un documento siguiendo los pasos indicados en la sección anterior, [“Aplicación de un tema a un documento” en la página 40](#).

## Aplicación de aspectos a los componentes

Los aspectos son símbolos que un componente utiliza para mostrar su apariencia. Pueden ser símbolos gráficos o símbolos de clip de película. La mayoría de los aspectos contienen formas que representan la apariencia del componente. Algunos aspectos contienen sólo código ActionScript que dibuja el componente en el documento.

Los componentes de Macromedia v2 son clips compilados, cuyos elementos no se pueden ver en la biblioteca. No obstante, con Flash se instalan archivos FLA que contienen todos los aspectos de componente. Estos archivos FLA reciben el nombre de *temas*. Cada tema tiene un comportamiento y una apariencia diferentes, aunque contiene aspectos con los mismos nombres de símbolo e identificadores de vínculo. Esto le permite arrastrar un tema al escenario de un documento para cambiar su apariencia. Para más información sobre los temas, consulte [“Temas” en la página 39](#). Los archivos FLA del tema también pueden utilizarse para editar aspectos de componente. Los aspectos se encuentran en la carpeta Themes del panel Biblioteca de cada FLA del tema.

Cada componente está formado por varios aspectos. Por ejemplo, la flecha abajo del subcomponente ScrollBar está formada por tres aspectos: ScrollDownArrowDisabled, ScrollDownArrowUp y ScrollDownArrowDown. Algunos componentes comparten aspectos. Los componentes que utilizan barras de desplazamiento (incluidos ComboBox, List y ScrollPane) comparten los aspectos de la carpeta ScrollBar Skins. Puede editar los aspectos ya existentes y crear aspectos nuevos para cambiar la apariencia de un componente.

El archivo .as que define cada clase de componente contiene código que carga aspectos específicos del componente. Cada aspecto de un componente tiene una propiedad de aspecto que se asigna al identificador de vinculación del símbolo de un aspecto. Por ejemplo, el estado Presionado (abajo) de la flecha abajo de ScrollBar tiene el nombre de propiedad de aspecto `downArrowDownName`. El valor predeterminado de la propiedad `downArrowDownName` es `"ScrollDownArrowDown"`, que es el identificador de vinculación del símbolo de aspecto. Los aspectos se pueden editar y aplicar a un componente mediante las propiedades de dichos aspectos. No tiene que editar el archivo .as del componente para cambiar sus propiedades de aspecto; simplemente, puede pasar los valores de propiedad del aspecto a la función constructora del componente cuando éste se crea en el documento.

Seleccione uno de los siguientes procedimientos para aplicar aspectos a un componente según lo que desee hacer:

- Para sustituir todos los aspectos de un documento por un conjunto nuevo (en el que cada tipo de componente comparta la misma apariencia), aplique un tema (véase [“Temas” en la página 39](#)).

**Nota:** este método de aplicación de aspectos está recomendado para principiantes, ya que no es preciso crear ningún script.

- Para utilizar diferentes aspectos para varias instancias del mismo componente, edite los aspectos existentes y defina las propiedades del aspecto (véase la sección siguiente, [“Edición de aspectos de componentes” en la página 42](#) y [“Aplicación de un aspecto editado a un componente” en la página 43](#)).
- Para cambiar aspectos de un subcomponente (como una barra de desplazamiento de un componente List), ponga el componente en una subclase (véase [“Aplicación de un aspecto editado a un subcomponente” en la página 44](#)).
- Para cambiar aspectos de un subcomponente a los que no se tiene acceso directo desde el componente principal (como un componente List de un componente ComboBox), sustituya las propiedades del aspecto en el prototipo (véase [“Cambio de propiedades de aspecto en el prototipo” en la página 47](#)).

**Nota:** los métodos indicados están enumerados de mayor a menor, basándose en su facilidad de uso.

## Edición de aspectos de componentes

Si desea utilizar un aspecto determinado para una instancia de un componente, pero dispone de otro aspecto de otra instancia del componente, debe abrir un archivo FLA del tema y crear un símbolo de aspecto nuevo. Los componentes se han diseñado para facilitar el uso de aspectos diferentes para instancias diferentes.

**Para editar un aspecto, siga este procedimiento:**

- 1 Seleccione Archivo > Abrir y abra el archivo FLA del tema que desea utilizar como plantilla.
- 2 Seleccione Archivo > Guardar como y, a continuación, seleccione un nombre exclusivo como `MyTheme fla`.
- 3 Seleccione el aspecto o aspectos que desea editar (en este ejemplo, se trata de RadioTrueUp). Los aspectos se encuentran en la carpeta Themes > MMDefault > *Componente Assets* (en este ejemplo, Themes > MMDefault > RadioButton Assets > States).
- 4 Seleccione Duplicar en el menú Opciones de Biblioteca (o haga clic con el botón derecho del ratón sobre el símbolo) y asígnele un nombre exclusivo como `MyRadioTrueUp`.

- 5 Seleccione el botón Avanzado en el cuadro de diálogo Propiedades de símbolo y seleccione Exportar para ActionScript.  
Se introducirá automáticamente un identificador de vinculación que coincida con el nombre de símbolo.
- 6 Haga doble clic en el aspecto nuevo de la biblioteca para abrirlo en modo de edición de símbolos.
- 7 Modifique el clip de película o elimínelo y cree uno nuevo.  
Puede que necesite seleccionar Ver > Acercar para subir el grado de aumento. Si se edita un aspecto, es preciso mantener el punto de registro para que el aspecto se vea correctamente. La esquina superior izquierda de todos los símbolos editados debe estar en (0,0).
- 8 Cuando haya terminado de editar el símbolo de aspecto, haga clic en el botón Atrás situado en la parte izquierda de la barra de información en el área superior del escenario para volver al modo de edición de documentos.
- 9 Seleccione Archivo > Guardar, pero no cierre MyTheme fla. Ahora debe crear un documento nuevo donde aplicar el aspecto editado a un componente.  
Para más información, consulte la sección siguiente, [“Aplicación de un aspecto editado a un componente” en la página 43](#), [“Aplicación de un aspecto editado a un subcomponente” en la página 44](#) o [“Cambio de propiedades de aspecto en el prototipo” en la página 47](#). Para obtener información sobre cómo aplicar un aspecto nuevo, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).

**Nota:** los cambios efectuados en los aspectos de componente no aparecen al ver componentes en el escenario mediante la función Vista previa dinámica.

## Aplicación de un aspecto editado a un componente

Cuando haya acabado de editar un aspecto, tendrá que aplicarlo a un componente de un documento. Puede utilizar el método `createClassObject()` para crear dinámicamente las instancias del componente, o bien puede poner manualmente las instancias del componente en el escenario. Existen dos maneras diferentes de aplicar aspectos a instancias de un componente, según cómo añada los componentes al documento.

**Para crear un componente de forma dinámica y aplicar un aspecto editado, siga este procedimiento:**

- 1 Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
- 2 Seleccione Archivo > Guardar y asígnele un nombre exclusivo como **DynamicSkinning fla**.
- 3 Arrastre los componentes desde el panel Componentes hasta el escenario, incluido el componente cuyo aspecto ha editado (en este ejemplo, se trata de **RadioButton**) y elimínelos.  
De esta manera, se añaden los símbolos a la biblioteca del documento, aunque no los vuelve visibles en éste.
- 4 Arrastre **MyRadioTrueUp** y cualquier otro símbolo que haya personalizado desde **MyTheme fla** al escenario de **DynamicSkinning fla** y elimínelos.  
De esta manera, se añaden los símbolos a la biblioteca del documento, aunque no los vuelve visibles en éste.

- 5 Abra el panel Acciones e introduzca lo siguiente en el fotograma 1:

```
import mx.controls.RadioButton
createClassObject(RadioButton, "myRadio", 0, {trueUpIcon:"MyRadioTrueUp",
  etiqueta: "My Radio Button"});
```

- 6 Seleccione Control > Probar película.

**Para añadir manualmente un componente al escenario y aplicar un aspecto editado, siga este procedimiento:**

- 1 Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
- 2 Seleccione Archivo > Guardar y asígnele un nombre exclusivo como **ManualSkinning fla**.
- 3 Arrastre los componentes desde el panel Componentes hasta el escenario, incluido el componente cuyo aspecto ha editado (en este ejemplo, se trata de RadioButton).
- 4 Arrastre MyRadioTrueUp y cualquier otro símbolo que haya personalizado desde MyTheme fla al escenario de ManualSkinning fla y elimínelos.

De esta manera, se añaden los símbolos a la biblioteca del documento, aunque no los vuelve visibles en éste.

- 5 Seleccione el componente RadioButton en el escenario y abra el panel Acciones.
- 6 Asocie el código siguiente a la instancia RadioButton:

```
onClipEvent(initialize){
  trueUpIcon = "MyRadioTrueUp";
}
```

- 7 Seleccione Control > Probar película.

## Aplicación de un aspecto editado a un subcomponente

En determinadas situaciones, es posible que desee modificar los aspectos de un subcomponente de un componente y se encuentre con que las propiedades de dichos aspectos no están disponibles directamente (por ejemplo, no hay forma directa de modificar los aspectos de la barra de desplazamiento de un componente List). El código siguiente le permite tener acceso a los aspectos de la barra de desplazamiento. Todas las barras de desplazamiento que se creen después de ejecutar este código también tendrán los aspectos nuevos.

Si un componente está formado por subcomponentes, dichos subcomponentes se identifican en la entrada del componente en el [Capítulo 4, “Diccionario de componentes”](#), en la [página 49](#).

**Para aplicar un aspecto nuevo a un subcomponente, siga este procedimiento:**

- 1 Siga los pasos indicados en [“Edición de aspectos de componentes” en la página 42](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto ScrollDownArrowDown y asígnele el nombre nuevo **MyScrollDownArrowDown**.
- 2 Seleccione Archivo > Nuevo para crear un documento de Flash nuevo.
- 3 Seleccione Archivo > Guardar y asígnele un nombre exclusivo como **SubcomponentProject fla**.
- 4 Haga doble clic en el componente List del panel Componentes para añadirlo al escenario y presione Retroceso para eliminarlo del mismo.

De esta manera, se añade el componente al panel Biblioteca, aunque no lo vuelve visible en el documento.



- 5 Arrastre MyScrollDownArrowDown y cualquier otro símbolo que haya editado desde MyTheme.fla al escenario de SubcomponentProject.fla y elimínelo.

De esta manera, se añade el componente al panel Biblioteca, aunque no lo vuelve visible en el documento.

- 6 Siga uno de estos procedimientos:

- Si desea cambiar todas las barras de desplazamiento de un documento, introduzca el código siguiente en el panel Acciones, en el fotograma 1 de la línea de tiempo:

```
import mx.controls.List
import mx.controls.scrollClasses.ScrollBar
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
```

A continuación puede introducir el código siguiente en el fotograma 1 para crear una lista de forma dinámica:

```
createClassObject(List, "myListBox", 0, {dataProvider: ["AL","AR","AZ",
    "CA","HI","ID", "KA","LA","MA"]});
```

O bien, puede arrastrar un componente List desde la biblioteca al escenario.

- Si desea cambiar una barra de desplazamiento específica de un documento, introduzca el código siguiente en el panel Acciones, en el fotograma 1 de la línea de tiempo:

```
import mx.controls.List
import mx.controls.scrollClasses.ScrollBar
var oldName = ScrollBar.prototype.downArrowDownName;
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
createClassObject(List, "myList1", 0, {dataProvider: ["AL","AR","AZ",
    "CA","HI","ID", "KA","LA","MA"]});
myList1.redraw(true);
ScrollBar.prototype.downArrowDownName = oldName;
```

**Nota:** se debe definir información suficiente para que las barras de desplazamiento aparezcan, o definir la propiedad `vScrollPolicy` en `true`.

- 7 Seleccione Control > Probar película.

También puede definir los aspectos de un subcomponente para todos los componentes de un documento definiendo la propiedad del aspecto en el objeto prototipo del subcomponente en la sección `#initclip` del símbolo del aspecto. Para más información sobre el objeto prototipo, consulte `Function.prototype` en el Diccionario de ActionScript de la Ayuda.

**Para utilizar `#initclip` para aplicar un aspecto editado a todos los componentes de un documento, siga este procedimiento:**

- 1 Siga los pasos indicados en [“Edición de aspectos de componentes” en la página 42](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto `ScrollDownArrowDown` y asígnele el nombre nuevo **MyScrollDownArrowDown**.
- 2 Seleccione Archivo > Nuevo para crear un documento de Flash nuevo. Guárdelo con un nombre exclusivo, por ejemplo, **SkinsInitExample.fla**.
- 3 Seleccione el símbolo `MyScrollDownArrowDown` en la biblioteca del ejemplo de la biblioteca del tema editado, arrástrelo al escenario de **SkinsInitExample.fla** y elimínelo.  
Esto añade el símbolo a la biblioteca sin hacerlo visible en el escenario.
- 4 Seleccione `MyScrollDownArrowDown` en la biblioteca **SkinsInitExample.fla** y seleccione Vinculación en el menú Opciones.
- 5 Active la casilla de verificación Exportar para ActionScript. Haga clic en Aceptar.  
Se seleccionará automáticamente Exportar en primer fotograma.

- 6 Haga doble clic en MyScrollDownArrowDown de la biblioteca para abrirlo en el modo de edición de símbolos.
- 7 Introduzca el código siguiente en el fotograma 1 del símbolo MyScrollDownArrowDown:

```
#initclip 10
import mx.controls.scrollClasses.ScrollBar;
ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
#endinitclip
```

- 8 Para añadir un componente List al documento, siga uno de estos procedimientos:
  - Arrastre un componente List desde el panel Componentes al escenario. Introduzca suficientes parámetros de Label para que aparezca la barra de desplazamiento vertical.
  - Arrastre un componente List desde el panel Componentes al escenario y elimínelo. Introduzca el código siguiente en el fotograma 1 de la línea de tiempo principal de SkinsInitExample.fla:

```
createClassObject(mx.controls.List, "myListBox1", 0, {dataProvider:
  ["AL", "AR", "AZ", "CA", "HI", "ID", "KA", "LA", "MA"]});
```

**Nota:** añada información suficiente para que aparezca la barra de desplazamiento vertical, o defina vScrollPolicy en true.

En el siguiente ejemplo se explica cómo aplicar aspectos a algo que ya está en el escenario. En este ejemplo sólo se aplican aspectos a los componentes List; no se aplican aspectos a las barras de desplazamiento TextArea o ScrollPane.

**Para utilizar #initclip para aplicar un aspecto editado a componentes específicos de un documento, siga este procedimiento:**

- 1 Siga los pasos indicados en [“Edición de aspectos de componentes” en la página 42](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto ScrollDownArrowDown y asígnele el nombre nuevo **MyScrollDownArrowDown**.
- 2 Seleccione Archivo > Nuevo y cree un documento de Flash.
- 3 Seleccione Archivo > Guardar y asigne al archivo un nombre exclusivo como **MyVScrollTest.fla**.
- 4 Arrastre MyScrollDownArrowDown desde la biblioteca del tema a la biblioteca MyVScrollTest.fla.
- 5 Seleccione Insertar > Nuevo símbolo y asígnele un nombre exclusivo como **MyVScrollBar**.
- 6 Active la casilla de verificación Exportar para ActionScript. Haga clic en Aceptar. Se seleccionará automáticamente Exportar en primer fotograma.
- 7 Introduzca el código siguiente en el fotograma 1 del símbolo MyVScrollBar:
 

```
#initclip 10
import MyVScrollBar
Object.registerClass("VScrollBar", MyVScrollBar);
#endinitclip
```
- 8 Arrastre un componente List desde el panel Componentes al escenario.
- 9 En el inspector de propiedades, introduzca todos los parámetros de Label necesarios para que aparezca la barra de desplazamiento vertical.
- 10 Seleccione Archivo > Guardar.
- 11 Seleccione Archivo > Nuevo y cree un archivo de ActionScript nuevo.

12 Introduzca el código siguiente:

```
import mx.controls.VScrollBar
import mx.controls.List
class MyVScrollBar extends VScrollBar{
    function init():Void{
        if (_parent instanceof List){
            downArrowDownName = "MyScrollDownArrowDown";
        }
        super.init();
    }
}
```

13 Seleccione Archivo > Guardar y guarde este archivo como **MyVScrollBar.as**.

14 Haga clic en un área vacía del escenario y, en el inspector de propiedades, seleccione el botón Configuración de publicación.

15 Seleccione el botón Configuración de la versión ActionScript.

16 Haga clic en el botón Más (+) para añadir una nueva ruta de clases y seleccione el botón Destino para desplazarse hasta la ubicación del archivo MyComboBox.as en el disco duro.

17 Seleccione Control > Probar película.

## Cambio de propiedades de aspecto en el prototipo

Si un componente no admite directamente variables del aspecto, puede definir el componente como subclase y sustituir sus aspectos. Por ejemplo, el componente ComboBox no admite directamente la aplicación de aspectos a su lista desplegable, ya que utiliza un componente List como lista desplegable.

Si un componente está formado por subcomponentes, dichos subcomponentes se identifican en la entrada del componente en el [Capítulo 4, “Diccionario de componentes”, en la página 49](#).

### Para aplicar aspectos a un subcomponente, siga este procedimiento:

1 Siga los pasos indicados en [“Edición de aspectos de componentes” en la página 42](#), pero edite un aspecto de la barra de desplazamiento. En este ejemplo, edite el aspecto ScrollDownArrowDown y asígnele el nombre nuevo **MyScrollDownArrowDown**.

2 Seleccione Archivo > Nuevo y cree un documento de Flash.

3 Seleccione Archivo > Guardar y asigne al archivo un nombre exclusivo como **MyComboTest fla**.

4 Arrastre MyScrollDownArrowDown desde la biblioteca del tema hasta el escenario de MyComboTest fla y elimínelo.

Esto añade el símbolo a la biblioteca, pero no lo hace visible en el escenario.

5 Seleccione Insertar > Nuevo símbolo y asígnele un nombre exclusivo como **MyComboBox**.

6 Seleccione la casilla de verificación Exportar para ActionScript y haga clic en Aceptar. Se seleccionará automáticamente Exportar en primer fotograma.

7 Introduzca el código siguiente en el panel Acciones en las acciones del fotograma 1 de MyComboBox:

```
#initclip 10
import MyComboBox
Object.registerClass("ComboBox", MyComboBox);
#endinitclip
```

8 Arrastre un componente ComboBox al escenario.

9 En el inspector de propiedades, introduzca todos los parámetros de Label necesarios para que aparezca la barra de desplazamiento vertical.

10 Seleccione Archivo > Guardar.

11 Seleccione Archivo > Nuevo y cree un archivo de ActionScript nuevo (sólo en Flash Professional).

12 Introduzca el código siguiente:

```
import mx.controls.ComboBox
import mx.controls.scrollClasses.ScrollBar
class MyComboBox extends ComboBox{
    function getDropdown():Object{
        var oldName = ScrollBar.prototype.downArrowDownName;
        ScrollBar.prototype.downArrowDownName = "MyScrollDownArrowDown";
        var r = super.getDropdown();
        ScrollBar.prototype.downArrowDownName = oldName;
        return r;
    }
}
```

13 Seleccione Archivo > Guardar y guarde este archivo como **MyComboBox.as**.

14 Haga clic en un área vacía del escenario y, en el inspector de propiedades, seleccione el botón Configuración de publicación.

15 Seleccione el botón Configuración de la versión ActionScript.

16 Haga clic en el botón Más (+) para añadir una nueva ruta de clases y seleccione el botón Destino para desplazarse hasta la ubicación del archivo MyComboBox.as en el disco duro.

17 Seleccione Control > Probar película.

# CAPÍTULO 4

## Diccionario de componentes

En este capítulo se describen los componentes y la interfaz API (interfaz de programación de aplicaciones) de cada uno de ellos.

La descripción de cada componente contiene la información siguiente:

- Interacción con el teclado
- Vista previa dinámica
- Accesibilidad
- Configuración de los parámetros del componente
- Utilización del componente en una aplicación
- Personalización del componente con estilos y aspectos
- Métodos, propiedades y eventos de ActionScript

Los componentes se presentan en orden alfabético, pero en las tablas siguientes podrá encontrarlos por categoría:

### Componentes de la interfaz de usuario (IU)

| Componente                                                            | Descripción                                                                                                                                                                                                         |
|-----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Componente Accordion (sólo para Flash Professional)</a>   | Conjunto de vistas verticales superpuestas con botones en la parte superior que permiten a los usuarios cambiar de vista.                                                                                           |
| <a href="#">Componente Alert (sólo Flash Professional)</a>            | Ventana que presenta al usuario una pregunta y botones para elegir la respuesta.                                                                                                                                    |
| <a href="#">Componente Button</a>                                     | Botón que puede cambiarse de tamaño y personalizarse con un icono personalizado.                                                                                                                                    |
| <a href="#">Componente CheckBox</a>                                   | Permite hacer una elección booleana (true o false).                                                                                                                                                                 |
| <a href="#">Componente ComboBox</a>                                   | Permite seleccionar una opción en una lista de desplazamiento de opciones. Este componente puede tener un campo de texto seleccionable en la parte superior de la lista que permite a los usuarios buscar la lista. |
| <a href="#">Componente DateChooser (sólo para Flash Professional)</a> | Permite a los usuarios seleccionar una o varias fechas en un calendario.                                                                                                                                            |

| Componente                                          | Descripción                                                                                                                                                             |
|-----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Componente DateField (sólo para Flash Professional) | Campo de texto no seleccionable con un icono de calendario. Si el usuario hace clic en cualquier punto del recuadro de delimitación, aparece un componente DateChooser. |
| Componente DataGrid (sólo para Flash Professional)  | Permite a los usuarios mostrar y manipular varias columnas de datos.                                                                                                    |
| Componente Label                                    | Campo de texto no editable de una línea.                                                                                                                                |
| Componente List                                     | Permite seleccionar una o varias opciones en una lista de desplazamiento.                                                                                               |
| Componente Loader                                   | Contenedor de un cargador de archivos SWF o JPEG.                                                                                                                       |
| Componente Menu (sólo para Flash Professional)      | Permite seleccionar un comando en una lista; menú de aplicación estándar del escritorio.                                                                                |
| Componente MenuBar (sólo para Flash Professional)   | Barra horizontal de menús.                                                                                                                                              |
| Componente NumericStepper                           | Flechas para aumentar o reducir una cifra con el ratón.                                                                                                                 |
| Componente ProgressBar                              | Muestra el progreso de un proceso, normalmente de carga.                                                                                                                |
| Componente RadioButton                              | Permite seleccionar entre opciones que se excluyen entre sí.                                                                                                            |
| Componente ScrollPane                               | Muestra películas, mapas de bits y archivos SWF en un área limitada con barras de desplazamiento automático.                                                            |
| Componente TextArea                                 | Campo de texto de varias líneas opcionalmente editable.                                                                                                                 |
| Componente TextInput                                | Campo de texto de una línea opcionalmente editable.                                                                                                                     |
| Componente Tree (sólo para Flash Professional)      | Permite manipular información jerárquica.                                                                                                                               |
| Componente Window                                   | Ventana desplazable con barra de título, texto, borde y botón de cierre que muestra contenido al usuario.                                                               |

## Componentes de datos

| Componente                                                    | Descripción                                                                                                                                                                 |
|---------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Clases de vinculación de datos (sólo para Flash Professional) | Estas clases implementan la funcionalidad de vinculación de datos en tiempo de ejecución de Flash.                                                                          |
| Componente DataHolder (sólo para Flash Professional)          | Almacena información y puede utilizarse como conector entre componentes.                                                                                                    |
| Interfaz API de DataProvider                                  | Este componente es el modelo para las listas de datos de acceso lineal. Este modelo proporciona capacidades sencillas de manipulación de matrices que difunden sus cambios. |
| Componente DataSet (sólo para Flash Professional)             | Bloque para crear aplicaciones gestionadas por datos.                                                                                                                       |

| Componente                                                                | Descripción                                                                                                                                                                                      |
|---------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Componente RDBMSResolver (sólo para Flash Professional)</a>   | Permite guardar datos en cualquier fuente de datos admitida. Este componente resolver convierte el XML que se puede recibir y analizar mediante un servicio Web, JavaBean, servlet o página ASP. |
| <a href="#">Clases de servicios Web (sólo para Flash Professional)</a>    | Estas clases permiten acceder a servicios Web que utilizan el Protocolo sencillo de acceso a objetos (SOAP) que se encuentra en el paquete mx.services.                                          |
| <a href="#">Clase WebServiceConnector (sólo para Flash Professional)</a>  | Proporciona acceso sin script a llamadas de método de servicio Web.                                                                                                                              |
| <a href="#">Componente XMLConnector (sólo para Flash Professional)</a>    | Lee y graba documentos XML con los métodos HTTP GET y POST.                                                                                                                                      |
| <a href="#">Componente XUpdateResolver (sólo para Flash Professional)</a> | Permite guardar datos en cualquier fuente de datos admitida. Este componente resolver convierte DeltaPacket a XUpdate.                                                                           |

## Componentes multimedia

| Componente                 | Descripción                                                    |
|----------------------------|----------------------------------------------------------------|
| Componente MediaController | Controla la reproducción de flujo de medios en una aplicación. |
| Componente MediaDisplay    | Muestra los flujos de medios en una aplicación.                |
| Componente MediaPlayer     | Combinación de componentes MediaDisplay y MediaController.     |

Para más información sobre estos componentes, consulte [“Componentes multimedia \(sólo para Flash Professional\)” en la página 350](#).

## Administradores

| Componente                         | Descripción                                               |
|------------------------------------|-----------------------------------------------------------|
| <a href="#">Clase DepthManager</a> | Administra la profundidad de apilamiento de los objetos.  |
| <a href="#">Clase FocusManager</a> |                                                           |
| <a href="#">Clase PopUpManager</a> | Permite crear y eliminar ventanas emergentes.             |
| <a href="#">Clase StyleManager</a> | Permite registrar estilos y administra estilos heredados. |

## Pantallas

| Componente                                  | Descripción                                                                             |
|---------------------------------------------|-----------------------------------------------------------------------------------------|
| Clase Form (sólo para Flash Professional)   | Permite manipular las pantallas de aplicaciones de formularios en tiempo de ejecución.  |
| Clase Screen (sólo para Flash Professional) | Clase base para las clases Slide y Form.                                                |
| Clase Slide (sólo para Flash Professional)  | Permite manipular las pantallas de presentación de diapositivas en tiempo de ejecución. |

## Componente Accordion (sólo para Flash Professional)

El componente Accordion es un navegador que contiene una secuencia de elementos secundarios que se muestra de uno en uno. Los elementos secundarios deben ser una subclase de la clase UIObject (que incluye todos los componentes y pantallas que se han creado utilizando la versión 2 de la arquitectura de componentes de Macromedia), pero a menudo los elementos secundarios son una subclase de la clase View. Esto incluye clips de película asignados con la clase mx.core.View. Para mantener el orden de tabulación en los elementos secundarios de accordion, éstos deben ser también instancias de la clase View.

Un accordion crea y gestiona botones de encabezado que se pueden presionar para desplazarse por los elementos secundarios de accordion. Un accordion tiene un diseño vertical con botones de encabezado que abarcan la anchura del componente. Hay un encabezado asociado con cada secundario y cada encabezado pertenece al accordion y no a ese secundario. Cuando un usuario hace clic en un encabezado, se muestra el secundario asociado debajo de dicho encabezado. La transición al nuevo secundario utiliza una animación de transición.

Un accordion con elementos secundarios acepta una selección y cambia el aspecto de sus encabezados para mostrar la selección. Cuando el usuario desplaza el tabulador hasta un accordion, el encabezado seleccionado muestra el indicador de selección. Un accordion que no tenga elementos secundarios no acepta selecciones. Al hacer clic en componentes seleccionables dentro del elemento secundario seleccionado quedarán seleccionados. Cuando la instancia de Accordion esté seleccionada, podrá controlarla con las teclas siguientes:

| Tecla                           | Descripción                                                                                                                                                               |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Flecha abajo, flecha derecha    | Desplaza la selección al encabezado secundario siguiente. La selección se ajusta desde el último encabezado secundario al primero sin cambiar el secundario seleccionado. |
| Flecha arriba, flecha izquierda | Desplaza la selección al encabezado secundario anterior. La selección se ajusta desde el primer encabezado secundario al último sin cambiar el secundario seleccionado.   |
| Fin                             | Selecciona el último elemento secundario.                                                                                                                                 |
| Intro/Espacio                   | Selecciona el secundario asociado con el encabezado que está seleccionado.                                                                                                |
| Inicio                          | Selecciona el primer elemento secundario.                                                                                                                                 |
| Av Pág                          | Selecciona el siguiente elemento secundario. La selección se ajusta desde el último elemento secundario al primer elemento secundario.                                    |



| Tecla                  | Descripción                                                                                                                                                                                    |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Re Pág                 | Selecciona el secundario anterior. La selección se ajusta desde el primer elemento secundario al último elemento secundario.                                                                   |
| Mayúsculas + Tabulador | Desplaza la selección al componente anterior. Este componente puede estar dentro del elemento secundario seleccionado o fuera de accordion; nunca será otro encabezado en el mismo accordion.  |
| Tabulador              | Desplaza la selección al componente siguiente. Este componente puede estar dentro del elemento secundario seleccionado o fuera de accordion; nunca será otro encabezado en el mismo accordion. |

Los lectores de pantalla no pueden acceder al componente Accordion.

## Utilización del componente Accordion (sólo para Flash Professional)

El componente Accordion se puede utilizar para presentar formularios de múltiples partes. Por ejemplo, un accordion con tres elementos secundarios puede presentar formularios en los que el usuario escriba su dirección de envío y facturación e información de pago para una transacción de comercio electrónico. Con un accordion en lugar de múltiples páginas Web, el tráfico en el servidor se minimiza, lo que permite al usuario optimizar el progreso y la navegabilidad de la aplicación.

### Parámetros de Accordion

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Accordion en el inspector de propiedades o en el panel Inspector de componentes:

**childSymbols** Matriz que especifica los identificadores de vinculación de los símbolos de biblioteca que se utilizarán para crear los elementos secundarios de accordion. El valor predeterminado es [] (matriz vacía).

**childNames** Matriz que especifica los nombres de instancia de los elementos secundarios de accordion. El valor predeterminado es [] (matriz vacía).

**childLabels** Matriz que especifica las etiquetas de texto que se deben utilizar en los encabezados de accordion. El valor predeterminado es [] (matriz vacía).

**childIcons** Matriz que especifica los identificadores de vinculación de los símbolos de biblioteca que se utilizarán como iconos de los encabezados de accordion. El valor predeterminado es [] (matriz vacía).

Puede escribir código ActionScript para controlar éstas y otras opciones del componente Accordion utilizando sus propiedades, métodos y eventos. Para más información, consulte [“Clase Accordion \(sólo para Flash Professional\)” en la página 58](#).

## Creación de una aplicación con el componente Accordion

En este ejemplo, un desarrollador de aplicaciones está creando la sección de finalización de compra de una tienda en línea. El diseño requiere un accordion con tres formularios donde los usuarios introducen sus direcciones de envío y facturación y la información de pago. Los formularios de dirección de envío y dirección de facturación son idénticos.

### Para utilizar pantallas con el fin de añadir un componente Accordion a una aplicación:

- 1 En Flash, seleccione Archivo > Nuevo y, a continuación, Aplicación de formularios Flash.
- 2 Haga doble clic en el texto formulario1 e introduzca el nombre **addressForm**.  
Aunque no aparece en la biblioteca, la pantalla addressForm es un símbolo de la clase Screen (que es una subclase de la clase View), que un accordion puede utilizar como elemento secundario.
- 3 Con el formulario seleccionado, en el inspector de propiedades, defina el valor de la propiedad visible como `false`.  
Esto oculta los contenidos del formulario en la aplicación; el formulario sólo aparece en el Accordion.
- 4 Arrastre componentes como Label y TextInput del panel Componentes al formulario para crear un formulario con una dirección ficticia; organícelos y defina sus propiedades en el panel Parámetros del panel Inspector de componentes.  
Coloque los elementos del formulario en la esquina superior izquierda del formulario. Esta esquina se coloca en la esquina superior izquierda de Accordion.
- 5 Repita los pasos del 2 al 4 para crear una pantalla denominada **checkoutForm**.
- 6 Cree un nuevo formulario denominado **accordionForm**.
- 7 Arrastre un componente Accordion del panel Componentes al formulario accordionForm y asígnele el nombre **myAccordion**.
- 8 Con myAccordion seleccionado, en el inspector de propiedades, siga este procedimiento:
  - Para la propiedad **childSymbols**, introduzca **addressForm**, **addressForm** y **checkoutForm**.  
Estas cadenas especifican los nombres de las pantallas que se utilizan para crear los elementos secundarios de accordion.
  - Nota:** los dos primeros elementos secundarios son instancias de la misma pantalla ya que el formulario con la dirección de envío y el formulario con la dirección de facturación tienen componentes idénticos.
  - Para la propiedad **childNames**, introduzca **shippingAddress**, **billingAddress** y **checkout**.  
Estas cadenas son los nombres ActionScript de los elementos secundarios de accordion.
  - Para la propiedad **childLabels**, introduzca **Shipping Address**, **Billing Address** y **Checkout**.  
Estas cadenas son las etiquetas de texto de los encabezados de accordion.
- 9 Seleccione Control > Probar película.

**Para añadir un componente Accordion a una aplicación, siga este procedimiento:**

- 1 Seleccione Archivo > Nuevo y cree un documento de Flash nuevo.
- 2 Seleccione Insertar > Nuevo símbolo y asígnele el nombre **AddressForm**.
- 3 En el cuadro de diálogo Crear un nuevo símbolo, haga clic en el botón Avanzado y seleccione Exportar para ActionScript. En el campo Clase de AS 2.0, introduzca **mx.core.View**.  
Para mantener el orden de tabulación en los secundarios de accordion, los secundarios deben ser también instancias de la clase View.
- 4 Arrastre componentes como Label y TextInput del panel Componentes al escenario para crear un formulario con una dirección ficticia; organícelos y defina sus propiedades en el panel Parámetros del panel Inspector de componentes.  
Coloque los elementos del formulario con respecto a 0, 0 (la parte central) del escenario. La coordenada 0, 0 del clip de película se coloca en la esquina superior izquierda de Accordion.
- 5 Seleccione Edición > Editar documento para volver a la línea de tiempo principal.
- 6 Repita los pasos del 2 al 5 para crear un clip de película denominado **CheckoutForm**.
- 7 Arrastre un componente Accordion desde el panel Componentes para añadirlo al escenario en la línea de tiempo principal.

- 8 En el inspector de propiedades, siga este procedimiento:

- Introduzca el nombre de instancia **myAccordion**.
- Para la propiedad `childSymbols`, introduzca **AddressForm**, **AddressForm** y **CheckoutForm**.

Estas cadenas especifican los nombres de los clips de película que se utilizan para crear los elementos secundarios de accordion.

**Nota:** los dos primeros elementos secundarios son instancias del mismo clip de película ya que el formulario con la dirección de envío y el formulario con la dirección de facturación son idénticos.

- Para la propiedad `childNames`, introduzca **shippingAddress**, **billingAddress** y **checkout**.  
Estas cadenas son los nombres ActionScript de los elementos secundarios de accordion.
- Para la propiedad `childLabels`, introduzca **Shipping Address**, **Billing Address** y **Checkout**.  
Estas cadenas son las etiquetas de texto de los encabezados de accordion.
- Para la propiedad `childIcons`, introduzca **AddressIcon**, **AddressIcon** y **CheckoutIcon**.  
Estas cadenas especifican los identificadores de vinculación de los símbolos de clip de película que se utilizan como iconos de los encabezados de accordion. Debe crear estos símbolos de clips de película si desea iconos en los encabezados.

- 9 Seleccione Control > Probar película.

**Para utilizar ActionScript para añadir elementos secundarios a un componente Accordion, siga este procedimiento:**

- 1 Seleccione Archivo > Nuevo y cree un documento de Flash.
- 2 Arrastre un componente Accordion desde el panel Componentes al escenario.
- 3 En el inspector de propiedades, introduzca el nombre de instancia **myAccordion**.
- 4 Arrastre un componente TextInput al escenario y elimínelo.

De este modo, lo añade a la biblioteca para que pueda crear una instancia dinámicamente en el paso 6.

- 5 En el panel Acciones del Fotograma 1 de la línea de tiempo, introduzca lo siguiente:

```
myAccordion.createChild("View", "shippingAddress", { label: "Shipping  
Address" });  
myAccordion.createChild("View", "billingAddress", { label: "Billing Address"  
});  
myAccordion.createChild("View", "payment", { label: "Payment" });
```

Este código llama al método `createChild()` para crear sus vistas secundarias.

- 6 En el panel Acciones del Fotograma 1, debajo del código introducido en el paso 4, introduzca el código siguiente:

```
var o = myAccordion.shippingAddress.createChild("TextInput", "firstName");  
o.move(20, 38);  
o.setSize(116, 20);  
o = myAccordion.shippingAddress.createChild("TextInput", "lastName");  
o.move(175, 38);  
o.setSize(145, 20);
```

Este código añade instancias de componentes (dos componentes `TextInput`) a los elementos secundarios de `accordion`.

## Personalización del componente Accordion (sólo para Flash Professional)

El componente `Accordion` puede transformarse horizontal o verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)).

El método `setSize()` y la herramienta Transformación sólo cambian la anchura de los encabezados de `accordion` y la anchura y altura de su área de contenido. La altura de los encabezados y la anchura y altura de los elementos secundarios no se ven afectados. Llamar al método `setSize()` es la única forma de cambiar el rectángulo de delimitación de un `accordion`.

Si los encabezados son demasiado pequeños para contener los textos de etiqueta, éstos se recortan. Si el área de contenido de un `accordion` es más pequeña que un elemento secundario, éste se recorta.

## Utilización de estilos con el componente Accordion

Es posible definir propiedades de estilo para cambiar el aspecto del borde y el fondo de un componente `Accordion`.

Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

Un componente Accordion admite los estilos de Halo siguientes:

| Estilo          | Descripción                                                                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| themeColor      | Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son “haloGreen”, “haloBlue” y “haloOrange”.                                                                                             |
| backgroundColor | Color del fondo.                                                                                                                                                                                                                         |
| borderColor     | Color del borde.                                                                                                                                                                                                                         |
| borderStyle     | Estilo de borde; los valores posibles son “none”, “solid”, “inset”, “outset”, “default”, “alert”. El valor predeterminado es el aspecto del borde del componente Window y el valor “alert” es el aspecto del borde del componente Alert. |
| headerHeight    | Altura de los botones del encabezado, expresada en píxeles.                                                                                                                                                                              |
| color           | Color del texto de encabezado.                                                                                                                                                                                                           |
| disabledColor   | Color de un accordion desactivado.                                                                                                                                                                                                       |
| fontFamily      | Nombre de fuente para las etiquetas del encabezado.                                                                                                                                                                                      |
| fontSize        | Tamaño en puntos para la fuente de las etiquetas del encabezado.                                                                                                                                                                         |
| fontStyle       | Estilo de la fuente para las etiquetas del encabezado; puede ser “normal” o “italic”.                                                                                                                                                    |
| fontWeight      | Grosor de la fuente para las etiquetas del encabezado; puede ser “normal” o “bold”.                                                                                                                                                      |
| textDecoration  | Decoración del texto; puede ser “none” o “underline”.                                                                                                                                                                                    |
| openDuration    | Duración, en milisegundos, de la animación de transición.                                                                                                                                                                                |
| openEasing      | Función de interpolación que utiliza la animación.                                                                                                                                                                                       |

## Utilización de aspectos con el componente Accordion

El componente Accordion utiliza aspectos para representar los estados visuales de sus botones de encabezado. Para aplicar un aspecto a los botones y la barra de título durante la edición, modifique los símbolos del aspecto en la carpeta de estados de aspecto Flash UI Components 2/ Themes/MMDefault/Accordion Assets de la biblioteca de uno de los archivos FLA de temas. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).

Un componente Accordion se compone de su borde y fondo, botones de encabezado y elementos secundarios. El borde y el fondo pueden aplicar estilos pero no aspectos. Los encabezados pueden aplicar aspectos pero no estilos, utilizando el subconjunto de aspectos heredados de los botones que se enumeran a continuación. Un componente Accordion utiliza las siguientes propiedades de aspecto para aplicar dinámicamente un aspecto a los botones de encabezado:

| Propiedad     | Descripción                        | Valor predeterminado |
|---------------|------------------------------------|----------------------|
| falseUpSkin   | Estado Arriba.                     | accordionHeaderSkin  |
| falseDownSkin | Estado Presionado.                 | accordionHeaderSkin  |
| falseOverSkin | Estado al situar el cursor encima. | accordionHeaderSkin  |
| trueUpSkin    | Estado Conmutado.                  | accordionHeaderSkin  |

## Clase Accordion (sólo para Flash Professional)

**Herencia** UIObject > UIComponent > View > Accordion

**Nombre de clase de ActionScript** mx.containers.Accordion

Un Accordion es un componente que contiene elementos secundarios que se visualizan de uno en uno. Cada elemento secundario tiene su correspondiente botón de encabezado que se crea cuando se crea el elemento secundario. Un elemento secundario debe ser una instancia de UIObject.

Un símbolo de clip de película se convierte automáticamente en una instancia de la clase UIObject cuando pasa a ser un elemento secundario de un accordion. Sin embargo, para mantener el orden de tabulación en los elementos secundarios de accordion, los secundarios deben ser también instancias de la clase View. Si utiliza un símbolo de clip de película como símbolo secundario, defina el campo Class de AS 2.0 como `mx.core.View` de forma que herede de la clase View.

Si una propiedad de la clase Accordion se define con ActionScript, sustituye el parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.Accordion.version);
```

**Nota:** el código siguiente devuelve undefined: `trace(myAccordionInstance.version);`.

## Resumen de métodos de la clase Accordion

| Método                                  | Descripción                                                                                                                                               |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Accordion.createChild()</code>    | Crea un elemento secundario para una instancia de accordion.                                                                                              |
| <code>Accordion.createSegment()</code>  | Crea un elemento secundario para una instancia de accordion. Los parámetros para este método son diferentes a los del método <code>createChild()</code> . |
| <code>Accordion.destroyChildAt()</code> | Elimina un elemento secundario en una posición especificada de índice.                                                                                    |
| <code>Accordion.getChildAt()</code>     | Obtiene una referencia a un elemento secundario en una posición especificada de índice.                                                                   |

Hereda todos los métodos de [UIObject](#), [UIComponent](#) y `mx.core.View`.

## Resumen de propiedades de la clase Accordion

| Propiedad                            | Descripción                                                    |
|--------------------------------------|----------------------------------------------------------------|
| <code>Accordion.numChildren</code>   | Número de elementos secundarios de una instancia de accordion. |
| <code>Accordion.selectedChild</code> | Referencia al elemento secundario seleccionado.                |
| <code>Accordion.selectedIndex</code> | Posición de índice del elemento secundario seleccionado.       |

Hereda todas las propiedades de [UIObject](#), [UIComponent](#) y `mx.core.View`.

## Resumen de eventos de la clase Accordion

| Evento                        | Descripción                                                                                                                                                                                                  |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Accordion.change</code> | Se difunde a todos los detectores registrados cada vez que cambian las propiedades <code>selectedIndex</code> y <code>selectedChild</code> de un accordion al hacer clic con el ratón o presionar una tecla. |

Hereda todos los eventos de [UIObject](#), [UIComponent](#) y `mx.core.View`.

## Accordion.change

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.change = function(eventObject){
    // introducir aquí el código propio
}
myAccordionInstance.addEventListener("change", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando cambian las propiedades `selectedIndex` y `selectedChild` de un `accordion`. Este evento sólo se difunde cuando al hacer clic con el ratón o presionar una tecla se cambia el valor `selectedChild` o `selectedIndex`, no cuando se cambia el valor con `ActionScript`. Este evento se difunde antes de que se produzca la animación de transición.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. El componente `Accordion` distribuye un evento `change` cuando se presiona uno de sus botones, y éste se controla mediante una función (denominada también *controlador*) asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase una referencia al controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente se define y pasa el controlador `myAccordionListener` al método `myAccordion.addEventListener()` como segundo parámetro. El objeto de evento se captura mediante el controlador `change` en el parámetro `evtObject`. Cuando se difunde el evento `change`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
myAccordionListener = new Object();
myAccordionListener.change = function(){
    trace("Changed to different view");
}
myAccordion.addEventListener("change", myAccordionListener);
```

## Accordion.createChild()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myAccordion.createChild(classOrSymbolName, instanceName[, initialProperties])
```

### Parámetros

*classOrSymbolName* Este parámetro puede ser la función constructora para la clase del `UIObject` de la que se va a crear una instancia o el nombre de la vinculación, una referencia al símbolo del que se va a crear una instancia. La clase debe ser `UIObject` o una subclase de `UIObject`, pero normalmente es `View` o una subclase de `View`.

*instanceName* Nombre de la nueva instancia.



*initialProperties* Parámetro opcional que especifica las propiedades iniciales para la nueva instancia. Puede utilizar las siguientes propiedades:

- *label* Esta cadena especifica la etiqueta de texto que la nueva instancia secundaria utiliza en su encabezado.
- *icon* Esta cadena especifica el identificador de vinculación del símbolo de biblioteca que utiliza el elemento secundario para el icono en su encabezado.

### Valor devuelto

Referencia a una instancia del UIObject que es el nuevo elemento secundario creado.

### Descripción

Método (heredado de View); crea un elemento secundario para el Accordion. Este nuevo elemento secundario se añade al final de la lista de elementos secundarios que pertenecen al Accordion. Utilice este método para colocar vistas dentro del accordion. El elemento secundario creado es una instancia de la clase o símbolo de clip de película especificado en el parámetro *classOrSymbolName*. Puede utilizar las propiedades *label* e *icon* para especificar una etiqueta de texto y un icono para el encabezado de accordion asociado para cada elemento secundario del parámetro *initialProperties*.

Cuando se crea un elemento secundario se le asigna un número de índice en el orden de creación y la propiedad *numChildren* aumenta en 1 unidad.

### Ejemplo

En el código siguiente se crea una instancia del símbolo de clip de película *PaymentForm* denominada *payment* como el último elemento secundario de *myAccordion*:

```
var child = myAccordion.createChild("PaymentForm", "payment", { label:
    "Payment", Icon: "payIcon" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

El código siguiente crea un elemento secundario que es una instancia de la clase *View*:

```
var child = myAccordion.createChild(mx.core.View, "payment", { label:
    "Payment", Icon: "payIcon" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

El código siguiente también crea un elemento secundario que es una instancia de la clase *View*, pero utiliza la importación para establecer una referencia con el constructor para la clase *View*:

```
import mx.core.View
var child = myAccordion.createChild(View, "payment", { label: "Payment", Icon:
    "payIcon" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

## Accordion.createSegment()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

```
myAccordion.createSegment(classOrSymbolName, instanceName[, label[, icon]])
```

### Parámetros

*classOrSymbolName* Este parámetro puede ser una referencia a la función constructora para la clase del UIObject de la que se va a crear una instancia o el nombre de la vinculación del símbolo del que se va a crear una instancia. La clase debe ser UIObject o una subclase de UIObject, pero normalmente es View o una subclase de View.

*instanceName* Nombre de la nueva instancia.

*label* Esta cadena especifica la etiqueta de texto que la nueva instancia secundaria utiliza en su encabezado. Este parámetro es opcional.

*icon* Esta cadena es una referencia al identificador de vinculación del símbolo de biblioteca que utiliza el secundario para el icono en su encabezado. Este parámetro es opcional.

### Valor devuelto

Referencia a la nueva instancia UIObject creada.

### Descripción

Método; crea un elemento secundario para el Accordion. Este nuevo elemento secundario se añade al final de la lista de elementos secundarios que pertenecen al Accordion. Utilice este método para colocar vistas dentro del accordion. El elemento secundario creado es una instancia de la clase o símbolo de clip de película especificado en el parámetro *classOrSymbolName*. Puede utilizar los parámetros *label* e *icon* para especificar una etiqueta de texto y un icono para el encabezado de accordion asociado para cada elemento secundario.

El método `createSegment()` difiere del método `addChild()` en que *label* e *icon* se pasan directamente como parámetros, no como propiedades de un parámetro *initialProperties*.

Cuando se crea un elemento secundario se le asigna un número de índice en el orden de creación y la propiedad `numChildren` aumenta en 1 unidad.

### Ejemplo

En el ejemplo siguiente se crea una instancia del símbolo de clip de película `PaymentForm` denominada `payment` como el último elemento secundario de `myAccordion`:

```
var child = myAccordion.createSegment("PaymentForm", "payment", "Payment",  
    "payIcon");  
child.cardType.text = "Visa";  
child.cardNumber.text = "1234567887654321";
```

El código siguiente crea un elemento secundario que es una instancia de la clase View:

```
var child = myAccordion.createSegment(mx.core.View, "payment", { label:
    "Payment", Icon: "payIcon" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

El código siguiente también crea un elemento secundario que es una instancia de la clase View, pero utiliza la importación para establecer una referencia con el constructor para la clase View:

```
import mx.core.View
var child = myAccordion.createSegment(View, "payment", { label: "Payment",
    Icon: "payIcon" });
child.cardType.text = "Visa";
child.cardNumber.text = "1234567887654321";
```

## Accordion.destroyChildAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myAccordion.destroyChildAt(index)
```

### Parámetros

*index* Número de índice del elemento secundario de accordion que se debe eliminar. A cada elemento secundario de un accordion se le asigna un número de índice basado en cero en el orden en el que se creó.

### Valor devuelto

Ninguno.

### Descripción

Método (heredado de View); elimina uno de los elementos secundarios de accordion. El elemento secundario que se va a eliminar se especifica mediante su índice, que pasa al método en el parámetro *index*. Al llamar a este método se elimina también el encabezado correspondiente.

Si se selecciona un elemento secundario eliminado, se elige un nuevo elemento secundario seleccionado. Si hay un elemento secundario a continuación, éste se selecciona. Si no hay ningún elemento secundario a continuación, se selecciona el elemento secundario anterior. Si no hay ningún elemento secundario anterior, la selección no se define.

**Nota:** llamar al método `destroyChildAt()` reduce la propiedad `numChildren` en 1 unidad.

### Ejemplo

El código siguiente elimina al último elemento secundario de `myAccordion`:

```
myAccordion.destroyChildAt(myAccordion.numChildren - 1);
```

### Véase también

[Accordion.createChild\(\)](#)

## Accordion.getChildAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myAccordion.getChildAt(index)
```

### Parámetros

*index* Número de índice de un elemento secundario de accordion. A cada elemento secundario de accordion se le asigna un índice basado en cero en el orden en que se creó.

### Valor devuelto

Referencia a la instancia de UIObject en el índice especificado.

### Descripción

Método; devuelve una referencia al elemento secundario en el índice especificado. A cada elemento secundario de accordion se le da un número de índice para su posición. Este número de índice está basado en cero, de modo que el primer elemento secundario es 0, el segundo elemento secundario es 1 y así sucesivamente.

### Ejemplo

El código siguiente obtiene una referencia al último elemento secundario de myAccordion:

```
var lastChild:UIObject = myAccordion.getChildAt(myAccordion.numChildren - 1);
```

## Accordion.numChildren

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myAccordion.numChildren
```

### Descripción

Propiedad (heredada de View); indica el número de elementos secundarios (UIObject secundarios) en una instancia de accordion. Los encabezados no se cuentan como elementos secundarios.

A cada elemento secundario de accordion se le da un número de índice para su posición. Este número de índice está basado en cero, de modo que el primer elemento secundario es 0, el segundo elemento secundario es 1 y así sucesivamente. El código `myAccordion.numChild - 1` siempre hace referencia al último elemento secundario que se ha añadido a un accordion. Por ejemplo, si hay 7 elementos secundarios en un accordion, el último tendrá el índice 6. La propiedad `numChildren` no está basada en cero, de modo que el valor de `myAccordion.numChildren` será 7. El resultado de `7 - 1` es 6, que es el número de índice del último elemento secundario.

### Ejemplo

En el ejemplo siguiente se selecciona el último elemento secundario:

```
myAccordion.selectedIndex = myAccordion.numChildren - 1;
```

## Accordion.selectedChild

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myAccordion.selectedChild
```

### Descripción

Propiedad; el elemento secundario seleccionado si existen uno o más elementos secundarios; no definido si no existen elementos secundarios. Esta propiedad es de tipo `UIObject` o de tipo no definido (`undefined`).

Si el accordion tiene elementos secundarios, el código `myAccordion.selectedChild` es equivalente al código `myAccordion.getChildAt(myAccordion.selectedIndex)`.

La definición de esta propiedad en un elemento secundario hace que el accordion inicie la animación de transición para visualizar el elemento secundario especificado.

Si se cambia el valor de `selectedChild` también se cambia el valor de `selectedIndex`.

El valor predeterminado es `myAccordion.getChildAt(0)` si el accordion tiene elementos secundarios. Si el accordion no tiene elementos secundarios, el valor predeterminado no está definido.

### Ejemplo

El ejemplo siguiente obtiene la etiqueta de la vista secundaria seleccionada:

```
var selectedLabel = myAccordion.selectedChild.label;
```

El ejemplo siguiente define el formulario de pago que será la vista del elemento secundario seleccionado:

```
myAccordion.selectedChild = myAccordion.payment;
```

### Véase también

[Accordion.selectedIndex](#)

## Accordion.selectedIndex

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myAccordion.selectedIndex
```

### Descripción

Propiedad; el índice basado en cero del elemento secundario seleccionado en un accordion con uno o más elementos secundarios. Para un accordion que no tiene vistas secundarias, el único valor válido es undefined.

A cada elemento secundario de accordion se le da un número de índice para su posición. Este número de índice está basado en cero, de modo que el primer elemento secundario es 0, el segundo elemento secundario es 1 y así sucesivamente. Los valores válidos de `selectedIndex` son 0, 1, 2... , n - 1, donde n es el número de elementos secundarios.

La definición de esta propiedad en un elemento secundario hace que el accordion inicie la animación de transición para visualizar el elemento secundario especificado.

Si se cambia el valor de `selectedIndex` también se cambia el valor de `selectedChild`.

### Ejemplo

El ejemplo siguiente recuerda el índice del elemento secundario seleccionado:

```
var oldSelectedIndex = myAccordion.selectedIndex;
```

En el ejemplo siguiente se selecciona el último elemento secundario:

```
myAccordion.selectedIndex = myAccordion.numChildren - 1;
```

### Véase también

[Accordion.selectedChild](#), [Accordion.numChildren](#)

## Componente Alert (sólo Flash Professional)

El componente Alert permite abrir una ventana que presenta botones de mensaje y respuesta. La ventana Alert tiene una barra de título que se puede rellenar con texto, un mensaje que se puede personalizar y botones con etiquetas que se pueden cambiar. Una ventana Alert puede tener cualquier combinación de los siguientes botones: Yes, No, OK y Cancel. Las etiquetas de texto se pueden cambiar en los botones utilizando las propiedades siguientes: [Alert.yesLabel](#), [Alert.noLabel](#), [Alert.okLabel](#) y [Alert.cancelLabel](#). No se puede cambiar el orden de los botones en una ventana Alert; el orden de los botones es siempre OK, Yes, No, Cancel.

Para abrir una ventana Alert se debe llamar al método [Alert.show\(\)](#). Para llamar al método correctamente, el componente Alert debe estar en la biblioteca. Debe arrastrar el componente Alert del panel Components al escenario y, a continuación, eliminarlo del escenario. De esta manera se añade el componente a la biblioteca, aunque éste no se vuelve visible en el documento.

La vista previa dinámica para el componente Alert es una ventana vacía.

Los lectores de pantalla pueden acceder al texto y a los botones de una ventana Alert. Cuando se añade el componente Alert a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.AlertAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda.

## Utilización del componente Alert (sólo para Flash Professional)

El componente Alert se puede utilizar siempre que se desee anunciar algo al usuario. Puede por ejemplo abrir una ventana Alert cuando un usuario no rellene correctamente un formulario, cuando un valor alcance un precio determinado o cuando un usuario salga de una aplicación sin guardar la sesión.

### Parámetros de Alert

No hay parámetros de edición para el componente Alert. Debe llamar al método `ActionScript Alert.show()` para abrir una ventana Alert. Puede utilizar otras propiedades `ActionScript` para modificar la ventana Alert de una aplicación. Para más información, consulte “Clase Alert (sólo para Flash Professional)” en la página 69.

## Creación de aplicaciones con el componente Alert

El procedimiento siguiente explica cómo añadir un componente Alert a una aplicación durante la edición. En este ejemplo, el componente Alert aparece cuando un valor alcanza un precio determinado.

**Para crear una aplicación con el componente Alert, siga este procedimiento:**

- 1 En el panel Componentes haga doble clic en el componente Alert para añadirlo al escenario.
- 2 Presione la tecla Retroceso (Windows) o Suprimir (Macintosh) para suprimir el componente del escenario.

De esta manera, se añade el componente a la biblioteca, aunque éste no se vuelve visible en la aplicación.

- 3 En el panel Acciones, especifique el código siguiente en el Fotograma 1 de la línea de tiempo para definir un controlador de eventos para el evento `click`:

```
import mx.controls.Alert
myClickHandler = function (evt){
    if (evt.detail == Alert.OK){
        trace("start stock app");
        // startStockApplication();
    }
}
Alert.show("Launch Stock Application?", "Stock Price Alert", Alert.OK |
Alert.CANCEL, this, myClickHandler, "stockIcon", Alert.OK);
```

Este código crea una ventana Alert con los botones OK y Cancel. Cuando se presiona uno de estos botones, se llama a la función `myClickHandler`. Sin embargo, cuando se presiona el botón OK, se llama al método `startStockApplication()`.

- 4 Control > Probar película.

## Personalización del componente Alert (sólo para Flash Professional)

La propia posición Alert en el centro del componente que se pasó como su parámetro *parent*. El elemento principal debe ser un `UIComponent`. Si es un clip de película, puede registrar el clip como `mx.core.View` de forma que herede de `UIComponent`.

La ventana Alert se expande automáticamente de forma horizontal para que quepa el texto del mensaje o cualquiera de los botones visualizados. Si desea visualizar grandes cantidades de texto, incluya saltos de línea en el texto.

El componente Alert no responde al método `setSize()`.

### Utilización de estilos con el componente Alert

Es posible definir propiedades de estilo para cambiar el aspecto de un componente Alert. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

Un componente Alert admite los siguientes estilos de Halo:

| Estilo                               | Descripción                                                                                                                                  |
|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>themeColor</code>              | Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son “haloGreen”, “haloBlue” y “haloOrange”. |
| <code>color</code>                   | Texto de la etiqueta de un componente.                                                                                                       |
| <code>disabledColor</code>           | Color desactivado para el texto.                                                                                                             |
| <code>fontFamily</code>              | Nombre de la fuente del texto.                                                                                                               |
| <code>fontSize</code>                | Tamaño de la fuente en puntos.                                                                                                               |
| <code>fontStyle</code>               | Estilo de la fuente; puede ser “normal” o “italic”.                                                                                          |
| <code>fontWeight</code>              | Grosor de la fuente; puede ser “normal” o “bold”.                                                                                            |
| <code>textDecoration</code>          | Decoración del texto; puede ser “none” o “underline”.                                                                                        |
| <code>buttonStyleDeclaration</code>  | Clase (static) <code>CSSStyleDeclaration</code> para los estilos de texto del botón.                                                         |
| <code>messageStyleDeclaration</code> | Clase (static) <code>CSSStyleDeclaration</code> para el texto del mensaje, el borde y los estilos de fondo.                                  |
| <code>titleStyleDeclaration</code>   | Clase (static) <code>CSSStyleDeclaration</code> para los estilos de texto del título.                                                        |

### Utilización de aspectos con el componente Alert

El componente Alert utiliza los aspectos de `Window` para representar los estados visuales de los botones y la barra de título. Para aplicar un aspecto a los botones y la barra de título durante la edición, modifique los símbolos del aspecto en la carpeta de estados de aspectos Flash UI Components 2/Themes/MMDefault/Window Assets de la biblioteca de uno de los archivos FLA de temas. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).



Hay un código ActionScript en la clase RectBorder.as que el componente Alert utiliza para dibujar sus bordes. Puede utilizar estilos RectBorder para modificar un componente Alert como se indica a continuación:

```
var myAlert = Alert.show("This is a test of errors", "Error", Alert.OK |  
    Alert.CANCEL, this);  
myAlert.setStyle("borderStyle", "inset");
```

Para más información sobre estilos RectBorder, consulte [“Utilización de aspectos con el componente List” en la página 314](#).

Un componente Alert utiliza las siguientes propiedades de aspecto para aplicar dinámicamente un aspecto a los botones y a la barra de título.

| Propiedad       | Descripción                                   | Valor predeterminado |
|-----------------|-----------------------------------------------|----------------------|
| buttonUp        | Estado Arriba del botón.                      | ButtonSkin           |
| buttonDown      | Estado Presionado del botón.                  | ButtonSkin           |
| buttonOver      | Estado del botón al pasar el cursor sobre él. | ButtonSkin           |
| titleBackground | Barra de título de la ventana.                | TitleBackground      |

## Clase Alert (sólo para Flash Professional)

**Herencia** UIObject > UIComponent > View > ScrollView > Window > Alert

**Nombre de clase de ActionScript** mx.controls.Alert

Para utilizar el componente Alert, se debe arrastrar un componente Alert al escenario y eliminarlo, de modo que el componente esté en la biblioteca del documento pero no visible en la aplicación. A continuación, se llama a `Alert.show()` para abrir una ventana Alert. Puede pasar a `Alert.show()` parámetros que añaden un mensaje, una barra de título y botones a la ventana Alert.

Puesto que ActionScript es asíncrono, el componente Alert no produce bloqueos, de modo que las líneas de código ActionScript después de la llamada a `Alert.show()` se ejecutan inmediatamente. Debe añadir detectores para controlar los eventos `click` que se difunden cuando un usuario presiona un botón y, a continuación, seguir con el código después de que se haya difundido el evento.

**Nota:** en entornos operativos donde se produzcan bloqueos (por ejemplo, Microsoft Windows), no se devolverá una llamada a `Alert.show()` hasta que el usuario haya llevado a cabo una acción, como pulsar un botón.

## Resumen de métodos de la clase Alert

| Evento                    | Descripción                                       |
|---------------------------|---------------------------------------------------|
| <code>Alert.show()</code> | Crea una ventana Alert con parámetros opcionales. |

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

## Resumen de propiedades de la clase Alert

| Propiedad                       | Descripción                                                                  |
|---------------------------------|------------------------------------------------------------------------------|
| <code>Alert.buttonHeight</code> | Altura de cada botón, expresada en píxeles. El valor predeterminado es 22.   |
| <code>Alert.buttonWidth</code>  | Anchura de cada botón, expresada en píxeles. El valor predeterminado es 100. |
| <code>Alert.cancelLabel</code>  | Texto de la etiqueta del botón Cancelar.                                     |
| <code>Alert.noLabel</code>      | Texto de la etiqueta del botón No.                                           |
| <code>Alert.okLabel</code>      | Texto de la etiqueta del botón Aceptar.                                      |
| <code>Alert.yesLabel</code>     | Texto de la etiqueta del botón Sí.                                           |

Hereda todas las propiedades de [UIObject](#) y [UIComponent](#).

## Resumen de eventos de la clase Alert

| Evento                   | Descripción                                                     |
|--------------------------|-----------------------------------------------------------------|
| <code>Alert.click</code> | Se difunde cuando se hace clic en un botón de la ventana Alert. |

Hereda todos los eventos de [UIObject](#) y [UIComponent](#).

## Alert.buttonHeight

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`Alert.buttonHeight`

### Descripción

Propiedad (clase); una propiedad de clase (static) que cambia la altura de los botones.

### Véase también

[Alert.buttonWidth](#)

## Alert.buttonWidth

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
Alert.buttonWidth
```

### Descripción

Propiedad (clase); una propiedad de clase (static) que cambia la anchura de los botones.

### Véase también

[Alert.buttonHeight](#)

## Alert.click

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
clickHandler = function(eventObject){  
    // introducir aquí el código  
}  
Alert.show(message[, title[, flags[, parent[, clickHandler[, icon[,  
    defaultButton]]]]]])
```

### Descripción

Evento; se difunde al detector registrado cuando se hace clic en el botón OK, Yes, No o Cancel.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. El componente Alert distribuye un evento `click` cuando se hace clic en uno de sus botones y el evento se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método [Alert.show\(\)](#) y pase el nombre del controlador como parámetro. Cuando se hace clic en un botón de la ventana Alert, se llamará al detector.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `Alert.click` tiene una propiedad `detail` adicional cuyo valor es uno de los siguientes, dependiendo de en qué botón se ha hecho clic: `Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente se define un controlador denominado `myClickHandler` y se pasa al método `Alert.show()` como el quinto parámetro. El objeto de evento se captura mediante el controlador `myClickHandler` en el parámetro `evt`. A continuación, la propiedad `detail` del objeto de evento se utiliza dentro de una sentencia `trace` para enviar el nombre del botón en el que se hizo clic (`Alert.OK` o `Alert.CANCEL`) al panel Salida, como se indica a continuación:

```
myClickHandler = function(evt){
    if(evt.detail == Alert.OK){
        trace(Alert.okLabel);
    }else if (evt.detail == Alert.CANCEL){
        trace(Alert.cancelLabel);
    }
}
Alert.show("This is a test of errors", "Error", Alert.OK | Alert.CANCEL, this,
myClickHandler);
```

## Alert.cancelLabel

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
Alert.cancelLabel
```

### Descripción

Propiedad (clase); una propiedad de clase (static) que indica el texto de la etiqueta del botón Cancel.

### Ejemplo

En el ejemplo siguiente, se define la etiqueta del botón Cancel como “cancellation”:

```
Alert.cancelLabel = "cancellation";
```

## Alert.noLabel

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
Alert.noLabel
```

### Descripción

Propiedad (clase); una propiedad de clase (static) que indica el texto de la etiqueta del botón No.

## Ejemplo

En el ejemplo siguiente, se define la etiqueta del botón No como “nyet”:

```
Alert.noLabel = "nyet";
```

## Alert.okLabel

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
Alert.okLabel
```

### Descripción

Propiedad (clase); una propiedad de clase (static) que indica el texto de la etiqueta del botón OK.

## Ejemplo

En el ejemplo siguiente, se define la etiqueta del botón OK como “okay”:

```
Alert.okLabel = "okay";
```

## Alert.show()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
Alert.show(message[, title[, flags[, parent[, clickHandler[, icon[,  
defaultButton]]]]]])
```

### Parámetros

*message* Mensaje que se mostrará.

*title* Texto de la barra de título del componente Alert. Este parámetro es opcional. Si no se especifica el parámetro *title*, la barra de título estará vacía.

*flags* Parámetro opcional que indica el botón o botones que se mostrarán en la ventana Alert. El valor predeterminado es `Alert.OK`, que muestra un botón “OK”. Cuando utilice más de un valor, separe los valores con el carácter `|`. Este valor puede ser uno o más de los siguientes:

- `Alert.OK`
- `Alert.CANCEL`
- `Alert.YES`
- `Alert.NO`

También puede utilizar `Alert.NONMODAL` para indicar que la ventana `Alert` no es modal. Una ventana que no es modal permite al usuario interactuar con otras ventanas de la aplicación.

*parent* Ventana principal del componente `Alert`. La ventana `Alert` se centra en la ventana principal. Utilice el valor `null` o `undefined` para especificar la línea de tiempo `_root`. La ventana principal debe heredar de la clase `UIComponent`. Puede registrar la ventana principal con `mx.core.View` para que herede de `UIComponent`. Este parámetro es opcional.

*clickHandler* Controlador para los eventos `click` difundidos cuando se hace clic en los botones. Además de las propiedades del objeto del evento `click` estándar hay una propiedad `detail`, que contiene el valor del indicador de botón en el que se ha hecho clic (`Alert.OK`, `Alert.CANCEL`, `Alert.YES`, `Alert.NO`). Este controlador puede ser una función o un objeto. Para más información, consulte el [Capítulo 2, “Utilización de detectores de eventos de componente”](#), en la página 26.

*icon* Cadena que es el identificador de vinculación de un símbolo de la biblioteca que se utilizará como un icono que se mostrará a la izquierda del texto. Este parámetro es opcional.

*defaultButton* Indica el botón en el que se hace clic cuando un usuario presiona Intro (Windows) o Retorno (Macintosh). Este parámetro puede tener uno de estos valores:

- `Alert.OK`
- `Alert.CANCEL`
- `Alert.YES`
- `Alert.NO`

### Valor devuelto

La instancia de la clase `Alert` que crea el usuario.

### Descripción

Método (class); método de clase (static) que muestra una ventana `Alert` con un mensaje, un título opcional, botones opcionales y un icono opcional. El título del componente `Alert` aparece en la parte superior de la ventana y se alinea a la izquierda. El icono aparece a la izquierda del texto del mensaje. El botón aparece centrado debajo del texto del mensaje y el icono.

### Ejemplo

El código siguiente es un sencillo ejemplo de una ventana `Alert` modal con un botón OK:

```
Alert.show("Hello, world!");
```

En el código siguiente, se define un controlador de acciones del ratón que envía un mensaje al panel Salida en el que se indica el botón en el que se ha hecho clic:

```
myClickHandler = function(evt){
    trace (evt.detail + "was clicked");
}
Alert.show("This is a test of errors", "Error", Alert.OK | Alert.CANCEL, this,
    myClickHandler);
```

**Nota:** la propiedad de detalle del objeto de evento devuelve un número para representar cada botón. El botón OK es el 4, el botón Cancel es el 8, el botón Yes es el 1 y el botón No es el 2.

## Alert.yesLabel

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
Alert.yesLabel
```

### Descripción

Propiedad (clase); una propiedad de clase (static) que indica el texto de la etiqueta del botón Yes.

### Ejemplo

En el ejemplo siguiente, se define la etiqueta del botón OK como “da”:

```
Alert.yesLabel = "da";
```

## Componente Button

El componente Button es un botón rectangular de la interfaz de usuario que puede cambiarse de tamaño. También se le puede añadir un icono personalizado y cambiar su comportamiento de botón de comando a conmutador. El conmutador permanece presionado cuando se hace clic en él y recupera su estado original al repetirse el clic.

Un botón puede estar activado o desactivado en una aplicación. Si está desactivado, el botón no recibe la entrada del ratón ni del teclado. Si está activado, el botón se selecciona cuando el usuario hace clic sobre él o presiona el tabulador hasta su posición. Cuando una instancia de Button está seleccionada, es posible utilizar las siguientes teclas para controlarla:

| Tecla                  | Descripción                                                             |
|------------------------|-------------------------------------------------------------------------|
| Mayúsculas + Tabulador | Desplaza la selección al objeto anterior.                               |
| Barra espaciadora      | Presiona o libera el componente y activa el evento <code>click</code> . |
| Tabulador              | Desplaza la selección al objeto siguiente.                              |

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase FocusManager” en la página 292](#).

La vista previa dinámica de cada instancia de Button refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes. Sin embargo, en la vista previa dinámica los iconos personalizados se representan en el escenario con un cuadrado gris.

Cuando se añade el componente Button a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad del componente Button:

```
mx.accessibility.ButtonAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda.

## Utilización del componente Button

Un botón es una parte fundamental de cualquier formulario o aplicación Web. Utilice botones siempre que necesite que el usuario inicie un evento. Por ejemplo, la mayoría de los formularios contienen el botón “Enviar” y las presentaciones suelen llevar los botones “Anterior” y “Siguiente”.

Para añadir un icono a un botón, es necesario seleccionar o crear un clip de película o un símbolo gráfico para utilizarlo como icono. El símbolo debe registrarse en la posición 0, 0 para que el botón tenga el diseño adecuado. Seleccione el símbolo del icono en el panel Biblioteca, abra el cuadro de diálogo Vinculación desde el menú de opciones e introduzca un identificador de vinculación. Éste es el valor que debe introducir como parámetro icon en el inspector de propiedades o en el panel Inspector de componentes. También puede introducirlo en la propiedad [Button.icon](#) de ActionScript.

**Nota:** si el icono es más grande que el botón, sobresaldrá por los bordes de éste.

## Parámetros de Button

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Button en el inspector de propiedades o en el panel Inspector de componentes:

**label** define el valor del texto del botón; el valor predeterminado es Button.

**icon** añade un icono personalizado al botón. El valor es el identificador de vinculación de un clip de película o símbolo gráfico de la biblioteca; no hay valor predeterminado.

**toggle** convierte el botón en un conmutador. Si el valor es true, el botón permanece en el estado Presionado cuando está presionado y recupera el estado Arriba cuando se vuelve a presionar. Si el valor es false, el botón se comporta como un botón de comando normal; el valor predeterminado es false.

**selected** especifica si el botón está presionado (true) o no (false) cuando el parámetro toggle es true. El valor predeterminado es false.

**labelPlacement** orienta el texto de la etiqueta del botón con respecto al icono. Este parámetro puede tener uno de estos cuatro valores: left, right, top o bottom; el valor predeterminado es right. Para más información, consulte [Button.labelPlacement](#).

Puede escribir código ActionScript para controlar éstas y otras opciones de componentes Button con sus propiedades, métodos y eventos. Para más información, consulte [Clase Button](#).

## Creación de aplicaciones con el componente Button

El siguiente procedimiento explica cómo añadir un componente Button a una aplicación durante la edición. En este ejemplo se trata de un botón de Ayuda con un icono personalizado que abre el sistema de Ayuda cuando el usuario lo presiona.



**Para crear una aplicación con el componente Button, siga este procedimiento:**

- 1 Arrastre un componente Button desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia **helpBtn**.
- 3 En el inspector de propiedades, siga este procedimiento:
  - Introduzca **Help** en el parámetro label.
  - Introduzca **HelpIcon** en el parámetro icon.  
Para utilizar un icono, en la biblioteca debe haber un clip de película o un símbolo gráfico con un identificador de vinculación para emplearlo como parámetro icon. En este ejemplo, el identificador de vinculación es HelpIcon.
  - Defina la propiedad **toggle** como true (verdadera).
- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
clippyListener = new Object();
clippyListener.click = function (evt){
    clippyHelper.enabled = evt.target.selected;
}
helpBtn.addEventListener("click", clippyListener);
```

La última línea de código añade un controlador de eventos `click` a la instancia `helpBtn`. El controlador activa y desactiva la instancia `clippyHelper`, que podría ser un panel Ayuda de alguna clase.

## Personalización del componente Button

El componente Button puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)) o cualquier método o propiedad aplicable de la clase Button (véase [Clase Button](#)). Cuando se modifica el tamaño del botón, no cambia el tamaño del icono ni de la etiqueta.

El recuadro de delimitación de una instancia de Button es invisible y designa el área activa de la instancia. Si se aumenta el tamaño de la instancia, también aumenta el tamaño del área activa. Si el recuadro de delimitación es demasiado pequeño para que encaje la etiqueta, ésta se recorta para ajustarse al espacio disponible.

Si el icono es más grande que el botón, sobresaldrá por los bordes de éste.

## Utilización de estilos con el componente Button

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de Button. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

El componente Button admite los siguientes estilos de Halo:

| Estilo        | Descripción                                                                                                                                  |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| themeColor    | Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son "haloGreen", "haloBlue" y "haloOrange". |
| color         | Texto de la etiqueta de un componente.                                                                                                       |
| disabledColor | Color desactivado para el texto.                                                                                                             |
| fontFamily    | Nombre de la fuente del texto.                                                                                                               |
| fontSize      | Tamaño de la fuente en puntos.                                                                                                               |
| fontStyle     | Estilo de la fuente: puede ser "normal" o "italic".                                                                                          |
| fontWeight    | Grosor de la fuente: puede ser "normal" o "bold".                                                                                            |

## Utilización de aspectos con el componente Button

El componente Button utiliza la interfaz API de dibujo de ActionScript para dibujar estados de botón. Para aplicar aspectos al componente Button durante la edición, modifique el código ActionScript del archivo ButtonSkin.as que se encuentra en la carpeta First Run\Classes\mx\skins\halo.

Si utiliza el método `UIObject.createClassObject()` para crear dinámicamente una instancia del componente Button (en tiempo de ejecución), podrá asignarle aspectos dinámicamente. Para asignar aspectos a un componente en tiempo de ejecución, defina las propiedades de aspecto en el parámetro `initObject` que pasa al método `createClassObject()`. Estas propiedades de aspecto definen los nombres de los símbolos que se utilizarán como estados del botón, ya sea con o sin icono.

Si define el parámetro `icon` durante la edición o la propiedad `icon` de ActionScript en tiempo de ejecución, se asignará el mismo identificador de vinculación a tres estados del icono: `falseUpIcon`, `falseDownIcon` y `trueUpIcon`. Si desea designar un icono exclusivo para alguno de los ocho estados del icono (por ejemplo, para que aparezca un icono diferente cuando el usuario presione el botón), debe definir las propiedades del parámetro `initObject` que pasa al método `createClassObject()`.

El código siguiente crea un objeto denominado `initObject` que se utilizará como parámetro `initObject` y define las propiedades de aspecto en identificadores de vinculación de símbolos nuevos. La última línea de código llama al método `createClassObject()` para crear una instancia nueva de la clase Button con las propiedades pasadas en el parámetro `initObject`:

```
var initObject = new Object();
initObject.falseUpIcon = "MyFalseUpIcon";
initObject.falseDownIcon = "MyFalseDownIcon";
initObject.trueUpIcon = "MyTrueUpIcon";
createClassObject(mx.controls.Button, "ButtonInstance", 0, initObject);
```

Para más información, consulte ["Aplicación de aspectos a los componentes" en la página 41](#) y `UIObject.createClassObject()`.

Si un botón está activado, muestra su estado Sobre cuando el puntero pasa sobre él. El botón recibe la selección de entrada y muestra el estado Presionado cuando se hace clic en él. El botón recupera el estado Sobre cuando se suelta el ratón. Si el puntero se retira del botón estando el ratón presionado, el botón recupera su estado original y conserva la selección de entrada. Si el parámetro toggle está definido en true, el estado del botón no cambia hasta que el ratón se suelta estando sobre él.

Si el botón está desactivado, muestra el estado Desactivado sea cual sea la acción del usuario.

El componente Button utiliza las siguientes propiedades de aspecto:

| Propiedad         | Descripción                                                                   |
|-------------------|-------------------------------------------------------------------------------|
| falseUpSkin       | Estado Arriba. El valor predeterminado es RectBorder.                         |
| falseDownSkin     | Estado Presionado. El valor predeterminado es RectBorder.                     |
| falseOverSkin     | Estado Sobre. El valor predeterminado es RectBorder.                          |
| falseDisabledSkin | Estado Desactivado. El valor predeterminado es RectBorder.                    |
| trueUpSkin        | Estado Conmutado. El valor predeterminado es RectBorder.                      |
| trueDownSkin      | Estado Presionado conmutado. El valor predeterminado es RectBorder.           |
| trueOverSkin      | Estado Sobre conmutado. El valor predeterminado es RectBorder.                |
| trueDisabledSkin  | Estado Desactivado conmutado. El valor predeterminado es RectBorder.          |
| falseUpIcon       | Estado Arriba del icono. El valor predeterminado es undefined.                |
| falseDownIcon     | Estado Presionado del icono. El valor predeterminado es undefined.            |
| falseOverIcon     | Estado Sobre del icono. El valor predeterminado es undefined.                 |
| falseDisabledIcon | Estado Desactivado del icono. El valor predeterminado es undefined.           |
| trueUpIcon        | Estado Conmutado del icono. El valor predeterminado es undefined.             |
| trueOverIcon      | Estado Sobre conmutado del icono. El valor predeterminado es undefined.       |
| trueDownIcon      | Estado Presionado conmutado del icono. El valor predeterminado es undefined.  |
| trueDisabledIcon  | Estado Desactivado conmutado del icono. El valor predeterminado es undefined. |

## Clase Button

**Herencia** UIObject > UIComponent > SimpleButton > Button

**Nombre de clase de ActionScript** mx.controls.Button

Las propiedades de la clase Button permiten añadir un icono a un botón, crear una etiqueta de texto o indicar si el botón actúa como botón de comando o como conmutador en tiempo de ejecución.

Si una propiedad de la clase Button se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

El componente Button utiliza FocusManager para sustituir el rectángulo de selección predeterminado de Flash Player por un rectángulo de selección personalizado con esquinas redondeadas. Para más información, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.Button.version);
```

**Nota:** el código siguiente devuelve undefined: `trace(myButtonInstance.version);`

La clase del componente Button es distinta del objeto Button incorporado de ActionScript.

## Resumen de métodos de la clase Button

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

## Resumen de propiedades de la clase Button

| Método                                                  | Descripción                                                                                                                                              |
|---------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">SimpleButton.emphasized</a>                 | Indica si el botón tiene el aspecto de un botón de comando predeterminado.                                                                               |
| <a href="#">SimpleButton.emphasizedStyleDeclaration</a> | Declaración de estilos cuando la propiedad <code>emphasized</code> está definida en <code>true</code> .                                                  |
| <a href="#">Button.icon</a>                             | Especifica un icono para una instancia del botón.                                                                                                        |
| <a href="#">Button.label</a>                            | Especifica el texto que aparece dentro de un botón.                                                                                                      |
| <a href="#">Button.labelPlacement</a>                   | Especifica la orientación del texto de la etiqueta con respecto a un icono.                                                                              |
| <a href="#">Button.selected</a>                         | Cuando la propiedad <code>toggle</code> es <code>true</code> , especifica si el botón está presionado ( <code>true</code> ) o no ( <code>false</code> ). |
| <a href="#">Button.toggle</a>                           | Indica si el botón se comporta como un conmutador.                                                                                                       |

Hereda todas las propiedades de [UIObject](#) y [UIComponent](#).

## Resumen de eventos de la clase Button

| Método                       | Descripción                                                                                                     |
|------------------------------|-----------------------------------------------------------------------------------------------------------------|
| <a href="#">Button.click</a> | Se difunde cuando se presiona el ratón sobre una instancia del botón o cuando se presiona la barra espaciadora. |

Hereda todos los eventos de [UIObject](#) y [UIComponent](#).

## Button.click

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(click){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.click = function(eventObject){  
    ...  
}  
buttonInstance.addEventListener("click", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta el botón del ratón) en un botón o cuando éste está seleccionado y se presiona la barra espaciadora.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `Button`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia del componente `Button` `myButtonComponent`, envía “\_level0.myButtonComponent” al panel Salida:

```
on(click){  
    trace(this);  
}
```

Observe que `this` se comporta de forma diferente cuando se utiliza dentro de un controlador `on()` asociado con un símbolo de botón normal de Flash. Cuando `this` se utiliza dentro de un controlador `on()` asociado con un símbolo de botón, hace referencia a la línea de tiempo que contiene el botón. Por ejemplo, el código siguiente, asociado con la instancia `myButton` del símbolo de botón, envía “\_level0” al panel Salida:

```
on(release) {  
    trace(this);  
}
```

**Nota:** el objeto incorporado `Button` de `ActionScript` no tiene eventos `click`; el evento más próximo es `release`.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*buttonInstance*) distribuye un evento (en este caso, *click*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. El objeto *Event* tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método *addEventListener()* (véase *UIEventDispatcher.addEventListener()*) de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 602.

### Ejemplo

En este ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se hace clic en un botón denominado *buttonInstance*. La primera línea del código etiqueta el botón. La segunda línea especifica que el botón actúe como conmutador. La tercera línea crea un objeto detector denominado *form*. La cuarta línea define una función para el evento *click* en el objeto detector. Dentro de la función hay una acción *trace* que utiliza el objeto *Event* pasado automáticamente a la función (en este ejemplo *eventObj*) para generar un mensaje. La propiedad *target* de un objeto *Event* es el componente que ha generado el evento (en este ejemplo *buttonInstance*). A la propiedad *Button.selected* se accede desde la propiedad *target* del objeto *Event*. La última línea llama al método *addEventListener()* desde *buttonInstance* y pasa como parámetros el evento *click* y el objeto detector *form*, como en el ejemplo siguiente:

```
buttonInstance.label = "Click Test"
buttonInstance.toggle = true;
form = new Object();
form.click = function(eventObj){
    trace("The selected property has changed to " + eventObj.target.selected);
}
buttonInstance.addEventListener("click", form);
```

El código siguiente también envía un mensaje al panel Salida cuando se hace clic en *buttonInstance*. El controlador *on()* debe asociarse directamente con *buttonInstance*, como en el ejemplo siguiente:

```
on(click){
    trace("button component was clicked");
}
```

### Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## SimpleButton.emphasized

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*buttonInstance.emphasized*

### Descripción

Propiedad; indica si el botón está en estado `emphasized` (`true`) o no (`false`). El estado `emphasized` equivale al aspecto de un botón de comando predeterminado. En general, utilice la propiedad `FocusManager.defaultPushButton` en lugar de definir directamente la propiedad `emphasized`. El valor predeterminado es `false`.

La propiedad `emphasized` es una propiedad estática de la clase `SimpleButton`. Por tanto, debe acceder a ella directamente desde `SimpleButton`, como en el ejemplo siguiente:

```
SimpleButton.emphasizedStyleDeclaration = "foo";
```

Si no está utilizando `FocusManager.defaultPushButton`, es posible que sólo desee definir un botón en el estado `emphasized`, o utilizar el estado `emphasized` para cambiar el texto de un color a otro. En el siguiente ejemplo se define la propiedad `emphasized` de la instancia del botón `myButton`:

```
_global.styles.foo = new CSSStyleDeclaration();
_global.styles.foo.color = 0xFF0000;
SimpleButton.emphasizedStyleDeclaration = "foo";
myButton.emphasized = true;
```

### Véase también

[SimpleButton.emphasizedStyleDeclaration](#)

## SimpleButton.emphasizedStyleDeclaration

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*buttonInstance.emphasizedStyleDeclaration*

### Descripción

Propiedad; cadena que indica la declaración de estilos que da formato a un botón cuando la propiedad `emphasized` se define en `true`.

### Véase también

[SimpleButton.emphasized](#)

## Button.icon

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*buttonInstance.icon*

### Descripción

Propiedad; cadena que especifica el identificador de vinculación de un símbolo de la biblioteca que se utilizará como icono en una instancia del botón. El icono puede ser un símbolo de clip de película o un símbolo gráfico con un punto de registro en la esquina superior izquierda. Si el icono es demasiado grande para encajarlo en el botón, deberá modificar el tamaño del botón; ni el botón ni el icono cambian de tamaño automáticamente. Si el icono es más grande que el botón, sobresaldrá por los bordes del botón.

Para crear un icono personalizado, cree un clip de película o un símbolo gráfico. Seleccione el símbolo en el escenario en el modo de edición de símbolos e introduzca 0 en los cuadros X e Y del inspector de propiedades. En el panel Biblioteca, seleccione el clip de película y elija Vinculación en el menú Opciones. Seleccione Exportar para ActionScript e introduzca un identificador en el cuadro de texto Identificador.

El valor predeterminado es una cadena vacía (" ") que indica que no hay icono.

Utilice la propiedad `labelPlacement` para definir la posición del icono con respecto al botón.

### Ejemplo

El código siguiente asigna el clip de película del panel Biblioteca que tiene el identificador de vinculación `happiness` a la instancia `Button` como icono:

```
myButton.icon = "happiness"
```

### Véase también

[Button.labelPlacement](#)

## Button.label

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*buttonInstance.label*



## Descripción

Propiedad; especifica la etiqueta de texto para una instancia del botón. De forma predeterminada, la etiqueta aparece centrada en el botón. Cuando se llama a este método, se sustituye el parámetro de edición de etiquetas especificado en el inspector de propiedades o en el panel Inspector de componentes. El valor predeterminado es "Button".

## Ejemplo

El código siguiente define la etiqueta en "Remove from list":

```
buttonInstance.label = "Remove from list";
```

## Véase también

[Button.labelPlacement](#)

## Button.labelPlacement

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
buttonInstance.labelPlacement
```

## Descripción

Propiedad; define la posición de la etiqueta con respecto al icono. El valor predeterminado es "right". Los siguientes son los cuatro valores posibles; el icono y la etiqueta están siempre centrados vertical y horizontalmente en el área de delimitación del botón:

- "right" La etiqueta aparece a la derecha del icono.
- "left" La etiqueta aparece a la izquierda del icono.
- "bottom" La etiqueta aparece debajo del icono.
- "top" La etiqueta aparece encima del icono.

## Ejemplo

El código siguiente coloca la etiqueta a la izquierda del icono. La segunda línea del código envía el valor de la propiedad `labelPlacement` al panel Salida:

```
iconInstance.labelPlacement = "left";  
trace(iconInstance.labelPlacement);
```

## Button.selected

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*buttonInstance.selected*

## Descripción

Propiedad; valor booleano que especifica si un botón está presionado (*true*) o no (*false*). El valor de la propiedad *toggle* debe ser *true* para definir la propiedad *selected* en *true*. Si la propiedad *toggle* es *false*, asignar el valor *true* a la propiedad *selected* no tiene efecto alguno. El valor predeterminado es *false*.

El evento *click* no se activa cuando el valor de la propiedad *selected* cambia con *ActionScript*. Se activa cuando el usuario interactúa con el botón.

## Ejemplo

En el siguiente ejemplo, la propiedad *toggle* está definida en *true* y la propiedad *selected* también está definida en *true*, lo que coloca el botón en estado Presionado. La acción *trace* envía el valor *true* al panel Salida:

```
ButtonInstance.toggle = true; // toggle ha de ser true para poder definir la
propiedad selected
ButtonInstance.selected = true; //muestra el estado Conmutado del botón
trace(ButtonInstance.selected); //acción trace envía el valor true
```

## Véase también

[Button.toggle](#)

## Button.toggle

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*buttonInstance.toggle*

### Descripción

Propiedad; valor booleano que especifica si un botón actúa como conmutador (*true*) o como botón de comando (*false*); el valor predeterminado es *false*. Cuando se presiona un conmutador, permanece en estado Presionado hasta que se vuelve a hacer clic sobre él.

### Ejemplo

El código siguiente define la propiedad *toggle* en *true*, por lo que la instancia *myButton* se comporta como un conmutador:

```
myButton.toggle = true;
```

## API CellRenderer

La interfaz API CellRenderer es un conjunto de propiedades y métodos que los componentes basados en listas (List, DataGrid, Tree y Menu) utilizan para manipular y mostrar el contenido de las celdas personalizadas en cada una de sus filas. Las celdas personalizadas pueden contener un componente creado previamente como, por ejemplo, CheckBox, o cualquier clase que se cree.

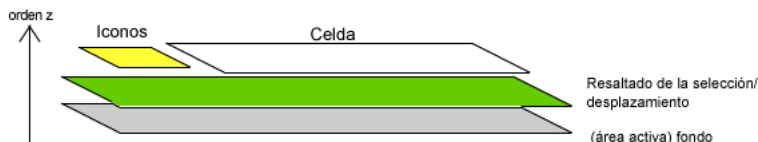
### Clase List

Si va a utilizar la API CellRenderer es importante que conozca bien el funcionamiento de la clase List. Los componentes DataGrid, Tree y Menu son una extensión de la clase List; por tanto, si comprende el funcionamiento de la clase List comprenderá también cómo funcionan estos componentes.

**Nota:** un componente es una clase, pero una clase no es necesariamente un componente.

### Composición de la clase List

La clase List está compuesta por filas. Estas filas muestran los resaltados que aparecen al seleccionar o pasar el cursor sobre elementos, se utilizan como estados de acierto en la selección de filas y son un elemento fundamental en el desplazamiento. Además de los resaltados de selección y de los iconos (como los iconos de nodo y las flechas de ampliación de un componente Tree), una fila se compone de una celda (o, en el caso de DataGrid, de muchas celdas). De forma predeterminada, estas celdas son objetos TextField que implementan la API CellRenderer. No obstante, se puede indicar a una clase List que utilice una clase de componente diferente como celda para cada fila. La única condición es que la clase implemente la API CellRenderer, que utiliza la clase List para comunicarse con la celda.



*Orden de apilamiento de una fila de un componente List o DataGrid.*

**Nota:** si una celda contiene controladores de evento de botón (onPress, etc), puede que el área activa de fondo no reciba la información necesaria para activar los eventos.

### Comportamiento de desplazamiento de la clase List

La clase List utiliza un algoritmo bastante complejo para el desplazamiento. Una lista sólo presenta las filas que puede mostrar de una vez; los elementos que excedan el valor de la propiedad rowCount no obtendrán filas. Al desplazarse por la lista, las filas se mueven hacia arriba o hacia abajo (en función del sentido del desplazamiento). La lista recicla las filas que desaparecen de la vista, las reinicializa y las utiliza para las nuevas filas que van apareciendo en la vista definiendo el nuevo elemento de la vista como valor de la fila antigua y colocando la fila antigua donde el nuevo elemento aparece en la vista.

Debido a este comportamiento, no se puede esperar que una celda se utilice para un único valor. Puesto que las filas se reciclan, el procesador de celdas debe saber cómo debe establecer completamente el estado de la celda cuando se define con un valor nuevo. Por ejemplo, si el procesador de celdas crea un icono para mostrar un elemento, deberá eliminar dicho icono cuando se procese otro elemento con él. Supongamos que el procesador de celdas es un contenedor que se va a ir rellenando con diferentes valores de elemento, y que debe saber cómo pasar completamente de la visualización de un valor a la de otro. De hecho, es necesario que la celda sepa incluso cómo procesar correctamente elementos no definidos, lo cual puede significar eliminar contenido antiguo de la celda.

## Utilización de la interfaz API `CellRenderer`

Debe escribir una clase con cuatro métodos (`CellRenderer.getPreferredHeight()`, `CellRenderer.getPreferredWidth()`, `CellRenderer.setSize()`, `CellRenderer.setValue()`) que el componente basado en la lista utilizará para comunicarse con la celda.

A la celda se le asignan automáticamente dos métodos y una propiedad (`CellRenderer.getCellIndex()`, `CellRenderer.getDataLabel()` y `CellRenderer.listOwner`) para que pueda comunicarse con el componente basado en lista. Por ejemplo, supongamos que una celda contiene una casilla de verificación que hace que se seleccione una fila al hacer clic en la casilla. La celda procesada necesita una referencia al componente basado en lista que la contiene para llamar a la propiedad `selectedIndex` del componente basado en lista. Además, la celda debe saber qué índice de elemento está representando actualmente para poder definir el número correcto en la propiedad `selectedIndex`; para ello, la celda puede utilizar `CellRenderer.listOwner` y `CellRenderer.getCellIndex()`. No es necesario implementar estas interfaces API: la celda las recibe automáticamente cuando se coloca en el componente basado en lista.

## Métodos que se deben implementar para la API `CellRenderer`

Debe escribir una clase con los métodos siguientes para que List, DataGrid, Tree o Menu puedan comunicarse con la celda:

| Nombre                                         | Descripción                                         |
|------------------------------------------------|-----------------------------------------------------|
| <code>CellRenderer.getPreferredHeight()</code> | Devuelve la altura preferida de una celda.          |
| <code>CellRenderer.getPreferredWidth()</code>  | Devuelve la anchura preferida de una celda.         |
| <code>CellRenderer.setSize()</code>            | Define la anchura y la altura de una celda.         |
| <code>CellRenderer.setValue()</code>           | Define el contenido que va a mostrarse en la celda. |

## Métodos proporcionados por la interfaz API CellRenderer

A continuación se indican los métodos que List, DataGrid, Tree y Menu asignan a la celda cuando se crea ésta en el componente. No es necesario implementar estos métodos.

| Nombre                                   | Descripción                                                                                                                |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <code>CellRenderer.getDataLabel()</code> | Devuelve una cadena que contiene el nombre del campo de datos del procesador de celdas.                                    |
| <code>CellRenderer.getCellIndex()</code> | Devuelve un objeto con dos campos, <code>columnIndex</code> y <code>rowIndex</code> , que indican la posición de la celda. |

## Propiedades proporcionadas por la interfaz API CellRenderer

A continuación se indica la propiedad que List, DataGrid, Tree y Menu asignan a la celda cuando se crea en el componente. No es necesario implementar esta propiedad.

| Nombre                              | Descripción                                          |
|-------------------------------------|------------------------------------------------------|
| <code>CellRenderer.listOwner</code> | Referencia al componente List que contiene la celda. |

### CellRenderer.getDataLabel()

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.

#### Sintaxis

```
componentInstance.getDataLabel()
```

#### Parámetros

Ninguno.

#### Valor devuelto

Una cadena.

#### Descripción

Método; devuelve una cadena que contiene el nombre del campo de datos del procesador de celdas.

#### Ejemplo

El código siguiente facilita a la celda averiguar que está presentando el campo de datos "Price". Ahora, la variable `p` es igual a "Price":

```
var p = getDataLabel();
```

## CellRenderer.getCellIndex()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.getCellIndex()
```

### Parámetros

Ninguno.

### Valor devuelto

Un objeto con dos campos: `columnIndex` e `itemIndex`.

### Descripción

Método; devuelve un objeto con dos campos, `columnIndex` e `itemIndex`, que ubican la celda en la cuadrícula. Cada campo es un entero que indica la posición de `column` y la posición de elemento de una celda. El valor de `columnIndex` es siempre 0, excepto para el componente `DataGrid`.

### Ejemplo

En este ejemplo se edita un `dataProvider` de `DataGrid` desde una celda:

```
var index = getCellIndex();  
var colName = listOwner.getColumnAt(index.columnIndex).columnName;  
listOwner.dataProvider.editField(index.itemIndex, colName, someVal);
```

## CellRenderer.getPreferredHeight()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.getPreferredHeight()
```

### Parámetros

Ninguno.

### Valor devuelto

La altura correcta de la celda.

## Descripción

Método; devuelve la altura preferida de una celda. Es especialmente importante para obtener la altura correcta del texto de la celda. Si define un valor más alto que el de la propiedad `rowHeight` del componente, la celda se extenderá sobre las filas hacia arriba y hacia abajo.

## Ejemplo

En este ejemplo se devuelve el valor 20, lo cual indica que la celda debe tener una altura de 20 píxeles.

```
function getPreferredHeight(Void) :Number
{
    return 20;
}
```

## CellRenderer.getPreferredWidth()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*componentInstance*.getPreferredWidth()

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; devuelve la anchura preferida de una celda. Si especifica una anchura superior a la del componente, la celda quedará cortada.

## Ejemplo

En este ejemplo se devuelve el valor 3, lo cual indica que el tamaño de la celda tiene que ser tres veces mayor que la longitud de la cadena que presenta.

```
function getPreferredHeight(Void) : Number
{
    return myString.length*3;
}
```

## CellRenderer.listOwner

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*componentInstance.listOwner*

### Descripción

Propiedad; referencia a la lista propietaria de la celda. Esta lista puede ser un componente DataGrid, Tree o List.

### Ejemplo

En este ejemplo se busca el elemento seleccionado de la lista en una celda:

```
var s = listOwner.selectedItem;
```

## CellRenderer.setSize()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*componentInstance.setSize(width, height)*

### Parámetros

*width* Número que indica la anchura con la que se diseña el componente.

*height* Número que indica la altura con la que se diseña el componente.

### Valor devuelto

Ninguno.

### Descripción

Método; sirve para que la lista indique a sus celdas con qué tamaño deben diseñarse. El procesador de celdas debe llevar a cabo un diseño de modo que se ajuste al área descrita. Si no, la presentación visual de la celda puede extenderse por otras partes de la lista y aparecer rota.

### Ejemplo

En este ejemplo se da tamaño a una imagen que aparece en la celda para que quepa en los límites especificados por la lista:

```
function setSize(w:Number, h:Number) : Void
{
    image._width = w-2;
    image._height = w-2;
    image._x = image._y = 1;
}
```



## CellRenderer.setValue()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.setValue(suggested, item, selected)
```

### Parámetros

*suggested* Valor que debe utilizarse para el texto del procesador de celdas, si procede.

*item* Objeto que es el elemento completo que debe procesarse. El procesador de celdas puede utilizar todas las propiedades de este objeto que desee para la representación.

*selected* Valor booleano que indica si la fila en la que está la celda está seleccionada (true) o no lo está (false).

### Valor devuelto

Ninguno.

### Descripción

Método; toma los valores dados y crea una representación de éstos en la celda. Esto borra las diferencias entre lo que se mostraba en la celda y lo que se debe mostrar para el nuevo elemento. Es importante recordar que las celdas podían mostrar varios valores mientras existían en la lista. Es el método más importante de los procesadores de celdas.

### Ejemplo

En este ejemplo se carga una imagen en un componente Loader dentro de la celda, según el valor que se haya pasado:

```
function setValue(suggested, item, selected) : Void
{
    //borrar el cargador
    loader.contentPath = undefined;
    // la lista tiene URL de diferentes imágenes en su proveedor de datos
    if (suggested!=undefined)
        loader.contentPath = suggested;
}
```

## Componente CheckBox

Una casilla de verificación es un cuadrado que puede seleccionarse o no seleccionarse. Cuando se selecciona, en el cuadro aparece una marca. A una casilla de verificación se le puede añadir una etiqueta de texto, que puede colocarse a la izquierda, derecha, arriba o abajo.

Una casilla de verificación puede estar activada o desactivada en una aplicación. Si está activada y el usuario hace clic en ella o en su etiqueta, la casilla recibe la selección de entrada y muestra su aspecto presionado. Si el usuario desplaza el puntero fuera del área de delimitación de la casilla o de su etiqueta mientras presiona el botón del ratón, el componente recupera su aspecto original y conserva la selección de entrada. El estado de una casilla de verificación no cambia hasta que se suelta el ratón sobre el componente. Además, una casilla de verificación tiene dos estados Desactivado (seleccionado y no seleccionado) que no permiten la interacción con el ratón ni el teclado.

Si la casilla de verificación está desactivada, muestra su aspecto desactivado sea cual sea la acción del usuario. Si está desactivado, el botón no recibe la entrada del ratón ni del teclado.

La instancia de CheckBox recibe la selección cuando el usuario hace clic sobre ella o presiona el tabulador hasta su posición. Cuando una instancia de CheckBox está seleccionada, se pueden utilizar las siguientes teclas para controlarla:

| Tecla                  | Descripción                                                                            |
|------------------------|----------------------------------------------------------------------------------------|
| Mayúsculas + Tabulador | Desplaza la selección al elemento anterior.                                            |
| Barra espaciadora      | Selecciona o anula la selección del componente y activa el evento <code>click</code> . |
| Tabulador              | Desplaza la selección al elemento siguiente.                                           |

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase FocusManager” en la página 292](#).

La vista previa dinámica de cada instancia de CheckBox refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes.

Cuando se añade el componente CheckBox a una aplicación, se puede utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.CheckBoxAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#).

## Utilización del componente CheckBox

Una casilla de verificación es una parte fundamental de cualquier formulario o aplicación Web. Utilice casillas de verificación siempre que necesite reunir un conjunto de valores `true` o `false` que no se excluyan mutuamente. Por ejemplo, un formulario que recopila información personal sobre un cliente podría contener una lista de aficiones que el cliente debe elegir; cada afición llevaría al lado una casilla de verificación.

## Parámetros de CheckBox

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente CheckBox en el inspector de propiedades o en el panel Inspector de componentes:

**label** define el valor del texto de la casilla de verificación; el valor predeterminado es `defaultValue`.

**selected** define el valor inicial de la casilla de verificación en marcada (`true`) o sin marcar (`false`).

**labelPlacement** orienta el texto de la etiqueta en la casilla de verificación. Este parámetro puede tener uno de estos cuatro valores: `left`, `right`, `top` o `bottom`; el valor predeterminado es `right`. Para más información, consulte [CheckBox.labelPlacement](#).

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales de componentes CheckBox con sus propiedades, métodos y eventos. Para más información, consulte [Clase CheckBox](#).

## Creación de aplicaciones con el componente CheckBox

El siguiente procedimiento explica cómo añadir un componente CheckBox a una aplicación durante la edición. El ejemplo siguiente es un formulario para una aplicación de citas en línea. El formulario es un cuestionario que busca citas posibles para el cliente. El formulario debe llevar una casilla de verificación con la etiqueta “Restrict Age” que permita al cliente limitar la búsqueda a un grupo de edad determinado. Si selecciona la casilla “Restrict Age”, el cliente puede introducir la edad máxima y mínima en dos campos de texto que sólo se activan cuando se selecciona la casilla “Restrict Age”.

**Para crear una aplicación con el componente CheckBox, siga este procedimiento:**

- 1 Arrastre dos componentes TextInput desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca los nombres de instancia `minimumAge` y `maximumAge`.
- 3 Arrastre un componente CheckBox desde el panel Componentes al escenario.
- 4 En el inspector de propiedades, siga este procedimiento:
  - Introduzca **restrictAge** como nombre de instancia.
  - Introduzca **Restrict Age** como parámetro `label`.
- 5 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
restrictAgeListener = new Object();
restrictAgeListener.click = function (evt){
    minimumAge.enabled = evt.target.selected;
    maximumAge.enabled = evt.target.selected;
}
restrictAge.addEventListener("click", restrictAgeListener);
```

Este código crea un controlador de eventos `click` que activa y desactiva los componentes de los campos de texto `minimumAge` y `maximumAge`, ya situados en el escenario. Para más información sobre el evento `click`, consulte [CheckBox.click](#). Para más información sobre el componente TextInput, consulte [“Componente TextInput” en la página 554](#).

## Personalización del componente CheckBox

El componente CheckBox puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (`UIObject.setSize()`) o cualquier método o propiedad aplicable de la clase CheckBox (véase [Clase CheckBox](#)). Cuando se modifica el tamaño de la casilla de verificación, no cambia el tamaño de la etiqueta ni del icono de la casilla; sólo cambia el tamaño del recuadro de delimitación.

El recuadro de delimitación de una instancia de CheckBox es invisible y designa el área activa de la instancia. Si se aumenta el tamaño de la instancia, también aumenta el tamaño del área activa. Si el recuadro de delimitación es demasiado pequeño para que encaje la etiqueta, ésta se recorta para ajustarse al espacio disponible.

## Utilización de estilos con el componente CheckBox

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de CheckBox. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

El componente CheckBox admite los siguientes estilos de Halo:

| Estilo                      | Descripción                                                                                                                                  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>themeColor</code>     | Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son “haloGreen”, “haloBlue” y “haloOrange”. |
| <code>color</code>          | Texto de la etiqueta de un componente.                                                                                                       |
| <code>disabledColor</code>  | Color desactivado para el texto.                                                                                                             |
| <code>fontFamily</code>     | Nombre de la fuente del texto.                                                                                                               |
| <code>fontSize</code>       | Tamaño de la fuente en puntos.                                                                                                               |
| <code>fontStyle</code>      | Estilo de la fuente: puede ser “normal” o “italic”.                                                                                          |
| <code>fontWeight</code>     | Grosor de la fuente: puede ser “normal” o “bold”.                                                                                            |
| <code>textDecoration</code> | Decoración del texto: puede ser “none” o “underline”.                                                                                        |

## Utilización de aspectos con el componente CheckBox

El componente CheckBox utiliza símbolos del panel Biblioteca para representar estados de botón. Para aplicar aspectos al componente CheckBox durante la edición, modifique los símbolos del panel Biblioteca. Los aspectos del componente CheckBox se encuentran en la carpeta Flash UI Components 2/Themes/MMDefault/CheckBox Assets/states de la biblioteca del archivo HaloTheme.fla o SampleTheme.fla. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).

El componente CheckBox utiliza las siguientes propiedades de aspecto:

| Propiedad         | Descripción                                                          |
|-------------------|----------------------------------------------------------------------|
| falseUpSkin       | Estado Arriba. El valor predeterminado es RectBorder.                |
| falseDownSkin     | Estado Presionado. El valor predeterminado es RectBorder.            |
| falseOverSkin     | Estado Sobre. El valor predeterminado es RectBorder.                 |
| falseDisabledSkin | Estado Desactivado. El valor predeterminado es RectBorder.           |
| trueUpSkin        | Estado Conmutado. El valor predeterminado es RectBorder.             |
| trueDownSkin      | Estado Presionado conmutado. El valor predeterminado es RectBorder.  |
| trueOverSkin      | Estado Sobre conmutado. El valor predeterminado es RectBorder.       |
| trueDisabledSkin  | Estado Desactivado conmutado. El valor predeterminado es RectBorder. |

## Clase CheckBox

**Herencia** UIObject > UIComponent > SimpleButton > Button > CheckBox

**Nombre de clase de ActionScript** mx.controls.CheckBox

Las propiedades de la clase CheckBox permiten crear etiquetas de texto y colocarlas a la izquierda, derecha, encima o debajo de una casilla de verificación en tiempo de ejecución.

Si una propiedad de la clase CheckBox se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

El componente CheckBox utiliza FocusManager para sustituir el rectángulo de selección predeterminado de Flash Player por un rectángulo de selección personalizado con esquinas redondeadas. Para más información, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.CheckBox.version);
```

**Nota:** el código siguiente devuelve undefined: `trace(myCheckBoxInstance.version);`.

## Resumen de métodos de la clase CheckBox

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

## Resumen de propiedades de la clase `CheckBox`

| Propiedad                            | Descripción                                                                                                   |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------|
| <code>CheckBox.label</code>          | Especifica el texto que aparece junto a una casilla de verificación.                                          |
| <code>CheckBox.labelPlacement</code> | Especifica la orientación del texto de la etiqueta con respecto a la casilla de verificación.                 |
| <code>CheckBox.selected</code>       | Especifica si la casilla de verificación está seleccionada ( <code>true</code> ) o no ( <code>false</code> ). |

Hereda todas las propiedades de `UIObject` y `UIComponent`.

## Resumen de eventos de la clase `CheckBox`

| Evento                      | Descripción                                                                           |
|-----------------------------|---------------------------------------------------------------------------------------|
| <code>CheckBox.click</code> | Se activa cuando se presiona el ratón sobre una instancia de casilla de verificación. |

Hereda todos los eventos de `UIObject` y `UIComponent`.

## `CheckBox.click`

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(click){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.click = function(eventObject){  
    ...  
}  
checkBoxInstance.addEventListener("click", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta el botón del ratón) en la casilla de verificación o cuando ésta está seleccionada y se presiona la barra espaciadora.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `CheckBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la casilla de verificación `myCheckBox`, envía “\_level0.myCheckBox” al panel Salida:

```
on(click){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*checkBoxInstance*) distribuye un evento (en este caso, `click`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. El objeto `Event` tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `addEventListener()` (véase [UIEventDispatcher.addEventListener\(\)](#)) de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

Este ejemplo, escrito en un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se hace clic en un botón denominado `checkBoxInstance`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `click` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto `Event` pasado automáticamente a la función (en este ejemplo `eventObj`) para generar un mensaje. La propiedad `target` de un objeto `Event` es el componente que ha generado el evento (en este ejemplo `checkBoxInstance`). A la propiedad `CheckBox.selected` se accede desde la propiedad `target` del objeto `Event`. La última línea llama al método `addEventListener()` desde `checkBoxInstance` y pasa como parámetros el evento `click` y el objeto detector `form`, como en el ejemplo siguiente:

```
form = new Object();
form.click = function(eventObj){
    trace("The selected property has changed to " + eventObj.target.selected);
}
checkBoxInstance.addEventListener("click", form);
```

El código siguiente también envía un mensaje al panel Salida cuando se hace clic en `checkBoxInstance`. El controlador `on()` debe asociarse directamente con `checkBoxInstance`, como en el ejemplo siguiente:

```
on(click){
    trace("check box component was clicked");
}
```

### Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## CheckBox.label

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*checkBoxInstance.label*

### Descripción

Propiedad; indica la etiqueta de texto de la casilla de verificación. De forma predeterminada, la etiqueta aparece a la derecha de la casilla de verificación. Cuando se define, esta propiedad sustituye al parámetro `label` especificado en el panel de parámetros del clip.

### Ejemplo

El código siguiente define el texto que aparece junto al componente `CheckBox` y envía el valor al panel Salida:

```
checkBox.label = "Remove from list";  
trace(checkBox.label)
```

### Véase también

[CheckBox.labelPlacement](#)

## CheckBox.labelPlacement

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*checkBoxInstance.labelPlacement*

### Descripción

Propiedad; cadena que indica la posición de la etiqueta con respecto a la casilla de verificación. A continuación se indican los cuatro valores posibles (las líneas discontinuas representan el área de delimitación del componente; son invisibles en el documento):

- `"right"` La casilla de verificación se fija en la esquina superior izquierda del área de delimitación. La etiqueta aparece a la derecha de la casilla. Éste es el valor predeterminado.



- `"left"` La casilla de verificación se fija en la esquina superior derecha del área de delimitación. La etiqueta aparece a la izquierda de la casilla.





- "bottom" La etiqueta se coloca debajo de la casilla de verificación. El conjunto de casilla y etiqueta se centra horizontal y verticalmente.



- "top" La etiqueta se coloca encima de la casilla de verificación. El conjunto de casilla y etiqueta se centra horizontal y verticalmente.



El área de delimitación del componente se puede cambiar durante la edición con el comando Transformar o en tiempo de ejecución con la propiedad `UIObject.setSize()`. Para más información, consulte ["Personalización del componente CheckBox" en la página 96](#).

### Ejemplo

En el ejemplo siguiente se define la posición de la etiqueta a la izquierda de la casilla de verificación:

```
checkBox_mc.labelPlacement = "left";
```

### Véase también

[CheckBox.label](#)

## CheckBox.selected

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
checkBoxInstance.selected
```

### Descripción

Propiedad; valor booleano que selecciona (`true`) o anula la selección (`false`) de la casilla de verificación.

### Ejemplo

En el ejemplo siguiente se selecciona la instancia `checkbox1`:

```
checkbox1.selected = true;
```

## Componente ComboBox

Un cuadro combinado puede ser estático o editable. Un cuadro combinado estático permite al usuario hacer una sola selección en una lista desplegable. Un cuadro combinado editable permite al usuario introducir texto directamente en un campo de texto de la parte superior de la lista y seleccionar una opción en una lista desplegable. Si la lista desplegable llega al final del documento, se abre hacia arriba en lugar de hacia abajo. El cuadro combinado se compone de tres subcomponentes: un componente Button, un componente TextInput y un componente List.

Cuando se hace la selección en la lista, la etiqueta de la selección se copia en el campo de texto de la parte superior del cuadro combinado. No importa si la selección se realiza con el ratón o con el teclado.

El componente ComboBox se selecciona al hacer clic en el cuadro de texto o en el botón. Cuando un componente ComboBox está seleccionado y es editable, todas las pulsaciones del teclado se dirigen al cuadro de texto y se gestionan según las reglas del componente TextInput (véase [“Componente TextInput” en la página 554](#)), con excepción de las teclas siguientes:

| Tecla                  | Descripción                                |
|------------------------|--------------------------------------------|
| Control + Flecha abajo | Abre la lista desplegable y la selecciona. |
| Mayúsculas + Tabulador | Desplaza la selección al objeto anterior.  |
| Tabulador              | Desplaza la selección al objeto siguiente. |

Cuando un componente ComboBox está seleccionado y es estático, las pulsaciones alfanuméricas desplazan la selección hacia arriba y hacia abajo en la lista desplegable al siguiente elemento cuya inicial coincida con el carácter introducido. También puede utilizar las siguientes teclas para controlar un cuadro combinado estático:

| Tecla                   | Descripción                                                                                           |
|-------------------------|-------------------------------------------------------------------------------------------------------|
| Control + Flecha abajo  | Abre la lista desplegable y la selecciona.                                                            |
| Control + Flecha arriba | Cierra la lista desplegable si está abierta en las versiones autónoma y de navegador de Flash Player. |
| Flecha abajo            | La selección desciende un elemento.                                                                   |
| Fin                     | La selección se desplaza al final de la lista.                                                        |
| Escape                  | Cierra la lista desplegable y vuelve a seleccionar el cuadro combinado en el modo Probar película.    |
| Intro                   | Cierra la lista desplegable y vuelve a seleccionar el cuadro combinado.                               |
| Inicio                  | La selección se desplaza al principio de la lista.                                                    |
| Av Pág                  | La selección avanza una página.                                                                       |
| Re Pág                  | La selección retrocede una página.                                                                    |

| Tecla                  | Descripción                                |
|------------------------|--------------------------------------------|
| Mayúsculas + Tabulador | Desplaza la selección al objeto anterior.  |
| Tabulador              | Desplaza la selección al objeto siguiente. |

Cuando la lista desplegable de un cuadro combinado está seleccionada, las pulsaciones alfanuméricas desplazan la selección hacia arriba y hacia abajo en la lista al siguiente elemento cuya inicial coincida con el carácter introducido. También puede utilizar las siguientes teclas para controlar una lista desplegable:

| Tecla                   | Descripción                                                                                                                                                                      |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Control + Flecha arriba | Si la lista desplegable está abierta, se vuelve a seleccionar el cuadro de texto y la lista desplegable se cierra en las versiones independiente y de navegador de Flash Player. |
| Flecha abajo            | La selección desciende un elemento.                                                                                                                                              |
| Fin                     | El punto de inserción se desplaza al final del cuadro de texto.                                                                                                                  |
| Intro                   | Si la lista desplegable está abierta, se vuelve a seleccionar el cuadro de texto y se cierra la lista desplegable.                                                               |
| Escape                  | Si la lista desplegable está abierta, se vuelve a seleccionar el cuadro de texto y se cierra la lista desplegable en el modo Probar película.                                    |
| Inicio                  | El punto de inserción se desplaza al principio del cuadro de texto.                                                                                                              |
| Av Pág                  | La selección avanza una página.                                                                                                                                                  |
| Re Pág                  | La selección retrocede una página.                                                                                                                                               |
| Tabulador               | Desplaza la selección al objeto siguiente.                                                                                                                                       |
| Mayús + Fin             | Selecciona el texto desde el punto de inserción a la posición de fin.                                                                                                            |
| Mayús + Inicio          | Selecciona el texto desde el punto de inserción a la posición de inicio.                                                                                                         |
| Mayús + Tabulador       | Desplaza la selección al objeto anterior.                                                                                                                                        |
| Flecha arriba           | La selección asciende un elemento.                                                                                                                                               |

**Nota:** para las teclas Av Pág y Re Pág, una página está formada por los elementos que caben en pantalla menos uno. Por ejemplo, para avanzar por una lista desplegable que presenta las líneas de diez en diez, la pantalla muestra los elementos 0-9, 9-18, 18-27 y así sucesivamente, solapando un elemento por página.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase FocusManager” en la página 292](#).

La vista previa dinámica de cada instancia del componente ComboBox del escenario refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes. Sin embargo, la lista desplegable no se abre en la vista previa dinámica y el primer elemento aparece como elemento seleccionado.

Cuando se añade el componente ComboBox a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.ComboBoxAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda.

## Utilización del componente ComboBox

Utilice el componente ComboBox en formularios o aplicaciones en los que haya que elegir una sola opción en una lista. Por ejemplo, puede emplear una lista desplegable de provincias en un formulario de dirección de cliente. En supuestos más complejos, utilice cuadros combinados editables. Por ejemplo, en una aplicación de itinerarios, podría emplear un cuadro combinado editable para que el usuario introduzca las direcciones de origen y de destino. La lista desplegable contendría las direcciones introducidas con anterioridad.

### Parámetros de ComboBox

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente ComboBox en el inspector de propiedades o en el panel Inspector de componentes:

**editable** determina si el componente ComboBox es editable (true) o sólo seleccionable (false). El valor predeterminado es false.

**labels** rellena el componente ComboBox con una matriz de valores de texto.

**data** asocia un valor de datos con cada elemento del componente ComboBox. El parámetro data es una matriz.

**rowCount** establece el número máximo de elementos que se pueden mostrar de una vez sin utilizar la barra de desplazamiento. El valor predeterminado es 5.

Puede escribir código ActionScript para definir opciones adicionales para instancias de ComboBox con los métodos, propiedades y eventos de la clase ComboBox. Para más información, consulte [Clase ComboBox](#).

### Creación de aplicaciones con el componente ComboBox

En el siguiente procedimiento se explica cómo añadir un componente ComboBox a una aplicación durante la edición. En este ejemplo, el cuadro combinado presenta una lista de ciudades para seleccionar en la lista desplegable.

**Para crear una aplicación con el componente ComboBox, siga este procedimiento:**

- 1 Arrastre un componente ComboBox desde el panel Componentes al escenario.
- 2 Seleccione la herramienta Transformación y cambie el tamaño del componente en el escenario. Sólo es posible cambiar el tamaño de un cuadro combinado en el escenario durante la edición. Normalmente sólo hace falta modificar la anchura del cuadro para adaptarla al contenido.
- 3 Seleccione el cuadro combinado y, en el inspector de propiedades, introduzca el nombre de instancia **comboBox**.

- 4 En el panel Inspector de componentes o en el inspector de propiedades, siga este procedimiento:
  - Introduzca **Minneapolis**, **Portland** y **Keene** en el parámetro label. Haga doble clic en el campo del parámetro label para abrir el cuadro de diálogo Valores. A continuación, haga clic en el signo más para añadir elementos.
  - Introduzca **MN.swf**, **OR.swf** y **NH.swf** en el parámetro data.  
Son archivos SWF imaginarios que, por ejemplo, podrían cargarse cuando el usuario seleccionara una ciudad en el cuadro combinado.
- 5 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
form = new Object();
form.change = function (evt){
    trace(evt.target.selectedItem.label);
}
comboBox.addEventListener("change", form);
```

La última línea de código añade un controlador de eventos change a la instancia comboBox.  
Para más información, consulte [ComboBox.change](#).

## Personalización del componente ComboBox

El componente ComboBox puede transformarse horizontal y verticalmente durante la edición. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar.

Si el texto es demasiado largo para encajar en el cuadro combinado, se trunca. Para encajar el texto de la etiqueta, deberá cambiar el tamaño del cuadro combinado durante la edición.

En los cuadros combinados editables, el área activa es sólo el botón, no el cuadro de texto. En los cuadros combinados estáticos, el área activa está formada por el botón y el cuadro de texto.

## Utilización de estilos con el componente ComboBox

Es posible definir propiedades de estilo para cambiar el aspecto de un componente ComboBox. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

Los cuadros combinados sólo tienen dos estilos exclusivos. Los estilos del botón, el cuadro de texto y la lista desplegable del cuadro combinado se asignan a través de dichos componentes, como se indica a continuación:

- El botón es una instancia de Button y utiliza sus estilos. (Véase [“Utilización de estilos con el componente Button” en la página 77](#).)
- El texto es una instancia de TextInput y utiliza sus estilos. (Véase [“Utilización de estilos con el componente TextInput” en la página 556](#).)
- La lista desplegable es una instancia de List y utiliza sus estilos. (Véase [“Utilización de estilos con el componente List” en la página 313](#).)

El componente ComboBox utiliza los siguientes estilos de Halo:

| Estilo         | Descripción                                                                                                                                                                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| themeColor     | Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son "haloGreen", "haloBlue" y "haloOrange".                                                                                                                                                                                                                           |
| color          | Texto de la etiqueta de un componente.                                                                                                                                                                                                                                                                                                                                 |
| disabledColor  | Color desactivado para el texto.                                                                                                                                                                                                                                                                                                                                       |
| fontFamily     | Nombre de la fuente del texto.                                                                                                                                                                                                                                                                                                                                         |
| fontSize       | Tamaño de la fuente en puntos.                                                                                                                                                                                                                                                                                                                                         |
| fontStyle      | Estilo de la fuente: puede ser "normal" o "italic".                                                                                                                                                                                                                                                                                                                    |
| fontWeight     | Grosor de la fuente: puede ser "normal" o "bold".                                                                                                                                                                                                                                                                                                                      |
| textDecoration | Decoración del texto: puede ser "none" o "underline".                                                                                                                                                                                                                                                                                                                  |
| openDuration   | Número de milisegundos para abrir la lista desplegable. El valor predeterminado es 250.                                                                                                                                                                                                                                                                                |
| openEasing     | Referencia a la función de interpolación que controla la animación de la lista desplegable. De forma predeterminada, va a la pantalla de inicio y luego a la pantalla de desconexión del sitio Web. Para obtener más funciones, descargue una lista del <a href="http://www.robertpenner.com/easing/">sitio Web de Robert Penner en www.robertpenner.com/easing/</a> . |

## Utilización de aspectos con el componente ComboBox

El componente ComboBox utiliza símbolos del panel Biblioteca para representar estados de botón. El componente ComboBox tiene variables de aspecto para la flecha abajo. Aparte, utiliza la barra de desplazamiento y los aspectos de lista. Para asignar aspectos al componente ComboBox durante la edición, modifique símbolos del panel Biblioteca y vuelva a exportar el componente como SWC. Los aspectos del componente CheckBox se encuentran en la carpeta Flash UI Components 2/Themes/MMDefault/ComboBox Assets/states de la biblioteca del archivo HaloTheme fla o SampleTheme fla. Para más información, consulte ["Aplicación de aspectos a los componentes" en la página 41](#).

El componente ComboBox utiliza las siguientes propiedades de aspecto:

| Propiedad                  | Descripción                                                                   |
|----------------------------|-------------------------------------------------------------------------------|
| ComboDownArrowDisabledName | Estado Desactivado de la flecha abajo. El valor predeterminado es RectBorder. |
| ComboDownArrowDownName     | Estado Presionado de la flecha abajo. El valor predeterminado es RectBorder.  |
| ComboDownArrowUpName       | Estado Arriba de la flecha abajo. El valor predeterminado es RectBorder.      |
| ComboDownArrowOverName     | Estado Sobre de la flecha abajo. El valor predeterminado es RectBorder.       |

## Clase ComboBox

**Herencia** UIObject > UIComponent > ComboBase > ComboBox

**Nombre de clase de ActionScript** mx.controls.ComboBox

El componente ComboBox combina tres subcomponentes independientes: Button, TextInput y List. La mayoría de las interfaces API de cada subcomponente están disponibles directamente en el componente ComboBox y se enumeran en las tablas de métodos, propiedades y eventos de la clase ComboBox.

La lista desplegable de un cuadro combinado se obtiene con un objeto Array o un objeto DataProvider. Si utiliza un objeto DataProvider, la lista cambia en tiempo de ejecución. La fuente de datos de ComboBox puede modificarse dinámicamente cambiando a otro objeto Array o DataProvider.

Los elementos de una lista de un cuadro combinado se indexan por posición a partir del número 0. Un elemento puede ser:

- Un tipo de datos simple.
- Un objeto que contiene una propiedad `label` y una propiedad `data`.

**Nota:** un objeto puede utilizar la propiedad `ComboBox.labelFunction` o `ComboBox.labelField` para determinar la propiedad `label`.

Si el elemento es un tipo de datos simple distinto de una cadena, se convierte en una cadena. Si el elemento es un objeto, la propiedad `label` debe ser una cadena y la propiedad `data` puede ser cualquier valor de ActionScript.

Los métodos del componente ComboBox a los que se suministran elementos tienen dos parámetros, `label` y `data`, que hacen referencia a las propiedades del mismo nombre. Los métodos que devuelven elementos devuelven objetos.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.ComboBox.version);
```

**Nota:** el código siguiente devuelve `undefined`: `trace(myComboBoxInstance.version);`.

## Resumen de métodos de la clase `ComboBox`

| Propiedad                             | Descripción                                                       |
|---------------------------------------|-------------------------------------------------------------------|
| <code>ComboBox.addItem()</code>       | Añade un elemento al final de la lista.                           |
| <code>ComboBox.addItemAt()</code>     | Añade un elemento al final de la lista en el índice especificado. |
| <code>ComboBox.close()</code>         | Cierra la lista desplegable.                                      |
| <code>ComboBox.getItemAt()</code>     | Devuelve el elemento del índice especificado.                     |
| <code>ComboBox.open()</code>          | Abre la lista desplegable.                                        |
| <code>ComboBox.removeAll()</code>     | Elimina todos los elementos de la lista.                          |
| <code>ComboBox.removeItemAt()</code>  | Elimina un elemento de la lista en la posición especificada.      |
| <code>ComboBox.replaceItemAt()</code> | Sustituye un elemento de la lista por otro elemento especificado. |

Hereda todos los métodos de `UIObject` y `UIComponent`.

## Resumen de propiedades de la clase `ComboBox`

| Propiedad                           | Descripción                                                                               |
|-------------------------------------|-------------------------------------------------------------------------------------------|
| <code>ComboBox.dataProvider</code>  | Modelo de datos de los elementos de la lista.                                             |
| <code>ComboBox.dropdown</code>      | Devuelve una referencia al componente <code>List</code> contenido en el cuadro combinado. |
| <code>ComboBox.dropdownWidth</code> | Anchura de la lista desplegable, expresada en píxeles.                                    |
| <code>ComboBox.editable</code>      | Indica si el cuadro combinado es editable o no.                                           |
| <code>ComboBox.labelField</code>    | Indica qué campo de datos ha de utilizarse como etiqueta para la lista desplegable.       |
| <code>ComboBox.labelFunction</code> | Especifica una función para calcular el campo de etiqueta de la lista desplegable.        |
| <code>ComboBox.length</code>        | Sólo lectura. Longitud de la lista desplegable.                                           |
| <code>ComboBox.rowCount</code>      | Número máximo de elementos de la lista que se muestran de una vez.                        |
| <code>ComboBox.selectedIndex</code> | Índice del elemento seleccionado en la lista desplegable.                                 |
| <code>ComboBox.selectedItem</code>  | Valor del elemento seleccionado en la lista desplegable.                                  |
| <code>ComboBox.text</code>          | Cadena de texto del cuadro de texto.                                                      |
| <code>ComboBox.textField</code>     | Referencia al componente <code>TextInput</code> del cuadro combinado.                     |
| <code>ComboBox.value</code>         | Valor del cuadro de texto (editable) o lista desplegable (estático).                      |

Hereda todas las propiedades de `UIObject` y `UIComponent`.



## Resumen de eventos de la clase ComboBox

| Evento                             | Descripción                                                                                          |
|------------------------------------|------------------------------------------------------------------------------------------------------|
| <code>ComboBox.change</code>       | Se difunde cuando el valor del cuadro combinado cambia como resultado de la interacción del usuario. |
| <code>ComboBox.close</code>        | Se difunde cuando la lista desplegable comienza a cerrarse.                                          |
| <code>ComboBox.enter</code>        | Se difunde cuando se presiona la tecla Intro.                                                        |
| <code>ComboBox.itemRollOut</code>  | Se difunde cuando el puntero se desplaza fuera de un elemento de la lista desplegable.               |
| <code>ComboBox.itemRollOver</code> | Se difunde cuando el ratón se desplaza sobre un elemento de la lista desplegable.                    |
| <code>ComboBox.open</code>         | Se difunde cuando la lista desplegable comienza a abrirse.                                           |
| <code>ComboBox.scroll</code>       | Se difunde cuando se recorre la lista desplegable.                                                   |

Hereda todos los eventos de [UIObject](#) y [UIComponent](#).

### ComboBox.addItem()

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.

#### Sintaxis

Sintaxis 1:

```
comboBoxInstance.addItem(label[, data])
```

Sintaxis 2:

```
comboBoxInstance.addItem({label:label[, data:data]})
```

Sintaxis 3:

```
comboBoxInstance.addItem(obj);
```

#### Parámetros

*label* Cadena que indica la etiqueta del elemento nuevo.

*data* Datos del elemento; pueden ser cualquier tipo de datos. Este parámetro es opcional.

*obj* Objeto con una propiedad label y una propiedad data opcional.

#### Valor devuelto

Índice en el que se ha añadido el elemento.

#### Descripción

Método; añade un elemento nuevo al final de la lista.

### Ejemplo

El código siguiente añade un elemento a la instancia `myComboBox`:

```
myComboBox.addItem("esto es un elemento");
```

## ComboBox.addItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
comboBoxInstance.addItemAt(index, label[, data])
```

### Parámetros

*index* Número 0 o superior que indica la posición en la que insertar el elemento (índice del elemento nuevo).

*label* Cadena que indica la etiqueta del elemento nuevo.

*data* Datos del elemento; pueden ser cualquier tipo de datos. Este parámetro es opcional.

### Valor devuelto

Índice en el que se ha añadido el elemento.

### Descripción

Método; añade un elemento nuevo al final de la lista en el índice especificado con el parámetro *index*. Los índices superiores a `ComboBox.length` se omiten.

### Ejemplo

El código siguiente inserta un elemento en el índice 3, que es la cuarta posición de la lista del cuadro combinado (la primera posición es 0):

```
myBox.addItemAt(3, "This is the fourth Item");
```

## ComboBox.change

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

### Sintaxis 1:

```
on(change){  
    // aquí código propio  
}
```

### Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("change", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando el valor del cuadro combinado cambia como resultado de la interacción del usuario.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myBox` del componente `ComboBox`, envía “\_level0.myBox” al panel Salida:

```
on(change){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`comboBoxInstance`) distribuye un evento (en este caso, `change`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (`listenerObject`) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (`eventObject`) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `addEventListener()` (véase [UIEventDispatcher.addEventListener\(\)](#)) de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 602](#).

## Ejemplo

En el ejemplo siguiente se envía al panel Salida el nombre de instancia del componente que ha generado el evento `change`:

```
form.change = function(eventObj){  
    trace("Value changed to " + eventObj.target.value);  
}  
myCombo.addEventListener("change", form);
```

## Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## ComboBox.close()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
myComboBox.close()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; cierra la lista desplegable.

### Ejemplo

En el ejemplo siguiente se cierra la lista desplegable del cuadro combinado `myBox`:

```
myBox.close();
```

### Véase también

[ComboBox.open\(\)](#)

## ComboBox.close

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(close){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.close = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("close", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando la lista del cuadro combinado comienza a plegarse.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myBox` del componente `ComboBox`, envía “\_level0.myBox” al panel Salida:

```
on(close){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso, `close`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida cuando la lista desplegable comienza a cerrarse:

```
form.close = function(){
    trace("The combo box has closed");
}
myCombo.addEventListener("close", form);
```

## Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## ComboBox.dataProvider

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*comboBoxInstance*.dataProvider

## Descripción

Propiedad; modelo de datos de los elementos que se ven en una lista. El valor de esta propiedad puede ser una matriz o cualquier objeto que implemente la interfaz `DataProvider`. El valor predeterminado es `[]`. Ésta es una propiedad del componente `List`, pero se puede acceder a ella directamente desde una instancia del componente `ComboBox`.

El componente `List` y otros componentes para datos añaden métodos al prototipo del objeto `Array` para ceñirse a la interfaz `DataProvider` (para más detalles, consulte `DataProvider.as`). Por lo tanto, cualquier matriz que exista a la vez que la lista tiene automáticamente todos los métodos necesarios (`addItem()`, `getItemAt()`, etc.) para ser el modelo de la lista y puede utilizarse para difundir cambios de modelo a varios componentes.

Si la matriz contiene objetos, accede a las propiedades `labelField` o `labelFunction` para determinar qué partes del elemento deben mostrarse. El valor predeterminado es `"label"`, por lo que, si existe este campo, es el campo que se muestra; en caso contrario, se muestra una lista de todos los campos separados por comas.

**Nota:** si la matriz contiene cadenas y no objetos en todos los índices, la lista no puede ordenar los elementos y mantener el estado de la selección. Al ordenar se pierde la selección.

Como proveedor de datos para `List` pueden elegirse todas las instancias que implementen la interfaz `DataProvider`. Entre ellas se encuentran los juegos de registros de `Flash Remoting`, los conjuntos de datos de `Firefly`, etc.

## Ejemplo

Este ejemplo utiliza una matriz de cadenas para rellenar la lista desplegable:

```
comboBox.dataProvider = ["Ground Shipping", "2nd Day Air", "Next Day Air"];
```

Este ejemplo crea una matriz proveedora de datos y la asigna a la propiedad `dataProvider`:

```
myDP = new Array();
list.dataProvider = myDP;

for (var i=0; i<accounts.length; i++) {
    // estos cambios en DataProvider se difundirán a la lista
    myDP.addItem({ label: accounts[i].name,
                   data: accounts[i].accountID });
}
```

## ComboBox.dropdown

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
myComboBox.dropdown
```

## Descripción

Propiedad (sólo lectura); devuelve una referencia al componente List contenido en el cuadro combinado. No se crea una instancia en el cuadro combinado para el subcomponente List hasta que se tenga que visualizar. No obstante, cuando se accede a la propiedad `dropdown`, se crea la lista.

## Véase también

[ComboBox.dropdownWidth](#)

## ComboBox.dropdownWidth

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox.change*

### Descripción

Propiedad; límite de anchura de la lista desplegable, expresado en píxeles. El valor predeterminado es la anchura del componente ComboBox (instancia de TextInput más instancia de SimpleButton).

### Ejemplo

El código siguiente define `dropdownWidth` en 150 píxeles:

```
myComboBox.dropdownWidth = 150;
```

## Véase también

[ComboBox.dropdown](#)

## ComboBox.editable

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox.editable*

### Descripción

Propiedad; indica si el cuadro combinado es editable (`true`) o no (`false`). Un cuadro combinado editable permite introducir en el cuadro de texto valores que no se ven en la lista desplegable. Si el cuadro combinado no es editable, en el cuadro de texto sólo pueden introducirse los valores de la lista desplegable. El valor predeterminado es `false`.

Si el cuadro combinado se define en editable, el campo de texto del cuadro combinado se borra. Además, el índice (y el elemento) seleccionado se define en un valor no definido. Para convertir un cuadro combinado en editable conservando el elemento seleccionado, utilice el código siguiente:

```
var ix = myComboBox.selectedIndex;
myComboBox.editable = true; // borra el campo de texto.
myComboBox.selectedIndex = ix; // vuelve a copiar la etiqueta en el campo de
    texto.
```

### Ejemplo

El código siguiente convierte myComboBox en editable:

```
myComboBox.editable = true;
```

## ComboBox.enter

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(enter){
    // aquí código propio
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.enter = function(eventObject){
    // aquí código propio
}
comboBoxInstance.addEventListener("enter", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando se presiona la tecla Intro en el cuadro de texto. Este evento sólo se difunde desde cuadros combinados editables. Éste es un evento TextInput que se difunde desde un cuadro combinado. Para más información, consulte [TextInput.enter](#).

El primer ejemplo de sintaxis utiliza un controlador on() que debe asociarse directamente con una instancia del componente ComboBox. La palabra clave this, utilizada en un controlador on() asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia myBox del componente ComboBox, envía “\_level0.myBox” al panel Salida:

```
on(enter){
    trace(this);
}
```



El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso, *enter*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida cuando la lista desplegable comienza a cerrarse:

```
form.enter = function(){
    trace("The combo box enter event was triggered");
}
myCombo.addEventListener("enter", form);
```

### Véase también

`UIEventDispatcher.addEventListener()`

## ComboBox.getItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*comboBoxInstance.getItemAt(index)*

### Parámetros

*index* Número mayor o igual que 0 y menor que `ComboBox.length`. Índice del elemento que debe recuperarse.

### Valor devuelto

Valor u objeto del elemento indexado. Si el índice está fuera de rango, el valor es `undefined`.

### Descripción

Método; recupera el elemento del índice especificado.

### Ejemplo

El código siguiente muestra el elemento de la posición 4 del índice:

```
trace(myBox.getItemAt(4).label);
```

## ComboBox.itemRollOut

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(itemRollOut){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.itemRollOut = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("itemRollOut", listenerObject)
```

### Objeto Event

Además de las propiedades estándar del objeto Event, el evento `itemRollOut` tiene una propiedad adicional: `index`. `index` es el número del elemento del que ha salido el puntero.

### Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero sale de los elementos de la lista desplegable. Éste es un evento List que se difunde desde un cuadro combinado. Para más información, consulte [List.itemRollOut](#).

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente ComboBox. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myBox` del componente ComboBox, envía “\_level0.myBox” al panel Salida:

```
on(itemRollOut){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso, `itemRollOut`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 602.

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

## Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar de qué número de índice de elemento ha salido el puntero:

```
form.itemRollOut = function (eventObj) {  
    trace("Item #" + eventObj.index + " has been rolled out of.");  
}  
myCombo.addEventListener("itemRollOut", form);
```

## Véase también

[ComboBox.itemRollOver](#), [UIEventDispatcher.addEventListener\(\)](#)

## ComboBox.itemRollOver

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(itemRollOver){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.itemRollOver = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("itemRollOver", listenerObject)
```

### Objeto Event

Además de las propiedades estándar del objeto Event, el evento `itemRollOver` tiene una propiedad adicional: `index`. `index` es el número del elemento sobre el que ha pasado el puntero.

### Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero se desplaza por los elementos de la lista desplegable. Éste es un evento List que se difunde desde un cuadro combinado. Para más información, consulte [List.itemRollOver](#).

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myBox` del componente `ComboBox`, envía “\_level0.myBox” al panel Salida:

```
on(itemRollOver){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso, *itemRollOver*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

### Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar por qué número de índice de elemento ha pasado el puntero:

```
form.itemRollOver = function (eventObj) {  
    trace("Item #" + eventObj.index + " has been rolled over.");  
}  
myCombo.addEventListener("itemRollOver", form);
```

### Véase también

[ComboBox.itemRollOut](#), [UIEventDispatcher.addEventListener\(\)](#)

## ComboBox.labelField

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox*.labelField

### Descripción

Propiedad; nombre del campo de los objetos de matriz `dataProvider` que se utilizará como campo de etiqueta. Ésta es una propiedad del componente `List` que está disponible desde una instancia del componente `ComboBox`. Para más información, consulte [List.labelField](#).

El valor predeterminado es `undefined`.

## Ejemplo

En el ejemplo siguiente se define la propiedad `dataProvider` en una matriz de cadenas y la propiedad `labelField` para que indique que el campo `name` debe utilizarse como etiqueta de la lista desplegable:

```
myComboBox.dataProvider = [
    {name:"Gary", gender:"male"},
    {name:"Susan", gender:"female"} ];

myComboBox.labelField = "name";
```

## Véase también

[List.labelFunction](#)

## ComboBox.labelFunction

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox.labelFunction*

### Descripción

Propiedad; función que calcula la etiqueta de un elemento `dataProvider`. Es preciso definir la función. El valor predeterminado es `undefined`.

## Ejemplo

En el ejemplo siguiente se crea un proveedor de datos y después define una función que especifica qué debe utilizarse como etiqueta en la lista desplegable:

```
myComboBox.dataProvider = [
    {firstName:"Nigel", lastName:"Pegg", age:"really young"},
    {firstName:"Gary", lastName:"Grossman", age:"young"},
    {firstName:"Chris", lastName:"Walcott", age:"old"},
    {firstName:"Greg", lastName:"Yachuk", age:"really old"} ];

myComboBox.labelFunction = function(itemObj){
    return (itemObj.lastName + ", " + itemObj.firstName);
}
```

## Véase también

[List.labelField](#)

## ComboBox.length

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox.length*

### Descripción

Propiedad (sólo lectura); longitud de la lista desplegable. Ésta es una propiedad del componente List que está disponible desde una instancia de ComboBox. Para más información, consulte [List.length](#). El valor predeterminado es 0.

### Ejemplo

En el ejemplo siguiente se almacena el valor `length` en una variable:

```
dropdownItemCount = myBox.length;
```

## ComboBox.open()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox.open()*

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Propiedad; abre la lista desplegable.

### Ejemplo

El código siguiente abre la lista desplegable de la instancia `combo1`:

```
combo1.open();
```

### Véase también

[ComboBox.close\(\)](#)

# ComboBox.open

## Disponibilidad

Flash Player 6 versión 79.

## Edición

Flash MX 2004.

## Sintaxis

Sintaxis 1:

```
on(open){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.open = function(eventObject){  
    // aquí código propio  
}  
comboBoxInstance.addEventListener("open", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando la lista desplegable comienza a aparecer.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente ComboBox. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myBox` del componente ComboBox, envía “\_level0.myBox” al panel Salida:

```
on(open){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*comboBoxInstance*) distribuye un evento (en este caso, `open`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

### Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar de qué número de índice de elemento ha salido el puntero:

```
form.open = function () {  
    trace("The combo box has opened with text " + myBox.text);  
}  
myBox.addEventListener("open", form);
```

### Véase también

[ComboBox.close](#), [UIEventDispatcher.addEventListener\(\)](#)

## ComboBox.removeAll()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
comboBoxInstance.removeAll()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; elimina todos los elementos de la lista. Éste es un método del componente List que está disponible desde una instancia del componente ComboBox.

### Ejemplo

El código siguiente borra la lista:

```
myCombo.removeAll();
```

### Véase también

[ComboBox.removeItemAt\(\)](#), [ComboBox.replaceItemAt\(\)](#)

## ComboBox.removeItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.



### Sintaxis

```
listInstance.removeItemAt(index)
```

### Parámetros

*index* Número que indica la posición del elemento que va a eliminarse. El valor está basado en cero.

### Valor devuelto

Objeto; elemento eliminado (undefined si no hay elementos).

### Descripción

Método; elimina el elemento de la posición especificada en *index*. Los índices de la lista a partir del índice que indica el parámetro *index* se contraen una posición. Éste es un método del componente List que está disponible desde una instancia del componente ComboBox.

### Ejemplo

El código siguiente elimina el elemento de la posición 3 del índice:

```
myCombo.removeItemAt(3);
```

### Véase también

```
ComboBox.removeAll(), ComboBox.replaceItemAt()
```

## ComboBox.replaceltemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
comboBoxInstance.replaceItemAt(index, label[, data])
```

### Parámetros

*index* Número 0 o superior que indica la posición en la que insertar el elemento (índice del elemento nuevo).

*label* Cadena que indica la etiqueta del elemento nuevo.

*data* Datos del elemento. Este parámetro es opcional.

### Valor devuelto

Ninguno.

### Descripción

Método; sustituye el contenido del elemento del índice que especifica el parámetro *index*. Éste es un método del componente List que está disponible desde el componente ComboBox.

### Ejemplo

En el ejemplo siguiente se cambia la posición del tercer índice:

```
myCombo.replaceItemAt(3, "new label");
```

### Véase también

[ComboBox.removeAll\(\)](#), [ComboBox.removeItemAt\(\)](#)

## ComboBox.rowCount

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
myComboBox.rowCount
```

### Descripción

Propiedad; número máximo de filas visibles en la lista desplegable. El valor predeterminado es 5.

Si la lista desplegable contiene un número de elementos mayor o igual que la propiedad `rowCount`, cambia de tamaño y, si es necesario, muestra una barra de desplazamiento. Si la lista desplegable contiene menos elementos que la propiedad `rowCount`, adapta su tamaño al número de elementos de la lista.

Este comportamiento difiere del componente List, que siempre muestra el número de filas que especifique la propiedad `rowCount` aunque quede espacio vacío.

Si el valor es negativo o fraccionario, el comportamiento es undefined.

### Ejemplo

En el ejemplo siguiente se especifica que el cuadro combinado tenga 20 filas visibles o menos:

```
myComboBox.rowCount = 20;
```

## ComboBox.scroll

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

### Sintaxis 1:

```
on(scroll){
    // aquí código propio
}
```

### Sintaxis 2:

```
listenerObject = new Object();
listenerObject.scroll = function(eventObject){
    // aquí código propio
}
comboBoxInstance.addEventListener("scroll", listenerObject)
```

## Objeto Event

Además de las propiedades estándar del objeto Event, el evento `scroll` tiene una propiedad adicional, `direction`. Es una cadena con dos valores posibles: "horizontal" o "vertical". En los eventos `scroll` de `ComboBox`, el valor siempre es "vertical".

## Descripción

Evento; se difunde a todos los detectores registrados cuando se recorre la lista desplegable. Éste es un evento del componente `List` disponible para `ComboBox`.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `ComboBox`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myBox` del componente `ComboBox`, envía “\_level0.myBox” al panel Salida:

```
on(scroll){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`comboBoxInstance`) distribuye un evento (en este caso, `scroll`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (`listenerObject`) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (`eventObject`) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

## Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar hasta qué número de índice de elemento se ha desplazado el puntero:

```
form.scroll = function (eventObj) {
    trace("The list had been scrolled to item # " + eventObj.target.vPosition);
}
myCombo.addEventListener("scroll", form);
```

## Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## ComboBox.selectedIndex

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox*.selectedIndex

### Descripción

Propiedad; índice (número) del elemento seleccionado en la lista desplegable. El valor predeterminado es 0. Cuando se asigna esta propiedad, se borra la selección actual, se selecciona el elemento indicado y el cuadro de texto del cuadro combinado muestra la etiqueta del elemento indicado.

Si se asigna un valor `selectedIndex` fuera de rango, el valor se omite. Cuando se introduce texto en el campo de texto de un cuadro combinado editable, `selectedIndex` se establece en `undefined`.

### Ejemplo

El siguiente código selecciona el último elemento de la lista:

```
myComboBox.selectedIndex = myComboBox.length-1;
```

## Véase también

[ComboBox.selectedItem](#)

## ComboBox.selectedItem

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox*.selectedItem

### Descripción

Propiedad; valor del elemento seleccionado en la lista desplegable.

Si el cuadro combinado es editable y se introduce texto en el cuadro de texto, `selectedItem` devuelve `undefined`. Sólo contiene un valor cuando se selecciona un elemento en la lista desplegable o si el valor se define mediante `ActionScript`. Si el cuadro combinado es estático, el valor de `selectedItem` siempre es válido.

## Ejemplo

En el ejemplo siguiente se muestra el valor de `selectedItem` si el proveedor de datos contiene tipos de datos simples:

```
var item = myComboBox.selectedItem;  
trace("You selected the item " + item);
```

En el ejemplo siguiente se muestra el valor de `selectedItem` si el proveedor de datos contiene objetos con las propiedades `label` y `data`:

```
var obj = myComboBox.selectedItem;  
trace("You have selected the color named: " + obj.label);  
trace("The hex value of this color is: " + obj.data);
```

## Véase también

[ComboBox.dataProvider](#), [ComboBox.selectedIndex](#)

## ComboBox.text

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox*.text

### Descripción

Propiedad; texto del cuadro de texto. Este valor se puede obtener y definir con cuadros combinados editables. Con cuadros combinados estáticos, el valor es de sólo lectura.

## Ejemplo

En el ejemplo siguiente se define el valor actual de `text` de un cuadro combinado editable:

```
myComboBox.text = "California";
```

## ComboBox.textField

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox*.textField

## Descripción

Propiedad (sólo lectura); referencia al componente TextInput contenido en el componente ComboBox.

Esta propiedad permite acceder al componente TextInput subyacente para manipularlo. Por ejemplo, quizá desee cambiar la selección del cuadro de texto o limitar los caracteres que se pueden introducir en él.

## Ejemplo

El código siguiente restringe el cuadro de texto de `myComboBox` a los números aceptados:

```
myComboBox.textField.restrict = "0-9";
```

## ComboBox.value

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*myComboBox.value*

## Descripción

Propiedad (sólo lectura); si el cuadro combinado es editable, `value` devuelve el valor del cuadro de texto. Si el cuadro de texto es estático, `value` devuelve el valor de la lista desplegable. El valor de la lista desplegable es el campo `data` o, si el campo `data` no existe, el campo `label`.

## Ejemplo

En el ejemplo siguiente se introducen los datos en el cuadro combinado al definir la propiedad `dataProvider`. Después muestra `value` en el panel Salida. Finalmente, selecciona "California" y lo muestra en el cuadro de texto:

```
cb.dataProvider = [
    {label:"Alaska", data:"AZ"},
    {label:"California", data:"CA"},
    {label:"Washington", data:"WA"}];
cb.editable = true;
cb.selectedIndex = 1;
trace('Editable value is "California": '+ cb.value);
cb.editable = false;
cb.selectedIndex = 1;
trace('Non-editable value is "CA": '+ cb.value);
```

## Clases de vinculación de datos (sólo para Flash Professional)

Las clases de vinculación de datos proporcionan la funcionalidad en tiempo de ejecución para la función de vinculación de datos en Flash MX Professional 2004. Puede crear y configurar visualmente vinculaciones de datos en el entorno de edición de Flash mediante la ficha Vinculaciones del panel Inspector de componentes, así como crear y configurar mediante programación vinculaciones utilizando las clases incluidas en el paquete `mx.data.binding`.

Para ver información general sobre la vinculación de datos y sobre cómo crear visualmente vinculaciones de datos en la herramienta de edición de Flash, consulte “Vinculación de datos (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

### Disponibilidad de las clases de vinculación de datos en tiempo de ejecución (sólo para Flash Professional)

Para que las clases de vinculación de datos estén disponibles en tiempo de ejecución, el componente `DataBindingClasses` debe estar almacenado en la biblioteca del archivo FLA. Cuando se crean visualmente vinculaciones en el entorno de edición de Flash, este componente se añade automáticamente a la biblioteca del documento. Sin embargo, si sólo utiliza `ActionScript` para crear vinculaciones en el tiempo de ejecución, deberá añadir este componente de forma manual a la biblioteca del documento. Para más información sobre cómo añadir este componente al documento, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

### Clases del paquete `mx.data.binding` (sólo para Flash Professional)

En la tabla siguiente se enumeran las clases incluidas en el paquete `mx.data.binding`.

| Clase                                                                | Descripción                                                                                   |
|----------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| <a href="#">Clase Binding (sólo para Flash Professional)</a>         | Crea una vinculación entre dos puntos finales.                                                |
| <a href="#">Clase ComponentMixins (sólo para Flash Professional)</a> | Añade funciones específicas de vinculación de datos a los componentes.                        |
| <a href="#">Clase CustomFormatter (sólo para Flash Professional)</a> | Clase base para crear clases de formateador personalizado.                                    |
| <a href="#">Clase CustomValidator (sólo para Flash Professional)</a> | Clase base para crear clases de validador personalizado.                                      |
| <a href="#">Clase DataType (sólo para Flash Professional)</a>        | Ofrece un acceso de lectura y escritura a los campos de datos de una propiedad de componente. |
| <a href="#">Clase EndPoint (sólo para Flash Professional)</a>        | Define el origen o el destino de una vinculación.                                             |
| <a href="#">Clase TypedValue (sólo para Flash Professional)</a>      | Contiene un valor de datos e información sobre el tipo de datos del valor.                    |

## Clase Binding (sólo para Flash Professional)

**Nombre de clase de ActionScript**    `mx.data.binding.Binding`

La clase Binding define una asociación entre dos puntos finales: un *origen* y un *destino*. Detecta cambios en el punto final de origen y copia los datos modificados en el punto final de destino cada vez que el origen cambia.

Puede escribir vinculaciones personalizadas mediante la clase Binding (y las clases de soporte) o utilizar la ficha Vinculaciones del panel Inspector de componentes (Ventana > Paneles de desarrollo > Inspector de componentes).

**Nota:** para que esta clase esté disponible en tiempo de ejecución, debe incluir el componente DataBindingClasses en el documento FLA. Para más información, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

Para ver información general sobre las clases del paquete `mx.data.binding`, consulte [“Clases de vinculación de datos \(sólo para Flash Professional\)” en la página 131](#).

## Resumen de métodos de la clase Binding

| Método                         | Descripción                                                                                      |
|--------------------------------|--------------------------------------------------------------------------------------------------|
| <code>Binding.execute()</code> | Recoge los datos del componente de origen, les da formato y los asigna al componente de destino. |

## Constructor para la clase Binding

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
new Binding(source, destination, [format], [isTwoWay])
```

### Parámetros

*source*    Punto final de origen de la vinculación. Este parámetro es nominalmente de tipo `mx.data.binding.EndPoint`, pero puede ser cualquier objeto ActionScript que cuente con los campos `EndPoint` necesarios. Véase [Clase EndPoint \(sólo para Flash Professional\)](#).

*destination*    Punto final de destino de la vinculación. Este parámetro es nominalmente de tipo `mx.data.binding.EndPoint`, pero puede ser cualquier objeto ActionScript que cuente con los campos `EndPoint` necesarios. Véase [Clase EndPoint \(sólo para Flash Professional\)](#).



*format* (Opcional) Objeto que contiene información de formato. El objeto debe tener las propiedades siguientes:

- *cls* Clase `ActionScript` que amplía la clase `mx.data.binding.DataAccessor`.
- *settings* Objeto cuyas propiedades proporcionan una configuración opcional para la clase de formateador especificada por *cls*.

*isTwoWay* (Opcional) Valor booleano que especifica si el nuevo objeto `Binding` es bidireccional (`true`) o no lo es (`false`). El valor predeterminado es `false`.

### Valor devuelto

Ninguno.

### Descripción

Constructor; crea un objeto `Binding`. Puede vincular datos a cualquier objeto `ActionScript` que tenga propiedades y emita eventos como los componentes (entre otros).

Un objeto de vinculación existirá mientras el clip de película más interior contenga los componentes de origen y de destino. Por ejemplo, si un clip de película denominado “A” contiene los componentes “X” e “Y” y existe una vinculación entre “X” e “Y”, la vinculación existirá mientras exista el clip de película A.

**Nota:** no es necesario retener una referencia al nuevo objeto `Binding`, aunque es posible. Cuando se crea el objeto `Binding`, empieza a detectar de forma inmediata los eventos "modificados" que ha emitido cada `EndPoint`. Sin embargo, en algunos casos deberá guardar una referencia al nuevo objeto `Binding` para poder llamar a su método `execute()` más adelante (véase [Binding.execute\(\)](#)).

### Ejemplo

Ejemplo nº 1: en este ejemplo, la propiedad `text` de un componente `TextInput` (`src_txt`) está vinculada a la propiedad `text` de otro componente `TextInput` (`dest_txt`). Cuando el campo de texto `src_txt` deja de estar seleccionado (es decir, cuando se genera el evento `focusOut`), el valor de su propiedad `text` se copia en `dest_txt.text`.

```
import mx.data.binding.*;
var src = new EndPoint();
src.component = src_txt;
src.property = "text";
src.event = "focusOut";

var dest= new EndPoint();
dest.component = dest_txt;
dest.property = "text";

new Binding(src, dest);
```

Ejemplo nº 2: en este ejemplo se muestra cómo crear un objeto Binding que utiliza una clase personalizada de formateador. Para más información sobre la creación de clases de formateador personalizado, consulte [“Clase CustomFormatter \(sólo para Flash Professional\)” en la página 135](#).

```
import mx.data.binding.*;
var src = new EndPoint();
src.component = src_txt;
src.property = "text";
src.event = "focusOut";

var dest= new EndPoint();
dest.component = text_dest;
dest.property = "text";

new Binding(src, dest, {cls: mx.data.formatters.Custom, settings: {classname:
    "com.mycompany.SpecialFormatter"}});
```

## Binding.execute()

### Disponibilidad

Flash Player 6.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myBinding.execute([reverse])
```

### Parámetros

*reverse* Valor booleano que especifica si la vinculación debe ejecutarse también desde el destino al origen (*true*) o sólo desde el origen al destino (*false*). El valor predeterminado es *false*.

### Valor devuelto

Un valor *null* si la vinculación se ha ejecutado correctamente; en caso contrario, devuelve una matriz de mensajes de error (cadenas) que describen el error, o errores, que han impedido la ejecución de la vinculación.

### Descripción

Método; recoge los datos del componente de origen y los asigna al componente de destino. Si la vinculación utiliza un formateador, se da formato a los datos antes de asignarlos al destino.

Este método también valida los datos y provoca un evento *valid* o *invalid* que deben emitir los componentes de origen y de destino. Los datos se asignan al destino aunque no sea válido, a no ser que el destino sea de sólo lectura.

Si el valor del parámetro *reverse* es *true* y la vinculación es bidireccional, ésta se ejecuta a la inversa (desde el destino al origen).

## Ejemplo

El código siguiente, asociado con una instancia de componente Button, ejecuta la vinculación a la inversa (desde el componente de destino al componente de origen) cuando se hace clic en el botón.

```
on(click){
    _root.myBinding.execute(true);
}
```

## Clase CustomFormatter (sólo para Flash Professional)

**Nombre de clase de ActionScript** mx.data.binding.CustomFormatter

La clase CustomFormatter define dos métodos, `format()` y `unformat()`, que proporcionan la capacidad de transformar valores de datos de un tipo de datos determinado en una cadena y viceversa. De forma predeterminada, estos métodos no realizan ninguna función. Debe implementarlos en una subclase de `mx.data.binding.CustomFormatter`.

Para crear su propio formateador personalizado, primero se ha de crear una subclase de CustomFormatter que implemente los métodos `format()` y `unformat()`. A continuación, asigne esta clase a una vinculación entre componentes mediante la creación de un objeto Binding con ActionScript (véase [“Clase Binding \(sólo para Flash Professional\)” en la página 132](#)) o a través de la ficha Vinculaciones del panel Inspector de componentes. Para más información sobre la asignación de una clase de formateador mediante el Inspector de componentes, consulte [“Formateadores de esquema \(sólo para Flash Professional\)” en el apartado Utilización de Flash de la Ayuda](#).

También puede asignar una clase de formateador a una propiedad de componente en la ficha Esquema del panel Inspector de componentes. En este caso, sin embargo, el formateador sólo se utilizará cuando los datos sean necesarios con formato de cadena. En cambio, los formateadores que se asignan mediante el panel Vinculaciones, o que se crean con ActionScript, se utilizan siempre que se ejecuta la vinculación.

Para obtener un ejemplo en el que se muestra cómo escribir y asignar un formateador personalizado mediante ActionScript, véase [“Ejemplo de formateador personalizado” en la página 135](#).

**Nota:** para que esta clase esté disponible en tiempo de ejecución, debe incluir el componente DataBindingClasses en el documento FLA. Para más información, consulte [“Trabajo con vinculación de datos y servicios Web en tiempo de ejecución \(sólo Flash Professional\)” en el apartado Utilización de Flash de la Ayuda](#).

Para ver información general sobre las clases del paquete `mx.data.binding`, consulte [“Clases de vinculación de datos \(sólo para Flash Professional\)” en la página 131](#).

## Ejemplo de formateador personalizado

En el ejemplo siguiente se muestra cómo crear una clase de formateador personalizado y aplicarla a una vinculación entre dos componentes mediante ActionScript. En este ejemplo, el valor actual de un componente NumericStepper (su propiedad `value`) está vinculado al valor actual de un componente TextInput (su propiedad `text`). La clase de formateador personalizado da formato al valor numérico actual del componente NumericStepper (por ejemplo, 1, 2 o 3) como su correspondiente en palabras en inglés (por ejemplo, “one”, “two” o “three”) antes de asignarlo al componente TextInput.

### Para crear y utilizar un formateador personalizado:

1 En Flash MX Professional 2004, cree un archivo ActionScript.

2 Añada el siguiente código al archivo:

```
// NumberFormatter.as
class NumberFormatter extends mx.data.binding.CustomFormatter {
    // Aplicar formato a un valor numérico, devolver una cadena
    function format(rawValue) {
        var returnValue;
        var strArray = new Array('one', 'two', 'three');
        var numArray = new Array(1, 2, 3);
        returnValue = 0;
        for (var i = 0; i < strArray.length; i++) {
            if (rawValue == numArray[i]) {
                returnValue = strArray[i];
                break;
            }
        }
        return returnValue;
    } // convertir un valor con formato, devuelve un valor sin formato
    function unformat(formattedValue) {
        var returnValue;
        var strArray = new Array('one', 'two', 'three');
        var numArray = new Array(1, 2, 3);
        returnValue = "invalid";
        for (var i = 0; i < strArray.length; i++) {
            if (formattedValue == strArray[i]) {
                returnValue = numArray[i];
                break;
            }
        }
        return returnValue;
    }
}
```

3 Guarde el archivo ActionScript como NumberFormatter.as.

4 Cree un documento de Flash (FLA).

5 Abra el panel Componentes (Ventana > Paneles de desarrollo > Componentes).

6 Arrastre un componente TextInput al escenario y asígnele el nombre **textInput**.

7 Arrastre un componente NumericStepper al escenario y asígnele el nombre **stepper**.

8 Abra la línea de tiempo (Ventana > Línea de tiempo) y seleccione el primer fotograma de la capa 1.

9 Abra el panel Acciones (Ventana > Paneles de desarrollo > Acciones).

10 Añada el código siguiente al panel Acciones:

```
import mx.data.binding.*;
var x:NumberFormatter;
var customBinding = new Binding({component:stepper, property:"value",
    event:"change"}, {component:textInput, property:"text",
    event:"enter.change"}, {cls:mx.data.formatters.Custom,
    settings:{classname:"NumberFormatter"}});
```

La segunda línea de código (var x:NumberFormatter) garantiza que el código de bytes de la clase de formateador personalizado se incluya en el archivo SWF compilado.

11 Seleccione Ventana > Paneles > Otros paneles > Clases para abrir la biblioteca Clases.

12 Abra la biblioteca del documento; para ello, seleccione Ventana > Biblioteca.

13 Arrastre el componente `DataBindingClasses` desde la biblioteca `Clases` hasta la biblioteca del documento.

De este modo, las clases en tiempo de ejecución de vinculación de datos están disponibles para el documento. Para más información, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

14 Guarde el archivo FLA en la misma carpeta que contiene `NumberFormatter.as`.

15 Pruebe el archivo (Control > Probar película).

Haga clic en los botones del componente `NumericStepper` y observe el contenido de la actualización del componente `TextInput`.

## Resumen de métodos de la clase `CustomFormatter`

| Método                                  | Descripción                                                    |
|-----------------------------------------|----------------------------------------------------------------|
| <code>CustomFormatter.format()</code>   | Convierte un tipo de datos sin formato en una cadena de texto. |
| <code>CustomFormatter.unformat()</code> | Convierte una cadena de texto en un tipo de datos sin formato. |

### `CustomFormatter.format()`

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

Este método se llama de forma automática, no se debe invocar directamente.

#### Parámetros

*rawData* Datos a los que se va a asignar formato.

#### Valor devuelto

Un valor con formato.

#### Descripción

Método; convierte un tipo de datos sin formato en un nuevo objeto.

Este método no se implementa de forma predeterminada. Debe definir este método en la subclase de `mx.data.binding.CustomFormatter`.

#### Ejemplo

Véase “Ejemplo de formateador personalizado” en la página 135.

## CustomFormatter.unformat()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Este método se llama de forma automática, no se debe invocar directamente.

### Parámetros

*formattedData* Datos con formato que deben reconvertirse en un tipo de datos sin formato.

### Valor devuelto

Un valor sin formato.

### Descripción

Método; convierte una cadena, u otro tipo de datos, en un tipo de datos sin formato. En esta transformación debe realizarse la transformación exactamente inversa de [CustomFormatter.format\(\)](#).

Este método no se implementa de forma predeterminada. Debe definir este método en la subclase de `mx.data.binding.CustomFormatter`.

Para más información, consulte [“Ejemplo de formateador personalizado” en la página 135](#).

## Clase CustomValidator (sólo para Flash Professional)

**Nombre de clase de ActionScript** `mx.data.binding.CustomValidator`

La clase `CustomValidator` se utiliza para realizar una validación personalizada de un campo de datos contenido en un componente.

Para crear una clase de validación personalizada, primero debe crearse una subclase de `mx.data.binding.CustomValidator` que implemente un método denominado `validate()`. Este método recibe automáticamente un valor para validarlo. Para más información sobre cómo implementar este método, consulte [CustomValidator.validate\(\)](#).

A continuación, debe asignarse una clase de validador personalizado a un campo de un componente mediante la ficha Esquema del panel Inspector de componentes. Para ver un ejemplo sobre la creación y la utilización de una clase de validador personalizado, consulte el apartado Ejemplo en la entrada de [CustomValidator.validate\(\)](#).

### Para asignar un validador personalizado:

- 1 En el panel Inspector de componentes (Ventana > Inspector de componentes), seleccione la ficha Esquema.
- 2 Seleccione el campo que desea validar y, a continuación, seleccione Personalizado en el menú emergente Tipo de datos.

- 3 Seleccione el campo Opciones de validación (en la parte inferior de la ficha Esquema) y haga clic en el icono de lupa para abrir el cuadro de diálogo Configuración de validación personalizada.
- 4 En el cuadro de texto Clase ActionScript, especifique el nombre de la clase de validación personalizada que ha creado.  
Para que la clase que ha especificado se incluya en el archivo SWF publicado, debe guardarla en la ruta de clases.

**Nota:** para que esta clase esté disponible en tiempo de ejecución, debe incluir el componente `DataBindingClasses` en el documento FLA. Para más información, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

Para ver información general sobre las clases del paquete `mx.data.binding`, consulte “[Clases de vinculación de datos \(sólo para Flash Professional\)](#)” en la [página 131](#).

## Resumen de métodos de la clase `CustomValidator`

| Método                                         | Descripción                              |
|------------------------------------------------|------------------------------------------|
| <code>CustomValidator.validate()</code>        | Lleva a cabo la validación de los datos. |
| <code>CustomValidator.validationError()</code> | Notifica errores de validación.          |

### `CustomValidator.validate()`

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

Este método se llama de forma automática, no se debe invocar directamente.

#### Parámetros

*value* Datos que se van a validar. Pueden ser de cualquier tipo.

#### Valor devuelto

Ninguno.

#### Descripción

Método; se llama de forma automática para validar los datos que contiene el parámetro *value*. Debe implementar este método en la subclase de `CustomValidator`; la implementación predeterminada no realiza ninguna acción.

Puede utilizar el código ActionScript de su elección para examinar y validar los datos. Si los datos no son válidos, este método debe llamar a `this.validationError()` con el mensaje pertinente. Puede llamar a `this.validationError()` más de una vez si se detectan varios problemas de validación de los datos.

Puesto que el método `validate()` puede llamarse repetidamente, procure no añadir código a este método que tarde mucho tiempo en realizarse. La implementación de este método sólo debe comprobar la validación y notificar cualquier error mediante `CustomValidator.validationError()`. Asimismo, la implementación no debe realizar ninguna acción como resultado de la prueba de validación, como por ejemplo advertir al usuario final. En su lugar, cree detectores de eventos para los eventos `valid` e `invalid` y advierta al usuario final desde estos detectores de eventos (consulte el ejemplo que se muestra a continuación).

## Ejemplo

En el siguiente procedimiento se muestra cómo crear y utilizar una clase de validación personalizada. El método `validate()` de la clase `CustomValidator`, `OddNumbersOnly.as`, define como no válido cualquier valor que no sea un número impar. La validación tiene lugar cada vez que se modifica el valor de un componente `NumericStepper`, que está vinculado a la propiedad `text` de un componente `Label`.

### Para crear y utilizar una clase de validador personalizado:

- 1 En Flash MX Professional 2004, cree un archivo ActionScript (AS).
- 2 Añada el siguiente código al archivo AS:

```
class OddNumbersOnly extends mx.data.binding.CustomValidator
{
    public function validate(value) {
        // asegurarse de que el valor es un número
        var n = Number(value);
        if (String(n) == "NaN") {
            this.validationError("'" + value + "' is not a number.");
            return;
        }
        // asegurarse de que el número es impar
        if (n % 2 == 0) {
            this.validationError("'" + value + "' is not a odd number.");
            return;
        }
        // los datos son correctos, no debe realizarse ninguna acción, sólo
        devolver
    }
}
```

- 3 Guarde el archivo AS como `OddNumbersOnly.as`.  
**Nota:** el nombre del archivo AS debe coincidir con el nombre de la clase.
- 4 Cree un documento de Flash (FLA).
- 5 Abra el panel Componentes (Ventana > Paneles de desarrollo > Componentes).
- 6 Arrastre un componente `NumericStepper` desde el panel Componentes al escenario y asígnele el nombre **stepper**.
- 7 Arrastre un componente `Label` al escenario y asígnele el nombre **textLabel**.
- 8 Arrastre un componente `TextArea` al escenario y denomínelo **status**.
- 9 Seleccione el componente `NumericStepper` y abra el panel Inspector de componentes (Ventana > Paneles de desarrollo > Inspector de componentes).
- 10 Seleccione la ficha Vinculaciones en el panel Inspector de componentes y haga clic en el botón Añadir vinculación (+).



- 11 Seleccione la propiedad `Value` (la única) en el cuadro de diálogo Añadir vinculaciones y haga clic en Aceptar.
- 12 En el panel Inspector de componentes, haga doble clic en `bound to` en el panel Atributos de vinculación de la ficha Vinculaciones para abrir el cuadro de diálogo Vinculado a.
- 13 En el cuadro de diálogo Vinculado a, seleccione el componente Label del panel Ruta del componente y su propiedad `text` en el panel Ubicación del esquema. Haga clic en Aceptar.
- 14 Seleccione el componente Label del escenario y haga clic en la ficha Esquema del panel Inspector de componentes.
- 15 En el panel Atributos de esquema, seleccione Custom en el menú emergente data type.
- 16 Haga doble clic en el campo validation options del panel Atributos de esquema para abrir el cuadro de diálogo Configuración de validación personalizada.
- 17 En el cuadro de texto Clase `ActionScript`, especifique **OddNumbersOnly**, que es el nombre de la clase `ActionScript` que ha creado anteriormente. Haga clic en Aceptar.
- 18 Abra la línea de tiempo (Ventana > Línea de tiempo) y seleccione el primer fotograma de la capa 1.
- 19 Abra el panel Acciones (Ventana > Acciones).
- 20 Añada el código siguiente al panel Acciones:
 

```
function dataIsValid(evt) {
    if (evt.property == "text") {
        status.text = evt.messages;
    }
}

function dataIsValid(evt) {
    if (evt.property == "text") {
        status.text = "OK";
    }
}

textLabel.addEventListener("valid", dataIsValid);
textLabel.addEventListener("invalid", dataIsValid);
```
- 21 Guarde el archivo FLA como `OddOnly fla` en la misma carpeta que contiene `OddNumbersOnly.as`.
- 22 Pruebe el archivo SWF (Control > Probar película).
 

Haga clic en las flechas del componente `NumericStepper` para modificar su valor. Observe el mensaje que aparece en el componente `TextArea` cuando selecciona números pares e impares.

## CustomValidator.validationError()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
this.validationError(errorMessage)
```

**Nota:** este método sólo puede invocarse desde el interior de una clase de validador personalizado; la palabra clave `this` hace referencia al objeto `CustomValidator` actual.

## Parámetros

*errorMessage* Cadena que contiene el mensaje de error que debe notificarse.

## Valor devuelto

Ninguno.

## Descripción

Método; debe llamar a este método desde el método `validate()` de su subclase de `CustomValidator` para notificar los errores de validación. Si no llama a este método, se genera un evento `valid` cuando finaliza `validate()`. Si llama a este método una o varias veces desde el interior del método `validate()`, se genera un evento `invalid` después de que `validate()` devuelva un valor.

Todos los mensajes que se pasan a `validationError()` están disponibles en la propiedad “messages” del objeto de evento que se ha pasado al controlador de eventos `invalid`.

## Ejemplo

Véase el apartado Ejemplo de `CustomValidator.validate()`.

## Clase EndPoint (sólo para Flash Professional)

**Nombre de clase de ActionScript** `mx.data.binding.EndPoint`

La clase `EndPoint` define el origen o el destino de una vinculación. Los objetos `EndPoint` definen un valor constante, una propiedad de componente o un campo concreto de una propiedad de componente desde los que se pueden obtener datos o a los que se pueden asignar datos. También pueden definir un evento, o una lista de eventos, que detecta un objeto `Binding`; cuando el evento especificado tiene lugar, se ejecuta la vinculación.

Cuando se crea una vinculación con el constructor de la clase `Binding`, se pasan dos objetos `EndPoint`: uno para el origen y uno para el destino.

```
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

Los objetos `EndPoint`, `srcEndPoint` y `destEndPoint`, se pueden definir de la siguiente manera:

```
var srcEndPoint = new mx.data.binding.EndPoint();
var destEndPoint = new mx.data.binding.EndPoint();
srcEndPoint.component = source_txt;
srcEndPoint.property = "text";
srcEndPoint.event = "focusOut";
destEndPoint.component = dest_txt;
destEndPoint.property = "text";
```

El código anterior significa que “cuando el texto de origen deja de estar seleccionado, debe copiarse el valor de su propiedad `text` en la propiedad `text` del campo de texto de destino”.

También puede pasar objetos `ActionScript` genéricos al constructor `Binding`, en lugar de pasar objetos `EndPoint` contruidos explícitamente. El único requisito es que los objetos definan las propiedades de `EndPoint` necesarias, concretamente `component` y `property`. El código siguiente es equivalente al mostrado anteriormente.

```
var srcEndPoint = {component:source_txt, property:"text"};
var destEndPoint = {component:dest_txt, property:"text"};
new mx.data.binding.Binding(srcEndPoint, destEndPoint);
```

**Nota:** para que esta clase esté disponible en tiempo de ejecución, debe incluir el componente `DataBindingClasses` en el documento FLA. Para más información, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

Para ver información general sobre las clases del paquete `mx.data.binding`, consulte “[Clases de vinculación de datos \(sólo para Flash Professional\)](#)” en la [página 131](#).

## Resumen de propiedades de la clase `EndPoint`

| Método                          | Descripción                                                                                                     |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>EndPoint.constant</code>  | Valor constante.                                                                                                |
| <code>EndPoint.component</code> | Referencia a una instancia de componente.                                                                       |
| <code>EndPoint.property</code>  | Nombre de una propiedad de la instancia de componente especificada por <code>EndPoint.component</code> .        |
| <code>EndPoint.location</code>  | Ubicación de un campo de datos dentro de la propiedad de la instancia de componente.                            |
| <code>EndPoint.event</code>     | Nombre de un evento, o lista de eventos, que la instancia de componente emitirá cuando se modifiquen los datos. |

## Constructor de la clase `EndPoint`

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
new EndPoint()
```

### Valor devuelto

Ninguno.

### Descripción

Constructor; crea un objeto `EndPoint`.

### Ejemplo

En este ejemplo se crea un objeto `EndPoint` denominado `source_txt` y se asignan valores a sus propiedades `component` y `property`.

```
var source_obj = new mx.data.binding.EndPoint();
source_obj.component = myTextField;
source_obj.property = "text";
```

## EndPoint.constant

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*endPoint\_src.constant*

### Descripción

Propiedad; valor constante asignado a un objeto EndPoint. Esta propiedad sólo se puede aplicar a EndPoints que son el origen, y no el destino, de una vinculación entre componentes. El valor puede ser cualquier tipo de datos compatible con el destino de la vinculación. Si así se especifica, se pueden ignorar todas las demás propiedades de EndPoint del objeto EndPoint especificado.

### Ejemplo

En este ejemplo, el valor constante de la cadena “hello” se asigna a la propiedad constante de un objeto EndPoint.

```
var sourceEndPoint = new mx.data.binding.EndPoint();
sourceEndPoint.constant="hello";
```

## EndPoint.component

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*endPointObj.component*

### Descripción

Propiedad; referencia a una instancia de componente.

### Ejemplo

En este ejemplo se asigna una instancia del componente List (*listBox1*) como parámetro de componente de un objeto EndPoint.

```
var sourceEndPoint = new mx.data.binding.EndPoint();
sourceEndPoint.component=listBox1;
```

## EndPoint.property

### Disponibilidad

Flash Player 6 versión 79

### Edición

Flash MX Professional 2004.

### Sintaxis

*endPointObj.property*

### Descripción

Propiedad; especifica un nombre de propiedad de la instancia de componente especificada por [EndPoint.component](#) que contiene los datos vinculables.

**Nota:** [EndPoint.component](#) y [EndPoint.property](#) deben combinarse para formar una combinación válida de objeto/propiedad ActionScript.

### Ejemplo

En este ejemplo se vincula la propiedad `text` de un componente `TextInput` (`text_1`) a la misma propiedad de otro componente `TextInput` (`text_2`).

```
var sourceEndPoint = {component:text_1, property:"text"};
var destEndPoint = {component:text_2, property:"text"};
new Binding(sourceEndPoint, destEndPoint);
```

## EndPoint.location

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*endPointObj.location*

### Descripción

Propiedad; especifica la ubicación de un campo de datos en la propiedad de la instancia de componente. Existen cuatro maneras de especificar una ubicación: como cadena que contiene una expresión XPath o una ruta ActionScript, como matriz de cadenas o como objeto.

Las expresiones XPath sólo se pueden utilizar cuando los datos son un objeto XML. Para ver una lista de expresiones XPath admitidas, consulte “Expresiones XPath admitidas” en el apartado Utilización de Flash de la Ayuda. Véase el ejemplo 1 que se muestra más adelante.

Para los objetos XML y ActionScript, también puede especificar una cadena que contenga una ruta ActionScript. Una ruta ActionScript contiene los nombres de los campos separados por puntos (por ejemplo, “a.b.c”).

También puede especificar una matriz de cadenas como ubicación. Cada cadena de la matriz pasa a otro nivel de anidación. Puede utilizar esta técnica tanto con los datos XML como con los datos ActionScript. Véase el ejemplo 2 que se muestra más adelante. Cuando se utiliza con datos ActionScript, una matriz de cadenas es equivalente a utilizar código ActionScript; es decir, la matriz ["a", "b", "c"] es equivalente a "a.b.c".

Si especifica un objeto como ubicación, el objeto debe especificar dos propiedades: `path` e `indices`. La propiedad `path` es una matriz de cadenas, como se ha indicado anteriormente, excepto que una o varias de las cadenas especificadas puede ser la ficha especial "[n]". Para cada aparición de esta ficha en `path`, debe haber el elemento de índice correspondiente en `indices`. Mientras se evalúa la ruta, los índices se utilizan para crear índices en las matrices. El elemento de índice puede ser cualquier `EndPoint`. Este tipo de ubicación sólo se puede aplicar a los datos ActionScript, no a XML (véase el ejemplo 3 que aparece más adelante).

## Ejemplo

**Ejemplo 1:** en este ejemplo se utiliza una expresión XPath para especificar la ubicación de un nodo denominado `zip` en un objeto XML.

```
var sourceEndPoint = new mx.databinding.EndPoint();
var sourceObj=new Object();
sourceObj.xml=new XML("<zip>94103</zip>");
sourceEndPoint.component=sourceObj;
sourceEndPoint.property="xml";
sourceEndPoint.location="/zip";//
```

**Ejemplo 2:** en este ejemplo se utiliza una matriz de cadenas para pasar a una propiedad de un clip de película anidado.

```
var sourceEndPoint = new mx.data.binding.EndPoint();
//presuponer que existe movieClip1.ball.position
sourceEndPoint.component=movieClip1;
sourceEndPoint.property="ball";
//acceder a movieClip1.ball.position.x
sourceEndPoint.location=["position","x"];
```

**Ejemplo 3:** en este ejemplo se muestra cómo utilizar un objeto para especificar la ubicación de un campo de datos en una estructura de datos compleja.

```
var city=new Object();
city.theaters = [{theater: "t1", movies: [{name: "Good,Bad,Ugly"},
    {name:"Matrix Reloaded"}]}, {theater: "t2", movies: [{name: "Gladiator"},
    {name: "Catch me if you can"}]};
var srcEndPoint = new EndPoint();
srcEndPoint.component=city;
srcEndPoint.property="theaters";
srcEndPoint.location = {path: ["[n]","movies","[n]","name"], indices:
    [{constant:0},{constant:0}]};
```

## EndPoint.event

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*endPointObj.event*

### Descripción

Propiedad; especifica el nombre de un evento, o una matriz de nombres de evento, generado por el componente cuando se modifican los datos asignados a la propiedad vinculada. Cuando tiene lugar un evento, se ejecuta la vinculación.

El evento especificado sólo se aplica a los componentes que se utilizan como origen de una vinculación o como destino de una vinculación bidireccional. Para más información sobre la creación de vinculaciones bidireccionales, consulte [“Clase Binding \(sólo para Flash Professional\)” en la página 132](#).

### Ejemplo

En este ejemplo, la propiedad `text` de un componente `TextInput` (`src_txt`) se vincula a la misma propiedad de otro componente `TextInput` (`dest_txt`). La vinculación se ejecuta cuando el componente `src_txt` emite el evento `focusOut` o `enter`.

```
var source = {component:src_txt, property:"text", event:["focusOut",
    "enter"]};
var dest = {component:myTextArea, property:"text"};
var newBind = new mx.data.binding.Binding(source, dest);
```

## Clase ComponentMixins (sólo para Flash Professional)

**Nombre de clase de ActionScript**   `mx.data.binding.ComponentMixins`

La clase `ComponentMixins` define las propiedades y los métodos que se añaden automáticamente a cualquier objeto que sea origen o destino de una vinculación o a cualquier componente que sea destino de una llamada del método `ComponentMixins.initComponent()`. Estas propiedades y estos métodos no afectan a las funciones normales de los componentes, sino que añaden funciones útiles para la vinculación de datos.

**Nota:** para que esta clase esté disponible en tiempo de ejecución, debe incluir el componente `DataBindingClasses` en el documento FLA. Para más información, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

Para ver información general sobre las clases del paquete `mx.data.binding`, consulte [“Clases de vinculación de datos \(sólo para Flash Professional\)” en la página 131](#).

## Resumen de métodos de la clase ComponentMixins

| Método                                             | Descripción                                                                                                                   |
|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <code>ComponentMixins.getField()</code>            | Devuelve un objeto para obtener y configurar el valor de un campo en una ubicación específica de una propiedad de componente. |
| <code>ComponentMixins.initComponent()</code>       | Añade los métodos ComponentMixin a un componente.                                                                             |
| <code>ComponentMixins.refreshFromSources()</code>  | Ejecuta todas las vinculaciones que tengan a este componente como EndPoint de destino.                                        |
| <code>ComponentMixins.refreshDestinations()</code> | Ejecuta todas las vinculaciones que tengan a este objeto como EndPoint de origen.                                             |
| <code>ComponentMixins.validateProperty()</code>    | Comprueba si los datos de la propiedad indicada son válidos.                                                                  |

### ComponentMixins.getField()

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

*componentInstance.getField(propertyName, [location])*

#### Parámetros

*propertyName* Cadena que contiene el nombre de una propiedad del componente especificado.

*location* (opcional) Ubicación de un campo dentro de la propiedad de componente. Este parámetro resulta de utilidad si la propiedad de componente especificada por *propertyName* es una estructura compleja de datos y le interesa un campo determinado de dicha estructura. Esta propiedad puede tener una de las tres formas siguientes:

- Una cadena que contiene una expresión XPath. Esto es sólo válido para estructuras de datos XML. Para ver una lista de expresiones XPath admitidas, consulte “Expresiones XPath admitidas” en el apartado Utilización de Flash de la Ayuda.
- Una cadena que contiene nombres de campo, separados por puntos, por ejemplo "a.b.c". Esta forma puede utilizarse con datos complejos (ActionScript o XML).
- Una matriz de cadenas, en la que cada cadena es un nombre de campo, por ejemplo ["a", "b", "c"]. Esta forma puede utilizarse con datos complejos (ActionScript o XML).

#### Valor devuelto

Un objeto `DataType`.



## Descripción

Método; devuelve un objeto `DataType` cuyos métodos pueden utilizarse para obtener o establecer el valor de datos en la propiedad de componente en la ubicación de campo especificada. Para más información, consulte [“Clase `DataType` \(sólo para Flash Professional\)” en la página 153](#).

## Ejemplo

Este ejemplo utiliza el método `DataType.setAsString()` para establecer el valor de un campo ubicado en la propiedad de un componente. En este caso, la propiedad (`results`) es una estructura de datos complejos.

```
import mx.data.binding.*;
var field : DataType = myComponent.getField("results", "po.address.name1");
field.setAsString("Teri Randall");
```

## Véase también

[`DataType.setAsString\(\)`](#)

## ComponentMixins.initComponent()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
mx.data.binding.ComponentMixins.initComponent(componentInstance)
```

### Parámetros

*componentInstance*    Referencia a una instancia de componente.

### Valor devuelto

Ninguno.

### Descripción

Método (estático); añade todos los métodos `ComponentMixins` al componente especificado por *componentInstance*. Se llama automáticamente a este método para todos los componentes implicados en una vinculación de datos. Para que los métodos `ComponentMixins` estén disponibles para un componente no implicado en una vinculación de datos, debe llamar explícitamente a este método para dicho componente.

### Ejemplo

El código siguiente hace que los métodos `ComponentMixins` estén disponibles para un componente `DataSet`.

```
mx.data.binding.ComponentMixins.initComponent(_root.myDataSet);
```

## ComponentMixins.refreshFromSources()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*componentInstance*.refreshSources()

### Valor devuelto

Ninguno.

### Descripción

Método; ejecuta todas las vinculaciones para las que *componentInstance* es el objeto EndPoint de destino. Este método permite ejecutar vinculaciones que tienen fuentes constantes o fuentes que no emiten ningún evento de modificación de datos.

### Ejemplo

El ejemplo siguiente ejecuta todas las vinculaciones para las que la instancia de componente ListBox denominada *cityList* es el objeto EndPoint de destino.

```
cityList.refreshFromSources();
```

## ComponentMixins.refreshDestinations()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*componentInstance*.refreshDestinations()

### Valor devuelto

Ninguno.

### Descripción

Método; ejecuta todas las vinculaciones para las que *componentInstance* es el EndPoint de origen. Este método permite ejecutar vinculaciones cuyas fuentes no emiten ningún evento de modificación de datos.

### Ejemplo

En el ejemplo siguiente se ejecutan todas las vinculaciones para las que la instancia del componente DataSet denominada *user\_data* es el objeto EndPoint de origen.

```
user_data.refreshDestinations();
```

## ComponentMixins.validateProperty()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.validateProperty(propertyName)
```

### Parámetros

*propertyName* Cadena que contiene el nombre de una propiedad que pertenece a *componentInstance*.

### Valor devuelto

Una matriz o el valor `null`.

### Descripción

Método; determina si los datos de *propertyName* son válidos según la configuración de esquema de la propiedad. Las configuraciones de esquema son las que se especifican en la ficha Esquema del panel Inspector de componentes.

El método devuelve el valor `null` si los datos son válidos; de lo contrario, devuelve una matriz de mensajes de error en forma de cadenas.

La validación sólo se aplica a los campos cuya información de esquema se encuentra disponible. Si un campo es un objeto que contiene otros campos, se validará cada campo “secundario” y así sucesivamente. Cada campo individual distribuirá un evento `valid` o `invalid`, según sea necesario. Para cada campo de datos que contenga *propertyName*, esta función distribuirá eventos `valid` o `invalid` de la forma siguiente:

- Si el valor del campo es `null` y *no* es obligatorio, el método devolverá el valor `null`. No se genera ningún evento.
- Si el valor del campo es `null` y *es* obligatorio, se devolverá un error y se generará un evento `invalid`.
- Si el valor del campo no es `null` y el esquema del campo *no* tiene un validador, el método devolverá el valor `null`; no se generará ningún evento.
- Si el valor no es `null` y el esquema del campo *sí* define un validador, el objeto del validador procesará los datos. Si los datos son válidos, se generará un evento `valid` y se devolverá el valor `null`; de lo contrario, se generará un evento `invalid` y se devolverá una matriz de cadenas de error.

### Ejemplo

En los ejemplos siguientes se muestra cómo utilizar `validateProperty()` para garantizar que la longitud del texto introducido por el usuario sea válida. Para determinar la longitud válida, deberá establecer las opciones de validación de la cadena `DataType` en la ficha Esquema del panel Inspector de componentes. Si el usuario introduce una cadena con una longitud no válida en el campo de texto, los mensajes de error que devuelva el método `validateProperty()` aparecerán en el panel Salida.

**Para validar el texto introducido por un usuario en un componente TextInput:**

- 1 Arrastre un componente TextInput desde el panel Componentes (Ventana > Paneles de desarrollo > Componentes) hasta el escenario y denomínelo `zipCode_txt`.
- 2 Seleccione el componente TextInput y, en el panel Inspector de componentes (Ventana > Paneles de desarrollo > Componentes), haga clic en la ficha Esquema.
- 3 En el panel Árbol de esquema (panel superior de la ficha Esquema) seleccione la propiedad `text`.
- 4 En el panel Atributos de esquema (panel inferior de la ficha Esquema), seleccione ZipCode en el menú emergente data type.
- 5 Para abrir la línea de tiempo, si todavía no está abierta, seleccione Ventana > Línea de tiempo.
- 6 Haga clic en el primer fotograma de la capa 1 de la línea de tiempo y abra el panel Acciones (Ventana > Acciones).

- 7 Añada el código siguiente al panel Acciones:

```
// Añada métodos ComponentMixin al componente TextInput.
// Tenga en cuenta que es necesario realizar este paso sólo si el componente
// todavía no forma parte de una vinculación de datos,
// ya sea como origen o destino.
mx.data.binding.ComponentMixins.initComponent(zipCode_txt);
// Defina la función de detector de eventos para el componente:
validateResults = function (eventObj) {
    var errors:Array = eventObj.target.validateProperty("text");
    if (errors != null) {
        trace(errors);
    }
};
// Registre la función de detector con el componente:
zipCode_txt.addEventListener("enter", validateResults);
```

- 8 Seleccione Ventana > Otros paneles > Bibliotecas comunes > Clases para abrir la biblioteca Clases.
- 9 Para abrir la biblioteca del documento, seleccione Ventana > Biblioteca.
- 10 Arrastre el componente DataBindingClasses desde la biblioteca Clases hasta el panel Biblioteca del documento.

Este paso es necesario para que se pueda acceder desde el archivo SWF a las clases de tiempo de ejecución de la vinculación de datos durante el tiempo de ejecución. Para más información, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

- 11 Para probar el archivo SWF seleccione Control > Probar película.

En el componente TextInput del escenario, introduzca un código no válido de Estados Unidos, por ejemplo, uno que contenga sólo letras o uno que contenga menos de cinco números. Observe los mensajes de error que aparecen en el panel Salida.

## Clase `DataType` (sólo para Flash Professional)

**Nombre de clase de ActionScript** `mx.data.binding.DataType`

La clase `DataType` ofrece un acceso de lectura y escritura a los campos de datos de una propiedad de componente. Para obtener un objeto `DataType`, llame a la función `ComponentMixins.getField()` de un componente. A continuación, puede llamar a métodos del objeto `DataType` para obtener y definir el valor del campo.

La diferencia entre obtener y definir valores de campo mediante métodos de objeto `DataType` y obtener y definir los mismos valores directamente en la instancia de componente es que en el segundo caso los datos aparecen sin formato. En cambio, al obtener o definir valores de campo mediante métodos de la clase `DataType`, dichos valores se procesan según la configuración de esquema del campo.

Por ejemplo, mediante el código siguiente se obtiene el valor de una propiedad de componente directamente, que se asigna a una variable. La variable, `propVar`, contiene el valor actual sin formato de la propiedad `propName`.

```
var propVar = myComponent.propName;
```

En el ejemplo siguiente se obtiene el valor de la misma propiedad mediante el método `DataType.getAsString()`. En este caso, el valor asignado a `stringVar` es el valor de `propName` una vez que se haya procesado según la configuración de su esquema y, a continuación, se devuelve en forma de cadena.

```
var dataTypeObj:mx.data.binding.DataType = myComponent.getField("propName");  
var stringVar: String = dataTypeObj.getAsString();
```

Para más información sobre cómo especificar la configuración de esquema de un campo, consulte “Trabajo con esquemas en la ficha Esquema (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

También puede utilizar los métodos de la clase `DataType` para obtener o definir campos en diferentes tipos de datos. La clase `DataType` convierte automáticamente los datos sin formato al tipo deseado, si es posible. Por ejemplo, en el ejemplo anterior de códigos, los datos recuperados se convierten en el tipo `String`, aunque los datos sin formato sean de otro tipo.

El método `ComponentMixins.getField()` se encuentra disponible para los componentes que se han incluido en una vinculación de datos (ya sea origen, destino o índice) o que se han inicializado mediante el método `ComponentMixins.initComponent()`. Para más información, consulte “Clase `ComponentMixins` (sólo para Flash Professional)” en la página 147.

**Nota:** para que esta clase esté disponible en tiempo de ejecución, debe incluir el componente `DataBindingClasses` en el documento FLA. Para más información, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

Para ver información general sobre las clases del paquete `mx.data.binding`, consulte “Clases de vinculación de datos (sólo para Flash Professional)” en la página 131.

## Resumen de métodos de la clase `DataType`

| Método                                   | Descripción                                                                                 |
|------------------------------------------|---------------------------------------------------------------------------------------------|
| <code>DataType.getAsBoolean()</code>     | Recoge el valor actual del campo como booleano.                                             |
| <code>DataType.getAsNumber()</code>      | Recoge el valor actual del campo como número.                                               |
| <code>DataType.getAsString()</code>      | Recoge el valor actual del campo como valor de cadena.                                      |
| <code>DataType.getAnyTypedValue()</code> | Recoge el valor actual del campo.                                                           |
| <code>DataType.getTypedValue()</code>    | Recoge el valor actual del campo con el formato de la clase <code>DataType</code> indicada. |
| <code>DataType.setAnyTypedValue()</code> | Establece un nuevo valor en el campo.                                                       |
| <code>DataType.setAsBoolean()</code>     | Establece el campo según el nuevo valor, que se proporciona en forma de booleano.           |
| <code>DataType.setAsNumber()</code>      | Establece el campo según el nuevo valor, que se proporciona en forma de número.             |
| <code>DataType.setAsString()</code>      | Establece el campo según el nuevo valor, que se proporciona en forma de cadena.             |
| <code>DataType.setTypedValue()</code>    | Establece un nuevo valor en el campo.                                                       |

## Resumen de propiedades de la clase `DataType`

| Propiedad                       | Descripción                                                                               |
|---------------------------------|-------------------------------------------------------------------------------------------|
| <code>DataType.encoder</code>   | Proporciona una referencia para el objeto <code>Encoder</code> asociado con este campo.   |
| <code>DataType.formatter</code> | Proporciona una referencia para el objeto <code>Formatter</code> asociado con este campo. |
| <code>DataType.kind</code>      | Proporciona una referencia para el objeto <code>Kind</code> asociado con este campo.      |

## `DataType.encoder`

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`dataTypeObject.encoder`

## Descripción

Propiedad; proporciona una referencia para el objeto encoder asociado con este campo, en caso de que exista. Puede utilizar esta propiedad para acceder a todas las propiedades y los métodos definidos mediante el codificador específico aplicado al campo de la ficha Esquema en el panel Inspector de componentes.

Si no se ha aplicado ningún codificador al campo en cuestión, esta propiedad devolverá el valor `undefined`.

Para más información sobre los codificadores que se proporcionan con Flash MX Professional 2004, consulte “Codificadores de esquema (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

## Ejemplo

En el ejemplo siguiente se supone que el campo al que se accede (`isValid`) utiliza un codificador Boolean (`mx.data.encoders.Bool`). Este codificador se proporciona con Flash MX Professional 2004 y contiene una propiedad denominada `trueStrings` que especifica las cadenas que deben interpretarse como valores booleanos `true`. El código siguiente establece que la propiedad `trueStrings` del codificador de un campo sean las cadenas “yes” y “sí”.

```
var myField:mx.data.binding.DataType = dataSet.getField("isValid");
myField.encoder.trueStrings = "Yes,Sí";
```

## **DataType.formatter**

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*dataTypeObject.formatter*

## Descripción

Propiedad; proporciona una referencia para el objeto formateador asociado con este campo, en caso de que exista. Puede utilizar esta propiedad para acceder a todas las propiedades y métodos del objeto formateador que se aplica al campo en la ficha Esquema del panel Inspector de componentes.

Si no se ha aplicado ningún formateador al campo en cuestión, entonces la propiedad devolverá el valor `undefined`.

Para más información sobre los formateadores que se proporcionan con Flash MX Professional 2004, consulte “[Formateadores de esquema \(sólo para Flash Professional\)](#)” en el apartado Utilización de Flash de la Ayuda.

## Ejemplo

En este ejemplo se supone que el campo al que se accede está utilizando el formateador de número (`mx.data.formatters.NumberFormatter`) que se proporciona con Flash MX Professional 2004. Este formateador contiene una propiedad denominada `precision` que especifica el número de dígitos que deben visualizarse después de una coma decimal. Este código establece la propiedad `precision` en dos cifras decimales para el campo que utiliza dicho formateador.

```
var myField:DataType = dataGrid.getField("currentBalance");
myField.formatter.precision = 2;
```

## DataType.getAsBoolean()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataTypeObject.getAsBoolean()
```

### Valor devuelto

Valor booleano.

### Descripción

Método; recoge el valor actual del campo en forma de booleano. El valor se convierte en booleano, en caso necesario.

## Ejemplo

En este ejemplo, el campo denominado `propName` que pertenece a un componente denominado `myComponent` se recupera como valor booleano y se asigna a una variable.

```
var dataTypeObj:mx.data.binding.DataType = myComponent.getField("propName");
var propValue:Boolean = dataTypeObj.getAsBoolean();
```

## DataType.getAsNumber()

### Disponibilidad

Flash Player 6.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataTypeObject.getAsNumber()
```

### Valor devuelto

Un número.



### Descripción

Método; recoge el valor actual del campo en forma de número. El valor se convierte en número, en caso necesario.

### Ejemplo

En este ejemplo, el campo denominado `propName` que pertenece a un componente denominado `myComponent` se recupera en forma de número y se asigna a una variable.

```
var dataTypeObj:mx.data.binding.DataType = myComponent.getField("propName");  
var propValue:Number = dataTypeObj.getAsNumber();
```

### Véase también

[DataType.getAnyTypedValue\(\)](#)

## DataType.getAsString()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataTypeObject.getAsString()
```

### Valor devuelto

Una cadena.

### Descripción

Método; recoge el valor actual del campo en forma de cadena. El valor se convierte en cadena, en caso necesario.

### Ejemplo

En este ejemplo, una propiedad del componente denominado `propName` que pertenece a un componente denominado `myComponent` se recupera en forma de cadena y se asigna a una variable.

```
var dataTypeObj:mx.data.binding.DataType = myComponent.getField("propName");  
var propValue:String = dataTypeObj.getAsString();
```

### Véase también

[DataType.getAnyTypedValue\(\)](#)

## DataType.getAnyTypedValue()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataTypeObject.getAnyTypedValue(suggestedTypes)
```

### Parámetros

*suggestedTypes* Matriz de cadenas que especifica, en el orden descendente que desee, los tipos de datos preferidos para el campo. Para más información, consulte la sección Descripción que aparece a continuación.

### Valor devuelto

Valor actual del campo, con el formato de uno de los tipos de datos especificados en la matriz *suggestedTypes*.

### Descripción

Método; recoge el valor actual del campo, utilizando la información del esquema del campo, para procesar el valor. Si el campo puede proporcionar un valor como primer tipo de datos especificado en la matriz *suggestedTypes*, el método devolverá el valor del campo en forma de tipo de datos. De lo contrario, el método intentará extraer el valor del campo como segundo tipo de datos especificado en la matriz *suggestedTypes*, etc.

Si especifica `null` como uno de los elementos de la matriz *suggestedTypes*, el método devolverá el valor del campo del tipo de datos especificado en el panel Esquema. Si especifica `null` siempre se devolverá un valor; por lo tanto, utilice `null` sólo al final de la matriz.

Si un valor no puede devolverse con el formato de uno de los tipos sugeridos, se devolverá en el tipo especificado en el panel Esquema.

### Ejemplo

En este ejemplo se intenta obtener el valor de un campo (`productInfo.available`) en la propiedad `results` de un componente `XMLConnector`, primero como un número y, si no funciona, como una cadena.

```
import mx.data.binding.DataType;
import mx.data.binding.TypedValue;
var f: DataType = myXmlConnector.getField("results", "productInfo.available");
var b: TypedValue = f.getAnyTypedValue(["Number", "String"]);
```

### Véase también

[ComponentMixins.getField\(\)](#)

## DataType.getTypedValue()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataTypeObject.getTypedValue(requestedType)
```

### Parámetros

*requestedType* Cadena que contiene el nombre de un tipo de datos o `null`.

### Valor devuelto

Objeto TypedValue (véase [“Clase TypedValue \(sólo para Flash Professional\)” en la página 163](#))

### Descripción

Método; devuelve el valor del campo en la forma especificada por *requestedType*, en caso de que se haya especificado y de que el campo pueda proporcionar este valor en dicha forma. Si el campo no puede proporcionar su valor en la forma indicada, el método devolverá `null`.

Si se especifica `null` como *requestedType*, el método devolverá el valor del campo en su tipo predeterminado.

### Ejemplo

```
var bool:TypedValue = field.getTypedValue("Boolean");
```

## DataType.kind

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataTypeObject.kind
```

### Descripción

Propiedad; proporciona una referencia para el objeto Kind asociado con este campo. Puede utilizar esta propiedad para acceder a propiedades y métodos del objeto Kind.

## DataType.setAnyTypedValue()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataTypeObject.setAnyTypedValue(newTypedValue)
```

### Parámetros

*newValue* Valor del objeto TypedValue que debe definirse en el campo.

Para más información acerca de los objetos TypedValue, consulte [“Clase TypedValue \(sólo para Flash Professional\)” en la página 163](#).

### Valor devuelto

Matriz de cadenas que describen los errores que se producen mientras se intenta definir un valor nuevo. Los errores pueden producirse en una de las condiciones siguientes:

- Los datos proporcionados no pueden convertirse en el tipo de datos de este campo (por ejemplo, "abc" no puede convertirse en un tipo de datos Number).
- El tipo de los datos es correcto pero no cumple con los criterios de validación del campo.
- El campo es de sólo lectura.

**Nota:** el texto real de los mensajes varía en función del tipo de datos, los formateadores y los codificadores que se definen en el esquema del campo.

### Descripción

Método; define un valor nuevo en el campo, utilizando la información del esquema del campo para procesar el campo.

Este método funciona llamando primero a `DataType.setTypedValue()` para establecer el valor. Si no funciona, el método comprueba si el objeto de destino aceptará datos String, Boolean o Number; en tal caso, los intentos de utilizar la conversión en ActionScript funcionarán correctamente.

### Ejemplo

En este ejemplo se crea un nuevo objeto TypedValue (valor Boolean) y, a continuación, se asigna dicho valor al objeto DataType denominado `field`. Todos los errores que se produzcan se asignan a la matriz `errors`.

```
import mx.data.binding.*;
var t:TypedValue = new TypedValue (true, "Boolean");
var errors: Array = field.setAnyTypedValue (t);
```

### Véase también

[DataType.setTypedValue\(\)](#)

## **DataType.setAsBoolean()**

### **Disponibilidad**

Flash Player 6 versión 79.

### **Edición**

Flash MX Professional 2004.

### **Sintaxis**

```
dataTypeObject.setAsBoolean(newBooleanValue)
```

### **Parámetros**

*newBooleanValue*    Valor booleano.

### **Valor devuelto**

Ninguno.

### **Descripción**

Método; establece el campo según el valor nuevo, que se proporciona como un valor booleano. El valor se convierte en el tipo de datos adecuado para este campo y se almacena como tal.

### **Ejemplo**

```
var bool: Boolean = true;  
field.setAsBoolean (bool);
```

## **DataType.setAsNumber()**

### **Disponibilidad**

Flash Player 6 versión 79.

### **Edición**

Flash MX Professional 2004.

### **Sintaxis**

```
dataTypeObject.setAsNumber(newNumberValue)
```

### **Parámetros**

*newNumberValue*    Número.

### **Valor devuelto**

Ninguno.

### **Descripción**

Método; establece el campo según el valor nuevo, que se proporciona en forma de número. El valor se convierte en el tipo de datos adecuado para este campo y se almacena como tal.

### **Ejemplo**

```
var num: Number = 32;  
field.setAsNumber (num);
```

## **DataType.setAsString()**

### **Disponibilidad**

Flash Player 6 versión 79.

### **Edición**

Flash MX Professional 2004.

### **Sintaxis**

```
dataTypeObject.setAsString(newStringValue)
```

### **Parámetros**

*newStringValue*    Cadena.

### **Valor devuelto**

Ninguno.

### **Descripción**

Método; establece el campo según el valor nuevo, que se proporciona en forma de cadena. El valor se convierte en el tipo de datos adecuado para este campo y se almacena como tal.

### **Ejemplo**

```
var stringVal: String = "The new value";  
field.setAsString (stringVal);
```

## **DataType.setTypedValue()**

### **Disponibilidad**

Flash Player 6 versión 79.

### **Edición**

Flash MX Professional 2004.

### **Sintaxis**

```
dataTypeObject.setTypedValue(newTypedValue)
```

### **Parámetros**

*newValue*    Valor del objeto TypedValue que debe definirse en el campo.

Para más información acerca de los objetos TypedValue, consulte [“Clase TypedValue \(sólo para Flash Professional\)” en la página 163](#).

## Valor devuelto

Matriz de cadenas que describen los errores que se producen mientras se intenta definir un valor nuevo. Los errores pueden producirse en una de las condiciones siguientes:

- El tipo de datos que se proporciona no es aceptable.
- Los datos proporcionados no pueden convertirse en el tipo de datos de este campo, por ejemplo "abc" no puede convertirse en un tipo de datos Number.
- El tipo de los datos es correcto pero no cumple con los criterios de validación del campo.
- El campo es de sólo lectura.

**Nota:** el texto real de los mensajes varía en función del tipo de datos, los formateadores y los codificadores que se definen en el esquema del campo.

## Descripción

Método; define un valor nuevo en el campo, utilizando la información del esquema del campo para procesar el campo. Este método se comporta de forma similar a [DataType.setAnyTypedValue\(\)](#), salvo en que no realiza tantos intentos de convertir los datos en un tipo de datos aceptable. Para más información, consulte [DataType.setAnyTypedValue\(\)](#).

## Ejemplo

En este ejemplo se crea un nuevo objeto TypedValue (valor Boolean) y, a continuación, se asigna dicho valor al objeto DataType denominado `field`. Todos los errores que se produzcan se asignan a la matriz `errors`.

```
import mx.data.binding.*;
var bool:TypedValue = new TypedValue (true, "Boolean");
var errors: Array = field.setTypedValue (bool);
```

## Véase también

[DataType.setTypedValue\(\)](#)

## Clase TypedValue (sólo para Flash Professional)

**Nombre de clase de ActionScript**    `mx.data.binding.TypedValue`

TypedValue es un objeto que contiene un valor de datos, junto con información sobre el tipo de datos del valor. Varios métodos de la clase DataType devuelven objetos TypedValue. Estos objetos se proporcionan en forma de parámetros para dichos métodos. La información de tipo de datos del objeto TypedValue resulta útil para que los objetos DataType decidan el momento y el modo en los que se debe llevar a cabo la conversión de tipo.

**Nota:** para que esta clase esté disponible en tiempo de ejecución, debe incluir el componente DataBindingClasses en el documento FLA. Para más información, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

Para ver información general sobre las clases del paquete `mx.data.binding`, consulte “[Clases de vinculación de datos \(sólo para Flash Professional\)](#)” en la [página 131](#).

## Resumen de propiedades de la clase TypedValue

| Propiedad                        | Descripción                                                           |
|----------------------------------|-----------------------------------------------------------------------|
| <code>TypedValue.type</code>     | Contiene el esquema asociado con el valor del objeto TypedValue.      |
| <code>TypedValue.typeName</code> | Contiene el nombre del tipo de datos del valor del objeto TypedValue. |
| <code>TypedValue.value</code>    | Contiene el valor de datos del objeto TypedValue.                     |

## Constructor para la clase TypedValue

### Disponibilidad

Flash Player 6 versión 79.

### Sintaxis

```
new mx.data.binding.TypedValue(value, typeName, [type])
```

### Parámetros

*value* Valor de datos. Puede ser de cualquier tipo.

*typeName* Cadena que contiene el nombre del tipo de datos del valor.

*type* (Opcional) Objeto del esquema que describe el esquema de los datos más detalladamente. Este campo sólo es obligatorio en determinadas circunstancias, como por ejemplo, al establecer los datos en la propiedad `dataProvider` de un componente `DataSet`.

### Descripción

Constructor; crear un nuevo objeto TypedValue.

## TypedValue.type

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
typedValueObject.type
```

### Descripción

Propiedad; contiene el esquema asociado con el valor del objeto TypedValue. Sólo se utiliza en determinadas circunstancias.

### Ejemplo

En este ejemplo aparece “null” en el panel Salida.

```
var t: TypedValue = new TypedValue (true, "Boolean", null);  
trace(t.type);
```



## TypedValue.typeName

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*typedValueObject.typeName*

### Descripción

Propiedad; contiene el nombre del tipo de datos del valor del objeto TypedValue.

### Ejemplo

En este ejemplo aparece “Boolean” en el panel Salida.

```
var t: TypedValue = new TypedValue (true, "Boolean", null);  
trace(t.typeName);
```

## TypedValue.value

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*typedValueObject.value*

### Descripción

Propiedad; contiene el valor de datos del objeto TypedValue.

### Ejemplo

En este ejemplo aparece “true” en el panel Salida.

```
var t: TypedValue = new TypedValue (true, "Boolean", null);  
trace(t.value);
```

## Componente DataGrid (sólo para Flash Professional)

El componente DataGrid permite crear visualizaciones y aplicaciones de datos con muchas posibilidades. Puede utilizar el componente DataGrid para crear instancias de un juego de registros (recuperado de una consulta de base de datos de ColdFusion, Java o .Net) mediante Macromedia Flash Remoting y mostrarlo en columnas. También puede utilizar datos de un juego de datos o de una matriz para definir un componente DataGrid. El componente DataGrid de la versión 2 se ha mejorado para incluir el desplazamiento horizontal, un mejor soporte de eventos (incluido el soporte de eventos para celdas editables), capacidades de clasificación mejoradas y una optimización del rendimiento.

Puede cambiar el tamaño y personalizar características como la fuente, el color y los bordes de columna de una cuadrícula. Puede utilizar un clip de película personalizado como procesador de celdas para cualquier columna de una cuadrícula. Los procesadores de celdas muestran el contenido de una celda. Puede utilizar las barras de desplazamiento para mover los datos de una cuadrícula; también puede desactivar las barras de desplazamiento y utilizar los métodos DataGrid para crear una visualización de estilo de vista de página.

Si añade el componente DataGrid a una aplicación, puede utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al componente. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad del componente DataGrid:

```
mx.accessibility.DataGridAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda.

## Interacción con el componente DataGrid (sólo para Flash Professional)

Puede utilizar el ratón y el teclado para interactuar con un componente DataGrid.

Si `DataGrid.sortableColumns` es `true` y `DataGridColumn.sortOnHeaderRelease` es `true`, al hacer clic en el encabezado de una columna la cuadrícula se clasificará en función de los valores de celda de la columna.

Si `DataGrid.resizableColumns` es `true`, al hacer clic en el área entre las dos columnas se podrá cambiar el tamaño de las mismas.

Si hace clic en una celda editable, ésta quedará seleccionada; si hace clic en una celda no editable, no quedará seleccionada. Una celda individual puede editarse cuando las propiedades `DataGrid.editable` y `DataGridColumn.editable` son `true`.

Si se selecciona una instancia DataGrid haciendo clic en la misma o utilizando la tabulación, podrá utilizar las teclas siguientes para controlar dicha instancia:

| Tecla                         | Descripción                                                                                                                                                                                                                                                |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Flecha abajo                  | Cuando se edita una celda, el punto de inserción se desplaza al final del texto de la celda. Si una celda no puede editarse, la flecha abajo permite manejar la selección tal como lo hace el componente List.                                             |
| Flecha arriba                 | Cuando se edita una celda, el punto de inserción se desplaza al principio del texto de la celda. Si una celda no puede editarse, la flecha arriba permite manejar la selección tal como lo hace el componente List.                                        |
| Flecha derecha                | Cuando se edita una celda, el punto de inserción se desplaza un carácter hacia la derecha. Si una celda no puede editarse, la flecha derecha no hará nada.                                                                                                 |
| Flecha izquierda              | Cuando se edita una celda, el punto de inserción se desplaza un carácter hacia la izquierda. Si una celda no puede editarse, la flecha izquierda no hará nada.                                                                                             |
| Retorno/Intro/<br>Mayús+Intro | Si una celda puede editarse, se lleva a cabo el cambio y el punto de inserción se desplaza a la celda de la misma columna, en la fila siguiente (arriba o abajo, en función del sentido de desplazamiento activado).                                       |
| Mayús+Tabulador/<br>Tabulador | Desplaza la selección al elemento anterior. Si presiona la tecla Tabulador, la selección de la última columna de la cuadrícula se ajusta a la primera columna de la línea siguiente. Si se presiona Mayús+Tabulador, el ajuste se llevará a cabo al revés. |

## Utilización del componente DataGrid (sólo para Flash Professional)

Puede utilizar el componente DataGrid como base para los numerosos tipos de aplicaciones basadas en datos. Puede visualizar fácilmente una vista de tabla con formato de una consulta de base de datos (o de otros datos). Sin embargo, también puede utilizar las funciones del procesador de celdas para crear partes más sofisticadas y editables de la interfaz de usuario. A continuación se presentan usos prácticos del componente DataGrid:

- Cliente de correo Web
- Páginas de resultados de búsqueda
- Aplicaciones de hojas de cálculo, como las calculadoras de préstamos y las aplicaciones de impresos de declaración de la renta.

El componente DataGrid consta de dos conjuntos de interfaces API: la clase DataGrid y la clase DataGridColumn.

## Aspectos básicos del componente DataGrid: vista y modelo de datos

Conceptualmente, el componente DataGrid está compuesto por un modelo de datos y una vista que muestra los datos. El modelo de datos consta de tres partes principales:

- Proveedor de datos (DataProvider)

Lista de elementos con los que se rellena una cuadrícula de datos. A las matrices que se encuentran en el mismo fotograma en forma de componente DataGrid se les asignan métodos automáticamente (desde la API de DataProvider) que permiten manipular datos y difundir cambios en varias vistas. Todos los objetos que implementen la interfaz DataProvider pueden asignarse a la propiedad `DataGrid.dataProvider`, incluidos los juegos de registros, los juegos de datos, etc. El código siguiente crea un proveedor de datos llamado `myDP`:

```
myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel", price:"Cheap"});
```

- Elemento

Objeto de ActionScript que sirve para almacenar las unidades de información en las celdas de una columna. En realidad, una cuadrícula de datos es una lista que puede mostrar más de una columna de datos. Una lista puede concebirse como una matriz; cada espacio indexado de la lista es un elemento. En el caso del componente DataGrid, cada elemento está formado por campos. En el código siguiente, el contenido que se encuentra entre llaves ({} ) es un elemento:

```
myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel", price:"Cheap"});
```

- Campo

Identificadores que indican los nombres de las columnas de los elementos. Esto corresponde a la propiedad `columnNames` de la lista de columnas. En el componente List, los campos suelen ser `label` y `data`, pero en el caso del componente DataGrid los campos pueden ser cualquier identificador. En el código siguiente, los campos son `name` y `price`:

```
myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel", price:"Cheap"});
```

La vista está formada por tres partes principales:

- Fila

Se trata de un objeto de la vista que se ocupa de presentar los elementos de la cuadrícula estableciendo celdas. Cada fila está dispuesta de forma horizontal debajo de la anterior.

- Columna

Consta de objetos de la vista (instancias de la clase `DataGridColumn`) que muestran las columnas, por ejemplo la anchura, el color, el tamaño, etc.

Existen tres formas de añadir columnas a una cuadrícula de datos: asignando un objeto `DataProvider` a `DataGrid.dataProvider` (de este modo se genera automáticamente una columna para cada campo del primer elemento), estableciendo `DataGrid.columnNames` para especificar los campos que se mostrarán o utilizando el constructor de la clase `DataGridColumn` para crear columnas y llamar a `DataGrid.addColumn()` para añadirlas a la cuadrícula.

Para aplicar formato a las columnas, configure las propiedades de estilo para toda la cuadrícula de datos o defina los objetos `DataGridColumn`, configure los formatos de estilo de los mismos de forma individual y añádalos a la cuadrícula de datos.

- Celda

Se trata de un objeto de la vista que se encarga de presentar los campos individuales para cada elemento. Para comunicarse con la cuadrícula de datos, estos componentes deben implementar la interfaz `CellRenderer` (véase [“API CellRenderer” en la página 87](#)). En el caso de las cuadrículas de datos básicas, las celdas son objetos `TextField` de `ActionScript` incorporados.

## Parámetros de `DataGrid`

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente `DataGrid` en el inspector de propiedades o en el panel Inspector de componentes:

**multipleSelection** Valor booleano que indica si pueden seleccionarse varios elementos (`true`) o no (`false`). El valor predeterminado es `false`.

**rowHeight** Altura de cada una de las filas, en píxeles. Al modificar el tamaño de fuente no se altera la altura de la fila. El valor predeterminado es 20.

**editable** Valor booleano que indica si la cuadrícula es editable (`true`) o no (`false`). El valor predeterminado es `false`.

Es posible escribir `ActionScript` para controlar éstas y otras opciones adicionales para el componente `DataGrid` mediante sus propiedades, métodos y eventos. Para más información, consulte [“Clase `DataGrid` \(sólo para `Flash Professional`\)” en la página 171](#).

## Creación de una aplicación con el componente `DataGrid`

Para crear una aplicación con el componente `DataGrid`, primero debe determinar el origen de los datos. Los datos de una cuadrícula provienen de un juego de registros basado en una consulta de base de datos de `Macromedia ColdFusion`, `Java` o `.Net` mediante `Flash Remoting`. Los datos también pueden provenir de un juego de datos o de una matriz. Para colocar los datos en una cuadrícula, establezca la propiedad `DataGrid.dataProvider` en el juego de registros, el juego de datos o la matriz. También puede utilizar los métodos de las clases `DataGrid` y `DataGridColumn` para crear datos de forma local. Un objeto `Array` del mismo fotograma que el componente `DataGrid` copia los métodos, las propiedades y los eventos de la clase `DataProvider`.

**Para utilizar `Flash Remoting` para añadir un componente `DataGrid` a una aplicación:**

- 1 En `Flash`, seleccione `Archivo > Nuevo` y, a continuación, `Documento de Flash`.
- 2 En el panel `Componentes`, haga doble clic en el componente `DataGrid` para añadirlo al escenario.
- 3 En el inspector de propiedades, introduzca el nombre de la instancia **myDataGrid**.
- 4 En el panel `Acciones del Fotograma 1`, introduzca el código siguiente:  

```
myDataGrid.dataProvider = recordSetInstance;
```

El juego de registros `recordSetInstance` de `Flash Remoting` se asigna a la propiedad `dataProvider` de `myDataGrid`.
- 5 Seleccione `Control > Probar película`.

**Para utilizar un proveedor de datos local para añadir un componente DataGrid a una aplicación:**

- 1 En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
- 2 En el panel Componentes, haga doble clic en el componente DataGrid para añadirlo al escenario.
- 3 En el inspector de propiedades, introduzca el nombre de la instancia **myDataGrid**.

- 4 En el panel Acciones del Fotograma 1, introduzca el código siguiente:

```
myDP = new Array({name:"Chris", price:"Priceless"}, {name:"Nigel",  
    price:"Cheap"});  
myDataGrid.dataProvider = myDP;
```

Los campos `name` y `price` se utilizan como encabezados de columna y sus valores sirven para rellenar las celdas de cada fila.

- 5 Seleccione Control > Probar película.

## Personalización del componente DataGrid (sólo para Flash Professional)

Puede transformar un componente DataGrid horizontal y verticalmente durante la edición y el tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)). Si no dispone de ninguna barra de desplazamiento horizontal, los anchos de la columna se ajustarán de forma proporcional. Si se ajusta el tamaño de una columna, y por lo tanto de una celda, es posible que el texto aparezca recortado.

## Utilización de estilos con el componente DataGrid

Es posible definir propiedades de estilo para cambiar el aspecto de un componente DataGrid. El componente DataGrid hereda estilos Halo del componente List. Para más información, consulte [“Utilización de estilos con el componente List” en la página 313](#). El componente DataGrid también es compatible con los estilos de Halo siguientes:

| Estilo                       | Descripción                                                                                                                              |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>backgroundColor</code> | El color de fondo puede establecerse para toda la cuadrícula o para cada columna.                                                        |
| <code>labelStyle</code>      | El estilo de fuente puede establecerse para toda la cuadrícula o para cada columna.                                                      |
| <code>headerStyle</code>     | Declaración de estilo CSS de encabezado de columna que puede aplicarse a una cuadrícula o columna.                                       |
| <code>vGridLines</code>      | Valor booleano que indica si deben mostrarse las líneas verticales de la cuadrícula ( <code>true</code> ) o no ( <code>false</code> ).   |
| <code>hGridLines</code>      | Valor booleano que indica si deben mostrarse las líneas horizontales de la cuadrícula ( <code>true</code> ) o no ( <code>false</code> ). |
| <code>vGridLineColor</code>  | Color de las líneas verticales de la cuadrícula.                                                                                         |
| <code>hGridLineColor</code>  | Color de las líneas horizontales de la cuadrícula.                                                                                       |
| <code>headerColor</code>     | Color de los encabezados de columna.                                                                                                     |

Si la tabla anterior indica que puede establecerse un estilo para una columna, puede utilizar la sintaxis siguiente para establecerlo:

```
grid.getColumnAt(3).setStyle("backgroundColor", 0xff00aa)
```

## Utilización de aspectos con el componente DataGrid

Los aspectos que el componente DataGrid utiliza para representar sus estados visuales se incluyen en los subcomponentes que forman la cuadrícula de datos (ScrollPane y RectBorder). Para información sobre estos aspectos, véase [“Utilización de aspectos con el componente ScrollPane” en la página 501](#) y [“Utilización de aspectos con el componente List” en la página 314](#).

Sin embargo, las capas inferiores de desplazamiento y selección utilizan la interfaz API de dibujo de ActionScript. Para aplicar aspectos a estas partes de la cuadrícula de datos durante la edición, modifique el código ActionScript de los símbolos de aspecto en la carpeta de estados de aspectos Flash UI Components 2/Themes/MMDefault/datagrid/ de la biblioteca de uno de los archivos FLA de temas. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).

## Clase DataGrid (sólo para Flash Professional)

**Herencia** mx.core.UIObject > mx.core.UIComponent > mx.core.View > mx.core.ScrollView > mx.controls.listclasses.ScrollSelectList > mx.controls.List

**Nombre de clase de ActionScript** mx.controls.DataGrid

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.DataGrid.version);
```

**Nota:** el código siguiente devuelve `undefined`: `trace(myDataGridInstance.version);`.

## Resumen de métodos de la clase DataGrid

| Método                                   | Descripción                                                              |
|------------------------------------------|--------------------------------------------------------------------------|
| <code>DataGrid.addColumn()</code>        | Añade una columna a la cuadrícula de datos.                              |
| <code>DataGrid.addColumnAt()</code>      | Añade una columna a la cuadrícula de datos en una ubicación determinada. |
| <code>DataGrid.addItem()</code>          | Añade un elemento a la cuadrícula de datos.                              |
| <code>DataGrid.addItemAt()</code>        | Añade un elemento a la cuadrícula de datos en una ubicación determinada. |
| <code>DataGrid.editField()</code>        | Reemplaza los datos de celda en una ubicación especificada.              |
| <code>DataGrid.getColumnAt()</code>      | Obtiene una referencia a una columna en una ubicación especificada.      |
| <code>DataGrid.getColumnIndex()</code>   | Obtiene el índice de la columna.                                         |
| <code>DataGrid.removeAllColumns()</code> | Elimina todas las columnas de una cuadrícula de datos.                   |

| Método                                      | Descripción                                                                   |
|---------------------------------------------|-------------------------------------------------------------------------------|
| <code>DataGrid.removeColumnAt()</code>      | Elimina una columna de una cuadrícula de datos en una ubicación especificada. |
| <code>DataGrid.replaceItemAt()</code>       | Reemplaza un elemento de una ubicación especificada por otro.                 |
| <code>DataGrid.spaceColumnsEqually()</code> | Distribuye espacios uniformes en todas las columnas.                          |

Hereda todas las propiedades de [UIObject](#) y [UIComponent](#).

## Resumen de propiedades de la clase **DataGrid**

| Propiedad                              | Descripción                                                                                                                                                                 |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>DataGrid.columnCount</code>      | Sólo lectura. Número de columnas que aparecen en pantalla.                                                                                                                  |
| <code>DataGrid.columnNames</code>      | Matriz de nombres de campo de cada elemento que aparecen en forma de columnas.                                                                                              |
| <code>DataGrid.dataProvider</code>     | Modelo de datos de la cuadrícula de datos.                                                                                                                                  |
| <code>DataGrid.editable</code>         | Valor booleano que indica si la cuadrícula de datos es editable ( <code>true</code> ) o no ( <code>false</code> ).                                                          |
| <code>DataGrid.focusedCell</code>      | Define la celda que está seleccionada.                                                                                                                                      |
| <code>DataGrid.headerHeight</code>     | Altura de los encabezados de columna, expresada en píxeles.                                                                                                                 |
| <code>DataGrid.hScrollPolicy</code>    | Indica si la barra de desplazamiento horizontal está activa (" <code>on</code> "), no lo está (" <code>off</code> ") o aparece cuando es necesario (" <code>auto</code> "). |
| <code>DataGrid.resizableColumns</code> | Valor booleano que indica si puede cambiarse el tamaño de las columnas ( <code>true</code> ) o no ( <code>false</code> ).                                                   |
| <code>DataGrid.selectable</code>       | Valor booleano que indica si la cuadrícula de datos es seleccionable ( <code>true</code> ) o no ( <code>false</code> ).                                                     |
| <code>DataGrid.showHeaders</code>      | Valor booleano que indica que los encabezados de columna están visibles ( <code>true</code> ) o no ( <code>false</code> ).                                                  |
| <code>DataGrid.sortableColumns</code>  | Valor booleano que indica si las columnas pueden clasificarse ( <code>true</code> ) o no ( <code>false</code> ).                                                            |

## Resumen de eventos de la clase **DataGrid**

| Evento                             | Descripción                                       |
|------------------------------------|---------------------------------------------------|
| <code>DataGrid.cellEdit</code>     | Se difunde cuando se modifica el valor de celda.  |
| <code>DataGrid.cellFocusIn</code>  | Se difunde cuando se selecciona una celda.        |
| <code>DataGrid.cellFocusOut</code> | Se difunde cuando se deselecciona una celda.      |
| <code>DataGrid.cellPress</code>    | Se difunde cuando se presiona una celda.          |
| <code>DataGrid.change</code>       | Se difunde cuando se ha seleccionado un elemento. |



| Evento                              | Descripción                                                     |
|-------------------------------------|-----------------------------------------------------------------|
| <code>DataGrid.columnStretch</code> | Se difunde cuando un usuario modifica el tamaño de una columna. |
| <code>DataGrid.headerRelease</code> | Se difunde cuando un usuario presiona o libera un encabezado.   |

## DataGrid.addColumn()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.addColumn(dataGridColumn)
myDataGrid.addColumn(name)
```

### Parámetros

*dataGridColumn*    Instancia de la clase `DataGridColumn`.

*name*    Cadena que indica el nombre de un nuevo objeto `DataGridColumn` que debe insertarse.

### Valor devuelto

Se ha añadido una referencia al objeto `DataGridColumn`.

### Descripción

Método; añade una columna nueva al final de la cuadrícula de datos. Para más información, consulte [“Clase DataGridColumn \(sólo para Flash Professional\)” en la página 191](#).

### Ejemplo

El código siguiente añade un objeto `DataGridColumn` nuevo denominado Purple:

```
import mx.controls.gridclasses.DataGridColumn;
myGrid.addColumn(new DataGridColumn("Purple"));
```

## DataGrid.addColumnAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
myDataGrid.addColumnAt(index, name)
```

Sintaxis 2:

```
myDataGrid.addColumnAt(index, dataGridColumn)
```

## Parámetros

*index* Posición de índice en la que se añade el objeto `DataGridColumn`. La primera posición es 0.

*name* Cadena que indica el nombre del objeto `DataGridColumn`. Debe especificar el parámetro *index* o *dataGridColumn*.

*dataGridColumn* Instancia de la clase `DataGridColumn`.

## Valor devuelto

Se ha añadido una referencia al objeto `DataGridColumn`.

## Descripción

Método; añade una columna nueva en la posición especificada. Las columnas se desplazan a la derecha y sus índices se incrementan. Para más información, consulte [“Clase `DataGridColumn` \(sólo para Flash Professional\)” en la página 191](#).

## Ejemplo

En el ejemplo siguiente se inserta un objeto `DataGridColumn` nuevo denominado "Green" en las columnas segunda y cuarta:

```
import mx.controls.gridclasses.DataGridColumn;
myGrid.addColumnAt(1, "Green");
myGrid.addColumnAt(3, new DataGridColumn("Purple"));
```

## `DataGrid.addItem()`

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.addItem(item)
```

### Parámetros

*item* Instancia de un objeto que debe añadirse a una cuadrícula.

### Valor devuelto

Referencia a la instancia que se ha añadido.

### Descripción

Método; añade un elemento al final de la cuadrícula (después del último índice del elemento).

**Nota:** este método difiere del método `List.addItem()` en que se pasa un objeto en lugar de una cadena.

## Ejemplo

En el ejemplo siguiente se añade un objeto nuevo a la cuadrícula `myGrid`:

```
var anObject= {name:"Jim!!", age:30};  
var addedObject = myGrid.addItem(anObject);
```

## DataGrid.addItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.addItemAt(index, item)
```

### Parámetros

*index* Orden (en los nodos secundarios) que debe seguirse para añadir un nodo. La primera posición es 0.

*item* Cadena que muestra el nodo.

### Valor devuelto

Referencia a la instancia de objeto que se ha añadido.

### Descripción

Método; añade un elemento a la cuadrícula en la posición especificada.

## Ejemplo

En el ejemplo siguiente se inserta una instancia de objeto a la cuadrícula en la posición de índice 4:

```
var anObject= {name:"Jim!!", age:30};  
var addedObject = myGrid.addItemAt(4, anObject);
```

## DataGrid.cellEdit

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.cellEdit = function(eventObject){  
    // introducir aquí el código propio  
}  
myDataGridInstance.addEventListener("cellEdit", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando se ha modificado un valor de celda.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. El componente DataGrid distribuye un evento `cellEdit` cuando se ha modificado el valor de una celda y el evento se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `DataGrid.cellEdit` tiene cuatro propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino.

`itemIndex` Número que indica el índice de la fila de destino.

`oldValue` Valor anterior de la celda.

`type` Cadena "cellEdit".

Para más información, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, se define y pasa el controlador `myDataGridListener` al método `myDataGrid.addEventListener()` como segundo parámetro. El objeto de evento se captura mediante el controlador `cellEdit` del parámetro *eventObject*. Cuando se difunde el evento `cellEdit`, se envía una sentencia `trace` al panel Salida del modo siguiente:

```
myDataGridListener = new Object();
myDataGridListener.cellEdit = function(event){
    var cell = "(" + event.columnIndex + ", " + event.itemIndex + ")";
    trace("The value of the cell at " + cell + " has changed");
}
myDataGrid.addEventListener("cellEdit", myDataGridListener);
```

**Nota:** la cuadrícula debe poder editarse para que el código anterior funcione.

## DataGrid.cellFocusIn

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.cellFocusIn = function(eventObject){
    // introducir aquí el código propio
}
myDataGridInstance.addEventListener("cellFocusIn", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando se selecciona una celda determinada. Este evento se difunde cuando se difunden todos los eventos `editCell` y `cellFocusOut` de las celdas editadas anteriormente.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando un componente `DataGrid` distribuye un evento `cellFocusIn`, el evento se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `DataGrid.cellFocusIn` tiene tres propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino.

`itemIndex` Número que indica el índice de la fila de destino.

`type` Cadena "cellFocusIn".

Para más información, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, el controlador denominado `myListener` se define y pasa al método `grid.addEventListener()` como segundo parámetro. El objeto de evento se captura mediante el controlador `cellFocusIn` en el parámetro *eventObject*. Cuando se difunde el evento `cellFocusIn`, se envía una sentencia `trace` al panel Salida de la manera siguiente:

```
var myListener = new Object();
myListener.cellFocusIn = function(event) {
    var cell = "(" + event.columnIndex + ", " + event.itemIndex + ")";
    trace("The cell at " + cell + " has gained focus");
};
grid.addEventListener("cellFocusIn", myListener);
```

**Nota:** la cuadrícula debe poder editarse para que el código anterior funcione.

## DataGrid.cellFocusOut

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.cellFocusOut = function(eventObject){
    // introducir aquí el código propio
}
myDataGridInstance.addEventListener("cellFocusOut", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario sale de una celda que está seleccionada. Puede utilizar las propiedades del objeto de evento para aislar la celda de la que se ha salido. Este evento se difunde después del evento `cellEdit` y antes de que los eventos `cellFocusIn` posteriores se difundan para la celda siguiente.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando el componente `DataGrid` distribuye un evento `cellFocusOut`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `DataGrid.cellFocusOut` tiene tres propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino. La primera posición es 0.

`itemIndex` Número que indica el índice de la fila de destino. La primera posición es 0.

`type` La cadena "cellFocusOut".

Para más información, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, el controlador denominado `myListener` se define y pasa al método `grid.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `cellFocusOut` en el parámetro *eventObject*. Cuando se difunde el evento `cellFocusOut`, se envía una sentencia `trace` al panel Salida de la forma siguiente:

```
var myListener = new Object();
myListener.cellFocusOut = function(event) {
    var cell = "(" + event.columnIndex + ", " + event.itemIndex + ")";
    trace("The cell at " + cell + " has lost focus");
};
grid.addEventListener("cellFocusOut", myListener);
```

**Nota:** la cuadrícula debe poder editarse para que el código anterior funcione.

## DataGrid.cellPress

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.cellPress = function(eventObject){
    // introducir aquí el código propio
}
myDataGridInstance.addEventListener("cellPress", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario presiona el botón del ratón en una celda.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando el componente DataGrid difunde un evento `cellPress`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `DataGrid.cellPress` tiene tres propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino. La primera posición es 0.

`itemIndex` Número que indica el índice de la fila de destino. La primera posición es 0.

`type` Cadena "cellPress".

Para más información, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, el controlador denominado `myListener` se define y pasa al método `grid.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `cellPress` del parámetro *eventObject*. Cuando se difunde el evento `cellPress`, se envía una sentencia `trace` al panel Salida de la manera siguiente:

```
var myListener = new Object();
myListener.cellPress = function(event) {
    var cell = "(" + event.columnIndex + ", " + event.itemIndex + ")";
    trace("The cell at " + cell + " has been clicked");
};
grid.addEventListener("cellPress", myListener);
```

## DataGrid.change

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.change = function(eventObject){
    // introducir aquí el código propio
}
myDataGridInstance.addEventListener("change", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando se ha seleccionado un elemento.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando un componente DataGrid distribuye un evento `change`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `DataGrid.change` tiene una propiedad adicional, `type`, y su valor es `"change"`. Para más información, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, el controlador denominado `myListener` se define y pasa al método `grid.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `change` en el parámetro *eventObject*. Cuando se difunde el evento `change`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
var myListener = new Object();
myListener.change = function(event) {
    trace("The selection has changed to " + event.target.selectedIndex);
};
grid.addEventListener("change", myListener);
```

## DataGrid.columnCount

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.columnCount
```

### Descripción

Propiedad (sólo lectura); número de columnas que aparecen en pantalla.

### Ejemplo

En el ejemplo siguiente se obtiene el número de columnas visualizadas en la instancia `DataGrid` `grid`:

```
var c = grid.columnCount;
```



## DataGrid.columnNames

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.columnNames
```

### Descripción

Propiedad; matriz de los nombres de campo de cada elemento que aparecen en forma de columnas.

### Ejemplo

En el ejemplo siguiente se indica a la instancia `grid` que sólo muestre estos tres campos en forma de columna:

```
grid.columnNames = ["Name", "Description", "Price"];
```

## DataGrid.columnStretch

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.columnStretch = function(eventObject){  
    // introducir aquí el código propio  
}  
myDataGridInstance.addEventListener("columnStretch", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario cambia el tamaño de una columna horizontalmente.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando un componente DataGrid distribuye un evento `columnStretch`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `DataGrid.columnStretch` tiene dos propiedades adicionales:

`columnIndex` Número que indica el índice de la columna de destino. La primera posición es 0.

`type` Cadena "columnStretch".

Para más información, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

En el ejemplo siguiente, el controlador denominado `myListener` se define y pasa al método `grid.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `columnStretch` en el parámetro *eventObject*. Cuando se difunde el evento `columnStretch`, se envía una sentencia `trace` al panel Salida de la manera siguiente:

```
var myListener = new Object();
myListener.columnStretch = function(event) {
    trace("column " + event.columnIndex + " was resized");
};
grid.addEventListener("columnStretch", myListener);
```

## DataGrid.dataProvider

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.dataProvider
```

### Descripción

Propiedad; modelo de datos de los elementos que se ven en el componente `DataGrid`.

La cuadrícula de datos añade métodos al prototipo de la clase `Array` de modo que cada objeto `Array` se adapta a la interfaz `DataProvider` (véase el archivo `DataProvider.as` en la carpeta `Classes/mx/controls/listclasses`). Una matriz que se encuentre en el mismo fotograma o pantalla en forma de cuadrícula de datos tendrá todos los métodos (`addItem()`, `getItemAt()`, etc.) necesarios para que dicha cuadrícula sea el modelo de datos de una cuadrícula de datos. Asimismo, puede utilizarse para difundir cambios del modelo de datos en varios componentes.

En el componente `DataGrid` debe especificar los campos que aparecerán en la propiedad `DataGrid.columnNames`.

Si no define la columna que se ha establecido (estableciendo la propiedad `DataGrid.columnNames` o llamando al método `DataGrid.addColumn()`) para la cuadrícula de datos antes de que se haya establecido la propiedad `DataGrid.dataProvider`, dicha cuadrícula generará columnas para cada campo en el primer elemento del proveedor de datos, una vez que haya llegado dicho elemento.

Cualquier objeto que implemente la interfaz `DataProvider` puede utilizarse como un proveedor de datos para una cuadrícula de datos (incluidos las matrices, los juegos de datos y los juegos de registros de Flash Remoting).

### Ejemplo

En el ejemplo siguiente se crea una matriz que se utilizará como proveedor de datos y se le asigna directamente la propiedad `dataProvider`:

```
grid.dataProvider = [{name:"Chris", price:"Priceless"}, {name:"Nigel",  
    price:"cheap"}];
```

En el ejemplo siguiente se crea un nuevo objeto `Array` que tiene la clase `DataProvider`. Utiliza un bucle `for` para añadir 20 elementos a la cuadrícula:

```
myDP = new Array();  
for (var i=0; i<20; i++)  
    myDP.addItem({name:"Nivesh", price:"Priceless"});  
list.dataProvider = myDP
```

## DataGrid.editable

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.editable
```

### Descripción

Propiedad; determina si el usuario puede editar la cuadrícula de datos (`true`) o no (`false`). Esta propiedad debe ser `true` para que puedan editarse columnas individuales y puedan seleccionarse celdas. El valor predeterminado es `false`.

### Ejemplo

En el ejemplo siguiente se establece la posición de desplazamiento en la parte superior de la visualización:

```
myDataGrid.editable = true;
```

## DataGrid.editField()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.editField(index, colName, data)
```

## Parámetros

*index* Índice de la celda de destino. El valor está basado en cero.

*colName* Cadena que indica el nombre de la columna (campo) que contiene la celda de destino.

*data* Valor que debe almacenarse en la celda de destino. Este parámetro puede ser cualquier tipo de datos.

## Valor devuelto

Los datos que estaban en la celda.

## Descripción

Método; reemplaza los datos de celda en la ubicación especificada.

## Ejemplo

En el ejemplo siguiente se coloca un valor en la cuadrícula:

```
var prevValue = myGrid.editField(5, "Name", "Neo");
```

## DataGrid.focusedCell

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.focusedCell
```

### Descripción

Propiedad; únicamente en el modo editable, instancia de objeto que define la celda que está seleccionada. El objeto debe tener los campos `columnIndex` e `itemIndex`, los cuales son números enteros que indican el índice de la columna y el elemento de la celda. El origen es (0,0). El valor predeterminado es `undefined`.

### Ejemplo

En el ejemplo siguiente se establece la celda seleccionada en la tercera columna, cuarta fila:

```
grid.focusedCell = {columnIndex:2, itemIndex:3};
```

## DataGrid.getColumnAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.getColumnAt(index)
```

**Parámetros**

*index* Índice del objeto DataGridColumn que debe devolverse. El valor está basado en cero.

**Valor devuelto**

Objeto DataGridColumn.

**Descripción**

Método; obtiene una referencia para el objeto DataGridColumn en el índice especificado.

**Ejemplo**

En el ejemplo siguiente coloca el objeto DataGridColumn en el índice 4:

```
var aColumn = myGrid.getColumnAt(4);
```

**DataGrid.getColumnIndex()****Disponibilidad**

Flash Player 6 versión 79.

**Edición**

Flash MX Professional 2004.

**Sintaxis**

```
myDataGrid.getColumnIndex(index)
```

**Parámetros**

*index* Índice del objeto DataGridColumn que debe devolverse.

**Valor devuelto**

Objeto DataGridColumn.

**Descripción**

Método; obtiene una referencia para el objeto DataGridColumn en el índice especificado.

**DataGrid.headerHeight****Disponibilidad**

Flash Player 6 versión 79.

**Edición**

Flash MX Professional 2004.

**Sintaxis**

```
myDataGrid.headerHeight
```

**Descripción**

Propiedad; altura de la barra de encabezado de la cuadrícula de datos. El valor predeterminado es 20.

## Ejemplo

En el ejemplo siguiente se establece la posición de desplazamiento en la parte superior de la visualización:

```
myDataGrid.headerHeight = 30;
```

## DataGrid.headerRelease

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.headerRelease = function(eventObject){
    // introducir aquí el código propio
}
myDataGridInstance.addEventListener("headerRelease", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando se libera un encabezado de columna. Puede utilizar este evento con la propiedad `DataGridColumn.sortOnHeaderRelease` para que no se lleve a cabo la clasificación automática, sino que se realice según se desee.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando el componente `DataGrid` distribuye un evento `headerRelease`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `DataGrid.headerRelease` tiene dos propiedades adicionales:

`columnIndex`    Número que indica el índice de la columna de destino.

`type`    Cadena "headerRelease".

Para más información, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, el controlador denominado `myListener` se define y pasa al método `grid.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `headerRelease` en el parámetro *eventObject*. Cuando se difunde el evento `headerRelease`, se envía una sentencia `trace` al panel Salida de la manera siguiente:

```
var myListener = new Object();
myListener.headerRelease = function(event) {
    trace("column " + event.columnIndex + " header was pressed");
};
grid.addEventListener("headerRelease", myListener);
```

## DataGrid.hScrollPolicy

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.hScrollPolicy
```

### Descripción

Propiedad; especifica si la cuadrícula de datos tiene una barra de desplazamiento horizontal. Esta propiedad puede tener uno de los tres valores siguientes: "on", "off" o "auto". El valor predeterminado es "off".

Si establece `hScrollPolicy` en "off", las columnas se escalarán de forma proporcional y se adaptarán al ancho finito.

### Ejemplo

En el ejemplo siguiente se establece una política según la cual el desplazamiento horizontal se activa de forma automática:

```
myDataGrid.hScrollPolicy = "auto";
```

## DataGrid.removeAllColumns()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.removeAllColumns()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; elimina todos los objetos `DataGridColumn` de la cuadrícula de datos. La llamada a este método no afecta al proveedor de datos.

### Ejemplo

En el ejemplo siguiente se eliminan todos los objetos `DataGridColumn` de `myDataGrid`:

```
myDataGrid.removeAllColumns();
```

## DataGrid.removeColumnAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.removeColumnAt(index)
```

### Parámetros

*index* Índice de la columna que se va a eliminar.

### Valor devuelto

Una referencia al objeto DataGridColumn eliminado.

### Descripción

Método; elimina el objeto DataGridColumn en el índice especificado.

### Ejemplo

En el ejemplo siguiente se elimina el objeto DataGridColumn en el índice 2 de myDataGrid:

```
myDataGrid.removeColumnAt(2);
```

## DataGrid.replaceItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.replaceItemAt(index, item)
```

### Parámetros

*index* Índice del elemento que se va a sustituir.

*item* Objeto que es el valor de elemento que se va a utilizar como sustitución.

### Valor devuelto

El valor anterior.

### Descripción

Método; sustituye el elemento del índice especificado.



## Ejemplo

En el ejemplo siguiente se sustituye el elemento del índice 4 por el elemento definido en `aNewValue`:

```
var aNewValue = {name:"Jim", value:"tired"};
var prevValue = myGrid.replaceItemAt(4, aNewValue);
```

## DataGrid.resizableColumns

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.resizableColumns
```

### Descripción

Propiedad; valor booleano que determina si el usuario puede (`true`) o no puede (`false`) expandir las columnas de la cuadrícula. El valor de esta propiedad debe ser `true` si se desea que el tamaño de cada columna pueda modificarse por separado. El valor predeterminado es `true`.

## Ejemplo

En el ejemplo siguiente se impide a los usuarios la posibilidad de cambiar el tamaño de las columnas:

```
myDataGrid.resizableColumns = false;
```

## DataGrid.selectable

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.selectable
```

### Descripción

Propiedad; valor booleano que determina si el usuario puede seleccionar una cuadrícula de datos (`true`) o no (`false`). El valor predeterminado es `true`.

## Ejemplo

En el ejemplo siguiente se impide seleccionar la cuadrícula:

```
myDataGrid.selectable = false;
```

## DataGrid.showHeaders

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.showHeaders
```

### Descripción

Propiedad; valor booleano que determina si la cuadrícula de datos muestra los encabezados de columna (`true`) o no (`false`). Los encabezados de columna aparecen sombreados para distinguirlos de las demás filas de la cuadrícula. El usuario puede hacer clic en un encabezado de columna para ordenar el contenido de dicha columna si el valor de [DataGrid.sortableColumns](#) se establece en `true`. El valor predeterminado es `true`.

### Ejemplo

En el ejemplo siguiente se esconden los encabezados de columna:

```
myDataGrid.showHeaders = false;
```

### Véase también

[DataGrid.sortableColumns](#)

## DataGrid.sortableColumns

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.sortableColumns
```

### Descripción

Propiedad; valor booleano que determina si las columnas de una cuadrícula de datos pueden ordenarse (`true`) o no (`false`) cuando el usuario hace clic en un encabezado de columna. El valor de esta propiedad debe establecerse en `true` si se desea que las columnas puedan ordenarse. El valor de esta propiedad debe establecerse en `true` para poder difundir el evento `headerRelease`. El valor predeterminado es `true`.

### Ejemplo

En el ejemplo siguiente se desactiva la función de ordenación:

```
myDataGrid.sortableColumns = false;
```

### Véase también

[DataGrid.headerRelease](#)

## DataGrid.spaceColumnsEqually()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.spaceColumnsEqually()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; vuelve a espaciar las columnas de manera uniforme.

### Ejemplo

En el ejemplo siguiente se vuelven a espaciar las columnas de myGrid al presionar y luego soltar cualquier encabezado de columna:

```
myGrid.showHeaders = true
myGrid.dataProvider = [{guitar:"Flying V", name:"maggot"}, {guitar:"SG",
    name:"dreschie"}, {guitar:"jagstang", name:"vitapup"}];
gridLO = new Object();
gridLO.headerRelease = function(){
    myGrid.spaceColumnsEqually();
}
myGrid.addEventListener("headerRelease", gridLO);
```

## Clase DataGridColumn (sólo para Flash Professional)

**Nombre de clase de ActionScript** mx.controls.gridclassesDataGridColumn

Se pueden crear y configurar objetos DataGridColumn para utilizarlos como columnas de una cuadrícula de datos. Muchos de los métodos de la clase DataGrid tienen la función de administrar objetos DataGridColumn. Los objetos DataGridColumn se almacenan en una matriz basada en cero en la cuadrícula de datos, donde 0 indica la columna situada en el extremo izquierdo. Tras añadir o crear columnas, es posible llamar a `DataGrid.getColumnAt(index)` para acceder a las mismas.

Existen tres formas de añadir o crear columnas en una cuadrícula. Si desea configurar las columnas, es mejor utilizar el segundo o el tercer método antes de añadir datos a la cuadrícula, de manera que no haya que crear las columnas dos veces.

- Si se añade un proveedor de datos (`DataProvider`) o un elemento con varios campos a una cuadrícula en la que no se haya configurado ningún objeto `DataGridColumn`, se generarán automáticamente columnas en cada campo en orden inverso al del bucle `for..in`.
- `DataGrid.columnNames` incorpora los nombres de campo de los campos de elemento deseados y genera en orden objetos `DataGridColumn` para cada uno de los campos enumerados. Este sistema permite seleccionar y ordenar columnas rápidamente, lo cual reduce al mínimo las necesidades de configuración. Además, se elimina la información anterior de las columnas.
- La forma más flexible de añadir columnas consiste en generarlas previamente como objetos `DataGridColumn` y añadirlas a la cuadrícula de datos mediante `DataGrid.addColumn()`. Este método es útil porque permite añadir columnas con el tamaño y formato adecuados antes de que las columnas se incluyan en la cuadrícula (con lo que se reduce el consumo de recursos del procesador). Para más información, consulte “[Constructor para la clase DataGridColumn](#)” en la [página 193](#).

## Resumen de propiedades de la clase DataGridColumn

| Propiedad                                       | Descripción                                                                                                                                       |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>DataGridColumn.cellRenderer</code>        | Identificador de la vinculación de un símbolo que se va a utilizar para mostrar las celdas de la columna.                                         |
| <code>DataGridColumn.columnName</code>          | Sólo lectura. Nombre del campo asociado con la columna.                                                                                           |
| <code>DataGridColumn.editable</code>            | Valor booleano que indica si la columna es editable ( <code>true</code> ) o no ( <code>false</code> ).                                            |
| <code>DataGridColumn.headerRenderer</code>      | Nombre de una clase que se utilizará para mostrar el encabezado de la columna.                                                                    |
| <code>DataGridColumn.headerText</code>          | Texto del encabezado de la columna.                                                                                                               |
| <code>DataGridColumn.labelFunction</code>       | Función que determina qué campo de un elemento va a mostrarse.                                                                                    |
| <code>DataGridColumn.resizable</code>           | Valor booleano que indica si puede cambiarse el tamaño de la columna ( <code>true</code> ) o no ( <code>false</code> ).                           |
| <code>DataGridColumn.sortable</code>            | Valor booleano que indica si la columna puede ordenarse ( <code>true</code> ) o no ( <code>false</code> ).                                        |
| <code>DataGridColumn.sortOnHeaderRelease</code> | Valor booleano que indica si la columna está ordenada ( <code>true</code> ) o no ( <code>false</code> ) cuando el usuario pulsa en el encabezado. |
| <code>DataGridColumn.width</code>               | Anchura de la columna, expresada en píxeles.                                                                                                      |

## Constructor para la clase DataGridColumn

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
new DataGridColumn(name)
```

### Parámetros

*name* Cadena que indica el nombre del objeto DataGridColumn. Este parámetro corresponde al campo que se mostrará de cada elemento.

### Valor devuelto

Ninguno.

### Descripción

Constructor; crea un objeto DataGridColumn. Utilice este constructor para crear columnas y añadirlas al componente DataGrid. Después de crear objetos DataGridColumn podrá añadirlos a la cuadrícula de datos mediante una llamada a [DataGrid.addColumn\(\)](#).

### Ejemplo

En el ejemplo siguiente se crea un objeto DataGridColumn denominado `Location`:

```
import mx.controls.gridclasses.DataGridColumn;
var column = new DataGridColumn("Location");
```

## DataGridColumn.cellRenderer

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.getColumnAt(index).cellRenderer
```

### Descripción

Propiedad; identificador de la vinculación de un símbolo que se va a utilizar para mostrar las celdas de la columna. Todas las clases que se utilicen con esta propiedad deben implementar la interfaz CellRenderer (véase [“API CellRenderer” en la página 87](#)). El valor predeterminado es `undefined`.

### Ejemplo

En el ejemplo siguiente se utiliza un identificador de vinculación para definir un nuevo procesador de celdas:

```
myGrid.getColumnAt(3).cellRenderer = "MyCellRenderer";
```

## DataGridColumn.columnName

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.getColumnAt(index).columnName
```

### Descripción

Propiedad (sólo lectura); nombre del campo asociado con la columna. El valor predeterminado es el nombre denominado en el constructor DataGridColumn.

### Ejemplo

En el ejemplo siguiente se asigna a la variable `name` el nombre de la columna situada en la tercera posición del índice.

```
var name = myGrid.getColumnAt(3).columnName;
```

### Véase también

[Constructor para la clase DataGridColumn](#)

## DataGridColumn.editable

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.getColumnAt(index).editable
```

### Descripción

Propiedad; determina si el usuario puede editar la columna (`true`) o no (`false`). La propiedad [DataGrid.editable](#) debe tener un valor de `true` para que puedan editarse las distintas columnas, incluso cuando el valor de `DataGridColumn.editable` sea `true`. El valor predeterminado es `true`.

### Ejemplo

En el ejemplo siguiente se impide la posibilidad de editar la primera columna de una cuadrícula:

```
myDataGrid.getColumnAt(0).editable = false;
```

### Véase también

[DataGrid.editable](#)

## DataGridColumn.headerRenderer

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.getColumnAt(index).headerRenderer
```

### Descripción

Propiedad; cadena que indica el nombre de clase que se utilizará para mostrar el encabezado de la columna. Todas las clases que se utilicen con esta propiedad deben implementar la interfaz `CellRenderer` (véase [“API CellRenderer” en la página 87](#)). El valor predeterminado es `undefined`.

### Ejemplo

En el ejemplo siguiente se utiliza un identificador de vinculación para definir un nuevo procesador de encabezados:

```
myGrid.getColumnAt(3).headerRenderer = "MyHeaderRenderer";
```

## DataGridColumn.headerText

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.getColumnAt(index).headerText
```

### Descripción

Propiedad; texto del encabezado de la columna. El valor predeterminado es el nombre de la columna.

### Ejemplo

En el ejemplo siguiente se define “The Price” como texto de encabezado de la columna:

```
var myColumn = new DataGridColumn("price");  
myColumn.headerText = "The Price";
```

## DataGridColumn.labelFunction

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

`myDataGrid.getColumnAt(index).labelFunction`

## Descripción

Propiedad; especifica una función para determinar qué campo (o combinación de campos) de cada elemento va a mostrarse. Esta función recibe un parámetro *item*, que es el elemento que se va a presentar, y debe devolver una cadena que represente el texto que va a mostrarse. Esta propiedad puede utilizarse para crear columnas virtuales que no tengan campos equivalentes en el elemento.

## Ejemplo

En el ejemplo siguiente se crea una columna virtual:

```
var myCol = myGrid.addColumn("Subtotal");
myCol.labelFunction = function(item) {
    return "$" + (item.price + (item.price * salesTax));
};
```

## DataGridColumn.resizable

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

`myDataGrid.getColumnAt(index).resizable`

## Descripción

Propiedad; valor booleano que determina si el usuario puede cambiar el tamaño de la columna (*true*) o no (*false*). Para que esta propiedad pueda aplicarse, el valor de la propiedad `DataGrid.resizableColumns` debe establecerse en *true*. El valor predeterminado es *true*.

## Ejemplo

En el ejemplo siguiente se impide la posibilidad de cambiar el tamaño de la columna del índice 1:

```
myGrid.getColumnAt(1).resizable = false;
```

## DataGridColumn.sortable

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

`myDataGrid.getColumnAt(index).sortable`



### Descripción

Propiedad; valor booleano que indica si el usuario puede ordenar una columna (`true`) o no (`false`). Para que esta propiedad pueda aplicarse, el valor de la propiedad `DataGrid.sortableColumns` debe establecerse en `true`. El valor predeterminado es `true`.

### Ejemplo

En el ejemplo siguiente se impide la posibilidad de ordenar la columna del índice 1:

```
myGrid.getColumnAt(1).sortable = false;
```

## DataGridColumn.sortOnHeaderRelease

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.getColumnAt(index).sortOnHeaderRelease
```

### Descripción

Propiedad; valor booleano que indica si la columna se ordenará automáticamente cuando el usuario haga clic en el encabezado (`true`) o no (`false`). El valor de esta propiedad sólo puede ser `true` si el de `DataGridColumn.sortable` es `true`. Si el valor de `DataGridColumn.sortOnHeaderRelease` está establecido en `false`, el usuario puede capturar el evento `headerRelease` para realizar su propia ordenación.

El valor predeterminado es `true`.

### Ejemplo

En el ejemplo siguiente se permite al usuario capturar el evento `headerRelease` para que realice su propia ordenación:

```
myGrid.getColumnAt(7).sortOnHeaderRelease = false;
```

## DataGridColumn.width

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDataGrid.getColumnAt(index).width
```

### Descripción

Propiedad; número que indica la anchura de la columna, expresada en píxeles. El valor predeterminado es 50.

## Ejemplo

En el ejemplo siguiente se reduce el tamaño de una columna a la mitad del valor predeterminado:

```
myGrid.getColumnAt(4).width = 25;
```

## Componente DataHolder (sólo para Flash Professional)

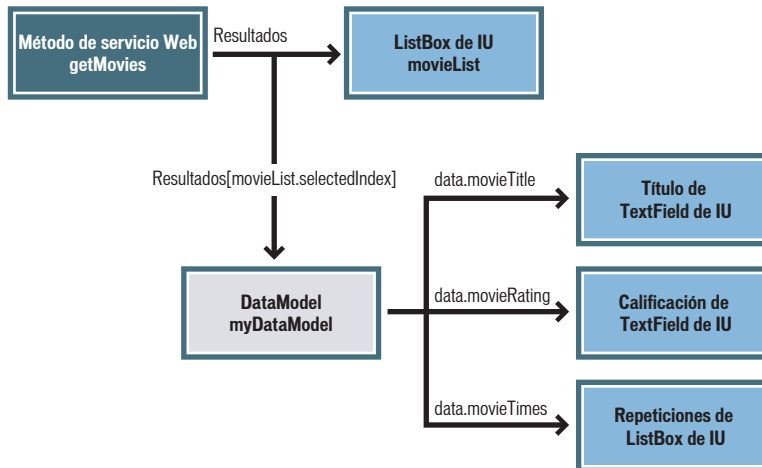
El componente DataHolder es un depósito para datos y un medio de generar eventos cuando se modifican dichos datos. Su función principal consiste en almacenar datos y actuar como conector de otros componentes mediante la vinculación de los datos.

Inicialmente, el componente DataHolder tiene una única propiedad que puede vincularse denominada `data`. Es posible añadir más propiedades utilizando la ficha Esquema del panel Inspector de componentes (Ventana > Paneles de desarrollo > Inspector de componentes). Para más información sobre el uso de la ficha Esquema, consulte “Trabajo con esquemas en la ficha Esquema (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

El usuario puede asignar cualquier tipo de datos a una propiedad DataHolder, ya sea creando una vinculación entre los datos y otra propiedad o utilizando su propio código ActionScript. Cuando el valor de los datos cambia, el componente DataHolder emite un evento cuyo nombre coincide con el de la propiedad y se ejecutan las vinculaciones asociadas con dicha propiedad.

El componente DataHolder resulta de utilidad cuando no es posible vincular componentes directamente entre sí (por ejemplo conectores, componentes de interfaz de usuario o componentes DataSet). A continuación se muestran algunos supuestos en los que se utiliza un componente DataHolder:

- Si un valor de datos se genera mediante código ActionScript, es posible que haya que vincularlo a otros componentes. En este caso, puede obtenerse un componente DataHolder que contenga propiedades vinculadas de la forma deseada. Cuando se asignen nuevos valores a esas propiedades (mediante ActionScript, por ejemplo), dichos valores se distribuirán al objeto que esté vinculado a los datos.
- Es posible que exista un valor de datos resultante de una vinculación de datos indexada compleja, como se muestra en el diagrama siguiente.



En este caso es conveniente vincular el valor de datos a un componente DataHolder (denominado *DataModel* en esta ilustración) y, a continuación, utilizarlo para realizar vinculaciones a la interfaz.

## Creación de una aplicación con el componente DataHolder (sólo para Flash Professional)

En este ejemplo se añade una propiedad de matriz al esquema del componente DataHolder (una matriz) cuyo valor lo determina el código ActionScript escrito por el usuario. Después se podrá vincular esa propiedad de matriz a la propiedad `dataProvider` de un componente DataGrid utilizando la ficha Vinculaciones del panel Inspector de componentes.

### Para utilizar el componente DataHolder en una aplicación sencilla:

- 1 En Flash MX Professional 2004, cree un nuevo archivo.
- 2 Abra el panel Componentes (Ventana > Paneles de desarrollo > Componentes), arrastre un componente TextArea al escenario y asígnele el nombre **dataHolder**.
- 3 Arrastre un componente DataGrid al escenario y asígnele el nombre **namesGrid**.
- 4 Seleccione el componente DataHolder y abra el panel Inspector de componentes (Ventana > Paneles de desarrollo > Inspector de componentes).
- 5 Haga clic en la ficha Esquema del panel Inspector de componentes.
- 6 Haga clic en el botón Añadir una propiedad de componente (+) situado en el panel superior de la ficha Esquema.
- 7 En el panel inferior de la ficha Esquema, escriba **namesArray** en el campo field name y seleccione Array en la lista desplegable data type.
- 8 Haga clic en la ficha Vinculaciones del panel Inspector de componentes y añada una vinculación entre la propiedad **namesArray** del componente DataHolder y la propiedad `dataProvider` del componente DataGrid.

Para más información sobre la creación de vinculaciones con la ficha Vinculaciones, consulte “Trabajo con vinculaciones en la ficha Vinculaciones (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

- 9 En la línea de tiempo (Ventana > Línea de tiempo), seleccione el primer fotograma de la capa 1 y abra el panel Acciones (Ventana > Paneles de desarrollo > Acciones).
- 10 Introduzca el código siguiente en el panel Acciones:

```
dataHolder.namesArray= [{name:"Tim"},{name:"Paul"},{name:"Jason"}];
```

Con este código se llena la matriz **namesArray** con varios objetos. Cuando se ejecuta esta asignación de variables, se ejecuta también la vinculación que se estableció con anterioridad entre el componente DataHolder y el componente DataGrid.

- 11 Para probar el archivo, seleccione Control > Probar película.

## Resumen de propiedades de la clase DataHolder

| Propiedad                    | Descripción                                                        |
|------------------------------|--------------------------------------------------------------------|
| <code>DataHolder.data</code> | Propiedad vinculable predeterminada para el componente DataHolder. |

## DataHolder.data

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`dataHolder.data`

### Descripción

Propiedad; elemento predeterminado del esquema de un objeto DataHolder. Esta propiedad no es una parte “permanente” del componente DataHolder. Se trata, por el contrario, de una propiedad vinculable predeterminada para cada instancia del componente. El usuario puede añadir propiedades vinculables propias o eliminar la propiedad `data` predeterminada utilizando la ficha Esquema del panel Inspector de componentes.

Para más información sobre el uso de la ficha Esquema, consulte “Trabajo con esquemas en la ficha Esquema (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

### Ejemplo

Para ver un ejemplo de cómo utilizar este componente, consulte [“Creación de una aplicación con el componente DataHolder \(sólo para Flash Professional\)” en la página 199.](#)

## Interfaz API de DataProvider

**Nombre de clase de ActionScript** `mx.controls.listclasses.DataProvider`

La interfaz API de DataProvider es un conjunto de métodos y propiedades que una fuente de datos debe tener para que una clase basada en listas pueda comunicarse con dicha fuente. Arrays, RecordSets y DataSet implementan esta interfaz API. Se puede crear una clase compatible con el proveedor de datos si se implementan todos los métodos y propiedades descritos en este documento. Un componente basado en listas podría entonces utilizar esa clase como proveedor de datos.

Los métodos de la interfaz API de DataProvider permiten consultar y modificar datos de los componentes que muestren datos (también denominados *vistas*). La interfaz API de DataProvider también difunde eventos de cambio si cambian los datos. Distintas vistas pueden utilizar el mismo proveedor de datos y recibir eventos de cambio.

Un proveedor de datos es una colección lineal (parecida a una matriz) de elementos. Cada elemento es un objeto compuesto de muchos campos de datos. Se puede acceder a cada elemento a través del índice correspondiente (tal como se haría en una matriz) mediante `DataProvider.getItemAt()`.

Normalmente los proveedores de datos se utilizan con matrices. Los componentes para datos aplican todos los métodos de la interfaz API del proveedor de datos a `Array.prototype` cuando hay un objeto Array en el mismo fotograma o en la misma pantalla donde está el componente para datos. De esta forma se puede utilizar una matriz existente como datos de las vistas que tengan una propiedad `dataProvider`.

Gracias a la interfaz API de DataProvider, los componentes de la v2 que permiten obtener vistas de datos (DataGrid, List, Tree, etc.) también pueden mostrar juegos de registros de Flash Remoting y datos del componente DataSet. La interfaz API de DataProvider es el lenguaje con el que los componentes para datos se comunican con sus proveedores de datos.

En la documentación de Macromedia Flash “DataProvider” es el nombre de la interfaz API, `dataProvider` es una propiedad de cada componente que permite obtener una vista de los datos y “proveedor de datos” es la expresión genérica que se emplea para indicar la fuente de datos.

## Métodos de la interfaz API de DataProvider

| Nombre                                     | Descripción                                                                                  |
|--------------------------------------------|----------------------------------------------------------------------------------------------|
| <code>DataProvider.addItem()</code>        | Añade un elemento al final del proveedor de datos.                                           |
| <code>DataProvider.addItemAt()</code>      | Añade un elemento al proveedor de datos en la posición especificada.                         |
| <code>DataProvider.editField()</code>      | Cambia un campo del proveedor de datos.                                                      |
| <code>DataProvider.getEditingData()</code> | Obtiene los datos para su edición desde un proveedor de datos.                               |
| <code>DataProvider.getItemAt()</code>      | Obtiene una referencia al elemento en la posición especificada.                              |
| <code>DataProvider.getItemID()</code>      | Devuelve el ID exclusivo del elemento.                                                       |
| <code>DataProvider.removeAll()</code>      | Elimina todos los elementos de un proveedor de datos.                                        |
| <code>DataProvider.removeItemAt()</code>   | Elimina un elemento del proveedor de datos en una posición especificada.                     |
| <code>DataProvider.replaceItemAt()</code>  | Sustituye por otro el elemento de una posición especificada.                                 |
| <code>DataProvider.sortItems()</code>      | Ordena los elementos de un proveedor de datos.                                               |
| <code>DataProvider.sortItemsBy()</code>    | Ordena los elementos de un proveedor de datos según una función de comparación especificada. |

## Propiedades de la interfaz API de DataProvider

| Nombre                           | Descripción                                   |
|----------------------------------|-----------------------------------------------|
| <code>DataProvider.length</code> | Número de elementos de un proveedor de datos. |

## Eventos de la interfaz API de DataProvider

| Nombre                                 | Descripción                                     |
|----------------------------------------|-------------------------------------------------|
| <code>DataProvider.modelChanged</code> | Se difunde cuando cambia el proveedor de datos. |

## DataProvider.addItem()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDP.addItem(item)
```

### Parámetros

*item* Objeto que contiene datos. Incluye también cualquier elemento de un proveedor de datos.

### Valor devuelto

Ninguno.

### Descripción

Método; añade un elemento nuevo al final del proveedor de datos.

Este método activa el evento `modelChanged` con el nombre de evento `addItem`s.

### Ejemplo

En el ejemplo siguiente se añade un elemento al final del proveedor de datos `myDP`:

```
myDP.addItem({ label: "this is an Item" });
```

## DataProvider.addItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDP.addItemAt(index, item)
```

### Parámetros

*index* Número igual o mayor que 0. Posición en la que insertar el elemento; índice del nuevo elemento.

*item* Objeto que contiene los datos del elemento.

### Valor devuelto

Ninguno.

### Descripción

Método; añade un nuevo elemento al proveedor de datos en el índice especificado. Se omiten los índices cuya longitud sea mayor que la del proveedor de datos.

Este método activa el evento `modelChanged` con el nombre de evento `addItem`s.

### Ejemplo

En el ejemplo siguiente se añade un elemento al proveedor de datos `myDP` en la cuarta posición:

```
myDP.addItemAt(3, {label : "this is the fourth Item"});
```

## DataProvider.editField()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDP.editField(index, fieldName, newData)
```

### Parámetros

*index* Número mayor o igual que cero. Índice del elemento.

*fieldName* Cadena que indica el nombre de campo del elemento que se modificará.

*newData* Nuevos datos que se colocarán en el proveedor de datos.

### Valor devuelto

Ninguno.

### Descripción

Método; cambia un campo del proveedor de datos.

Este método activa el evento `modelChanged` con el nombre de evento `updateField`.

### Ejemplo

En el código siguiente se modifica el campo `label` del tercer elemento:

```
myDP.editField(2, "label", "mynewData");
```

## DataProvider.getEditingData()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDP.getEditingData(index, fieldName)
```

### Parámetros

*index* Número mayor o igual que 0 y menor que `DataProvider.length`. Índice del elemento que debe recuperarse.

*fieldName* Cadena que indica el nombre del campo que se va a editar.

### Valor devuelto

Los datos con formato editables que se utilizarán.

### Descripción

Método; recupera los datos para su edición desde un proveedor de datos. De esta forma el modelo de datos puede proporcionar distintos formatos de datos para su edición y visualización.

### Ejemplo

Con el código siguiente se obtiene una cadena editable para el campo de precio (price):

```
trace(myDP.getEditingData(4, "price");
```

## DataProvider.getItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDP.getItemAt(index)
```

### Parámetros

*index* Número mayor o igual que 0 y menor que `DataProvider.length`. Índice del elemento que debe recuperarse.

### Valor devuelto

Una referencia al elemento recuperado; si el índice está fuera de rango, el valor es `undefined`.

### Descripción

Método; recupera una referencia al elemento en la posición especificada.

### Ejemplo

En el código siguiente se muestra la etiqueta del quinto elemento:

```
trace(myDP.getItemAt(4).label);
```

## DataProvider.getItemID()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 Professional.

### Sintaxis

```
myDP.getItemID(index)
```

### Parámetros

*index* Número mayor o igual que 0.



## Valor devuelto

Un número que es el ID exclusivo del elemento.

## Descripción

Método; devuelve un ID exclusivo del elemento. Este método se emplea principalmente en el seguimiento de una selección. Este ID se utiliza en componentes para datos con el fin de disponer de listas con los elementos seleccionados.

## Ejemplo

En este ejemplo se obtiene el ID del cuarto elemento:

```
var ID = myDP.getItemID(3);
```

## DataProvider.modelChanged

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.modelChanged = function(eventObject){  
    // introducir aquí el código propio  
}  
myMenu.addEventListener("modelChanged", listenerObject
```

### Descripción

Evento; se difunde a todos los detectores de vistas cuando se modifica el proveedor de datos. Normalmente, un detector se añade a un modelo mediante la asignación de la propiedad `dataProvider` correspondiente.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando el proveedor de datos experimenta algún cambio, difunde un evento `modelChanged` y los componentes para datos lo capturan para actualizar sus vistas, de forma que queden reflejados los cambios que hayan sufrido los datos.

El objeto del evento `Menu.modelChanged` tiene cinco propiedades adicionales:

- **eventName** La propiedad `eventName` se utiliza para dividir los eventos `modelChanged` en subcategorías. Los componentes para datos utilizan esta información para evitar tener que actualizar completamente la instancia (o vista) del componente que esté utilizando el proveedor de datos. A continuación se enumeran los valores admitidos de la propiedad `eventName`:
  - **updateAll** Es necesario actualizar toda la vista, excluida la posición de desplazamiento.
  - **addItem** Se ha añadido una serie de elementos.
  - **removeItems** Se ha eliminado una serie de elementos.
  - **updateItems** Es necesario actualizar una serie de elementos.
  - **sort** Se han ordenado los datos.
  - **updateField** Un campo de un elemento debe cambiarse y actualizarse.

- `updateColumn` Es necesario actualizar la definición de la totalidad de un campo que se encuentra dentro del proveedor de datos.
- `filterModel` El modelo se ha filtrado y es necesario actualizar la vista (restablecer `scrollTop`).
- `schemaLoaded` Se ha declarado la definición del campo del proveedor de datos.
- `firstItem` Índice del primer elemento afectado.
- `lastItem` Índice del último elemento afectado. El valor es igual al de `firstItem` si se ve afectado un único elemento.
- `removedIDs` Matriz de los identificadores de elemento que se han eliminado.
- `fieldName` Cadena que indica el nombre del campo que se verá afectado.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 602.

### Ejemplo

En el ejemplo siguiente, se define un controlador denominado `listener` y luego se pasa éste al método `addEventListener()` como segundo parámetro. El controlador `modelChanged` captura el objeto de evento en el parámetro `evt`. Cuando se difunde el evento `modelChanged`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
listener = new Object();
listener.modelChanged = function(evt){
    trace(evt.eventName);
}
myList.addEventListener("modelChanged", listener);
```

## DataProvider.length

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myDP.length`

### Descripción

Propiedad (sólo lectura); número de elementos del proveedor de datos.

### Ejemplo

En este ejemplo se envía al panel Salida una cantidad elementos equivalente al número especificado en el proveedor de datos `myArray`:

```
trace(myArray.length);
```

## **DataProvider.removeAll()**

### **Disponibilidad**

Flash Player 6 versión 79.

### **Edición**

Flash MX Professional 2004.

### **Sintaxis**

```
myDP.removeAll()
```

### **Parámetros**

Ninguno.

### **Valor devuelto**

Ninguno.

### **Descripción**

Método; elimina todos los elementos del proveedor de datos.

Este método activa el evento `modelChanged` con el nombre de evento `removeItems`.

### **Ejemplo**

En este ejemplo se eliminan todos los elementos del proveedor de datos:

```
myDP.removeAll();
```

## **DataProvider.removeItemAt()**

### **Disponibilidad**

Flash Player 6 versión 79.

### **Edición**

Flash MX Professional 2004.

### **Sintaxis**

```
myDP.removeItemAt(index)
```

### **Parámetros**

*index* Número mayor o igual que cero. Índice del elemento que se va a eliminar.

### **Valor devuelto**

Ninguno.

### **Descripción**

Método; elimina el elemento del índice especificado. Los índices que se encuentran después del índice eliminado se contraerán una posición.

Este método activa el evento `modelChanged` con el nombre de evento `removeItems`.

### Ejemplo

En este ejemplo se elimina el elemento de la cuarta posición.

```
myDP.removeItemAt(3);
```

## DataProvider.replaceItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDP.replaceItemAt(index, item)
```

### Parámetros

*index* Número mayor o igual que 0. Índice del elemento que va a modificarse.

*item* Objeto que es el nuevo elemento.

### Valor devuelto

Ninguno.

### Descripción

Método; sustituye el contenido del elemento del índice especificado.

Este método activa el evento `modelChanged` con el nombre de evento `removeItems`.

### Ejemplo

En este ejemplo se sustituye el elemento del índice 3 por el elemento que tiene la etiqueta “new label”:

```
myDP.replaceItemAt(3, {label : "new label"});
```

## DataProvider.sortItems()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDP.sortItems([compareFunc], [optionsFlag])
```

## Parámetros

*compareFunc* Referencia a una función que se utiliza para comparar dos elementos y determinar su orden de clasificación. Para más información, consulte `Array.sort()` en el apartado Diccionario de ActionScript de la Ayuda. Este parámetro es opcional.

*optionsFlag* Permite realizar muchos tipos distintos de ordenaciones en una única matriz sin necesidad de duplicar toda la matriz ni de volver a ordenarla repetidas veces. Este parámetro es opcional.

Los valores posibles de *optionsFlag* son:

- `Array.DESENDING`: ordena de mayor a menor.
- `Array.CASEINSENSITIVE`: ordena sin distinguir entre mayúsculas y minúsculas.
- `Array.NUMERIC`: ordena numéricamente si los dos elementos comparados son números. Si no se trata de números, utilice la comparación de cadenas (se puede realizar sin distinguir entre mayúsculas y minúsculas si se ha especificado el indicador correspondiente).
- `Array.UNIQUESORT`: si dos objetos de la matriz son idénticos o lo son sus campos de ordenación, este método devuelve un código de error (0) en lugar de una matriz ordenada.
- `Array.RETURNINDEXEDARRAY`: devuelve una matriz de índice de número entero que es el resultado de la ordenación. Por ejemplo, si se ordenase la siguiente matriz especificando el valor `Array.RETURNINDEXEDARRAY` en el parámetro *optionsFlag*, se devolvería la segunda línea de código y la matriz se mantendría sin alteraciones:

```
["a", "d", "c", "b"]  
[0, 3, 2, 1]
```

Estas opciones se pueden combinar en un único valor. Por ejemplo, en el código siguiente se combinan las opciones 3 y 1:

```
array.sort (Array.NUMERIC | Array.DESENDING)
```

## Valor devuelto

Ninguno.

## Descripción

Método; ordena los elemento del proveedor de datos de acuerdo con la función de comparación especificada en el parámetro *compareFunc* o de acuerdo con una o varias de las opciones especificadas en el parámetro *optionsFlag*.

Este método activa el evento `modelChanged` con el nombre de evento `sort`.

## Ejemplo

En este ejemplo la ordenación se basa en las etiquetas en mayúsculas. Los elementos a y b se pasan a la función y contienen campos `label` y `data`:

```
myList.sortItems(upperCaseFunc);  
function upperCaseFunc(a,b){  
    return a.label.toUpperCase() > b.label.toUpperCase();  
}
```

## DataProvider.sortItemsBy()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDP.sortItemsBy(fieldName, order, [optionsFlag])
```

### Parámetros

*fieldName* Cadena que especifica el nombre del campo que se va a utilizar para la ordenación. Normalmente, este valor es "label" o "data".

*order* Cadena que especifica si los elementos deben ordenarse en sentido ascendente ("ASC") o descendente ("DESC").

*optionsFlag* Permite realizar muchos tipos distintos de ordenaciones en una única matriz sin necesidad de duplicar toda la matriz ni de volver a ordenarla repetidas veces. Este parámetro es opcional.

Los valores posibles de *optionsFlag* son:

- `Array.DESENDING`: ordena de mayor a menor.
- `Array.CASEINSENSITIVE`: ordena sin distinguir entre mayúsculas y minúsculas.
- `Array.NUMERIC`: ordena numéricamente si los dos elementos comparados son números. Si no se trata de números, utilice la comparación de cadenas (se puede realizar sin distinguir entre mayúsculas y minúsculas si se ha especificado el indicador correspondiente).
- `Array.UNIQUESORT`: si dos objetos de la matriz son idénticos o lo son sus campos de ordenación, este método devuelve un código de error (0) en lugar de una matriz ordenada.
- `Array.RETURNINDEXEDARRAY`: devuelve una matriz de índice de número entero que es el resultado de la ordenación. Por ejemplo, si se ordenase la siguiente matriz especificando el valor `Array.RETURNINDEXEDARRAY` en el parámetro *optionsFlag*, se devolvería la segunda línea de código y la matriz se mantendría sin alteraciones:

```
["a", "d", "c", "b"]  
[0, 3, 2, 1]
```

Estas opciones se pueden combinar en un único valor. Por ejemplo, en el código siguiente se combinan las opciones 3 y 1:

```
array.sort (Array.NUMERIC | Array.DESENDING)
```

### Valor devuelto

Ninguno.

## Descripción

Método; ordena los elementos del proveedor de datos alfabéticamente o numéricamente y en el orden especificado mediante el nombre de campo especificado. Si los elementos de *fieldName* son una combinación de cadenas de texto y enteros, los enteros aparecen en primer lugar. El parámetro *fieldName* suele ser "label" o "data", pero los programadores avanzados pueden especificar cualquier valor primitivo. También existe la opción de utilizar el parámetro *optionsFlag* para especificar un estilo de ordenación.

Este método activa el evento `modelChanged` con el nombre de evento `sort`.

## Ejemplo

En el código siguiente, los elementos de una lista se ordenan en sentido ascendente por etiquetas:

```
myDP.sortItemsBy("label", "ASC");
```

## Componente DataSet (sólo para Flash Professional)

El componente DataSet permite al usuario trabajar con datos como colecciones de objetos con los que se pueden realizar operaciones como indexar, ordenar, buscar, filtrar y modificar.

La función del componente DataSet incluye DataSetIterator, un conjunto de métodos que permite atravesar y manipular una colección de datos, y DeltaPacket, un conjunto de interfaces y clases que permiten trabajar con actualizaciones de colecciones de datos. En la mayoría de los casos, dichas clases e interfaces no se utilizan directamente sino indirectamente mediante métodos establecidos por la clase DataSet.

Los elementos administrados por el componente DataSet también se denominan *objetos de transferencia*. Un objeto de transferencia muestra los datos de negocio que se encuentran en el servidor con atributos públicos o métodos de acceso para la lectura y escritura de datos. El componente DataSet permite a los desarrolladores trabajar con objetos sofisticados de la parte del cliente que reflejan su correspondiente de la parte del servidor; o, en su forma más sencilla, una colección de objetos anónimos con atributos públicos que representan los campos de un registro de datos. Para más información sobre los objetos de transferencia, consulte Core J2EE Patterns Transfer Object en [java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html](http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html).

**Nota:** el componente DataSet necesita Flash Player 7 o una versión posterior.

## Utilización del componente DataSet (sólo para Flash Professional)

Normalmente, en una aplicación se utiliza el componente DataSet en combinación con otros componentes para manipular y actualizar una fuente de datos: un componente Connector, que permite realizar una conexión con una fuente de datos externa, componentes de interfaz de usuario, que permiten visualizar datos de la fuente de datos, y un componente Resolver, que permite convertir actualizaciones realizadas en el juego de datos en un formato adecuado para enviarlo a la fuente de datos externa. A continuación podrá utilizarse la vinculación de datos para vincular entre sí las propiedades de los distintos componentes.

Para más información general sobre el componente DataSet y sobre cómo utilizarlo con otros componentes, consulte "Administración de datos (sólo para Flash Professional)" en el apartado Utilización de Flash de la Ayuda.

## Parámetros del componente DataSet

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente DataSet en el inspector de propiedades o en el panel Inspector de componentes:

**itemClassName** Nombre de la clase del objeto de transferencia del que se creará una instancia cada vez que sea necesario un elemento nuevo.

**Nota:** para que la clase especificada esté disponible en tiempo de ejecución, también es necesario hacer una referencia calificada a la clase en algún lugar del código del archivo SWF (por ejemplo, `var myItem:my.package.myItem;`).

**filtered** Si el valor es `true`, se aplica un filtro al juego de datos de manera que contenga únicamente los objetos que coincidan con los criterios de filtrado.

**logChanges** Si el valor es `true`, el juego de datos registra todas las mutaciones (cambios sobre los datos o las llamadas de método) que experimente la propiedad `deltaPacket`.

**readOnly** Si el valor es `true`, el juego de datos no puede modificarse.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales del componente DataSet utilizando sus propiedades, métodos y eventos. Para más información, consulte [“Clase DataSet \(sólo para Flash Professional\)” en la página 214](#).

## Creación de aplicaciones con el componente DataSet

Normalmente se utiliza el componente DataSet con otros componentes de interfaz de usuario que a menudo son componentes Connector, como XMLConnector o WebServiceConnector. Los elementos del juego de datos se llenan mediante el componente Connector o datos ActionScript sin formato y, a continuación, se vinculan con controles de interfaz de usuario (como los componentes List o DataGrid).

### Para crear una aplicación con el componente DataSet:

- 1 En Flash MX Professional 2004, seleccione Archivo > Nuevo. En la columna Tipo, seleccione Documento de Flash y haga clic en Aceptar.
- 2 Abra el panel Componentes (Ventana > Paneles de desarrollo > Componentes), si todavía no está abierto.
- 3 Arrastre un componente DataSet desde el panel Componentes al escenario. En el inspector de propiedades, asígnele el nombre **userData**.
- 4 Arrastre un componente DataGrid al escenario y asígnele el nombre **userGrid**.
- 5 Cambie el tamaño del componente DataGrid para que tenga aproximadamente 300 píxeles de ancho y 100 de alto.
- 6 Arrastre un componente Button al escenario y asígnele el nombre **nextBtn**.
- 7 En la línea de tiempo, seleccione el primer fotograma de la capa 1 y abra el panel Acciones (Ventana > Paneles de desarrollo > Acciones).
- 8 Añada el código siguiente al panel Acciones:

```
var recData = [{id:0, firstName:"Mick", lastName:"Jones"},
               {id:1, firstName:"Joe", lastName:"Strummer"},
               {id:2, firstName:"Paul", lastName:"Simonon"}];
userData.items = recData;
```

De esta forma se llena la propiedad `items` del objeto DataSet con una matriz de objetos que tienen tres propiedades cada uno: `firstName`, `lastName` e `id`.



- 9 Para vincular el contenido del componente DataSet con el del componente DataGrid, abra el panel Inspector de componentes (Ventana > Paneles de desarrollo > Inspector de componentes) y haga clic en la ficha Vinculaciones.
- 10 Seleccione el componente DataGrid (`userGrid`) en el escenario y haga clic en el botón Añadir vinculación (+) en el panel Inspector de componentes.
- 11 En el cuadro de diálogo Añadir vinculación, seleccione “dataProvider : Array” y haga clic en Aceptar.
- 12 Haga doble clic en el campo bound to del panel Inspector de componentes.
- 13 En el cuadro de diálogo Vinculado a que aparece seleccione “DataSet <userData>” en la columna Ruta del componente y, a continuación, seleccione “dataProvider : Array” en la columna Ubicación del esquema.
- 14 Para vincular el índice seleccionado del componente DataSet al índice seleccionado del componente DataGrid, haga clic nuevamente en el botón Añadir vinculación (+) del panel Inspector de componentes.
- 15 En el cuadro de diálogo que aparece, seleccione “selectedIndex : Number”. Haga clic en Aceptar.
- 16 Haga doble clic en el campo bound to del panel Inspector de componentes para abrir el cuadro de diálogo Vinculado a.
- 17 Seleccione “DataSet <userData>” en el campo Ruta del componente y, a continuación, seleccione “selectedIndex : Number” en la columna Ubicación del esquema.
- 18 Seleccione el componente Button (`nextBtn`) y abra el panel Acciones (Ventana > Paneles de desarrollo > Acciones) si todavía no está abierto.
- 19 Introduzca el código siguiente en el panel Acciones:

```
on(click){  
    _parent.userData.next();  
}
```

En este código se utiliza el método `DataSet.next()` para desplazarse al siguiente elemento de la colección de elementos del objeto DataSet. Como ya se había vinculado con anterioridad la propiedad `selectedIndex` del objeto DataGrid a la misma propiedad del objeto DataSet, si se cambia el elemento actual del objeto DataSet también cambiará el elemento actual (o seleccionado) del objeto DataGrid.

- 20 Guarde el archivo y seleccione Control > Probar película para probar el archivo SWF.  
El objeto DataGrid se llenará con los elementos especificados. Fíjese en cómo al hacer clic en el botón cambia el elemento seleccionado en el objeto DataGrid.

## Clase DataSet (sólo para Flash Professional)

Nombre de clase de ActionScript    `mx.data.components.DataSet`

### Resumen de métodos de la clase DataSet

| Método                                   | Descripción                                                                                               |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| <code>DataSet.addItem()</code>           | Añade el elemento seleccionado a la colección.                                                            |
| <code>DataSet.addSort()</code>           | Crea una nueva vista ordenada de los elementos de la colección.                                           |
| <code>DataSet.applyUpdates()</code>      | Notifica a los detectores que los cambios realizados en el objeto DataSet ya están preparados.            |
| <code>DataSet.changesPending()</code>    | Indica si existen elementos o no en el objeto DeltaPacket.                                                |
| <code>DataSet.clear()</code>             | Elimina todos los elementos de la vista actual de la colección.                                           |
| <code>DataSet.createItem()</code>        | Devuelve un elemento de la colección que acaba de inicializarse.                                          |
| <code>DataSet.disableEvents()</code>     | Detiene el envío de eventos DataSet a los detectores.                                                     |
| <code>DataSet.enableEvents()</code>      | Reanuda el envío de eventos DataSet a los detectores.                                                     |
| <code>DataSet.find()</code>              | Busca un elemento en la vista actual de la colección.                                                     |
| <code>DataSet.findFirst()</code>         | Busca la primera vez que aparece un elemento en la vista actual de la colección.                          |
| <code>DataSet.findLast()</code>          | Busca la última vez que aparece un elemento en la vista actual de la colección.                           |
| <code>DataSet.first()</code>             | Se desplaza al primer elemento de la vista actual de la colección.                                        |
| <code>DataSet.getItemId()</code>         | Devuelve el ID exclusivo del elemento especificado.                                                       |
| <code>DataSet.getIterator()</code>       | Devuelve un clon del repetidor actual.                                                                    |
| <code>DataSet.hasNext()</code>           | Indica si el repetidor actual se encuentra o no al final de la vista correspondiente de la colección.     |
| <code>DataSet.hasPrevious()</code>       | Indica si el repetidor actual se encuentra o no al principio de la vista correspondiente de la colección. |
| <code>DataSet.hasSort()</code>           | Indica si la ordenación especificada existe o no.                                                         |
| <code>DataSet.isEmpty()</code>           | Indica si la colección contiene elementos o no.                                                           |
| <code>DataSet.last()</code>              | Se desplaza al último elemento de la vista actual de la colección.                                        |
| <code>DataSet.loadFromSharedObj()</code> | Recupera el contenido de un objeto DataSet a partir de un objeto compartido.                              |
| <code>DataSet.locateById()</code>        | Desplaza el repetidor actual al elemento que tenga el ID especificado.                                    |
| <code>DataSet.next()</code>              | Se desplaza al siguiente elemento de la vista actual de la colección.                                     |
| <code>DataSet.previous()</code>          | Se desplaza al elemento anterior de la vista actual de la colección.                                      |
| <code>DataSet.removeAll()</code>         | Elimina todos los elementos de la colección.                                                              |
| <code>DataSet.removeItem()</code>        | Elimina el elemento especificado de la colección.                                                         |

| Método                                 | Descripción                                                                                                     |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>DataSet.removeRange()</code>     | Elimina la configuración del rango del repetidor actual.                                                        |
| <code>DataSet.removeSort()</code>      | Elimina el orden especificado del objeto DataSet.                                                               |
| <code>DataSet.saveToSharedObj()</code> | Guarda los datos del objeto DataSet en un objeto compartido.                                                    |
| <code>DataSet.setIterator()</code>     | Establece el repetidor actual del objeto DataSet.                                                               |
| <code>DataSet.setRange()</code>        | Establece la configuración del rango del repetidor actual.                                                      |
| <code>DataSet.skip()</code>            | Se desplaza hacia delante o hacia atrás un número especificado de elementos en la vista actual de la colección. |
| <code>DataSet.useSort()</code>         | Convierte en ordenación activa la que se haya especificado.                                                     |

## Resumen de propiedades de la clase DataSet

| Propiedad                          | Descripción                                                                                             |
|------------------------------------|---------------------------------------------------------------------------------------------------------|
| <code>DataSet.currentItem</code>   | Devuelve el elemento actual de la colección.                                                            |
| <code>DataSet.dataProvider</code>  | Devuelve la interfaz de DataProvider.                                                                   |
| <code>DataSet.deltaPacket</code>   | Devuelve los cambios realizados en la colección o asigna los cambios que se realizarán en la colección. |
| <code>DataSet.filtered</code>      | Indica si los elementos están filtrados o no.                                                           |
| <code>DataSet.filterFunc</code>    | Función definida por el usuario para filtrar elementos de la colección.                                 |
| <code>DataSet.items</code>         | Elementos de la colección.                                                                              |
| <code>DataSet.itemClassName</code> | Objeto que se creará al asignar elementos.                                                              |
| <code>DataSet.length</code>        | Especifica el número de elementos de la vista actual de la colección.                                   |
| <code>DataSet.logChanges</code>    | Indica si se registran o no los cambios realizados en la colección o en sus elementos.                  |
| <code>DataSet.properties</code>    | Contiene las propiedades (o los campos) de los objetos de transferencia que existen en esta colección.  |
| <code>DataSet.readOnly</code>      | Indica si la colección puede modificarse o no.                                                          |
| <code>DataSet.schema</code>        | Especifica el esquema de la colección en formato XML.                                                   |
| <code>DataSet.selectedIndex</code> | Contiene el índice del elemento actual que se encuentra dentro de la colección.                         |

## Resumen de eventos de la clase DataSet

| Evento                                  | Descripción                                                                                                     |
|-----------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>DataSet.addItem</code>            | Se difunde antes de añadir un elemento a la colección.                                                          |
| <code>DataSet.afterLoaded</code>        | Se difunde después de asignar la propiedad <code>items</code> .                                                 |
| <code>DataSet.deltaPacketChanged</code> | Se difunde cuando el paquete delta del objeto DataSet ha sufrido alteraciones y está preparado para utilizarse. |
| <code>DataSet.calcFields</code>         | Se difunde cuando deben actualizarse los campos calculados.                                                     |
| <code>DataSet.iteratorScrolled</code>   | Se difunde cuando cambia la posición del repetidor.                                                             |
| <code>DataSet.modelChanged</code>       | Se difunde cuando se ha realizado algún cambio en los elementos de la colección.                                |
| <code>DataSet.newItem</code>            | Se difunde cuando el objeto DataSet genera un nuevo elemento, pero antes de añadirlo a la colección.            |
| <code>DataSet.removeItem</code>         | Se difunde antes de eliminar un elemento.                                                                       |
| <code>DataSet.resolveDelta</code>       | Se difunde cuando se asigna un objeto DeltaPacket al objeto DataSet que contiene mensajes.                      |

### DataSet.addItem

#### Disponibilidad

Flash Player 7.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

```
on(addItem) {  
    // introducir aquí el código propio  
}  
listenerObject = new Object();  
listenerObject.addItem = function (eventObj) {  
    // introducir aquí el código propio  
}  
dataSet.addEventListener("addItem", listenerObject)
```

#### Descripción

Evento; se genera justo antes de insertar un objeto de transferencia a esta colección.

Si se especifica el valor `false` en la propiedad `result` del objeto en el que se ha producido el evento, se cancelará la operación de adición; en cambio si se establece el valor `true`, podrá realizarse dicha operación.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

**target** Objeto DataSet que generó el evento.

**type** Cadena "addItem".

**item** Referencia al elemento de la colección que se va a añadir.

**result** Valor booleano que especifica si el elemento especificado debe añadirse o no. De forma predeterminada, este valor es true.

### Ejemplo

El controlador de eventos `on(addItem)` siguiente (asociado con un objeto DataSet) cancela la adición del nuevo elemento si una función definida por el usuario denominada `userHasAdminPrivs()` devuelve el valor `false`; de lo contrario, sí se admite la adición.

```
on(addItem) {  
    if(globalObj.userHasAdminPrivs()) {  
        // Permitir la adición del elemento.  
        eventObj.result = true;  
    } else {  
        // No permitir la adición del elemento; el usuario no tiene privilegios de  
        administrador.  
        eventObj.result = false;  
    }  
}
```

### Véase también

[DataSet.removeItem](#)

## DataSet.addItem()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.addItem([obj])
```

### Parámetros

*obj* Objeto que se añadirá a esta colección. Este parámetro es opcional.

### Valor devuelto

Devuelve `true` si el elemento se añadió a la colección y `false` en caso contrario.

## Descripción

Método; añade a la colección el objeto de transferencia especificado para que pueda administrarse. El elemento recién añadido se convierte en el elemento actual del juego de datos. Si no se especifica ningún parámetro *obj*, se crea automáticamente un nuevo objeto mediante `DataSet.createItem()`.

La ubicación del nuevo elemento en la colección depende de si se ha especificado o no una ordenación para el repetidor actual. Si no se está utilizando una ordenación, el elemento especificado se añade al final de la colección. Si se está utilizando una ordenación, el elemento se añade a la colección en función de la posición que ocupe en la ordenación actual.

Para más información sobre la inicialización y generación de objetos de transferencia, consulte, `DataSet.createItem()`.

## Ejemplo

```
myDataSet.addItem(myDataSet.createItem());
```

## Véase también

`DataSet.createItem()`

## DataSet.addSort()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.addSort(name, fieldList, sortOptions)
```

### Parámetros

*name* Cadena que especifica el nombre de la ordenación.

*fieldList* Matriz de cadenas que especifica los nombres de campo en los que basar la ordenación.

*sortOptions* Uno o varios de los valores enteros (constantes) siguientes, que indican las opciones que se utilizan en esta ordenación. Separe los distintos valores mediante el operador OR en modo bit (`|`). El valor o los valores deben ser alguno de los siguientes:

- `DataSetIterator.Ascending` Ordena los elementos en sentido ascendente. Ésta es la opción de ordenación predeterminada si no se especifica ninguna.
- `DataSetIterator.Descending` Ordena los elementos en sentido descendente en función de las propiedades de elemento especificadas.
- `DataSetIterator.Unique` Impide la ordenación si existen campos con valores idénticos.
- `DataSetIterator.CaseInsensitive` No distingue entre mayúsculas y minúsculas al comparar dos cadenas durante la ordenación. De forma predeterminada, la ordenación distingue entre mayúsculas y minúsculas cuando la propiedad en la que se basa es una cadena.

Se emite una excepción `DataSetError` cuando se especifica `DataSetIterator.Unique` como opción de ordenación y los datos por ordenar no son exclusivos, cuando ya se ha añadido el nombre de ordenación especificado o cuando la propiedad especificada en la matriz `fieldList` no existe en este juego de datos.

#### Valor devuelto

Ninguno.

#### Descripción

Método; crea un nuevo orden ascendente o descendente para el repetidor actual en función de las propiedades especificadas en el parámetro `fieldList`. La nueva ordenación se asigna automáticamente al repetidor actual tras su creación y se guarda en la colección de ordenaciones para poder recuperarla en otro momento.

#### Ejemplo

En el código siguiente se crea una nueva ordenación denominada "rank" que realiza una ordenación exclusiva en sentido descendente y que distingue entre mayúsculas y minúsculas en el campo "classRank" del objeto `DataSet`.

```
myDataSet.addSort("rank", ["classRank"], DataSetIterator.Descending |  
    DataSetIterator.Unique | DataSetIterator.CaseInsensitive);
```

#### Véase también

[DataSet.removeSort\(\)](#)

## DataSet.afterLoaded

#### Disponibilidad

Flash Player 7.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

```
on(afterLoaded) {  
    // introducir aquí el código propio  
}  
listenerObject = new Object();  
listenerObject.afterLoaded = function (eventObj) {  
    // introducir aquí el código propio  
}  
dataSet.addEventListener("afterLoaded", listenerObject)
```

#### Descripción

Evento; se difunde inmediatamente después de asignar la propiedad `DataSet.items`.

El objeto de evento (`eventObj`) contiene las propiedades siguientes:

**target** Objeto `DataSet` que generó el evento.

**type** Cadena "afterLoaded".

## Ejemplo

En este ejemplo, aparece un formulario denominado `contactForm` (que no se muestra) cuando ya se han asignado los elementos del objeto `DataSet` `contact_ds`.

```
contact_ds.addEventListener("afterLoaded", loadListener);
loadListener = new Object();
loadListener.afterLoaded = function (eventObj) {
    if(eventObj.target == "contact_ds") {
        contactForm.visible = true;
    }
}
```

## DataSet.applyUpdates()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.applyUpdates()
```

### Valor devuelto

Ninguno.

### Descripción

Método; emite una señal que indica que la propiedad `DataSet.deltaPacket` tiene un valor al que se puede acceder mediante la vinculación de datos o directamente utilizando ActionScript. Antes de llamar a este método, el valor de la propiedad `DataSet.deltaPacket` es `null`. Este método no tiene ninguna consecuencia si se han desactivado los eventos mediante el método `DataSet.disableEvents()`.

Al llamar a este método también se crea un ID de transacción para la propiedad `DataSet.deltaPacket` actual y se emite un evento `deltaPacketChanged`. Para más información, consulte `DataSet.deltaPacket`.

## Ejemplo

En el código siguiente se llama al método `applyUpdates()` en `myDataSet`.

```
myDataSet.applyUpdates();
```

### Véase también

[DataSet.deltaPacket](#)

## DataSet.calcFields

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.



## Sintaxis

```
on(calcFields) {  
    // introducir aquí el código propio  
}  
listenerObject = new Object();  
listenerObject.calcFields = function (eventObj) {  
    // introducir aquí el código propio  
}  
dataSet.addEventListener("calcFields", listenerObject)
```

## Descripción

Evento; se genera cuando deben determinarse los valores de los campos calculados del elemento actual de la colección. Un campo calculado es un campo en el que la propiedad `kind` tiene establecido el valor `Calculated` en la ficha Esquema del panel Inspector de componentes. El detector de eventos `calcFields` creado por el usuario debe realizar el cálculo necesario y establecer el valor correspondiente al campo calculado.

También se llama a este evento cuando se actualiza el valor de un campo no calculado (es decir, un campo cuya propiedad `kind` tenga establecido el valor `Data` en la ficha Esquema del panel Inspector de componentes).

Para más información sobre la propiedad `kind`, consulte “Tipos de esquema (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

**Atención:** no cambie ninguno de los valores de los campos no calculados en este evento, ya que ello provocaría un “bucle infinito”. Establezca únicamente los valores de los campos calculados dentro del evento `calcFields`.

## DataSet.changesPending()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.changesPending()
```

### Valor devuelto

Valor booleano.

### Descripción

Método; devuelve `true` si la colección, o cualquier elemento de la colección, tiene cambios pendientes de enviar en un objeto `DeltaPacket`; de lo contrario, devuelve `false`.

### Ejemplo

En el código siguiente se activa el botón Guardar cambios (que no se muestra) si la colección `DataSet`, o cualquier elemento de la colección, ha sufrido modificaciones no realizadas todavía en un objeto `DeltaPacket`.

```
if( data_ds.changesPending() ) {  
    saveChanges_btn.enabled = true;  
}
```

## DataSet.clear()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.clear()
```

### Valor devuelto

Ninguno.

### Descripción

Método; elimina los elementos de la vista actual de la colección. La consideración de un elemento como “apto para ser visto” depende de la configuración de filtro y de rango actual especificada en el repetidor actual. Por tanto, es posible que la llamada a este método no borre la totalidad de los elementos de la colección. Si se desea borrar la totalidad de los elementos de una colección, con independencia de la vista del repetidor actual, utilice [DataSet.removeAll\(\)](#).

Si el valor de [DataSet.logChanges](#) es `true` al invocar este método, las entradas eliminadas se añaden a [DataSet.deltaPacket](#) para todos los elementos que se encuentran dentro de la colección.

### Ejemplo

En este ejemplo se eliminan todos los elementos de la vista actual de la colección DataSet. La eliminación de los elementos se registrará porque la propiedad `logChanges` tiene establecido el valor `true`.

```
myDataSet.logChanges= true;  
myDataSet.clear();
```

### Véase también

[DataSet.deltaPacket](#), [DataSet.logChanges](#)

## DataSet.createItem()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.createItem([ itemData ])
```

### Parámetros

*itemData*    Datos asociados con el elemento. Este parámetro es opcional.

## Valor devuelto

El elemento que acaba de generarse.

## Descripción

Método; genera un elemento que no está asociado con la colección. Es posible especificar la clase de objeto creada con la propiedad `DataSet.itemClassName`. Si no se especifica un valor `DataSet.itemClassName` y se omite el parámetro *itemData*, se generará un objeto anónimo. Las propiedades de este objeto anónimo tienen los valores predeterminados en función del esquema actual que se haya especificado en `DataSet.schema`.

Al invocar este método se envía una notificación a los detectores del evento `DataSet.newItem`, que pueden entonces manipular el elemento antes de que el método lo devuelva. Los datos opcionales especificados del elemento se utilizan para inicializar la clase especificada con la propiedad `DataSet.itemClassName` o se utilizan como elemento si `DataSet.itemClassName` está vacío.

Se emite una excepción `DataSetError` cuando la clase especificada con la propiedad `DataSet.itemClassName` no puede cargarse.

## Ejemplo

```
contact.itemClassName = "Contact";
var itemData = new XML("<contact_info><name>John Smith</name><phone>555.555.4567</phone><zip><pre>94025</pre><post>0556</post></zip></contact_info>");
contact.addItem(contact.createItem(itemData));
```

## Véase también

`DataSet.itemClassName`, `DataSet.newItem`, `DataSet.schema`

## DataSet.currentItem

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

`dataSet.currentItem`

## Descripción

Propiedad (sólo lectura); devuelve el elemento actual de la colección `DataSet`, o `null` si está vacía la colección o la vista del repetidor actual de la colección.

Esta propiedad permite el acceso directo al elemento que se encuentra dentro de la colección. No se hace ningún seguimiento de los cambios realizados a causa del acceso directo a este objeto (en la propiedad `DataSet.deltaPacket`), ni tampoco de la configuración del esquema que se haya aplicado a las propiedades de este objeto.

### Ejemplo

En el ejemplo siguiente se muestra el valor de la propiedad `customerName` definido en el elemento actual del juego de datos denominado `customerData`.

```
trace(customerData.currentItem.customerName);
```

## DataSet.dataProvider

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.dataProvider
```

### Descripción

Propiedad; interfaz `DataProvider` de este juego de datos. Esta propiedad proporciona datos a los controles de interfaz de usuario, como los componentes `List` y `DataGrid`.

### Ejemplo

En el código siguiente se asigna la propiedad `dataProvider` de un objeto `DataSet` a la propiedad correspondiente de un componente `DataGrid`.

```
myGrid.dataProvider = myDataSet.dataProvider;
```

## DataSet.deltaPacket

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.deltaPacket
```

### Descripción

Propiedad; devuelve un objeto `DeltaPacket` que contiene todas las operaciones de cambio realizadas en la colección `dataSet` y en los elementos correspondientes. El valor de esta propiedad es `null` hasta que se llama a `DataSet.applyUpdates()` en `dataSet`.

Cuando se llama a `DataSet.applyUpdates()`, se asigna un ID de transacción al objeto `DeltaPacket`. Este ID de transacción se utiliza para identificar el objeto `DeltaPacket` en un proceso de actualización completo que va al servidor y vuelve al cliente. Cualquier asignación posterior que realice un objeto `DeltaPacket` a la propiedad `deltaPacket` con un ID de transacción coincidente tiene la función de respuesta del servidor a los cambios enviados con anterioridad. Se utiliza un objeto `DeltaPacket` con un ID coincidente para actualizar la colección e informar de los errores especificados dentro del paquete.

Los errores o mensajes del servidor se notifican a los detectores del evento `DataSet.resolveDelta`. Observe que se omite la configuración de `DataSet.logChanges` cuando se asigna un objeto `DeltaPacket` a `DataSet.deltaPacket` con un ID coincidente. Si un objeto `DeltaPacket` no tiene un ID de transacción coincidente se actualiza la colección, como si la interfaz API de `DataSet` se utilizara directamente. Es posible que se creen entonces más entradas delta, lo cual dependerá de la configuración actual de `DataSet.logChanges` de `dataSet` y del objeto `DeltaPacket`.

Se emitirá una excepción `DataSetError` si se asigna un objeto `DeltaPacket` con un ID de transacción coincidente y si no se encuentra uno de los elementos del objeto `DeltaPacket` recién asignado en el objeto `DeltaPacket`.

#### Véase también

`DataSet.applyUpdates()`, `DataSet.logChanges`, `DataSet.resolveDelta`

## DataSet.deltaPacketChanged

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(deltaPacketChanged) {  
    // introducir aquí el código propio  
}  
listenerObject = new Object();  
listenerObject.deltaPacketChanged = function (eventObj) {  
    // introducir aquí el código propio  
}  
dataSet.addEventListener("deltaPacketChanged", listenerObject)
```

### Descripción

Evento; se difunde cuando la propiedad `deltaPacket` del objeto `DataSet` especificado ha sufrido cambios y está preparada para utilizarse.

#### Véase también

`DataSet.deltaPacket`

## DataSet.disableEvents()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.disableEvents()
```

### Valor devuelto

Ninguno.

### Descripción

Método; desactiva eventos para el objeto DataSet. Mientras los eventos están desactivados no se actualiza ningún control de interfaz de usuario (como el componente DataGrid) cuando se realizan cambios en los elementos de la colección o se desplaza el objeto DataSet hasta otro elemento de la colección.

Para volver a activar los eventos, es necesario llamar a `DataSet.enableEvents()`. El método `disableEvents()` puede llamarse varias veces y, para volver a activar la distribución de eventos, es necesario llamar al método `enableEvents()` el mismo número de veces.

### Ejemplo

En este ejemplo los eventos se desactivan antes de que los elementos de la colección sufran cambios, de forma que el rendimiento del sistema no se vea perjudicado porque el objeto DataSet intente actualizar los controles.

```
// Desactivar eventos para el juego de datos
myDataSet.disableEvents();
myDataSet.last();
while(myDataSet.hasPrevious()) {
    var price = myDataSet.price;
    price = price * 0.5; // Todo rebajado en un 50%
    myDataSet.price = price;
    myDataSet.previous();
}
//Indicar al juego de datos que ahora hay que actualizar los controles
myDataSet.enableEvents();
```

### Véase también

`DataSet.enableEvents()`

## DataSet.enableEvents()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.enableEvents()
```

### Valor devuelto

Ninguno.

### Descripción

Método; reactiva los eventos de los objetos DataSet después de que dichos eventos se hayan desactivado mediante una llamada a `DataSet.disableEvents()`. Para reactivar eventos para el objeto DataSet, debe llamarse al método `enableEvents()` un número de veces igual o mayor que el número de veces que se haya llamado a `disableEvents()`.

## Ejemplo

En este ejemplo los eventos se desactivan antes de que los elementos de la colección sufran cambios, de forma que el rendimiento del sistema no se vea perjudicado porque el objeto `DataSet` intente actualizar los controles.

```
// Desactivar eventos para el juego de datos
myDataSet.disableEvents();
myDataSet.last();
while(myDataSet.hasPrevious()) {
    var price = myDataSet.price;
    price = price * 0.5; // Todo rebajado en un 50%
    myDataSet.price = price;
    myDataSet.previous();
}
//Indicar al juego de datos que ahora hay que actualizar los controles
myDataSet.enableEvents();
```

## Véase también

[DataSet.disableEvents\(\)](#)

## DataSet.filtered

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

*dataSet.filtered*

### Descripción

Propiedad; valor booleano que indica si los datos del repetidor actual están filtrados o no. Si se establece en `true`, se llama a la función de filtro especificada por [DataSet.filterFunc](#) para cada elemento de la colección.

## Ejemplo

En el ejemplo siguiente se ha activado el filtrado en el objeto `DataSet` denominado `employee_ds`. Supongamos que cada registro de la colección `DataSet` contiene un campo denominado `empType`. La función de filtro siguiente devuelve `true` si el campo `empType` del elemento actual se ha establecido en "management"; de lo contrario, devuelve `false`.

```
employee_ds.filtered = true;
employee_ds.filterFunc = function(item:Object) {
    // filtrar a los empleados que ocupan cargos directivos...
    return(item.empType != "management");
}
```

## Véase también

[DataSet.filterFunc](#)

## DataSet.filterFunc

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.filterFunc = function(item:Object) {  
    // devolver true|false;  
};
```

### Descripción

Propiedad; especifica una función que determina los elementos que se incluyen en la vista actual de la colección. Cuando `DataSet.filtered` se establece en `true`, se llama a la función asignada a esta propiedad para cada objeto de transferencia de la colección. Para cada elemento que se pase a la función, deberá devolver el valor `true` si el elemento debe incluirse en la vista actual, o `false` si no debe incluirse.

### Ejemplo

En el ejemplo siguiente se ha activado el filtrado en el objeto `DataSet` denominado `employee_ds`. La función de filtro especificada devuelve `true` si el campo `empType` de cada elemento se ha establecido en "management"; de lo contrario, devuelve `false`.

```
employee_ds.filtered = true;  
employee_ds.filterFunc = function(item:Object) {  
    // filtrar a los empleados que ocupan cargos directivos...  
    return(item.empType != "management");  
}
```

### Véase también

[DataSet.filtered](#)

## DataSet.find()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.find(searchValues)
```

### Parámetros

*searchValues* Matriz que contiene uno o varios valores de campo que se encontrarán en la clasificación actual.

### Valor devuelto

Devuelve `true` si se encuentran los valores y `false` si no se encuentran.



## Descripción

Método; busca en la vista actual de la colección un elemento que tenga los valores de campo especificados por *searchValues*. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada. Si se encuentra algún elemento éste se convierte en el elemento actual del objeto DataSet.

Los valores especificados por *searchValues* deben estar en el mismo orden que la lista de campos especificada por la clasificación actual (véase el ejemplo siguiente).

Si la clasificación actual no es exclusiva, el objeto de transferencia encontrado no es determinante. Si desea encontrar el primero o el último objeto de transferencia que aparezca en una clasificación no exclusiva, utilice `DataSet.findFirst()` o `DataSet.findLast()`.

La conversión de los datos especificados depende del tipo del campo subyacente y del que se haya especificado en la matriz. Por ejemplo, si se especifica ["05-02-02"] como valor de búsqueda, el campo de fecha subyacente se utiliza para convertir el valor utilizando el método

`DataType.setAsString()` de la fecha. Si se especifica `[new Date().getTime()]`, se utiliza el método `DataType.setAsNumber()` de la fecha.

## Ejemplo

En este ejemplo se busca en la colección actual un elemento cuyos campos *name* e *id* contengan los valores "Bobby" y 105, respectivamente. Si se encuentra, se utilizará el método `DataSet.getItemId()` para obtener un identificador exclusivo del elemento de la colección y el método `DataSet.locateById()` para colocar el repetidor actual en dicho elemento.

```
var studentID:String = null;
studentData.addSort("id", ["name","id"]);
// Buscar el objeto de transferencia identificado mediante "Bobby" y 105.
// Obsérvese que el orden de los campos de búsqueda coincide con
// el de los campos especificados en el método addSort().
if(studentData.find(["Bobby", 105])) {
    studentID = studentData.getItemId();
}
// Ahora utilice el método locateById() para colocar el
// repetidor actual en el elemento de la colección cuyo ID coincida con
// studentID.
if(studentID != null) {
    studentData.locateById(studentID);
}
```

## Véase también

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

## DataSet.findFirst()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

`dataSet.findFirst(searchValues)`

## Parámetros

*searchValues* Matriz que contiene uno o varios valores de campo que se encontrarán en la clasificación actual.

## Valor devuelto

Devuelve `true` si se encuentran los elementos y `false` si no se encuentran.

## Descripción

Método; busca en la vista actual de la colección el primer elemento que tenga los valores de campo especificados por *searchValues*. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

Los valores especificados por *searchValues* deben estar en el mismo orden que la lista de campos especificada por la clasificación actual (véase el ejemplo siguiente).

La conversión de los datos especificados depende del tipo del campo subyacente y del que se haya especificado en la matriz. Por ejemplo, si el valor de búsqueda especificado es ["05-02-02"], el campo de fecha subyacente se utiliza para convertir el valor utilizando el método `setAsString()` de la fecha. Si el valor especificado es `[new Date().getTime()]`, se utiliza el método `setAsNumber()` de la fecha.

## Ejemplo

En este ejemplo se busca en la colección actual el primer elemento cuyos campos `name` y `age` contengan los valores "Bobby" y "13". Si se encuentra, se utilizará el método `DataSet.getItemId()` para obtener un identificador exclusivo del elemento de la colección y el método `DataSet.locateById()` para colocar el repetidor actual en dicho elemento.

```
var studentID:String = null;
studentData.addSort("nameAndAge", ["name", "age"]);
// Buscar el primer objeto de transferencia que tenga los valores
// especificados.
// Obsérvese que el orden de los campos de búsqueda coincide con
// el de los campos especificados en el método addSort().
if(studentData.findFirst(["Bobby", "13"])) {
    studentID = studentData.getItemId();
}
// Ahora utilice el método locateById() para colocar el
// repetidor actual en el elemento de la colección cuyo ID coincida con
// studentID
if(studentID != null) {
    studentData.locateById(studentID);
}
```

## Véase también

`DataSet.applyUpdates()`, `DataSet.getItemId()`, `DataSet.locateById()`

## DataSet.findLast()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.findLast(searchValues)
```

### Parámetros

*searchValues* Matriz que contiene uno o varios valores de campo que se encontrarán en la clasificación actual.

### Valor devuelto

Devuelve `true` si se encuentran los elementos y `false` si no se encuentran.

### Descripción

Método; busca en la vista actual de la colección el último elemento que tenga los valores de campo especificados por *searchValues*. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

Los valores especificados por *searchValues* deben estar en el mismo orden que la lista de campos especificada por la clasificación actual (véase el ejemplo siguiente).

La conversión de los datos especificados depende del tipo del campo subyacente y del que se haya especificado en la matriz. Por ejemplo, si el valor de búsqueda especificado es ["05-02-02"], el campo de fecha subyacente se utiliza para convertir el valor utilizando el método `setAsString()` de la fecha. Si el valor especificado es `[new Date().getTime()]`, se utiliza el método `setAsNumber()` de la fecha.

### Ejemplo

En este ejemplo se busca en la colección actual el último elemento cuyos campos `name` y `age` contengan los valores "Bobby" y "13". Si se encuentra, se utilizará el método `DataSet.getItemId()` para obtener un identificador exclusivo del elemento de la colección y el método `DataSet.locateById()` para colocar el repetidor actual en dicho elemento.

```
var studentID:String = null;
studentData.addSort("nameAndAge", ["name", "age"]);
// Buscar el último objeto de transferencia que tenga los valores
// especificados.
// Obsérvese que el orden de los campos de búsqueda coincide con
// el de los campos especificados en el método addSort().
if(studentData.findLast(["Bobby", "13"])) {
    studentID = studentData.getItemId();
}
// Ahora utilice el método locateById() para colocar el
// repetidor actual en el elemento de la colección cuyo ID coincida con
// studentID.
if(studentID != null) {
    studentData.locateById(studentID);
}
```

### Véase también

[DataSet.applyUpdates\(\)](#), [DataSet.getItemId\(\)](#), [DataSet.locateById\(\)](#)

## DataSet.first()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.first()
```

### Valor devuelto

Ninguno.

### Descripción

Método; convierte el primer elemento de la vista actual de la colección en el elemento actual. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

### Ejemplo

Con el siguiente código se coloca el juego de datos `userData` en el primer elemento de su colección y, a continuación, se muestra el valor de la propiedad `price` que dicho elemento contiene mediante la propiedad [DataSet.currentItem](#).

```
inventoryData.first();  
trace("The price of the first item is:" + inventoryData.currentItem.price);
```

### Véase también

[DataSet.last\(\)](#)

## DataSet.getItemId()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.getItemId([index])
```

### Parámetros

*index* Número que especifica el elemento de la vista actual de elementos para obtener el ID. Este parámetro es opcional.

### Valor devuelto

Una cadena.

## Descripción

Método; devuelve el identificador del elemento actual de la colección o el del elemento especificado por *index*. Dicho identificador es exclusivo sólo dentro de esta colección determinada y `DataSet.addItem()` los asigna automáticamente.

## Ejemplo

El código siguiente obtiene el ID exclusivo correspondiente al elemento actual de la colección y, a continuación, lo muestra en el panel Salida.

```
var itemNo:String = myDataSet.getItemId();
trace("Employee id("+ itemNo+ ")");
```

## Véase también

`DataSet.addItem()`

## DataSet.getIterator()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.getIterator()
```

### Valor devuelto

Un objeto `ValueListIterator`.

## Descripción

Método; devuelve un nuevo repetidor para esta colección. Dicho repetidor es una copia del repetidor actual que esté utilizándose y conserva la misma posición dentro de la colección. Este método está diseñado especialmente para usuarios avanzados que necesiten acceder a varias vistas de una misma colección de forma simultánea.

## Ejemplo

```
myIterator:ValueListIterator = myDataSet.getIterator();
myIterator.sortOn(["name"]);
myIterator.find({name:"John Smith"}).phone = "555-1212";
```

## DataSet.hasNext()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.hasNext()
```

### Valor devuelto

Valor booleano.

### Descripción

Método; devuelve `false` si el repetidor actual se encuentra al final de la vista correspondiente de la colección y `true` en caso contrario.

### Ejemplo

Este ejemplo se repite en todos los elementos de la vista actual de la colección (empezando por el principio) y realiza un cálculo en la propiedad `price` de cada elemento.

```
myDataSet.first();
while ( myDataSet.hasNext() ) {
    var price = myDataSet.currentItem.price;
    price = price * 0.5; // Todo rebajado en un 50%
    myDataSet.currentItem.price = price;
    myDataSet.next();
}
```

### Véase también

[DataSet.currentItem](#), [DataSet.first\(\)](#), [DataSet.next\(\)](#)

## DataSet.hasPrevious()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.hasPrevious()
```

### Valor devuelto

Valor booleano.

### Descripción

Método; devuelve `false` si el repetidor actual se encuentra al principio de la vista correspondiente de la colección y `true` en caso contrario.

### Ejemplo

Este ejemplo se repite en todos los elementos de la vista actual de la colección (empezando por el último elemento) y realiza un cálculo en la propiedad `price` de cada elemento.

```
myDataSet.last();
while(myDataSet.hasPrevious()) {
    var price = myDataSet.currentItem.price;
    price = price * 0.5; // Todo rebajado en un 50%
    myDataSet.currentItem.price = price;
    myDataSet.previous();
}
```

## Véase también

[DataSet.currentItem](#), [DataSet.skip\(\)](#), [DataSet.previous\(\)](#)

## DataSet.hasSort()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.hasSort(sortName)
```

### Parámetros

*sortName* Cadena que contiene el nombre de una clasificación creada con [DataSet.addSort\(\)](#)

### Valor devuelto

Valor booleano.

### Descripción

Método; devuelve true si la clasificación especificada por *sortName* existe, y false en caso contrario.

### Ejemplo

El código siguiente comprueba si existe una clasificación denominada “customerSort”. Si la clasificación ya existe, se convierte en la clasificación actual mediante el método [DataSet.useSort\(\)](#). Si no existe una clasificación con ese nombre, se creará una mediante el método [DataSet.addSort\(\)](#).

```
if(myDataSet.hasSort("customerSort"))
    myDataSet.useSort("customerSort");
} else {
    myDataSet.addSort("customerSort", ["customer"],
        DataSetIterator.Descending);
}
```

## Véase también

[DataSet.applyUpdates\(\)](#), [DataSet.useSort\(\)](#)

## DataSet.isEmpty()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.isEmpty()
```

### Valor devuelto

Valor booleano.

### Descripción

Método; devuelve `true` si el objeto `DataSet` especificado no contiene ningún elemento (es decir, si `dataSet.length == 0`).

### Ejemplo

La secuencia siguiente desactiva el botón Eliminar registro (de manera que no se muestra) si el objeto `DataSet` correspondiente está vacío.

```
if(userData.isEmpty()){
    delete_btn.enabled = false;
}
```

### Véase también

[DataSet.length](#)

## DataSet.items

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDataSet.items*

### Descripción

Propiedad; matriz de elementos administrados por *myDataSet*.

### Ejemplo

En este ejemplo se asigna una matriz de objetos a la propiedad `items` de un objeto `DataSet`.

```
var recData = [{id:0, firstName:"Mick", lastName:"Jones"},
               {id:1, firstName:"Joe", lastName:"Strummer"},
               {id:2, firstName:"Paul", lastName:"Simonon"}];
myDataSet.items = recData;
```

## DataSet.itemClassName

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

*dataSet.itemClassName*



## Descripción

Propiedad; cadena que indica el nombre de la clase que se creará cuando se añadan elementos a la colección. La clase que se especifique debe implementar la interfaz `TransferObject`, tal como se muestra a continuación.

```
interface mx.data.to.TransferObject {
    function clone():Object;
    function getPropertyData():Object;
    function setPropertyData(propData:Object):Void;
}
```

También es posible establecer la propiedad en el inspector de propiedades.

Para que la clase especificada esté disponible en tiempo de ejecución, también es necesario hacer una referencia completa a la clase en algún lugar del código del archivo SWF, como se muestra en el siguiente fragmento de código:

```
var myItem:my.package.myItem;
```

Se emite la excepción `DataSetError` si se intenta modificar el valor de esta propiedad después de que se haya cargado la matriz `DataSet.items`.

Para más información sobre la interfaz `TransferObject`, consulte [“Interfaz TransferObject” en la página 566](#).

## DataSet.iteratorScrolled

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(iteratorScrolled) {
    // introducir aquí el código propio
}
listenerObject = new Object();
listenerObject.iteratorScrolled = function (eventObj) {
    // introducir aquí el código propio
}
dataSet.addEventListener("iteratorScrolled", listenerObject)
```

### Descripción

Evento; se genera inmediatamente después de que el repetidor actual se haya desplazado hasta un nuevo elemento de la colección.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

**target** Objeto `DataSet` que generó el evento.

**type** Cadena "iteratorScrolled".

**scrolled** Número que especifica la cantidad de elementos por los que se ha desplazado un repetidor; un valor positivo indica que el repetidor se ha desplazado hacia delante en la colección, mientras que un valor negativo indica que el desplazamiento se ha realizado hacia atrás.

## Ejemplo

En este ejemplo, la barra de estado de una aplicación (que no se muestra) se actualiza cuando cambia la posición del repetidor actual.

```
on(iteratorScrolled) {  
    var dataSet:mx.data.components.DataSet = eventObj.target;  
    var statusBarText = dataSet.fullname+" Acct #:"  
    "+dataSet.getField("acctnum").getAsString();  
    setStatusBar(statusBarText);  
}
```

## DataSet.last()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.last()
```

### Valor devuelto

Ninguno.

### Descripción

Método; convierte el último elemento de la vista actual de la colección en el elemento actual.

## Ejemplo

El siguiente código asociado con un componente Button se dirige al último elemento de la colección DataSet.

```
function goLast(eventObj:obj) {  
    inventoryData.last();  
}  
goLast_btn.addEventListener("click", goLast);
```

### Véase también

[DataSet.first\(\)](#)

## DataSet.length

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.length
```

## Descripción

Propiedad (sólo lectura); especifica el número de elementos de la vista actual de la colección. El número de elementos que puede verse depende de la configuración actual del filtro y del rango.

## Ejemplo

En el siguiente ejemplo se avisa a los usuarios si no han creado el número suficiente de entradas en el juego de datos sí, por ejemplo, han utilizado el componente editable DataGrid.

```
if(myDataSet.length < MIN_REQUIRED) {  
    alert("You need at least "+MIN_REQUIRED);  
}
```

## DataSet.loadFromSharedObj()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.loadFromSharedObj(objName, [localPath])
```

### Parámetros

*objName* Cadena que especifica el nombre del objeto compartido que debe recuperarse. El nombre puede incluir barras diagonales (/) (por ejemplo, “direcciones/trabajo”). No se permite el uso de espacios ni de los siguientes caracteres en el nombre que se especifique:

~ % & \ ; : " ' , < > ? #

*localPath* Parámetro de cadena opcional que especifica la ruta completa o parcial al archivo SWF que ha creado el objeto compartido. Esta cadena se utiliza para determinar la ubicación del equipo del usuario en la que se encuentra almacenado el objeto. El valor predeterminado es la ruta completa del archivo SWF.

### Valor devuelto

Ninguno.

## Descripción

Método; carga todos los datos necesarios para restaurar esta colección DataSet a partir de un objeto compartido. Para guardar una colección DataSet en un objeto compartido, utilice [DataSet.saveToSharedObj\(\)](#). El método `DataSet.loadFromSharedObject()` sobrescribe los datos o cambios pendientes que existan dentro de esta colección DataSet. Tenga en cuenta que el nombre de esta colección DataSet se utiliza para identificar los datos dentro de un objeto compartido determinado.

Este método emite una excepción `DataSetError` si no se encuentra el objeto compartido especificado o si surge un problema al recuperar los datos de dicho objeto.

## Ejemplo

En este ejemplo se intenta cargar un objeto compartido denominado `webapp/customerInfo` asociado con el juego de datos denominado `myDataSet`. La llamada al método se realiza dentro de un bloque de código `try...catch`.

```
try {
    myDataSet.loadFromSharedObj("webapp/customerInfo");
}
catch(e:DataSetError) {
    trace("Unable to load shared object.");
}
```

## Véase también

[DataSet.saveToSharedObj\(\)](#)

## DataSet.locateById()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.locateById(id)
```

### Parámetros

*id*    Identificador de cadena del elemento de la colección que hay que buscar.

### Valor devuelto

Valor booleano.

### Descripción

Método; coloca el repetidor actual en el elemento de la colección cuyo ID coincida con *id*. El método devuelve `true` si el ID especificado puede correlacionarse con un elemento de la colección, y `false` en caso contrario.

## Ejemplo

En este ejemplo se utiliza `DataSet.find()` para buscar en la colección actual un elemento cuyos campos `name` e `id` contengan los valores "Bobby" y 105, respectivamente. Si se encuentra, se utilizará el método `DataSet.getItemId()` para obtener un identificador exclusivo de dicho elemento y el método `DataSet.locateById()` para colocar el repetidor actual en dicho elemento.

```
var studentID:String = null;
studentData.addSort("id", ["name","id"]);
if(studentData.find(["Bobby", 105])) {
    studentID = studentData.getItemId();
    studentData.locateById(studentID);
}
```

## Véase también

[DataSet.applyUpdates\(\)](#), [DataSet.find\(\)](#), [DataSet.getItemId\(\)](#)

## DataSet.logChanges

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

*dataSet.logChanges*

### Descripción

Propiedad; valor booleano que especifica si deben (*true*) o no deben (*false*) registrarse en [DataSet.deltaPacket](#) los cambios realizados en el juego de datos o en sus elementos.

Cuando esta propiedad se establece en *true* se registran las operaciones realizadas en los niveles de colección y elemento. Los cambios realizados en el nivel de colección incluyen las operaciones realizadas para añadir o eliminar elementos de la colección. Los cambios realizados en el nivel de elementos incluyen los cambios de propiedad realizados en los elementos y las llamadas a métodos realizadas en los elementos mediante el componente DataSet.

### Ejemplo

En el ejemplo siguiente se desactiva el registro para el objeto DataSet denominado *userData*.

```
userData.logChanges = false;
```

## Véase también

[DataSet.deltaPacket](#)

## DataSet.modelChanged

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Descripción

```
on(modelChanged) {  
    // introducir aquí el código propio  
}  
listenerObject = new Object();  
listenerObject.modelChanged = function (eventObj) {  
    // introducir aquí el código propio  
}  
dataSet.addEventListener("modelChanged", listenerObject)
```

## Descripción

Evento; se difunde cuando se produce algún cambio en la colección; por ejemplo, cuando se añaden elementos a la colección o se eliminan elementos de ella, cuando el valor de la propiedad de un elemento varía o cuando se filtra u ordena la colección.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

**target** Objeto DataSet que generó el evento.

**type** Cadena "iteratorScrolled".

**firstItem** Índice (número) del primer elemento de la colección afectado por el cambio.

**lastItem** Índice (número) del último elemento de la colección afectado por el cambio (es igual a **firstItem** si sólo se ve afectado un elemento).

**fieldName** Cadena que contiene el nombre del campo afectado. Esta propiedad no está definida (es decir, tiene un valor de *undefined*) salvo si el cambio se ha efectuado en una propiedad del objeto DataSet.

**eventName** Cadena que describe el cambio realizado. Puede tratarse de cualquiera de los valores siguientes:

| Valor de la cadena | Descripción                                                                                                       |
|--------------------|-------------------------------------------------------------------------------------------------------------------|
| "addItem"          | Se ha añadido una serie de elementos.                                                                             |
| "filterModel"      | El modelo se ha filtrado y es necesario actualizar la vista (restablecer la posición de desplazamiento).          |
| "removeItems"      | Se ha eliminado una serie de elementos.                                                                           |
| "schemaLoaded"     | Se ha declarado la definición de los campos del proveedor de datos.                                               |
| "sort"             | Se han ordenado los datos.                                                                                        |
| "updateAll"        | Es necesario actualizar toda la vista, excluida la posición de desplazamiento.                                    |
| "updateColumn"     | Es necesario actualizar la definición de la totalidad de un campo que se encuentra dentro del proveedor de datos. |
| "updateField"      | Un campo de un elemento ha cambiado y debe actualizarse.                                                          |
| "updateItems"      | Es necesario actualizar una serie de elementos.                                                                   |

## Ejemplo

En este ejemplo, el botón que sirve para eliminar elementos está desactivado si se han eliminado los elementos de la colección y si el objeto DataSet de destino no tiene más elementos.

```
on(modelChanged) {  
    delete_btn.enabled = ((eventObj.eventName == "removeItems") &&  
        (eventObj.target.isEmpty()));  
}
```

## Véase también

[DataSet.isEmpty\(\)](#)

## DataSet.newItem

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(newItem) {  
    // introducir aquí el código propio  
}  
listenerObject = new Object();  
listenerObject.newItem = function (eventObj) {  
    // introducir aquí el código propio  
}  
dataSet.addEventListener("newItem", listenerObject)
```

### Descripción

Evento; se difunde cuando se construye un nuevo objeto de transferencia mediante `DataSet.createItem()`. Un detector de este evento puede realizar modificaciones sobre el elemento antes de añadirlo a la colección.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

target    Objeto DataSet que generó el evento.

type    Cadena "iteratorScrolled".

item    Referencia al elemento creado.

### Ejemplo

En este ejemplo se realizan modificaciones en un elemento recién creado antes de añadirlo a la colección.

```
function newItemEvent(evt:Object):Void {  
    var employee:Object = evt.item;  
    employee.name = "newGuy";  
    // los datos de la propiedad son XML  
    employee.zip =  
        employee.getPropertyData().firstChild.childNodes[1].attributes.zip;  
}  
employees_ds.addEventListener("newItem", newItemEvent);
```

## DataSet.next()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.next()
```

**Valor devuelto**

Ninguno.

**Descripción**

Método; convierte el siguiente elemento de la vista actual de la colección en el elemento actual. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

**Ejemplo**

En este ejemplo se reproducen indefinidamente todos los elementos de un objeto DataSet, empezando por el primero, y se realiza un cálculo en un campo de cada elemento.

```
myDataSet.first();
while ( myDataSet.hasNext() ) {
    var price = myDataSet.price;
    price = price * 0.5; // Todo rebajado en un 50%
    myDataSet.price = price;
    myDataSet.next();
}
```

**Véase también**

[DataSet.first\(\)](#), [DataSet.hasNext\(\)](#)

**DataSet.previous()****Disponibilidad**

Flash Player 7.

**Edición**

Flash MX Professional 2004.

**Sintaxis**

```
dataSet.previous()
```

**Valor devuelto**

Ninguno.

**Descripción**

Método; convierte el elemento anterior de la vista actual de la colección en el elemento actual. Los elementos que aparecen en la vista actual dependen de la configuración de filtro y de rango especificada.

En este ejemplo se repiten indefinidamente todos los elementos de la vista actual de la colección, empezando por el último, y se realiza un cálculo en un campo de cada elemento.

```
myDataSet.last();
while(myDataSet.hasPrevious()) {
    var price = myDataSet.price;
    price = price * 0.5; // Todo rebajado en un 50%
    myDataSet.price = price;
    myDataSet.previous();
}
```



**Véase también**

`DataSet.first()`, `DataSet.hasNext()`

## **DataSet.properties**

### **Disponibilidad**

Flash Player 7.

### **Edición**

Flash MX Professional 2004.

### **Sintaxis**

`dataSet.properties`

### **Descripción**

Propiedad (sólo lectura); devuelve un objeto que contiene todas las propiedades expuestas (campos) correspondientes a cualquier objeto de transferencia que se encuentre dentro de esta colección.

### **Ejemplo**

En este ejemplo se muestran todos los nombres de las propiedades del objeto `DataSet` denominado `myDataSet`.

```
for(var i in myDataSet.properties) {  
    trace("field '"+i+ "' has value "+ myDataSet.properties[i]);  
}
```

## **DataSet.readOnly**

### **Disponibilidad**

Flash Player 7.

### **Edición**

Flash MX Professional 2004.

### **Sintaxis**

`dataSet.readOnly`

### **Descripción**

Propiedad; valor booleano que especifica si la colección puede modificarse (`false`) o es de sólo lectura (`true`). Si la propiedad se establece en `true` no podrán realizarse actualizaciones de la colección.

También es posible establecer la propiedad en el inspector de propiedades.

### Ejemplo

En el ejemplo siguiente se convierte el objeto DataSet denominado `myDataSet` en un objeto de sólo lectura y, a continuación, se intenta modificar el valor de una propiedad que pertenece al elemento actual de la colección. Con esta acción se emitirá una excepción.

```
myDataSet.readOnly = true;  
//Con esta acción se emitirá una excepción  
myDataSet.currentItem.price = 15;
```

### Véase también

[DataSet.currentItem](#)

## DataSet.removeAll()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.removeAll()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; elimina todos los elementos de la colección DataSet.

### Ejemplo

En este ejemplo se eliminan todos los elementos de la colección DataSet `contact_ds`:

```
contact_ds.removeAll();
```

## DataSet.removeItem

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

## Sintaxis

```
on(removeItem) {  
    // introducir aquí el código propio  
}  
listenerObject = new Object();  
listenerObject.removeItem = function (eventObj) {  
    // introducir aquí el código propio  
}  
dataSet.addEventListener("removeItem", listenerObject)
```

## Descripción

Evento; generado justo antes de eliminar un nuevo elemento de esta colección.

Si se especifica el valor `false` en la propiedad `result` del objeto en el que se ha producido el evento, se cancelará la operación de eliminación; en cambio si se establece el valor `true`, podrá realizarse dicha operación.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

`target` Objeto `DataSet` que generó el evento.

`type` Cadena "removeItem".

`item` Referencia al elemento de la colección que se va a eliminar.

`result` Valor booleano que especifica si el elemento debe o no eliminarse. De forma predeterminada, este valor es `true`.

## Ejemplo

En este ejemplo, el controlador de eventos `on(removeItem)` cancela la eliminación del nuevo elemento si una función definida por el usuario denominada `userHasAdminPrivs()` devuelve `false`; de lo contrario se permite llevar a cabo la eliminación.

```
on(removeItem) {  
    if(globalObj.userHasAdminPrivs()) {  
        // Permitir la eliminación del elemento.  
        eventObj.result = true;  
    } else {  
        // No permitir la eliminación del elemento; el usuario no tiene  
        privilegios de administrador.  
        eventObj.result = false;  
    }  
}
```

## Véase también

[DataSet.addItem](#)

## DataSet.removeItem()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.removeItem([ item ])
```

### Parámetros

*item* Elemento que debe eliminarse. Este parámetro es opcional.

### Valor devuelto

Valor booleano. Devuelve `true` si el elemento se eliminó con éxito, y `false` en caso contrario.

### Descripción

Método; elimina el elemento especificado de la colección, o bien elimina el elemento actual si se omite el parámetro *item*. Esta operación se registra en [DataSet.deltaPacket](#) si el valor de [DataSet.logChanges](#) es `true`.

### Ejemplo

El siguiente código, que se asocia con una instancia del componente Button, elimina el elemento actual del objeto DataSet denominado `userData`, que reside en la misma línea de tiempo que la instancia Button.

```
on(click){  
    _parent.userData.removeItem();  
}
```

### Véase también

[DataSet.deltaPacket](#), [DataSet.logChanges](#)

## DataSet.removeRange()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.removeRange()
```

### Valor devuelto

Ninguno.

## Descripción

Método; elimina la configuración especificada del punto final actual mediante `DataSet.setRange()` para el repetidor actual.

## Ejemplo

```
myDataSet.addSort("name_id", ["name", "id"]);
myDataSet.setRange(["Bobby", 105],["Cathy", 110]);
while ( myDataSet.hasNext() ) {
    myDataSet.gradeLevel = "5"; // cambiar todas las calificaciones de este rango
    myDataSet.next();
}
myDataSet.removeRange();
myDataSet.removeSort("name_id");
```

## Véase también

`DataSet.applyUpdates()`, `DataSet.hasNext()`, `DataSet.next()`, `DataSet.removeSort()`, `DataSet.setRange()`

## DataSet.removeSort()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.removeSort(sortName)
```

### Parámetros

*sortName*    Cadena que especifica el nombre de la clasificación que va a eliminarse.

### Valor devuelto

Ninguno.

### Descripción

Método; elimina la clasificación especificada de este objeto DataSet si dicha clasificación existe. Si la clasificación especificada no existe, el método emitirá una excepción `DataSetError`.

## Ejemplo

```
myDataSet.addSort("name_id", ["name", "id"]);
myDataSet.setRange(["Bobby", 105],["Cathy", 110]);
while ( myDataSet.hasNext() ) {
    myDataSet.gradeLevel = "5"; // cambiar todas las calificaciones de este rango
    myDataSet.next();
}
myDataSet.removeRange();
myDataSet.removeSort("name_id");
```

## Véase también

`DataSet.applyUpdates()`, `DataSet.hasNext()`, `DataSet.next()`, `DataSet.removeRange()`, `DataSet.setRange()`

## DataSet.resolveDelta

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(resolveDelta) {  
    // introducir aquí el código propio  
}  
listenerObject = new Object();  
listenerObject.resolveDelta = function (eventObj) {  
    // introducir aquí el código propio  
}  
dataSet.addEventListener("resolveDelta", listenerObject)
```

### Descripción

Evento; se difunde cuando se asigna un objeto DeltaPacket a [DataSet.deltaPacket](#), cuyo ID de transacción coincide con el de un objeto DeltaPacket, previamente recuperado a partir del objeto DataSet, y que tiene mensajes asociados con cualquiera de los objetos Delta o DeltaItem que contiene el objeto DeltaPacket.

Este evento permite enmendar los errores devueltos desde el servidor al intentar aplicar los cambios enviados anteriormente. Normalmente este evento se utiliza para mostrar un “cuadro de diálogo de soluciones” con los valores incorrectos, de forma que el usuario pueda realizar las modificaciones pertinentes en los datos y los pueda volver a enviar.

El objeto de evento (*eventObj*) contiene las propiedades siguientes:

target    Objeto DataSet que generó el evento.

type    Cadena "resolveDelta".

data    Matriz de objetos Delta y objetos DeltaItem asociados con mensajes de longitud distinta de cero.

## Ejemplo

En este ejemplo se muestra un formulario denominado `reconcileForm` (que no se muestra) y se llama a un método de ese objeto `form` (`setReconcileData()`) que permite al usuario corregir los valores devueltos como incorrectos por el servidor.

```
myDataSet.addEventListener("resolveDelta", resolveDelta);
function resolveDelta(eventObj:Object) {
    reconcileForm.visible = true;
    reconcileForm.setReconcileData(eventObj.data);
}
// en el código de reconcileForm
function setReconcileData(data:Array):Void {
    var di:DeltaItem;
    var ops:Array = ["property", "method"];
    var cl:Array;
    // cambiar lista
    var msg:String;
    for (var i = 0; i<data.length; i++) {
        cl = data[i].getChangeList();
        for (var j = 0; j<cl.length; j++) {
            di = cl[j];
            msg = di.getMessage();
            if (msg.length>0) {
                trace("The following problem occurred '"+msg+"' while performing a
'"+ops[di.kind]+"' modification on/with '"+di.name+"' current server value
["+di.curValue+"] , value sent ["+di.newValue+"] Please fix!");
            }
        }
    }
}
```

## DataSet.saveToSharedObj()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.saveToSharedObj(objName, [localPath])
```

### Parámetros

**objName** Cadena que especifica el nombre del objeto compartido que va a crearse. El nombre puede incluir barras diagonales (/) (por ejemplo, "direcciones/trabajo"). No se permite el uso de espacios ni de los siguientes caracteres en el nombre que se especifica:

~ % & \ ; : " ' , < > ? #

**localPath** Parámetro de cadena opcional que especifica la ruta completa o parcial al archivo SWF que ha creado el objeto compartido. Esta cadena se utiliza para determinar la ubicación del equipo del usuario en la que almacenará el objeto. El valor predeterminado es la ruta completa del archivo SWF.

### Valor devuelto

Ninguno.

### Descripción

Método; guarda todos los datos necesarios para restaurar esta colección DataSet en un objeto compartido. De esta forma los usuarios pueden trabajar aunque estén desconectados de los datos de origen, si se trata de recursos de red. Este método sobrescribe los datos que existen dentro del objeto de datos compartido de esta colección DataSet. Para restaurar una colección DataSet desde un objeto compartido, utilice `DataSet.loadFromSharedObj()`. Tenga en cuenta que el nombre de esta colección DataSet se utiliza para identificar los datos dentro de un objeto compartido determinado.

Si no puede crearse el objeto compartido o si se produce un problema al alinear los datos a éste, el método emitirá la excepción `DataSetError`.

### Ejemplo

En este ejemplo se llama a `saveToSharedObj()` en un bloque `try...catch` y se muestra un error si se produce algún problema al guardar los datos en el objeto compartido.

```
try {  
    myDataSet.saveToSharedObj("webapp/customerInfo");  
}  
catch(e:DataSetError) {  
    trace("Unable to create shared object");  
}
```

### Véase también

[DataSet.loadFromSharedObj\(\)](#)

## DataSet.schema

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

`dataSet.schema`



## Descripción

Propiedad; proporciona la representación en XML del esquema para este objeto DataSet. El código XML asignado a esta propiedad debe tener el formato siguiente:

```
<?xml version="1.0"?>
<propiedades>
  <property name="propertyName">
    <type name="dataType" />
    <encoder name="dataType">
      <options>
        <dataFormat>format options</dataFormat>
      </options>
    </encoder>
    <kind name="dataKind">
      <options/>
    </kind>
  </property>
  <property> ... </property>
  ...
</propiedades>
```

Se emitirá una excepción DataSetError si el código XML especificado no respeta el formato anterior.

## Ejemplo

```
myDataSet.schema = new XML("<properties><property name='billable'> ..etc.. </properties>");
```

## DataSet.selectedIndex

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.selectedIndex
```

## Descripción

Propiedad; especifica el índice seleccionado dentro de la colección. Esta propiedad puede vincularse al elemento seleccionado en un componente DataGrid o List, y viceversa. Si desea ver un ejemplo completo de demostración, consulte [“Creación de aplicaciones con el componente DataSet” en la página 212](#).

## Ejemplo

En el siguiente ejemplo se establece el índice seleccionado de un objeto DataSet (userData) en el índice seleccionado de un componente DataGrid (userGrid).

```
userData.selectedIndex = userGrid.selectedIndex;
```

## DataSet.setIterator()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.setIterator(iterator)
```

### Parámetros

*iterator* Objeto repetidor devuelto por una llamada a [DataSet.getIterator\(\)](#).

### Valor devuelto

Ninguno.

### Descripción

Método; asigna el repetidor especificado a este objeto DataSet y lo convierte en el repetidor actual. El repetidor especificado debe provenir de una llamada anterior a [DataSet.getIterator\(\)](#) en el objeto DataSet al que se asigne; de lo contrario, se emitirá una excepción DataSetError.

### Ejemplo

```
myIterator:ValueListIterator = myDataSet.getIterator();  
myIterator.sortOn(["name"]);  
myDataSet.setIterator(myIterator);
```

### Véase también

[DataSet.getIterator\(\)](#)

## DataSet.setRange()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.setRange(startValues, endValues)
```

### Parámetros

*startValues* Matriz de valores clave de las propiedades del primer objeto de transferencia del rango.

*endValues* Matriz de valores clave de las propiedades del último objeto de transferencia del rango.

## Valor devuelto

Ninguno.

## Descripción

Método; establece los puntos finales del repetidor actual. Los puntos finales definen un rango dentro del cual opera el repetidor. Sólo es válido si se ha establecido una clasificación válida para el repetidor actual mediante `DataSet.applyUpdates()`.

Establecer un rango para el repetidor actual es más eficaz que utilizar una función de filtro si lo que se desea es agrupar los valores (consulte `DataSet.filterFunc`).

## Ejemplo

```
myDataSet.addSort("name_id", ["name", "id"]);
myDataSet.setRange(["Bobby", 105],["Cathy", 110]);
while ( myDataSet.hasNext() ) {
    myDataSet.gradeLevel = "5"; // cambiar todas las calificaciones de este rango
    myDataSet.next();
}
myDataSet.removeRange();
myDataSet.removeSort("name_id");
```

## Véase también

`DataSet.applyUpdates()`, `DataSet.hasNext()`, `DataSet.next()`, `DataSet.removeRange()`, `DataSet.removeSort()`

## DataSet.skip()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.skip(offset)
```

### Parámetros

*offset*    Número entero que especifica el número de registros que se desplazará la posición del repetidor.

### Valor devuelto

Ninguno.

### Descripción

Método; desplaza la posición del repetidor actual hacia delante o hacia atrás en función del número especificado por *offset*. Los valores positivos de *offset* desplazan la posición del repetidor actual hacia delante, mientras que los negativos la desplazan hacia atrás.

Si el valor de *offset* especificado se encuentra fuera del principio (o del final) de la colección, el repetidor se colocará al principio (o al final) de la colección.

## Ejemplo

En este ejemplo se coloca el repetidor actual en el primer elemento de la colección, a continuación se desplaza al penúltimo elemento y por último se realiza un cálculo en un campo de dicho elemento.

```
myDataSet.first();  
// Desplazarse hasta el penúltimo elemento  
var itemsToSkip = myDataSet.length - 2;  
myDataSet.skip(itemsToSkip).price = myDataSet.amount * 10;
```

## DataSet.useSort()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
dataSet.useSort(sortName, order)
```

### Parámetros

*sortName* Cadena que contiene el nombre de la clasificación que va a utilizarse.

*order* Valor entero que indica el orden de la clasificación; el valor debe ser `DataSetIterator.Ascending` o `DataSetIterator.Descending`.

### Valor devuelto

Ninguno.

### Descripción

Método; cambia la clasificación del repetidor actual a la que se especifica en *sortName*, si existe. Si la clasificación especificada por *sortName* no existe, se emite una excepción `DataSetError`.

Para crear una clasificación, utilice `DataSet.applyUpdates()`.

## Ejemplo

Este código utiliza `DataSet.hasSort()` para determinar si una clasificación denominada "customer" existe o no. Si existe, el código llama a `DataSet.useSort()` para que "customer" sea la clasificación actual. De lo contrario, el código crea una clasificación con ese nombre mediante `DataSet.addSort()`.

```
if(myDataSet.hasSort("customer")) {  
    myDataSet.useSort("customer");  
} else {  
    myDataSet.addSort("customer", ["customer"], DataSetIterator.Descending);  
}
```

### Véase también

`DataSet.applyUpdates()`, `DataSet.hasSort()`

## Componente DateChooser (sólo para Flash Professional)

El componente DateChooser es un calendario que permite a los usuarios seleccionar una fecha. Tiene botones que permiten al usuario desplazarse por los meses y hacer clic en cualquier fecha para seleccionarla. Pueden establecerse parámetros que indican el nombre del mes y del día, el primer día de la semana o las fechas desactivadas. También es posible resaltar la fecha actual.

La vista previa dinámica de cada instancia de DateChooser refleja los valores indicados durante la edición en el inspector de propiedades o en el panel Inspector de componentes.

## Componente DateChooser (sólo para Flash Professional)

Utilice DateChooser siempre que desee que el usuario seleccione una fecha. Por ejemplo, podría utilizarse el componente DateChooser en un sistema de reservas de hotel en el que unas fechas determinadas puedan seleccionarse y otras queden desactivadas. También podría utilizarse el componente DateChooser en una aplicación que muestre, cuando el usuario elija una fecha, los eventos en curso, como por ejemplo actuaciones o reuniones.

### Parámetros de DateChooser

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente DateChooser en el inspector de propiedades o en el panel Inspector de componentes:

**monthNames** establece el nombre de los meses que se muestran en la fila de encabezado del calendario. El valor es una matriz y el valor predeterminado es ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"].

**dayNames** establece el nombre de los días de la semana. El valor es una matriz y el valor predeterminado es ["S", "M", "T", "W", "T", "F", "S"].

**firstDayOfWeek** indica el día de la semana que se muestra en la primera columna de DateChooser (0-6, con 0 como primer elemento de la matriz de dayNames). Esta propiedad cambia el orden de visualización de las columnas de los días.

**disabledDays** indica los días de la semana que están desactivados. Este parámetro es una matriz y puede tener hasta 7 valores. El valor predeterminado es [] (matriz vacía).

**showToday** indica si debe resaltarse la fecha actual o no. El valor predeterminado es `true`.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes de DateChooser utilizando sus propiedades, métodos y eventos. Para más información, consulte [“Clase DateChooser \(sólo para Flash Professional\)” en la página 259](#).

## Creación de aplicaciones con el componente DateChooser

El procedimiento siguiente explica cómo añadir un componente DateChooser a una aplicación durante la edición. En este ejemplo, DateChooser permite al usuario elegir una fecha en un sistema de reservas de una compañía aérea. Todas las fechas anteriores al 15 de octubre deben quedar desactivadas. También debe desactivarse un rango de días de diciembre, de manera que se cree un período de vacaciones que quede excluido; además los lunes deben quedar desactivados.

**Para crear una aplicación con el componente DateChooser, siga este procedimiento:**

- 1 En el panel Componentes haga doble clic en el componente DateChooser para añadirlo al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia **flightCalendar**.
- 3 En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para establecer un rango de fechas seleccionables:

```
flightCalendar.selectableRange = {rangeStart:new Date(2003, 9, 15),  
rangeEnd:new Date(2003, 11, 31)}
```

Este código asigna un valor a la propiedad `selectableRange` de un objeto ActionScript que contenga dos objetos Date con los nombres de variable `rangeStart` y `rangeEnd`. De esta forma se definen los límites superior e inferior de un rango dentro del cual el usuario puede seleccionar una fecha.

- 4 En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para establecer un rango de fechas desactivadas de días festivos:

```
flightCalendar.disabledRanges = [{rangeStart: new Date(2003, 11, 15),  
rangeEnd: new Date(2003, 11, 26)}];
```

- 5 En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para que los lunes queden desactivados:

```
flightCalendar.disabledDays=[1];
```

- 6 Control > Probar película.

## Personalización del componente DateChooser (sólo para Flash Professional)

El componente DateChooser puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)).

## Utilización de estilos con el componente DateChooser

Se pueden definir propiedades de estilo para cambiar el aspecto de una instancia del selector de fechas. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

El componente DateChooser admite los siguientes estilos de Halo:

| Estilo         | Descripción                                                                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| themeColor     | Color de la iluminación de la sustitución y de las fechas seleccionadas. Es el único estilo de color que no hereda su valor. Los valores posibles son "haloGreen", "haloBlue" y "haloOrange". |
| color          | Texto de la etiqueta de un componente.                                                                                                                                                        |
| disabledColor  | Color desactivado para el texto.                                                                                                                                                              |
| fontFamily     | Nombre de la fuente del texto.                                                                                                                                                                |
| fontSize       | Tamaño de la fuente en puntos.                                                                                                                                                                |
| fontStyle      | Estilo de la fuente; puede ser "normal" o "italic".                                                                                                                                           |
| fontWeight     | Grosor de la fuente; puede ser "normal" o "bold".                                                                                                                                             |
| textDecoration | Decoración del texto: puede ser "none" o "underline".                                                                                                                                         |

## Utilización de aspectos con el componente DateChooser

El componente DateChooser emplea aspectos para representar sus estados visuales. Para aplicar un aspecto al componente DateChooser durante la edición, modifique los símbolos del aspecto en la carpeta de estados de aspecto Flash UI Components 2/Themes/MMDefault/DateChooser Assets/Elements/Month de la biblioteca de uno de los archivos FLA de temas. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).

En este componente sólo se pueden aplicar aspectos a los botones de desplazamiento de mes. Un componente DateChooser utiliza las propiedades de aspecto siguientes:

| Propiedad         | Descripción                                                                               |
|-------------------|-------------------------------------------------------------------------------------------|
| falseUpSkin       | Estado Arriba. Los valores predeterminados son fwdMonthUp y backMonthUp.                  |
| falseDownSkin     | Estado Abajo. Los valores predeterminados son fwdMonthDown y backMonthDown.               |
| falseDisabledSkin | Estado Desactivado. Los valores predeterminados son fwdMonthDisabled y backMonthDisabled. |

## Clase DateChooser (sólo para Flash Professional)

**Herencia** UIObject > UIComponent > DateChooser

**Nombre de clase de ActionScript** mx.controls.DateChooser

Las propiedades de la clase DateChooser permiten acceder a la fecha seleccionada y al mes y día seleccionados. También se puede indicar el nombre de los días y los meses, las fechas desactivadas y seleccionables, establecer el primer día de la semana y decidir si debe resaltarse o no la fecha actual.

Si una propiedad de la clase DateChooser se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.DateChooser.version);
```

**Nota:** el código siguiente devuelve `undefined`: `trace(myDC.version)`;

## Resumen de métodos de la clase `DateChooser`

Hereda todos los métodos de `UIObject` y `UIComponent`.

## Resumen de propiedades de la clase `DateChooser`

| Propiedad                                | Descripción                                                                                                                    |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>DateChooser.dayNames</code>        | Matriz que indica el nombre de los días de la semana.                                                                          |
| <code>DateChooser.disabledDays</code>    | Matriz que indica los días de la semana desactivados para todas las fechas aplicables del selector de fechas.                  |
| <code>DateChooser.disabledRanges</code>  | Rango de fechas desactivadas o fecha única desactivada.                                                                        |
| <code>DateChooser.displayedMonth</code>  | Número que indica un elemento de la matriz <code>monthNames</code> para mostrarlo en el selector de fechas.                    |
| <code>DateChooser.displayedYear</code>   | Número que indica el año que se mostrará.                                                                                      |
| <code>DateChooser.firstDayOfWeek</code>  | Número que indica un elemento de la matriz <code>dayNames</code> que se mostrará en la primera columna del selector de fechas. |
| <code>DateChooser.monthNames</code>      | Matriz de cadenas que indica el nombre de los meses.                                                                           |
| <code>DateChooser.selectableRange</code> | Fecha única seleccionable o rango de fechas seleccionables.                                                                    |
| <code>DateChooser.selectedDate</code>    | Objeto <code>Date</code> que indica la fecha seleccionada.                                                                     |
| <code>DateChooser.showToday</code>       | Valor booleano que indica si debe resaltarse o no la fecha actual.                                                             |

Hereda todas las propiedades de `UIObject` y `UIComponent`.

## Resumen de eventos de la clase `DateChooser`

| Evento                          | Descripción                                        |
|---------------------------------|----------------------------------------------------|
| <code>DateChooser.change</code> | Se difunde cuando se selecciona una fecha.         |
| <code>DateChooser.scroll</code> | Se difunde cuando se presionan los botones de mes. |

Hereda todos los eventos de `UIObject` y `UIComponent`.



## DateChooser.change

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
on(change){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    ...  
}  
textInputInstance.addEventListener("change", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando se selecciona una fecha.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `DateChooser`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el selector de fechas `myDC`, envía “\_level0.myDC” al panel Salida:

```
on(change){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*chooserInstance*) distribuye un evento (en este caso, `change`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

Este ejemplo, escrito en un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se cambia un selector de fechas (DateChooser) denominado `myDC`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `change` en el objeto detector. Dentro de la función hay una acción `trace()` que utiliza el objeto Event pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto Event es el componente que ha generado el evento (en este ejemplo `myDC`). A la propiedad `NumericStepper.maximum` se accede desde la propiedad `target` del objeto Event. La última línea llama al método `UIEventDispatcher.addEventListener()` desde `myDC` y pasa como parámetros el evento `change` y el objeto detector `form`, como se indica a continuación:

```
form.change = function(eventObj){
    trace("date selected " + eventObj.target.selectedDate) ;
}
myDC.addEventListener("change", form);
```

## DateChooser.dayNames

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myDC.dayNames`

### Descripción

Propiedad; matriz que contiene el nombre de los días de la semana. El domingo es el primer día (en la posición de índice 0) y el resto de los nombres de día siguen en el orden habitual. El valor predeterminado es ["S", "M", "T", "W", "T", "F", "S"].

### Ejemplo

En el ejemplo siguiente el valor del quinto día de la semana, jueves (Thursday), cambia de "T" a "R":

```
myDC.dayNames[4] = "R";
```

## DateChooser.disabledDays

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myDC.disabledDays`

## Descripción

Propiedad; matriz que indica cuáles son los días de la semana que están desactivados. Todas las fechas del mes que caigan en el día especificado quedarán desactivadas. Los valores de los elementos de esta matriz pueden estar comprendidos entre el 0, para el domingo, y el 6, para el sábado. El valor predeterminado es [] (matriz vacía).

## Ejemplo

En el ejemplo siguiente se desactivan los domingos y los sábados, de forma que los usuarios sólo puedan seleccionar el resto de los días:

```
myDC.disabledDays = [0, 6];
```

## DateChooser.disabledRanges

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDC.disabledRanges
```

## Descripción

Propiedad; desactiva un único día o un rango de días. Esta propiedad es una matriz de objetos. Cada objeto de la matriz debe ser un objeto Date, que especifique una fecha única para desactivarla, o bien un objeto que contenga las propiedades rangeStart y rangeEnd (o al menos una de ellas), cuyos valores deben ser un objeto Date. Las propiedades rangeStart y rangeEnd describen los límites del rango de fechas. Si se omite alguna de dichas propiedades el rango quedará abierto por la fecha correspondiente.

El valor predeterminado de disabledRanges no está definido (undefined).

Especifique una fecha completa al definir fechas para la propiedad disabledRanges. Por ejemplo, new Date(2003,6,24) en lugar de new Date(). Si no se especifica una fecha completa, la hora devuelta es 00:00:00.

## Ejemplo

En el ejemplo siguiente se define una matriz con los objetos Date rangeStart y rangeEnd, que desactivan las fechas comprendidas entre el 7 de mayo y el 7 de junio:

```
myDC.disabledRanges = [ {rangeStart: new Date(2003, 4, 7), rangeEnd: new  
    Date(2003, 5, 7)}];
```

En el ejemplo siguiente se desactivan todas las fechas posteriores al 7 de noviembre:

```
myDC.disabledRanges = [ {rangeStart: new Date(2003, 10, 7)} ];
```

En el ejemplo siguiente se desactivan todas las fechas anteriores al 7 de octubre:

```
myDC.disabledRanges = [ {rangeEnd: new Date(2002, 9, 7)} ];
```

En el ejemplo siguiente se desactiva únicamente el 7 de diciembre:

```
myDC.disabledRanges = [ new Date(2003, 11, 7) ];
```

## DateChooser.displayedMonth

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myDC.displayedMonth`

### Descripción

Propiedad; número que indica el mes que se visualiza. El número indica un elemento de la matriz `monthNames`, donde 0 indica el primer mes. El valor predeterminado es el mes correspondiente a la fecha actual.

### Ejemplo

En el ejemplo siguiente, se define diciembre como el mes que se visualizará:

```
myDC.displayedMonth = 11;
```

### Véase también

[DateChooser.displayedYear](#)

## DateChooser.displayedYear

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myDC.displayedYear`

### Descripción

Propiedad; número de cuatro dígitos que indica el año que se visualiza. El valor predeterminado es el año actual.

### Ejemplo

En el ejemplo siguiente, se define el año 2010 como el año que se visualizará:

```
myDC.displayedYear = 2010;
```

### Véase también

[DateChooser.displayedMonth](#)

## DateChooser.firstDayOfWeek

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDC.firstDayOfWeek*

### Descripción

Propiedad; número que indica el día de la semana (de 0 a 6, donde 0 indica el primer elemento de la matriz `dayNames`) que se visualiza en la primera columna del componente `DateChooser`. Si se cambia esta propiedad se cambiará el orden de las columnas de día pero no se modificará el orden de la propiedad `dayNames`. El valor predeterminado es 0 (domingo).

### Ejemplo

En el ejemplo siguiente se establece el lunes como primer día de la semana:

```
myDC.firstDayOfWeek = 1;
```

### Véase también

[DateChooser.dayNames](#)

## DateChooser.monthNames

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDC.monthNames*

### Descripción

Propiedad; matriz de cadenas que indica los nombres de mes en la parte superior del componente `DateChooser`. El valor predeterminado es ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"].

### Ejemplo

En el ejemplo siguiente se definen los nombres de mes para la instancia `myDC`:

```
myDC.monthNames = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",  
"Sept", "Oct", "Nov", "Dec"];
```

## DateChooser.scroll

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
on(scroll){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.scroll = function(eventObject){  
    ...  
}  
scrollPaneInstance.addEventListener("scroll", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando se presiona un botón de mes.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `DateChooser`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el selector de fechas `myDC`, envía “\_level0.myDC” al panel Salida:

```
on(scroll){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*myDC*) distribuye un evento (en este caso, `scroll`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto `Event` del evento `scroll` tiene una propiedad adicional, `detail`, que puede tener uno de los valores siguientes: `nextMonth`, `previousMonth`, `nextYear`, `previousYear`.

Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 602](#).

## Ejemplo

En este ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se presiona un botón de mes en una instancia de `DateChooser` denominada `myDC`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `scroll` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto `Event` pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto `Event` es el componente que ha generado el evento, en este ejemplo `myDC`. La última línea llama al método

`UIEventDispatcher.addEventListener()` desde `myDC` y pasa como parámetros el evento `scroll` y el objeto detector `form`:

```
form = new Object();
form.scroll = function(eventObj){
    trace(eventObj.detail);
}
spInstance.addEventListener("scroll", form);
```

## DateChooser.selectableRange

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myDC.selectableRange`

### Descripción

Propiedad; establece una fecha única seleccionable o un rango de fechas seleccionables. El usuario no podrá desplazarse fuera del rango seleccionable. El valor de esta propiedad es un objeto compuesto por dos objetos `Date` denominados `rangeStart` y `rangeEnd`. Las propiedades `rangeStart` y `rangeEnd` describen los límites del rango de fechas elegible. Si sólo se define `rangeStart`, se activarán todas las fechas posteriores a la señalada en `rangeStart`. Si sólo se define `rangeEnd`, se activarán todas las fechas anteriores a la señalada en `rangeEnd`. El valor predeterminado es `undefined`.

Si lo que se desea es activar un único día, puede utilizarse un único objeto `Date` como valor de `selectableRange`.

Al definir fechas deben especificarse de forma completa. Por ejemplo, `new Date(2003,6,24)` en lugar de `new Date()`. Si no se especifica una fecha completa, la hora devuelta es `00:00:00`.

El valor de `DateChooser.selectedDate` queda establecido como no definido (`undefined`) si cae fuera del rango seleccionable.

Los valores de `DateChooser.displayedMonth` y `DateChooser.displayedYear` se establecen en el último mes más próximo del intervalo seleccionable si el mes actual cae fuera de dicho rango. Por ejemplo, si el mes que se muestra como actual es agosto (August), y el rango seleccionable comprende el periodo que va desde junio de 2003 a julio de 2003 el mes que se mostrará cambiará a julio de 2003 (July).

## Ejemplo

En el ejemplo siguiente se define como rango seleccionable las fechas comprendidas entre el 7 de mayo y el 7 de junio, ambas inclusive:

```
myDC.selectableRange = {rangeStart: new Date(2001, 4, 7), rangeEnd: new Date(2003, 5, 7)};
```

En el ejemplo siguiente se define como rango seleccionable el 7 de mayo y las fechas posteriores a ese día:

```
myDC.selectableRange = {rangeStart: new Date(2003, 4, 7)};
```

En el ejemplo siguiente se define como rango seleccionable el 7 de junio y las fechas anteriores a ese día:

```
myDC.selectableRange = {rangeEnd: new Date(2003, 5, 7) };
```

En el ejemplo siguiente se define el 7 de junio como única fecha seleccionable:

```
myDC.selectableRange = new Date(2003, 5, 7);
```

## DateChooser.selectedDate

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDC.selectedDate*

### Descripción

Propiedad; objeto Date que indica la fecha seleccionada si el valor se encuentra dentro del valor especificado en la propiedad `selectableRange`. El valor predeterminado es `undefined`.

La propiedad `selectedDate` no puede definirse con un valor que se encuentre dentro del intervalo especificado en `disabledRange` o fuera del especificado en `selectableRange` ni con un valor que coincida con un día que se haya desactivado. Si la propiedad `selectedDate` se define con cualquiera de las fechas indicadas, el valor quedará como no definido (`undefined`).

### Ejemplo

En el ejemplo siguiente se define el 7 de junio como fecha seleccionada:

```
myDC.selectedDate = new Date(2003, 5, 7);
```

## DateChooser.showToday

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.



## Sintaxis

```
myDC.showToday
```

## Descripción

Propiedad; esta propiedad determina si debe resaltarse o no la fecha actual. El valor predeterminado es `true`.

## Ejemplo

En el ejemplo siguiente se desactiva el resaltado de la fecha correspondiente al día de hoy:

```
myDC.showToday = false;
```

# Componente DateField (sólo para Flash Professional)

El componente DateField es un campo de texto no seleccionable que muestra la fecha con un icono de calendario situado a su derecha. Si no se ha seleccionado ninguna fecha, el campo de texto estará vacío y se muestra el mes correspondiente al día de hoy en el selector de fechas. Cuando el usuario hace clic dentro del recuadro de delimitación del campo de fecha, aparece el selector de fechas con el mes correspondiente a la fecha seleccionada. Cuando el selector de fechas está abierto, el usuario puede usar los botones de desplazamiento para cambiar de mes y año y seleccionar una fecha. El selector de fechas se cierra cuando se selecciona una fecha.

La vista previa dinámica de DateField no refleja los valores indicados en el inspector de propiedades ni en el panel Inspector de componentes durante la edición porque se trata de un componente emergente que no puede verse durante la edición.

## Utilización del componente DateField (sólo para Flash Professional)

Utilice DateField siempre que desee que el usuario seleccione una fecha. Por ejemplo, puede utilizarse el componente DateField en un sistema de reservas de hotel en el que unas fechas determinadas puedan seleccionarse y otras queden desactivadas. También puede utilizarse el componente DateField en una aplicación que muestre, cuando el usuario elija una fecha, los eventos en curso, como por ejemplo actuaciones o reuniones.

## Parámetros de DateField

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente DateField en el inspector de propiedades o en el panel Inspector de componentes:

**monthNames** establece el nombre de los meses que se muestran en la fila de encabezado del calendario. El valor es una matriz y el valor predeterminado es ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"].

**dayNames** establece el nombre de los días de la semana. El valor es una matriz y el valor predeterminado es ["S", "M", "T", "W", "T", "F", "S"].

**firstDayOfWeek** indica el día de la semana que se muestra en la primera columna de DateChooser (0-6, con 0 como primer elemento de la matriz de dayNames). Esta propiedad cambia el orden de visualización de las columnas de los días.

El valor predeterminado es 0, que corresponde a "S".

**disabledDays** indica los días de la semana que están desactivados. Este parámetro es una matriz y puede tener hasta 7 valores. El valor predeterminado es [] (matriz vacía).

**showToday** indica si debe resaltarse la fecha actual o no. El valor predeterminado es true.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes DateField utilizando sus propiedades, métodos y eventos. Para más información, consulte [“Clase DateField \(sólo para Flash Professional\)” en la página 272](#).

## Creación de aplicaciones con el componente DateField

El procedimiento siguiente explica cómo añadir un componente DateField a una aplicación durante la edición. En este ejemplo, el componente DateField permite al usuario elegir una fecha en un sistema de reservas de una compañía aérea. Todas las fechas anteriores a la de hoy deben quedar desactivadas. También debe desactivarse un rango de 15 días de diciembre, de manera que se cree un período de vacaciones que quede excluido. Además, los lunes hay algunos vuelos que no están disponibles, por lo que todos los lunes correspondientes a dichos vuelos deben quedar desactivados.

**Para crear una aplicación con el componente DateField, siga este procedimiento:**

- 1 En el panel Componentes haga doble clic en el componente DateField para añadirlo al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia **flightCalendar**.
- 3 En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para establecer un rango de fechas seleccionables:

```
flightCalendar.selectableRange = {rangeStart:new Date(2001, 9, 1),  
rangeEnd:new Date(2003, 11, 1)};
```

Este código asigna un valor a la propiedad **selectableRange** de un objeto ActionScript que contenga dos objetos Date con los nombres de variable **rangeStart** y **rangeEnd**. De esta forma se definen los límites superior e inferior de un rango dentro del cual el usuario puede seleccionar una fecha.

- 4 En el panel Acciones introduzca el código siguiente en el fotograma 1 de la línea de tiempo para establecer los intervalos de fechas desactivadas: una durante el mes de diciembre y otra para todas las fechas anteriores a la actual:

```
flightCalendar.disabledRanges = [{rangeStart: new Date(2003, 11, 15),  
rangeEnd: new Date(2003, 11, 31)}, {rangeEnd: new Date(2003, 6, 16)}];
```

- 5 En el panel Acciones, introduzca el código siguiente en el fotograma 1 de la línea de tiempo para que los lunes queden desactivados:

```
flightCalendar.disabledDays=[1];
```

- 6 Control > Probar película.

## Personalización del componente DateField (sólo para Flash Professional)

El componente DateField puede transformarse horizontalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método **setSize()** (véase [UIObject.setSize\(\)](#)). Al definir la anchura no se cambian las dimensiones del selector de fecha en el componente DateField. No obstante, se puede utilizar la propiedad **pullDown** para acceder al componente DateChooser y definir sus dimensiones.

## Utilización de estilos con el componente DateField

Puede definir propiedades de estilo para cambiar el aspecto de una instancia de campo de fecha. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

El componente DateField admite los siguientes estilos de Halo:

| Estilo         | Descripción                                                                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| themeColor     | Color de la iluminación de la sustitución y de las fechas seleccionadas. Es el único estilo de color que no hereda su valor. Los valores posibles son "haloGreen", "haloBlue" y "haloOrange". |
| color          | Texto de la etiqueta de un componente.                                                                                                                                                        |
| disabledColor  | Color desactivado para el texto.                                                                                                                                                              |
| fontFamily     | Nombre de la fuente del texto.                                                                                                                                                                |
| fontSize       | Tamaño de la fuente en puntos.                                                                                                                                                                |
| fontStyle      | Estilo de la fuente; puede ser “normal” o “italic”.                                                                                                                                           |
| fontWeight     | Grosor de la fuente; puede ser “normal” o “bold”.                                                                                                                                             |
| textDecoration | Decoración del texto; puede ser “none” o “underline”.                                                                                                                                         |

## Utilización de aspectos con el componente DateField

El componente DateField utiliza aspectos para representar los estados visuales del icono emergente. Para aplicar aspectos al icono emergente durante la edición, modifique los símbolos de aspecto en la carpeta de estados de aspectos Flash UI Components 2/Themes/MMDefault/DateField Elements en la biblioteca de uno de los archivos FLA de temas. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).

En este componente sólo se puede aplicar aspectos al botón de icono emergente. Un componente DateField utiliza las propiedades de aspecto siguientes para aplicar dinámicamente un aspecto al icono emergente:

| Propiedad        | Descripción                             |
|------------------|-----------------------------------------|
| openDateUp       | Estado Arriba del icono emergente.      |
| openDateDown     | Estado Abajo del icono emergente.       |
| openDateOver     | Estado Sobre del icono emergente.       |
| openDateDisabled | Estado Desactivado del icono emergente. |

## Clase DateField (sólo para Flash Professional)

**Herencia** UIObject > UIComponent > ComboBase > DateField

**Nombre de clase de ActionScript** mx.controls.DateField

Las propiedades de la clase DateField permiten acceder a la fecha seleccionada, y al mes y día seleccionados. También se puede indicar el nombre de los días y los meses, las fechas desactivadas y seleccionables, establecer el primer día de la semana y decidir si debe resaltarse o no la fecha actual.

Si una propiedad de la clase DateField se define con ActionScript, se sustituye temporalmente el parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.DateField.version);
```

**Nota:** el código siguiente devuelve undefined: `trace(myDateFieldInstance.version);`.

### Resumen de métodos de la clase DateField

| Método                         | Descripción                                    |
|--------------------------------|------------------------------------------------|
| <code>DateField.close()</code> | Cierra el subcomponente DateChooser emergente. |
| <code>DateField.open()</code>  | Abre el subcomponente DateChooser emergente.   |

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

### Resumen de propiedades de la clase DateField

| Propiedad                             | Descripción                                                                                                                    |
|---------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>DateField.dateFormatter</code>  | Función que da formato a la fecha que va a visualizarse en el campo de texto.                                                  |
| <code>DateField.dayNames</code>       | Matriz que indica el nombre de los días de la semana.                                                                          |
| <code>DateField.disabledDays</code>   | Matriz que indica los días de la semana desactivados para todas las fechas aplicables del selector de fechas.                  |
| <code>DateField.disabledRanges</code> | Rango de fechas desactivadas o fecha única desactivada.                                                                        |
| <code>DateField.displayedMonth</code> | Número que indica un elemento de la matriz <code>monthNames</code> para mostrarlo en el selector de fechas.                    |
| <code>DateField.displayedYear</code>  | Número que indica el año que se mostrará.                                                                                      |
| <code>DateField.firstDayOfWeek</code> | Número que indica un elemento de la matriz <code>dayNames</code> que se mostrará en la primera columna del selector de fechas. |
| <code>DateField.monthNames</code>     | Matriz de cadenas que indica el nombre de los meses.                                                                           |
| <code>DateField.pullDown</code>       | Una referencia al subcomponente DateChooser. Es una propiedad de sólo lectura.                                                 |

| Propiedad                              | Descripción                                                        |
|----------------------------------------|--------------------------------------------------------------------|
| <code>DateField.selectableRange</code> | Fecha única seleccionable o rango de fechas seleccionables.        |
| <code>DateField.selectedDate</code>    | Objeto Date que indica la fecha seleccionada.                      |
| <code>DateField.showToday</code>       | Valor booleano que indica si debe resaltarse o no la fecha actual. |

Hereda todas las propiedades de [UIObject](#) y [UIComponent](#).

## Resumen de eventos de la clase DateField

| Evento                        | Descripción                                               |
|-------------------------------|-----------------------------------------------------------|
| <code>DateField.change</code> | Se difunde cuando se selecciona una fecha.                |
| <code>DateField.close</code>  | Se difunde cuando se cierra el subcomponente DateChooser. |
| <code>DateField.open</code>   | Se difunde cuando se abre el subcomponente DateChooser.   |
| <code>DateField.scroll</code> | Se difunde cuando se presionan los botones de mes.        |

Hereda todos los eventos de [UIObject](#) y [UIComponent](#).

## DateField.change

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
on(change){
    ...
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.change = function(eventObject){
    ...
}
myDF.addEventListener("change", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando se selecciona una fecha.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `DateField`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el campo de fecha `myDF`, envía “\_level0.myDF” al panel Salida:

```
on(change){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*chooserInstance*) distribuye un evento (en este caso, `change`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 602](#).

## Ejemplo

En este ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se cambia un campo de fecha denominado `myDF`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `change` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto `Event` pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto `Event` es el componente que ha generado el evento, en este ejemplo `myDF`. A la propiedad `DateField.selectedDate` se accede desde la propiedad `target` del objeto `Event`. La última línea llama al método `UIEventDispatcher.addEventListener()` desde `myDF` y pasa como parámetros el evento `change` y el objeto detector `form`, como se indica a continuación:

```
form.change = function(eventObj){
    trace("date selected " + eventObj.target.selectedDate) ;
}
myDF.addEventListener("change", form);
```

## DateField.close()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDF.close()
```

**Parámetros**

Ninguno.

**Valor devuelto**

Ninguno.

**Descripción**

Método; cierra el menú emergente.

**Ejemplo**

El código siguiente cierra el icono emergente del selector de fecha de la instancia de campo de fecha `myDF`:

```
myDF.close();
```

**DateField.close****Disponibilidad**

Flash Player 6 versión 79.

**Edición**

Flash MX Professional 2004.

**Sintaxis**

Sintaxis 1:

```
on(close){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.close = function(eventObject){  
    ...  
}  
myDF.addEventListener("close", listenerObject)
```

**Descripción**

Evento; se difunde a todos los detectores registrados cuando el subcomponente `DateChooser` se cierra como consecuencia de que un usuario ha hecho clic en un área fuera del icono o ha seleccionado una fecha.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `DateField`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el campo de fecha `myDF`, envía “\_level0.myDF” al panel Salida:

```
on(close){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*myDF*) distribuye un evento (en este caso, `close`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 602.

### Ejemplo

Este ejemplo, escrito en un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se cierra el selector de fecha de *myDF*. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `close` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto `Event` pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto `Event` es el componente que ha generado el evento, en este ejemplo *myDF*. A la propiedad se accede desde la propiedad `target` del objeto `Event`. La última línea llama al método `UIEventDispatcher.addEventListener()` desde *myDF* y pasa como parámetros el evento `close` y el objeto detector `form`, como se indica a continuación:

```
form.close = function(eventObj){
    trace("PullDown Closed" + eventObj.target.selectedDate);
}
myDF.addEventListener("close", form);
```

## DateField.dateFormatter

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDF*.dateFormatter

### Descripción

Propiedad; función que da formato a la fecha que va a visualizarse en el campo de texto. La función debe recibir un objeto `Date` como parámetro y devuelve una cadena en el formato que va a visualizarse.



## Ejemplo

En el ejemplo siguiente se define la función para que devuelva el formato de fecha que va a visualizarse:

```
myDF.dateFormatter = function(d:Date){  
    return d.getFullYear()+"/ "+(d.getMonth()+1)+"/ "+d.getDate();  
};
```

## DateField.dayNames

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDF.dayNames*

### Descripción

Propiedad; matriz que contiene el nombre de los días de la semana. El domingo es el primer día (en la posición de índice 0) y el resto de los nombres de día siguen en el orden habitual. El valor predeterminado es ["S", "M", "T", "W", "T", "F", "S"].

### Ejemplo

En el ejemplo siguiente el valor del quinto día de la semana, jueves (Thursday), cambia de “T” a “R”:

```
myDF.dayNames[4] = "R";
```

## DateField.disabledDays

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDF.disabledDays*

### Descripción

Propiedad; matriz que indica cuáles son los días de la semana que están desactivados. Todas las fechas del mes que caigan en el día especificado quedarán desactivadas. Los valores de los elementos de esta matriz pueden estar comprendidos entre el 0, para el domingo, y el 6, para el sábado. El valor predeterminado es [] (matriz vacía).

## Ejemplo

En el ejemplo siguiente se desactivan los domingos y los sábados de forma que los usuarios sólo puedan seleccionar el resto de los días:

```
myDF.disabledDays = [0, 6];
```

## DateField.disabledRanges

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDF.disabledRanges
```

### Descripción

Propiedad; desactiva un único día o un rango de días. Esta propiedad es una matriz de objetos. Cada objeto de la matriz debe ser un objeto `Date`, que especifique una fecha única para desactivarla, o bien un objeto que contenga las propiedades `rangeStart` y `rangeEnd` (o al menos una de ellas), cuyos valores deben ser un objeto `Date`. Las propiedades `rangeStart` y `rangeEnd` describen los límites del rango de fechas. Si se omite alguna de dichas propiedades el rango quedará abierto por la fecha correspondiente.

El valor predeterminado de `disabledRanges` no está definido (`undefined`).

Especifique una fecha completa al definir fechas para la propiedad `disabledRanges`. Por ejemplo, `new Date(2003,6,24)` en lugar de `new Date()`. Si no se especifica una fecha completa, la hora devuelta es 00:00:00.

## Ejemplo

En el ejemplo siguiente se define una matriz con los objetos `Date` `rangeStart` y `rangeEnd`, que desactivan las fechas comprendidas entre el 7 de mayo y el 7 de junio:

```
myDF.disabledRanges = [ {rangeStart: new Date(2003, 4, 7), rangeEnd: new  
    Date(2003, 5, 7)} ];
```

En el ejemplo siguiente se desactivan todas las fechas posteriores al 7 de noviembre:

```
myDF.disabledRanges = [ {rangeStart: new Date(2003, 10, 7)} ];
```

En el ejemplo siguiente se desactivan todas las fechas anteriores al 7 de octubre:

```
myDF.disabledRanges = [ {rangeEnd: new Date(2002, 9, 7)} ];
```

En el ejemplo siguiente se desactiva únicamente el 7 de diciembre:

```
myDF.disabledRanges = [ new Date(2003, 11, 7) ];
```

## DateField.displayedMonth

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDF*.displayedMonth

### Descripción

Propiedad; número que indica el mes que se visualiza. El número indica un elemento de la matriz `monthNames`, donde 0 indica el primer mes. El valor predeterminado es el mes correspondiente a la fecha actual.

### Ejemplo

En el ejemplo siguiente, se define diciembre como el mes que se visualizará:

```
myDF.displayedMonth = 11;
```

### Véase también

[DateField.displayedYear](#)

## DateField.displayedYear

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDF*.displayedYear

### Descripción

Propiedad; número que indica el año que se visualiza. El valor predeterminado es el año actual.

### Ejemplo

En el ejemplo siguiente, se define el año 2010 como el año que se visualizará:

```
myDF.displayedYear = 2010;
```

### Véase también

[DateField.displayedMonth](#)

## DateField.firstDayOfWeek

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDF.firstDayOfWeek
```

### Descripción

Propiedad; número que indica el día de la semana (de 0 a 6, con 0 como primer elemento de la matriz `dayNames`) que se visualiza en la primera columna del componente `DateField`. Si se cambia esta propiedad se cambiará el orden de las columnas de día pero no se modificará el orden de la propiedad `dayNames`. El valor predeterminado es 0 (domingo).

### Ejemplo

En el ejemplo siguiente se establece el lunes como primer día de la semana:

```
myDF.firstDayOfWeek = 1;
```

### Véase también

[DateField.dayNames](#)

## DateField.monthNames

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDF.monthNames
```

### Descripción

Propiedad; matriz de cadenas que indica los nombres de mes en la parte superior del componente `DateField`. El valor predeterminado es ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"].

### Ejemplo

En el ejemplo siguiente se definen los nombres de mes para la instancia `myDF`:

```
myDF.monthNames = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",  
    "Sept", "Oct", "Nov", "Dec"];
```

## DateField.open()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDF.open()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; abre el subcomponente DateChooser emergente.

### Ejemplo

El código siguiente abre el selector de fecha emergente de la instancia *df*:

```
df.open();
```

## DateField.open

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
on(open){  
  ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.open = function(eventObject){  
  ...  
}  
myDF.addEventListener("open", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando un subcomponente `DateChooser` se abre como consecuencia de que un usuario ha hecho clic en el icono.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `DateField`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el campo de fecha `myDF`, envía “\_level0.myDF” al panel Salida:

```
on(open){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*myDF*) distribuye un evento (en este caso, `open`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 602.

## Ejemplo

Este ejemplo, escrito en un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se abre un componente denominado `myDF`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `open` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto `Event` pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto `Event` es el componente que ha generado el evento, en este ejemplo `myDF`. A la propiedad `DateField.selectedDate` se accede desde la propiedad `target` del objeto `Event`. La última línea llama al método `UIEventDispatcher.addEventListener()` desde `myDF` y pasa como parámetros el evento `open` y el objeto detector `form`, como se indica a continuación:

```
form.open = function(eventObj){
    trace("Pop-up opened and date selected is " +
        eventObj.target.selectedDate) ;
}
myDF.addEventListener("open", form);
```

## DateField.pullDown

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

```
myDF.pullDown
```

## Descripción

Propiedad (sólo lectura); referencia al componente DateChooser contenido en el componente DateField. Cuando un usuario hace clic en el componente DateField se crea una instancia del subcomponente DateChooser. No obstante, si se hace referencia a la propiedad `pullDown` antes de que el usuario haga clic en el componente, se crea una instancia de DateChooser y después se oculta.

## Ejemplo

En el ejemplo siguiente se establece el valor `false` para la visibilidad del subcomponente DateChooser y, a continuación, se establece el tamaño de dicho subcomponente: 300 píxeles de altura y 300 píxeles de anchura.

```
myDF.pullDown._visible = false;  
myDF.pullDown.setSize(300,300);
```

## DateField.scroll

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

Sintaxis 1:

```
on(scroll){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.scroll = function(eventObject){  
    ...  
}  
myDF.addEventListener("scroll", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando se presiona un botón de mes.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente DateField. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el campo de fecha `myDF`, envía “\_level0.myDF” al panel Salida:

```
on(scroll){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*myDF*) distribuye un evento (en este caso, *scroll*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto Event del evento scroll tiene una propiedad adicional, *detail*, que puede tener uno de los valores siguientes: *nextMonth*, *previousMonth*, *nextYear*, *previousYear*.

Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 602](#).

### Ejemplo

En este ejemplo, escrito en un fotograma de la línea de tiempo, se envía un mensaje al panel Salida cuando se pulsa un botón de mes en una instancia de DateField denominada *myDF*. La primera línea de código crea un objeto detector denominado *form*. La segunda línea define una función para el evento *scroll* en el objeto detector. Dentro de la función hay una acción *trace* que utiliza el objeto Event pasado automáticamente a la función, en este ejemplo *eventObj*, para generar un mensaje. La propiedad *target* de un objeto Event es el componente que ha generado el evento, en este ejemplo *myDF*. La última línea llama al método `UIEventDispatcher.addEventListener()` desde *myDateField* y pasa como parámetros el evento *scroll* y el objeto detector *form*, como se indica a continuación:

```
form = new Object();
form.scroll = function(eventObj){
    trace(eventObj.detail);
}
myDF.addEventListener("scroll", form);
```

## DateField.selectableRange

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myDF*.selectableRange



## Descripción

Propiedad; establece una fecha única seleccionable o un rango de fechas seleccionables. El valor de esta propiedad es un objeto compuesto por dos objetos `Date` denominados `rangeStart` y `rangeEnd`. Las propiedades `rangeStart` y `rangeEnd` describen los límites del rango de fechas elegible. Si sólo se define `rangeStart`, se activarán todas las fechas posteriores a la señalada en `rangeStart`. Si sólo se define `rangeEnd`, se activarán todas las fechas anteriores a la señalada en `rangeEnd`. El valor predeterminado es `undefined`.

Si lo que se desea es activar un único día, puede utilizarse un único objeto `Date` como valor de `selectableRange`.

Al definir fechas deben especificarse de forma completa. Por ejemplo, `new Date(2003,6,24)` en lugar de `new Date()`. Si no se especifica una fecha completa, la hora devuelta es 00:00:00.

El valor de `DateField.selectedDate` queda establecido como no definido (`undefined`) si cae fuera del rango seleccionable.

Los valores de `DateField.displayedMonth` y `DateField.displayedYear` se establecen en el último mes más próximo del rango seleccionable si el mes actual cae fuera de dicho rango. Por ejemplo, si el mes que se muestra como actual es agosto (August), y el rango seleccionable comprende el periodo que va desde junio de 2003 a julio de 2003, el mes que se mostrará cambiará a julio de 2003 (July).

## Ejemplo

En el ejemplo siguiente se define como rango seleccionable las fechas comprendidas entre el 7 de mayo y el 7 de junio, ambas inclusive:

```
myDF.selectableRange = {rangeStart: new Date(2001, 4, 7), rangeEnd: new Date(2003, 5, 7)};
```

En el ejemplo siguiente se define como rango seleccionable el 7 de mayo y las fechas posteriores a ese día:

```
myDF.selectableRange = {rangeStart: new Date(2003, 4, 7)};
```

En el ejemplo siguiente se define como rango seleccionable el 7 de junio y las fechas anteriores a ese día:

```
myDF.selectableRange = {rangeEnd: new Date(2003, 5, 7)};
```

En el ejemplo siguiente se define el 7 de junio como única fecha seleccionable:

```
myDF.selectableRange = new Date(2003, 5, 7);
```

## DateField.selectedDate

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDF.selectedDate
```

### Descripción

Propiedad; objeto Date que indica la fecha seleccionada si el valor se encuentra dentro del valor especificado en la propiedad `selectableRange`. El valor predeterminado es `undefined`.

### Ejemplo

En el ejemplo siguiente se define el 7 de junio como fecha seleccionada:

```
myDF.selectedDate = new Date(2003, 5, 7);
```

## DateField.showToday

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myDF.showToday
```

### Descripción

Propiedad; esta propiedad determina si debe resaltarse o no la fecha actual. El valor predeterminado es `true`.

### Ejemplo

En el ejemplo siguiente se desactiva el resaltado de la fecha correspondiente al día de hoy:

```
myDF.showToday = false;
```

## Clase DepthManager

**Nombre de clase de ActionScript** `mx.managers.DepthManager`

La clase `DepthManager` añade funcionalidad a la clase `MovieClip` de ActionScript, la cual permite gestionar las asignaciones de profundidad de cualquier componente o clip de película, incluido `_root`. También permite gestionar profundidades reservadas en un clip especial de profundidad máxima situado en `_root` de servicios del sistema, como el cursor o las sugerencias.

Los métodos siguientes componen la interfaz API de clasificación de profundidad relativa:

- `DepthManager.createChildAtDepth()`
- `DepthManager.createClassChildAtDepth()`
- `DepthManager.setDepthAbove()`
- `DepthManager.setDepthBelow()`
- `DepthManager.setDepthTo()`

Los métodos siguientes componen la interfaz API de espacio de profundidad reservada:

- `DepthManager.createClassObjectAtDepth()`
- `DepthManager.createObjectAtDepth()`

## Resumen de métodos de la clase DepthManager

| Método                                               | Descripción                                                                                                           |
|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| <code>DepthManager.createChildAtDepth()</code>       | Crea un elemento secundario del símbolo especificado en la profundidad especificada.                                  |
| <code>DepthManager.createClassChildAtDepth()</code>  | Crea un objeto de la clase especificada en la profundidad especificada.                                               |
| <code>DepthManager.createClassObjectAtDepth()</code> | Crea una instancia de la clase especificada en la profundidad especificada en el clip especial de profundidad máxima. |
| <code>DepthManager.createObjectAtDepth()</code>      | Crea un objeto en la profundidad especificada en el clip de profundidad máxima.                                       |
| <code>DepthManager.setDepthAbove()</code>            | Establece la profundidad por encima de la instancia especificada.                                                     |
| <code>DepthManager.setDepthBelow()</code>            | Establece la profundidad por debajo de la instancia especificada.                                                     |
| <code>DepthManager.setDepthTo()</code>               | Establece la profundidad en la instancia especificada en el clip de profundidad máxima.                               |

### DepthManager.createChildAtDepth()

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.

#### Sintaxis

```
movieClipInstance.createChildAtDepth(linkageName, depthFlag[, initObj])
```

#### Parámetros

*linkageName* Identificador de vinculación. Este parámetro es una cadena.

*depthFlag* Uno de los valores siguientes: `DepthManager.kTop`, `DepthManager.kBottom`, `DepthManager.kTopmost`, `DepthManager.kNotopmost`. Todas las etiquetas de profundidad son propiedades estáticas de la clase `DepthManager`. Debe hacer referencia al paquete `DepthManager` (por ejemplo, `mx.managers.DepthManager.kTopmost`), o utilizar la sentencia `import` para importar el paquete `DepthManager`.

*initObj* Objeto de inicialización. Este parámetro es opcional.

#### Valor devuelto

Referencia al objeto creado.

#### Descripción

Método; crea una instancia secundaria del símbolo que especifica el parámetro *linkageName* en la profundidad que especifica el parámetro *depthFlag*.

## Ejemplo

El siguiente ejemplo crea una instancia `minuteHand` del clip de película `MinuteSymbol` y la coloca encima del reloj:

```
import mx.managers.DepthManager;
minuteHand = clock.createClassChildAtDepth("MinuteSymbol", DepthManager.kTop);
```

## DepthManager.createClassChildAtDepth()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
movieClipInstance.createClassChildAtDepth( className, depthFlag[, initObj] )
```

### Parámetros

*className* Nombre de clase.

*depthFlag* Uno de los valores siguientes: `DepthManager.kTop`, `DepthManager.kBottom`, `DepthManager.kTopmost`, `DepthManager.kNotopmost`. Todas las etiquetas de profundidad son propiedades estáticas de la clase `DepthManager`. Debe hacer referencia al paquete `DepthManager` (por ejemplo, `mx.managers.DepthManager.kTopmost`), o utilizar la sentencia `import` para importar el paquete `DepthManager`.

*initObj* Objeto de inicialización. Este parámetro es opcional.

### Valor devuelto

Referencia al elemento secundario creado.

### Descripción

Método; crea un elemento secundario de la clase que especifica el parámetro *className* en la profundidad que especifica el parámetro *depthFlag*.

## Ejemplo

El siguiente código dibuja un rectángulo de selección encima de todos los objetos `NoTopmost`:

```
import mx.managers.DepthManager
this.ring = createClassChildAtDepth(mx.skins.RectBorder, DepthManager.kTop);
```

El siguiente código crea una instancia de la clase `Button` y le atribuye un valor a su propiedad `label` como parámetro *initObj*:

```
import mx.managers.DepthManager
button1 = createClassChildAtDepth(mx.controls.Button, DepthManager.kTop,
    {label: "Top Button"});
```

## DepthManager.createClassObjectAtDepth()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
DepthManager.createClassObjectAtDepth(className, depthSpace[, initObj])
```

### Parámetros

*className* Nombre de clase.

*depthSpace* Uno de los valores siguientes: `DepthManager.kCursor`, `DepthManager.kTooltip`. Todas las etiquetas de profundidad son propiedades estáticas de la clase `DepthManager`. Debe hacer referencia al paquete `DepthManager` (por ejemplo, `mx.managers.DepthManager.kCursor`), o utilizar la sentencia `import` para importar el paquete `DepthManager`.

*initObj* Objeto de inicialización. Este parámetro es opcional.

### Valor devuelto

Referencia al objeto creado.

### Descripción

Método; crea un objeto de la clase que especifica el parámetro *className* en la profundidad que especifica el parámetro *depthSpace*. Este método se utiliza para acceder a los espacios de profundidad reservada en el clip de profundidad máxima.

### Ejemplo

En el ejemplo siguiente se crea un objeto de la clase `Button`:

```
import mx.managers.DepthManager
myCursorButton = createClassObjectAtDepth(mx.controls.Button,
    DepthManager.kCursor, {label: "Cursor"});
```

## DepthManager.createObjectAtDepth()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
DepthManager.createObjectAtDepth(linkageName, depthSpace[, initObj])
```

## Parámetros

*linkageName* Identificador de vinculación.

*depthSpace* Uno de los valores siguientes: `DepthManager.kCursor`, `DepthManager.kTooltip`. Todas las etiquetas de profundidad son propiedades estáticas de la clase `DepthManager`. Debe hacer referencia al paquete `DepthManager` (por ejemplo, `mx.managers.DepthManager.kCursor`), o utilizar la sentencia `import` para importar el paquete `DepthManager`.

*initObj* Objeto de inicialización.

## Valor devuelto

Referencia al objeto creado.

## Descripción

Método; crea un objeto en la profundidad especificada. Este método se utiliza para acceder a los espacios de profundidad reservada en el clip de profundidad máxima.

## Ejemplo

En el ejemplo siguiente se crea una instancia del símbolo `TooltipSymbol` y se coloca en la profundidad reservada para las sugerencias:

```
import mx.managers.DepthManager
myCursorTooltip = createObjectAtDepth("TooltipSymbol", DepthManager.kTooltip);
```

## DepthManager.setDepthAbove()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
movieClipInstance.setDepthAbove(instance)
```

## Parámetros

*instance* Nombre de instancia.

## Valor devuelto

Ninguno.

## Descripción

Método; establece la profundidad de un clip de película o una instancia de componente por encima de la profundidad de la instancia que especifica el parámetro *instance*.

## DepthManager.setDepthBelow()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
movieClipInstance.setDepthBelow(instance)
```

### Parámetros

*instance* Nombre de instancia.

### Valor devuelto

Ninguno.

### Descripción

Método; establece la profundidad de un clip de película o una instancia de componente por debajo de la profundidad de la instancia que especifica el parámetro *instance*.

### Ejemplo

El código siguiente establece la profundidad de la instancia `textInput` por debajo de la profundidad de `button`:

```
textInput.setDepthBelow(button);
```

## DepthManager.setDepthTo()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
movieClipInstance.setDepthTo(depth)
```

### Parámetros

*depth* Nivel de profundidad.

### Valor devuelto

Ninguno.

### Descripción

Método; establece la profundidad de *movieClipInstance* en el valor que especifica *depth*. Este método desplaza una instancia a otra profundidad para dar cabida a otro objeto.

## Ejemplo

En el ejemplo siguiente se define la profundidad de la instancia `mc1` en 10:

```
mc1.setDepthTo(10);
```

Para más información sobre la profundidad y el orden de apilamiento, consulte “Determinación del siguiente valor superior de profundidad disponible” en el apartado Guía de referencia de ActionScript de la Ayuda.

## Clase FocusManager

`FocusManager` se utiliza para especificar el orden en el que se seleccionan los componentes cuando el usuario presiona el tabulador para desplazarse por una aplicación. Es posible emplear la interfaz API de `FocusManager` para definir un botón en el documento que reciba la entrada del teclado cuando el usuario presione Intro (Windows) o Retorno (Macintosh). Por ejemplo, cuando un usuario rellena un formulario, debe pasar de un campo a otro con el tabulador y presionar Intro (Windows) o Retorno (Macintosh) para enviar el formulario.

Todos los componentes implementan compatibilidad `FocusManager`; no es necesario escribir código alguno para invocarlo. `FocusManager` también interactúa con `System Manager`, que activa y desactiva instancias de `FocusManager` a medida que se activan y desactivan ventanas emergentes. Cada ventana modal tiene una instancia de `FocusManager`, por lo que los componentes de la ventana se adscriben a un conjunto de tabuladores propio, lo que impide que el usuario pase a componentes de otras ventanas al tabular.

`FocusManager` reconoce los grupos de botones de opción (los que tienen definida una propiedad `RadioButton.groupName`) y selecciona la instancia del grupo que tiene una propiedad `selected` definida en `true`. Cuando se presiona el tabulador, `FocusManager` comprueba si el objeto siguiente tiene el mismo `groupName` que el objeto activo. Si es así, desplaza automáticamente la selección al objeto siguiente que tenga un `groupName` diferente. También pueden utilizar esta función los grupos de componentes que admitan la propiedad `groupName`.

`FocusManager` gestiona los cambios de selección producidos con clics del ratón. Si el usuario hace clic en un componente, el componente se selecciona.

`FocusManager` no selecciona automáticamente ningún componente de una aplicación. Ni en la ventana principal ni en ninguna ventana emergente habrá seleccionado ningún componente de forma predeterminada a menos que efectúe una llamada a `FocusManager.setFocus()` para un componente.

## Utilización de FocusManager

`FocusManager` no selecciona automáticamente ningún componente. Debe escribir un script que efectúe una llamada a `FocusManager.setFocus()` para un componente si desea que dicho componente esté seleccionado cuando se cargue una aplicación.

Para que la selección pueda desplazarse por la aplicación, defina la propiedad `tabIndex` en los objetos que deban seleccionarse (botones incluidos). Cuando el usuario presione el tabulador, `FocusManager` buscará el objeto activado cuya propiedad `tabIndex` sea superior al valor actual de `tabIndex`. Cuando `FocusManager` alcance la propiedad `tabIndex` más alta, volverá a cero. De este modo, en el ejemplo siguiente el objeto `comment` (probablemente un componente `TextArea`) es el primero en seleccionarse y después se selecciona el objeto `okButton`:

```
comment.tabIndex = 1;  
okButton.tabIndex = 2;
```



También se puede utilizar el panel Accesibilidad para asignar un valor de índice de tabulación.

Si ningún elemento del escenario tiene un valor de índice de tabulación, FocusManager utiliza el orden z. El orden z se configura principalmente con el orden en el que los componentes se arrastran al escenario. Sin embargo, también se pueden utilizar los comandos Modificar/Organizar/Traer al frente/Atrás para determinar el orden z final.

Para crear un botón que se seleccione cuando el usuario presione la tecla Intro (Windows) o Retorno (Macintosh), defina la propiedad `FocusManager.defaultPushButton` en el nombre de instancia del botón deseado, como en el ejemplo siguiente:

```
focusManager.defaultPushButton = okButton;
```

**Nota:** FocusManager detecta cuándo se colocan los objetos en el escenario (el orden de profundidad de los objetos) pero no sus posiciones relativas en el mismo. Es diferente al modo en el que Flash Player gestiona la tabulación.

## Parámetros de FocusManager

No hay parámetros de edición para FocusManager. Es preciso utilizar los métodos y propiedades de ActionScript de la clase FocusManager en el panel Acciones. Para más información, consulte la [Clase FocusManager](#).

## Creación de aplicaciones con FocusManager

El siguiente procedimiento crea un plan de selección en una aplicación de Flash.

- 1 Arrastre el componente TextInput desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, asígnele el nombre de instancia **comment**.
- 3 Arrastre el componente Button desde el panel Componentes al escenario.
- 4 En el inspector de propiedades, asígnele el nombre de instancia **okButton** y defina el parámetro label en **OK**.
- 5 En el fotograma 1 del panel Acciones, introduzca lo siguiente:

```
comment.tabIndex = 1;
okButton.tabIndex = 2;
focusManager.setFocus(comment);
focusManager.defaultPushButton = okButton;
lo = new Object();
lo.click = function(evt){
    trace(evt.target + " was clicked");
}
okButton.addEventListener("click", lo);
```

Este código establece el orden de tabulación y especifica el botón predeterminado que recibirá el evento `click` cuando el usuario presione Intro (Windows) o Retorno (Macintosh).

## Personalización de FocusManager

Se puede cambiar el color del anillo de selección en el tema Halo cambiando el valor del estilo `themeColor`.

FocusManager utiliza un aspecto FocusRect para seleccionar. Este aspecto puede reemplazarse o modificarse, y es posible sustituir `UIComponent.drawFocus` con subclases para emplear indicadores de selección personalizados.

## Clase FocusManager

**Herencia** UIObject > UIComponent > FocusManager

**Nombre de clase de ActionScript** mx.managers.FocusManager

### Resumen de métodos de la clase FocusManager

| Método                                                 | Descripción                                                                                                    |
|--------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>FocusManager.setFocus()</code>                   | Devuelve una referencia al objeto seleccionado.                                                                |
| <code>FocusManager.sendDefaultPushButtonEvent()</code> | Envía un evento <code>click</code> a los objetos detectores registrados en el botón de comando predeterminado. |
| <code>FocusManager.setFocus()</code>                   | Selecciona el objeto especificado.                                                                             |

### Resumen de propiedades de la clase FocusManager

| Método                                             | Descripción                                                                                                                                                                                          |
|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>FocusManager.defaultPushButton</code>        | Objeto que recibe el evento <code>click</code> cuando el usuario presiona la tecla Intro (Windows) o Retorno (Macintosh).                                                                            |
| <code>FocusManager.defaultPushButtonEnabled</code> | Indica si la gestión con el teclado está activada ( <code>true</code> ) o desactivada ( <code>false</code> ) para el botón de comando predeterminado. El valor predeterminado es <code>true</code> . |
| <code>FocusManager.enabled</code>                  | Indica si la gestión con el tabulador está activada ( <code>true</code> ) o desactivada ( <code>false</code> ). El valor predeterminado es <code>true</code> .                                       |
| <code>FocusManager.nextTabIndex</code>             | Valor siguiente de la propiedad <code>tabIndex</code> .                                                                                                                                              |

## FocusManager.defaultPushButton

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004.

### Sintaxis

`focusManager.defaultPushButton`

## Descripción

Propiedad; especifica el botón de comando predeterminado para una aplicación. Cuando el usuario presiona la tecla Intro (Windows) o Retorno (Macintosh), los detectores del botón de comando predeterminado reciben un evento `click`. El valor predeterminado es `undefined` y el tipo de datos de esta propiedad es objeto.

FocusManager utiliza la declaración de estilo `emphasized` de la clase `SimpleButton` para indicar visualmente el botón de comando predeterminado actual.

El valor de la propiedad `defaultPushButton` es siempre el botón seleccionado. El botón de comando predeterminado no se selecciona inicialmente por haber definido la propiedad `defaultPushButton`. Si hay varios botones en una aplicación, el botón seleccionado actualmente recibe el evento `click` cuando se presiona la tecla Intro o Retorno. Si hay algún otro componente seleccionado cuando se presiona Intro o Retorno, se restablece el valor original de la propiedad `defaultPushButton`.

## Ejemplo

El código siguiente define el botón de comando predeterminado en la instancia `OKButton`:

```
FocusManager.defaultPushButton = OKButton;
```

## Véase también

```
FocusManager.defaultPushButtonEnabled,  
FocusManager.sendDefaultPushButtonEvent()
```

## FocusManager.defaultPushButtonEnabled

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
focusManager.defaultPushButtonEnabled
```

## Descripción

Propiedad; valor booleano que determina si la gestión con el teclado está activada (`true`) o no (`false`) para el botón de comando predeterminado. Si `defaultPushButtonEnabled` se define con el valor `false`, los componentes pueden recibir la entrada de las teclas Retorno o Intro y gestionarla internamente. Deberá volver a activar la gestión con el botón de comando predeterminado observando el método `onKillFocus()` del componente (véase `MovieClip.onKillFocus` en el apartado Diccionario de ActionScript de la Ayuda) o el evento `focusOut`. El valor predeterminado es `true`.

## Ejemplo

El código siguiente desactiva la gestión con el botón de comando predeterminado:

```
focusManager.defaultPushButtonEnabled = false;
```

## FocusManager.enabled

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
focusManager.enabled
```

### Descripción

Propiedad; Valor booleano que determina si una gestión con el tabulador está activada (`true`) o no (`false`) para un grupo determinado de objetos de selección. (Por ejemplo, otra ventana emergente podría tener su propio FocusManager.) Si `enabled` se define en `false`, los componentes pueden recibir la entrada del tabulador y gestionarla internamente. Deberá volver a activar la gestión de FocusManager observando el método `onKillFocus()` del componente (véase `MovieClip.onKillFocus` en el apartado Diccionario de ActionScript de la Ayuda) o el evento `focusOut`. El valor predeterminado es `true`.

### Ejemplo

El código siguiente desactiva la tabulación:

```
focusManager.enabled = false;
```

## FocusManager.getFocus()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
focusManager.getFocus()
```

### Parámetros

Ninguno.

### Valor devuelto

Referencia al objeto seleccionado.

### Descripción

Método; devuelve una referencia al objeto seleccionado actualmente.

### Ejemplo

El código siguiente selecciona `myOKButton` si el objeto actualmente seleccionado es `myInputText`:

```
if (focusManager.getFocus() == myInputText)
{
    focusManager.setFocus(myOKButton);
}
```

### Véase también

[FocusManager.setFocus\(\)](#)

## FocusManager.nextTabIndex

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

`FocusManager.nextTabIndex`

### Descripción

Propiedad; siguiente número de índice de tabulador disponible. Utilice esta propiedad para definir dinámicamente la propiedad `tabIndex` de un objeto.

### Ejemplo

El código siguiente asigna a la instancia `mycheckbox` el siguiente valor `tabIndex` más alto:

```
mycheckbox.tabIndex = focusManager.nextTabIndex;
```

### Véase también

[UIComponent.tabIndex](#)

## FocusManager.sendDefaultPushButtonEvent()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

`focusManager.sendDefaultPushButtonEvent()`

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

## Descripción

Método; envía un evento `click` a los objetos detectores registrados en el botón de comando predeterminado. Utilice este método para enviar un evento `click` mediante programación.

## Ejemplo

El código siguiente activa el evento `click` del botón de comando predeterminado y rellena los campos de nombre de usuario y contraseña cuando un usuario selecciona la instancia de `CheckBox` `chb` (la casilla de verificación debe tener la etiqueta “Conexión automática”):

```
name_txt.tabIndex = 1;
password_txt.tabIndex = 2;
chb.tabIndex = 3;
submit_ib.tabIndex = 4;

focusManager.defaultPushButton = submit_ib;

chbObj = new Object();
chbObj.click = function(o){
    if (chb.selected == true){
        name_txt.text = "Jody";
        password_txt.text = "foobar";
        focusManager.sendDefaultPushButtonEvent();
    } else {
        name_txt.text = "";
        password_txt.text = "";
    }
}
chb.addEventListener("click", chbObj);

submitObj = new Object();
submitObj.click = function(o){
    if (password_txt.text != "foobar"){
        trace("error on submit");
    } else {
        trace("Yeah! sendDefaultPushButtonEvent worked!");
    }
}
submit_ib.addEventListener("click", submitObj);
```

## Véase también

[FocusManager.defaultPushButton](#), [FocusManager.sendDefaultPushButtonEvent\(\)](#)

## FocusManager.setFocus()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
focusManager.setFocus(object)
```

### Parámetros

*object* Referencia al objeto que debe seleccionarse.

## Valor devuelto

Ninguno.

## Descripción

Método; selecciona el objeto especificado.

## Ejemplo

El código siguiente selecciona myOKButton:

```
focusManager.setFocus(myOKButton);
```

## Véase también

[FocusManager.setFocus\(\)](#)

# Clase Form (sólo para Flash Professional)

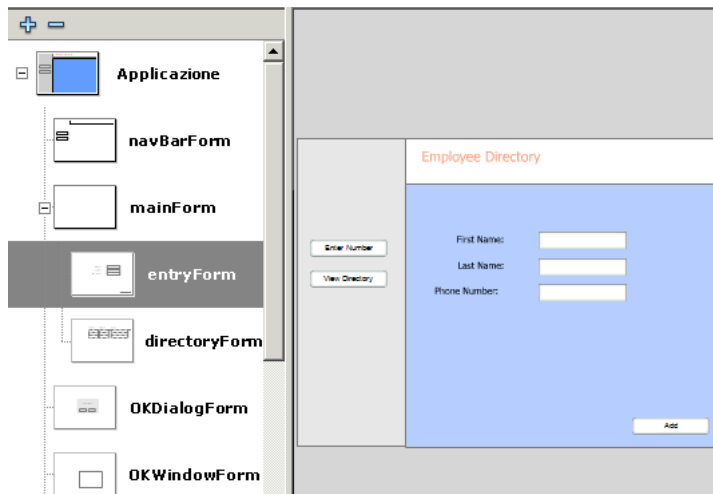
**Herencia** UIObject > UIComponent > View > Loader > Screen > Form

**Nombre de clase de ActionScript** mx.screens.Form

La clase Form proporciona el comportamiento en tiempo de ejecución de los formularios creados por el usuario en el panel Contorno de pantalla en Flash MX Professional 2004. Para ver información general sobre cómo trabajar con las pantallas, consulte “Información general sobre el trabajo con pantallas (sólo en Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

## Utilización de la clase Form (sólo para Flash Professional)

Los formularios hacen la función de contenedores para objetos gráficos (elementos de interfaz de usuario de una aplicación, por ejemplo) y de estados de aplicación. En el panel Contorno de pantalla puede visualizar los diferentes estados de la aplicación que está creando, donde cada formulario es un estado de aplicación diferente. Por ejemplo, en la figura siguiente se muestra el panel Contorno de pantalla de una aplicación de ejemplo que se ha diseñado con formularios.



*Vista de contorno de pantalla de una aplicación de formulario de ejemplo*

En la figura se muestra el contorno de una aplicación de ejemplo de directorio de empleados denominada “Employee Directory”, compuesta por varios formularios. El formulario denominado “entryForm” (que aparece seleccionado en la figura anterior) contiene varios objetos de interfaz de usuario, entre ellos campos de entrada de texto, etiquetas y un botón de comando. El desarrollador puede presentar fácilmente este formulario al usuario alternando su visibilidad (mediante la propiedad `Form.visible`), a la vez que alterna la visibilidad de otros formularios.

En el panel Comportamientos (Ventana > Paneles de desarrollo > Comportamientos) también puede asociar comportamientos y controles con los formularios. Para más información sobre cómo añadir transiciones y controles a las pantallas, consulte “Utilización de comportamientos para crear controles y transiciones para pantallas (sólo en Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

Puesto que la clase `Form` es una extensión de la clase `Loader`, puede cargar fácilmente contenido externo (un archivo SWF o un archivo JPEG) en un formulario. Por ejemplo, el contenido de un formulario puede ser un archivo SWF independiente, que puede contener formularios. De este modo, se pueden dividir en módulos las aplicaciones de formularios, lo cual facilita el mantenimiento de las aplicaciones y reduce el tiempo de descarga inicial. Para más información, consulte “Carga de contenido externo en pantallas (sólo para Flash Professional)” en la página 487.

## Parámetros del objeto form

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del objeto `form` en el inspector de propiedades o en el panel Inspector de componentes:

**autoload** Indica si el contenido especificado por el parámetro `contentPath` debe cargarse automáticamente (`true`) o si debe esperar hasta que se llame al método `Loader.load()` (`false`). El valor predeterminado es `true`.

**contentPath** Especifica el contenido del formulario. Este valor puede ser el identificador de vinculación de un clip de película o la URL absoluta o relativa de un archivo SWF o JPG que se carga en la diapositiva. De forma predeterminada, el contenido cargado se recorta para que quepa en la diapositiva.

**visible** Especifica si el formulario se ve (`true`) o está oculto (`false`) al cargarse la primera vez.

## Resumen de métodos de la clase Form

| Método                           | Descripción                                                  |
|----------------------------------|--------------------------------------------------------------|
| <code>Form.getChildForm()</code> | Devuelve el formulario secundario en un índice especificado. |

Hereda todos los métodos de `UIObject`, `UIComponent`, `View`, `Componente Loader` y `Clase Screen` (sólo para Flash Professional).



## Resumen de propiedades de la clase Form

| Propiedad                            | Descripción                                                                                                                                 |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Form.currentFocusedForm</code> | Devuelve el último formulario hoja que contiene la selección actual global.                                                                 |
| <code>Form.indexInParentForm</code>  | Devuelve el índice (basado en cero) de este formulario en la lista de subformularios de su formulario principal.                            |
| <code>Form.visible</code>            | Especifica si el formulario está visible cuando el formulario, la diapositiva, el clip de película o el archivo SWF principal está visible. |
| <code>Form.numChildForms</code>      | Devuelve el número de formularios secundarios que contiene este formulario.                                                                 |
| <code>Form.parentIsForm</code>       | Indica si el objeto principal de este formulario es también un formulario o si no lo es.                                                    |
| <code>Form.rootForm</code>           | Devuelve la raíz del árbol de formularios, o subárbol, que contiene el formulario.                                                          |

Hereda todas las propiedades de [UIObject](#), [UIComponent](#), [View](#), [Componente Loader](#) y [Clase Screen](#) (sólo para Flash Professional).

### Form.currentFocusedForm

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

```
mx.screens.Form.currentFocusedForm
```

#### Descripción

Propiedad (sólo lectura); devuelve el objeto form que contiene la selección actual global. La selección actual puede estar en el propio formulario o en un clip de película, objeto de texto o componente que se halle dentro de dicho formulario. Puede devolver el valor null en caso de que no haya selección actual.

#### Ejemplo

El código siguiente, asociado con un botón (que no se muestra), muestra el nombre del formulario que contiene la selección actual.

```
trace("The form with the current focus is: " +  
    mx.screens.Form.currentFocusedForm);
```

## Form.getChildForm()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myForm.getChildForm(childIndex)
```

### Parámetros

*childIndex*    Número que indica el índice (basado en cero) del formulario secundario para devolver.

### Valor devuelto

Un objeto Form.

### Descripción

Método; devuelve el formulario secundario de *myForm* cuyo índice es *childIndex*.

### Ejemplo

En el ejemplo siguiente se muestra un panel Salida con los nombres de todos los objetos form secundarios que pertenecen al objeto Form raíz denominado Application.

```
for (var i:Number = 0; i < _root.Application.numChildForms; i++) {  
    var childForm:mx.screens.Form = _root.Application.getChildForm(i);  
    trace(childForm._name);  
}
```

### Véase también

[Form.numChildForms](#)

## Form.indexInParentForm

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myForm.indexInParentForm
```

### Descripción

Propiedad (sólo lectura); contiene el índice (basado en cero) de *myForm* en la lista de formularios secundarios de su formulario principal. Si el objeto principal de *myForm* es una pantalla en lugar de un formulario (por ejemplo, una diapositiva), *indexInParentForm* es siempre 0.

### Ejemplo

```
var myIndex:Number = myForm.indexInParent;
if (myForm == myForm._parent.getChildForm(myIndex)) {
    trace("I'm where I should be");
}
```

### Véase también

[Form.getChildForm\(\)](#)

## Form.numChildForms

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myForm.numChildForms*

### Descripción

Propiedad (sólo lectura); devuelve el número de formularios secundarios que contiene *myForm*. Esta propiedad no incluye las diapositivas que contiene *myForm*, únicamente los formularios.

### Ejemplo

El código siguiente se repite en todos los formularios secundarios que contiene *myForm* y muestra sus nombres en el panel Salida.

```
var howManyKids:Number = myForm.numChildForms;
for(i=0; i<howManyKids; i++) {
    var childForm = myForm.getChildForm(i);
    trace(childForm._name);
}
```

### Véase también

[Form.getChildForm\(\)](#)

## Form.parentIsForm

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myForm.parentIsForm*

## Descripción

Propiedad (sólo lectura): devuelve un valor booleano (`true` o `false`) que indica si el objeto principal del formulario especificado también es un formulario (`true`) o si no lo es (`false`). Si se devuelve el valor `false`, *myForm* es el objeto raíz de su jerarquía de formularios.

## Ejemplo

```
if (myForm.parentIsForm) {  
    trace("I have "+myForm._parent.numChildScreens+" sibling screens");  
} else {  
    trace("I am the root form and have no siblings");  
}
```

## Form.rootForm

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myForm.rootForm*

## Descripción

Propiedad (sólo lectura); devuelve el formulario superior de la jerarquía de formularios que contiene *myForm*. Si *myForm* se encuentra en un objeto que no es un formulario (esto es, una diapositiva), la propiedad devuelve *myForm*.

## Ejemplo

En el ejemplo siguiente, se coloca una referencia al formulario raíz de *myForm* en una variable denominada *root*. Si el valor asignado a *root* hace referencia a *myForm*, *myForm* se encuentra en la parte superior de su árbol de formularios.

```
var root:my.screens.Form = myForm.rootForm;  
if(rootForm == myForm) {  
    trace("myForm is the top form in its tree");  
}
```

## Form.visible

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myForm.visible*

## Descripción

Propiedad; determina si *myForm* está visible cuando su formulario, diapositiva, clip de película o película principal esté visible. También puede establecer esta propiedad mediante el inspector de propiedades del entorno de edición de Flash.

Si esta propiedad tiene el valor *true*, *myForm* recibe un evento *reveal*; si tiene el valor *false*, *myForm* recibe un evento *hide*. Puede asociar transiciones con los formularios para que se ejecuten cuando un formulario reciba uno de estos eventos. Para más información sobre cómo añadir transiciones a las pantallas, consulte “Utilización de comportamientos para crear controles y transiciones para pantallas (sólo en Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

## Ejemplo

El código siguiente, asociado con una instancia del componente Button, establece el valor *false* para la propiedad *visible* del formulario que contiene el botón.

```
on(click){  
    _parent.visible = true;  
}
```

## Componente Label

Un componente Label es una línea de texto. Es posible especificar que el componente Label se formatee con HTML. También se puede controlar la alineación y tamaño de una etiqueta. Los componentes Label no tienen bordes, no se pueden seleccionar y no difunden eventos.

La vista previa dinámica de cada instancia de Label refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes. Label no tiene bordes, por lo que el único modo de verlo de forma previa dinámica es definir su parámetro *text*. El parámetro *autoSize* no se admite en la vista previa dinámica.

Cuando se añade el componente Label a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.LabelAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda.

## Utilización del componente Label

Utilice el componente Label para crear etiquetas de texto para otros componentes de un formulario, como la etiqueta “Nombre:” situada a la izquierda de un campo *TextInput* que acepta el nombre del usuario. Si está creando una aplicación con componentes basados en la versión 2 (v2) de la arquitectura de componentes de Macromedia, es recomendable utilizar un componente Label en lugar de un campo de texto normal, ya que así podrá utilizar estilos para lograr una apariencia uniforme.

## Parámetros de Label

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Label en el inspector de propiedades o en el panel Inspector de componentes:

**text** indica el texto de la etiqueta; el valor predeterminado es Label.

**html** indica si la etiqueta va a formatearse con HTML (`true`) o no (`false`). Si el parámetro `html` se define en `true`, el componente Label no podrá formatearse con estilos. El valor predeterminado es `false`.

**autoSize** indica cómo se cambia de tamaño y se alinea la etiqueta para que encaje el texto. El valor predeterminado es `none`. El parámetro puede tener cualquiera de estos cuatro valores:

- `none`: la etiqueta no cambia de tamaño ni se alinea para que encaje el texto.
- `left`: los lados derecho e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados izquierdo y superior no cambian de tamaño.
- `center`: el lado inferior de la etiqueta cambia de tamaño para que encaje el texto. El centro horizontal de la etiqueta permanece fijo en su posición central horizontal original.
- `right`: los lados izquierdo e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados superior y derecho no cambian de tamaño.

**Nota:** la propiedad `autoSize` del componente Label es diferente de la propiedad `autoSize` del objeto TextField incorporado de ActionScript.

Puede escribir código ActionScript para definir opciones adicionales para instancias de Label con sus métodos, propiedades y eventos. Para más información, consulte [Clase Label](#).

## Creación de aplicaciones con el componente Label

El siguiente procedimiento explica cómo añadir un componente Label a una aplicación durante la edición. En este ejemplo, la etiqueta se encuentra junto a un cuadro combinado de fechas en una aplicación de carrito de la compra.

**Para crear una aplicación con el componente Label, siga este procedimiento:**

- 1 Arrastre un componente Label desde el panel Componentes al escenario.
- 2 En el panel Inspector de componentes, introduzca **Expiration Date** en el parámetro `label`.

## Personalización del componente Label

El componente Label puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. También es posible definir el parámetro de edición `autoSize`; en tal caso, el recuadro de delimitación de la vista previa dinámica no varía, pero la etiqueta sí cambia de tamaño. Para más información, consulte [“Parámetros de Label” en la página 306](#). En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()` o `Label.autoSize`).

## Utilización de estilos con el componente Label

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de Label. Todo el texto de la instancia del componente Label debe compartir el mismo estilo. Por ejemplo, no es posible definir el estilo `color` en "blue" para una palabra de la etiqueta y en "red" para la segunda palabra de la misma etiqueta.

Si el nombre de una propiedad de estilo termina por "Color", significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son.

Para más información sobre estilos, consulte ["Utilización de estilos para personalizar el texto y el color de un componente" en la página 31](#).

Un componente Label admite los siguientes estilos:

| Estilo                      | Descripción                                                 |
|-----------------------------|-------------------------------------------------------------|
| <code>color</code>          | Color predeterminado del texto.                             |
| <code>embedFonts</code>     | Fuentes que incorporar al documento.                        |
| <code>fontFamily</code>     | Nombre de la fuente del texto.                              |
| <code>fontSize</code>       | Tamaño de la fuente en puntos.                              |
| <code>fontStyle</code>      | Estilo de la fuente; puede ser "normal" o "italic".         |
| <code>fontWeight</code>     | Grosor de la fuente; puede ser "normal" o "bold".           |
| <code>textAlign</code>      | Alineación del texto; puede ser "left", "right" o "center". |
| <code>textDecoration</code> | Decoración del texto; puede ser "none" o "underline".       |

## Utilización de aspectos con el componente Label

El componente Label no puede utilizar aspectos.

Para más información sobre asignación de aspectos a componentes, consulte ["Aplicación de aspectos a los componentes" en la página 41](#).

## Clase Label

**Herencia** UIObject > Label

**Nombre de clase de ActionScript** mx.controls.Label

Las propiedades de la clase Label permiten especificar en tiempo de ejecución el texto de la etiqueta, indicar si el texto se va a formatear con HTML e indicar si la etiqueta cambia de tamaño automáticamente para que encaje el texto.

Si una propiedad de la clase Label se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.Label.version);
```

**Nota:** el código siguiente devuelve undefined: `trace(myLabelInstance.version);`.

## Resumen de métodos de la clase Label

Hereda todos los métodos de [UIObject](#).

## Resumen de propiedades de la clase Label

| Propiedad                      | Descripción                                                                                                                                                                                                                       |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Label.autoSize</a> | Cadena que indica cómo una etiqueta cambia de tamaño y se alinea para que encaje el valor de su propiedad <code>text</code> . Hay cuatro valores posibles: "none", "left", "center" y "right". El valor predeterminado es "none". |
| <a href="#">Label.html</a>     | Valor booleano que indica si la etiqueta puede formatearse con HTML ( <code>true</code> ) o no ( <code>false</code> ).                                                                                                            |
| <a href="#">Label.text</a>     | Texto de la etiqueta.                                                                                                                                                                                                             |

Hereda todas las propiedades de [UIObject](#).

## Resumen de eventos de la clase Label

Hereda todos los eventos de [UIObject](#).

### Label.autoSize

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.

#### Sintaxis

*labelInstance.autoSize*

#### Descripción

Propiedad; cadena que indica cómo una etiqueta cambia de tamaño y se alinea para que encaje el valor de su propiedad `text`. Hay cuatro valores posibles: "none", "left", "center" y "right". El valor predeterminado es "none".

- `none`: la etiqueta no cambia de tamaño ni se alinea para que encaje el texto.
- `left`: los lados derecho e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados izquierdo y superior no cambian de tamaño.
- `center`: el lado inferior de la etiqueta cambia de tamaño para que encaje el texto. El centro horizontal de la etiqueta permanece fijo en su posición central horizontal original.
- `right`: los lados izquierdo e inferior de la etiqueta cambian de tamaño para que encaje el texto. Los lados superior y derecho no cambian de tamaño.

**Nota:** la propiedad `autoSize` del componente `Label` es diferente de la propiedad `autoSize` del objeto `TextField` incorporado de `ActionScript`.



## Label.html

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*labelInstance.html*

### Descripción

Propiedad; valor booleano que indica si la etiqueta puede formatearse con HTML (*true*) o no (*false*). El valor predeterminado es *false*. Los componentes Label con la propiedad *html* definida en *true* no pueden formatearse con estilos.

Con el componente Label no puede utilizarse la etiqueta HTML `<font color>` aunque *Label.html* esté definido en *true*. Por ejemplo, en este caso el texto “Hello” aparece en negro y no en rojo, como haría si pudiera utilizarse `<font color>`:

```
lbl.html = true;  
lbl.text = "<font color=\"#FF0000\">Hello</font> World";
```

Para recuperar el texto normal de un texto con formato HTML, defina la propiedad *HTML* en *false* y, a continuación, acceda a la propiedad *text*. Esto eliminará el formato HTML. Tal vez desee copiar el texto de la etiqueta en un componente Label o TextArea de fuera de la pantalla antes de recuperar el texto normal.

### Ejemplo

En el ejemplo siguiente se define la propiedad *html* en *true* para que la etiqueta pueda formatearse con HTML. A continuación, la propiedad *text* se define en una cadena que incluye formato con HTML:

```
labelControl.html = true;  
labelControl.text = "The <b>Royal</b> Nonesuch";
```

La palabra “Royal” se muestra en negrita.

## Label.text

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*labelInstance.text*

### Descripción

Propiedad; texto de una etiqueta. El valor predeterminado es "Label".

## Ejemplo

El código siguiente define la propiedad `text` de la instancia `labelControl` de `Label` y envía el valor al panel Salida:

```
labelControl.text = "The Royal Nonesuch";  
trace(labelControl.text);
```

## Componente List

El componente `List` es un cuadro de lista de desplazamiento de selección única o múltiple. Una lista también puede mostrar gráficos, incluidos otros componentes. Los elementos que se visualizan en el componente `List` se añaden con el cuadro de diálogo `Valores` que aparece al hacer clic en los campos de parámetros `data` o `labels`. Para añadir elementos a la lista también pueden utilizarse los métodos `List.addItem()` y `List.addItemAt()`.

El componente `List` utiliza un índice basado en cero donde el elemento con el índice 0 aparece en primer lugar. Si añade, elimina o reemplaza elementos de lista mediante los métodos y propiedades de la clase `List`, es posible que tenga que especificar el índice del elemento de lista.

El componente `List` se selecciona cuando se hace clic en él o se presiona el tabulador hasta su posición y, a continuación, se pueden utilizar las siguientes teclas para controlarlo:

| Tecla                | Descripción                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------------------|
| Teclas alfanuméricas | Salta al siguiente elemento con <code>Key.getAscii()</code> como el primer carácter de su etiqueta. |
| Control              | Tecla conmutadora. Permite seleccionar y anular la selección de varios elementos no contiguos.      |
| Flecha abajo         | La selección desciende un elemento.                                                                 |
| Inicio               | La selección se desplaza al principio de la lista.                                                  |
| Av Pág               | La selección avanza una página.                                                                     |
| Re Pág               | La selección retrocede una página.                                                                  |
| Mayús                | Tecla de selección contigua. Permite seleccionar elementos contiguos.                               |
| Flecha arriba        | La selección asciende un elemento.                                                                  |

**Nota:** para las teclas `Av Pág` y `Re Pág`, una página está formada por los elementos que caben en pantalla menos uno. Por ejemplo, para avanzar por una lista desplegable que presenta las líneas de diez en diez, la pantalla muestra los elementos 0-9, 9-18, 18-27 y así sucesivamente, solapando un elemento por página.

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase `FocusManager`” en la página 292](#).

La vista previa dinámica de cada instancia del componente `List` refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel `Inspector de componentes`.

Cuando se añade el componente `List` a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.ListAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda.

## Utilización del componente List

Utilice una lista cuando el usuario deba elegir una o varias opciones entre muchas. Por ejemplo, un usuario visita un sitio Web de comercio electrónico y debe elegir el artículo que va a comprar. Hay 30 artículos; el usuario recorre la lista y hace clic en el que desea seleccionar.

También puede diseñar una lista que utilice clips de película personalizados como filas para poder mostrar más información al usuario. Por ejemplo, en una aplicación de correo electrónico, cada buzón podría ser un componente `List` y cada fila podría tener iconos para indicar la prioridad y el estado.

## Parámetros del componente List

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente `List` en el inspector de propiedades o en el panel Inspector de componentes:

**data** Matriz de valores que rellenan los datos de la lista. El valor predeterminado es `[]` (matriz vacía). No hay propiedad en tiempo de ejecución equivalente.

**labels** Matriz de valores de texto que rellenan los valores de etiqueta de la lista. El valor predeterminado es `[]` (matriz vacía). No hay propiedad en tiempo de ejecución equivalente.

**multipleSelection** Valor booleano que indica si pueden seleccionarse varios valores (`true`) o no (`false`). El valor predeterminado es `false`.

**rowHeight** indica la altura de cada fila, expresada en píxeles. El valor predeterminado es 20. Si se define una fuente, la altura de la fila no cambia.

Puede escribir código `ActionScript` para definir opciones adicionales para instancias de `List` con sus métodos, propiedades y eventos. Para más información, consulte [Clase List](#).

## Creación de aplicaciones con el componente List

El siguiente procedimiento explica cómo añadir un componente `List` a una aplicación durante la edición. En este ejemplo, la lista tiene tres elementos.

**Para añadir un componente List simple a una aplicación, siga este procedimiento:**

- 1 Arrastre un componente List desde el panel Componentes al escenario.
- 2 Seleccione la lista y elija Modificar > Transformar para cambiar su tamaño y adaptarlo a la aplicación.
- 3 En el inspector de propiedades, siga este procedimiento:
  - Introduzca el nombre de instancia **myList**.
  - Introduzca Item1, Item2 y Item3 como parámetro labels.
  - Introduzca item1.html, item2.html, item3.html como parámetro data.
- 4 Elija Control > Probar película para ver la lista con sus elementos.

En la aplicación podría utilizar los valores de la propiedad data para abrir archivos HTML.

El siguiente procedimiento explica cómo añadir un componente List a una aplicación durante la edición. En este ejemplo, la lista tiene tres elementos.

**Para añadir un componente List a una aplicación, siga este procedimiento:**

- 1 Arrastre un componente List desde el panel Componentes al escenario.
- 2 Seleccione la lista y elija Modificar > Transformar para cambiar su tamaño y adaptarlo a la aplicación.
- 3 En el panel Acciones, introduzca el nombre de instancia **myList**.
- 4 Seleccione el fotograma 1 de la línea de tiempo y, en el panel Acciones, introduzca lo siguiente:

```
myList.dataProvider = myDP;
```

Si ha definido un proveedor de datos denominado `myDP`, la lista se llenará de datos. Para más información sobre los proveedores de datos, consulte [List.dataProvider](#).

- 5 Elija Control > Probar película para ver la lista con sus elementos.

## Personalización del componente List

El componente List puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `List.setSize()` (véase [UIObject.setSize\(\)](#)).

Cuando se cambia el tamaño de una lista, las filas se contraen horizontalmente y recortan el texto que contienen. Verticalmente, la lista añade o elimina las filas necesarias. Las barras de desplazamiento se colocan automáticamente. Para más información sobre barras de desplazamiento, consulte [“Componente ScrollPane” en la página 499](#).

## Utilización de estilos con el componente List

Es posible definir propiedades de estilo para cambiar el aspecto de un componente List.

El componente List utiliza los siguientes estilos de Halo:

| Estilo                              | Descripción                                                                                                                                                                                         |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>alternatingRowColors</code>   | Especifica los colores de las filas en un patrón alterno. El valor puede ser una matriz de más de dos colores, por ejemplo, <code>0xFF00FF</code> , <code>0xCC6699</code> y <code>0x996699</code> . |
| <code>backgroundColor</code>        | Color de fondo de la lista. Este estilo se define en una declaración de estilo de clase, <code>ScrollSelectList</code> .                                                                            |
| <code>borderColor</code>            | Sección negra de un borde tridimensional o sección en color de un borde bidimensional.                                                                                                              |
| <code>borderStyle</code>            | Estilo del recuadro de delimitación. Los valores posibles son: “none”, “solid”, “inset” y “outset”. Este estilo se define en una declaración de estilo de clase, <code>ScrollSelectList</code> .    |
| <code>defaultIcon</code>            | Nombre del icono predeterminado que se utilizará en las filas de la lista. El valor predeterminado es <code>undefined</code> .                                                                      |
| <code>rollOverColor</code>          | Color de una fila por la que se desplaza el puntero.                                                                                                                                                |
| <code>selectionColor</code>         | Color de una fila seleccionada.                                                                                                                                                                     |
| <code>selectionEasing</code>        | Referencia a la ecuación de aceleración (función) utilizada para controlar la interpolación programada.                                                                                             |
| <code>disabledColor</code>          | Color desactivado para el texto.                                                                                                                                                                    |
| <code>textRollOverColor</code>      | Color del texto cuando el puntero se desplaza sobre él.                                                                                                                                             |
| <code>textSelectedColor</code>      | Color del texto cuando está seleccionado.                                                                                                                                                           |
| <code>selectionDisabledColor</code> | Color de una fila si se ha seleccionado y desactivado.                                                                                                                                              |
| <code>selectionDuration</code>      | Duración de las transiciones al seleccionar elementos.                                                                                                                                              |
| <code>useRollOver</code>            | Determina si el resaltado se activa cuando el puntero se desplaza sobre una fila.                                                                                                                   |

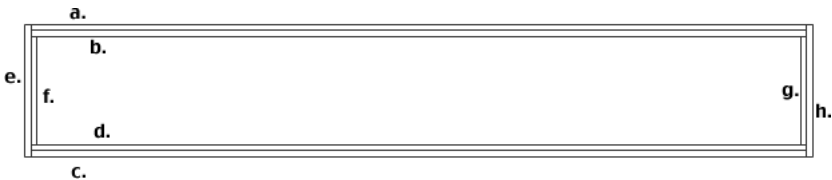
El componente List también utiliza las propiedades de estilo del componente Label (véase [“Utilización de estilos con el componente Label” en la página 307](#)), el componente ScrollPane (véase [“Componente ScrollPane” en la página 499](#)) y RectBorder.

## Utilización de aspectos con el componente List

Todos los aspectos del componente List están incluidos en los subcomponentes que forman la lista ([Componente ScrollPane](#) y [RectBorder](#)). Para más información, consulte “[Componente ScrollPane](#)” en la [página 499](#). Puede utilizar el método `setStyle()` (véase `UIObject.setStyle()`) para cambiar las propiedades de estilo `RectBorder` siguientes:

| estilos <code>RectBorder</code> | Posición en el borde |
|---------------------------------|----------------------|
| <code>borderColor</code>        | a                    |
| <code>highlightColor</code>     | b                    |
| <code>borderColor</code>        | c                    |
| <code>shadowColor</code>        | d                    |
| <code>borderCapColor</code>     | e                    |
| <code>shadowCapColor</code>     | f                    |
| <code>shadowCapColor</code>     | g                    |
| <code>borderCapColor</code>     | h                    |

Las propiedades de estilo definen las siguientes posiciones en el borde:



## Clase List

**Herencia** `UIObject` > `UIComponent` > `View` > `ScrollView` > `ScrollSelectList` > `List`

**Nombre de clase de `ActionScript`** `mx.controls.List`

El componente List consta de tres partes:

- Elementos
- Filas
- Proveedor de datos

Un elemento es un objeto de `ActionScript` que sirve para almacenar las unidades de información de la lista. Una lista puede concebirse como una matriz; cada espacio indexado de la matriz es un elemento. Un elemento es un objeto que, normalmente, tiene una propiedad `label` que se muestra y una propiedad `data` que se utiliza para almacenar datos.

Una fila es un componente que se utiliza para mostrar un elemento. Las filas las suministra de forma predeterminada la lista (se utiliza la clase `SelectableRow`) o las suministra el usuario, normalmente con una subclase de la clase `SelectableRow`. La clase `SelectableRow` implementa la interfaz `CellRenderer`, que es el conjunto de propiedades y métodos que permiten que la lista pueda manipular cada fila y enviar datos e información sobre el estado (por ejemplo, tamaño, selección, etc.) a la lista para su visualización.

Un proveedor de datos es el modelo de datos de la lista de elementos de una lista. Si en un mismo fotograma hay una lista y una matriz, ésta recibe automáticamente métodos que permiten manipular datos y difundir cambios a varias vistas. Puede crear una instancia de `Array` u obtener una en un servidor y utilizarla como modelo de datos para varios componentes `List`, `ComboBox`, `DataGrid`, etc. El componente `List` posee un conjunto de métodos proxy para su proveedor de datos (por ejemplo, `addItem()` y `removeItem()`). Si no se proporciona ningún proveedor de datos externo a la lista, estos métodos crean automáticamente una instancia de proveedor de datos que se presenta mediante `List.dataProvider`.

Para añadir un componente `List` al orden de tabulación de una aplicación, defina su propiedad `tabIndex` (véase [UIComponent.tabIndex](#)). El componente `List` utiliza `FocusManager` para sustituir el rectángulo de selección predeterminado de `Flash Player` por un rectángulo de selección personalizado con esquinas redondeadas. Para más información, consulte “[Creación de un desplazamiento personalizado de la selección](#)” en la página 27.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.List.version);
```

**Nota:** el código siguiente devuelve `undefined`: `trace(myListInstance.version);`.

## Resumen de métodos de la clase `List`

| Método                              | Descripción                                                                    |
|-------------------------------------|--------------------------------------------------------------------------------|
| <code>List.addItem()</code>         | Añade un elemento al final de la lista.                                        |
| <code>List.addItemAt()</code>       | Añade un elemento a la lista en el índice especificado.                        |
| <code>List.getItemAt()</code>       | Devuelve el elemento del índice especificado.                                  |
| <code>List.removeAll()</code>       | Elimina todos los elementos de la lista.                                       |
| <code>List.removeItemAt()</code>    | Elimina el elemento en el índice especificado.                                 |
| <code>List.replaceItemAt()</code>   | Sustituye por otro el elemento del índice especificado.                        |
| <code>List.setPropertiesAt()</code> | Aplica las propiedades especificadas al elemento especificado.                 |
| <code>List.sortItems()</code>       | Ordena los elementos de la lista según la función de comparación especificada. |
| <code>List.sortItemsBy()</code>     | Ordena los elementos de la lista según la propiedad especificada.              |

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

## Resumen de propiedades de la clase List

| Propiedad                              | Descripción                                                                                                                                  |
|----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">List.cellRenderer</a>      | Asigna la clase o el símbolo que se va utilizar para mostrar cada fila de la lista.                                                          |
| <a href="#">List.dataProvider</a>      | Origen de los elementos de la lista.                                                                                                         |
| <a href="#">List.hPosition</a>         | Posición horizontal de la lista.                                                                                                             |
| <a href="#">List.hScrollPolicy</a>     | Indica si la barra de desplazamiento horizontal se muestra ("on") o no ("off").                                                              |
| <a href="#">List.iconField</a>         | Campo de cada elemento que se utiliza para especificar iconos.                                                                               |
| <a href="#">List.iconFunction</a>      | Función que determina qué icono utilizar.                                                                                                    |
| <a href="#">List.labelField</a>        | Especifica el campo de cada elemento que se utilizará como texto de etiqueta.                                                                |
| <a href="#">List.labelFunction</a>     | Función que determina qué campos de cada elemento se utilizarán para el texto de etiqueta.                                                   |
| <a href="#">List.length</a>            | Longitud de la lista en elementos. Es una propiedad de sólo lectura.                                                                         |
| <a href="#">List.maxHPosition</a>      | Especifica el número de píxeles que la lista puede desplazarse a la derecha cuando <a href="#">List.hScrollPolicy</a> está definida en "on". |
| <a href="#">List.multipleSelection</a> | Indica si en la lista se permite la selección múltiple ( <code>true</code> ) o no ( <code>false</code> ).                                    |
| <a href="#">List.rowCount</a>          | Número de filas que son al menos parcialmente visibles en la lista.                                                                          |
| <a href="#">List.rowHeight</a>         | Altura de las filas de la lista, expresada en píxeles.                                                                                       |
| <a href="#">List.selectable</a>        | Indica si la lista es seleccionable ( <code>true</code> ) o no ( <code>false</code> ).                                                       |
| <a href="#">List.selectedIndex</a>     | Índice de una selección en una lista de selección única.                                                                                     |
| <a href="#">List.selectedIndices</a>   | Matriz de los elementos seleccionados en una lista de selección múltiple.                                                                    |
| <a href="#">List.selectedItem</a>      | Elemento seleccionado en una lista de selección única. Es una propiedad de sólo lectura.                                                     |
| <a href="#">List.selectedItems</a>     | Objetos del elemento seleccionado en una lista de selección múltiple. Es una propiedad de sólo lectura.                                      |
| <a href="#">List.vPosition</a>         | Desplaza la lista para que el elemento visible en la parte superior sea el número asignado.                                                  |
| <a href="#">List.vScrollPolicy</a>     | Indica si la barra de desplazamiento vertical se muestra ("on"), no se muestra ("off") o se muestra cuando es necesario ("auto").            |

Hereda todas las propiedades de [UIObject](#) y [UIComponent](#).



## Resumen de eventos de la clase List

| Evento                         | Descripción                                                                                 |
|--------------------------------|---------------------------------------------------------------------------------------------|
| <code>List.change</code>       | Se difunde siempre que la selección cambia por la interacción del usuario.                  |
| <code>List.itemRollOut</code>  | Se difunde cuando el puntero se desplaza por elementos de la lista y después sale de ellos. |
| <code>List.itemRollOver</code> | Se difunde cuando el puntero se desplaza por elementos de la lista.                         |
| <code>List.scroll</code>       | Se difunde cuando se recorre la lista.                                                      |

Hereda todos los eventos de [UIObject](#) y [UIComponent](#).

### List.addItem()

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.

#### Sintaxis

```
listInstance.addItem(label[, data])  
listInstance.addItem(itemObject)
```

#### Parámetros

*label* Cadena que indica la etiqueta del elemento nuevo.  
*data* Datos del elemento. Este parámetro es optativo y puede ser cualquier tipo de datos.  
*itemObject* Objeto de elemento que normalmente tiene propiedades *label* y *data*.

#### Valor devuelto

Índice en el que se ha añadido el elemento.

#### Descripción

Método; añade un elemento nuevo al final de la lista.

En el primer ejemplo de sintaxis, se crea siempre un objeto de elemento con la propiedad *label* especificada y, si se ha especificado, la propiedad *data*.

En el segundo ejemplo de sintaxis, se añade el objeto de elemento especificado.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

#### Ejemplo

Las dos siguientes líneas de código añaden un elemento a la instancia `myList`. Para probar este código, arrastre un componente List al escenario y asígnele el nombre de instancia **myList**. Añada el código siguiente al fotograma 1 de la línea de tiempo:

```
myList.addItem("esto es un elemento");  
myList.addItem({label:"Gordon",age:"very old",data:123});
```

## List.addItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.addItemAt(index, label[, data])
```

```
listInstance.addItemAt(index, itemObject)
```

### Parámetros

*label* Cadena que indica la etiqueta del elemento nuevo.

*data* Datos del elemento. Este parámetro es optativo y puede ser cualquier tipo de datos.

*index* Número mayor o igual que cero que indica la posición del elemento.

*itemObject* Objeto de elemento que normalmente tiene propiedades *label* y *data*.

### Valor devuelto

Índice en el que se ha añadido el elemento.

### Descripción

Método; añade un elemento nuevo en la posición especificada por el parámetro *index*.

En el primer ejemplo de sintaxis, se crea siempre un objeto de elemento con la propiedad *label* especificada y, si se ha especificado, la propiedad *data*.

En el segundo ejemplo de sintaxis, se añade el objeto de elemento especificado.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

### Ejemplo

La siguiente línea de código añade un elemento en la posición del tercer índice, que es el cuarto elemento de la lista:

```
myList.addItemAt(3,{label:'Red',data:0xFF0000});
```

## List.cellRenderer

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.cellRenderer
```

## Descripción

Propiedad; asigna el procesador de celdas que va a utilizarse en cada fila de la lista. Esta propiedad debe ser una referencia a un objeto de clase o un identificador de vinculación del símbolo. Todas las clases que se utilicen con esta propiedad deben implementar “API `CellRenderer`” en [la página 87](#).

## Ejemplo

En el ejemplo siguiente se utiliza un identificador de vinculación para definir un nuevo procesador de celdas:

```
myList.cellRenderer = "ComboBoxCell";
```

## List.change

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(change){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    // aquí código propio  
}  
listInstance.addEventListener("change", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando el índice seleccionado de la lista cambia como resultado de la interacción del usuario.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `List`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myBox` del componente `List`, envía “\_level0.myBox” al panel Salida:

```
on(click){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso, *change*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

Finalmente, se llama al método `addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

### Ejemplo

En el ejemplo siguiente se envía al panel Salida el nombre de instancia del componente que ha generado el evento *change*:

```
form.change = function(eventObj){
    trace("Value changed to " + eventObj.target.value);
}
myList.addEventListener("change", form);
```

### Véase también

`UIEventDispatcher.addEventListener()`

## List.dataProvider

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

`listInstance.dataProvider`

### Descripción

Propiedad; modelo de datos de los elementos que se ven en una lista. El valor de esta propiedad puede ser una matriz o cualquier objeto que implemente la interfaz *DataProvider*. El valor predeterminado es []. Para más información sobre la interfaz *DataProvider*, consulte [“Interfaz API de DataProvider” en la página 200](#).

El componente *List* y otros componentes para datos añaden métodos al prototipo del objeto *Array* para ceñirse a la interfaz *DataProvider*. Por lo tanto, cualquier matriz que exista a la vez que la lista tiene automáticamente todos los métodos necesarios (`addItem()`, `getItemAt()`, etc.) para ser el modelo de datos de la lista y puede utilizarse para difundir cambios de modelo a varios componentes.

Si la matriz contiene objetos, accede a las propiedades `List.labelField` o `List.labelFunction` para determinar qué partes del elemento deben mostrarse. El valor predeterminado es "label", por lo que si existe un campo `label`, es el campo que se muestra; en caso contrario, se muestra una lista de todos los campos separados por comas.

**Nota:** si la matriz contiene cadenas y no objetos en todos los índices, la lista no puede ordenar los elementos y mantener el estado de la selección. Al ordenar se pierde la selección.

Como proveedor de datos del componente `List` pueden elegirse todas las instancias que implementen la interfaz `DataProvider`. Entre ellas se encuentran los juegos de registros de Flash Remoting, los conjuntos de datos de Firefly, etc.

### Ejemplo

Este ejemplo utiliza una matriz de cadenas para rellenar la lista:

```
list.dataProvider = ["Ground Shipping", "2nd Day Air", "Next Day Air"];
```

Este ejemplo crea una matriz proveedora de datos y la asigna a la propiedad `dataProvider`:

```
myDP = new Array();
list.dataProvider = myDP;

for (var i=0; i<accounts.length; i++) {
    // estos cambios en DataProvider se difundirán a la lista
    myDP.addItem({ label: accounts[i].name,
                    data: accounts[i].accountID });
}
```

## List.getItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.getItemAt(index)
```

### Parámetros

*index* Número mayor o igual que 0 y menor que `List.length`. Índice del elemento que debe recuperarse.

### Valor devuelto

Objeto del elemento indexado. Undefined si el índice está fuera de rango.

### Descripción

Método; recupera el elemento del índice especificado.

### Ejemplo

El código siguiente muestra la etiqueta del elemento de la posición 4 del índice:

```
trace(myList.getItemAt(4).label);
```

## List.hPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance.hPosition*

### Descripción

Propiedad; desplaza la lista el número de píxeles especificado en sentido horizontal. `hPosition` no puede definirse a menos que el valor de `hScrollPolicy` sea "on" y la lista tenga un `maxHPosition` superior a 0.

### Ejemplo

En el ejemplo siguiente se obtiene la posición de desplazamiento horizontal de `myList`:

```
var scrollPos = myList.hPosition;
```

En el ejemplo siguiente se define la posición de desplazamiento horizontal en el extremo izquierdo:

```
myList.hPosition = 0;
```

## List.hScrollPolicy

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance.hScrollPolicy*

### Descripción

Propiedad; cadena que determina si se muestra o no la barra de desplazamiento horizontal; el valor puede ser "on" u "off". El valor predeterminado es "off". La barra de desplazamiento horizontal no mide el texto; es preciso definir una posición máxima de desplazamiento horizontal, véase [List.maxHPosition](#).

**Nota:** `List.hScrollPolicy` no admite el valor "auto".

### Ejemplo

El código siguiente activa el desplazamiento horizontal de la lista hasta 200 píxeles:

```
myList.hScrollPolicy = "on";  
myList.Box.maxHPosition = 200;
```

### Véase también

[List.hPosition](#), [List.maxHPosition](#)

## List.iconField

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance*.iconField

### Descripción

Propiedad; especifica el nombre del campo que va a utilizarse como identificador de icono. Si el valor del campo es `undefined`, se utiliza el icono predeterminado especificado en el estilo `defaultIcon`. Si el estilo `defaultIcon` es `undefined`, no se utiliza ningún icono.

### Ejemplo

En el ejemplo siguiente se define la propiedad `iconField` en la propiedad `icon` de cada elemento:

```
list.iconField = "icon"
```

### Véase también

[List.iconFunction](#)

## List.iconFunction

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance*.iconFunction

### Descripción

Propiedad; especifica la función que va a utilizarse para determinar qué icono utilizará cada fila para mostrar su elemento. Esta función recibe un parámetro *item*, que es el elemento que se va a presentar, y debe devolver una cadena que represente el identificador de símbolo del icono.

## Ejemplo

En el ejemplo siguiente se añaden iconos que indican si el archivo es un documento de imagen o de texto. Si el campo `data.fileExtension` contiene un valor "jpg" o "gif", el icono utilizado será "pictureIcon", y así sucesivamente:

```
list.iconFunction = function(item){
    var type = item.data.fileExtension;
    if (type=="jpg" || type=="gif") {
        return "pictureIcon";
    } else if (type=="doc" || type=="txt") {
        return "docIcon";
    }
}
```

## List.ItemRollOut

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(itemRollOut){
    // aquí código propio
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.itemRollOut = function(eventObject){
    // aquí código propio
}
listInstance.addEventListener("itemRollOut", listenerObject)
```

### Objeto Event

Además de las propiedades estándar del objeto Event, el evento `itemRollOut` tiene una propiedad adicional: `index`. `index` es el número del elemento del que ha salido el puntero.

### Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero sale de los elementos de la lista.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente List. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myList` del componente List, envía “\_level0.myList” al panel Salida:

```
on(itemRollOut){
    trace(this);
}
```



El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso, *itemRollOut*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método *UIEventDispatcher.addEventListener()* de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar por qué número de índice de elemento ha pasado el puntero:

```
form.itemRollOut = function (eventObj) {  
    trace("Item #" + eventObj.index + " has been rolled out.");  
}  
myList.addEventListener("itemRollOut", form);
```

### Véase también

[List.itemRollOver](#)

## List.itemRollOver

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(itemRollOver){  
    // aquí código propio  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.itemRollOver = function(eventObject){  
    // aquí código propio  
}  
listInstance.addEventListener("itemRollOver", listenerObject)
```

### Objeto Event

Además de las propiedades estándar del objeto *Event*, el evento *itemRollOver* tiene una propiedad adicional: *index*. *index* es el número del elemento sobre el que ha pasado el puntero.

## Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero se desplaza por los elementos de la lista.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `List`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myList` del componente `List`, envía “\_level0.myList” al panel Salida:

```
on(itemRollOver){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*listInstance*) distribuye un evento (en este caso, `itemRollOver`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 602.

## Ejemplo

En el ejemplo siguiente se envía un mensaje al panel Salida para indicar por qué número de índice de elemento ha pasado el puntero:

```
form.itemRollOver = function (eventObj) {
    trace("Item #" + eventObj.index + " has been rolled over.");
}
myList.addEventListener("itemRollOver", form);
```

## Véase también

[List.itemRollOut](#)

## List.labelField

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance*.labelField

## Descripción

Propiedad; especifica un campo en cada elemento para utilizarlo como texto que se muestra. Esta propiedad toma el valor del campo y lo utiliza como etiqueta. El valor predeterminado es "label".

## Ejemplo

En el ejemplo siguiente se define la propiedad `labelField` en el campo "name" de cada elemento. "Nina" aparecería como la etiqueta del elemento añadido en la segunda línea de código:

```
list.labelField = "name";  
list.addItem({name: "Nina", age: 25});
```

## Véase también

[List.labelFunction](#)

## List.labelFunction

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance*.labelFunction

## Descripción

Propiedad; especifica la función que se debe utilizar para decidir qué campo (o combinación de campos) va a mostrarse de cada elemento. Esta función recibe un parámetro *item*, que es el elemento que se va a presentar, y debe devolver una cadena que represente el texto que va a mostrarse.

## Ejemplo

En el ejemplo siguiente, la etiqueta muestra algunos detalles formateados de los elementos:

```
list.labelFunction = function(item){  
    return "The price of product " + item.productID + ", " + item.productName +  
        " es EUR"  
    + item.price;  
}
```

## Véase también

[List.labelField](#)

## List.length

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance.length*

### Descripción

Propiedad (sólo lectura); número de elementos de la lista.

### Ejemplo

En el ejemplo siguiente se coloca el valor de `length` en una variable:

```
var len = myList.length;
```

## List.maxHPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance.maxHPosition*

### Descripción

Propiedad; especifica el número de píxeles que puede desplazarse la lista cuando [List.hScrollPolicy](#) está definida en "on". La lista no mide con precisión la anchura del texto que contiene. Es preciso definir `maxHPosition` para indicar la cantidad de desplazamiento necesario. Si no se define la propiedad, la lista no se desplazará horizontalmente.

### Ejemplo

En el ejemplo siguiente se crea una lista con 400 píxeles de desplazamiento horizontal:

```
myList.hScrollPolicy = "on";  
myList.maxHPosition = 400;
```

### Véase también

[List.hScrollPolicy](#)

## List.multipleSelection

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.multipleSelection
```

### Descripción

Propiedad; indica si se permite la selección múltiple (`true`) o única (`false`). El valor predeterminado es `false`.

### Ejemplo

En el ejemplo siguiente se hace una prueba para determinar si pueden seleccionarse varios elementos:

```
if (myList.multipleSelection){  
    // aquí código propio  
}
```

El ejemplo siguiente permite realizar varias selecciones en la lista:

```
myList.selectMultiple = true;
```

## List.removeAll()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.removeAll()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; elimina todos los elementos de la lista.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

### Ejemplo

El código siguiente borra la lista:

```
myList.removeAll();
```

## List.removeItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.removeItemAt(index)
```

### Parámetros

*index* Cadena que indica la etiqueta del elemento nuevo. Valor mayor que cero y menor que `List.length`.

### Valor devuelto

Objeto; elemento eliminado (undefined si no hay elementos).

### Descripción

Método; elimina el elemento de la posición especificada en *index*. Los índices de la lista a partir del índice que indica el parámetro *index* se contraen una posición.

Cuando se llama a este método, se modifica el proveedor de datos del componente List. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

### Ejemplo

El código siguiente elimina el elemento de la posición 3 del índice:

```
myList.removeItemAt(3);
```

## List.replaceItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.replaceItemAt(index, label[, data])  
listInstance.replaceItemAt(index, itemObject)
```

## Parámetros

*index* Número mayor que cero y menor que `List.length` que indica la posición en la que insertar el elemento (índice del elemento nuevo).

*label* Cadena que indica la etiqueta del elemento nuevo.

*data* Datos del elemento. Este parámetro es optativo y puede ser de cualquier tipo.

*itemObject* Objeto que se utiliza como el elemento, que normalmente contiene las propiedades `label` y `data`.

## Valor devuelto

Ninguno.

## Descripción

Método; sustituye el contenido del elemento del índice que especifica el parámetro *index*.

Cuando se llama a este método, se modifica el proveedor de datos del componente `List`. Si el proveedor de datos se comparte con otros componentes, éstos también se actualizan.

## Ejemplo

En el ejemplo siguiente se cambia la posición del cuarto índice:

```
myList.replaceItemAt(3, "new label");
```

## List.rowCount

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.rowCount
```

## Descripción

Propiedad; número de filas que son al menos parcialmente visibles en la lista. Resulta útil cuando la escala de la lista está expresada en píxeles y hace falta contar las filas. Por el contrario, si se define el número de filas, se muestra un número exacto de filas sin que aparezca la fila final partida.

El código `myList.rowCount = num` es equivalente al código `myList.setSize(myList.width, h)` (donde `h` es la altura necesaria para mostrar el número `num` de elementos).

El valor predeterminado se basa en la altura de la lista definida durante la edición o con el método `list.setSize()` (véase [UIObject.setSize\(\)](#)).

## Ejemplo

En el ejemplo siguiente se revela el número de elementos visibles de una lista:

```
var rowCount = myList.rowCount;
```

En el ejemplo siguiente, la lista muestra cuatro elementos:

```
myList.rowCount = 4;
```

Este ejemplo elimina la fila parcial del final de la lista, si la hay:

```
myList.rowCount = myList.rowCount;
```

Este ejemplo define una lista con el número mínimo de filas que puede mostrar por completo:

```
myList.rowCount = 1;  
trace("myList has "+myList.rowCount+" rows");
```

## List.rowHeight

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.rowHeight
```

### Descripción

Propiedad; altura de cada fila de la lista, expresada en píxeles. Las filas no aumentan de tamaño para que encaje la fuente configurada, por lo que definir la propiedad `rowHeight` es la mejor manera de asegurar que los elementos se muestren por completo. El valor predeterminado es 20.

### Ejemplo

En el ejemplo siguiente se definen las filas en 30 píxeles:

```
myList.rowHeight = 30;
```

## List.scroll

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.



## Sintaxis

### Sintaxis 1:

```
on(scroll){  
    // aquí código propio  
}
```

### Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.scroll = function(eventObject){  
    // aquí código propio  
}  
listInstance.addEventListener("scroll", listenerObject)
```

## Objeto Event

Además de las propiedades estándar del objeto Event, el evento `scroll` tiene una propiedad adicional, `direction`. Es una cadena con dos valores posibles: "horizontal" o "vertical". En los eventos `scroll` de `ComboBox`, el valor es siempre "vertical".

## Descripción

Evento; se difunde a todos los detectores registrados cuando se recorre la lista.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente List. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myList` del componente List, envía "`_level0.myList`" al panel Salida:

```
on(scroll){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`listInstance`) distribuye un evento (en este caso, `scroll`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (`listenerObject`) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (`eventObject`) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte "[Objetos Event](#)" en la página 602.

## Ejemplo

En el ejemplo siguiente se envía al panel Salida el nombre de instancia del componente que ha generado el evento `change`:

```
form.scroll = function(eventObj){  
    trace("list scrolled");  
}  
myList.addEventListener("scroll", form);
```

## List.selectable

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance.selectable*

### Descripción

Propiedad; valor booleano que indica si la lista es seleccionable (*true*) o no (*false*). El valor predeterminado es *true*.

## List.selectedIndex

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance.selectedIndex*

### Descripción

Propiedad; índice seleccionado en una lista de selección única. El valor es *undefined* si no se selecciona nada; el valor es igual al último elemento seleccionado si se seleccionan varios. Si se asigna un valor a *selectedIndex*, se borra la selección actual y se selecciona el elemento indicado.

### Ejemplo

En este ejemplo se selecciona un elemento después del elemento seleccionado actualmente. Si no se selecciona nada, se selecciona el elemento 0:

```
var selIndex = myList.selectedIndex;
myList.selectedIndex = (selIndex==undefined ? 0 : selIndex+1);
```

### Véase también

[List.selectedIndices](#), [List.selectedItem](#), [List.selectedItems](#)

## List.selectedIndices

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*listInstance.selectedIndices*

## Descripción

Propiedad; matriz de índices de los elementos seleccionados. Si se asigna esta propiedad, se sustituye la selección actual. Si `selectedIndices` se define en una matriz de longitud 0 (o de valor no definido), se borra la selección actual. El valor es `undefined` si no se selecciona nada.

La propiedad `selectedIndices` muestra la lista en el orden en el que se han seleccionado los elementos. Si se hace clic en el segundo elemento, después en el tercero y después en el primero, `selectedIndices` devuelve `[1,2,0]`.

## Ejemplo

En el ejemplo siguiente se obtienen los índices seleccionados:

```
var selIndices = myList.selectedIndices;
```

En el ejemplo siguiente se seleccionan cuatro elementos:

```
var myArray = new Array (1,4,5,7);  
myList.selectedIndices = myArray;
```

## Véase también

[List.selectedIndex](#), [List.selectedItem](#), [List.selectedItems](#)

## List.selectedItem

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*listInstance.selectedItem*

## Descripción

Propiedad (sólo lectura); objeto de elemento de una lista de selección única. (En una lista de selección múltiple con varios elementos seleccionados, `selectedItem` devuelve el elemento seleccionado en último lugar). Si no hay selección, el valor es `undefined`.

## Ejemplo

En este ejemplo se muestra la etiqueta seleccionada:

```
trace(myList.selectedItem.label);
```

## Véase también

[List.selectedIndex](#), [List.selectedIndices](#), [List.selectedItems](#)

## List.selectedItems

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance.selectedItems*

### Descripción

Propiedad (sólo lectura); matriz de objetos del elemento seleccionado. En una lista de selección múltiple, `selectedItems` permite acceder a los objetos del conjunto de elementos seleccionados.

### Ejemplo

En el ejemplo siguiente se obtiene la matriz de objetos de elemento seleccionados:

```
var myObjArray = myList.selectedItems;
```

### Véase también

[List.selectedIndex](#), [List.selectedItem](#), [List.selectedIndices](#)

## List.setPropertiesAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance.setPropertiesAt(index, styleObj)*

### Parámetros

*index* Número mayor que cero o menor que `List.length` que indica el índice del elemento que va a cambiarse.

*styleObj* Objeto que enumera las propiedades y valores que van a definirse.

### Valor devuelto

Ninguno.

### Descripción

Método; aplica las propiedades especificadas en el parámetro *styleObj* al elemento especificado en el parámetro *index*. Las propiedades admitidas son `icon` y `backgroundColor`.

### Ejemplo

En el ejemplo siguiente se cambia el color del cuarto elemento a negro y se le asigna un icono:

```
myList.setPropertiesAt(3, {backgroundColor:0x000000, icon: "file"});
```

## List.sortItems()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.sortItems(compareFunc)
```

### Parámetros

*compareFunc* Referencia a una función. Esta función se utiliza para comparar dos elementos y determinar su orden de clasificación.

Para más información, consulte `Array.sort()` en el apartado Diccionario de ActionScript de la Ayuda.

### Valor devuelto

Índice en el que se ha añadido el elemento.

### Descripción

Método; ordena los elementos de la lista según el parámetro *compareFunc*.

### Ejemplo

En el ejemplo siguiente se ordenan los elementos por etiquetas en mayúsculas. Observe que los parámetros *a* y *b* que pasan a la función son elementos que tienen las propiedades `label` y `data`:

```
myList.sortItems(uppercaseFunc);  
function uppercaseFunc(a,b){  
    return a.label.toUpperCase() > b.label.toUpperCase();  
}
```

### Véase también

[List.sortItemsBy\(\)](#)

## List.sortItemsBy()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.sortItemsBy(fieldName, order)
```

## Parámetros

*fieldName* Cadena que especifica el nombre de la propiedad que va a utilizarse para ordenar. Normalmente, este valor es "label" o "data".

*order* Cadena que especifica si los elementos deben clasificarse en orden ascendente ("ASC") o descendente ("DESC").

## Valor devuelto

Ninguno.

## Descripción

Método; clasifica los elementos de la lista alfabética o numéricamente y en el orden especificado con la propiedad *fieldName*. Si los elementos de *fieldName* son una combinación de cadenas de texto y enteros, los enteros aparecen en primer lugar. El parámetro *fieldName* suele ser "label" o "data", pero puede especificarse cualquier valor de datos primitivos.

## Ejemplo

En el código siguiente, los elementos de la lista `surnameMenu` se clasifican en orden ascendente por etiquetas:

```
surnameMenu.sortItemsBy("label", "ASC");
```

## Véase también

[List.sortItems\(\)](#)

## List.vPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
listInstance.vPosition
```

### Descripción

Propiedad; ordena la lista de tal forma que el índice es el elemento visible en la parte superior. Si el índice está fuera de límite, se utiliza el índice más próximo que esté dentro del límite. El valor predeterminado es 0.

### Ejemplo

En el ejemplo siguiente se define la posición de la lista en el elemento del primer índice:

```
myList.vPosition = 0;
```

## List.vScrollPolicy

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*listInstance.vScrollPolicy*

### Descripción

Propiedad; cadena que determina si la lista admite desplazamiento vertical o no. Esta propiedad puede tener uno de estos valores: "on", "off" o "auto". El valor "auto" muestra la barra de desplazamiento cuando es necesario.

### Ejemplo

En el ejemplo siguiente se desactiva la barra de desplazamiento:

```
myList.vScrollPolicy = "off";
```

Aun así, puede crear desplazamiento con [List.vPosition](#).

### Véase también

[List.vPosition](#)

## Componente Loader

El componente Loader es un contenedor que puede mostrar archivos SWF o JPEG. Se puede adaptar el tamaño del contenido al contenedor o cambiar el tamaño del contenedor para que quepa el contenido. De forma predeterminada, es el contenido el que se ajusta al componente Loader. También se puede cargar el contenido en tiempo de ejecución y controlar el progreso de carga.

El componente Loader no se puede seleccionar. Sin embargo, sí se puede seleccionar el contenido cargado en el componente e interactuar con él. Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase FocusManager” en la página 292](#).

La vista previa dinámica de cada instancia de Loader refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes.

Es posible activar la accesibilidad del contenido cargado en un componente Loader. En tal caso, utilice el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#).

## Utilización del componente Loader

Utilice un componente Loader siempre que necesite captar contenido de una ubicación remota y traerlo a una aplicación de Flash. Por ejemplo, puede utilizar un componente Loader para añadir el logotipo de una empresa (archivo JPEG) a un formulario. También puede utilizar un componente Loader para aprovechar trabajo ya terminado en Flash. Por ejemplo, si ya ha creado una aplicación de Flash y desea ampliarla, puede utilizar el componente Loader para traer la aplicación antigua a una nueva, quizá como sección de una interfaz de fichas. En otro ejemplo, puede utilizar el componente Loader en una aplicación que muestra fotografías. Utilice `Loader.load()`, `Loader.percentLoaded` y `Loader.complete` para controlar el tiempo de carga de imágenes y ver barras de progreso durante la carga.

## Parámetros del componente Loader

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Loader en el inspector de propiedades o en el panel Inspector de componentes:

**autoload** indica si el contenido debe cargarse automáticamente (true) o si es preciso esperar hasta que se llame al método `Loader.load()` (false). El valor predeterminado es true.

**contentPath** URL absoluta o relativa que indica el archivo que debe cargarse en el cargador. La ruta relativa debe hacer referencia al archivo SWF que carga el contenido. La dirección URL debe estar en el mismo subdominio que la URL donde reside actualmente el contenido de Flash. Tanto si van a utilizarse en Flash Player como si se van a probar en modo Probar película, todos los archivos SWF deben almacenarse en la misma carpeta y los nombres de archivo no deben incluir especificaciones de carpeta ni de unidad de disco. El valor predeterminado es undefined hasta que se inicia la carga.

**scaleContent** indica si el contenido se redimensiona para adaptarse al componente Loader (true) o si el componente Loader se redimensiona para adaptarse al contenido (false). El valor predeterminado es true.

Puede escribir código ActionScript para definir opciones adicionales para instancias de Loader con sus métodos, propiedades y eventos. Para más información, consulte [Clase Loader](#).

## Creación de aplicaciones con el componente Loader

El siguiente procedimiento explica cómo añadir un componente Loader a una aplicación durante la edición. En este ejemplo, el componente Loader carga un logotipo JPEG de una URL imaginaria.

**Para crear una aplicación con el componente Loader, siga este procedimiento:**

- 1 Arrastre un componente Loader desde el panel Componentes al escenario.
- 2 Seleccione un componente Loader en el escenario y utilice la herramienta Transformación libre para adaptar su tamaño a las dimensiones del logotipo empresarial.
- 3 En el inspector de propiedades, introduzca el nombre de instancia **logo**.
- 4 Seleccione el cargador en el escenario y en el panel Inspector de componentes, e introduzca `http://corp.com/websites/logo/corplogo.jpg` para el parámetro `contentPath`.



## Personalización del componente Loader

El componente Loader puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`).

El comportamiento de ajuste de tamaño del componente Loader se controla con la propiedad `scaleContent`. Cuando `scaleContent = true`, el contenido se redimensiona para adaptarse a los límites del componente Loader (y vuelve a redimensionarse cuando se llama al método `UIObject.setSize()`). Cuando la propiedad es `scaleContent = false`, el tamaño del componente se ajusta al tamaño del contenido y el método `UIObject.setSize()` no tiene efecto alguno.

## Utilización de estilos con el componente Loader

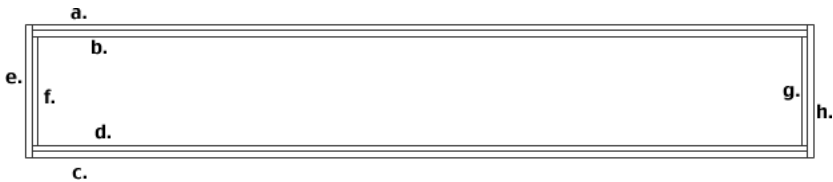
El componente Loader no utiliza estilos.

## Utilización de aspectos con el componente Loader

El componente Loader utiliza `RectBorder`, que a su vez utiliza la interfaz API de dibujo de `ActionScript`. Puede utilizar el método `setStyle()` (véase `UIObject.setStyle()`) para cambiar las propiedades de estilo `RectBorder` siguientes:

| estilos <code>RectBorder</code> | Letra |
|---------------------------------|-------|
| <code>borderColor</code>        | a     |
| <code>highlightColor</code>     | b     |
| <code>borderColor</code>        | c     |
| <code>shadowColor</code>        | d     |
| <code>borderCapColor</code>     | e     |
| <code>shadowCapColor</code>     | f     |
| <code>shadowCapColor</code>     | g     |
| <code>borderCapColor</code>     | h     |

Las propiedades de estilo definen las siguientes posiciones en el borde:



## Clase Loader

**Herencia** UIObject > UIComponent > View > Loader

**Nombre de clase de ActionScript** mx.controls.Loader

Las propiedades de la clase Loader permiten definir el contenido que va a cargarse y controlar el progreso de carga en tiempo de ejecución.

Si una propiedad de la clase Loader se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

Para más información, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.Loader.version);
```

**Nota:** el código siguiente devuelve undefined: `trace(myLoaderInstance.version);`.

### Resumen de métodos de la clase Loader

| Método                        | Descripción                                                                |
|-------------------------------|----------------------------------------------------------------------------|
| <a href="#">Loader.load()</a> | Carga el contenido especificado en la propiedad <code>contentPath</code> . |

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

### Resumen de propiedades de la clase Loader

| Propiedad                            | Descripción                                                                                                                                                                                                       |
|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Loader.autoLoad</a>      | Valor booleano que indica si el contenido se carga automáticamente ( <code>true</code> ) o si es preciso llamar a <a href="#">Loader.load()</a> ( <code>false</code> ).                                           |
| <a href="#">Loader.bytesLoaded</a>   | Propiedad de sólo lectura que indica el número de bytes que se han cargado.                                                                                                                                       |
| <a href="#">Loader.bytesTotal</a>    | Propiedad de sólo lectura que indica el número total de bytes del contenido.                                                                                                                                      |
| <a href="#">Loader.content</a>       | Referencia al contenido especificado en la propiedad <a href="#">Loader.contentPath</a> . Es una propiedad de sólo lectura.                                                                                       |
| <a href="#">Loader.contentPath</a>   | Cadena que indica la URL del contenido que va a cargarse.                                                                                                                                                         |
| <a href="#">Loader.percentLoaded</a> | Número que indica el porcentaje de contenido cargado. Es una propiedad de sólo lectura.                                                                                                                           |
| <a href="#">Loader.scaleContent</a>  | Valor booleano que indica si el contenido se redimensiona para adaptarse al componente Loader ( <code>true</code> ) o si el componente Loader se redimensiona para adaptarse al contenido ( <code>false</code> ). |

Hereda todas las propiedades de [UIObject](#) y [UIComponent](#).

## Resumen de eventos de la clase Loader

| Evento                          | Descripción                                        |
|---------------------------------|----------------------------------------------------|
| <a href="#">Loader.complete</a> | Se activa cuando el contenido termina de cargarse. |
| <a href="#">Loader.progress</a> | Se activa mientras se carga el contenido.          |

Hereda todas las propiedades de [UIObject](#) y [UIComponent](#)

### Loader.autoLoad

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.

#### Sintaxis

```
loaderInstance.autoLoad
```

#### Descripción

Propiedad; valor booleano que indica si el contenido se carga automáticamente (`true`) o si es preciso esperar a que se llame a [Loader.load\(\)](#) (`false`). El valor predeterminado es `true`.

#### Ejemplo

El código siguiente define que el componente Loader espere a que se llame a [Loader.load\(\)](#):

```
loader.autoLoad = false;
```

### Loader.bytesLoaded

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.

#### Sintaxis

```
loaderInstance.bytesLoaded
```

#### Descripción

Propiedad (sólo lectura); número de bytes del contenido cargado. El valor predeterminado es 0 hasta que el contenido comienza a cargarse.

## Ejemplo

El código siguiente crea un componente `ProgressBar` y un componente `Loader`. A continuación crea un objeto detector con un controlador de eventos `progress` que muestra el progreso de la carga. El detector se registra en la instancia de `Loader` de este modo:

```
createClassObject(mx.controls.ProgressBar, "pBar", 0);
createClassObject(mx.controls.Loader, "loader", 1);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, Loader.
    pBar.setProgress(loader.bytesLoaded, loader.bytesTotal); // mostrar progreso
}
loader.addEventListener("progress", loadListener);
loader.content = "logo.swf";
```

Cuando se crea una instancia con el método `createClassObject()`, es preciso colocarla en el escenario con los métodos `move()` y `setSize()`. Véase [UIObject.move\(\)](#) y [UIObject.setSize\(\)](#).

## Véase también

[Loader.bytesTotal](#), [UIObject.createClassObject\(\)](#)

## Loader.bytesTotal

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
loaderInstance.bytesTotal
```

### Descripción

Propiedad (sólo lectura); tamaño del contenido en bytes. El valor predeterminado es 0 hasta que el contenido comienza a cargarse.

## Ejemplo

El código siguiente crea un componente `ProgressBar` y un componente `Loader`. A continuación crea un objeto detector de carga con un controlador de eventos `progress` que muestra el progreso de la carga. El detector se registra en la instancia de `Loader` de este modo:

```
createClassObject(mx.controls.ProgressBar, "pBar", 0);
createClassObject(mx.controls.Loader, "loader", 1);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, Loader.
    pBar.setProgress(loader.bytesLoaded, loader.bytesTotal); // mostrar progreso
}
loader.addEventListener("progress", loadListener);
loader.content = "logo.swf";
```

Véase también

[Loader.bytesLoaded](#)

## Loader.complete

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(complete){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.complete = function(eventObject){  
    ...  
}  
loaderInstance.addEventListener("complete", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando el contenido ha terminado de cargarse.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `Loader`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myLoaderComponent` del componente `Loader`, envía “\_level0.myLoaderComponent” al panel Salida:

```
on(complete){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*loaderInstance*) distribuye un evento (en este caso, *complete*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente se crea un componente Loader y después se define un objeto detector con un controlador de eventos `complete` que define la propiedad `visible` del componente Loader en `true`:

```
createClassObject(mx.controls.Loader, "loader", 0);
loadListener = new Object();
loadListener.complete = function(eventObj){
    loader.visible = true;
}
loader.addEventListener("complete", loadListener);
loader.contentPath = "logo.swf";
```

## Loader.content

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*loaderInstance.content*

### Descripción

Propiedad (sólo lectura); referencia al contenido del componente Loader. El valor es `undefined` hasta que comienza la carga.

### Véase también

[Loader.contentPath](#)

## Loader.contentPath

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*loaderInstance.contentPath*

### Descripción

Propiedad; cadena que indica la URL absoluta o relativa del archivo que va a cargarse en el componente Loader. La ruta relativa debe hacer referencia al archivo SWF que carga el contenido. La URL debe pertenecer al mismo subdominio que la URL del archivo SWF que va a cargarse.

Si utiliza Flash Player o el modo Probar película en Flash, todos los archivos SWF deben almacenarse en la misma carpeta y los nombres de archivo no pueden incluir información de carpeta ni de unidad de disco.

## Ejemplo

En el ejemplo siguiente se indica a la instancia de Loader que muestre el contenido del archivo “logo.swf”:

```
loader.contentPath = "logo.swf";
```

## Loader.load()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
loaderInstance.load(path)
```

### Parámetros

*path* Parámetro optativo que especifica el valor de la propiedad `contentPath` antes de iniciarse la carga. Si no se especifica ningún valor, se utiliza el valor actual de `contentPath`.

### Valor devuelto

Ninguno.

### Descripción

Método; indica al componente Loader que comience a cargar su contenido.

## Ejemplo

El código siguiente crea una instancia de Loader y define la propiedad `autoload` en `false` para que el componente espere a que se llame al método `load()` para comenzar a cargar el contenido. A continuación llama a `load()` e indica el contenido que debe cargarse:

```
createClassObject(mx.controls.Loader, "loader", 0);  
loader.autoload = false;  
loader.load("logo.swf");
```

## Loader.percentLoaded

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
loaderInstance.percentLoaded
```

## Descripción

Propiedad (sólo lectura); número que indica el porcentaje de contenido cargado. Normalmente, esta propiedad se utiliza para presentar el progreso de una forma de lectura fácil para el usuario. Utilice el siguiente código para redondear la cifra al número entero más cercano:

```
Math.round(bytesLoaded/bytesTotal*100))
```

## Ejemplo

En el ejemplo siguiente se crea una instancia de Loader y, a continuación, un objeto detector con un controlador de progreso que averigua el porcentaje cargado y lo envía al panel Salida:

```
createClassObject(Loader, "loader", 0);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, Loader.
    trace("logo.swf is " + loader.percentLoaded + "% loaded."); // track loading
    progress
}
loader.addEventListener("complete", loadListener);
loader.content = "logo.swf";
```

## Loader.progress

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(progress){
    ...
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.progress = function(eventObject){
    ...
}
loaderInstance.addEventListener("progress", listenerObject)
```



## Descripción

Evento; se difunde a todos los detectores registrados mientras se carga el contenido. Este evento se activa cuando la carga se activa con el parámetro `autoload` o llamando a `Loader.load()`. El evento `progress` no siempre se difunde. El evento `complete` puede difundirse sin que se distribuyan eventos `progress`. Esto sucede especialmente cuando el contenido cargado es un archivo local.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `Loader`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myLoaderComponent` del componente `Loader`, envía “\_level0.myLoaderComponent” al panel Salida:

```
on(progress){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*loaderInstance*) distribuye un evento (en este caso, `progress`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la [página 602](#).

## Ejemplo

El código siguiente crea una instancia de `Loader` y después un objeto detector con un controlador de eventos para el evento `progress` que envía un mensaje al panel Salida con el porcentaje de contenido cargado:

```
createClassObject(mx.controls.Loader, "loader", 0);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // es decir, Loader.
    trace("logo.swf is " + loader.percentLoaded + "% loaded."); // track loading
    progress
}
loader.addEventListener("progress", loadListener);
loader.contentPath = "logo.swf";
```

## Loader.scaleContent

### Disponibilidad

Flash Player 6 versión 79.

## Edición

Flash MX 2004.

## Sintaxis

*loaderInstance.scaleContent*

## Descripción

Propiedad; indica si el contenido se redimensiona para adaptarse al componente Loader (*true*) o si el componente Loader se redimensiona para adaptarse al contenido (*false*). El valor predeterminado es *true*.

## Ejemplo

El código siguiente indica al componente Loader que cambie de tamaño para ajustarse al tamaño de su contenido:

```
loader.strechContent = false;
```

# Componentes multimedia (sólo para Flash Professional)

Los componentes de streaming facilitan la incorporación de streaming en las presentaciones de Flash. Estos componentes permiten presentar los medios de varias maneras.

A continuación figuran tres componentes multimedia de los que puede disponer:

- El componente *MediaDisplay* permite reproducir sin interrupción el elemento multimedia en el contenido de Flash sin necesidad de utilizar una interfaz de usuario como soporte. Este componente puede utilizarse con datos de vídeo y de audio. El usuario de la aplicación no tendrá control sobre los medios cuando el componente *MediaDisplay* funcione sin la interfaz de usuario.
- El componente *MediaController* complementa al componente *MediaDisplay* proporcionando una interfaz de usuario que controla la reproducción de los medios utilizando controles estándar (reproducción, pausa, etc.). Los medios no se cargan nunca en *MediaController* y éste tampoco los reproduce; *MediaController* sólo se utiliza para controlar la reproducción en una instancia *MediaPlayback* o *MediaDisplay*. El componente *MediaController* presenta un “cajón” que muestra el contenido de los controles de reproducción al colocar el ratón sobre el componente.
- El componente *MediaPlayback* es una combinación de los componentes *MediaDisplay* y *MediaController*; dicho componente proporciona métodos para reproducir sin interrupción el contenido de los medios.

Debe considerar los puntos siguientes, relacionados con los componentes multimedia:

- Los componentes multimedia requieren Flash Player 7 o posterior.
- Los componentes multimedia no admiten la funcionalidad para explorar hacia adelante y hacia atrás. No obstante, puede conseguir esta funcionalidad moviendo el deslizador del reproductor.
- Sólo se reflejan en la vista previa dinámica el tamaño del componente y el aspecto del controlador.
- Los componentes multimedia no admiten la accesibilidad.

## Interacción con los componentes multimedia (sólo para Flash Professional)

Los componentes de flujo de medios MediaPlayer y MediaController responden a las acciones realizadas con el ratón y el teclado. El componente MediaDisplay no responde a los eventos de ratón ni de teclado. En la tabla siguiente se resumen las acciones para los componentes MediaPlayer y MediaController después de seleccionarlos:

| Destino                                                 | Navegación                                          | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------------------------------------------------------|-----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Controles de reproducción de un controlador determinado | Pasar el ratón por encima                           | Se resalta el botón.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Controles de reproducción de un controlador determinado | Hacer clic o presionar el botón izquierdo del ratón | <p>Los usuarios pueden manipular la reproducción de medios de audio y vídeo mediante los controles de reproducción de un controlador determinado haciendo clic en los controles de reproducción para activar los efectos correspondientes.</p> <p>Los botones Pausar/Reproducir e Ir al principio/Ir al final siguen los comportamientos estándar de los botones. Al presionar el botón del ratón, el botón de la pantalla se resalta en estado Presionado y al soltar el botón del ratón, el botón de la pantalla vuelve a tener el aspecto de no estar seleccionado.</p> <p>El botón Ir al final se desactiva cuando se reproducen archivos de medios FLV.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Controles deslizantes de un controlador determinado     | Mover deslizador hacia atrás y hacia adelante       | <p>El deslizador del reproductor indica la posición del usuario en el contenido multimedia; el selector de pantalla se mueve horizontalmente (de forma predeterminada) para indicar la reproducción de principio (izquierda) a fin (derecha). El deslizador del reproductor se mueve de abajo arriba cuando los controles están orientados verticalmente. A medida que el selector del indicador se mueve de izquierda a derecha, resalta el espacio de pantalla anterior para indicar que el contenido del mismo se ha vuelto a reproducir o seleccionar. El espacio de pantalla situado delante del selector del indicador permanece sin resaltar hasta que pasa el indicador. Los usuarios pueden arrastrar el selector del indicador para actuar sobre la posición de la reproducción del contenido multimedia. La reproducción automática de los medios comienza en el punto en que se suelta el ratón, si los medios se están reproduciendo. Si se detienen los medios, el deslizador puede moverse y soltarse, y los medios permanecerán en pausa.</p> <p>También hay un deslizador de volumen, que puede moverse de izquierda (sin sonido) a derecha (volumen máximo) en el diseño horizontal y vertical.</p> |

| Destino                                    | Navegación               | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Navegación del controlador de reproducción | Tab, Mayús+Tab           | Cambia la selección de un botón a otro en el componente del controlador, donde se resaltará el elemento seleccionado. Esta navegación funciona con los controles Pausar/Reproducir, Ir al principio, Ir al final, Volumen desactivado y Volumen máx. La selección cambia de izquierda a derecha y de arriba abajo a medida que los usuarios se desplazan por los elementos con la tecla de tabulación. La combinación de Mayús+Tab mueve la selección de derecha a izquierda y de abajo arriba. Tras seleccionar un elemento mediante la tecla Tab, el control pasa inmediatamente al botón Reproducir/Pausar. Cuando se selecciona el botón Volumen máx. y se presiona la tecla Tab, la selección pasa de un control al siguiente en el índice de fichas del escenario. |
| Botón de un control determinado            | Espacio o Intro/ Retorno | Marca el elemento seleccionado. Tras presionar la tecla adecuada, el botón aparece en estado Presionado. Al soltarla, el botón vuelve a tener el estado de selección que se obtiene cuando se pasa el ratón por encima.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

## Componentes multimedia (sólo para Flash Professional)

Antes de empezar a utilizar los componentes multimedia, es recomendable entender cómo funcionan. En esta sección se proporciona información general sobre el funcionamiento de los componentes multimedia. La mayor parte de las propiedades que figuran en esta sección pueden configurarse directamente con el panel Inspector de componentes. Véase [“Utilización del panel Inspector de componentes con componentes multimedia” en la página 358](#).

Aparte de las propiedades de diseño descritas más adelante en esta sección, para los componentes `MediaDisplay` y `MediaPlayer` pueden establecerse las propiedades siguientes:

- El tipo de medios, que puede establecerse en MP3 o FLV (véase `Media.mediaType` y `Media.setMedia()`).
- La ruta del contenido relativo o absoluto, que alberga el archivo de medios que se va a reproducir sin interrupción (véase `Media.contentPath`).
- Objetos de cuepoint, junto con el nombre correspondiente, la hora y las propiedades del reproductor (véase `Media.addCuePoint()` y `Media.cuePoints`). El nombre del cuepoint es arbitrario y debe establecerse de modo que tenga un significado al utilizar eventos de detector y de trazado. Un cuepoint difunde un evento `cuePoint` cuando el valor de su propiedad de tiempo es igual al de la ubicación de la cabeza lectora del componente `MediaPlayer` o `MediaDisplay` con el que está asociado. La propiedad del reproductor es una referencia a la instancia `MediaPlayer` con la que está asociada. Los cuepoints pueden quitarse posteriormente mediante `Media.removeCuePoint()` y `Media.removeAllCuePoints()`.

Los componentes de flujo de medios difunden varios eventos relacionados. Para poner en movimiento otros elementos pueden utilizarse los eventos de difusión siguientes:

- Un evento `change` se difunde continuamente mediante los componentes `MediaDisplay` y `MediaPlayback` mientras se reproducen los medios. (Véase [Media.change](#).)
- Un evento `progress` se difunde continuamente mediante los componentes `MediaDisplay` y `MediaPlayback` mientras se cargan los medios. (Véase [Media.progress](#).)
- Un evento `click` se difunde mediante los componentes `MediaController` y `MediaPlayback` al hacer clic en el botón Pausa/Reproducir. En este caso, la propiedad `detail` del objeto del evento proporciona información acerca del botón en el que se ha hecho clic. (Véase [Media.click](#).)
- Un evento `volume` se difunde mediante los componentes `MediaController` y `MediaPlayback` cuando el usuario ajusta los controles de volumen. (Véase [Media.volume](#).)
- Un evento `playheadChange` se difunde mediante los componentes `MediaController` y `MediaPlayback` cuando el usuario mueve el deslizador del reproductor. (Véase [Media.playheadChange](#).)

El componente `MediaDisplay` funciona junto con el componente `MediaController`. Al estar combinados, los componentes se comportan de una manera similar al componente `MediaPlayback`, pero ofrecen más flexibilidad con respecto al diseño. Por lo tanto, si desea dar flexibilidad al aspecto y funcionamiento de los medios cuando los vaya a presentar, utilice los componentes `MediaDisplay` y `MediaController`. De lo contrario, el componente `MediaPlayback` es la mejor elección.

## Componente MediaDisplay

Al colocar un componente `MediaDisplay` en el escenario, dicho componente se dibuja sin que haya ninguna interfaz de usuario visible. Se trata simplemente de un recuadro en el que albergar y reproducir los medios. El aspecto de cualquier medio de vídeo que se reproduzca en un componente `MediaDisplay` resulta afectado por las siguientes propiedades:

- [Media.aspectRatio](#)
- [Media.autoSize](#)
- `Height`
- `Width`

**Nota:** el usuario no podrá ver nada, a menos que se reproduzca algún medio.

La propiedad [Media.aspectRatio](#) tiene prioridad sobre las otras propiedades. Cuando [Media.aspectRatio](#) se establece en `true` (valor predeterminado), el componente siempre vuelve a ajustar el tamaño de los medios que se están reproduciendo una vez establecido el tamaño del componente, a fin de asegurar que se mantiene la proporción de los medios.

Para los archivos FLV, cuando [Media.autoSize](#) se establece en `true`, los medios que se van a reproducir se mostrarán según el tamaño establecido como preferido, independientemente del tamaño del componente. Esto implica que, a menos que el tamaño de la instancia de `MediaDisplay` sea igual al tamaño de los medios, éstos rebasarán los límites de la instancia o no llenarán el tamaño de la misma. Cuando [Media.autoSize](#) se establece en `false`, el tamaño de la instancia se utilizará mientras sea posible, al tiempo que respetará la proporción. Si [Media.autoSize](#) y [Media.aspectRatio](#) se establecen como `false`, se utilizará el tamaño exacto del componente.

**Nota:** como no hay ninguna imagen que pueda mostrarse con los archivos MP3, establecer `Media.autoSize` no tendrá ningún efecto. Para los archivos MP3, el tamaño utilizable mínimo es 60 píxeles de altura por 256 píxeles de anchura en la orientación predeterminada.

El componente `MediaDisplay` también admite la propiedad `Media.volume`. Esta propiedad acepta valores enteros del 0 al 100, donde 0 equivale a que no haya sonido y 100 al volumen máximo. El valor predeterminado es 75.

## Componente `MediaController`

La interfaz para el componente `MediaController` depende de las propiedades `Media.controllerPolicy` y `Media.backgroundStyle` correspondientes. La propiedad `Media.controllerPolicy` determina si el juego de controles de los medios está siempre visible, contraído o si sólo es visible al colocar el puntero del ratón sobre la porción de control del componente. Al contraerse, el controlador dibuja una barra de progreso modificada, que es una combinación de la barra de carga y la barra de reproducción. La barra de progreso muestra el progreso de los bytes que se están cargando en la parte inferior de la barra y el progreso de la cabeza lectora justo encima. El estado ampliado dibuja una versión ampliada de la barra de reproducción/barra de carga, que contiene los elementos siguientes:

- Etiquetas de texto a la izquierda, que indican el estado de reproducción (sin interrupción o pausada) y a la derecha, que indican la ubicación de la cabeza lectora en segundos.
- Indicador de la ubicación de la cabeza lectora
- Un deslizador, que los usuarios pueden arrastrar para desplazarse por los medios.

Con el componente `MediaController` se proporcionan también los siguientes elementos:

- Un botón de estado Pausar/Reproducir
- Un grupo de dos botones: Ir al principio e Ir al final, que sirven para desplazarse al principio y al final de los medios, respectivamente.
- Un control de volumen, que consta de un deslizador, un botón para quitar el sonido y otro botón de volumen máximo.

Tanto el estado contraído como el estado ampliado del componente `MediaController` utilizan la propiedad `Media.backgroundStyle`. Esta propiedad determina si el controlador dibuja un fondo cromático (valor predeterminado) o permite que el fondo de la película se visualice desde detrás de los controles.

El componente `MediaController` tiene una configuración de orientación, `Media.horizontal`, que puede utilizarse para dibujar el componente con la orientación horizontal (valor predeterminado) o vertical. Con la orientación horizontal, la barra de reproducción hace un seguimiento de los medios que se están reproduciendo de izquierda a derecha. Con la orientación vertical, la barra de reproducción hace un seguimiento de los medios de abajo arriba.

Los componentes `MediaDisplay` y `MediaController` pueden asociarse entre sí a través de los métodos `Media.associateDisplay()` y `Media.associateController()`. Cuando se les llama, estos métodos permiten a la instancia `MediaController` actualizar sus controles en función de los eventos difundidos desde la instancia `MediaDisplay`, así como que el componente `MediaDisplay` reaccione a la configuración creada por el usuario desde `MediaController`.

## Componente MediaPlayer

El componente MediaPlayer es una combinación de los controles MediaController y MediaDisplay. Ambos subcomponentes se encuentran en MediaPlayer. Las porciones MediaController y MediaDisplay siempre se escalan para adaptarse al tamaño total de la instancia de componente MediaPlayer.

El componente MediaPlayer utiliza `Media.controlPlacement` para determinar el diseño de los controles. Entre las posibles colocaciones del control se incluyen los valores `top`, `bottom`, `left` y `right`, que indican dónde se dibujarán los controles en relación con la pantalla. Por ejemplo, el valor `right` daría a un control la orientación vertical y lo colocaría a la derecha de la pantalla.

## Utilización de componentes multimedia (sólo para Flash Professional)

El rápido aumento de la utilización de elementos multimedia, destinados a proporcionar información a los usuarios de Internet, ha tenido como consecuencia que se desee ofrecer a los usuarios un método para reproducir los medios sin interrupción y controlarlos. A continuación figuran supuestos de ejemplo en los que pueden utilizarse los componentes multimedia:

- Mostrar los medios que presentan a una empresa
- Reproducir sin interrupción películas o vistas previas de películas
- Reproducir sin interrupción canciones o fragmentos de canciones
- Proporcionar material de aprendizaje multimedia

## Utilización del componente MediaPlayer

Suponga que debe desarrollar un sitio Web para sus clientes que permita a los usuarios del sitio Web obtener una vista previa de los DVD y CD puestos a la venta en un entorno de medios dinámicos. El ejemplo siguiente muestra los pasos que deben seguirse para conseguir esto; en él se presupone que el sitio Web está preparado para poder insertar en él componentes de reproducción sin interrupción.

### Para crear un documento de Flash que muestre una vista previa de un CD o DVD:

- 1 En Flash, seleccione Archivo > Nuevo; a continuación, seleccione Documento de Flash.
- 2 Abra el panel Componentes (Ventana > Paneles de desarrollo > Componentes) y haga doble clic en el componente MediaPlayer, que coloca una instancia del componente en el escenario.
- 3 Seleccione la instancia de componente MediaPlayer e introduzca el nombre de instancia **myMedia** en el inspector de propiedades.
- 4 En el panel Inspector de componentes (Ventana > Paneles de desarrollo > Inspector de componentes), establezca el tipo de medios que desee en función del tipo de medios que se reproducirá sin interrupción (MP3 o FLV).
- 5 Si ha seleccionado FLV, introduzca la duración del vídeo en los cuadros de texto Video Length; utilice el formato HH:MM:SS.
- 6 Introduzca la ubicación del vídeo de vista previa en el cuadro de texto URL. Por ejemplo, puede introducir `http://my.web.com/video previews/AMovieName.flv`.
- 7 Establezca las opciones que desee en las casillas de verificación Automatically Play, Use Preferred Media Size y Respect Aspect Ratio.
- 8 Establezca la colocación del control en el lado del componente MediaPlayer que desee.

- 9 Añada un cuepoint casi al final del elemento multimedia que se vaya a utilizar junto con un detector para abrir una ventana emergente que informe al usuario de que la película está a la venta. Asigne al cuepoint el nombre **cuePointName** y establezca la hora del cuepoint casi al final del clip (unos segundos antes). Para conseguir esto, siga estos pasos:
  - a Haga doble clic en un componente Window para hacerlo aparecer en el escenario.
  - b Elimine el componente Window. De este modo se coloca un elemento llamado Window en la biblioteca.
  - c Cree un cuadro de texto y escriba un texto para informar al usuario de que la película está a la venta.
  - d Convierta este cuadro de texto en un clip de película seleccionando Modificar > Convertir en símbolo y asígnele el nombre **mySale\_mc**.
  - e Haga clic con el botón derecho del ratón en el clip de película **mySale\_mc** situado en la biblioteca; seleccione Vinculación y, a continuación, la opción Exportar para ActionScript. Esto coloca el clip de película en la biblioteca de tiempo de ejecución.
- 10 Añada el siguiente código ActionScript al fotograma 1. Con este código se crea un detector para abrir una ventana emergente que informa al usuario de que la película está a la venta.

```
// Importar las clases necesarias para crear la ventana emergente
dinámicamente

import mx.containers.Window;
import mx.managers.PopUpManager;

// Crear un objeto detector para lanzar la ventana emergente de venta
var saleListener = new Object();

saleListener.cuePoint = function(evt){

var saleWin = PopUpManager.createPopUp(_root, Window, false, {closeButton:
    true, title: "Movie Sale ", contentPath: "mySale_mc"});

// Ampliar la ventana para que quepa el contenido

saleWin.setSize(80, 80);
var delSaleWin = new Object();
delSaleWin.click = function(evt){
saleWin.deletePopUp();
}
saleWin.addEventListener("click", delSaleWin);

}

myMedia.addEventListener("cuePoint", saleListener);
```



## Utilización de los componentes **MediaDisplay** y **MediaController**

Suponga que decide tener más control sobre el aspecto y funcionamiento de la visualización de los medios. Por esta razón, debe utilizar **MediaDisplay** y **MediaController** al mismo tiempo para proporcionar la experiencia deseada. En el ejemplo siguiente se muestran los pasos equivalentes a los del ejemplo anterior que servirán para crear una aplicación Flash que muestre vistas previas de CD y DVD.

### Para crear un documento de Flash que muestre una vista previa de un CD o DVD:

- 1 En Flash, seleccione Archivo > Nuevo; a continuación, seleccione Documento de Flash.
- 2 En el panel Componentes (Ventana > Paneles de desarrollo > Componentes), haga doble clic en los componentes **MediaController** y **MediaDisplay**. De este modo se colocará una instancia de cada componente en el escenario.
- 3 Seleccione la instancia **MediaDisplay** e introduzca el nombre de instancia **myDisplay** en el inspector de propiedades.
- 4 Seleccione la instancia **MediaController** e introduzca el nombre de instancia **myController** en el inspector de propiedades.
- 5 Inicie el panel Inspector de componentes desde el inspector de propiedades y defina el tipo de elemento multimedia que se reproducirá sin interrupción (MP3 o FLV).
- 6 Si ha seleccionado FLV, introduzca la duración del vídeo en los cuadros de texto de duración de vídeo, en el formato HH:MM:SS.
- 7 Introduzca la ubicación del vídeo de vista previa en el cuadro de texto URL. Por ejemplo, puede introducir <http://my.web.com/preview/AMovieName.flv>.
- 8 Establezca las opciones que desee en las casillas de verificación Automatically Play, Use Preferred Media Size y Respect Aspect Ratio.
- 9 Seleccione la instancia de **MediaController** y, en el panel Inspector de componentes, establezca la orientación como vertical definiendo la propiedad `horizontal` como `false`.
- 10 En el panel Inspector de componentes, establezca `None` como valor de `backgroundStyle`.  
Esta acción especifica que la instancia de **MediaController** no debe dibujar un fondo sino que debe llenar los elementos multimedia entre los controles.
- 11 Utilice un comportamiento para asociar las instancias de **MediaController** y **MediaDisplay** de manera que la instancia de **MediaController** refleje de forma precisa el movimiento de la cabeza lectora y otras configuraciones de la instancia de **MediaDisplay**, y ésta responda a los clics de los usuarios.
  - a Seleccione la instancia de **MediaDisplay** y, en el inspector de propiedades, introduzca el nombre de instancia **myMediaDisplay**.
  - b Seleccione la instancia de **MediaController** que activará el comportamiento.
  - c En el panel Comportamientos (Ventana > Paneles de desarrollo > Comportamientos), haga clic en el botón Añadir (+) y seleccione Medios > Visualización asociada.
  - d En la ventana Visualización asociada, seleccione **myMediaDisplay** en `_root` y haga clic en Aceptar.

Para más información sobre cómo utilizar comportamientos con componentes multimedia, consulte [“Control de componentes multimedia mediante comportamientos” en la página 359](#).

## Utilización del panel Inspector de componentes con componentes multimedia

El panel Inspector de componentes facilita la definición de los parámetros, las propiedades de componentes multimedia, etc. Para utilizar este panel, haga clic en el componente del escenario que desee y, con el inspector de propiedades abierto, haga clic en Iniciar inspector de componentes. El panel Inspector de componentes sirve para realizar las siguientes operaciones:

- Reproducir automáticamente el elemento multimedia (véanse `Media.activePlayControl` y `Media.autoPlay`)
- Mantener o ignorar la proporción del elemento multimedia (véase `Media.aspectRatio`)
- Determinar si se cambiará automáticamente el tamaño del elemento multimedia para adaptarlo a la instancia de componente (véase `Media.autoSize`)
- Activar o desactivar el fondo cromático (véase `Media.backgroundColorStyle`)
- Especificar la ruta del elemento multimedia con formato de URL (véase `Media.contentPath`)
- Especificar la visibilidad de los controles de reproducción (véase `Media.controllerPolicy`)
- Añadir objetos de cuepoint (véase `Media.addCuePoint()`)
- Eliminar objetos de cuepoint (véase `Media.removeCuePoint()`)
- Establecer la orientación de las instancias de `MediaController` (véase `Media.horizontal`)
- Establecer el tipo de elemento multimedia que se va a reproducir (véase `Media.setMedia()`)
- Establecer el tiempo de reproducción del elemento multimedia (véase `Media.totalTime`)
- Establecer los últimos dígitos de la visualización de tiempo para indicar los milisegundos o los fotogramas por segundo (fps)

Es importante que comprenda algunos conceptos del panel Inspector de componentes para trabajar con él:

- El control de tiempo del vídeo se elimina al seleccionar el tipo de vídeo MP3, ya que esta información se lee automáticamente cuando se utilizan archivos MP3. En el caso de los archivos FLV, debe introducir el tiempo total del elemento multimedia (`Media.totalTime`) para que la barra de reproducción del componente `MediaPlayback` (o cualquier otro componente `MediaController` de sonido) pueda reflejar de forma precisa el progreso de la reproducción.
- Si se selecciona el tipo de archivo FLV, aparecerá una opción de milisegundos y, si la opción de milisegundos no está seleccionada, un menú emergente de control de fotogramas por segundo (FPS). Si la opción de milisegundos está seleccionada, el control de fotogramas por segundo no aparece. En este modo, el tiempo que aparece en la barra de reproducción en tiempo de ejecución tiene el formato HH:MM:SS.mmm (H = horas, M = minutos, S = segundos, m = milisegundos) y se utiliza el valor de segundos para los cuepoints. Si la opción de milisegundos no está seleccionada, el control de fotogramas por segundo está activado y el tiempo de la barra de reproducción tiene el formato HH:MM:SS.FF (F = fotogramas por segundo). Se utiliza el valor de fotogramas para los cuepoints.

**Nota:** la propiedad FPS sólo puede establecerse mediante el panel Inspector de componentes. Si se utiliza ActionScript para definir un valor fps, éste no tendrá ningún efecto y será omitido.

## Control de componentes multimedia mediante comportamientos

Los comportamientos son scripts predefinidos de ActionScript que se añaden a una instancia de objeto, como un componente `MediaDisplay`, para controlar dicho objeto. Los comportamientos permiten añadir al documento las prestaciones, el control y la flexibilidad de la codificación de ActionScript sin que el propio usuario tenga que crear el código de ActionScript.

Para controlar un componente multimedia con un comportamiento, debe utilizar el panel Comportamientos a fin de aplicar dicho comportamiento a una instancia de componente multimedia determinada. Para ello, debe especificar el evento que activará el comportamiento (como por ejemplo, alcanzar un cuepoint específico), seleccionar un objeto de destino (los componentes multimedia que resultarán afectados por el comportamiento) y, si es necesario, seleccionar valores de configuración para el comportamiento (como el clip de película del elemento multimedia al que desplazarse).

Con Flash MX Professional 2004 se entregan los siguientes comportamientos que sirven para controlar componentes multimedia incorporados.

| Comportamiento                                    | Propósito                                                                                                                                                                                                                     | Parámetros                                                                     |
|---------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Controlador asociado                              | Asocia un componente <code>MediaController</code> con un componente <code>MediaDisplay</code>                                                                                                                                 | Nombre de instancia de los componentes <code>MediaController</code> de destino |
| Visualización asociada                            | Asocia un componente <code>MediaDisplay</code> con un componente <code>MediaController</code>                                                                                                                                 | Nombre de instancia de los componentes <code>MediaController</code> de destino |
| Navegación por cuepoints en fotogramas con nombre | Coloca una acción en una instancia de <code>MediaDisplay</code> o <code>MediaPlayback</code> que indica a un clip de película determinado que se desplace a un fotograma cuyo nombre sea igual al de un cuepoint especificado | Nombre de fotograma y nombre de cuepoint (los nombres deben ser idénticos)     |
| Navegación por cuepoints en diapositivas          | Desplaza un documento de Flash de diapositiva a una diapositiva cuyo nombre sea igual al de un cuepoint especificado                                                                                                          | Nombre de diapositiva y nombre de cuepoint (los nombres deben ser idénticos)   |

### Para asociar un componente `MediaDisplay` con un componente `MediaController`:

- 1 Coloque una instancia de `MediaDisplay` y una instancia de `MediaController` en el escenario.
- 2 Seleccione la instancia de `MediaDisplay` e introduzca el nombre de instancia **myMediaDisplay** con el inspector de propiedades.
- 3 Seleccione la instancia de `MediaController` que activará el comportamiento.
- 4 En el panel Comportamientos (Ventana > Paneles de desarrollo > Comportamientos), haga clic en el botón Añadir (+) y seleccione Medios > Visualización asociada.
- 5 En la ventana Visualización asociada, seleccione **myMediaDisplay** en **\_root** y haga clic en Aceptar.

**Nota:** si ha asociado el componente `MediaDisplay` con el componente `MediaController`, no es necesario que asocie el componente `MediaController` con el componente `MediaDisplay`.

### Para asociar el componente **MediaController** con el componente **MediaDisplay**:

- 1 Coloque una instancia de **MediaDisplay** y una instancia de **MediaController** en el escenario.
- 2 Seleccione la instancia de **MediaController** e introduzca el nombre de instancia **myMediaController** con el inspector de propiedades.
- 3 Seleccione la instancia de **MediaDisplay** que activará el comportamiento.
- 4 En el panel Comportamientos (Ventana > Paneles de desarrollo > Comportamientos), haga clic en el botón Añadir (+) y seleccione Medios > Controlador asociado.
- 5 En la ventana Controlador asociado, seleccione **myMediaController** en **\_root** y haga clic en Aceptar.

### Para utilizar un comportamiento **Navegación por cuepoints** en fotogramas con nombre:

- 1 Coloque una instancia de componente **MediaDisplay** o **MediaPlayback** en el escenario.
- 2 Seleccione el fotograma al que se desplazará el elemento multimedia e introduzca el nombre de fotograma **myLabeledFrame** con el inspector de propiedades.
- 3 Seleccione la instancia de **MediaDisplay** o de **MediaPlayback**.
- 4 En el panel Inspector de componentes, haga clic en el botón Añadir (+) e introduzca el tiempo de cuepoint en el formato HH:MM:SS:mmm o HH:MM:SS:FF y asigne el nombre **myLabeledFrame** al cuepoint.

El cuepoint indica el tiempo que debe transcurrir antes de desplazarse al fotograma seleccionado. Por ejemplo, si desea saltar al fotograma **myLabeledFrame** de la película al cabo de 5 segundos, introduzca 5 en el cuadro de texto de los segundos y **myLabeledFrame** en el cuadro de texto Nombre.

- 5 En el panel Comportamientos (Ventana > Paneles de desarrollo > Comportamientos), haga clic en el botón Añadir (+) y seleccione Medios > Navegación por cuepoints en fotogramas con nombre.
- 6 En la ventana Navegación por cuepoints en fotogramas con nombre, seleccione el clip **\_root** y haga clic en Aceptar.

### Para utilizar un comportamiento **Navegación por cuepoints** en diapositivas:

- 1 Abra el nuevo documento como presentación de diapositivas de Flash.
- 2 Coloque una instancia de componente **MediaDisplay** o **MediaPlayback** en el escenario.
- 3 En el panel Contorno de pantalla situado a la izquierda del escenario, haga clic en el botón Insertar pantalla (+) para añadir una segunda diapositiva; a continuación, seleccione la segunda diapositiva y asígnele el nombre **mySlide**.
- 4 Seleccione la instancia de **MediaDisplay** o de **MediaController**.
- 5 En el panel Inspector de componentes, haga clic en el botón Añadir (+) e introduzca el tiempo de cuepoint con el formato HH:MM:SS:mmm o HH:MM:SS:FF y asigne el nombre **MySlide** al cuepoint.

El cuepoint indica el tiempo que debe transcurrir antes de desplazarse a la diapositiva seleccionada. Por ejemplo, si desea saltar a la diapositiva **mySlide** de la película al cabo de 5 segundos, introduzca 5 en el cuadro de texto de los segundos y **mySlide** en el cuadro de texto Nombre.

- 6 En el panel Comportamientos (Ventana > Paneles de desarrollo > Comportamientos), haga clic en el botón Añadir (+) y seleccione Medios > Navegación por cuepoints en diapositivas.
- 7 En la ventana Navegación por cuepoints en diapositivas, seleccione Presentación en el clip **\_root** y haga clic en Aceptar.

## Parámetros de los componentes multimedia (sólo para Flash Professional)

En las tablas siguientes se enumeran parámetros de edición que pueden establecerse para una instancia de componente multimedia determinada en el inspector de propiedades:

### Parámetros de los componentes MediaDisplay

| Nombre                                                         | Tipo          | Valor predeterminado | Descripción                                                                                                                                                                                                                                       |
|----------------------------------------------------------------|---------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Automatically Play<br>( <a href="#">Media.autoPlay</a> )       | Booleano      | Seleccionado         | Determina si el elemento multimedia se reproduce en cuanto se carga.                                                                                                                                                                              |
| Use Preferred Media Size<br>( <a href="#">Media.autoSize</a> ) | Booleano      | Seleccionado         | Determina si el elemento multimedia asociado con la instancia de MediaDisplay se adapta al tamaño del componente o simplemente utiliza su tamaño predeterminado.                                                                                  |
| FPS                                                            | Entero        | 30                   | Indica el número de fotogramas por segundo. Si la opción Milliseconds está seleccionada, este control está desactivado.                                                                                                                           |
| Cuepoints<br>( <a href="#">Media.cuePoints</a> )               | Matriz        | No definido          | Matriz de objetos de cuepoint, cada uno con un nombre y una posición en el tiempo en un formato válido HH:MM:SS:FF (opción de milisegundos seleccionada) o HH:MM:SS:mmm.                                                                          |
| FLV o MP3<br>( <a href="#">Media.mediaType</a> )               | "FLV" o "MP3" | "FLV"                | Indica el tipo de elemento multimedia que se va a reproducir.                                                                                                                                                                                     |
| Milliseconds                                                   | Booleano      | No seleccionado      | Determina si la barra de reproducción utiliza fotogramas o milisegundos y si los cuepoints utilizan segundos o fotogramas. Si se selecciona esta opción, el control FPS no aparece.                                                               |
| URL ( <a href="#">Media.contentPath</a> )                      | Cadena        | No definido          | Cadena que contiene la ruta y el nombre de archivo del elemento multimedia que va a reproducirse.                                                                                                                                                 |
| Video Length<br>( <a href="#">Media.totalTime</a> )            | Entero        | No definido          | Tiempo total necesario para reproducir el elemento multimedia FLV. Esta configuración es necesaria para que la barra de reproducción funcione correctamente. Este control sólo es visible cuando el valor del tipo de elemento multimedia es FLV. |

## Parámetros de los componentes MediaController

| Nombre                                                           | Tipo                       | Valor predeterminado | Descripción                                                                                                                                                                |
|------------------------------------------------------------------|----------------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| activePlayControl<br>( <a href="#">Media.activePlayControl</a> ) | Cadena: "pause" o "play"   | "pause"              | Determina si la barra de reproducción está en modo de reproducción o de pausa después de la creación de instancias.                                                        |
| backgroundStyle<br>( <a href="#">Media.backgroundStyle</a> )     | Cadena: "default" o "none" | "default"            | Determina si se extraerá el fondo cromático para la instancia de MediaController.                                                                                          |
| controllerPolicy<br>( <a href="#">Media.controllerPolicy</a> )   | "auto", "on" u "off"       | "auto"               | Determina si el controlador se abre o se cierra en función de la posición del ratón, o si se bloquea una vez que está abierto o cerrado.                                   |
| horizontal<br>( <a href="#">Media.horizontal</a> )               | Booleano                   | true                 | Determina si la porción de controlador de la instancia está orientada vertical u horizontalmente. El valor true indica que el componente estará orientado horizontalmente. |
| enabled                                                          | Booleano                   | true                 | Determina si el usuario puede modificar este control. El valor true indica que el control puede ser modificado.                                                            |
| visible                                                          | Booleano                   | true                 | Determina si el usuario puede visualizar este control. El valor true indica que el control está visible.                                                                   |
| minHeight                                                        | Entero                     | 0                    | Altura mínima en píxeles permitida para esta instancia.                                                                                                                    |
| minWidth                                                         | Entero                     | 0                    | Anchura mínima en píxeles permitida para esta instancia.                                                                                                                   |

## Parámetros de los componentes MediaPlayer

| Nombre                                                          | Tipo                                      | Valor predeterminado | Descripción                                                                                                                                                                                        |
|-----------------------------------------------------------------|-------------------------------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Control Placement<br>( <a href="#">Media.controlPlacement</a> ) | "top",<br>"bottom",<br>"left",<br>"right" | "bottom"             | Posición del controlador. El valor está relacionado con la orientación.                                                                                                                            |
| <a href="#">Media.controllerPolicy</a>                          | Booleano                                  | true                 | Determina si el controlador se abre o se cierra en función de la posición del ratón.                                                                                                               |
| Automatically Play<br>( <a href="#">Media.autoPlay</a> )        | Booleano                                  | Seleccionado         | Determina si el elemento multimedia se reproduce en cuanto se carga.                                                                                                                               |
| Use Preferred Media Size<br>( <a href="#">Media.autoSize</a> )  | Booleano                                  | Seleccionado         | Determina si la instancia de MediaController cambia de tamaño para adaptarse al elemento multimedia o utiliza otra configuración.                                                                  |
| FPS                                                             | Entero                                    | 30                   | Número de fotogramas por segundo. Si la opción Milliseconds está seleccionada, este control está desactivado.                                                                                      |
| Cuepoints<br>( <a href="#">Media.cuePoints</a> )                | Matriz                                    | No definido          | Matriz de objetos de cuepoint, cada uno con un nombre y una posición en el tiempo en un formato válido HH:MM:SS:mm (opción de milisegundos seleccionada) o HH:MM:SS:FF.                            |
| FLV o MP3<br>( <a href="#">Media.mediaType</a> )                | "FLV" o<br>"MP3"                          | "FLV"                | Indica el tipo de elemento multimedia que se va a reproducir.                                                                                                                                      |
| Milliseconds                                                    | Booleano                                  | No seleccionado      | Determina si la barra de reproducción utiliza fotogramas o milisegundos, y si los cuepoints utilizan segundos o fotogramas. Cuando esta opción está seleccionada, el control FPS está desactivado. |
| URL ( <a href="#">Media.contentPath</a> )                       | Cadena                                    | No definido          | Cadena que contiene la ruta y el nombre de archivo del elemento multimedia que va a reproducirse.                                                                                                  |
| Video Length<br>( <a href="#">Media.totalTime</a> )             | Entero                                    | No definido          | Tiempo total necesario para reproducir el elemento multimedia FLV. Esta configuración es necesaria para que la barra de reproducción funcione correctamente.                                       |

## Creación de aplicaciones con componentes multimedia (sólo para Flash Professional)

La creación de contenido de Flash mediante componentes multimedia es bastante fácil y suele requerir pocos pasos.

Este ejemplo muestra la manera de crear una aplicación para reproducir un pequeño archivo multimedia disponible públicamente.

### Para añadir un componente multimedia a una aplicación:

- 1 En Flash, seleccione Archivo > Nuevo; a continuación, seleccione Documento de Flash.
- 2 En el panel Componentes (Ventana > Paneles de desarrollo > Componentes), haga doble clic en el componente MediaPlayer para añadirlo al escenario.
- 3 En el inspector de propiedades, introduzca el nombre de instancia **myMedia**.
- 4 En el inspector de propiedades, haga clic en Iniciar inspector de componentes.
- 5 En el panel Inspector de componentes, introduzca **http://www.cathphoto.com/c.flv** en el cuadro de texto URL.
- 6 Seleccione Control > Probar película, para ver la reproducción del elemento multimedia.

## Personalización de los componentes multimedia (sólo para Flash Professional)

Si desea cambiar la apariencia de los componentes multimedia, puede aplicarles aspectos. Para obtener una guía completa sobre la personalización de componentes, consulte el [Capítulo 3, “Personalización de componentes”](#), en la [página 31](#).

### Utilización de estilos con componentes multimedia

No se admiten estilos con los componentes multimedia.

### Utilización de aspectos con componentes multimedia

Aunque los componentes multimedia no admiten la aplicación dinámica de aspectos, puede abrir el documento de origen de los componentes multimedia y cambiar sus activos para conseguir el aspecto deseado. Es aconsejable realizar una copia de este archivo para trabajar; de este modo, podrá disponer del documento de origen instalado en caso necesario. Puede encontrar el documento de origen del componente multimedia en las ubicaciones siguientes:

- Windows: C:\Documents and Settings\*usuario*\Configuración local\Datos de programa\Macromedia\Flash MX2004\*idioma*\Configuration\ComponentFLA fla
- Macintosh Disco duro: Users:*nombreusuario*:Library:Application Support:Macromedia:Flash MX 2004:*idioma*:Configuration:ComponentFLA fla

Para más información sobre los aspectos de componente, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).



## Clase Media (sólo para Flash Professional)

**Herencia** mx.core.UIComponent

**Nombres de clase de ActionScript** mx.controls.MediaController, mx.controls.MediaDisplay, mx.controls.MediaPlayback

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles para la propia clase. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.MediaPlayback.version);
```

**Nota:** el código `trace(myMediaInstance.version);` devuelve `undefined`.

### Resumen de métodos para la clase Media

| Método                                   | Componentes                 | Descripción                                                                                            |
|------------------------------------------|-----------------------------|--------------------------------------------------------------------------------------------------------|
| <code>Media.addCuePoint()</code>         | MediaDisplay, MediaPlayback | Añade un objeto de cuepoint a la instancia de componente.                                              |
| <code>Media.associateController()</code> | MediaDisplay                | Asocia una instancia de MediaDisplay con una instancia de MediaController.                             |
| <code>Media.associateDisplay()</code>    | MediaController             | Asocia una instancia de MediaController con una instancia de MediaDisplay.                             |
| <code>Media.displayFull()</code>         | MediaPlayback               | Convierte la instancia de componente al modo de reproducción de pantalla completa.                     |
| <code>Media.displayNormal()</code>       | MediaPlayback               | Convierte la instancia de componente al tamaño de pantalla original.                                   |
| <code>Media.getCuePoint()</code>         | MediaDisplay, MediaPlayback | Devuelve un objeto de cuepoint.                                                                        |
| <code>Media.play()</code>                | MediaDisplay, MediaPlayback | Reproduce el elemento multimedia asociado con la instancia de componente en un punto inicial concreto. |
| <code>Media.pause()</code>               | MediaDisplay, MediaPlayback | Detiene la cabeza lectora en el punto actual de la línea de tiempo del elemento multimedia.            |
| <code>Media.removeAllCuePoints()</code>  | MediaDisplay, MediaPlayback | Elimina todos los objetos de cuepoint asociados con una instancia de componente determinada.           |
| <code>Media.removeCuePoint()</code>      | MediaDisplay, MediaPlayback | Elimina un cuepoint concreto asociado con una instancia de componente determinada.                     |
| <code>Media.setMedia()</code>            | MediaDisplay, MediaPlayback | Establece el tipo de elemento multimedia y la ruta del tipo de elemento multimedia especificado.       |
| <code>Media.stop()</code>                | MediaDisplay, MediaPlayback | Detiene la cabeza lectora y la desplaza a la posición 0, que es el principio del elemento multimedia.  |

## Resumen de propiedades para la clase Media

| Propiedad                            | Componentes                       | Descripción                                                                                                                                                                                                          |
|--------------------------------------|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Media.activePlayControl</code> | MediaController                   | Determina el estado del componente cuando éste se carga en tiempo de ejecución.                                                                                                                                      |
| <code>Media.aspectRatio</code>       | MediaDisplay,<br>MediaPlayback    | Determina si la instancia de componente mantiene su proporción de vídeo.                                                                                                                                             |
| <code>Media.autoPlay</code>          | MediaDisplay,<br>MediaPlayback    | Determina si la instancia de componente comenzará inmediatamente a almacenarse en el búfer y a reproducirse.                                                                                                         |
| <code>Media.autoSize</code>          | MediaDisplay,<br>MediaPlayback    | Determina qué tamaño adopta la porción de visualización de elemento multimedia del componente MediaDisplay o MediaPlayback.                                                                                          |
| <code>Media.backgroundColor</code>   | MediaController                   | Determina si la instancia de componente dibuja su fondo cromático.                                                                                                                                                   |
| <code>Media.bytesLoaded</code>       | MediaDisplay,<br>MediaPlayback    | Número de bytes cargados disponibles para reproducir.                                                                                                                                                                |
| <code>Media.bytesTotal</code>        | MediaDisplay,<br>MediaPlayback    | Número de bytes que se van a cargar en la instancia de componente.                                                                                                                                                   |
| <code>Media.contentPath</code>       | MediaDisplay,<br>MediaPlayback    | Cadena que contiene la ruta relativa y el nombre de archivo del elemento multimedia que va a reproducirse sin interrupción o con interrupción.                                                                       |
| <code>Media.controllerPolicy</code>  | MediaController,<br>MediaPlayback | Determina si los controles del componente están ocultos durante la reproducción y sólo se muestran al activarse un evento de pasar el ratón por encima, o si los controles están visibles u ocultos en todo momento. |
| <code>Media.controlPlacement</code>  | MediaPlayback                     | Determina la posición de los controles del componente con relación a este último.                                                                                                                                    |
| <code>Media.cuePoints</code>         | MediaDisplay,<br>MediaPlayback    | Matriz de objetos de cuepoint que se han asignado a una instancia de componente determinada.                                                                                                                         |
| <code>Media.horizontal</code>        | MediaController                   | Determina la orientación de la instancia de componente.                                                                                                                                                              |
| <code>Media.mediaType</code>         | MediaDisplay,<br>MediaPlayback    | Determina el tipo de elemento multimedia que va a reproducirse.                                                                                                                                                      |
| <code>Media.playheadTime</code>      | MediaDisplay,<br>MediaPlayback    | Contiene la posición actual de la cabeza lectora (en segundos) en la línea de tiempo del elemento multimedia que se está reproduciendo.                                                                              |
| <code>Media.playing</code>           | MediaDisplay,<br>MediaPlayback    | Devuelve una valor booleano que indica si una determinada instancia de componente está reproduciendo un elemento multimedia.                                                                                         |
| <code>Media.preferredHeight</code>   | MediaDisplay,<br>MediaPlayback    | Valor predeterminado de la altura de un archivo multimedia FLV.                                                                                                                                                      |

| Propiedad                         | Componentes                    | Descripción                                                              |
|-----------------------------------|--------------------------------|--------------------------------------------------------------------------|
| <code>Media.preferredWidth</code> | MediaDisplay,<br>MediaPlayback | Valor predeterminado de la anchura de un archivo multimedia FLV.         |
| <code>Media.totalTime</code>      | MediaDisplay,<br>MediaPlayback | Entero que indica la longitud total en segundos del elemento multimedia. |
| <code>Media.volume</code>         | MediaDisplay,<br>MediaPlayback | Entero de 0 (mínimo) a 100 (máximo) que representa el nivel de volumen.  |

## Resumen de eventos para la clase Media

| Evento                            | Componentes                       | Descripción                                                                                                                                                   |
|-----------------------------------|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Media.change</code>         | MediaDisplay,<br>MediaPlayback    | Se difunde continuamente mientras se reproduce el elemento multimedia.                                                                                        |
| <code>Media.click</code>          | MediaController,<br>MediaPlayback | Se difunde cuando el usuario hace clic en el botón Reproducir/Pausar.                                                                                         |
| <code>Media.complete</code>       | MediaDisplay,<br>MediaPlayback    | Notificación de que la cabeza lectora ha alcanzado el final del elemento multimedia.                                                                          |
| <code>Media.cuePoint</code>       | MediaDisplay,<br>MediaPlayback    | Notificación de que la cabeza lectora ha alcanzado un cuepoint determinado.                                                                                   |
| <code>Media.playheadChange</code> | MediaController,<br>MediaPlayback | Difundido por la instancia de componente cuando un usuario mueve el control deslizante del reproductor o hace clic en el botón Ir al principio o Ir al final. |
| <code>Media.progress</code>       | MediaDisplay,<br>MediaPlayback    | Se genera continuamente hasta que el elemento multimedia está totalmente descargado.                                                                          |
| <code>Media.volume</code>         | MediaController,<br>MediaPlayback | Se difunde cuando el usuario ajusta el volumen.                                                                                                               |

## Media.activePlayControl

### Se aplica a

MediaController

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.activePlayControl
```

## Descripción

Propiedad; valor booleano que determina el estado en que se encuentra el componente `MediaController` al cargarse en tiempo de ejecución. `true` indica que el componente `MediaController` debe estar en estado de reproducción en tiempo de ejecución, y `false` indica que dicho componente se encuentra en estado de pausa durante el tiempo de ejecución. Esta propiedad debe establecerse junto con la propiedad `autoPlay`, de manera que ambas estén en pausa o en reproducción en tiempo de ejecución. El valor predeterminado es `true`.

## Ejemplo

En el ejemplo siguiente se indica que el control estará en pausa cuando se cargue por primera vez en tiempo de ejecución:

```
myMedia.activePlayControl = false;
```

## Véase también

[Media.autoPlay](#)

## Media.addCuePoint()

### Se aplica a

`MediaDisplay`, `MediaPlayback`

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.addCuePoint(cuePointName, cuePointTime)
```

### Parámetros

*cuePointName* Cadena que puede utilizarse para nombrar el cuepoint.

*cuePointTime* Número que indica, en segundos, cuándo se difundirá un evento `cuePoint`.

### Valor devuelto

Ninguno.

### Descripción

Método; añade un objeto de cuepoint a una instancia de componente `MediaPlayback` o `MediaDisplay`. Cuando el tiempo de la cabeza lectora y el del cuepoint son iguales, se difunde un evento `cuePoint`.

### Ejemplo

En el código siguiente se añade un cuepoint llamado `Homerun` a `myMedia` en el punto de los 16 segundos.

```
myMedia.addCuePoint("Homerun", 16);
```

## Véase también

`Media.cuePoint`, `Media.cuePoints`, `Media.getCuePoint()`, `Media.removeAllCuePoints()`, `Media.removeCuePoint()`

## Media.aspectRatio

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.aspectRatio
```

### Descripción

Propiedad; valor booleano que determina si una instancia de MediaDisplay o MediaPlayer mantiene su proporción de vídeo durante la reproducción. `true` indica que la proporción debe mantenerse; `false` indica que la proporción puede cambiar durante la reproducción. El valor predeterminado es `true`.

### Ejemplo

En el ejemplo siguiente se indica que la proporción puede cambiar durante la reproducción:

```
myMedia.aspectRatio = false;
```

## Media.associateController()

### Se aplica a

MediaDisplay

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.associateController(instanceName)
```

### Parámetros

*instanceName* Cadena que indica el nombre de instancia del componente MediaController que se va a asociar.

### Valor devuelto

Ninguno.

## Descripción

Método; asocia una instancia de componente `MediaDisplay` con una determinada instancia de `MediaController`.

Si ha asociado una instancia de `MediaController` con una instancia de `MediaDisplay` mediante `Media.associateDisplay()`, no es necesario utilizar `Media.associateController()`.

## Ejemplo

En el código siguiente se asocia `myMedia` con `myController`:

```
myMedia.associateController(myController);
```

## Véase también

[Media.associateDisplay\(\)](#)

## Media.associateDisplay()

### Se aplica a

`MediaController`

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.associateDisplay(instanceName)
```

### Parámetros

*instanceName* Cadena que indica el nombre de instancia del componente `MediaDisplay` que se va asociar.

### Valor devuelto

Ninguno.

## Descripción

Método; asocia una instancia de componente `MediaController` con una determinada instancia de `MediaDisplay`.

Si ha asociado una instancia de `MediaDisplay` con una instancia de `MediaController` mediante `Media.associateController()`, no es necesario utilizar `Media.associateDisplay()`.

## Ejemplo

En el código siguiente se asocia `myMedia` con `myDisplay`:

```
myMedia.associateDisplay(myDisplay);
```

## Véase también

[Media.associateController\(\)](#)

## Media.autoPlay

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.autoPlay
```

### Descripción

Propiedad; valor booleano que determina si la instancia de MediaPlayer o MediaDisplay comenzará inmediatamente a almacenarse en el búfer y a reproducirse. `true` indica que el control se almacenará en el búfer y se reproducirá en tiempo de ejecución; `false` indica que el control se detendrá en tiempo de ejecución. Esta propiedad depende de las propiedades `contentPath` y `mediaType`. Si `contentPath` y `mediaType` no se establecen, no habrá reproducción en tiempo de ejecución. El valor predeterminado es `true`.

### Ejemplo

En el ejemplo siguiente se indica que el control no se iniciará al cargarse por primera vez en tiempo de ejecución:

```
myMedia.autoPlay = false;
```

### Véase también

[Media.contentPath](#), [Media.mediaType](#)

## Media.autoSize

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.autoSize
```

## Descripción

Propiedad; valor booleano que determina qué tamaño adopta la porción de visualización de elemento multimedia del componente `MediaDisplay` o `MediaPlayback`.

En el componente `MediaDisplay`, la propiedad se comporta de la manera siguiente:

- Si establece el valor `true` para esta propiedad, Flash mostrará el elemento multimedia con su tamaño preferido, independientemente del tamaño del componente. Esto implica que, a menos que el tamaño de la instancia de `MediaDisplay` sea igual al tamaño de los medios, éstos rebasarán los límites de la instancia o no llenarán el tamaño de la misma.
- Si establece el valor `false`, Flash utilizará el tamaño de la instancia mientras sea posible, al tiempo que respetará la proporción. Si `Media.autoSize` y `Media.aspectRatio` se establecen como `false`, se utilizará el tamaño exacto del componente.

En el componente `MediaPlayback`, la propiedad se comporta de la manera siguiente:

- Si establece el valor `true` para esta propiedad, Flash mostrará el elemento multimedia con su tamaño preferido, siempre que el área del elemento multimedia de reproducción sea inferior al tamaño preferido. En tal caso, Flash reducirá el tamaño del elemento multimedia para adaptarlo al interior de la instancia, respetando la proporción. Si el tamaño preferido es inferior al tamaño del área del elemento multimedia de la instancia, parte de dicha área no se utilizará.
- Si establece el valor `false`, Flash utilizará el tamaño de la instancia mientras sea posible, al tiempo que respetará la proporción. Si `Media.autoSize` y `Media.aspectRatio` se establecen en `false`, se llenará el área de elemento multimedia del componente. Esta área se define como el área situada sobre los controles (en el diseño predeterminado), con un margen de 8 píxeles alrededor que forma los bordes del componente.

El valor predeterminado es `true`.

## Ejemplo

En el ejemplo siguiente se indica que el control no volverá a reproducirse en función del tamaño del elemento multimedia:

```
myMedia.autoSize = false;
```

## Véase también

[Media.aspectRatio](#)

## Media.backgroundStyle

### Se aplica a

`MediaController`

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.backgroundStyle
```



## Descripción

Propiedad; el valor "default" indica que se dibujará el fondo cromático para la instancia de MediaController, mientras que el valor "none" indica que no se dibujará el fondo cromático. El valor predeterminado es "default".

No se trata de una propiedad de estilo y, por lo tanto, no se verá afectada por la configuración de estilos.

## Ejemplo

En el ejemplo siguiente se indica que no se extraerá fondo cromático para el control:

```
myMedia.backgroundStyle = "none";
```

## Media.bytesLoaded

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.bytesLoaded
```

## Descripción

Propiedad de sólo lectura; el número de bytes ya cargados en el componente que están disponibles para su reproducción. El valor predeterminado es undefined.

## Ejemplo

En el código siguiente se crea una variable llamada PlaybackLoad que se establecerá con el número de bytes cargados en el bucle for.

```
// crear variable que indique el número de bytes cargados
var PlaybackLoad = myMedia.bytesLoaded;
// realizar alguna función hasta que la reproducción esté preparada
for (PlaybackLoad < 150) {
    someFunction();
}
```

## Media.bytesTotal

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

## Edición

Flash MX Professional 2004.

## Sintaxis

```
myMedia.bytesTotal
```

## Descripción

Propiedad; número de bytes que se van a cargar en el componente MediaPlayer o MediaDisplay. El valor predeterminado es `undefined`.

## Ejemplo

En el ejemplo siguiente se indica al usuario el tamaño del elemento multimedia que se va a reproducir sin interrupción:

```
myTextField.text = myMedia.bytesTotal;
```

## Media.change

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

## Edición

Flash MX Professional 2004.

## Sintaxis

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    // introduzca aquí el código propio  
}  
myMedia.addEventListener("change", listenerObject)
```

## Descripción

Evento; difundido por los componentes MediaDisplay y MediaPlayer mientras se reproduce el elemento multimedia. El porcentaje completo puede recuperarse de la instancia de componente. Consulte el ejemplo siguiente.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `Media.change` tiene dos propiedades adicionales:

`target` Referencia al objeto difusor.

`type` Cadena "change", que indica el tipo de evento.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente se utiliza un detector de objetos para determinar la posición de la cabeza lectora (`Media.playheadTime`), a partir de la que puede calcularse el porcentaje completo:

```
var myPlayerListener = new Object();
myPlayerListener.change = function(eventObject){
    var myPosition = myPlayer.playheadTime;
    var myPercentPosition = (myPosition/totalTime);
}
myPlayer.addEventListener("change", myPlayerListener);
```

## Véase también

[Media.playing](#), [Media.pause\(\)](#)

## Media.click

### Se aplica a

[MediaController](#), [MediaPlayback](#)

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
var myMediaListener = new Object()
myMediaListener.click = function(){
    // introduzca aquí el código propio
}
myPlayer.addEventListener("click", myMediaListener);
```

### Descripción

Evento; se difunde cuando el usuario hace clic en el botón Reproducir/Pausar. El campo de detalle debe utilizarse para determinar en qué botón se ha hecho clic. El objeto del evento `Media.click` tiene las siguientes propiedades:

`detail` Cadena "pause" o "play".

`target` Referencia a la instancia de componente `MediaController` o `MediaPlayback`.

`type` Cadena "click".

### Ejemplo

En el ejemplo siguiente se abre una ventana emergente cuando el usuario hace clic en Reproducir:

```
var myMediaListener = new Object()
myMediaListener.click = function(){
    PopUpManager.createPopup(_root, mx.containers.Window, false, {contentPath:
        movieSale});
}
myMedia.addEventListener("click", myMediaListener);
```

## Media.complete

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.complete = function(eventObject){  
    // introduzca aquí el código propio  
}  
myMedia.addEventListener("complete", listenerObject)
```

### Descripción

Evento; notificación de que la cabeza lectora ha llegado al final del elemento multimedia. El objeto del evento `Media.complete` tiene las siguientes propiedades:

`target` Referencia a la instancia del componente `MediaDisplay` o `MediaPlayer`.

`type` Cadena "complete".

### Ejemplo

En el ejemplo siguiente se utiliza un detector de objetos para determinar cuándo acaba la reproducción del elemento multimedia:

```
var myListener = new Object();  
myListener.complete = function(eventObject) {  
    trace("media is Finished");  
};  
myMedia.addEventListener("complete", myListener);
```

## Media.contentPath

### Se aplica a

MediaController

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.contentPath
```

## Descripción

Propiedad; cadena que contiene la ruta relativa y el nombre de archivo del elemento multimedia que va a reproducirse sin interrupción o con interrupción. El método `Media.setMedia()` es el único procedimiento admitido para establecer esta propiedad mediante ActionScript. El valor predeterminado es `undefined`.

## Ejemplo

En el ejemplo siguiente se muestra en un cuadro de texto el nombre de la película que se reproduce:

```
myTextField.text = myMedia.contentPath;
```

## Véase también

`Media.setMedia()`

## Media.controllerPolicy

### Se aplica a

MediaController, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.controllerPolicy
```

## Descripción

Propiedad; determina si el componente MediaController (o el subcomponente de controlador del componente MediaPlayer) está oculto cuando se crean instancias de él, y si sólo se muestra cuando el usuario pasa el ratón sobre el controlador en estado contraído.

Los valores posibles para esta propiedad son los siguientes:

- "on" indica que los controles están siempre ampliados.
- "off" indica que los controles están siempre contraídos.
- "auto" indica que el control permanecerá en estado contraído hasta que el usuario pase el ratón por el área activa. El área activa coincide con el área en la que se dibuja el control contraído. El control permanece ampliado hasta que el ratón sale del área activa.

**Nota:** el área activa se amplía y se contrae con el controlador.

## Ejemplo

En el ejemplo siguiente se mantiene el controlador abierto constantemente:

```
myMedia.controllerPolicy = "on";
```

## Media.controlPlacement

### Se aplica a

MediaPlayback

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.controlPlacement
```

### Descripción

Propiedad; determina la posición de la porción de controlador del componente MediaPlayback con relación a su visualización. Los valores posibles son "top", "bottom", "left" y "right". El valor predeterminado es "bottom".

### Ejemplo

En el ejemplo siguiente, la porción de controlador del componte MediaPlayback se sitúa en el lado derecho:

```
myMedia.controlPlacement = "right";
```

## Media.cuePoint

### Se aplica a

MediaDisplay, MediaPlayback

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.cuePoint = function(eventObject){  
    // introduzca aquí el código propio  
}  
myMedia.addEventListener("cuePoint", listenerObject)
```

## Descripción

Evento; notificación de que la cabeza lectora ha alcanzado el cuepoint. El objeto del evento `Media.cuePoint` tiene las siguientes propiedades:

`name` Cadena que indica el nombre del cuepoint.

`time` Número que indica, en fotogramas o segundos, cuándo se ha alcanzado el cuepoint.

`target` Referencia al objeto de cuepoint.

`type` Cadena "cuePoint".

## Ejemplo

En el ejemplo siguiente se utiliza un detector de objetos para determinar cuándo se ha alcanzado un cuepoint:

```
var myCuePointListener = new Object();
myCuePointListener.cuePoint = function(eventObject){
    trace("heard " + eventObject.type + ", " + eventObject.target);
}
myPlayback.addEventListener("cuePoint", myCuePointListener);
```

## Véase también

[Media.addCuePoint\(\)](#), [Media.cuePoints](#), [Media.getCuePoint\(\)](#)

## Media.cuePoints

### Se aplica a

`MediaDisplay`, `MediaPlayback`

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myMedia.cuePoints[N]`

## Descripción

Propiedad; matriz de objetos de cuepoint asignados a una instancia del componente `MediaPlayback` o `MediaDisplay`. Dentro de la matriz, cada objeto de cuepoint puede tener un nombre, un tiempo en segundos o fotogramas y una propiedad de reproductor (que es el nombre de instancia del componente con el que está asociado). El valor predeterminado es una matriz vacía `[]`.

## Ejemplo

En el ejemplo siguiente se elimina el tercer cuepoint si se reproduce la vista previa de una acción:

```
if(myVariable == actionPreview) {
    myMedia.removeCuePoint(myMedia.cuePoints[2]);
}
```

### Véase también

`Media.addCuePoint()`, `Media.getCuePoint()`, `Media.removeCuePoint()`

## Media.displayFull()

### Se aplica a

MediaPlayback

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.displayFull()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; establece el modo de pantalla completa para la instancia del componente MediaPlayback. Es decir, el componente se amplía para llenar todo el escenario. Para que el componente recupere su tamaño normal, utilice `Media.displayNormal()`.

### Ejemplo

En el código siguiente se fuerza la ampliación del componente para que ocupe el escenario:

```
myMedia.displayFull();
```

### Véase también

`Media.displayNormal()`

## Media.displayNormal()

### Se aplica a

MediaPlayback

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.displayNormal()
```



### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; restablece el tamaño normal de la instancia de MediaPlayer tras haberse utilizado el método `Media.displayFull()`.

### Ejemplo

En el código siguiente se restablece el tamaño original de un componente MediaPlayer:

```
myMedia.displayNormal();
```

### Véase también

[Media.displayFull\(\)](#)

## Media.getCuePoint()

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.getCuePoint(cuePointName)
```

### Parámetros

Ninguno.

### Valor devuelto

*cuePointName* Cadena proporcionada cuando se utilizó `Media.addCuePoint()`.

### Descripción

Método; devuelve un objeto de cuepoint por su nombre de cuepoint.

### Ejemplo

En el código siguiente se recupera un cuepoint llamado `myCuePointName`.

```
myMedia.removeCuePoint(myMedia.getCuePoint("myCuePointName"));
```

### Véase también

[Media.addCuePoint\(\)](#), [Media.cuePoint](#), [Media.cuePoints](#), [Media.removeCuePoint\(\)](#)

## Media.horizontal

### Se aplica a

MediaController

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.horizontal
```

### Descripción

Propiedad; determina si el componente MediaController se mostrará en posición vertical u horizontal. `true` indica que el componente se mostrará en posición horizontal; `false` indica la posición vertical. Si se establece el valor `false`, la cabeza lectora y el indicador del progreso de carga se desplazan de abajo arriba. El valor predeterminado es `true`.

### Ejemplo

En el ejemplo siguiente se muestra el componente MediaController en posición vertical:

```
myMedia.horizontal = false;
```

## Media.mediaType

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.mediaType
```

### Descripción

Propiedad; contiene el valor del tipo de elemento multimedia que se va a reproducir. Las opciones son: FLV y MP3. El valor predeterminado es "FLV". Véase “Importación de archivos FLV (Macromedia Flash Video)” en el apartado Utilización de Flash de la Ayuda.

### Ejemplo

En el ejemplo siguiente se determina el tipo de elemento multimedia que se reproduce actualmente:

```
var currentMedia = myMedia.mediaType;
```

**Véase también**

`Media.setMedia()`

## **Media.pause()**

**Se aplica a**

MediaDisplay, MediaPlayer

**Disponibilidad**

Flash Player 7.

**Edición**

Flash MX Professional 2004.

**Sintaxis**

```
myMedia.pause()
```

**Parámetros**

Ninguno.

**Valor devuelto**

Nada.

**Descripción**

Método; detiene la cabeza lectora en la ubicación actual.

**Ejemplo**

En el código siguiente se detiene la reproducción.

```
myMedia.pause();
```

## **Media.play()**

**Se aplica a**

MediaDisplay, MediaPlayer

**Disponibilidad**

Flash Player 7.

**Edición**

Flash MX Professional 2004.

**Sintaxis**

```
myMedia.play(startingPoint)
```

**Parámetros**

*startingPoint* Valor entero no negativo que indica el punto inicial (en segundos) en el que el elemento multimedia debe empezar a reproducirse.

## Valor devuelto

Ninguno.

## Descripción

Método; reproduce el elemento multimedia asociado con la instancia de componente en el punto inicial especificado. El valor predeterminado es el valor actual de `playheadTime`.

## Ejemplo

En el código siguiente se indica que el componente multimedia debe empezar a reproducirse en el segundo 120:

```
myMedia.play(120);
```

## Véase también

[Media.pause\(\)](#)

## Media.playheadChange

### Se aplica a

MediaController, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.playheadChange = function(eventObject){  
    // introduzca aquí el código propio  
}  
myMedia.addEventListener("playheadChange", listenerObject)
```

### Descripción

Evento; difundido por el componente MediaController o MediaPlayer cuando el usuario mueve el deslizador del reproductor o hace clic en el botón Ir al principio o Ir al final. El objeto del evento `Media.playheadChange` tiene las siguientes propiedades:

**detail** Número que indica qué porcentaje del elemento multimedia se ha reproducido.

**type** Cadena "playheadChange".

### Ejemplo

En el ejemplo siguiente se envía el porcentaje reproducido al panel Salida cuando el usuario deja de arrastrar la cabeza lectora:

```
var controlListen = new Object();  
controlListen.playheadChange = function(eventObject){  
    trace(eventObject.detail);  
}  
myMedia.addEventListener("playheadChange", controlListen);
```

## Media.playheadTime

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myMedia.playheadTime`

### Descripción

Propiedad; contiene la posición actual de la cabeza lectora (en segundos) en la línea de tiempo del elemento multimedia que se está reproduciendo. El valor predeterminado es la ubicación de la cabeza lectora.

### Ejemplo

En el ejemplo siguiente se establece una variable con la ubicación de la cabeza lectora, expresada en segundos:

```
var myPlayhead = myMedia.playheadTime;
```

## Media.playing

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myMedia.playing`

### Descripción

Propiedad de sólo lectura; devuelve un valor booleano que indica si se está reproduciendo el elemento multimedia. `true` indica que el elemento multimedia se está reproduciendo; `false` indica que el usuario ha detenido la reproducción.

### Ejemplo

En el código siguiente se determina si el elemento multimedia se está reproduciendo o si se ha detenido:

```
if(myMedia.playing == true){  
    some function;  
}
```

**Véase también**

[Media.change](#)

## Media.preferredHeight

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myMedia.preferredHeight`

### Descripción

Propiedad; se establece en función del valor de altura predeterminado de un FLV. Esta propiedad se aplica sólo a los elementos multimedia FLV, dado que la altura de los archivos MP3 es fija. Esta propiedad puede utilizarse para establecer los parámetros de altura y anchura (más un margen para el propio componente). El valor predeterminado es `undefined` si no se define un elemento multimedia FLV.

### Ejemplo

En el ejemplo siguiente se establece el tamaño de una instancia de MediaPlayer en función de la instancia que se está reproduciendo y cuenta con el margen de píxeles necesario para la instancia de componente:

```
if(myPlayback.contentPath != undefined){
    var mediaHeight = myPlayback.preferredHeight;
    var mediaWidth = myPlayback.preferredWidth;
    myPlayback.setSize((mediaWidth + 20), (mediaHeight + 70));
}
```

## Media.preferredWidth

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myMedia.preferredWidth`

## Descripción

Propiedad; se establece en función del valor de achura predeterminado de un FLV. El valor predeterminado es `undefined`.

## Ejemplo

En el ejemplo siguiente se establece la achura deseada de la variable `mediaWidth`:

```
var mediaWidth = myMedia.preferredWidth;
```

## Media.progress

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.progress = function(eventObject){  
    // introduzca aquí el código propio  
}  
myMedia.addEventListener("progress", listenerObject)
```

### Descripción

Evento; se genera continuamente hasta que el elemento multimedia se ha descargado completamente. El objeto del evento `Media.progress` tiene las siguientes propiedades:

`target` Referencia a la instancia del componente MediaDisplay o MediaPlayer.

`type` Cadena "progress".

### Ejemplo

En el ejemplo siguiente se detecta el progreso:

```
var myProgressListener = new Object();  
myProgressListener.progress = function(){  
    // lightMovieClip debe parpadear durante el progreso  
    var lightVisible = lightMovieClip.visible;  
    lightMovieClip.visible = !lightVisible;  
}
```

## Media.removeAllCuePoints()

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

**Edición**

Flash MX Professional 2004.

**Sintaxis**

```
myMedia.removeAllCuePoints()
```

**Parámetros**

Ninguno.

**Valor devuelto**

Ninguno.

**Descripción**

Método; elimina todos los objetos de cuepoint asociados con una instancia de componente.

**Ejemplo**

En el código siguiente se eliminan todos los objetos de cuepoint:

```
myMedia.removeAllCuePoints();
```

**Véase también**

[Media.addCuePoint\(\)](#), [Media.cuePoints](#), [Media.removeCuePoint\(\)](#)

**Media.removeCuePoint()****Se aplica a**

MediaDisplay, MediaPlayer

**Disponibilidad**

Flash Player 7.

**Edición**

Flash MX Professional 2004.

**Sintaxis**

```
myMedia.removeCuePoint(cuePoint)
```

**Parámetros**

*cuePoint* Referencia a un objeto de cuepoint asignado previamente mediante [Media.addCuePoint\(\)](#).

**Valor devuelto**

Ninguno.

**Descripción**

Método; elimina un cuepoint específico asociado con una instancia de componente.



### Ejemplo

En el código siguiente se elimina un cuepoint llamado `myCuePoint`:

```
myMedia.removeCuePoint(getCuePoint("myCuePoint"));
```

### Véase también

```
Media.addCuePoint(), Media.cuePoints, Media.removeAllCuePoints()
```

## Media.setMedia()

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.setMedia(contentPath, mediaType)
```

### Parámetros

*contentPath* Cadena que indica la ruta y el nombre de archivo del elemento multimedia que se va a reproducir.

*mediaType* Cadena utilizada para definir el tipo de elemento multimedia como FLV o MP3. Este parámetro es opcional.

### Valor devuelto

Ninguno.

### Descripción

Método; define mediante un argumento URL el tipo de elemento multimedia y la ruta del tipo de elemento multimedia especificado. El valor predeterminado de *contentPath* es que no esté definido.

Este método es el único procedimiento admitido para establecer la ruta del contenido y el tipo de elemento multimedia para los componentes MediaPlayer y MediaDisplay.

### Ejemplo

En el código siguiente se proporciona un elemento multimedia nuevo para una instancia de componente.

```
myMedia.setMedia("http://www.RogerMoore.com/moonraker.flv", "FLV");
```

## Media.stop()

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.stop()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; detiene la cabeza lectora y la desplaza a la posición 0, que es el principio del elemento multimedia.

### Ejemplo

En el código siguiente se detiene la cabeza lectora y se desplaza al punto de tiempo 0.

```
myMedia.stop()
```

## Media.totalTime

### Se aplica a

MediaDisplay, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.totalTime
```

## Descripción

Propiedad; longitud total del elemento multimedia, expresada en segundos. Dado que el formato de archivo FLV no proporciona su tiempo de reproducción a un componente multimedia hasta que se ha cargado completamente, es necesario introducir manualmente `Media.totalTime` para que el deslizador del reproductor pueda reflejar de forma precisa el tiempo de reproducción real del elemento multimedia. El valor predeterminado de los archivos MP3 es el tiempo de reproducción del elemento multimedia. En los archivos FLV, el valor predeterminado es `undefined`.

Esta propiedad no puede establecerse para los archivos MP3, ya que la información se encuentra en el objeto `Sound`.

## Ejemplo

En el ejemplo siguiente se establece el tiempo de reproducción del elemento multimedia, expresado en segundos:

```
myMedia.totalTime = 151;
```

## Media.volume

### Se aplica a

`MediaDisplay`, `MediaPlayer`

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMedia.volume
```

### Descripción

Propiedad; almacena el valor entero de la configuración del volumen, que puede oscilar entre 0 y 100. El valor predeterminado de esta propiedad es 75.

### Ejemplo

En el ejemplo siguiente se establece el volumen máximo de reproducción del elemento multimedia:

```
myMedia.volume = 100;
```

### Véase también

[Media.volume](#), [Media.pause\(\)](#)

## Media.volume

### Se aplica a

MediaController, MediaPlayer

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.volume = function(eventObject){  
    // introduzca aquí el código propio  
}  
myMedia.addEventListener("volume", listenerObject)
```

### Descripción

Evento; se difunde cuando el usuario ajusta el valor del volumen. El objeto del evento `Media.volume` tiene las siguientes propiedades:

`detail` Valor entero entre 0 y 100 que representa el nivel del volumen.

`type` Cadena "volume".

### Ejemplo

En el ejemplo siguiente se notifica al usuario que se ajusta el volumen:

```
var myVolListener = new Object();  
myVolListener.volume = function(){  
    mytextfield.text = "Volume adjusted!";  
}  
myMedia.addEventListener("volume", myVolListener);
```

### Véase también

[Media.volume](#)

## Componente Menu (sólo para Flash Professional)

El componente Menu permite a los usuarios seleccionar un elemento de un menú emergente, como por ejemplo, el menú Archivo o Edición de la mayoría de las aplicaciones de software.

Normalmente, un menú se abre en una aplicación cuando un usuario pasa el ratón sobre un activador de menús con forma de botón o hace clic en éste. También puede configurar en un script que el componente Menu se abra cuando un usuario presione una tecla determinada.

Los componentes Menu se crean siempre de forma dinámica en tiempo de ejecución. Debe añadir el componente al documento desde el panel Componentes, y eliminarlo para añadirlo a la biblioteca. A continuación, utilice el código siguiente para crear un menú con ActionScript:

```
var myMenu = mx.controls.Menu.createMenu(parent, menuDataProvider);
```

Utilice el código siguiente para abrir un menú en una aplicación:

```
myMenu.show(x, y);
```

Un evento `menuShow` se difunde a todos los detectores de la instancia `Menu` inmediatamente antes de que el menú se presente, para poder actualizar el estado de los elementos de menú. Del mismo modo, inmediatamente después de ocultarse una instancia `Menu`, se difunde un evento `menuHide`.

Los elementos de un menú se describen mediante XML. Para más información, consulte [“Componente Menu: vista y datos” en la página 395](#).

Los lectores de pantalla no pueden acceder al componente `Menu`.

## Interacción con el componente `Menu` (sólo para Flash Professional)

Puede utilizar el ratón y el teclado para interactuar con un componente `Menu`.

Tras abrir un `Menu`, éste permanece visible hasta que se cierra mediante un script o hasta que el usuario hace clic con el ratón fuera del menú o en un elemento activado.

Al hacer clic se selecciona un elemento de menú, excepto los siguientes tipos de elementos de menú:

- Elementos desactivados o separadores    Pasar el ratón por ellos o hacer clic no tiene ningún efecto (el menú permanece visible).
- Anclajes para un submenú    Si se pasa el ratón por ellos el submenú se activa; los clics no tienen ningún efecto; si se pasa el ratón por cualquier otro elemento del submenú, éste se cierra.

Al seleccionar un elemento, se envía un evento `Menu.change` a todos los detectores del menú, el menú se oculta y tienen lugar las siguientes acciones en función del tipo de elemento:

- `check`    Se activa o desactiva el atributo `selected` del elemento.
- `radio`    El elemento pasa a ser la selección actual del grupo de opciones correspondiente.

Al mover el ratón, se activan los eventos `Menu.rollOut` y `Menu.rollOver`.

Al presionar el ratón fuera del menú, se cierra el menú y se activa un evento `Menu.menuHide`.

Al liberar el ratón en un elemento activado, los tipos de elemento se ven afectados como se indica a continuación:

- `check`    Se activa o desactiva el atributo `selected` del elemento.
- `radio`    El atributo `selected` del elemento se establece en `true`, y el atributo `selected` del elemento previamente seleccionado en el grupo de opciones se establece en `false`. La propiedad `selection` del objeto de grupo de opciones correspondiente se establece con un valor que hace referencia al elemento de menú seleccionado.
- Elemento no definido y nivel superior de un menú jerárquico    Se activa o desactiva la visibilidad del menú jerárquico.

Cuando se selecciona una instancia Menu al hacer clic en ella o mediante la tabulación, puede utilizar las teclas siguientes para controlarla:

| Tecla                         | Descripción                                                                                                                                                 |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Flecha abajo<br>Flecha arriba | Mueve la selección hacia abajo y hacia arriba por las filas del menú. La selección pasa de la última fila a la primera y viceversa.                         |
| Flecha derecha                | Abre un submenú o desplaza la selección al menú siguiente de una barra de menús (si existe una).                                                            |
| Flecha izquierda              | Cierra un submenú y vuelve a seleccionar el menú principal (si existe uno), o desplaza la selección al menú anterior de una barra de menús (si existe una). |
| Intro                         | Abre un submenú, o hace clic y suelta en una fila si no existe ningún submenú.                                                                              |

**Nota:** si un menú está abierto, puede presionar la tecla de tabulación para salir de éste. Debe realizar una selección o cerrar el menú presionando la tecla Escape.

## Utilización del componente Menu (sólo para Flash Professional)

Puede utilizar el componente Menu para crear menús de opciones seleccionables individualmente, como el menú Archivo o Edición de la mayoría de las aplicaciones de software. También puede utilizar el componente Menu para crear menús contextuales que aparecen cuando un usuario presiona una zona interactiva o una tecla modificadora. Utilice el componente Menu con el componente MenuBar para crear una barra de menús horizontal con menús que se amplían bajo cada elemento de la barra de menús.

Al igual que los menús de escritorio estándar, el componente Menu admite elementos de menú cuyas funciones se dividen en las siguientes categorías generales:

**Activadores de comandos** Estos elementos activan eventos; estos eventos se gestionan con código.

**Anclajes de submenús** Estos elementos son anclajes que abren submenús.

**Botones de opción** Estos elementos funcionan en grupos; sólo pueden seleccionarse de uno en uno.

**Elementos de casilla de verificación** Estos elementos representan un valor booleano (`true` o `false`).

**Separadores** Estos elementos proporcionan una línea horizontal sencilla que divide los elementos de un menú en distintos grupos visuales.

## Componente Menu: vista y datos

Conceptualmente, el componente Menu está compuesto por un modelo de datos y una vista que muestra los datos. La clase Menu es la vista y contiene los métodos de configuración visual. La [Clase MenuDataProvider](#) añade métodos al objeto prototipo XML global (de modo muy parecido a como lo hace la clase DataProvider con el objeto Array); estos métodos permiten construir externamente proveedores de datos y añadirlos a varias instancias de menú. El proveedor de datos difunde los cambios realizados a todas las vistas de cliente. (Véase [“Clase MenuDataProvider” en la página 418.](#))

Una instancia Menu es una colección jerárquica de elementos XML que se corresponde con elementos de menú individuales. Los atributos definen el comportamiento y el aspecto del elemento de menú correspondiente que aparece en la pantalla. La colección puede convertirse fácilmente a XML o de XML, que sirve para describir menús (la etiqueta menu) y elementos (la etiqueta menuitem). La clase XML incorporada de ActionScript es la base del modelo subyacente del componente Menu.

Un menú sencillo de dos elementos puede describirse en XML con dos subelementos de menú:

```
<menu>
  <menuitem label="Up" />
  <menuitem label="Down" />
</menu>
```

**Nota:** los nombres de etiqueta de los nodos XML (menu y menuitem) no son importantes; en el menú se utilizan los atributos y sus relaciones de anidación.

## Menús jerárquicos

Para crear menús jerárquicos, incorpore elementos XML en un elemento XML de nivel superior, de la siguiente manera:

```
<menu>
  <menuitem label="MenuItem A" >
    <menuitem label="SubMenuItem 1-A" />
    <menuitem label="SubMenuItem 2-A" />
  </menuitem>
  <menuitem label="MenuItem B" >
    <menuitem label="SubMenuItem 1-B" />
    <menuitem label="SubMenuItem 2-B" />
  </menuitem>
</menu>
```

**Nota:** esto convierte al elemento de menú principal en un anclaje del menú emergente, que no genera eventos cuando se selecciona.

## Atributos XML de elementos de menú

Los atributos de un elemento XML de elemento de menú determinan el contenido que se va visualizar, el comportamiento del elemento de menú y cómo se expone en ActionScript. En la tabla siguiente se describen los atributos de un elemento de menú XML:

| Nombre de atributo | Tipo                                                        | Valor predeterminado | Descripción                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|-------------------------------------------------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| label              | Cadena                                                      | undefined            | Texto que se visualiza para representar un elemento de menú. Este atributo es necesario para todos los tipos de elementos, excepto los separadores.                                                                                                                                                                                                               |
| type               | separator, check, radio, normal o no definido               | undefined            | Tipo de elemento de menú: separator, check box, radio button o normal (un activador de comandos o submenús). Si este atributo no existe, el valor predeterminado es normal.                                                                                                                                                                                       |
| icon               | Cadena                                                      | undefined            | Identificador de vinculación de un activo de imagen. Este atributo no es necesario. Este atributo no está disponible para los tipos check, radio o separator.                                                                                                                                                                                                     |
| instanceName       | Cadena                                                      | undefined            | Identificador que puede utilizarse para hacer referencia a la instancia del elemento de menú desde la instancia del menú raíz. Por ejemplo, puede hacerse referencia a un elemento de menú llamado <i>yellow</i> como <i>myMenu.yellow</i> . Este atributo no es necesario.                                                                                       |
| groupName          | Cadena                                                      | undefined            | Identificador que puede utilizarse para asociar varios elementos de botón de opción en un grupo de opciones y para exponer el estado de un grupo de opciones desde la instancia del menú raíz. Por ejemplo, puede hacerse referencia a un grupo de opciones llamado <i>colors</i> como <i>myMenu.colors</i> . Este atributo sólo es necesario para el tipo radio. |
| selected           | false, true o false o true (una cadena o un valor booleano) | false                | Valor booleano que indica si un elemento check o radio está activado (true) o desactivado (false). Este atributo no es necesario.                                                                                                                                                                                                                                 |
| enabled            | false, true o false o true (una cadena o un valor booleano) | true                 | Valor booleano que indica si este elemento de menú puede seleccionarse (true) o no (false). Este atributo no es necesario.                                                                                                                                                                                                                                        |



## Tipos de elementos de menú

Hay cuatro tipos de elementos de menú, especificados por el atributo `type`:

```
<menu>
  <menuitem label="Normal item" />
  <menuitem type="separator" />
  <menuitem label="CheckBox Item" type="check" instanceName="check_1"/>
  <menuitem label="RadioButton Item" type="radio" groupName="radioGroup_1" /
>
</menu>
```

### Elementos de menú normales

El elemento de menú `normal` no tiene un atributo `type`, lo cual significa que el atributo `type` tiene el valor predeterminado `normal`. Los elementos normales pueden ser activadores de comandos o activadores de submenús, según si tienen subelementos anidados.

### Elementos de menú de separador

Los elementos de menú cuyo atributo `type` es `separator` actúan como divisores visuales de un menú. En el siguiente XML se crean tres elementos de menú, `Top`, `Middle` y `Bottom`, con separadores entre ellos:

```
<menu>
  <menuitem label="Top" />
  <menuitem type="separator" />
  <menuitem label="Middle" />
  <menuitem type="separator" />
  <menuitem label="Bottom" />
</menu>
```

Todos los elementos separadores están desactivados. Hacer clic en un elemento separador o pasar el cursor sobre éste no tiene efecto alguno.

### Elementos de menú de casilla de verificación

Los elementos de menú cuyo atributo `type` es `check` actúan como elementos de casilla de verificación en el menú. Si el atributo `selected` tiene el valor `true`, aparece una marca de verificación junto a la etiqueta del elemento de menú. Cuando un elemento de casilla de verificación se selecciona, su estado cambia automáticamente y se difunde un evento `change` a todos los detectores del menú raíz. En el ejemplo siguiente se definen tres elementos de menú de casillas de verificación:

```
<menu>
  <menuitem label="Apples" type="check" instanceName="buyApples"
    selected="true" />
  <menuitem label="Oranges" type="check" instanceName="buyOranges"
    selected="false" />
  <menuitem label="Bananas" type="check" instanceName="buyBananas"
    selected="false" />
</menu>
```

Se pueden utilizar los nombres de instancia de ActionScript para tener acceso a los elementos de menú directamente desde el propio menú, como se indica en el ejemplo siguiente:

```
myMenu.setMenuItemSelected(myMenu.buyapples, true);
myMenu.setMenuItemSelected(myMenu.buyoranges, false);
```

**Nota:** el atributo `selected` sólo debe modificarse mediante el método `setMenuItemSelected(item, b)`. Se puede examinar directamente el atributo `selected`, pero devuelve el valor de cadena `true` o `false`.

## Elementos de menú de botón de opciones

Los elementos de menú cuyo atributo `type` es `radio` pueden agruparse de modo que pueda seleccionarse sólo un elemento cada vez. Un grupo de opciones se crea indicando el mismo valor en el atributo `groupName` de los elementos de menú, como en el ejemplo siguiente:

```
<menu>
  <menuitem label="Center" type="radio" groupName="alignment_group"
    instanceName="center_item"/>
  <menuitem type="separator" />
  <menuitem label="Top" type="radio" groupName="alignment_group" />
  <menuitem label="Bottom" type="radio" groupName="alignment_group" />
  <menuitem label="Right" type="radio" groupName="alignment_group" />
  <menuitem label="Left" type="radio" groupName="alignment_group" />
</menu>
```

Cuando el usuario selecciona uno de los elementos, la selección actual cambia automáticamente y se difunde un evento `change` a todos los detectores del menú raíz. El elemento seleccionado actualmente de un grupo de opciones está disponible en ActionScript con la propiedad `selection`, como se indica a continuación:

```
var selectedItem = myMenu.alignment_group.selection;
myMenu.alignment_group = myMenu.center_item;
```

Cada uno de los valores `groupName` debe ser exclusivo en el ámbito de la instancia del menú raíz.

**Nota:** el atributo `selected` sólo debe modificarse mediante el método `setMenuItemSelected(item, b)`. Se puede examinar directamente el atributo `selected`, pero devuelve el valor de cadena `true` o `false`.

## Exposición de elementos de menú en ActionScript

Puede asignar un identificador exclusivo a cada elemento de menú con el atributo `instanceName`; de esta forma se puede acceder directamente al elemento de menú desde el menú raíz. Por ejemplo, en el código XML siguiente se proporcionan atributos `instanceName` a cada elemento de menú:

```
<menu>
  <menuitem label="Item 1" instanceName="item_1" />
  <menuitem label="Item 2" instanceName="item_2" >
    <menuitem label="SubItem A" instanceName="sub_item_A" />
    <menuitem label="SubItem B" instanceName="sub_item_B" />
  </menuitem>
</menu>
```

Puede utilizar `ActionScript` para acceder directamente a las instancias de objeto correspondientes y a sus atributos desde el componente `Menu`, como se indica a continuación:

```
var aMenuItem = myMenu.item_1;
myMenu.setMenuItemEnabled(item_2, true);
var aLabel = myMenu.sub_item_A.label;
```

**Nota:** cada atributo `instanceName` debe ser exclusivo en el ámbito de la instancia del componente del menú raíz (incluidos todos los submenús del menú raíz).

## Propiedades de objeto de inicialización

El parámetro *initObject* (objeto de inicialización) es un concepto fundamental en la creación del diseño del componente `Menu`. El parámetro *initObject* es un objeto con propiedades. Cada propiedad representa uno de los posibles atributos XML de un elemento de menú. Para obtener una descripción de las propiedades permitidas en el parámetro *initObject*, véase [“Atributos XML de elementos de menú” en la página 396](#).)

El parámetro *initObject* se utiliza en los métodos siguientes:

- `Menu.addItem()`
- `Menu.addItemAt()`
- `MenuDataProvider.addItem()`
- `MenuDataProvider.addItemAt()`

En el ejemplo siguiente se crea un parámetro *initObject* con dos propiedades, `label` e `instanceName`:

```
var i = myMenu.addItem({label:"myMenuItem", instanceName:"myFirstItem"});
```

Varias propiedades se combinan para crear un tipo concreto de elemento de menú. Para crear determinados tipos de elementos de menús (normales, de separador, de casilla de verificación o de botón de opciones), es necesario asignar propiedades específicas.

Por ejemplo, puede inicializar un elemento de menú normal con el parámetro *initObject* siguiente:

```
myMenu.addItem({label:"myMenuItem", enabled:true, icon:"myIcon",
    instanceName:"myFirstItem"});
```

Puede inicializar un elemento de menú de separador con el parámetro *initObject* siguiente:

```
myMenu.addItem({type:"separator"});
```

Puede inicializar un elemento de casilla de verificación con el parámetro *initObject* siguiente:

```
myMenu.addItem({type:"check", label:"myMenuCheck", enabled:false,
    selected:true, instanceName:"myFirstCheckItem"})
```

Puede inicializar un elemento de botón de opciones con el parámetro *initObject* siguiente:

```
myMenu.addItem({type:"radio", label:"myMenuRadio1", enabled:true,
    selected:false, groupName:"myRadioGroup" instanceName:"myFirstRadioItem"})
```

Recuerde que debe tratar los atributos `instanceName`, `groupName` y `type` de un elemento de menú como de sólo lectura. Sólo debe definirlos al crear un elemento (por ejemplo, en una llamada a `addItem()`). Si modifica estos atributos después de crearlos, los resultados pueden ser imprevisibles.

## Parámetros del componente Menu

No hay parámetros de edición para el componente Menu.

Utilice ActionScript para controlar el componente Menu con sus propiedades, métodos y eventos. Para más información, consulte [“Clase Menu \(sólo para Flash Professional\)” en la página 403](#).

## Creación de aplicaciones con el componente Menu

En el ejemplo siguiente un desarrollador de aplicaciones está creando una aplicación y utiliza el componente Menu para exponer algunos de los comandos que el usuario puede emitir como, por ejemplo, Open, Close, Save, etc.

**Para crear una aplicación con el componente Menu:**

- 1 Seleccione Archivo > Nuevo y cree un documento de Flash.
- 2 Arrastre el componente Menu del panel Componentes al escenario y elimínelo.  
De este modo, el componente se añade a la biblioteca sin añadirse a la aplicación. Los menús se crean de forma dinámica con ActionScript.
- 3 Arrastre un componente Button desde el panel Componentes al escenario.  
Si se hace clic en el botón se activa el menú.
- 4 En el inspector de propiedades, asigne al botón el nombre de instancia **commandBtn**, y cambie su propiedad text por **Commands**.
- 5 En el panel Acciones, en el primer fotograma, introduzca el código siguiente para añadir un detector de eventos que detecte los eventos click de la instancia commandBtn:

```
var listener = new Object();
listener.click = function(evtObj){
    var button = evtObj.target;
    if(button.menu == undefined) {
        // Crear una instancia de menú y añadir algunos elementos
        button.menu = mx.controls.Menu.createMenu();
        button.menu.addItem("Open");
        button.menu.addItem("Close");
        button.menu.addItem("Save");
        button.menu.addItem("Revert");
        // Añadir un detector de cambios que recoja las selecciones de
        elementos
        var changeListener = new Object();
        changeListener.change = function(event) {
            var item = event.menuItem;
            trace("Item selected: " + item.attributes.label);
        }
        button.menu.addEventListener("change", changeListener);
    }
    button.menu.show(button.x, button.y + button.height);
}
commandBtn.addEventListener("click", listener);
```

- 6 Seleccione Control > Probar película.

Haga clic en el botón Commands para que aparezca el menú. Seleccione elementos de menú para ver las acciones trace que notifican qué elemento se ha seleccionado en la ventana Salida.

### Para utilizar datos XML desde un servidor a fin de crear y rellenar un menú:

- 1 Seleccione Archivo > Nuevo y cree un documento de Flash.
- 2 Arrastre el componente Menu del panel Componentes al escenario y elimínelo.  
De este modo, el componente se añade a la biblioteca sin añadirse a la aplicación. Los menús se crean de forma dinámica con ActionScript.
- 3 En el panel Acciones, añada el código siguiente al primer fotograma para crear un menú y algunos elementos:

```
var myMenu = mx.controls.Menu.createMenu();  
// Importar un archivo XML  
var loader = new XML();  
loader.menu = myMenu;  
loader.ignoreWhite = true;  
loader.onLoad = function(success) {  
    //Cuando llegan los datos, se pasan al menú  
    if (success){  
        this.menu.dataProvider = this.firstChild;  
    }  
};  
loader.load(url);
```

**Nota:** los elementos de menú se describen mediante el nodo secundario del primer nodo secundario del documento XML.

- 4 Seleccione Control > Probar película.

### Para crear una cadena XML correcta para crear y rellenar un menú:

- 1 Seleccione Archivo > Nuevo y cree un documento de Flash.
- 2 Arrastre el componente Menu del panel Componentes al escenario y elimínelo.  
De este modo, el componente se añade a la biblioteca sin añadirse a la aplicación. Los menús se crean de forma dinámica con ActionScript.
- 3 En el panel Acciones, añada el código siguiente al primer fotograma para crear un menú y algunos elementos:

```
// Crear una cadena XML que contiene una definición de menú  
var s = "";  
s += "<menu>";  
s += "<menuitem label='Undo' />";  
s += "<menuitem type='separator' />";  
s += "<menuitem label='Cut' />";  
s += "<menuitem label='Copy' />";  
s += "<menuitem label='Paste' />";  
s += "<menuitem label='Clear' />";  
s += "<menuitem type='separator' />";  
s += "<menuitem label='Select All' />";  
s += "</menu>";  
// Crear un objeto XML a partir de la cadena  
var xml = new XML(s);  
xml.ignoreWhite = true;  
// Crear un menú a partir del primer secundario del objeto XML  
var myMenu = mx.controls.Menu.createMenu(_root, xml.firstChild);
```

- 4 Seleccione Control > Probar película.

**Para utilizar la clase `MenuDataProvider` a fin de crear y rellenar un menú:**

- 1 Seleccione Archivo > Nuevo y cree un documento de Flash.
- 2 Arrastre el componente Menu del panel Componentes al escenario y elimínelo.  
De este modo, el componente se añade a la biblioteca sin añadirse a la aplicación. Los menús se crean de forma dinámica con `ActionScript`.
- 3 En el panel Acciones, añada el código siguiente al primer fotograma para crear un menú y algunos elementos:

```
// Crear un objeto XML para que actúe como un almacén
var xml = new XML();

// El elemento que se cree a continuación no aparecerá en el menú.
// La llamada al método 'createMenu' (a continuación) espera
// recibir un elemento raíz cuyos nodos secundarios se convertirán
// en los elementos. Ésta es una manera sencilla de crear
// este elemento raíz y asignarle un nombre
// sencillo.
var theMenuElement = xml.addItem("Edit");

// Añadir elementos de menú
theMenuElement.addItem({label:"Undo"});
theMenuElement.addItem({type:"separator"});
theMenuElement.addItem({label:"Cut"});
theMenuElement.addItem({label:"Copy"});
theMenuElement.addItem({label:"Paste"});
theMenuElement.addItem({label:"Clear", enabled:"false"});
theMenuElement.addItem({type:"separator"});
theMenuElement.addItem({label:"Select All"});
// Crear el objeto de menú
var theMenuControl = mx.controls.Menu.createMenu(_root, theMenuElement);
```

- 4 Seleccione Control > Probar película.

## Personalización del componente Menu

El menú adapta su tamaño horizontal al del texto más extenso. Asimismo puede llamar al método `setSize()` para adaptar el tamaño del componente. El tamaño máximo permitido de los iconos es de 16 X 16 píxeles.

## Utilización de estilos con el componente Menu

Llame al método `setStyle()` para cambiar el estilo de los menús, sus elementos y sus submenús. Un componente `Menu` admite los siguientes estilos de Halo:

| Estilo                              | Descripción                                                                                    |
|-------------------------------------|------------------------------------------------------------------------------------------------|
| <code>themeColor</code>             | Color del fondo del menú. Es el único estilo de color que no hereda su valor.                  |
| <code>color</code>                  | Color del texto de un elemento de menú.                                                        |
| <code>disabledColor</code>          | Color desactivado para el texto.                                                               |
| <code>fontFamily</code>             | Nombre de la fuente del texto.                                                                 |
| <code>fontSize</code>               | Tamaño de la fuente en puntos.                                                                 |
| <code>fontStyle</code>              | Estilo de la fuente; puede ser <code>normal</code> o <code>italic</code> .                     |
| <code>fontWeight</code>             | Grosor de la fuente; puede ser <code>normal</code> o <code>bold</code> .                       |
| <code>rolloverColor</code>          | Color de sustitución de los elementos de menú.                                                 |
| <code>selectionColor</code>         | Elementos seleccionados y elementos que contienen submenús.                                    |
| <code>selectionDisabledColor</code> | Elementos seleccionados y elementos que contienen submenús y que están desactivados.           |
| <code>textRolloverColor</code>      | Color del texto al pasar el ratón sobre un elemento.                                           |
| <code>textDecoration</code>         | Decoración del texto; puede ser <code>none</code> o <code>underline</code> .                   |
| <code>textDisabledColor</code>      | Color del texto desactivado.                                                                   |
| <code>textSelectedColor</code>      | Color del texto de un elemento de menú seleccionado.                                           |
| <code>popupDuration</code>          | Duración de la transición durante la apertura de un menú. Si no hay transición, el valor es 0. |

## Utilización de aspectos con el componente Menu

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

### Clase Menu (sólo para Flash Professional)

**Herencia** `UIObject` > `UIComponent` > `View` > `ScrollView` > `ScrollSelectList` > `Menu`

**Nombre de clase de ActionScript** `mx.controls.Menu`

## Resumen de métodos de la clase Menu

| Método                                  | Descripción                                                                                             |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------|
| <code>Menu.addItem()</code>             | Añade un elemento de menú al menú.                                                                      |
| <code>Menu.addItemAt()</code>           | Añade un elemento de menú al menú en una ubicación determinada.                                         |
| <code>Menu.createMenu()</code>          | Crea una instancia de la clase Menu. Se trata de un método estático.                                    |
| <code>Menu.getItemAt()</code>           | Obtiene una referencia a un elemento de menú en una ubicación determinada.                              |
| <code>Menu.hide()</code>                | Cierra un menú.                                                                                         |
| <code>Menu.indexOf()</code>             | Devuelve el índice de un elemento de menú determinado.                                                  |
| <code>Menu.removeAll()</code>           | Elimina todos los elementos de un menú.                                                                 |
| <code>Menu.removeItemAt()</code>        | Elimina un elemento de un menú en una ubicación determinada.                                            |
| <code>Menu.setMenuItemEnabled()</code>  | Indica si un elemento de menú está activado ( <code>true</code> ) o desactivado ( <code>false</code> ). |
| <code>Menu.setMenuItemSelected()</code> | Indica si un elemento de menú está seleccionado ( <code>true</code> ) o no ( <code>false</code> ).      |
| <code>Menu.show()</code>                | Abre un menú en una ubicación especificada o en una ubicación anterior.                                 |

Hereda todos los métodos de [UIObject](#), [UIComponent](#), [ScrollView](#) y [ScrollSelectList](#).

## Resumen de propiedades de la clase Menu

| Propiedad                      | Descripción                 |
|--------------------------------|-----------------------------|
| <code>Menu.dataProvider</code> | Fuente de datos de un menú. |

Hereda todas las propiedades de [UIObject](#), [UIComponent](#), [ScrollView](#) y [ScrollSelectList](#).

## Resumen de eventos de la clase Menu

| Evento                     | Descripción                                                   |
|----------------------------|---------------------------------------------------------------|
| <code>Menu.change</code>   | Se difunde cuando un usuario selecciona un elemento.          |
| <code>Menu.menuHide</code> | Se difunde cuando un menú se cierra.                          |
| <code>Menu.menuShow</code> | Se difunde cuando un menú se abre.                            |
| <code>Menu.rollOut</code>  | Se difunde cuando el puntero deja de estar sobre un elemento. |
| <code>Menu.rollOver</code> | Se difunde cuando el puntero pasa sobre un elemento.          |

Hereda todos los eventos de [UIObject](#), [UIComponent](#), [ScrollView](#) y [ScrollSelectList](#).



## Menu.addItem()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
myMenu.addItem(initObject)
```

Sintaxis 2:

```
myMenu.addItem(childMenuItem)
```

### Parámetros

*initObject* Objeto que contiene las propiedades que inicializan los atributos de un elemento de menú. Véase [“Atributos XML de elementos de menú” en la página 396](#).

*childMenuItem* Objeto de nodo XML.

### Valor devuelto

Una referencia al nodo XML añadido.

### Descripción

Método; la sintaxis 1 añade un elemento de menú al final del menú. El elemento de menú se crea a partir de los valores proporcionados en el parámetro *initObject*. La sintaxis 2 añade al final del menú un elemento que es un nodo XML creado previamente (con la forma de objeto XML). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

### Ejemplo

Sintaxis 1: en el ejemplo siguiente se añade un elemento al final de un menú:

```
myMenu.addItem({ label:"Item 1", type:"radio", selected:false,  
    enabled:true, instanceName:"radioItem1", groupName:"myRadioGroup" } );
```

Sintaxis 2: en el ejemplo siguiente se traslada un nodo de un menú a la raíz de otro menú:

```
myMenu.addItem(mySecondMenu.getMenuItemAt(mySecondMenu, 3));
```

## Menu.addItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

### Sintaxis 1:

```
myMenu.addItemAt(index, initObject)
```

### Sintaxis 2:

```
myMenu.addItemAt(index, childMenuItem)
```

## Parámetros

*index* Entero que indica el orden (entre los nodos secundarios) en el que se añade el elemento.

*initObject* Objeto que contiene propiedades que inicializan los atributos de un elemento de menú. Véase [“Atributos XML de elementos de menú” en la página 396](#).

*childMenuItem* Objeto de nodo XML.

## Valor devuelto

Una referencia al nodo XML añadido.

## Descripción

Método; la sintaxis 1 añade un elemento (nodo secundario) en una ubicación determinada de un menú. El elemento de menú se crea a partir de los valores proporcionados en el parámetro *initObject*. La sintaxis 2 añade en una ubicación determinada del menú un elemento que es un nodo XML creado previamente (con la forma de objeto XML). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

## Ejemplo

Sintaxis 1: en el ejemplo siguiente se añade un nodo nuevo como segundo secundario de la raíz del menú.

```
myMenu.addItemAt(1, { label:"Item 1", instanceName:"radioItem1",  
    type:"radio", selected:false, enabled:true, groupName:"myRadioGroup" } );
```

Sintaxis 2: en el ejemplo siguiente se traslada un nodo de un menú a la posición de cuarto secundario de la raíz de otro menú:

```
myMenu.addItemAt(3, mySecondMenu.getMenuItemAt(mySecondMenu, 3));
```

## Menu.change

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    // introduzca aquí el código propio  
}  
myMenu.addEventListener("change", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cada vez que el usuario provoca un cambio en el menú.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando el componente Menu difunde un evento `change`, éste se controla mediante una función (también denominada *controlador*), asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `Menu.change` tiene además las propiedades adicionales siguientes:

- `menuBar` Referencia a la instancia de `MenuBar` principal del componente Menu de destino. Si el componente Menu de destino no pertenece a un menú, este valor es `undefined`.
- `menu` Referencia a la instancia de menú donde está ubicado el elemento de destino.
- `menuItem` Nodo XML, que es el elemento de menú seleccionado.
- `groupName` Cadena que indica el nombre del grupo de botón de opciones al que pertenece el elemento. Si el elemento no pertenece a un grupo de botón de opciones, este valor es `undefined`.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, se define un controlador denominado `listener` y se pasa al método `myMenu.addEventListener()` como segundo parámetro. El controlador `change` captura el objeto de evento en el parámetro `event`. Cuando se difunde el evento `change`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
listener = new Object();
listener.change = function(evt){
    trace("Menu item chosen: "+evt.menuItem.attributes.label);
}
myMenu.addEventListener("change", listener);
```

## Menu.createMenu()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`Menu.createMenu(parent, mdp)`

## Parámetros

*parent* Instancia MovieClip. El clip de película es el componente principal que contiene la nueva instancia de menú. Este parámetro es opcional.

*mdp* Instancia MenuDataProvider que describe esta instancia de menú. Este parámetro es opcional.

## Valor devuelto

Una referencia a la nueva instancia de menú.

## Descripción

Método (estático); crea una instancia de menú y, si así lo desea, la asocia con el principal especificado, con el MenuDataProvider especificado como fuente de datos para los elementos de menú.

Si el argumento *parent* se omite o es nulo, el menú se asocia con la línea de tiempo *\_root*.

Si el argumento *mdp* se omite o es nulo, el menú no tiene elementos. Debe efectuar una llamada a los métodos *addMenu()* o *setDataProvider()* para rellenar el menú.

## Ejemplo

En el ejemplo siguiente, la línea 1 crea un MenuDataProvider que es un objeto XML con los métodos de la clase MenuDataProvider. La línea siguiente añade un elemento de menú (New) con un submenú (File, Project y Resource). El bloque de código siguiente añade más elementos al menú principal. El tercer bloque de código crea un menú vacío asociado con *myParentClip*, lo rellena con la fuente de datos *myMDP* y lo abre en las coordenadas 100, 20, como se indica a continuación:

```
var myMDP = new XML();

var newItem = myMDP.addMenuItem({label:"New"});
newItem.addMenuItem({label:"File..."});
newItem.addMenuItem({label:"Project..."});
newItem.addMenuItem({label:"Resource..."});

myMDP.addMenuItem({label:"Open", instanceName:"miOpen"});
myMDP.addMenuItem({label:"Save", instanceName:"miSave"});
myMDP.addMenuItem({type:"separator"});
myMDP.addMenuItem({label:"Quit", instanceName:"miQuit"});

var myMenu = mx.controls.Menu.createMenu(myParentClip, myMDP);

myMenu.show(100, 20);
```

Para probar este código, colóquelo en el panel Acciones en el fotograma 1 de la línea de tiempo principal. Arrastre un componente Menu del panel Componentes al escenario y elimínelo. De este modo, lo añade a la biblioteca sin colocarlo en el documento.

## Menu.dataProvider

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myMenu.dataProvider`

### Descripción

Propiedad; fuente de datos de los elementos de un componente Menu.

`Menu.dataProvider` es un objeto de nodo XML. Si se define esta propiedad, se sustituye la fuente de datos existente del componente Menu.

El valor predeterminado es `undefined`.

**Nota:** todas las instancias de XML o de XMLNode reciben automáticamente los métodos y las propiedades de la interfaz API MenuDataProvider cuando se utilizan con el componente Menu.

### Ejemplo

En el ejemplo siguiente se importa un archivo XML y se le asigna la propiedad `Menu.dataProvider`:

```
var myMenuDP = new XML();
myMenuDP.load("http://myServer.myDomain.com/source.xml");
myMenuDP.onLoad = function(){
    myMenuControl.dataProvider = myMenuDP;
}
```

## Menu.getMenuItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myMenu.getMenuItemAt(index)`

### Parámetros

*index* Entero que indica el índice del nodo en el menú.

### Valor devuelto

Una referencia al nodo especificado.

### Descripción

Método; devuelve una referencia al nodo secundario especificado del menú.

## Ejemplo

En el ejemplo siguiente se obtiene una referencia al segundo nodo secundario de `myMenu` y se copia el valor en la variable `myItem`:

```
var myItem = myMenu.getMenuItemAt(1);
```

## Menu.hide()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.hide()
```

### Parámetros

*index* Índice del elemento de menú.

### Valor devuelto

Ninguno.

### Descripción

Método; cierra un menú con efectos de transición opcionales.

### Ejemplo

En el ejemplo siguiente se repliega un menú extendido:

```
myMenu.hide();
```

### Véase también

[Menu.show\(\)](#)

## Menu.indexOf()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.indexOf(item)
```

### Parámetros

*item* Referencia a un nodo XML que describe un elemento de menú.

## Valor devuelto

El índice del elemento de menú especificado, o el valor undefined si el elemento no pertenece a este menú.

## Descripción

Método; devuelve el índice del elemento de menú especificado en esta instancia de menú.

## Ejemplo

En este ejemplo se añade un elemento de menú a un elemento principal y se obtiene el índice del elemento dentro de su elemento principal:

```
var myItem = myMenu.addItem({label:"That item"});  
var myIndex = myMenu.indexOf(myItem);
```

## Menu.menuHide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.menuHide = function(eventObject){  
    // introduzca aquí el código propio  
}  
myMenu.addEventListener("menuHide", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cada vez que se cierra un menú.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando el componente Menu distribuye un evento menuHide, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe llamar al método addEventListener() y pasarle el nombre del controlador y el nombre del objeto detector como parámetros.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento Menu.menuHide tiene dos propiedades adicionales:

- **menuBar** Referencia a la instancia de MenuBar principal del componente Menu de destino. Si el componente Menu de destino no pertenece a un MenuBar, este valor es undefined.
- **menu** Referencia a la instancia de Menu que está oculta.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, se define un controlador denominado `form` y se pasa al método `myMenu.addEventListener()` como segundo parámetro. El objeto de evento se captura mediante el controlador `menuHide` en el parámetro `event`. Cuando se difunde el evento `menuHide`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
form = new Object();
form.menuHide = function(evt){
    trace("Menu closed: "+evt.menu);
}
myMenu.addEventListener("menuHide", form);
```

## Véase también

[Menu.menuShow](#)

## Menu.menuShow

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.menuShow = function(eventObject){
    // introduzca aquí el código propio
}
myMenu.addEventListener("menuShow", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cada vez que se abre un menú. Todos los nodos principales abren menús para mostrar los elementos secundarios.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando un componente `Menu` distribuye un evento `menuShow`, éste se controla mediante una función (también denominada *controlador*) asociada con el objeto detector (*listenerObject*), que crea el usuario. Debe llamar al método `addEventListener()` y pasarle el nombre del controlador y el objeto detector como parámetros.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `Menu.menuShow` tiene dos propiedades adicionales:

- `menuBar` Referencia a la instancia de `MenuBar` principal del componente `Menu` de destino. Si el componente `Menu` de destino no pertenece a un `Menu`, este valor es `undefined`.
- `menu` Referencia a la instancia de menú que se muestra.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).



## Ejemplo

En el ejemplo siguiente, se define un controlador denominado `form` y se pasa al método `myMenu.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `menuShow` en el parámetro `evtObject`. Cuando se difunde el evento `menuShow`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
form = new Object();
form.menuShow = function(evt){
    trace("Menu opened: "+evt.menu);
}
myMenu.addEventListener("menuShow", form);
```

## Véase también

[Menu.menuHide](#)

## Menu.removeAll()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.removeAll()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; elimina todos los elementos y actualiza el menú.

## Ejemplo

En el ejemplo siguiente se eliminan todos los nodos del menú:

```
myMenu.removeAll();
```

## Menu.removeItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.removeItemAt(index)
```

## Parámetros

*index* Índice del elemento de menú que se va a eliminar.

## Valor devuelto

Una referencia al elemento de menú que se devuelve (nodo XML). Este valor es undefined si no existe ningún elemento en esta posición.

## Descripción

Método; elimina un elemento de menú y todos sus elementos secundarios en el índice especificado. Si no hay un elemento de menú en dicho índice, la llamada a este método no tiene efecto alguno.

## Ejemplo

En el ejemplo siguiente se elimina un elemento de menú en el índice 3:

```
var item = myMenu.removeItemAt(3);
```

## Menu.rollOut

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.rollOut = function(eventObject){
    // introduzca aquí el código propio
}
myMenu.addEventListener("rollOut", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero deja de estar sobre un elemento de menú.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando un componente Menu difunde un evento rollOut, éste se controla mediante una función (también denominada *controlador*), asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `Menu.rollOut` tiene una propiedad adicional:

- `menuItem` Referencia al elemento de menú (nodo XML) que el puntero ha dejado de señalar.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, se define un controlador denominado `form` y se pasa al método `myMenu.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `rollOut` del parámetro `event`. Cuando se difunde el evento `rollOut`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
form = new Object();
form.rollOut = function(evt){
    trace("Menu rollOut: "+evt.menuItem.attributes.label);
}
myMenu.addEventListener("rollOut", form);
```

## Menu.rollOver

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.rollOver = function(eventObject){
    // introduzca aquí el código propio
}
myMenu.addEventListener("rollOver", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando el puntero pasa sobre un elemento de menú.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. Cuando el componente Menu difunde un evento `change`, éste se controla mediante una función (también denominada *controlador*), asociada con el objeto detector (*listenerObject*) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `Menu.rollOver` tiene una propiedad adicional:

`menuItem` Referencia al elemento de menú (nodo XML) sobre el que pasa el puntero.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, se define un controlador denominado `form` y se pasa al método `myMenu.addEventListener()` como segundo parámetro. El objeto del evento se captura mediante el controlador `rollOver` del parámetro `event`. Cuando se difunde el evento `rollOver`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
form = new Object();
form.rollOver = function(evt){
    trace("Menu rollOver: "+evt.menuItem.attributes.label);
}
myMenu.addEventListener("rollOver", form);
```

## Menu.setMenuItemEnabled()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.setMenuItemEnabled(item, enable)
```

### Parámetros

*item*   Nodo XML. El nodo del elemento del menú destino en el proveedor de datos.

*enable*   Valor booleano que indica si el elemento está activado (`true`) o desactivado (`false`).

### Valor devuelto

Ninguno.

### Descripción

Método; cambia el atributo `enabled` del elemento de destino por el estado que indique el parámetro `enable`. Si esta llamada provoca un cambio de estado, el elemento se vuelve a dibujar con el estado nuevo.

## Ejemplo

En el ejemplo siguiente se desactiva el segundo secundario de `myMenu`:

```
var myItem = myMenu.getMenuItemAt(1);
myMenu.setMenuItemEnabled(myItem, false);
```

### Véase también

[Menu.setMenuItemSelected\(\)](#)

## Menu.setMenuItemSelected()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.setMenuItemSelected(item, select)
```

### Parámetros

*item*   Nodo XML. El nodo del elemento del menú destino en el proveedor de datos.

*select*   Valor booleano que indica si el elemento está seleccionado (*true*) o no lo está (*false*). Si el elemento es una casilla de verificación, este valor indica si la casilla está visible o no. Si el elemento es un botón de opciones, pasa a ser la selección actual del grupo de opciones.

### Valor devuelto

Ninguno.

### Descripción

Método; cambia el atributo *selected* del elemento por el estado especificado por el parámetro *select*. Si esta llamada provoca un cambio de estado, el elemento se vuelve a dibujar con el estado nuevo. Sólo es aplicable a los elementos cuyo atributo *type* sea "radio" o "check", porque hace aparecer o desaparecer su punto o su marca de verificación. Una llamada a este método en un elemento cuyo atributo *type* sea "normal" o "separator" no tendrá ningún efecto.

### Ejemplo

En el ejemplo siguiente se deselecciona el segundo secundario de *myMenu*:

```
var myItem = myMenu.getMenuItemAt(1);  
myMenu.setMenuItemSelected(myItem, false);
```

## Menu.show()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.show(x, y)
```

### Parámetros

*x*   Coordenada *x*.

*y*   Coordenada *y*.

### Valor devuelto

Ninguno.

### Descripción

Método; abre un menú en una ubicación determinada. El menú adopta el tamaño adecuado para que todos los elementos del nivel superior estén visibles, y la esquina superior izquierda se coloca en la ubicación determinada del sistema de coordenadas que ha proporcionado el elemento principal del componente.

Si los parámetros  $x$  e  $y$  se omiten, el menú aparece en la ubicación anterior.

### Ejemplo

En el ejemplo siguiente se extiende un menú:

```
myMenu.show(10, 10);
```

### Véase también

[Menu.hide\(\)](#)

## Clase MenuDataProvider

La clase MenuDataProvider es una aplicación API de “decoración” (adición) que añade funcionalidad a la clase global XMLNode. Esta funcionalidad permite que instancias de XML asignadas a una propiedad dataProvider de Menu manipulen sus propios datos así como las vistas de menú asociadas mediante la interfaz API MenuDataProvider.

Conceptos clave:

- MenuDataProvider es una API de “decoración” (adición). No es necesario crear una instancia de ella para utilizarla.
- Los menús aceptan de modo nativo XML como propiedad dataProvider.
- Si se crea una instancia de una clase Menu, todas las instancias de XML del archivo SWF se “decoran” con la interfaz API MenuDataProvider.
- Sólo los métodos de la interfaz API MenuDataProvider difunden eventos a los controles de menú. Ello no significa que no pueda utilizar métodos XML nativos, pero no son eventos de difusión que actualicen las vistas de menú.
  - Utilice los métodos de la interfaz API MenuDataProvider para controlar el modelo de datos.
  - Utilice los métodos XML para operaciones de sólo lectura como el desplazamiento por la jerarquía del menú.
- Todos los elementos del menú son objetos XML decorados con la interfaz API MenuDataProvider.
- Los cambios que se realicen en los atributos del elemento no se reflejan en el menú en pantalla hasta que se realiza una actualización.

## Resumen de métodos de la clase MenuDataProvider

| Método                                       | Descripción                                                                |
|----------------------------------------------|----------------------------------------------------------------------------|
| <code>MenuDataProvider.addItem()</code>      | Añade un elemento secundario.                                              |
| <code>MenuDataProvider.addItemAt()</code>    | Añade un elemento secundario en una ubicación determinada.                 |
| <code>MenuDataProvider.getItemAt()</code>    | Obtiene una referencia a un elemento de menú en una ubicación determinada. |
| <code>MenuDataProvider.indexOf()</code>      | Devuelve el índice de un elemento de menú determinado.                     |
| <code>MenuDataProvider.removeItem()</code>   | Elimina un elemento de menú.                                               |
| <code>MenuDataProvider.removeItemAt()</code> | Elimina un elemento de menú de una ubicación determinada.                  |

### MenuDataProvider.addItem()

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

Sintaxis 1:

```
myMenu.addItem(initObject)
```

Sintaxis 2:

```
myMenu.addItem(childMenuItem)
```

#### Parámetros

*initObject* Objeto que contiene los atributos específicos que inicializan los atributos de un elemento de menú. Para más información, consulte [“Atributos XML de elementos de menú” en la página 396](#).

*childMenuItem* Nodo XML.

#### Valor devuelto

Una referencia a un objeto XMLNode.

#### Descripción

Método; la sintaxis 1 añade un elemento secundario al final de un elemento de menú principal (que puede ser el propio menú). El elemento de menú se crea a partir de los valores proporcionados en el parámetro *initObject*. La sintaxis 2 añade al final de un elemento de menú principal un elemento secundario que está definido en el parámetro *childMenuItem* XML especificado.

## Ejemplo

En el ejemplo siguiente se añade un nodo nuevo a un nodo especificado en el menú:

```
myMenuDP.firstChild.addMenuItem("Inbox", { label:"Item 1",  
    icon:"radioItemIcon", type:"radio", selected:false, enabled:true,  
    instanceName:"radioItem1", groupName:"myRadioGroup" } );
```

## MenuDataProvider.addItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
myMenu.addItemAt(index, initObject)
```

Sintaxis 2:

```
myMenu.addItemAt(index, childMenuItem)
```

### Parámetros

*index* Entero.

*initObject* Objeto que contiene los atributos específicos que inicializan los atributos de un elemento de menú. Para más información, consulte [“Atributos XML de elementos de menú” en la página 396](#).

*childMenuItem* Nodo XML.

### Valor devuelto

Una referencia al nodo XML añadido.

### Descripción

Método; la sintaxis 1 añade un elemento secundario en la posición de índice especificada al elemento de menú principal (que puede ser el propio menú). El elemento de menú se crea a partir de los valores proporcionados en el parámetro *initObject*. La sintaxis 2 añade un elemento secundario que está definido en el parámetro *childMenuItem* XML especificado al índice especificado de un elemento de menú principal.

## Ejemplo

Sintaxis 1: en el ejemplo siguiente se añade un nodo nuevo como segundo hijo de la raíz del menú.

```
myMenu.addItemAt(1, { label:"Item 1", type:"radio", selected:false,  
    enabled:true, instanceName:"radioItem1", groupName:"myRadioGroup" } );
```



## MenuDataProvider.getMenuItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.getMenuItemAt(index)
```

### Parámetros

*index* Entero que indica la posición del menú.

### Valor devuelto

Una referencia al nodo XML especificado.

### Descripción

Método; devuelve una referencia al elemento de menú secundario especificado del elemento de menú actual.

### Ejemplo

En el ejemplo siguiente se busca el nodo que se quiere obtener y, después, se obtiene el segundo hijo de `myMenuItem`:

```
var myMenuItem = myMenuDP.firstChild.firstChild;  
myMenuItem.getMenuItemAt(1);
```

## MenuDataProvider.indexOf()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.indexOf(item)
```

### Parámetros

*item* Referencia al nodo XML que describe el elemento de menú.

### Valor devuelto

El índice del elemento de menú especificado. Devuelve el valor `undefined` si el elemento no pertenece a este menú.

### Descripción

Método; devuelve el índice del elemento de menú especificado en este elemento de menú principal.

## Ejemplo

En el ejemplo siguiente se añade un elemento de menú a un elemento principal y se obtiene el índice del elemento:

```
var myItem = myParentItem.addMenuItem({label:"That item"});  
var myIndex = myParentItem.indexOf(myItem);
```

## MenuDataProvider.removeItem()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.removeItem()
```

### Parámetros

Ninguno.

### Valor devuelto

Una referencia al elemento de menú eliminado (nodo XML); se devuelve el valor undefined si se produce un error.

### Descripción

Método; elimina el elemento de destino y sus nodos secundarios.

## Ejemplo

En el ejemplo siguiente se elimina myMenuItem de su elemento principal:

```
myMenuItem.removeItem();
```

## MenuDataProvider.removeItemAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenu.removeItemAt(index)
```

### Parámetros

*index*   Índice del elemento de menu.

### Valor devuelto

Una referencia al elemento de menú eliminado. Este valor es undefined si no existe ningún elemento en esta posición.

## Descripción

Método; elimina el elemento secundario del elemento de menú especificado en el parámetro *index*. Si no hay un elemento de menú en dicho índice, la llamada a este método no tiene efecto alguno.

## Ejemplo

En el ejemplo siguiente se elimina el cuarto elemento:

```
myMenuDP.removeMenuItemAt(3);
```

## Componente MenuBar (sólo para Flash Professional)

El componente MenuBar sirve para crear una barra de menús horizontal con menús emergentes y comandos, como las barras de menú Archivo y Edición de la mayoría de las aplicaciones de software (por ejemplo, Macromedia Flash). La barra de menús complementa las funciones del componente Menu ya que proporciona una interfaz en la que se puede hacer clic que muestra y oculta menús que se comportan como un grupo para la interactividad del ratón y del teclado.

El componente MenuBar permite crear con pocos pasos un menú de aplicación. Para crear una barra de menús, puede asignar un proveedor de datos XML a la barra de menús que describa una serie de menús, o bien puede utilizar el método `MenuBar.addMenu()` para agregar instancias de menú de una en una.

Cada uno de los menús de la barra de menús se compone de dos partes: el menú y el botón que hace que el menú se abra (denominado activador de menú). Estos activadores de menú en los que se hace clic aparecen en la barra de menús como una etiqueta de texto cuyo estado de resaltado de borde puede ser saliente o entrante; dichos estados reaccionan ante interacciones del ratón y del teclado.

Cuando se presiona un activador de menú, el menú correspondiente se abre por debajo de éste. El menú permanecerá activo hasta que el activador se presione de nuevo o hasta que se seleccione un elemento de menú o se haga clic fuera del área del menú.

Además de crear activadores de menú que muestran u ocultan menús, el componente MenuBar crea un comportamiento de grupo entre una serie de menús. Ello permite al usuario explorar un gran número de opciones de comando desplazándose por la serie de activadores o utilizando las teclas de flecha para desplazarse por las listas. El ratón y el teclado funciona conjuntamente de forma interactiva para que el usuario pueda desplazarse por los menús del componente MenuBar.

Un usuario no puede desplazarse por los menús de una barra de menús. Si la anchura de los menús es superior a la anchura de la barra de menús, éstos se enmascaran.

Los lectores de pantalla no pueden acceder al componente MenuBar.

## Interacción con el componente MenuBar (sólo para Flash Professional)

Puede utilizar el ratón y el teclado para interactuar con un componente MenuBar.

Si deslaza el ratón por un activador de menú se muestra un resaltado de borde externo alrededor de la etiqueta del activador.

Si se selecciona una instancia de MenuBar haciendo clic en ésta o utilizando el tabulador, podrá utilizar las teclas siguientes para controlarla:

| Tecla            | Descripción                                                  |
|------------------|--------------------------------------------------------------|
| Flecha abajo     | Desplaza la selección a la fila inferior siguiente del menú. |
| Flecha arriba    | Desplaza la selección a la fila superior siguiente del menú. |
| Flecha derecha   | Desplaza la selección al botón siguiente.                    |
| Flecha izquierda | Desplaza la selección al botón anterior.                     |
| Intro/Escape     | Cierra un menú abierto.                                      |

**Nota:** si un menú está abierto, no se puede pulsar la tecla Tabulador para cerrarlo. Debe realizar una selección o cerrar el menú con la tecla Escape.

## Componente MenuBar (sólo para Flash Professional)

Puede utilizar el componente MenuBar para añadir un conjunto de menús (por ejemplo, Archivo, Edición, Especial, Ventana, etcétera) al borde superior de una aplicación.

### Parámetros del componente MenuBar

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente MenuBar en el inspector de propiedades o en el panel Inspector de componentes:

**labels** Matriz que añade activadores de menú a la barra de menús con las etiquetas especificadas. El valor predeterminado es [] (matriz vacía).

Puede escribir código ActionScript para controlar éstas y otras opciones del componente MenuBar utilizando sus propiedades, métodos y eventos. Para más información, consulte [“Clase MenuBar” en la página 427](#).

### Creación de aplicaciones con el componente MenuBar

En este ejemplo se arrastra un componente MenuBar al escenario, se añade código para rellenar la instancia con menús y se asocian detectores a los menús para responder a la selección del elemento de menú.

### Para utilizar un componente **MenuBar** en una aplicación:

- 1 Seleccione Archivo > Nuevo y cree un documento de Flash.
- 2 Arrastre el componente **MenuBar** desde el panel Componentes al escenario.
- 3 Coloque el menú en la parte superior del escenario si desea un diseño estándar.
- 4 Seleccione el componente **MenuBar** y, en el inspector de propiedades, introduzca el nombre de instancia **myMenuBar**.
- 5 En el panel Acciones del Fotograma 1, introduzca el código siguiente:

```
var menu = myMenuBar.addMenu("File");
menu.addItem({label:"New", instanceName:"newInstance"});
menu.addItem({label:"Open", instanceName:"openInstance"});
menu.addItem({label:"Close", instanceName:"closeInstance"});
```

Con este código se añade un menú File a la instancia de barra de menús. Después se utiliza la interfaz API Menu para añadir tres elementos de menú: New, Open y Close.

- 6 En el panel Acciones del Fotograma 1, introduzca el código siguiente:

```
var listen = new Object();
listen.change = function(evt){
    var menu = evt.menu;
    var item = evt.menuItem
    if (item == menu.newInstance){
        myNew();
        trace(item);
    }else if (item == menu.openInstance){
        myOpen()
        trace(item);
    }
}
menu.addEventListener("change",listen);
```

Con este código se crea un objeto detector, `listen`, que utiliza el objeto de evento, `evt`, para captar selecciones de elemento de menú.

**Nota:** debe llamar al método `addEventListener` para registrar el detector con la instancia de menú, no con la instancia de la barra de menús.

- 7 Seleccione Control > Probar película para probar el componente **MenuBar**.

## Personalización del componente **MenuBar** (sólo para Flash Professional)

Este componente adopta el tamaño adecuado en función de las etiquetas de los activadores que se proporcionan en la propiedad `dataProvider` o los métodos de la clase **MenuBar**. Cuando hay un botón de activador en una barra de menús, éste mantiene un tamaño fijo que depende de los estilos de fuente y de la longitud del texto.

## Utilización de estilos con el componente MenuBar

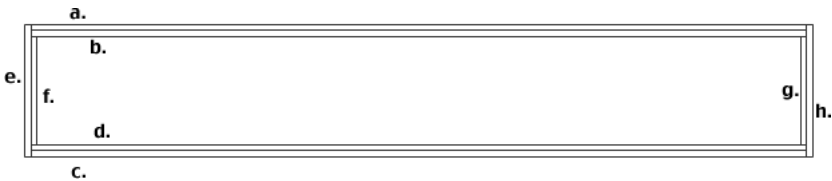
El componente MenuBar crea una etiqueta de activador para cada uno de los menús de un grupo. Utilice los estilos para cambiar el aspecto de las etiquetas de los activadores. El componente MenuBar admite los siguientes estilos de Halo:

| Estilo         | Descripción                                                                                                                                              |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| themeColor     | Color de resaltado de la selección. Es el único estilo de color que no hereda su valor. Los valores posibles son "haloGreen", "haloBlue" y "haloOrange". |
| color          | Texto de la etiqueta de un componente.                                                                                                                   |
| disabledColor  | Color desactivado para el texto.                                                                                                                         |
| fontFamily     | Nombre de la fuente del texto.                                                                                                                           |
| fontSize       | Tamaño de la fuente en puntos.                                                                                                                           |
| fontStyle      | Estilo de la fuente; puede ser "normal" o "italic".                                                                                                      |
| fontWeight     | Grosor de la fuente; puede ser "normal" o "bold".                                                                                                        |
| textDecoration | Decoración del texto; puede ser "none" o "underline".                                                                                                    |
| popupDuration  | Milisegundos que tarda un menú emergente en abrirse. El valor predeterminado es 0.                                                                       |

El componente MenuBar también utiliza la clase RectBorder para dibujar los resaltados interno y externo alrededor de la etiqueta cuando un usuario interactúa con el componente. Puede utilizar el método `setStyle()` (véase `UIObject.setStyle()`) para cambiar las propiedades de estilo RectBorder siguientes:

| Estilos RectBorder | Posición en el borde |
|--------------------|----------------------|
| borderColor        | a                    |
| highlightColor     | b                    |
| borderColor        | c                    |
| shadowColor        | d                    |
| borderCapColor     | e                    |
| shadowCapColor     | f                    |
| shadowCapColor     | g                    |
| borderCapColor     | h                    |

Las propiedades de estilo definen las siguientes posiciones en el borde:



## Utilización de aspectos con el componente MenuBar

El componente MenuBar utiliza los aspectos del componente Menu para representar estados visuales. Para más información sobre los aspectos del componente Menu, consulte [“Utilización de aspectos con el componente Menu” en la página 403](#).

### Clase MenuBar

**Herencia**    UIObject > UIComponent > MenuBar

**Nombre de clase de ActionScript**    mx.controls.MenuBar

### Resumen de métodos de la clase MenuBar

| Método                                  | Descripción                                                                                                                |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <code>MenuBar.addMenu()</code>          | Añade un menú a la barra de menús.                                                                                         |
| <code>MenuBar.addMenuAt()</code>        | Añade un menú a la barra de menús en una ubicación determinada.                                                            |
| <code>MenuBar.getMenuAt()</code>        | Obtiene una referencia a un menú en una ubicación determinada.                                                             |
| <code>MenuBar.getMenuEnabledAt()</code> | Devuelve un valor booleano que indica si el menú está activado ( <code>true</code> ) o desactivado ( <code>false</code> ). |
| <code>MenuBar.removeMenuAt()</code>     | Elimina un menú de una barra de menús en una ubicación determinada.                                                        |
| <code>MenuBar.setMenuEnabledAt()</code> | Valor booleano que indica si el menú esta activado ( <code>true</code> ) o desactivado ( <code>false</code> ).             |

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

### Resumen de propiedades de la clase MenuBar

| Propiedad                          | Descripción                                                                                                                  |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------|
| <code>MenuBar.dataProvider</code>  | Modelo de datos de una barra de menús.                                                                                       |
| <code>MenuBar.labelField</code>    | Cadena que determina qué atributo de cada XMLNode se va a utilizar como texto de etiqueta del elemento de la barra de menús. |
| <code>MenuBar.labelFunction</code> | Función que determina lo que se va a visualizar como etiqueta de cada elemento de la barra de menús.                         |

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

### MenuBar.addMenu()

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX Professional 2004.

## Sintaxis

Sintaxis 1:

```
myMenuBar.addMenu(label)
```

Sintaxis 2:

```
myMenuBar.addMenu(label, menuDataProvider)
```

## Parámetros

*label* Cadena que indica la etiqueta del nuevo menú.

*menuDataProvider* Instancia de XML o de XMLNode que describe el menú y sus elementos. Si el valor es una instancia de XML, se utiliza el elemento `firstChild` de la instancia.

## Valor devuelto

Una referencia al nuevo objeto de menú.

## Descripción

Método; la sintaxis 1 añade un menú único y un activador de menú al final de la barra de menús con el valor especificado en el parámetro *label*. La sintaxis 2 añade un menú único y un activador de menús definidos en el parámetro *menuDataProvider* XML especificado.

## Ejemplo

Sintaxis 1: en el ejemplo siguiente se añade un menú File y, a continuación, se utiliza el método `Menu.addItem()` para añadir los elementos de menú New y Open:

```
menu = myMenuBar.addMenu("File");
menu.addItem({label:"New", instanceName:newInstance});
menu.addItem("{label:'Open', instanceName:'openInstance'}");
```

Sintaxis 2: en el ejemplo siguiente se añade un menú Font con los elementos de menú Bold e Italic que están definidos en el `menuDataProvider` `myMenuDP2`:

```
var myMenuDP2 = new XML();
myMenuDP2.addItem({type:"check", label:"Bold", instanceName:"check1"});
myMenuDP2.addItem({type:"check", label:"Italic", instanceName:"check2"});
menu = myMenuBar.addMenu("Font",myMenuDP2);
```

## MenuBar.addMenuAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

Sintaxis 1:

```
myMenuBar.addMenuAt(index, label)
```

Sintaxis 2:

```
myMenuBar.addMenuAt(index, label, menuDataProvider)
```



## Parámetros

*index* Entero que indica la posición en la que debe insertarse el menú. La primera posición es 0. Para realizar la adición al final del menú, llame al método `MenuBar.addMenu(label)`.

*label* Cadena que indica la etiqueta del nuevo menú.

*menuDataProvider* Instancia de XML o de XMLNode que describe el menú. Si el valor es una instancia de XML, se utiliza el elemento `firstChild` de la instancia.

## Valor devuelto

Una referencia al nuevo objeto de menú.

## Descripción

Método; la sintaxis 1 añade un único menú y un activador de menú en el índice (*index*) especificado con el valor indicado en el parámetro *label*. La sintaxis 2 añade un menú único y un activador de menú con etiqueta en el índice especificado. El contenido del menú se define en el parámetro *menuDataProvider*.

## Ejemplo

Sintaxis 1: en el ejemplo siguiente se coloca un menú a la izquierda de todos los menús de la barra de menús:

```
menu = myMenuBar.addMenuAt(0,"Toreador");
menu.addMenuItem("About Macromedia Flash", instanceName:"aboutInst");
menu.addMenuItem("Preferences", instanceName:"PrefInst");
```

Sintaxis 2: en el ejemplo siguiente se añade un menú Edit con los elementos de menú Undo, Redo, Cut y Copy, que están definidos en *menuDataProvider* myMenuDP:

```
var myMenuDP = new XML();
myMenuDP.addMenuItem({label:"Undo", instanceName:"undoInst"});
myMenuDP.addMenuItem({label:"Redo", instanceName:"redoInst"});
myMenuDP.addMenuItem({type:"separator"});
myMenuDP.addMenuItem({label:"Cut", instanceName:"cutInst"});
myMenuDP.addMenuItem({label:"Copy", instanceName:"copyInst"});

myMenuBar.addMenuAt(0,"Edit",myMenuDP);
```

## MenuBar.dataProvider

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenuBar.dataProvider
```

## Descripción

Propiedad; modelo de datos de los elementos de un componente MenuBar.

`menuBar.dataProvider` es un objeto de nodo XML. Si se define esta propiedad se sustituye el modelo de datos existente del componente MenuBar. Los nodos secundarios que tenga el proveedor de datos se utilizan como elementos de la propia barra de menús; los subnodos de estos nodos secundarios se utilizan como elementos de sus respectivos menús.

El valor predeterminado es `undefined`.

**Nota:** todas las instancias de XML o de XMLNode reciben automáticamente los métodos y las propiedades de la interfaz API MenuDataProvider cuando se utilizan con el componente MenuBar.

## Ejemplo

En el ejemplo siguiente se importa un archivo XML y éste se asigna a la propiedad `MenuBar.dataProvider`:

```
var myMenuBarDP = new XML();
myMenuBarDP.load("http://myServer.myDomain.com/source.xml");
myMenuBarDP.onLoad = function(success){
    if (success){
        myMenuBar.dataProvider = myMenuBarDP;
    } else {
        trace("error loading XML file");
    }
}
```

## MenuBar.getMenuAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenuBar.getMenuAt(index)
```

### Parámetros

*index* Entero que indica la posición del menú.

### Valor devuelto

Una referencia al menú en el índice especificado. Si no hay ningún menú en esta posición, este valor es `undefined`.

### Descripción

Método; devuelve una referencia al menú en el índice especificado.

## Ejemplo

Puesto que el método `getMenuAt()` devuelve una instancia, es posible añadir elementos a un menú en un índice especificado. En el ejemplo siguiente, después de utilizar el parámetro de edición de etiquetas para crear los activadores de menú File, Edit y View, el código siguiente añade los elementos New y Open al menú File en tiempo de ejecución:

```
menu = myMenuBar.getMenuAt(0);
menu.addItem({label:"New",instanceName:"newInst"});
menu.addItem({label:"Open",instanceName:"openInst"});
```

## MenuBar.getMenuEnabledAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenuBar.getMenuEnabledAt(index)
```

### Parámetros

*index*   Índice del elemento MenuBar.

### Valor devuelto

Valor booleano que indica si este menú puede elegirse (`true`) o no (`false`).

### Descripción

Método; devuelve un valor booleano que indica si este menú puede elegirse (`true`) o no (`false`).

## Ejemplo

En el ejemplo siguiente se llama al método en el menú en la primera posición de `myMenuBar`:

```
myMenuBar.getMenuEnabledAt(0);
```

## MenuBar.labelField

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenuBar.labelField
```

### Descripción

Propiedad; cadena que determina el atributo de cada nodo XML que se va a utilizar como texto de etiqueta del menú. Esta propiedad también se pasa a los menús que se crean desde la barra de menús. El valor predeterminado es "label".

Después de definir la propiedad `dataProvider`, esta propiedad es de sólo lectura.

### Ejemplo

En el ejemplo siguiente se utiliza el atributo `name` de cada nodo como texto de etiqueta:

```
myMenuBar.labelField = "name";
```

## MenuBar.labelFunction

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenuBar.labelFunction
```

### Descripción

Propiedad; función que determina lo que se debe mostrar en cada texto de etiqueta de menú. La función acepta el nodo XML asociado con un elemento como parámetro y devuelve la cadena que se va a utilizar como texto de etiqueta. Esta propiedad se pasa a los menús que se crean desde la barra de menús. El valor predeterminado es `undefined`.

Después de definir la propiedad `dataProvider`, esta propiedad es de sólo lectura.

### Ejemplo

En el ejemplo siguiente se crea una etiqueta personalizada a partir de los atributos del nodo:

```
myMenuBar.labelFunction = function(node){  
    var a = node.attributes;  
    return "The Price for " + a.name + " is " + a.price;  
};
```

## MenuBar.removeMenuAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenuBar.removeMenuAt(index)
```

### Parámetros

*index*   Índice del elemento MenuBar.

### Valor devuelto

Una referencia al elemento de MenuBar devuelto. Este valor es undefined si no existe ningún elemento en esta posición.

### Descripción

Método; elimina el menú en el índice especificado. Si no hay un elemento de menú en dicho índice, la llamada a este método no tiene efecto alguno.

### Ejemplo

En el ejemplo siguiente se elimina el menú en el índice 4:

```
myMenuBar.removeMenuAt(4);
```

## MenuBar.setEnabledAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myMenuBar.setEnabledAt(index, boolean)
```

### Parámetros

*index* Índice del elemento MenuBar para definir.

*boolean* Valor booleano que indica si el elemento de menú del índice especificado está activado (true) o no (false).

### Valor devuelto

Ninguno.

### Descripción

Método; activa el menú en el índice especificado. Si no hay un menú en dicho índice, la llamada a este método no tiene efecto alguno.

### Ejemplo

En el ejemplo siguiente se obtiene el objeto MenuBarColumn en el índice 3:

```
myMenuBar.setEnabledAt(3);
```

## Componente NumericStepper

El componente NumericStepper permite recorrer un conjunto ordenado de números. El componente muestra un número junto a pequeños botones de flecha arriba y flecha abajo. Cuando el usuario presiona estos botones, el número aumenta o disminuye de forma gradual. Si el usuario hace clic en uno de los botones de flecha, el número aumenta o disminuye según el valor del parámetro `stepSize`, hasta que el usuario suelta el ratón o hasta que se alcanza el valor máximo o mínimo.

El componente NumericStepper sólo gestiona datos numéricos. Además, para ver más de dos lugares numéricos (por ejemplo, los números 5246 o 1,34), es preciso cambiar el tamaño del componente durante la edición.

Un componente NumericStepper puede estar activado o desactivado en una aplicación. Si está desactivado, no recibe la entrada del ratón ni del teclado. Si está activado y la selección interna está definida en el cuadro de texto, se selecciona al presionar el tabulador hasta su posición o al hacer clic en él. Cuando una instancia de NumericStepper está seleccionada, es posible utilizar las siguientes teclas para controlarla:

| Tecla                  | Descripción                                                               |
|------------------------|---------------------------------------------------------------------------|
| Flecha abajo           | El valor cambia en una unidad.                                            |
| Izquierda              | Desplaza el punto de inserción a la izquierda dentro del cuadro de texto. |
| Derecha                | Desplaza el punto de inserción a la derecha dentro del cuadro de texto.   |
| Mayúsculas + Tabulador | Desplaza la selección al objeto anterior.                                 |
| Tabulador              | Desplaza la selección al objeto siguiente.                                |
| Flecha arriba          | El valor cambia en una unidad.                                            |

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase FocusManager” en la página 292](#).

La vista previa dinámica de cada instancia de NumericStepper refleja el valor indicado en el parámetro `value` durante la edición con el inspector de propiedades o con el panel Inspector de componentes. Sin embargo, en la vista previa dinámica ni el teclado ni el ratón pueden interactuar con los botones de NumericStepper.

Cuando se añade el componente NumericStepper a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.NumericStepperAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#).

## Utilización del componente NumericStepper

Utilice NumericStepper siempre que desee que el usuario seleccione un valor numérico. Por ejemplo, puede utilizar un componente NumericStepper en un formulario para que el usuario indique la fecha de caducidad de su tarjeta de crédito. En otro ejemplo, puede utilizar un componente NumericStepper para que el usuario aumente o disminuya el tamaño de la fuente.

### Parámetros de NumericStepper

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente NumericStepper en el inspector de propiedades o en el panel Inspector de componentes:

**value** define el valor del paso actual. El valor predeterminado es 0.

**minimum** define el valor mínimo del paso. El valor predeterminado es 0.

**maximum** define el valor máximo del paso. El valor predeterminado es 10.

**stepSize** define la unidad de cambio de paso. El valor predeterminado es 1.

Puede escribir código `ActionScript` para controlar éstas y otras opciones adicionales para los componentes NumericStepper utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase NumericStepper](#).

### Creación de aplicaciones con el componente NumericStepper

El siguiente procedimiento explica cómo añadir un componente NumericStepper a una aplicación durante la edición. En este ejemplo, el componente NumericStepper permite que el usuario seleccione una película de entre 0 y 5 estrellas con incrementos de media estrella.

**Para crear una aplicación con el componente Button, siga este procedimiento:**

- 1 Arrastre un componente NumericStepper desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia **starStepper**.
- 3 En el inspector de propiedades, siga este procedimiento:
  - Introduzca 0 en el parámetro **minimum**.
  - Introduzca 5 en el parámetro **maximum**.
  - Introduzca 0,5 en el parámetro **stepSize**.
  - Introduzca 0 en el parámetro **value**.
- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
movieRate = new Object();
movieRate.change = function (eventObject){
    starChart.value = eventObject.target.value;
}
starStepper.addEventListener("change", movieRate);
```

La última línea de código añade un controlador de eventos `change` a la instancia `starStepper`. El controlador indica al clip de película `starChart` que muestre la cantidad de estrellas que especifica la instancia `starStepper` (para ver cómo funciona este código, debe crear un clip de película `starChart` con una propiedad `value` que muestre las estrellas).

## Personalización del componente NumericStepper

El componente NumericStepper puede transformarse horizontal y verticalmente durante la edición o en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)) o cualquier método o propiedad aplicable de la clase NumericStepper. Véase [Clase NumericStepper](#).

Si se cambia el tamaño del componente NumericStepper, no cambia el tamaño de los botones de flecha arriba y flecha abajo. Si al componente NumericStepper se le asigna un tamaño superior a la altura predeterminada, los botones se fijan en la parte superior e inferior del componente. Los botones siempre aparecen a la derecha del cuadro de texto.

## Utilización de estilos con el componente NumericStepper

Es posible definir propiedades de estilo para cambiar el aspecto de una instancia de NumericStepper. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

El componente NumericStepper admite los siguientes estilos de Halo:

| Estilo                      | Descripción                                                                                                                                  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>themeColor</code>     | Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son “haloGreen”, “haloBlue” y “haloOrange”. |
| <code>color</code>          | Texto de la etiqueta de un componente.                                                                                                       |
| <code>disabledColor</code>  | Color desactivado para el texto.                                                                                                             |
| <code>fontFamily</code>     | Nombre de la fuente del texto.                                                                                                               |
| <code>fontSize</code>       | Tamaño de la fuente en puntos.                                                                                                               |
| <code>fontStyle</code>      | Estilo de la fuente; puede ser “normal” o “italic”.                                                                                          |
| <code>fontWeight</code>     | Grosor de la fuente; puede ser “normal” o “bold”.                                                                                            |
| <code>textDecoration</code> | Decoración del texto; puede ser “none” o “underline”.                                                                                        |
| <code>textAlign</code>      | Alineación del texto; puede ser “left”, “right” o “center”.                                                                                  |



## Utilización de aspectos con el componente NumericStepper

El componente NumericStepper emplea aspectos para representar sus estados visuales. Para asignar aspectos a los componentes NumericStepper durante la edición, modifique símbolos de aspecto de la biblioteca y vuelva a exportar el componente como SWC. Los símbolos de aspecto se encuentran en la carpeta Flash UI Components 2/Themes/MMDefault/Stepper Elements/states de la biblioteca. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).

Si un componente NumericStepper está activado, los botones de flecha arriba y flecha abajo muestran el estado Sobre cuando el puntero se desplaza sobre ellos. Los botones muestran el estado Presionado cuando se hace clic en ellos y recuperan el estado Sobre cuando se suelta el ratón. Si el puntero se desplaza fuera de los botones estando el ratón presionado, los botones recuperan su estado original.

Si el componente está desactivado, muestra el estado Desactivado sea cual sea la acción del usuario.

El componente NumericStepper utiliza las siguientes propiedades de aspecto:

| Propiedad         | Descripción                                                                              |
|-------------------|------------------------------------------------------------------------------------------|
| upArrowUp         | Estado Arriba de la flecha arriba. El valor predeterminado es StepUpArrowUp.             |
| upArrowDown       | Estado Presionado de la flecha arriba. El valor predeterminado es StepUpArrowDown.       |
| upArrowOver       | Estado Sobre de la flecha arriba. El valor predeterminado es StepUpArrowOver.            |
| upArrowDisabled   | Estado Desactivado de la flecha arriba. El valor predeterminado es StepUpArrowDisabled.  |
| downArrowUp       | Estado Arriba de la flecha abajo. El valor predeterminado es StepDownArrowUp.            |
| downArrowDown     | Estado Presionado de la flecha abajo. El valor predeterminado es StepDownArrowDown.      |
| downArrowOver     | Estado Sobre de la flecha abajo. El valor predeterminado es StepDownArrowOver.           |
| downArrowDisabled | Estado Desactivado de la flecha abajo. El valor predeterminado es StepDownArrowDisabled. |

## Clase NumericStepper

**Herencia** UIObject > UIComponent > NumericStepper

**Nombre de clase de ActionScript** mx.controls.NumericStepper

Las propiedades de la clase NumericStepper permiten indicar los valores mínimo y máximo de paso, la cantidad de cada paso en unidades y el valor actual del paso en tiempo de ejecución.

Si una propiedad de la clase NumericStepper se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

El componente NumericStepper utiliza FocusManager para sustituir el rectángulo de selección predeterminado de Flash Player por un rectángulo de selección personalizado con esquinas redondeadas. Para más información, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.NumericStepper.version);
```

**Nota:** el código siguiente devuelve undefined: `trace(myNumericStepperInstance.version);`.

### Resumen de métodos de la clase NumericStepper

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

### Resumen de propiedades de la clase NumericStepper

| Propiedad                                    | Descripción                                                                             |
|----------------------------------------------|-----------------------------------------------------------------------------------------|
| <a href="#">NumericStepper.maximum</a>       | Número que indica el valor máximo del rango.                                            |
| <a href="#">NumericStepper.minimum</a>       | Número que indica el valor mínimo del rango.                                            |
| <a href="#">NumericStepper.nextValue</a>     | Número que indica el siguiente valor de la secuencia. Es una propiedad de sólo lectura. |
| <a href="#">NumericStepper.previousValue</a> | Número que indica el valor anterior de la secuencia. Es una propiedad de sólo lectura.  |
| <a href="#">NumericStepper.stepSize</a>      | Número que indica la unidad de cambio de cada paso.                                     |
| <a href="#">NumericStepper.value</a>         | Número que indica el valor actual del componente.                                       |

Hereda todas las propiedades de [UIObject](#) y [UIComponent](#).

### Resumen de eventos de la clase NumericStepper

| Evento                                | Descripción                                |
|---------------------------------------|--------------------------------------------|
| <a href="#">NumericStepper.change</a> | Se activa cuando cambia el valor del paso. |

Hereda todos los eventos de [UIObject](#) y [UIComponent](#).

## NumericStepper.change

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(click){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    ...  
}  
stepperInstance.addEventListener("change", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando se cambia el valor del componente.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `NumericStepper`. La palabra clave `this`, utilizada en un controlador `on()` asociado a un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myStepper`, envía “\_level0.myStepper” al panel Salida:

```
on(click){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*stepperInstance*) distribuye un evento (en este caso, *change*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

Este ejemplo, escrito en un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se cambia un componente denominado `myNumericStepper`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `change` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto `Event` pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto `Event` es el componente que ha generado el evento, en este ejemplo `myNumericStepper`. A la propiedad `NumericStepper.value` se accede desde la propiedad `target` del objeto `Event`. La última línea llama al método `UIEventDispatcher.addEventListener()` desde `myNumericStepper` y pasa como parámetros el evento `change` y el objeto detector `form`:

```
form = new Object();
form.change = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    //es decir, NumericStepper.
    trace("Value changed to " + eventObj.target.value);
}
myNumericStepper.addEventListener("change", form);
```

## NumericStepper.maximum

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*stepperInstance*.maximum

### Descripción

Propiedad; valor máximo del rango del componente. Esta propiedad puede contener un número de tres cifras decimales como máximo. El valor predeterminado es 10.

### Ejemplo

En el ejemplo siguiente se define el valor máximo del rango del componente en 20:

```
myStepper.maximum = 20;
```

### Véase también

[NumericStepper.minimum](#)

## NumericStepper.minimum

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*stepperInstance.minimum*

## Descripción

Propiedad; valor mínimo del rango del componente. Esta propiedad puede contener un número de tres cifras decimales como máximo. El valor predeterminado es 0.

## Ejemplo

En el ejemplo siguiente se define el valor mínimo del rango del componente en 100:

```
myStepper.minimum = 100;
```

## Véase también

[NumericStepper.maximum](#)

## NumericStepper.nextValue

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*stepperInstance.nextValue*

## Descripción

Propiedad (sólo lectura); siguiente valor de la secuencia. Esta propiedad puede contener un número de tres cifras decimales como máximo.

## Ejemplo

En el ejemplo siguiente se define la propiedad `stepSize` en 1 y el valor inicial en 4, por lo que el valor de `nextValue` sería equivalente a 5:

```
myStepper.stepSize = 1;  
myStepper.value = 4;  
trace(myStepper.nextValue);
```

## Véase también

[NumericStepper.previousValue](#)

## NumericStepper.previousValue

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*stepperInstance.previousValue*

## Descripción

Propiedad (sólo lectura); valor anterior de la secuencia. Esta propiedad puede contener un número de tres cifras decimales como máximo.

## Ejemplo

En el ejemplo siguiente se define la propiedad `stepSize` en 1 y el valor inicial en 4, por lo que el valor de `nextValue` sería equivalente a 3:

```
myStepper.stepSize = 1;  
myStepper.value = 4;  
trace(myStepper.previousValue);
```

## Véase también

[NumericStepper.nextValue](#)

## NumericStepper.stepSize

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*stepperInstance.stepSize*

## Descripción

Propiedad; cantidad en unidades que cambia con respecto al valor actual. El valor predeterminado es 1. Este valor no puede ser 0. Esta propiedad puede contener un número de tres cifras decimales como máximo.

## Ejemplo

En el ejemplo siguiente se define el valor actual de `value` en 2 y la unidad `stepSize` en 2. El valor de `nextValue` es 4:

```
myStepper.value = 2;  
myStepper.stepSize = 2;  
trace(myStepper.nextValue);
```

## NumericStepper.value

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*stepperInstance.value*

## Descripción

Propiedad; valor actual que se muestra en el área de texto del componente. El valor no se muestra si no corresponde al rango y al incremento de paso definidos en la propiedad `stepSize`. Esta propiedad puede contener un número de tres cifras decimales como máximo.

## Ejemplo

En el ejemplo siguiente se define el valor actual de `value` en 10 y envía el valor al panel Salida:

```
myStepper.value = 10;  
trace(myStepper.value);
```

## Clase PopUpManager

**Nombre de clase de ActionScript**    `mx.managers.PopUpManager`

La clase `PopUpManager` permite crear ventanas superpuestas que pueden ser modales o amodales (una ventana modal no permite interactuar con otras ventanas mientras está activa). Puede llamar a `PopUpManager.createPopUp()` para crear una ventana superpuesta y llamar a `PopUpManager.deletePopUp()` de la instancia de `Window` para eliminar una ventana emergente.

## Resumen de métodos de la clase PopUpManager

| Evento                                  | Descripción                                                                               |
|-----------------------------------------|-------------------------------------------------------------------------------------------|
| <code>PopUpManager.createPopUp()</code> | Crea una ventana emergente.                                                               |
| <code>PopUpManager.deletePopUp()</code> | Elimina una ventana emergente creada llamando a <code>PopUpManager.createPopUp()</code> . |

## PopUpManager.createPopUp()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
PopUpManager.createPopUp(parent, class, modal [, initobj, outsideEvents])
```

### Parámetros

*parent*    Referencia para que aparezca una ventana.

*class*    Referencia a la clase de objeto que desea crear.

*modal*    Valor booleano que indica si la ventana es modal (`true`) o no (`false`).

*initobj*    Objeto que contiene propiedades de inicialización. Este parámetro es opcional.

*outsideEvents*    Valor booleano que indica si se activa un evento cuando el usuario hace clic fuera de la ventana (`true`) o no (`false`). Este parámetro es opcional.

### Valor devuelto

Referencia a la ventana creada.

### Descripción

Método; si es modal, una llamada a `createPopUp()` busca la ventana principal situada más arriba a partir de la principal y crea una instancia de clase. Si es amodal, una llamada a `createPopUp()` crea una instancia de clase secundaria con respecto a la ventana principal.

### Ejemplo

El código siguiente crea una ventana modal cuando se hace clic en el botón:

```
lo = new Object();
lo.click = function(){
    mx.managers.PopUpManager.createPopUp(_root, mx.containers.Window, true);
}
button.addEventListener("click", lo);
```

## PopUpManager.deletePopUp()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004

### Sintaxis

```
windowInstance.deletePopUp();
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; elimina una ventana emergente y quita el estado modal. La ventana superpuesta es la encargada de llamar a `PopUpManager.deletePopUp()` cuando se está eliminando la ventana.

### Ejemplo

El código siguiente crea una ventana modal denominada `win` con un botón de cierre, y elimina la ventana cuando se hace clic en el botón de cierre:

```
import mx.managers.PopUpManager
import mx.containers.Window
win = PopUpManager.createPopUp(_root, Window, true, {closeButton:true});
lo = new Object();
lo.click = function(){
    win.deletePopUp();
}
win.addEventListener("click", lo);
```



## Componente ProgressBar

El componente ProgressBar muestra el progreso de carga mientras el usuario espera a que se cargue el contenido. El proceso de carga puede ser determinado o indeterminado. Una barra de progreso determinado es la representación lineal del progreso de una tarea en el tiempo y se utiliza cuando la cantidad de contenido que va a cargarse es conocida. Una barra de progreso indeterminado se utiliza cuando la cantidad de contenido que va a cargarse es desconocida. Puede añadirse una etiqueta que muestre el progreso del contenido que se está cargando.

Los componentes están definidos de forma predeterminada en Exportar en primer fotograma. Esto significa que los componentes se cargan en la aplicación antes de que se represente el primer fotograma. Si desea crear un preloader para una aplicación, tendrá que anular la selección de Exportar en primer fotograma en el cuadro de diálogo Propiedades de vinculación de cada componente (menú de opciones del panel Biblioteca > Vinculación). El componente ProgressBar debe definirse en Exportar en primer fotograma, porque debe visualizarse primero, mientras el resto del contenido se carga en Flash Player.

La vista previa dinámica de cada instancia de ProgressBar refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes. En la vista previa dinámica se reflejan los siguientes parámetros: conversion, direction, label, labelPlacement, mode y source.

### Utilización del componente ProgressBar

Una barra de progreso permite ver el progreso de carga de un contenido. Esta información es fundamental para los usuarios mientras interactúan con una aplicación.

Hay varios modos de utilizar el componente ProgressBar; el modo se define con el parámetro mode. Los modos utilizados con más frecuencia son “event” (eventos) y “polled” (sondeo). Estos modos utilizan el parámetro source para especificar un proceso de carga que emita eventos progress y complete (modo de eventos) o que revele los métodos getBytesLoaded y getBytesTotal (modo de sondeo). También puede utilizar el componente ProgressBar en modo manual si define manualmente las propiedades maximum, minimum e indeterminate junto con las llamadas al método `ProgressBar.setProgress()`.

### Parámetros de ProgressBar

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente ProgressBar en el inspector de propiedades o en el panel Inspector de componentes:

**mode** Modo en el que funciona la barra de progreso. Este valor puede ser uno de los siguientes: event, polled o manual. El valor predeterminado es event.

**source** Cadena que va a convertirse en el objeto que represente el nombre de instancia del origen.

**direction** Sentido en el que se llena la barra de progreso. Este valor puede ser right o left; el valor predeterminado es right.

**label** Texto que indica el progreso de la carga. Este parámetro es una cadena con el formato "%1 de %2 cargado (%3%%)"; %1 es un marcador de posición para los bytes actuales cargados, %2 es un marcador de posición para los bytes totales cargados y %3 es un marcador de posición para el porcentaje del contenido cargado. Los caracteres "%" son un marcador de posición del carácter "%". Si el valor de %2 es desconocido, se sustituye por "?". Si el valor es undefined, la etiqueta no se muestra.

**labelPlacement** Posición de la etiqueta con respecto a la barra de progreso. Este parámetro puede tener uno de estos valores: top, bottom, left, right, center. El valor predeterminado es bottom.

**conversion** Número por el que se dividen los valores %1 y %2 de la cadena de la etiqueta antes de que se muestren. El valor predeterminado es 1.

Puede escribir código ActionScript para controlar éstas y otras opciones de componentes ProgressBar con sus propiedades, métodos y eventos. Para más información, consulte [Clase ProgressBar](#).

## Creación de aplicaciones con el componente ProgressBar

El siguiente procedimiento explica cómo añadir un componente ProgressBar a una aplicación durante la edición. En este ejemplo, la barra de progreso se utiliza en modo de eventos. En modo de eventos, el contenido que se carga debe emitir los eventos `progress` y `complete` que la barra de progreso utiliza para mostrar el progreso. El componente Loader emite estos eventos. Para más información, consulte [“Componente Loader” en la página 339](#).

**Para crear una aplicación con el componente ProgressBar en modo de eventos, siga este procedimiento:**

- 1 Arrastre un componente ProgressBar desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, siga este procedimiento:
  - Introduzca el nombre de la instancia **pBar**.
  - Seleccione evento para el parámetro `mode`.
- 3 Arrastre un componente Loader desde el panel Componentes al escenario.
- 4 En el inspector de propiedades, introduzca el nombre de la instancia **loader**.
- 5 En el escenario, seleccione la barra de progreso; en el inspector de propiedades, introduzca **loader** para el parámetro `source`.
- 6 Seleccione el fotograma 1 en la línea de tiempo, abra el panel Acciones e introduzca el código siguiente para cargar un archivo JPEG en el componente Loader:

```
loader.autoLoad = false;
loader.contentPath = "http://imagecache2.allposters.com/images/86/
017_PP0240.jpg";
pBar.source = loader;
// no cargar hasta haber invocado el método de carga
loader.load();
```

En el ejemplo siguiente se utiliza la barra de progreso en modo de sondeo. En este modo, ProgressBar utiliza los métodos `getBytesLoaded` y `getBytesTotal` del objeto de origen para mostrar el progreso de la operación.

**Para crear una aplicación con el componente ProgressBar en modo de sondeo, siga este procedimiento:**

- 1 Arrastre un componente ProgressBar desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, siga este procedimiento:
  - Introduzca el nombre de la instancia **pBar**.
  - Seleccione **polled** para el parámetro **mode**.
  - Introduzca **loader** para el parámetro **source**.
- 3 Seleccione el fotograma 1 en la línea de tiempo; abra el panel Acciones e introduzca el código siguiente para crear un objeto Sound denominado **loader** e invocar al método **loadSound()** para cargar un sonido en el objeto **Sound**:

```
var loader:Object = new Sound();
loader.loadSound("http://soundamerica.com/sounds/sound_fx/A-E/air.wav",
    true);
```

En el ejemplo siguiente se utiliza la barra de progreso en modo manual. Para mostrar el progreso de la operación al utilizar el modo manual, debe definir las propiedades **maximum**, **minimum** e **indeterminate** junto con el método **setProgress()**, pero no necesita definir la propiedad **source**.

**Para crear una aplicación con el componente ProgressBar en modo manual, siga este procedimiento:**

- 1 Arrastre un componente ProgressBar desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, siga este procedimiento:
  - Introduzca el nombre de la instancia **pBar**.
  - Seleccione **manual** para el parámetro **mode**.
- 3 Seleccione el fotograma 1 en la línea de tiempo; abra el panel Acciones e introduzca el código siguiente que actualiza la barra de progreso de forma manual cada vez que se descarga un archivo con una llamada al método **setProgress()**:

```
for(var:Number i=1; i <= total; i++){
    // insertar código para cargar archivo
    // insertar código para cargar archivo
    pBar.setProgress(i, total);
}
```

## Personalización del componente ProgressBar

El componente ProgressBar puede transformarse horizontalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice **UIObject.setSize()**.

Los extremos izquierdo y derecho de la barra de progreso y el gráfico de la guía de deslizamiento tienen un tamaño fijo. Cuando se cambia el tamaño de una barra de progreso, la parte central se ajusta a la distancia entre los extremos. Si la barra de progreso es demasiado pequeña, es posible que no se pueda representar correctamente.

## Utilización de estilos con el componente **ProgressBar**

Se pueden definir propiedades de estilo para cambiar el aspecto de una instancia de la barra de progreso. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

El componente **ProgressBar** admite los siguientes estilos de Halo:

| Estilo                      | Descripción                                                                                                                                  |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>themeColor</code>     | Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son “haloGreen”, “haloBlue” y “haloOrange”. |
| <code>color</code>          | Texto de la etiqueta de un componente.                                                                                                       |
| <code>disabledColor</code>  | Color desactivado para el texto.                                                                                                             |
| <code>fontFamily</code>     | Nombre de la fuente del texto.                                                                                                               |
| <code>fontSize</code>       | Tamaño de la fuente en puntos.                                                                                                               |
| <code>fontStyle</code>      | Estilo de la fuente; puede ser “normal” o “italic”.                                                                                          |
| <code>fontWeight</code>     | Grosor de la fuente; puede ser “normal” o “bold”.                                                                                            |
| <code>textDecoration</code> | Decoración del texto; puede ser “none” o “underline”.                                                                                        |

## Utilización de aspectos con el componente **ProgressBar**

El componente **ProgressBar** indica sus estados mediante los símbolos de clip de película siguientes: `TrackMiddle`, `TrackLeftCap`, `TrackRightCap` y `BarMiddle`, `BarLeftCap`, `BarRightCap` e `IndBar`. El símbolo `IndBar` se utiliza para barras de progreso indeterminado. Para aplicar aspectos al componente **ProgressBar** durante la edición, modifique los símbolos de la biblioteca y vuelva a exportar el componente como archivo SWC. Los símbolos se encuentran en la carpeta `Flash UI Components 2/Themes/MMDefault/ProgressBar Elements` de la biblioteca del archivo `HaloTheme fla` o `SampleTheme fla`. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 41](#).

Si utiliza el método `UIObject.createClassObject()` para crear una instancia del componente **ProgressBar** de forma dinámica (en tiempo de ejecución), podrá aplicar también el aspecto de forma dinámica. Para asignar aspectos a un componente en tiempo de ejecución, defina las propiedades de aspecto en el parámetro `initObject` que pasa al método `createClassObject()`. Las propiedades de aspecto definen los nombres de los símbolos que se van a utilizar como los estados de la barra de progreso.

Un componente `ProgressBar` utiliza las propiedades de aspecto siguientes:

| Propiedad                        | Descripción                                                                                                    |
|----------------------------------|----------------------------------------------------------------------------------------------------------------|
| <code>progTrackMiddleName</code> | Parte central ampliable de la guía de deslizamiento. El valor predeterminado es <code>ProgTrackMiddle</code> . |
| <code>progTrackLeftName</code>   | Extremo izquierdo con tamaño fijo. El valor predeterminado es <code>ProgTrackLeft</code> .                     |
| <code>progTrackRightName</code>  | Extremo derecho con tamaño fijo. El valor predeterminado es <code>ProgTrackRight</code> .                      |
| <code>progBarMiddleName</code>   | Gráfico de barra central ampliable. El valor predeterminado es <code>ProgBarMiddle</code> .                    |
| <code>progBarLeftName</code>     | Extremo izquierdo de la barra con tamaño fijo. El valor predeterminado es <code>ProgBarLeft</code> .           |
| <code>progBarRightName</code>    | Extremo derecho de la barra con tamaño fijo. El valor predeterminado es <code>ProgBarRight</code> .            |
| <code>progIndBarName</code>      | Gráfico de barra indeterminado. El valor predeterminado es <code>ProgIndBar</code> .                           |

## Clase `ProgressBar`

**Herencia** `UIObject` > `ProgressBar`

**Nombre de clase de ActionScript** `mx.controls.ProgressBar`

Si una propiedad de la clase `ProgressBar` se define con `ActionScript`, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.ProgressBar.version);
```

**Nota:** el código siguiente devuelve `undefined`: `trace(myProgressBarInstance.version);`.

## Resumen de métodos de la clase `ProgressBar`

| Método                                 | Descripción                                    |
|----------------------------------------|------------------------------------------------|
| <code>ProgressBar.setProgress()</code> | Define el progreso de la barra en modo manual. |

Hereda todos los métodos de [UIObject](#).

## Resumen de propiedades de la clase **ProgressBar**

| Propiedad                                | Descripción                                                                                                     |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>ProgressBar.conversion</code>      | Número utilizado para convertir el valor de los bytes actuales cargados y el valor total de los bytes cargados. |
| <code>ProgressBar.direction</code>       | Dirección que rellena la barra de progreso.                                                                     |
| <code>ProgressBar.indeterminate</code>   | Indica que se desconoce el valor total de los bytes de origen.                                                  |
| <code>ProgressBar.label</code>           | Texto que acompaña a la barra de progreso.                                                                      |
| <code>ProgressBar.labelPlacement</code>  | Ubicación de la etiqueta con respecto a la barra de progreso.                                                   |
| <code>ProgressBar.maximum</code>         | Valor máximo de la barra de progreso en modo manual.                                                            |
| <code>ProgressBar.minimum</code>         | Valor mínimo de la barra de progreso en modo manual.                                                            |
| <code>ProgressBar.mode</code>            | Modo en el que la barra de progreso carga el contenido.                                                         |
| <code>ProgressBar.percentComplete</code> | Número que indica el porcentaje cargado.                                                                        |
| <code>ProgressBar.source</code>          | Contenido para cargar cuyo progreso está controlado por la barra de progreso.                                   |
| <code>ProgressBar.value</code>           | Indica la cantidad del progreso realizado. Es una propiedad de sólo lectura.                                    |

Hereda todas las propiedades de [UIObject](#).

## Resumen de eventos de la clase **ProgressBar**

| Evento                            | Descripción                                                                  |
|-----------------------------------|------------------------------------------------------------------------------|
| <code>ProgressBar.complete</code> | Se activa cuando se completa la carga.                                       |
| <code>ProgressBar.progress</code> | Se activa a medida que se carga el contenido en modo de eventos o de sondeo. |

Hereda todos los eventos de [UIObject](#).

## **ProgressBar.complete**

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

Sintaxis 1:

```
on(complete){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.complete = function(eventObject){  
    ...  
}  
pBar.addEventListener("complete", listenerObject)
```

## Objeto Event

Además de las propiedades estándar del objeto Event, existen dos propiedades adicionales definidas para el evento `ProgressBar.complete`: `current` (el valor cargado igual al total) y `total` (el valor total).

## Descripción

Evento; se difunde a todos los detectores cuando se ha completado el progreso de carga.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `ProgressBar`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `pBar`, envía “\_level0.pBar” al panel Salida:

```
on(complete){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*pBar*) distribuye un evento (en este caso, `complete`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 602.

## Ejemplo

En este ejemplo se crea un objeto detector `form` con una función `callback complete` que envía un mensaje al panel Salida con el valor de la instancia `pBar`, como en el ejemplo siguiente:

```
form.complete = function(eventObj){  
    // eventObj.target es el componente que ha generado el evento change,  
    // por ejemplo, Progress Bar.  
    trace("Value changed to " + eventObj.target.value);  
}  
pBar.addEventListener("complete", form);
```

## Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## ProgressBar.conversion

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*pBarInstance.conversion*

### Descripción

Propiedad; número que define un valor de conversión para los valores entrantes. Divide los valores actual y total, los promedia y muestra el valor convertido en la propiedad `label`. El valor predeterminado es 1.

### Ejemplo

El código siguiente muestra el valor del progreso de carga, expresado en kilobytes:

```
pBar.conversion = 1024;
```

## ProgressBar.direction

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*pBarInstance.direction*

### Descripción

Propiedad; indica la dirección de relleno de la barra de progreso. El valor predeterminado es "right".



### Ejemplo

El código siguiente hace que la barra de progreso se rellene de derecha a izquierda:

```
pBar.direction = "left";
```

## ProgressBar.indeterminate

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
pBarInstance.indeterminate
```

### Descripción

Propiedad; valor booleano que indica si la barra de progreso tiene un relleno discontinuo y un origen de carga de tamaño desconocido (*true*), o un relleno continuo y un origen de carga con un tamaño conocido (*false*).

### Ejemplo

El código siguiente crea una barra de progreso determinada con un relleno continuo de izquierda a derecha:

```
pBar.direction = "right";  
pBar.indeterminate = false;
```

## ProgressBar.label

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
pBarInstance.label
```

### Descripción

Propiedad; texto que indica el progreso de carga. La propiedad es una cadena con el formato "%1 de %2 cargado (%3%%)"; %1 es un marcador de posición para los bytes actuales cargados, %2 es un marcador de posición para los bytes totales cargados y %3 es un marcador de posición para el porcentaje del contenido cargado. Los caracteres "%" son un marcador de posición del carácter "%". Si el valor de %2 es desconocido, se sustituye por "??". Si el valor es undefined, la etiqueta no se muestra. El valor predeterminado es "LOADING %3%%"

### Ejemplo

El código siguiente define el texto que aparece junto a la barra de progreso con el formato "4 archivos cargados":

```
pBar.label = "%1 archivos cargados";
```

### Véase también

[ProgressBar.labelPlacement](#)

## ProgressBar.labelPlacement

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
pBarInstance.labelPlacement
```

### Descripción

Propiedad; define la ubicación de la etiqueta con respecto a la barra de progreso. Los valores posibles son "left", "right", "top", "bottom" y "center".

### Ejemplo

El código siguiente establece que la etiqueta aparezca por encima de la barra de progreso:

```
pBar.label = "%1 de %2 cargado (%3%)";  
pBar.labelPlacement = "top";
```

### Véase también

[ProgressBar.label](#)

## ProgressBar.maximum

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
pBarInstance.maximum
```

### Descripción

Propiedad; valor más alto para la barra de progreso cuando la propiedad [ProgressBar.mode](#) está definida en "manual".

### Ejemplo

El código siguiente define la propiedad máxima del total de fotogramas de una aplicación de Flash en proceso de carga:

```
pBar.maximum = _totalframes;
```

### Véase también

[ProgressBar.minimum](#), [ProgressBar.mode](#)

## ProgressBar.minimum

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
pBarInstance.minimum
```

### Descripción

Propiedad; valor más bajo de la barra de progreso cuando la propiedad [ProgressBar.mode](#) está definida en "manual".

### Ejemplo

El código siguiente define el valor mínimo de la barra de progreso:

```
pBar.minimum = 0;
```

### Véase también

[ProgressBar.maximum](#), [ProgressBar.mode](#)

## ProgressBar.mode

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
pBarInstance.mode
```

## Descripción

Propiedad; modo en el que la barra de progreso carga el contenido. Este valor puede ser uno de los siguientes: "event", "polled" o "manual". Los modos utilizados con más frecuencia son "event" (eventos) y "polled" (sondeo). Estos modos utilizan el parámetro `source` para especificar un proceso de carga que emite eventos `progress` y `complete`, como un componente `Loader` (modo de eventos), o que revela los métodos `getBytesLoaded` y `getBytesTotal`, como un objeto `MovieClip` (modo de sondeo). También puede utilizar el componente `ProgressBar` en modo manual si define manualmente las propiedades `maximum`, `minimum` e `indeterminate` junto con las llamadas al método `ProgressBar.setProgress()`.

Un objeto `Loader` se debe utilizar como fuente en el modo de eventos. Cualquier objeto que expone los métodos `getBytesLoaded()` y `getBytesTotal()` puede utilizarse como fuente en el modo de sondeo. (Incluido un objeto `Custom` o el objeto `_root`)

## Ejemplo

El código siguiente define la barra de progreso en modo de eventos:

```
pBar.mode = "event";
```

## ProgressBar.percentComplete

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
pBarInstance.percentComplete
```

### Descripción

Propiedad (sólo lectura); devuelve el porcentaje realizado en el proceso. Este valor se promedia. A continuación se indica la fórmula utilizada para calcular el porcentaje:

$$100 * (\text{valor} - \text{mínimo}) / (\text{máximo} - \text{mínimo})$$

## Ejemplo

El código siguiente envía el valor de la propiedad `percentComplete` al panel Salida:

```
trace("percent complete = " + pBar.percentComplete);
```

## ProgressBar.progress

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

Sintaxis 1:

```
on(progress){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.progress = function(eventObject){  
    ...  
}  
pBarInstance.addEventListener("progress", listenerObject)
```

## Objeto Event

Además de las propiedades estándar del objeto Event, existen dos propiedades adicionales definidas para el evento `ProgressBar.progress`: `current` (el valor cargado igual al total) y `total` (el valor total).

## Descripción

Evento; se difunde a todos los detectores registrados siempre que cambia el valor de una barra de progreso. Este evento sólo se difunde cuando la propiedad `ProgressBar.mode` está definida en "manual" o "polled".

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `ProgressBar`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myPBar`, envía “\_level0.myPBar” al panel Salida:

```
on(progress){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`pBarInstance`) distribuye un evento (en este caso, `progress`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (`listenerObject`) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (`eventObject`) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 602](#).

## Ejemplo

En este ejemplo se crea un objeto detector, `form`, y se define en él un controlador de eventos `progress`. El detector `form` se registra en la última línea del código de la instancia `pBar`. Cuando se activa el evento `progress`, `pBar` difunde el evento al detector `form` que llama a la función `callback progress`, como se indica a continuación:

```
var form:Object = new Object();
form.progress = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // por ejemplo, Progress Bar.
    trace("Value changed to " + eventObj.target.value);
}
pBar.addEventListener("progress", form);
```

## Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## ProgressBar.setProgress()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
pBarInstance.setProgress(completed, total)
```

### Parámetros

*completed* número que indica la cantidad del progreso realizado. Se pueden utilizar las propiedades `ProgressBar.label` y `ProgressBar.conversion` para mostrar el número en valor porcentual o cualquier unidad que seleccione, en función de la fuente de la barra de progreso.

*total* número que indica el progreso que debe realizarse hasta alcanzar el 100%.

### Valor devuelto

Número que indica la cantidad del progreso realizado.

### Descripción

Método; define el estado de la barra de progreso para que refleje la cantidad de progreso realizado cuando la propiedad `ProgressBar.mode` está definida en `"manual"`. Puede hacer una llamada a este método para que la barra refleje el estado de un proceso que no sea de carga. El argumento completo se asigna a la propiedad `value`; el argumento total se asigna a la propiedad máxima. La propiedad mínima no cambia.

## Ejemplo

El código siguiente llama al método `setProgress()` en función del progreso realizado por la línea de tiempo de una aplicación de Flash:

```
pBar.setProgress(_currentFrame, _totalFrames);
```

## ProgressBar.source

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*pBarInstance.source*

### Descripción

Propiedad; referencia a la instancia que se va a cargar y de la que mostrará el proceso de carga. El contenido que se carga debe emitir un evento `progress` del que se obtienen los valores total y actual. Este evento sólo se difunde cuando `ProgressBar.mode` está definida en "event" o "polled". El valor predeterminado es `undefined`.

ProgressBar se puede utilizar con los contenidos de una aplicación, incluido `_root`.

### Ejemplo

En este ejemplo se define la instancia `pBar` para que muestre el progreso de carga de un componente Loader con el nombre de instancia `loader`:

```
pBar.source = loader;
```

### Véase también

[ProgressBar.mode](#)

## ProgressBar.value

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*pBarInstance.value*

### Descripción

Propiedad (sólo lectura); indica la cantidad del progreso realizado. La propiedad es un número entre el valor de `ProgressBar.minimum` y `ProgressBar.maximum`. El valor predeterminado es 0.

## Componente RadioButton

El componente `RadioButton` permite hacer que el usuario seleccione una opción de un conjunto de opciones. El componente `RadioButton` debe utilizarse en un grupo formado al menos por dos instancias de `RadioButton`. Sólo es posible seleccionar un miembro del grupo cada vez. Al seleccionar un botón de opción de un grupo, se anula la selección del botón de opción seleccionado en dicho grupo. Puede definir el parámetro `groupName` para indicar el grupo al que pertenece el botón de opción.

Los botones de opción puede estar activados o desactivados. Cuando el usuario desplaza el tabulador hasta un grupo de botones de opción, la selección recae sólo en el botón elegido. El usuario puede presionar las teclas de flecha para cambiar la selección dentro del grupo. Un botón de opción en estado Desactivado no recibe la entrada del ratón ni del teclado.

El grupo de componentes `RadioButton` se selecciona cuando el usuario hace clic sobre él o presiona el tabulador hasta su posición. Cuando la selección esté sobre el grupo `RadioButton` podrá controlarla con las teclas siguientes:

| Tecla           | Descripción                                                                              |
|-----------------|------------------------------------------------------------------------------------------|
| Arriba/Derecha  | La selección se desplaza al botón anterior del grupo de botones de opción.               |
| Abajo/Izquierda | La selección se desplaza al botón siguiente del grupo de botones de opción.              |
| Tabulador       | Desplaza la selección desde el grupo de botones de opción hasta el componente siguiente. |

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase FocusManager” en la página 292](#).

La vista previa dinámica de cada instancia de `RadioButton` del escenario refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes. Sin embargo, la vista previa no muestra la exclusión mutua de la selección. Si define el parámetro `seleccionado` en `true` para dos botones de opción del mismo grupo, ambos botones aparecerán seleccionados, si bien sólo la última instancia creada estará seleccionada en tiempo de ejecución. Para más información, consulte [“Parámetros de RadioButton” en la página 461](#).

Cuando se añade el componente `RadioButton` a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de la pantalla puedan acceder a él. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.RadioButtonAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte [“Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda](#).



## Utilización del componente RadioButton

Los botones de opción son una parte fundamental de cualquier formulario o aplicación Web. Puede utilizar los botones de opción siempre que necesite que un usuario seleccione un elemento de un grupo de opciones. Por ejemplo, podría emplear botones de opción en un formulario para preguntar qué tarjeta de crédito va a utilizar el cliente para pagar.

### Parámetros de RadioButton

Los siguientes son parámetros de edición que se pueden definir para cada instancia del componente RadioButton en el inspector de propiedades o en el panel Inspector de componentes:

**label** define el valor del texto de la etiqueta que aparece en el botón; el valor predeterminado es Radio Button.

**data** es el valor de los datos asociados con el botón de opción. No hay un valor predeterminado.

**groupName** es el nombre del grupo del botón de opción. El valor predeterminado es radioGroup.

**selected** define el valor inicial del botón de opción en seleccionado (true) o no (false). Un botón de opción seleccionado muestra un punto en su interior. Sólo un botón de opción de un grupo puede tener un valor selected definido en true. Si el grupo contiene más de un botón de opción definido en true, sólo se seleccionará el botón cuya instancia se haya hecho en último lugar. El valor predeterminado es false.

**labelPlacement** orienta al texto de la etiqueta que muestra el botón. Este parámetro puede tener uno de estos cuatro valores: left, right, top o bottom; el valor predeterminado es right. Para más información, consulte [RadioButton.labelPlacement](#).

Puede escribir código ActionScript para definir otras opciones para las instancias de RadioButton con los métodos, propiedades y eventos de la clase RadioButton. Para más información, consulte [Clase RadioButton](#).

### Creación de aplicaciones con el componente RadioButton

El procedimiento siguiente explica cómo añadir componentes RadioButton a una aplicación durante la edición. En este ejemplo, los botones de opción se utilizan para presentar una pregunta “¿Es aficionado a Flash?”, que requiere una respuesta afirmativa o negativa. Los datos del grupo de botones de opción aparecen en un componente TextArea con el nombre de instancia theVerdict.

**Para crear una aplicación con el componente RadioButton, siga este procedimiento:**

- 1 Arrastre dos componentes RadioButton desde el panel Componentes al escenario.
- 2 Seleccione uno de los botones de opción, vaya al panel Inspector de componentes y siga este procedimiento:
  - Introduzca Yes para el parámetro de la etiqueta.
  - Introduzca Aficionado a Flash para el parámetro de los datos.
- 3 Seleccione el otro botón de opción, vaya al panel Inspector de componentes y siga este procedimiento:
  - Introduzca No para el parámetro de la etiqueta.
  - Introduzca Anti-Flash para el parámetro de los datos.

- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
flashistListener = new Object();
flashistListener.click = function (evt){
    theVerdict.text = evt.target.selection.data
}
radioGroup.addEventListener("click", flashistListener);
```

La última línea del código añade un controlador de eventos `click` al grupo de botones de opción `radioGroup`. El controlador define la propiedad `text` de la instancia `theVerdict` del componente `TextArea` con el valor de la propiedad `data` del botón de opción seleccionado en el grupo de botones de opción `radioGroup`. Para más información, consulte [RadioButton.click](#).

## Personalización del componente RadioButton

El componente `RadioButton` puede transformarse horizontal o verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [“UIObject.setSize\(\)” en la página 617](#)).

El recuadro de delimitación de un componente `RadioButton` es invisible y designa el área activa del componente. Si aumenta el tamaño del componente, aumenta también el tamaño del área activa.

Si el tamaño de la etiqueta supera el del recuadro de delimitación del componente, la etiqueta se recorta para ajustarse al espacio disponible.

## Utilización de estilos con el componente RadioButton

Puede definir propiedades de estilo para cambiar el aspecto de una instancia de `RadioButton`. Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

Un componente `RadioButton` utiliza los siguientes estilos de Halo:

| Estilo                     | Descripción                                                                                                                                  |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>themeColor</code>    | Fondo de un componente. Es el único estilo de color que no hereda su valor. Los valores posibles son “haloGreen”, “haloBlue” y “haloOrange”. |
| <code>color</code>         | Texto de la etiqueta de un componente.                                                                                                       |
| <code>disabledColor</code> | Color desactivado para el texto.                                                                                                             |
| <code>fontFamily</code>    | Nombre de la fuente del texto.                                                                                                               |
| <code>fontSize</code>      | Tamaño de la fuente en puntos.                                                                                                               |
| <code>fontStyle</code>     | Estilo de la fuente; puede ser “normal” o “italic”.                                                                                          |
| <code>fontWeight</code>    | Grosor de la fuente; puede ser “normal” o “bold”.                                                                                            |

## Utilización de aspectos con el componente RadioButton

Se pueden aplicar aspectos al componente RadioButton durante la edición si se modifican los símbolos del componente en la biblioteca. Los aspectos del componente RadioButton se encuentran en la carpeta siguiente de la biblioteca de HaloTheme fla o SampleTheme fla: Flash UI Components 2/Themes/MMDefault/RadioButton Assets/States. Véase [“Aplicación de aspectos a los componentes” en la página 41](#).

Si hay un botón activado y no seleccionado, al mover el puntero sobre él muestra su estado de roll over. Cuando se hace clic sobre un botón de opción no seleccionado, se selecciona y muestra el estado false, correspondiente a Presionado. Si el usuario suelta el ratón, el botón de opción muestra el estado true y el botón de opción seleccionado en el grupo recupera su estado false. Si se mueve el puntero fuera del botón de opción al tiempo que se presiona el ratón, el botón de opción recupera el estado false y retiene la selección de entrada.

Si se desactiva un botón de opción o un grupo de botones de opción, muestra su estado Desactivado sea cual sea la acción del usuario.

Si utiliza el método `UIObject.createClassObject()` para crear dinámicamente una instancia del componente RadioButton, también podrá asignarle aspectos dinámicamente. Para asignar aspectos a un componente RadioButton de forma dinámica, deberá pasar las propiedades del aspecto al método `UIObject.createClassObject()`. Para más información, consulte [“Aplicación de aspectos a los componentes” en la página 41](#). Las propiedades de aspecto indican qué símbolo se debe utilizar para mostrar un componente.

Un componente RadioButton utiliza las propiedades de aspecto siguientes:

| Nombre            | Descripción                                                                                       |
|-------------------|---------------------------------------------------------------------------------------------------|
| falseUpIcon       | Estado Sin marcar. El valor predeterminado es <code>radioButtonFalseUp</code> .                   |
| falseDownIcon     | Estado Presionado sin marcar. El valor predeterminado es <code>radioButtonFalseDown</code> .      |
| falseOverIcon     | Estado Sobre sin marcar. El valor predeterminado es <code>radioButtonFalseOver</code> .           |
| falseDisabledIcon | Estado Desactivado sin marcar. El valor predeterminado es <code>radioButtonFalseDisabled</code> . |
| trueUpIcon        | Estado Marcado. El valor predeterminado es <code>radioButtonTrueUp</code> .                       |
| trueDisabledIcon  | Estado Desactivado marcado. El valor predeterminado es <code>radioButtonTrueDisabled</code> .     |

## Clase RadioButton

**Herencia** UIObject > UIComponent > SimpleButton > Button > RadioButton

**Nombre de paquete de ActionScript** mx.controls.RadioButton

Las propiedades de la clase RadioButton permiten crear en tiempo de ejecución una etiqueta de texto y definir su posición con respecto al botón de opción. También puede asignar valores de datos a los botones de opción, asignarlos a grupos y seleccionarlos en función de los valores de datos o el nombre de instancia.

Si una propiedad de la clase RadioButton se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

El componente `RadioButton` utiliza `FocusManager` para anular el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información sobre la creación de desplazamientos de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#).

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.RadioButton.version);
```

**Nota:** el código siguiente devuelve `undefined`: `trace(myRadioButtonInstance.version);`.

## Resumen de métodos de la clase `RadioButton`

Hereda todos los métodos de [UIObject](#), [UIComponent](#), [SimpleButton](#) y [Clase Button](#).

## Resumen de propiedades de la clase `RadioButton`

| Propiedad                                  | Descripción                                                                                                                 |
|--------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <a href="#">RadioButton.data</a>           | Valor asociado con una instancia de botón de opción.                                                                        |
| <a href="#">RadioButton.groupName</a>      | Nombre de grupo de un grupo de botones de opción o una instancia de botón de opción.                                        |
| <a href="#">RadioButton.label</a>          | Texto que aparece junto al botón de opción.                                                                                 |
| <a href="#">RadioButton.labelPlacement</a> | Orientación del texto de la etiqueta con respecto al botón de opción.                                                       |
| <a href="#">RadioButton.selected</a>       | Define el estado de la instancia de botón de opción que se va a seleccionar y anula la selección del elegido anteriormente. |
| <a href="#">RadioButton.selectedData</a>   | Selecciona el botón de opción en un grupo de botones de opción con el valor de datos especificado.                          |
| <a href="#">RadioButton.selection</a>      | Referencia al botón de opción seleccionado actualmente en un grupo de botones de opción.                                    |

Hereda todas las propiedades de [UIObject](#), [UIComponent](#), [SimpleButton](#) y [Clase Button](#)

## Resumen de eventos de la clase `RadioButton`

| Evento                            | Descripción                                                         |
|-----------------------------------|---------------------------------------------------------------------|
| <a href="#">RadioButton.click</a> | Se activa cuando se presiona el ratón sobre una instancia de botón. |

Hereda todos los eventos de [UIObject](#), [UIComponent](#), [SimpleButton](#) y [Clase Button](#)

# RadioButton.click

## Disponibilidad

Flash Player 6 versión 79.

## Edición

Flash MX 2004.

## Sintaxis

Sintaxis 1:

```
on(click){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.click = function(eventObject){  
    ...  
}  
radioButtonGroup.addEventListener("click", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se presiona y se suelta) sobre el botón de opción o al seleccionar el botón con las teclas de flecha. El evento también se difunde si se presiona la barra espaciadora o las teclas de flecha cuando la selección esté sobre un grupo de botones de opción pero no hay ninguno de ellos seleccionado.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `RadioButton`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con el botón de opción `myRadioButton`, envía “\_level0.myRadioButton” al panel Salida:

```
on(click){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*radioButtonInstance*) distribuye un evento (en este caso, `click`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. El objeto `Event` tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 602](#).

## Ejemplo

Este ejemplo, escrito sobre un fotograma de la línea de tiempo, envía un mensaje al panel Salida cuando se hace clic en un botón de opción de `radioGroup`. La primera línea de código crea un objeto detector denominado `form`. La segunda línea define una función para el evento `click` en el objeto detector. Dentro de la función hay una acción `trace` que utiliza el objeto `Event` pasado automáticamente a la función, en este ejemplo `eventObj`, para generar un mensaje. La propiedad `target` de un objeto `Event` es el componente que ha generado el evento. Es posible acceder a las propiedades de la instancia desde la propiedad `target` (en este ejemplo, se accede a la propiedad `RadioButton.selection`). La última línea llama al método `UIEventDispatcher.addEventListener()` desde `radioGroup` y lo pasa al evento `click` y al objeto detector `form` como parámetro, como en el ejemplo siguiente:

```
form = new Object();
form.click = function(eventObj){
    trace("The selected radio instance is " + eventObj.target.selection);
}
radioGroup.addEventListener("click", form);
```

El código siguiente envía también un mensaje al panel Salida cuando se hace clic en `radioButtonInstance`. El controlador `on()` debe asociarse directamente con `radioButtonInstance`, como en el ejemplo siguiente:

```
on(click){
    trace("radio button component was clicked");
}
```

## RadioButton.data

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*radioButtonInstance.data*

### Descripción

Propiedad; especifica los datos que se asociarán con una instancia de botón de opción. La definición de esta propiedad sustituye el valor del parámetro de datos definido durante la edición en el inspector de propiedades o en el panel Inspector de componentes. La propiedad `data` puede ser cualquier tipo de datos.

### Ejemplo

En el ejemplo siguiente se asigna el valor de datos `"#FF00FF"` a la instancia de botón de opción `radioOne`:

```
radioOne.data = "#FF00FF";
```

## RadioButton.groupName

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
radioButtonInstance.groupName  
radioButtonGroup.groupName
```

### Descripción

Propiedad; define el nombre de grupo para una instancia o un grupo de botones de opción. Puede utilizar esta propiedad para obtener o definir un nombre de grupo para una instancia de botón de opción o un nombre de grupo para un grupo de botones de opción. Si se llama a este método, se sustituye el valor del parámetro groupName definido durante la edición. El valor predeterminado es "radioGroup".

### Ejemplo

En el ejemplo siguiente se define el nombre de grupo de una instancia de botón de opción en "colorChoice" y, a continuación, se cambia el nombre de grupo por "sizeChoice". Para probar este ejemplo, coloque en el escenario un botón de opción con el nombre de instancia myRadioButton e introduzca el código siguiente en el fotograma 1:

```
myRadioButton.groupName = "colorChoice";  
trace(myRadioButton.groupName);  
colorChoice.groupName = "sizeChoice";  
trace(colorChoice.groupName);
```

## RadioButton.label

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
radioButtonInstance.label
```

### Descripción

Propiedad; especifica la etiqueta de texto para el botón de opción. De forma predeterminada, la etiqueta aparece a la derecha del botón de opción. Si se llama a este método, se sustituye el parámetro de etiqueta especificado durante la edición. Si el texto de la etiqueta es demasiado extenso para caber en el recuadro de delimitación del componente, se recorta el texto.

### Ejemplo

En el ejemplo siguiente se define la propiedad de etiqueta de la instancia radioButton:

```
radioButton.label = "Remove from list";
```

## RadioButton.labelPlacement

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
radioButtonInstance.labelPlacement  
radioButtonGroup.labelPlacement
```

### Descripción

Propiedad; cadena que indica la posición de la etiqueta con respecto a un botón de opción. Puede definir esta propiedad para una sola instancia o para un grupo de botones de opción. Si la propiedad se define para un grupo, la etiqueta se coloca en la posición apropiada a cada botón del grupo.

Éstos son los cuatro valores posibles:

- "right" El botón de opción se fija en la esquina superior izquierda del área de delimitación. La etiqueta aparece a la derecha del botón de opción.
- "left" El botón de opción se fija en la esquina superior derecha del área de delimitación. La etiqueta aparece a la izquierda del botón de opción.
- "bottom" La etiqueta aparece por debajo del botón de opción. El conjunto de botón de opción y etiqueta se centra horizontal y verticalmente. Si el recuadro de delimitación del botón de opción no es lo suficientemente grande, la etiqueta se recortará.
- "top" La etiqueta se coloca por encima del botón de opción. El conjunto de botón de opción y etiqueta se centra horizontal y verticalmente. Si el recuadro de delimitación del botón de opción no es lo suficientemente grande, la etiqueta se recortará.

### Ejemplo

El código siguiente coloca la etiqueta a la izquierda de cada botón de opción de `radioGroup`:

```
radioGroup.labelPlacement = "left";
```

## RadioButton.selected

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
radioButtonInstance.selected  
radioButtonGroup.selected
```



### Descripción

Propiedad; valor booleano que define el estado de un botón de opción en seleccionado (*true*) y anula la selección de cualquier otro botón seleccionado anteriormente, o bien define el botón de opción en no seleccionado (*false*).

### Ejemplo

La primera línea del código define la instancia de `mcButton` en *true*. La segunda línea del código devuelve el valor de la propiedad seleccionada, como en el ejemplo siguiente:

```
mcButton.selected = true;  
trace(mcButton.selected);
```

## RadioButton.selectedData

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
radioButtonGroup.selectedData
```

### Descripción

Propiedad; selecciona el botón de opción con el valor de datos especificado y anula la selección del botón seleccionado anteriormente. Si no se ha especificado la propiedad `data` de una instancia seleccionada, se selecciona y devuelve el valor de etiqueta de la instancia seleccionada. La propiedad `selectedData` puede ser cualquier tipo de datos.

### Ejemplo

En el ejemplo siguiente se selecciona el botón de opción con el valor `"#FF00FF"` del grupo de opciones `colorGroup` y envía el valor al panel Salida:

```
colorGroup.selectedData = "#FF00FF";  
trace(colorGroup.selectedData);
```

## RadioButton.selection

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
radioButtonInstance.selection  
radioButtonGroup.selection
```

## Descripción

Propiedad; se comporta de forma distinta si la propiedad se obtiene o se define. Si la propiedad se obtiene, devuelve la referencia a un objeto del botón seleccionado en un grupo de botones de opción. Si la propiedad se define, selecciona el botón de opción especificado (que se pasa como una referencia a un objeto) de un grupo de botones de opción y anula la selección del botón seleccionado anteriormente.

## Ejemplo

En el ejemplo siguiente se selecciona el botón de opción con el nombre de instancia `color1` y envía su nombre de instancia al panel Salida:

```
colorGroup.selection = color1;  
trace(colorGroup.selection._name)
```

## Componente RDBMSResolver (sólo para Flash Professional)

Los componentes resolver se utilizan junto con el componente DataSet (parte de la función de administración de datos de la arquitectura de datos de Macromedia Flash). Los componentes resolver le permiten convertir los cambios realizados en los datos de la aplicación en un formato que sea apropiado para la fuente de datos externa que está actualizando. Estos componentes no tienen aspectos visuales en tiempo de ejecución.

Si se utiliza el componente DataSet en la aplicación, éste genera un conjunto optimizado de instrucciones (DeltaPacket) que describe los cambios realizados en los datos en tiempo de ejecución. Los componentes resolver convierten este conjunto de instrucciones en el formato apropiado (paquete de actualización). Cuando se envía una actualización al servidor, éste envía una respuesta (paquete de resultados) que contiene otras actualizaciones o errores que se producen como consecuencia de la operación de actualización. Los componentes resolver pueden convertir esta información de nuevo en un DeltaPacket que se puede aplicar al componente DataSet para que esté sincronizado con la fuente de datos externa. Los componentes resolver le permiten mantener su aplicación y una fuente de datos externa sincronizados sin escribir código ActionScript adicional.

El componente RDBMSResolver convierte el código XML que puede recibir y analizar un servicio Web, un JavaBean, un servlet o una página ASP. El código XML contiene la información necesaria y el formato para actualizar cualquier base de datos relacional SQL estándar. Existe un componente resolver paralelo, XUpdateResolver (véase [“Componente XUpdateResolver \(sólo para Flash Professional\)” en la página 677](#)) para devolver datos a un servidor basado en XML. Para más información sobre componentes DataSet, consulte [“Componente DataSet \(sólo para Flash Professional\)” en la página 211](#). Para más información sobre conectores, consulte [“WebServiceConnector \(sólo para Flash Professional\)” en la página 646](#) y [“Componente XMLConnector \(sólo para Flash Professional\)” en la página 668](#). Para más información sobre la arquitectura de datos de Flash, consulte [“Componentes resolver \(sólo para Flash Professional\)” en el apartado Utilización de Flash de la Ayuda](#).

El componente RDBMSResolver convierte los cambios efectuados en los datos de la aplicación en un paquete XML que se puede enviar a una fuente de datos externa.

**Nota:** RDBMSResolver se puede utilizar para enviar actualizaciones de datos a cualquier fuente de datos externa que pueda analizar XML y generar declaraciones SQL a partir de una base de datos; por ejemplo, una página ASP, un servlet Java o un componente ColdFusion.

Las actualizaciones del componente RDBMSResolver se envían en formato de paquete de actualización XML que se transfiere a la base de datos a través de un objeto conector. El componente resolver se conecta a la propiedad DeltaPacket de un componente DataSet, envía su propio paquete de actualización a un conector, vuelve a recibir los errores de servidor del conector y los vuelve a transferir al componente DataSet. Todo eso mediante propiedades que pueden vincularse.

## Utilización del componente RDBMSResolver (sólo para Flash Professional)

Utilice este componente RDBMSResolver sólo cuando la aplicación Flash contenga un componente Dataset y deba enviar de nuevo una actualización a la fuente de datos. Este componente resuelve los datos que se desea devolver a una base de datos relacional.

Para más información sobre cómo trabajar con el componente RDBMSResolver, consulte “Componentes resolver (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

## Parámetros del componente RDBMSResolver

**tableName** Cadena que representa en el código XML el nombre de la tabla de base de datos que se debe actualizar. Debe tener el mismo valor que el valor de entrada del elemento Resolver.fieldInfo que se debe actualizar. Si no hay actualizaciones para este campo, este valor debe dejarse en blanco, que es el valor predeterminado.

**updateMode** Enumerador que determina el modo en el que se identifican los campos clave durante la generación del paquete de actualización XML. El valor predeterminado es umUsingKey. Los posibles valores son los siguientes:

- **umUsingAll** Utiliza los valores anteriores de todos los campos modificados para identificar el registro que debe actualizarse. Es el valor más seguro que debe utilizarse para realizar actualizaciones, porque garantiza que otro usuario no haya modificado el registro desde que se ha recuperado. Sin embargo, este método es muy lento y genera un paquete de actualización más grande.
- **umUsingModified** Utiliza los valores anteriores de todos los campos modificados para identificar el registro que debe actualizarse. Este valor garantiza que otro usuario no haya modificado los mismos campos del registro desde que se ha recuperado.
- **umUsingKey** Valor predeterminado de esta propiedad. Esta configuración utiliza el valor anterior de los campos clave. Esto implica un modelo de “simultaneidad optimista” que utiliza la mayor parte de los sistemas de bases de datos actualmente y garantiza que se modifique el mismo registro que se ha recuperado de la base de datos. Los cambios sobrescriben los cambios que los demás usuarios hayan efectuado en los mismos datos.

**nullValue** Cadena que representa un valor de campo nulo. Se puede personalizar, para evitar confundirlo con una cadena vacía ("" ) o con otro valor válido. El valor predeterminado es { \_NULL\_ }.

**fieldInfo** Colección que representa uno o más campos clave que identifican registros de forma exclusiva. Si la fuente de datos es una tabla de base de datos, debe tener uno o más campos que cifren en clave de forma exclusiva los registros que la componen. Además, puede que se hayan calculado o unido campos de otras tablas. Estos campos deben identificarse para que se puedan establecer los campos clave dentro del paquete de actualización XML y para que los campos que no se hayan actualizado se omitan en el paquete de actualización XML.

El componente `RDBMSResolver` contiene una propiedad `FieldInfo` con este fin. Esta propiedad de colección permite definir un número ilimitado de campos con propiedades que identifican los campos que exigen un manejo especial. Cada elemento `FieldInfo` de la colección contiene tres propiedades:

- `fieldName` Nombre de un campo. Debe asignarse a un campo del componente `DataSet`.
- `ownerName` Valor opcional que se utiliza para identificar campos que no son “propiedad” de la misma tabla definida en el parámetro `TableName` del componente resolver. Si este valor es el mismo valor que el del parámetro `TableName` o está en blanco, el campo se suele incluir en el paquete de actualización XML. Si es otro valor, este campo se excluye del paquete de actualización.
- `isKey` Propiedad booleana que debe establecerse en `true` para que todos los campos clave de la tabla se actualicen.

En el ejemplo siguiente se muestran los elementos `FieldInfo` que se crean para actualizar campos en la tabla `Customer`. Debe identificar los campos clave en la tabla `Customer`. La tabla `Customer` tiene un solo campo clave, `id`; por lo tanto, el usuario debe crear un elemento de campo con los valores siguientes:

```
FieldName = "id"
OwnerName = <--! deje este valor en blanco -->
IsKey = "true"
```

Además, el campo `custType` se añade mediante una unión con la consulta. Este campo debe excluirse de la actualización, por lo que se debe crear un elemento de campo con los valores siguientes:

```
FieldName = "custType"
OwnerName = "JoinedField"
IsKey = "false"
```

Cuando se definen elementos de campo, Flash Player puede utilizarlos para generar de forma automática todo el código XML, que se utiliza para actualizar una tabla.

## Resumen de propiedades para el componente `RDBMSResolver`

| Propiedad                              | Descripción                                                                                                                                                        |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>RDBMSResolver.deltaPacket</code> | Copia de la propiedad <code>DeltaPacket</code> del componente <code>DataSet</code> .                                                                               |
| <code>RDBMSResolver.fieldInfo</code>   | Número ilimitado de campos con propiedades que identifican los campos <code>DataSet</code> que exigen un manejo especial como campo clave o campo no actualizable. |
| <code>RDBMSResolver.nullValue</code>   | Indicador de que el valor de un campo es nulo.                                                                                                                     |
| <code>RDBMSResolver.tableName</code>   | Nombre de tabla que figura en el código XML de la tabla de base de datos que se va a actualizar.                                                                   |
| <code>RDBMSResolver.updateMode</code>  | Valor que determina cómo se identifican los campos cuando se está generando el paquete de actualización XML.                                                       |

| Propiedad                                | Descripción                                                                                                                                                                                                                                                                     |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>RDBMSResolver.updatePacket</code>  | Copia de la propiedad <code>updatePacket</code> de conector que contiene los datos con formato XML más recientes para que el conector los devuelva al componente <code>DataSet</code> después de que el servidor haya recibido la petición de actualización de esta aplicación. |
| <code>RDBMSResolver.updateResults</code> | Copia de la propiedad <code>Results</code> del conector, que devuelve errores con formato XML o actualizaciones para el componente <code>DataSet</code> .                                                                                                                       |

## Resumen de métodos del componente RDBMSResolver

| Método                                    | Descripción                                                                                                                                                                                                                              |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>RDBMSResolver.addFieldInfo()</code> | Añade un nuevo elemento a la colección <code>fieldInfo</code> , que se utiliza para configurar un componente resolver de forma dinámica durante la ejecución, en lugar de utilizar el Inspector de componentes en el entorno de edición. |

## Resumen de eventos del componente RDBMSResolver

| Evento                                        | Descripción                                                                                                                                                                                                                      |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>RDBMSResolver.beforeApplyUpdates</code> | Se define en la aplicación; el componente resolver lo llama para realizar modificaciones personalizadas en el código XML de la propiedad <code>updatePacket</code> antes de que se vincule al conector.                          |
| <code>RDBMSResolver.reconcileResults</code>   | Se define en la aplicación; el componente resolver lo llama para igualar las actualizaciones entre la propiedad <code>updatePacket</code> enviada al servidor y la propiedad <code>updatePacket</code> devuelta por el servidor. |
| <code>RDBMSResolver.reconcileUpdates</code>   | Se define en la aplicación; el componente resolver lo llama para igualar la actualización recibida por el servidor y la actualización pendiente.                                                                                 |

## RDBMSResolver.addFieldInfo()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.addFieldInfo("fieldName", "ownerName", "isKey")
```

## Parámetros

- fieldName* Cadena; proporciona el nombre del campo que este objeto de información describe.
- ownerName* Cadena; proporciona el nombre de la tabla que es propietaria de este campo. Puede dejarse en blanco ("") si es igual que la propiedad de nombre de tabla de la instancia resolver.
- isKey* Valor booleano; indica si este campo es un campo clave.

## Valor devuelto

Ninguno.

## Descripción

Método; añade un nuevo elemento a la colección *fieldInfo* XML en el paquete de actualización. Utilice este método si debe configurar un componente resolver de forma dinámica durante la ejecución en lugar de utilizar el Inspector de componentes en el entorno de edición.

## Ejemplo

En el ejemplo siguiente se crea un componente resolver y se proporciona el nombre de la tabla y el del campo clave, y se impide que el campo *personTypeName* se actualice:

```
var myResolver:RDBMSResolver = new RDBMSResolver();
myResolver.tableName = "Customers";
// Configura el campo id como campo clave
// y el campo personTypeName para que no se actualice.
myResolver.addFieldInfo("id", "", true);
myResolver.addFieldInfo("personTypeName", "JoinedField", false);
// Configura las vinculaciones de datos
//...
```

## RDBMSResolver.beforeApplyUpdates

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.beforeApplyUpdates(eventObject)
```

## Parámetros

*eventObject* Objeto de evento resolver; describe las personalizaciones del paquete XML antes de que se envíe la actualización a través del conector a la base de datos. Este objeto de evento resolver debe contener las propiedades siguientes:

| Propiedad           | Descripción                                           |
|---------------------|-------------------------------------------------------|
| <i>target</i>       | Objeto; resolver que activa este evento.              |
| <i>type</i>         | Cadena; nombre del evento.                            |
| <i>updatePacket</i> | Objeto XML; objeto XML que está a punto de aplicarse. |

## Valor devuelto

Ninguno.

## Descripción

Propiedad; propiedad del tipo `deltaPacket` que recibe un `deltaPacket` que debe convertirse en `xupdatePacket` y produce un `deltaPacket` a partir de los resultados de cualquier servidor incluidos en la propiedad `updateResults`. Este controlador de eventos proporciona un método para realizar modificaciones personalizadas en el código XML antes de enviar los datos actualizados a un conector.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade un `delta` con mensajes al `deltaPacket` de nuevo para que se pueda reenviar la próxima vez que el `deltaPacket` se envíe al servidor. Debe escribir código que maneje deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente `deltaPacket`.

## Ejemplo

En el ejemplo siguiente se añaden los datos de autenticación de usuario al paquete XML:

```
on (beforeApplyUpdates) {  
    // añadir datos de autenticación de usuarios  
    var userInfo = new XML( ""+getUserId()+ ""+getPassword()+" " );  
    updatePacket.firstChild.appendChild( userInfo );  
}
```

## RDBMSResolver.deltaPacket

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

*resolveData*.`deltaPacket`

## Descripción

Propiedad; propiedad del tipo `deltaPacket` que recibe un `deltaPacket` que debe convertirse en un `updatePacket` y emite un `deltaPacket` a partir de los resultados de cualquier servidor incluidos en la propiedad `updateResults`.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade un `delta` con mensajes al `deltaPacket` de nuevo para que se pueda reenviar la próxima vez que el `deltaPacket` se envíe al servidor. Debe escribir código que maneje deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente `deltaPacket`.

## RDBMSResolver.fieldInfo

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

*resolveData.fieldInfo*

### Descripción

Propiedad; la propiedad del tipo `Collection` especifica una colección de un número ilimitado de campos con propiedades que identifican campos `DataSet` que exigen un especial manejo, porque el campo es un campo clave o un campo no actualizable. Cada elemento `FieldInfo` de la colección contiene tres propiedades:

| Propiedad              | Descripción                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>fieldName</code> | Nombre de campo del caso determinado. Este nombre de campo debe asignarse a un nombre de campo de <code>DataSet</code> .                                                                                                                                                                                                                                                                                     |
| <code>ownerName</code> | Esta propiedad adicional es el nombre del propietario de este campo si este campo no es "propiedad" de la misma tabla definida en el parámetro <code>TableName</code> del componente. Si se rellena con el mismo valor que el parámetro o se deja en blanco, el campo se suele incluir en el paquete de actualización XML. Si se rellena con otro valor, este campo se excluye del paquete de actualización. |
| <code>isKey</code>     | Valor booleano establecido en <code>true</code> para todos los campos clave de la tabla que se debe actualizar.                                                                                                                                                                                                                                                                                              |

## RDBMSResolver.nullValue

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

*resolveData.deltaPacket*

### Descripción

Propiedad; propiedad del tipo cadena que se utiliza para proporcionar un valor nulo para el valor de un campo. Se puede personalizar, para evitar confundirlo con una cadena vacía (" ") o con otro valor válido. La cadena predeterminada es `{_NULL_}`.



## RDBMSResolver.reconcileResults

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.reconcileResults(eventObject)
```

### Parámetros

*eventObject* Objeto de evento resolver; describe el objeto de evento utilizado para comparar dos *updatePacket* Este objeto de evento resolver debe contener las propiedades siguientes:

| Propiedad | Descripción                              |
|-----------|------------------------------------------|
| target    | Objeto; resolver que activa este evento. |
| type      | Cadena; nombre del evento.               |

### Valor devuelto

Ninguno.

### Descripción

Evento; lo llama el componente resolver para comparar dos paquetes después de recibir los resultados del servidor y de que éstos se hayan aplicado al *deltaPacket*.

Un solo paquete *updateResults* puede contener los dos resultados de las operaciones que había en el *deltaPacket* e información sobre actualizaciones realizadas por otros clientes. Cuando se recibe un nuevo *updatePacket*, los resultados de la operación y las actualizaciones de las bases de datos se dividen en dos *updatePacket* y se colocan por separado en la propiedad *deltaPacket*. El evento *reconcileResults* se activa justo antes de que el *deltaPacket* que contiene los resultados de la operación se envíe mediante la vinculación de datos.

### Ejemplo

En el ejemplo siguiente se igualan dos *updatePacket* y se devuelven y borran las actualizaciones si son correctas:

```
on (reconcileResults) {  
    // examinar resultados  
    if( examine( updateResults ))  
        myDataSet.purgeUpdates();  
    else  
        displayErrors( results );  
}
```

## RDBMSResolver.reconcileUpdates

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.reconcileUpdates(eventObject)
```

### Parámetros

*eventObject* Objeto de evento resolver; describe las personalizaciones del paquete XML antes de que la actualización se envíe a través del conector a la base de datos. Este objeto de evento resolver debe contener las propiedades siguientes:

| Propiedad | Descripción                              |
|-----------|------------------------------------------|
| target    | Objeto; resolver que activa este evento. |
| type      | Cadena; nombre del evento.               |

### Valor devuelto

Ninguno.

### Descripción

Evento; lo llama el componente resolver cuando se han recibido resultados del servidor después de aplicar las actualizaciones de un `deltaPacket`. Un solo paquete `updateResults` puede contener los dos resultados de las operaciones que había en el `deltaPacket` e información sobre las actualizaciones que han realizado otros clientes. Cuando se recibe un nuevo `updatePacket`, los resultados de las operaciones y las actualizaciones de la base de datos se dividen en dos `deltaPacket`, que se colocan por separado en la propiedad `deltaPacket`. El evento `reconcileUpdates` se activa antes de que el `deltaPacket` que contiene las actualizaciones de base de datos se envíe mediante la vinculación de datos.

### Ejemplo

En el ejemplo siguiente se igualan dos resultados y se borran las actualizaciones si han sido correctas:

```
on (reconcileUpdates) {  
    // examinar resultados  
    if( examine( updateResults ))  
        myDataSet.purgeUpdates();  
    else  
        displayErrors( results );  
}
```

## RDBMSResolver.tableName

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

*resolveData.deltaPacket*

### Descripción

Propiedad; propiedad del tipo cadena que se utiliza para representar el nombre de tabla del código XML de la tabla de base de datos que debe actualizarse. También determina los campos que se tienen que enviar en el `updatePacket`. Para determinar esto, el componente `resolver` compara el valor de esta propiedad con el valor proporcionado para la propiedad `fieldInfo.ownerName`. Si un campo no tiene ninguna entrada en la propiedad de colección `fieldInfo`, el campo se incluye en el `updatePacket`. Si un campo tiene una entrada en la propiedad de colección `fieldInfo` y el valor de la propiedad `ownerName` está en blanco o es idéntico a la propiedad `tableName` del componente, el campo se incluye en el `updatePacket`. Si un campo tiene una entrada en la propiedad de colección `fieldInfo` y el valor de la propiedad `ownerName` no está en blanco y es distinto al de la propiedad `tableName` del componente `resolver`, el campo no se incluye en el `updatePacket`.

## RDBMSResolver.updateMode

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

*resolveData.deltaPacket*

## Descripción

Propiedad; propiedad que contiene varios valores que determinan cómo se identifican los campos clave cuando el paquete de actualización XML se está generando. Las tres cadenas siguientes son valores de esta propiedad:

| Valor                        | Descripción                                                                                                                                                                                                                                                                                                                                                                                         |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>umUsingAll</code>      | Utilice los valores anteriores de todos los campos modificados para identificar el registro que se debe actualizar. Es el valor más seguro que se puede utilizar para realizar actualizaciones, porque garantiza que otro usuario no ha modificado el conjunto del registro desde que lo ha recuperado. Sin embargo, este método es más lento y genera un paquete de actualización de mayor tamaño. |
| <code>umUsingModified</code> | Utilice los valores anteriores de todos los campos modificados para identificar el registro que se debe actualizar. Este valor garantiza que otro usuario no haya modificado los mismos campos del registro desde que se ha recuperado.                                                                                                                                                             |
| <code>umUsingKey</code>      | Valor predeterminado de esta propiedad. Utiliza el valor anterior de los campos clave. Esto implica un modelo de "simultaneidad optimista" que la mayor parte de los sistemas de bases de datos utilizan actualmente y que garantiza que se modifique el mismo registro que se ha recuperado de la base de datos. Los cambios sobrescribirán los cambios en los mismos datos de otro usuario.       |

## RDBMSResolver.updatePacket

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

`resolveData.deltaPacket`

### Descripción

Descripción

Propiedad; propiedad del tipo XML que se utiliza para vincular una propiedad de conector que transmite de nuevo el paquete de actualización de cambios convertido al servidor para que el origen de los datos se pueda actualizar. Se trata de un documento XML que contiene el paquete de cambios de DataSet.

## RDBMSResolver.updateResults

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.deltaPacket
```

### Descripción

Propiedad; propiedad del tipo `deltaPacket` que contiene los resultados de una actualización devuelta por el servidor a través de un conector. Utilice esta propiedad para transmitir errores y datos actualizados del servidor a un `DataSet`; por ejemplo, cuando el servidor asigna nuevos ID para un campo asignado de forma automática. Vincule esta propiedad con la propiedad `Results` del conector de manera que pueda recibir los resultados de una actualización y transmitir los resultados de nuevo al `DataSet`.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade un delta con mensajes al `deltaPacket` de nuevo para que se pueda reenviar la próxima vez que el `deltaPacket` se envíe al servidor. Debe escribir código que maneje deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente `deltaPacket`.

## Interfaz API del componente Remote Procedure Call (RPC)

**Nombre de clase de ActionScript** `mx.data.components.RPC`

La interfaz API del componente Remote Procedure Call (RPC) es una interfaz (un conjunto de métodos, propiedades y eventos) que se puede implementar mediante un componente de MX 2004 v2. La interfaz API del componente RPC define un modo sencillo de envío de parámetros y recepción de resultados desde y a un servicio Web.

Los componentes que implementan la interfaz API de RPC son los componentes `WebServiceConnector` y `XMLConnector`. Estos componentes actúan como conectores entre una fuente de datos externa, como un servicio Web o un archivo XML, y un componente de interfaz de usuario de la aplicación.

Un componente RPC tiene la capacidad de realizar una llamada a una sola función externa, pasar parámetros y recibir resultados. El componente puede realizar una llamada a la misma función varias veces. Para hacer una llamada a varias funciones debe utilizar varios componentes.

## Resumen de propiedades de la clase de componente RPC

| Propiedad                                    | Descripción                                                                                                        |
|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>RPC.multipleSimultaneousAllowed</code> | Indica si se pueden realizar varias llamadas a la vez.                                                             |
| <code>RPC.params</code>                      | Especifica los datos que se enviarán al servidor cuando se ejecute la siguiente operación <code>trigger()</code> . |
| <code>RPC.results</code>                     | Identifica los datos que se han recibido del servidor como resultado de una operación <code>trigger()</code> .     |
| <code>RPC.suppressInvalidCalls</code>        | Indica si debe suprimirse una llamada si hay parámetros no válidos.                                                |

## Resumen de métodos de la clase del componente RPC

| Método                     | Descripción                                |
|----------------------------|--------------------------------------------|
| <code>RPC.trigger()</code> | Inicia una llamada a procedimiento remoto. |

## Resumen de eventos de la clase del componente RPC

| Evento                  | Descripción                                                                                                                                                                          |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>RPC.result</code> | Se difunde cuando una operación Remote Procedure Call se realiza correctamente.                                                                                                      |
| <code>RPC.send</code>   | Se difunde cuando hay una función <code>trigger()</code> en marcha, después de que los datos se hayan recopilado pero antes de que queden invalidados y se inicie la llamada remota. |
| <code>RPC.status</code> | Se difunde cuando se inicia una llamada a procedimiento remoto para informar al usuario del estado de la operación.                                                                  |

## RPC.multipleSimultaneousAllowed

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.multipleSimultaneousAllowed;
```

### Descripción

Propiedad; indica si se pueden realizar varias llamadas al mismo tiempo. Si su valor es `false`, la función `trigger()` realizará una llamada si ya se está realizando otra llamada. Se emitirá un evento `status`, con el código `CallAlreadyInProgress`. Si su valor es `true`, la llamada se realizará.

Cuando se están realizando varias llamadas de forma simultánea, no hay ninguna garantía de que finalicen en el mismo orden en el que se han activado. Además, Flash Player puede limitar el número de operaciones de red simultáneas. Este límite varía para cada versión y plataforma.

## RPC.params

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.params;
```

### Descripción

Propiedad; especifica datos que se enviarán al servidor cuando se ejecute la siguiente operación `trigger()`. Cada componente RPC define cómo se utilizan estos datos y qué tipos son válidos.

## RPC.result

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.addEventListener("result", myListenerObject);
```

### Descripción

Evento; se difunde cuando una operación Remote Procedure Call se realiza correctamente.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena "result"
- `target`: referencia al objeto que ha emitido el evento, por ejemplo un componente `WebServiceConnector`

Puede recuperar el valor de resultado real mediante la propiedad `results`.

## RPC.results

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.results;
```

## Descripción

Propiedad; identifica los datos que se han recibido del servidor como resultado de una operación `trigger()`. Cada componente RPC define cómo se recogen estos datos y qué tipos son válidos. Estos datos aparecen cuando la operación RPC ha finalizado correctamente, tal como señala el evento `result`. Está disponible hasta que el componente se descarga o hasta la siguiente operación RPC.

Es posible que los datos que se devuelvan sean de gran tamaño. Puede gestionarlos de dos maneras:

- Seleccione un clip de película, una línea de tiempo o una pantalla adecuada como elemento principal para el componente RPC. El almacenamiento del componente estará disponible para la eliminación de datos innecesarios cuando el elemento principal no esté presente.
- En ActionScript, puede asignar `null` a esta propiedad en cualquier momento.

## RPC.send

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.addEventListener("send", myListenerObject);
```

## Descripción

Evento; se difunde durante el proceso de una operación `trigger()`, después de que se hayan recogido los datos de parámetros pero antes de que se validen y se inicie la operación Remote Procedure Call. Es una buena ubicación para incluir código que modifique los datos de parámetros antes de la llamada.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena "send"
- `target`: referencia al objeto que ha emitido el evento, por ejemplo un componente `WebServiceConnector`

Puede recuperar o modificar los valores de parámetros reales mediante la propiedad `params`.

## RPC.status

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.addEventListener("status", myListenerObject);
```



## Descripción

Evento; se difunde cuando se inicia una operación Remote Procedure Call para informar al usuario del estado de la operación.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena "status"
- `target`: referencia al objeto que ha emitido el evento; por ejemplo, un componente `WebServiceConnector`
- `code`: cadena que da el nombre de la condición específica que se ha producido.
- `data`: objeto cuyo contenido depende del código.

A continuación se detallan los códigos y los datos asociados disponibles para el evento status:

| Código                | Datos                 | Descripción                                                                                                                                                                                                                                                                                                                                        |
|-----------------------|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StatusChange          | {callsInProgress:nnn} | Este evento se emite siempre que se inicia o finaliza una llamada de servicio Web. El elemento "nnn" es el número de llamadas que se están realizando en este momento.                                                                                                                                                                             |
| CallAlreadyInProgress | sin datos             | Este evento se emite si (a) se realiza una llamada a la función <code>trigger()</code> , (b) el valor de <code>multipleSimultaneousAllowed</code> es <code>false</code> y (c) ya se está realizando una llamada. Después de que se produzca este evento, el intento de llamada se considera realizado y no habrá ningún evento "result" ni "send". |
| InvalidParams         | sin datos             | Este evento se emite si la función <code>trigger()</code> ha detectado que la propiedad "params" no contenía datos válidos. Si el valor de la propiedad "suppressInvalidCalls" es <code>true</code> , el intento de llamada se considera realizado y no habrá ningún evento "result" ni "send".                                                    |

## RPC.suppressInvalidCalls

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.suppressInvalidCalls;
```

### Descripción

Propiedad; indica si debe suprimirse una llamada en caso de que los parámetros no sean válidos. Si su valor es `true`, la función `trigger()` no realizará una llamada en caso de que los parámetros vinculados no superen la validación. Se emitirá un evento "status", con el código `InvalidParams`. Si su valor es `false`, la llamada se realizará con los datos no válidos, como sea necesario.

## RPC.trigger()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.trigger();
```

### Descripción

Método; inicia una Remote Procedure Call. Cada componente RPC define exactamente lo que esto implica. Si la operación es correcta, el resultado de la operación aparecerá en la propiedad `results` del componente RPC.

El método `trigger()` sigue estos pasos:

- 1 Si hay datos vinculados a la propiedad `params`, el método ejecuta todas las vinculaciones para garantizar que haya datos actualizados disponibles. Esto también provoca una validación de datos.
- 2 Si los datos no son válidos y el valor de `suppressInvalidCalls` es `true`, la operación se interrumpe.
- 3 Si la operación continúa, se emite el evento `send`.
- 4 La llamada remota real se inicia a través del método de conexión iniciado (por ejemplo, HTTP).

## Clase Screen (sólo para Flash Professional)

**Herencia** UIObject > UIComponent > View > Loader > Screen

**Nombre de clase de ActionScript** mx.screens.Screen

La clase `Screen` es la clase base de las pantallas que se crean en el panel Contorno de pantalla de Flash MX Professional 2004. Las pantallas son contenedores de alto nivel para crear aplicaciones y presentaciones. Para ver información general sobre cómo trabajar con pantallas, consulte “Trabajo con pantallas (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

La clase `Screen` tiene dos subclases principales: `Slide` y `Form`.

La clase `Slide` proporciona el funcionamiento de ejecución de las presentaciones de diapositivas. La clase `Slide` contiene capacidades de navegación y secuenciación integradas, así como la capacidad de asociar fácilmente transiciones entre las diapositivas mediante comportamientos. Los objetos de diapositiva mantienen una noción de “estado” y permiten al usuario avanzar hasta la diapositiva o el estado siguiente o anterior: cuando se muestra la diapositiva siguiente, la anterior se oculta. Para más información sobre la utilización de la clase `Slide` para controlar las presentaciones de diapositivas, consulte [“Clase Slide \(sólo para Flash Professional\)” en la página 514](#).

La clase `Form` proporciona el entorno de ejecución para las aplicaciones de formularios. Los formularios pueden solaparse y ser contenedores de otros componentes o estar contenidos en éstos. A diferencia de las diapositivas, los formularios no proporcionan ninguna capacidad de secuenciación ni de navegación. Para más información sobre la utilización de la clase `Form`, consulte [“Clase Form \(sólo para Flash Professional\)” en la página 299](#).

La clase Screen proporciona funciones comunes tanto para diapositivas como para formularios.

**Las pantallas saben cómo gestionar sus elementos secundarios** Cada pantalla incorpora una propiedad integrada, que es un conjunto de pantallas, en forma de elementos secundarios. Esta colección se determina mediante el diseño de la jerarquía de pantallas en el panel Pantallas. Las pantallas pueden tener un número cualquiera de elementos secundarios (incluso ninguno) que, a su vez, también pueden tener elementos secundarios.

**Las pantallas pueden ocultar/mostrar sus elementos secundarios** Puesto que las pantallas son, básicamente, una colección de clips de película anidados, una pantalla puede controlar la visibilidad de sus elementos secundarios. En el caso de las aplicaciones de formularios, los elementos de toda una pantalla se visualizan a la vez de forma predeterminada; en el caso de las presentaciones de diapositivas, las pantallas individuales se muestran de una en una.

**Eventos de difusión de pantallas** Por ejemplo, puede activar un sonido para que se reproduzca o empezar a reproducir algún archivo de vídeo cuando se visualice una pantalla concreta.

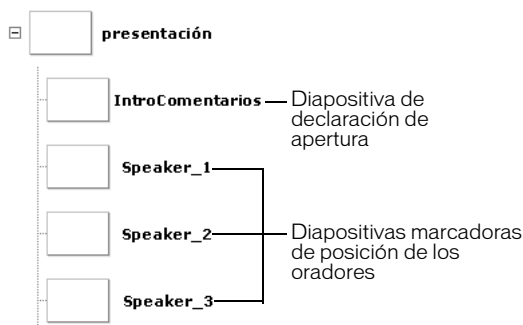
## Carga de contenido externo en pantallas (sólo para Flash Professional)

La clase Screen amplía la clase Loader (véase [“Componente Loader” en la página 339](#)), lo cual proporciona la capacidad de gestionar y cargar con facilidad archivos SWF (y JPEG) externos. La clase Loader contiene una propiedad denominada `contentPath` que especifica el URL de un archivo SWF o JPEG externo o el identificador de vínculo de un clip de película de la biblioteca.

Con esta función, puede cargar un árbol de pantallas externo (o cualquier archivo SWF externo) como elemento secundario de cualquier nodo de pantalla. Esto proporciona una manera útil de dividir en módulos las películas basadas en pantallas y dividirlos en varios archivos SWF independientes.

Por ejemplo, suponga que dispone de una presentación de diapositivas en la que tres personas van a aportar, cada una, una sola sección. Puede solicitar a cada orador que cree una presentación de diapositivas independiente (SWF). A continuación puede crear una “presentación de diapositivas maestra” que contenga tres diapositivas marcadoras de posición, una para cada presentación de diapositivas que creen los oradores. Para cada diapositiva marcadora de posición, puede hacer que su propiedad `contentPath` señale a cada uno de los archivos SWF.

Por ejemplo, la presentación de diapositivas “maestra” se puede organizar, como se muestra en la ilustración siguiente:



*Estructura de la presentación de diapositivas SWF “maestra”*

Suponga que los oradores le hayan proporcionado tres archivos SWF: `speaker_1.swf`, `speaker_2.swf` y `speaker_3.swf`. Puede ensamblar con facilidad la presentación global señalando la propiedad `contentPath` de cada diapositiva marcadora de posición mediante `ActionScript` o estableciendo la propiedad `contentPath` del inspector de propiedades para cada diapositiva.

```
Speaker_1.contentPath = speaker_1.swf;
Speaker_2.contentPath = speaker_2.swf;
Speaker_3.contentPath = speaker_3.swf;
```

También puede establecer la propiedad `contentPath` de cada diapositiva mediante el inspector de propiedades. Tenga en cuenta que, de forma predeterminada, cuando se establece la propiedad `contentPath` de una diapositiva en el inspector de propiedades (o con código, como se mostraba anteriormente), el archivo SWF especificado se cargará en el momento en que se haya cargado el archivo SWF de la “presentación maestra”. Para reducir el tiempo de carga inicial, considere la posibilidad de establecer la propiedad `contentPath` con un controlador `on(reveal)` asociado con cada diapositiva.

```
// Asociado con la diapositiva Speaker_1
on(reveal) {
    this.contentPath="speaker_1.swf";
}
```

Como alternativa, podría establecer `false` como valor de la propiedad `autoLoad` de la diapositiva (heredada de la clase `Loader`) y, a continuación, realizar una llamada al método `load()` en la diapositiva (también heredada de la clase `Loader`) cuando se haya mostrado la diapositiva.

```
// Asociado con la diapositiva Speaker_1
on(reveal) {
    this.load();
}
```

## Creación de referencias a pantallas cargadas con `ActionScript`

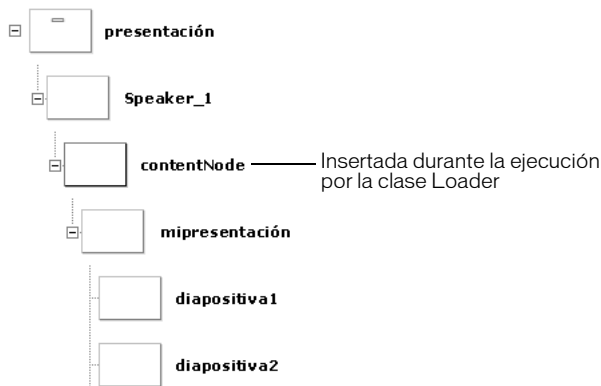
La clase `Loader` crea un clip de película interno denominado `contentNode` en el que carga el archivo SWF o JPEG especificado por la propiedad `contentPath`. Este clip de película añade, efectivamente, un nodo de pantalla adicional entre la diapositiva “marcadora de posición” (que ha creado en la presentación “maestra” anterior) y la primera diapositiva de la presentación de diapositivas cargada.

Por ejemplo, suponga que el archivo SWF creado para la diapositiva marcadora de posición `Speaker_1` (véase la ilustración anterior) tuviera la estructura siguiente, como se muestra en el panel Contorno de pantalla:



*Estructura de la presentación de diapositivas SWF “Speaker 1”*

Durante la ejecución, cuando el archivo SWF Speaker 1 se carga en la diapositiva marcadora de posición, toda la presentación de diapositivas tiene la estructura siguiente:



*Estructura de la presentación “maestra” y “speaker” (ejecución)*

Las propiedades y los métodos de las clases Screen, Slide y Form “ignoran” este nodo `contentHolder` en la medida de lo posible. Es decir, en la ilustración anterior, la diapositiva denominada mipresentación (y sus subdiapositivas) forma parte del árbol de diapositivas contiguo que nace en la diapositiva presentación y no se considera un subárbol independiente.

## Resumen de métodos de la clase Screen

| Método                                  | Descripción                                                             |
|-----------------------------------------|-------------------------------------------------------------------------|
| <a href="#">Screen.getChildScreen()</a> | Devuelve la pantalla secundaria de esta pantalla en un índice concreto. |

Hereda todos los métodos de [UIObject](#), [UIComponent](#), [View](#) y [Componente Loader](#).

## Resumen de propiedades de la clase Screen

| Propiedad                                   | Descripción                                                                                                                                     |
|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Screen.currentFocusedScreen</a> | Devuelve la pantalla que contiene la selección actual global.                                                                                   |
| <a href="#">Screen.indexInParent</a>        | Devuelve el índice de la pantalla (basado en cero) en la lista de pantallas secundarias de su pantalla principal.                               |
| <a href="#">Screen.numChildScreens</a>      | Devuelve el número de pantallas secundarias que contiene la pantalla.                                                                           |
| <a href="#">Screen.parentIsScreen</a>       | Devuelve un valor booleano ( <code>true</code> o <code>false</code> ) que indica si el objeto principal de la pantalla también es una pantalla. |
| <a href="#">Screen.rootScreen</a>           | Devuelve la pantalla raíz del subárbol que contiene la pantalla.                                                                                |

Hereda todas las propiedades de [UIObject](#), [UIComponent](#), [View](#) y [Componente Loader](#).

## Resumen de eventos de la clase Screen

| Evento                                    | Descripción                                                                                                                                         |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Screen.allTransitionsInDone</code>  | Se difunde cuando todas las transiciones de entrada aplicadas a una pantalla han finalizado.                                                        |
| <code>Screen.allTransitionsOutDone</code> | Se difunde cuando todas las transiciones de salida aplicadas a una pantalla han finalizado.                                                         |
| <code>Screen.mouseDown</code>             | Se difunde cuando se ha presionado el botón del ratón sobre un objeto (forma o clip de película) que es propiedad directa de la pantalla.           |
| <code>Screen.mouseDownSomewhere</code>    | Se difunde cuando se ha presionado el botón del ratón en algún lugar del escenario, pero no necesariamente en un objeto propiedad de esta pantalla. |
| <code>Screen.mouseMove</code>             | Se difunde cuando el ratón se mueve mientras se encuentra sobre una pantalla.                                                                       |
| <code>Screen.mouseOut</code>              | Se difunde cuando el ratón se mueve del interior al exterior de la pantalla.                                                                        |
| <code>Screen.mouseOver</code>             | Se difunde cuando el ratón se mueve del exterior al interior de la pantalla.                                                                        |
| <code>Screen.mouseUp</code>               | Se difunde cuando el botón del ratón se ha soltado sobre un objeto (forma o clip de película) que es propiedad directa de la pantalla.              |
| <code>Screen.mouseUpSomewhere</code>      | Se difunde cuando el botón del ratón se ha soltado en algún lugar del escenario, pero no necesariamente en un objeto propiedad de esta pantalla.    |

Hereda todos los eventos de [UIObject](#), [UIComponent](#), [View](#) y [Componente Loader](#).

### Screen.allTransitionsInDone

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

```
on(allTransitionsInDone) {  
    // aquí código propio  
}  
listenerObject = new Object();  
listenerObject.allTransitionsInDone = function(eventObject){  
    // introduzca aquí el código propio  
}  
screenObj.addEventListener("allTransitionsInDone", listenerObject)
```

## Descripción

Evento; se difunde cuando todas las transiciones de entrada aplicadas a esta pantalla han finalizado. El administrador de transiciones asociado con *myScreen* difunde el evento `allTransitionsInDone`.

## Ejemplo

En el ejemplo siguiente, un botón (`nextSlide_btn`) contenido en la diapositiva `mySlide` se visualiza después de que todas las transiciones de entrada aplicadas a `mySlide` hayan finalizado.

```
// Asociado con mySlide:
on(allTransitionsInDone) {
    this.nextSlide_btn._visible = true;
}
```

## Véase también

[Screen.allTransitionsOutDone](#)

## Screen.allTransitionsOutDone

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(allTransitionsOutDone) {
    // aquí código propio
}
listenerObject = new Object();
listenerObject.mouseDownOutside = function(eventObject){
    // introduzca aquí el código propio
}
screenObj.addEventListener("allTransitionsOutDone", listenerObject)
```

## Descripción

Evento; se difunde cuando todas las transiciones de salida aplicadas a la pantalla han finalizado. El administrador de transiciones asociado con *myScreen* difunde el evento `allTransitionsOutDone`.

## Véase también

[Screen.currentFocusedScreen](#)

## Screen.currentFocusedScreen

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

```
mx.screens.Screen.currentFocusedScreen
```

## Descripción

Propiedad estática (sólo lectura); devuelve una referencia al objeto de pantalla “última diapositiva hoja” que contiene la selección actual global. La selección puede estar en la propia pantalla o en un clip de película, objeto de texto o componente dentro de esa pantalla. Se establece de forma predeterminada en `null` si no hay una selección actual.

## Ejemplo

El ejemplo siguiente muestra el nombre de la pantalla seleccionada actualmente en el panel Salida.

```
var currentFocus:mx.screens.Screen = mx.screens.Screen.currentFocusedScreen;
trace("Current screen is: " + currentFocus._name);
```

## Screen.getChildScreen()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

```
myScreen.getChildScreen(index)
```

## Parámetros

*childIndex*    Número que indica el índice (basado en cero) de la pantalla secundaria que debe devolverse.

## Valor devuelto

Un objeto Screen.

## Descripción

Método; devuelve el objeto Screen secundario de *myScreen* cuyo índice es *childIndex*.

## Ejemplo

El ejemplo siguiente muestra en el panel Salida los nombres de todas las pantallas secundarias que pertenecen a la pantalla raíz denominada *presentación*.

```
for (var i:Number = 0; i < _root.presentación.numChildScreens; i++) {
    var childScreen:mx.screens.Screen = _root.presentación.getChildScreen(i);
    trace(childScreen._name);
}
```



## Screen.indexInParent

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myScreen*.indexInParent

### Descripción

Propiedad (sólo lectura); contiene el índice (basado en cero) de *myScreen* en la lista de subpantallas de su pantalla principal.

### Ejemplo

En el ejemplo siguiente se muestra la posición relativa de la pantalla *myScreen* en la lista de pantallas secundarias de su pantalla principal.

```
var numChildren:Number = myScreen._parent.numChildScreens;  
var myIndex:Number = myScreen.indexInParent;  
trace("I'm child slide # " + myIndex + " out of " + numChildren + " screens.");
```

## Screen.mouseDown

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(mouseDown) {  
    // aquí código propio  
}  
listenerObject = new Object();  
listenerObject.mouseDown = function(eventObj){  
    // introduzca aquí el código propio  
}  
screenObj.addEventListener("mouseDown", listenerObject)
```

### Descripción

Evento; se difunde cuando se ha presionado el botón del ratón sobre un objeto (por ejemplo, una forma o un clip de película) que es propiedad directa de la pantalla.

Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObj*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

El código siguiente muestra el nombre de la pantalla que ha capturado el evento de ratón en el panel Salida.

```
on(mouseDown) {  
    trace("Mouse down event on: " + eventObj.target._name);  
}
```

## Screen.mouseDownSomewhere

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(mouseDown) {  
    // aquí código propio  
}  
listenerObject = new Object();  
listenerObject.mouseDownSomewhere = function(eventObject){  
    // introduzca aquí el código propio  
}  
screenObj.addEventListener("mouseDownSomewhere", listenerObject)
```

### Descripción

Evento; se difunde cuando se presiona el botón del ratón, pero no necesariamente en la pantalla especificada.

Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObj*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Screen.mouseMove

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(mouseDown) {  
    // aquí código propio  
}  
listenerObject = new Object();  
listenerObject.mouseMove = function(eventObject){  
    // introduzca aquí el código propio  
}  
screenObj.addEventListener("mouseMove", listenerObject)
```

## Descripción

Evento; se difunde cuando el ratón se mueve mientras se encuentra sobre la pantalla. Este evento sólo se envía cuando el ratón se encuentra sobre el recuadro de delimitación de esta pantalla.

Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObj*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

**Nota:** el uso de este evento puede afectar al rendimiento del sistema y se debe utilizar con prudencia.

## Screen.mouseOut

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(mouseOut) {  
    // aquí código propio  
}  
listenerObject = new Object();  
listenerObject.mouseOut = function(eventObject){  
    // introduzca aquí el código propio  
}  
screenObj.addEventListener("mouseOut", listenerObject)
```

### Descripción

Evento; se difunde cuando el ratón se mueve del interior del recuadro de delimitación de la pantalla al exterior de éste.

Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObj*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

**Nota:** el uso de este evento puede afectar al rendimiento del sistema y se debe utilizar con prudencia.

## Screen.mouseOver

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(mouseDown) {  
    // aquí código propio  
}  
listenerObject = new Object();  
listenerObject.mouseOver = function(eventObject){  
    // introduzca aquí el código propio  
}  
screenObj.addEventListener("mouseOver", listenerObject)
```

### Descripción

Evento; se difunde cuando el ratón se mueve del exterior del recuadro de delimitación de la pantalla al interior de éste.

Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObj*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

**Nota:** el uso de este evento puede afectar al rendimiento del sistema y se debe utilizar con prudencia.

## Screen.mouseUp

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(mouseUp) {  
    // aquí código propio  
}  
listenerObject = new Object();  
listenerObject.mouseUP = function(eventObject){  
    // introduzca aquí el código propio  
}  
screenObj.addEventListener("mouseUp", listenerObject)
```

### Descripción

Evento; se difunde cuando se suelta el ratón sobre la pantalla.

Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObj*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Screen.mouseUpSomewhere

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(mouseUpSomewhere) {  
    // aquí código propio  
}  
listenerObject = new Object();  
listenerObject.mouseUpSomewhere = function(eventObject){  
    // introduzca aquí el código propio  
}  
screenObj.addEventListener("mouseUpSomewhere", listenerObject)
```

### Descripción

Evento; se difunde cuando se presiona el botón del ratón, pero no necesariamente sobre la pantalla especificada.

Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObj*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Screen.numChildScreens

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myScreen.numChildScreens
```

### Descripción

Propiedad (sólo lectura); devuelve el número de pantallas secundarias que contiene *myScreen*.

### Ejemplo

El ejemplo siguiente muestra los nombres de todas las pantallas secundarias que pertenecen a *myScreen*.

```
var howManyKids:Number = myScreen.numChildScreens;  
for(i=0; i<howManyKids; i++) {  
    var childScreen = myScreen.getChildScreen(i);  
    trace(childScreen._name);  
}
```

### Véase también

[Screen.getChildScreen\(\)](#)

## Screen.parentIsScreen

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myScreen.parentIsScreen*

### Descripción

Propiedad (sólo lectura): devuelve un valor Booleano (*true* o *false*) que indica si el objeto principal de la pantalla especificada también es una pantalla (*true*) o no (*false*). Si su valor es *false*, *myScreen* está en la raíz de su jerarquía de pantallas.

### Ejemplo

El código siguiente determina si el objeto principal de la pantalla *myScreen* también es una pantalla. En tal caso, se supone que *myScreen* es la diapositiva raíz o maestra de la presentación y, por lo tanto, no tiene diapositivas del mismo nivel. De no ser así, si *myScreen.parentIsScreen* es *true*, el número de diapositivas del mismo nivel de *myScreen* aparecerá en el panel Salida.

```
if (myScreen.parentIsScreen) {  
    trace("I have "+myScreen._parent.numChildScreens+" sibling screens");  
} else {  
    trace("I am the root screen and have no siblings");  
}
```

## Screen.rootScreen

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myScreen.rootScreen*

### Descripción

Propiedad (sólo lectura); devuelve la pantalla situada en la parte superior de la jerarquía de pantallas que contiene *myScreen*.

### Ejemplo

El ejemplo siguiente muestra el nombre de la pantalla raíz que contiene la pantalla *myScreen*.

```
var myRoot:mx.screens.Screen = myScreen.rootScreen;
```

## Componente ScrollPane

El componente ScrollPane muestra clips de película, archivos JPEG y archivos SWF en un área desplazable. Se pueden activar barras de desplazamiento para que muestren imágenes en un área limitada. También es posible mostrar el contenido que se carga desde una ubicación local o a través de Internet. El uso de ActionScript permite definir el contenido del panel de desplazamiento durante la edición y en tiempo de ejecución.

Cuando el panel de desplazamiento está seleccionado, se seleccionarán los marcadores si el contenido del panel de desplazamiento tiene tabulaciones válidas. Después de la última tabulación del contenido, la selección se desplazará al componente siguiente. La selección no se dirige nunca a las barras de desplazamiento vertical y horizontal del panel de desplazamiento.

La instancia de ScrollPane se selecciona cuando el usuario hace clic sobre ella o presiona el tabulador hasta su posición. Cuando la selección esté sobre la instancia de ScrollPane, podrá controlarla con las teclas siguientes:

| Tecla         | Descripción                                                               |
|---------------|---------------------------------------------------------------------------|
| Flecha abajo  | Mueve el contenido una línea de desplazamiento vertical hacia arriba.     |
| Fin           | Mueve el contenido hasta la parte inferior del panel de desplazamiento.   |
| Izquierda     | Mueve el contenido una línea de desplazamiento horizontal a la derecha.   |
| Inicio        | Mueve el contenido hasta la parte superior del panel de desplazamiento.   |
| Av Pág        | Mueve el contenido una página vertical hacia arriba.                      |
| Re Pág        | Mueve el contenido una página vertical hacia abajo.                       |
| Derecha       | Mueve el contenido una línea de desplazamiento horizontal a la izquierda. |
| Flecha arriba | Mueve el contenido una línea de desplazamiento vertical hacia abajo.      |

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase FocusManager” en la página 292](#).

La vista previa dinámica de cada instancia de ScrollPane refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes.

## Utilización del componente ScrollPane

Puede utilizar un panel de desplazamiento para mostrar cualquier contenido que sea más grande que el área en la que se ha cargado. Por ejemplo, si tiene una imagen grande que debe mostrarse en un espacio reducido de una aplicación, puede cargarla en un panel de desplazamiento.

Para definir un panel de desplazamiento que permita a los usuarios arrastrar el contenido en el panel, defina el parámetro `scrollDrag` en `true`; en el contenido aparecerá un cursor en forma de mano. A diferencia de lo que ocurre en la mayoría de componentes, los eventos se difunden al presionar el botón del ratón y siguen difundiéndose hasta que se suelta el botón. Si el contenido de un panel de desplazamiento tiene tabulaciones válidas, deberá definir `scrollDrag` en `false` ya que, de lo contrario, cada interacción del ratón con el contenido invocará un arrastre del desplazamiento.

## Parámetros de ScrollPane

Los siguientes son parámetros de edición que se pueden definir para cada instancia del componente ScrollPane en el inspector de propiedades o en el panel Inspector de componentes:

**contentPath** indica el contenido que se va a cargar en el panel de desplazamiento. Este valor puede ser una ruta relativa a un archivo SWF o JPEG local, o bien una ruta relativa o absoluta a un archivo en Internet. También puede ser el identificador de vinculación de un símbolo de clip de película de la biblioteca definido en Exportar para ActionScript.

**hLineScrollSize** indica el número de unidades que se mueve la barra de desplazamiento horizontal cada vez que se presiona un botón de flecha. El valor predeterminado es 5.

**hPageScrollSize** indica el número de unidades que se mueve la barra de desplazamiento horizontal cada vez que se presiona la guía de deslizamiento. El valor predeterminado es 20.

**hScrollPolicy** muestra las barras de desplazamiento horizontal. El valor puede ser "on", "off" o "auto". El valor predeterminado es "auto".

**scrollDrag** es un valor booleano que permite al usuario desplazar el contenido en el panel de desplazamiento (true) o no (false). El valor predeterminado es false.

**vLineScrollSize** indica el número de unidades que se mueve la barra de desplazamiento vertical cada vez que se presiona un botón de flecha. El valor predeterminado es 5.

**vPageScrollSize** indica el número de unidades que se mueve la barra de desplazamiento vertical cada vez que se presiona la guía de deslizamiento. El valor predeterminado es 20.

**vScrollPolicy** muestra las barras de desplazamiento vertical. El valor puede ser "on", "off" o "auto". El valor predeterminado es "auto".

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes de ScrollPane utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase ScrollPane](#).

## Creación de aplicaciones con el componente ScrollPane

En el procedimiento siguiente se explica cómo añadir un componente ScrollPane a una aplicación durante la edición. En este ejemplo, el panel de desplazamiento carga un archivo SWF que contiene un logotipo.

**Para crear una aplicación con el componente ScrollPane, siga este procedimiento:**

- 1 Arrastre un componente ScrollPane desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de la instancia **myScrollPane**.
- 3 En el inspector de propiedades, introduzca **logo.swf** para el parámetro contentPath.
- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
scrollListener = new Object();
scrollListener.scroll = function (evt){
    txtPosition.text = myScrollPane.vPosition;
}
myScrollPane.addEventListener("scroll", scrollListener);
completeListener = new Object();
completeListener.complete = function() {
    trace("logo.swf has completed loading.");
}
myScrollPane.addEventListener("complete", completeListener);
```



El primer bloque del código es un controlador de eventos `scroll` en la instancia `myScrollPane` que muestra el valor de la propiedad `vPosition` en una instancia de `TextField` denominada `txtPosition` que se encuentra ya en el escenario. El segundo bloque del código crea un controlador de eventos para el evento `complete` que envía un mensaje al panel Salida.

## Personalización del componente `ScrollPane`

El componente `ScrollPane` puede transformarse horizontal y verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase [UIObject.setSize\(\)](#)) o cualquier método o propiedad aplicable de la clase `ScrollPane`. Véase [Clase ScrollPane](#). Si `ScrollPane` es demasiado pequeño, es posible que el contenido no se muestre correctamente.

`ScrollPane` coloca el punto de registro de su contenido en la esquina superior izquierda del panel.

Cuando se desactiva la barra de desplazamiento horizontal, la barra vertical se muestra de arriba abajo sobre el lado derecho del panel de desplazamiento. Si se desactiva la barra de desplazamiento vertical, la barra horizontal se muestra de izquierda a derecha sobre el borde inferior del panel de desplazamiento. También se pueden desactivar las dos barras de desplazamiento.

Cuando se cambia el tamaño del panel de desplazamiento, los botones no se modifican, la guía de deslizamiento y el deslizador se expanden o se contraen y cambia el tamaño de sus áreas activas.

## Utilización de estilos con el componente `ScrollPane`

`ScrollPane` no admite estilos, pero las barras de desplazamiento que utiliza, sí.

## Utilización de aspectos con el componente `ScrollPane`

A diferencia de las barras de desplazamiento que utiliza, el componente `ScrollPane` no tiene aspectos propios.

## Clase `ScrollPane`

**Herencia** `UIObject` > `UIComponent` > `View` > `ScrollView` > `ScrollPane`

**Nombre de clase de `ActionScript`** `mx.containers.ScrollPane`

Las propiedades de la clase `ScrollPane` permiten definir el contenido, controlar el progreso de carga y ajustar la cantidad de desplazamiento en tiempo de ejecución.

Si una propiedad de la clase `ScrollPane` se define con `ActionScript`, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

Para definir un panel de desplazamiento que permita a los usuarios arrastrar el contenido del panel, establezca la propiedad `scrollDrag` en `true`; en el contenido aparecerá un cursor en forma de mano. A diferencia de lo que ocurre en la mayoría de componentes, los eventos se difunden al presionar el botón del ratón y siguen difundiéndose hasta que se suelta el botón. Si el contenido de un panel de desplazamiento tiene tabulaciones válidas, deberá definir `scrollDrag` en `false` ya que, de lo contrario, cada interacción del ratón con el contenido invocará un arrastre del desplazamiento.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.containers.ScrollPane.version);
```

**Nota:** el código siguiente devuelve `undefined`: `trace(myScrollPaneInstance.version);`.

## Resumen de métodos de la clase `ScrollPane`

| Método                                   | Descripción                                                         |
|------------------------------------------|---------------------------------------------------------------------|
| <code>ScrollPane.getBytesLoaded()</code> | Devuelve el número de bytes del contenido cargado.                  |
| <code>ScrollPane.getBytesTotal()</code>  | Devuelve el número total de bytes del contenido que se va a cargar. |
| <code>ScrollPane.refreshPane()</code>    | Vuelve a cargar el contenido del panel de desplazamiento.           |

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

## Resumen de propiedades de la clase `ScrollPane`

| Método                                  | Descripción                                                                                                                                                                                                      |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ScrollPane.content</code>         | Referencia al contenido cargado en el panel de desplazamiento.                                                                                                                                                   |
| <code>ScrollPane.contentPath</code>     | URL absoluta o relativa del archivo SWF o JPEG que se carga en el panel de desplazamiento.                                                                                                                       |
| <code>ScrollPane.hLineScrollSize</code> | Cantidad de contenido que se va a desplazar horizontalmente cuando se presione un botón de flecha.                                                                                                               |
| <code>ScrollPane.hPageScrollSize</code> | Cantidad de contenido que se va a desplazar horizontalmente cuando se presione la guía de deslizamiento.                                                                                                         |
| <code>ScrollPane.hPosition</code>       | Posición horizontal en píxeles del panel de desplazamiento.                                                                                                                                                      |
| <code>ScrollPane.hScrollPolicy</code>   | Estado de la barra de desplazamiento horizontal. Puede estar siempre activada ("on"), siempre desactivada ("off"), o bien activarse cuando sea necesario ("auto"). El valor predeterminado es "auto".            |
| <code>ScrollPane.scrollDrag</code>      | Indica si va a haber desplazamiento cuando el usuario presione y arrastre en el panel <code>ScrollPane</code> ( <code>true</code> ) o no ( <code>false</code> ). El valor predeterminado es <code>false</code> . |
| <code>ScrollPane.vLineScrollSize</code> | Cantidad de contenido que se va a desplazar verticalmente cuando se presione un botón de flecha.                                                                                                                 |
| <code>ScrollPane.vPageScrollSize</code> | Cantidad de contenido que se va a desplazar verticalmente cuando se presione la guía de deslizamiento.                                                                                                           |
| <code>ScrollPane.vPosition</code>       | Posición vertical en píxeles del panel de desplazamiento.                                                                                                                                                        |
| <code>ScrollPane.vScrollPolicy</code>   | Estado de la barra de desplazamiento vertical. Puede estar siempre activada ("on"), siempre desactivada ("off"), o bien activarse cuando sea necesario ("auto"). El valor predeterminado es "auto".              |

Hereda todas las propiedades de [UIObject](#) y [UIComponent](#).

## Resumen de eventos de la clase ScrollPane

| Método                           | Descripción                                                              |
|----------------------------------|--------------------------------------------------------------------------|
| <code>ScrollPane.complete</code> | Se difunde cuando se carga el contenido del panel de desplazamiento.     |
| <code>ScrollPane.progress</code> | Se difunde mientras se carga el contenido de la barra de desplazamiento. |
| <code>ScrollPane.scroll</code>   | Se difunde al presionar la barra de desplazamiento.                      |

Hereda todos los eventos de [UIObject](#) y [UIComponent](#).

### ScrollPane.complete

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.

#### Sintaxis

Sintaxis 1:

```
on(complete){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.complete = function(eventObject){  
    ...  
}  
scrollPaneInstance.addEventListener("complete", listenerObject)
```

#### Descripción

Evento; se difunde a todos los detectores registrados cuando el contenido ha terminado de cargarse.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `ScrollPane`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myScrollPaneComponent` del componente `ScrollPane`, envía “\_level0.myScrollPaneComponent” al panel Salida:

```
on(complete){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*ScrollPaneInstance*) distribuye un evento (en este caso, *complete*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método *UIEventDispatcher.addEventListener()* de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 602](#).

### Ejemplo

En el ejemplo siguiente se crea un objeto detector con un controlador de eventos *complete* para la instancia *ScrollPane*:

```
form.complete = function(eventObj){  
    // insertar código para controlar el evento  
}  
scrollPane.addEventListener("complete",form);
```

## ScrollPane.content

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*ScrollPaneInstance*.content

### Descripción

Propiedad (sólo lectura); referencia al contenido del panel de desplazamiento. El valor es *undefined* hasta que comienza la carga.

### Ejemplo

En este ejemplo se establece la variable *mcLoaded* en el valor de la propiedad *content*:

```
var mcLoaded = scrollPane.content;
```

### Véase también

[ScrollPane.contentPath](#)

## ScrollPane.contentPath

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*ScrollPaneInstance*.contentPath

### Descripción

Propiedad; cadena que indica una URL absoluta o relativa del archivo SWF o JPEG que se carga en el panel de desplazamiento. La ruta relativa debe hacer referencia al archivo SWF que carga el contenido.

Si carga el contenido utilizando una URL relativa, el contenido cargado deberá ser relativo a la ubicación del archivo SWF que contiene el panel de desplazamiento. Por ejemplo, una aplicación que utilice un componente ScrollPane situado en el directorio /scrollpane/nav/example.swf podría cargar el contenido del directorio /scrollpane/content/flash/logo.swf con la siguiente propiedad contentPath: ". ./content/flash/logo.swf".

### Ejemplo

En el ejemplo siguiente se indica al panel de desplazamiento que muestre el contenido de una imagen procedente de Internet:

```
ScrollPane.contentPath ="http://imagecache2.allposters.com/images/43/033_302.jpg";
```

En el ejemplo siguiente se indica al panel de desplazamiento que muestre el contenido de un símbolo de la biblioteca:

```
ScrollPane.contentPath ="movieClip_Name";
```

En el ejemplo siguiente se indica al panel de desplazamiento que muestre el contenido del archivo local "logo.swf":

```
ScrollPane.contentPath ="logo.swf";
```

### Véase también

[ScrollPane.content](#)

## ScrollPane.getBytesLoaded()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*ScrollPaneInstance*.getBytesLoaded()

## Parámetros

Ninguno.

## Valor devuelto

Número de bytes cargados en el panel de desplazamiento.

## Descripción

Método; devuelve el número de bytes cargados en la instancia `ScrollPane`. Mientras se carga el contenido, invoque este método en intervalos regulares para comprobar el progreso de la operación.

## Ejemplo

En este ejemplo se crea una instancia de la clase `ScrollPane` denominada `scrollPane`. A continuación, se define un objeto detector denominado `loadListener` con un controlador de eventos `progress` que llama al método `getBytesLoaded()` para que le ayude a determinar el progreso de carga:

```
createClassObject(mx.containers.ScrollPane, "scrollPane", 0);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que genera el evento change
    var bytesLoaded = scrollPane.getBytesLoaded();
    var bytesTotal = scrollPane.getBytesTotal();
    var percentComplete = Math.floor(bytesLoaded/bytesTotal);

    if (percentComplete < 5 ) // la carga acaba de comenzar
    {
        trace(" Starting loading contents from internet");
    }
    else if(percentComplete = 50) //50% completado
    {
        trace(" 50% contents downloaded ");
    }
}
scrollPane.addEventListener("progress", loadListener);
scrollPane.contentPath = "http://www.geocities.com/hcls_matrix/Images/homeview5.jpg";
```

## ScrollPane.getBytesTotal()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
scrollPaneInstance.getBytesTotal()
```

### Parámetros

Ninguno.

**Valor devuelto**

Un número.

**Descripción**

Método; devuelve el número total de bytes que se van a cargar en la instancia ScrollPane.

**Véase también**

[ScrollPane.getBytesLoaded\(\)](#)

**ScrollPane.hLineScrollSize****Disponibilidad**

Flash Player 6 versión 79.

**Edición**

Flash MX 2004.

**Sintaxis**

*scrollPaneInstance.hLineScrollSize*

**Descripción**

Propiedad; número de píxeles que se va a mover el contenido al presionar la flecha izquierda o derecha en la barra de desplazamiento horizontal. El valor predeterminado es 5.

**Ejemplo**

En este ejemplo se incrementa en 10 unidades el desplazamiento horizontal:

```
scrollPane.hLineScrollSize = 10;
```

**ScrollPane.hPageScrollSize****Disponibilidad**

Flash Player 6 versión 79.

**Edición**

Flash MX 2004.

**Sintaxis**

*scrollPaneInstance.hPageScrollSize*

**Descripción**

Propiedad; número de píxeles que se va a mover el contenido al presionar la guía en la barra de desplazamiento horizontal. El valor predeterminado es 20.

**Ejemplo**

En este ejemplo se incrementa en 30 unidades el desplazamiento horizontal de la página:

```
scrollPane.hPageScrollSize = 30;
```

## ScrollPane.hPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*scrollPaneInstance.hPosition*

### Descripción

Propiedad; posición en píxeles de la barra de desplazamiento horizontal. La posición 0 se encuentra a la izquierda de la barra.

### Ejemplo

En este ejemplo se establece la barra de desplazamiento en 20:

```
scrollPane.hPosition = 20;
```

## ScrollPane.hScrollPolicy

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*scrollPaneInstance.hScrollPolicy*

### Descripción

Propiedad; determina si la barra de desplazamiento horizontal está presente siempre ("on"), nunca ("off") o aparece en función del tamaño de la imagen ("auto"). El valor predeterminado es "auto".

### Ejemplo

El código siguiente mantiene siempre activas las barras desplazamiento:

```
scrollPane.hScrollPolicy = "on";
```

## ScrollPane.progress

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.



## Sintaxis

Sintaxis 1:

```
on(progress){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.progress = function(eventObject){  
    ...  
}  
scrollPaneInstance.addEventListener("progress", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados mientras se carga el contenido. El evento `progress` no siempre se difunde; el evento `complete` puede difundirse sin que se distribuyan eventos `progress`. Esto sucede especialmente cuando el contenido cargado es un archivo local. Este evento se activa cuando el contenido empieza a cargarse al establecer el valor de la propiedad `contentPath`.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `ScrollPane`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `mySPComponent` del componente `ScrollPane`, envía “\_level0.mySPComponent” al panel Salida:

```
on(progress){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`ScrollPaneInstance`) distribuye un evento (en este caso, `progress`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (`listenerObject`) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (`eventObject`) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

El código siguiente crea una instancia de `ScrollPane`, llamada `scrollPane` y un objeto detector con un controlador de eventos para el evento `progress` que envía un mensaje al panel Salida con el número de bytes del contenido que se ha cargado:

```
createClassObject(mx.containers.ScrollPane, "scrollPane", 0);
loadListener = new Object();
loadListener.progress = function(eventObj){
    // eventObj.target es el componente que genera el evento progress
    // en este caso, scrollPane
    trace("logo.swf has loaded " + scrollPane.getBytesLoaded() + " Bytes.");
    // llevar seguimiento de progreso de carga
}
scrollPane.addEventListener("complete", loadListener);
scrollPane.contentPath = "logo.swf";
```

## ScrollPane.refreshPane()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
scrollPaneInstance.refreshPane()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; actualiza el panel de desplazamiento después de cargar el contenido. Este método vuelve a cargar el contenido. Utilice este método si, por ejemplo, ha cargado un formulario en `ScrollPane` y se ha modificado una propiedad de entrada (por ejemplo, en un campo de texto) con ActionScript. En tal caso, llame a `refreshPane()` para volver a cargar el mismo formulario con los nuevos valores para las propiedades de entrada.

## Ejemplo

En el ejemplo siguiente se actualiza la instancia `sp` del panel de desplazamiento:

```
sp.refreshPane();
```

## ScrollPane.scroll

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

Sintaxis 1:

```
on(scroll){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.scroll = function(eventObject){  
    ...  
}  
scrollPaneInstance.addEventListener("scroll", listenerObject)
```

## Objeto Event

Además de las propiedades estándar del objeto Event, hay una propiedad `type` definida para el evento `scroll`; el valor es `"scroll"`. También hay una propiedad `direction` con los valores `"vertical"` y `"horizontal"` posibles.

## Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario presiona los botones de la barra de desplazamiento, el deslizador o la guía de deslizamiento. A diferencia de otros eventos, `scroll` se difunde cuando el usuario presiona la barra de desplazamiento y continúa difundiéndose hasta que la suelta.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `ScrollPane`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `sp`, envía `"_level0.sp"` al panel Salida:

```
on(scroll){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (`ScrollPaneInstance`) distribuye un evento (en este caso, `scroll`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (`listenerObject`) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (`eventObject`) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte ["Objetos Event" en la página 602](#).

## Ejemplo

En este ejemplo se crea un objeto detector `form` con una función callback `scroll` que se registra en la instancia `spInstance`. Debe rellenar `spInstance` de contenido, como en el ejemplo siguiente:

```
spInstance.contentPath = "mouse3.jpg";
form = new Object();
form.scroll = function(eventObj){
    trace("ScrollPane scrolled");
}
spInstance.addEventListener("scroll", form);
```

## Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## ScrollPane.scrollDrag

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
scrollPaneInstance.scrollDrag
```

### Descripción

Propiedad; valor booleano que indica si se va a producir un desplazamiento cuando el usuario presione y arrastre en `ScrollPane` (`true`) o no (`false`). El valor predeterminado es `false`.

## Ejemplo

En este ejemplo se activa el desplazamiento con ratón en el interior del panel de desplazamiento:

```
scrollPane.scrollDrag = true;
```

## ScrollPane.vLineScrollSize

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
scrollPaneInstance.vLineScrollSize
```

### Descripción

Propiedad; número de píxeles que se moverá el área de visualización al presionar el botón de flecha arriba o abajo en una barra de desplazamiento vertical. El valor predeterminado es 5.

### Ejemplo

Este código incrementa en 10 la cantidad del área de visualización que se mueve al presionar los botones de flecha de la barra de desplazamiento:

```
scrollPane.vLineScrollSize = 10;
```

## ScrollPane.vPageScrollSize

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

```
scrollPaneInstance.vPageScrollSize
```

### Descripción

Propiedad; número de píxeles que se moverá el área de visualización al presionar la guía en una barra de desplazamiento vertical. El valor predeterminado es 20.

### Ejemplo

Este código incrementa en 30 la cantidad del área de visualización que se mueve al presionar los botones de flecha de la barra de desplazamiento vertical:

```
scrollPane.vPageScrollSize = 30;
```

## ScrollPane.vPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
scrollPaneInstance.vPosition
```

### Descripción

Propiedad; posición en píxeles de la barra de desplazamiento vertical. El valor predeterminado es 0.

## ScrollPane.vScrollPolicy

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
scrollPaneInstance.vScrollPolicy
```

## Descripción

Propiedad; determina si la barra de desplazamiento vertical está presente siempre ("on"), nunca ("off") o si aparece en función del tamaño de la imagen ("auto"). El valor predeterminado es "auto".

## Ejemplo

El código siguiente mantiene siempre activas las barras de desplazamiento vertical:

```
scrollPane.vScrollPolicy = "on";
```

## Clase Slide (sólo para Flash Professional)

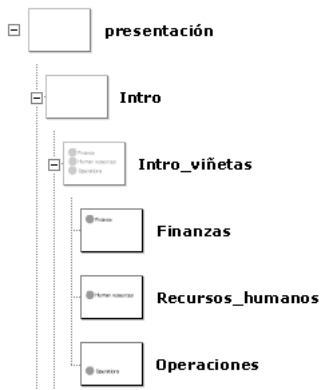
**Herencia** UIObject > UIComponent > View > Loader > Screen > Slide

**Nombre de clase de ActionScript** mx.screens.Slide

La clase Slide corresponde a un nodo de la presentación jerárquica de diapositivas. En Flash MX Professional 2004 puede crear presentaciones de diapositivas mediante el panel Contorno de pantalla. Para ver información general sobre cómo trabajar con pantallas, consulte “Información general sobre el trabajo con pantallas (sólo en Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

La clase Slide amplía la clase Screen (véase “[Clase Screen \(sólo para Flash Professional\)](#)” en la [página 486](#)) y ofrece capacidades de navegación y secuenciación integradas entre las diapositivas, además de la capacidad de asociar fácilmente transiciones entre diapositivas utilizando los comportamientos. Las diapositivas almacenan información sobre su “estado”, gracias a lo cual el usuario puede pasar a la diapositiva siguiente o anterior de una presentación y, cuando aparece la siguiente diapositiva, la anterior se oculta.

Tenga en cuenta que sólo puede desplazarse por (o “detenerse en”) las diapositivas que no contienen diapositivas secundarias ni diapositivas “hoja”. Por ejemplo, la ilustración siguiente muestra el contenido del panel Contorno de pantalla de un ejemplo de presentación de diapositivas.



Cuando se inicie dicha presentación, se “detendrá” de forma predeterminada en la diapositiva llamada Finanzas, que es la primera diapositiva de la presentación que no contiene diapositivas secundarias.

Asimismo, tenga en cuenta que las diapositivas secundarias “heredan” el aspecto visual (gráficos y el resto del contenido) de las diapositivas principales. Por ejemplo, en la ilustración anterior, además del contenido de la diapositiva Finanzas, el usuario puede ver todo el contenido de las diapositivas Intro y Presentación.

**Nota:** la clase Slide deriva de la clase Loader (véase [“Clase Loader” en la página 342](#)), que permite cargar fácilmente archivos SWF externos (o JPEG) en una diapositiva determinada. Esto permite dividir en módulos las presentaciones de diapositivas y reducir el tiempo de descarga inicial. Para más información, consulte [“Carga de contenido externo en pantallas \(sólo para Flash Professional\)” en la página 487](#).

## Utilización de la clase Slide (sólo para Flash Professional)

Los métodos y las propiedades de la clase Slide se utilizan para controlar las presentaciones de diapositivas que se crean mediante el panel Contorno de pantalla (Ventana > Pantallas para obtener información sobre la presentación de diapositivas (por ejemplo, para determinar el número de diapositivas secundarias que contiene una diapositiva principal) o para desplazarse entre las diapositivas de una presentación (por ejemplo, para crear unos botones “Diapositiva siguiente” y “Diapositiva anterior”).

También puede utilizar uno de los comportamientos incorporados para controlar las presentaciones de diapositivas que se encuentran disponibles en el panel Comportamientos (Ventana > Paneles de desarrollo > Comportamientos). Para más información sobre cómo utilizar comportamientos con diapositivas, consulte [“Adición de control a las pantallas mediante comportamientos \(sólo en Flash Professional\)” en el apartado Utilización de Flash de la Ayuda](#).

## Parámetros de diapositivas

A continuación se indican los parámetros de edición que se pueden definir para cada diapositiva en el inspector de propiedades o en el panel Inspector de componentes:

**autoKeyNav** determina si la diapositiva responde a la navegación mediante el teclado predeterminado y, en tal caso, también determina el modo en que lo hace. Para más información, consulte [Slide.autoKeyNav](#).

**autoload** indica si el contenido especificado por el parámetro `contentPath` debe cargarse automáticamente (true) o si debe esperar hasta que se llame al método `Loader.load()` (false). El valor predeterminado es true.

**contentPath** especifica el contenido de la diapositiva. Este valor puede ser el identificador de vinculación de un clip de película o la URL absoluta o relativa de un archivo SWF o JPG que se carga en la diapositiva. De forma predeterminada, el contenido cargado se recorta para que quepa en la diapositiva.

**overlayChildren** especifica si las diapositivas secundarias de la diapositiva siguen viéndose al desplazarse de una diapositiva secundaria a la siguiente (true) o no (false).

**playHidden** especifica si la diapositiva continúa reproduciéndose cuando está oculta (true) o no (false).

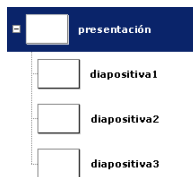
## Utilización de la clase Slide para crear una presentación de diapositivas

Los métodos y las propiedades de la clase Slide se utilizan para controlar las presentaciones de diapositivas que se crean en el panel Contorno de pantalla (Ventana > Pantallas) en el entorno de edición de Flash. Tenga en cuenta que el panel Comportamientos (Ventana > Paneles de desarrollo > Comportamientos) contiene varios comportamientos para posibilitar el desplazamiento entre diapositivas. En este ejemplo, se escribe código ActionScript para crear botones Siguiente y Anterior para presentaciones de diapositivas.

### Para crear una presentación de diapositivas con navegación:

- 1 En Flash, seleccione Archivo > Nuevo.
- 2 Haga clic en la ficha General y seleccione Presentación de Flash en el campo Tipo.
- 3 En el panel Contorno de pantalla, haga clic en el botón Insertar pantalla (+) dos veces para crear dos diapositivas debajo de la diapositiva presentación.

El panel Contorno de pantalla tiene el aspecto siguiente:



- 4 Seleccione diapositiva1 en el panel Contorno de pantalla y, utilizando la herramienta Texto, añada un campo de texto en el que se lea “Ésta es la diapositiva 1”.
- 5 Repita el paso anterior para diapositiva2 y diapositiva3 y cree campos de texto en cada diapositiva en los que se lea “Ésta es la diapositiva 2” y “Ésta es la diapositiva 3”, respectivamente.
- 6 Seleccione la diapositiva presentación y abra el panel Componentes (Ventana > Paneles de desarrollo > Componentes).
- 7 Arrastre un componente Button desde el panel Componentes a la parte inferior de la página.
- 8 En el inspector de propiedades (Ventana > Propiedades) escriba “Diapositiva siguiente” para la propiedad Label del componente Button.
- 9 Para abrir el panel Acciones si todavía no está abierto, seleccione Ventana > Paneles de desarrollo > Acciones.
- 10 Introduzca el código siguiente en el panel Acciones:

```
on(click){  
    _parent.currentSlide.gotoNextSlide();  
}
```
- 11 Pruebe el archivo SWF (Control > Probar película) y pase a la diapositiva siguiente haciendo clic en el botón Diapositiva siguiente.



## Resumen de métodos de la clase Slide

| Propiedad                              | Descripción                                                                        |
|----------------------------------------|------------------------------------------------------------------------------------|
| <code>Slide.getChildSlide()</code>     | Devuelve la diapositiva secundaria de esta diapositiva en un índice determinado.   |
| <code>Slide.gotoFirstSlide()</code>    | Desplaza al primer nodo hoja de la jerarquía de subdiapositivas de la diapositiva. |
| <code>Slide.gotoLastSlide()</code>     | Desplaza al último nodo hoja de la jerarquía de subdiapositivas de la diapositiva. |
| <code>Slide.gotoNextSlide()</code>     | Desplaza a la diapositiva siguiente.                                               |
| <code>Slide.gotoPreviousSlide()</code> | Desplaza a la diapositiva siguiente.                                               |
| <code>Slide.gotoSlide()</code>         | Desplaza a una diapositiva cualquiera.                                             |

Hereda todos los métodos de [UIObject](#), [UIComponent](#), [View](#), [Componente Loader](#) y [Clase Screen](#) (sólo para Flash Professional).

## Resumen de propiedades de la clase Slide

| Propiedad                                | Descripción                                                                                                                                                  |
|------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Slide.autoKeyNav</code>            | Determina si se utilizan las funciones del teclado predeterminado o no para pasar a la diapositiva siguiente/anterior.                                       |
| <code>Slide.currentSlide</code>          | Devuelve la diapositiva secundaria de la diapositiva que contiene la diapositiva activa.                                                                     |
| <code>Slide.currentSlide</code>          | Devuelve la diapositiva que está activa.                                                                                                                     |
| <code>Slide.currentFocusedSlide</code>   | Devuelve la última diapositiva hoja que está seleccionada.                                                                                                   |
| <code>Slide.defaultKeydownHandler</code> | Controlador callback que sustituye la navegación por las diapositivas mediante la presión de teclas predeterminada (flecha izquierda y derecha).             |
| <code>Slide.firstSlide</code>            | Devuelve la primera diapositiva secundaria de la diapositiva que no tiene secundarios.                                                                       |
| <code>Slide.getChildSlide()</code>       | Devuelve la diapositiva secundaria en un índice especificado.                                                                                                |
| <code>Slide.indexInParentSlide</code>    | Devuelve el índice de la diapositiva (basado en cero) en la lista de subdiapositivas de la diapositiva principal.                                            |
| <code>Slide.lastSlide</code>             | Devuelve la última diapositiva secundaria de la diapositiva que no tiene objetos secundarios.                                                                |
| <code>Slide.nextSlide</code>             | Devuelve la siguiente diapositiva de nodo hoja.                                                                                                              |
| <code>Slide.numChildSlides</code>        | Devuelve el número de diapositivas secundarias que contiene la diapositiva.                                                                                  |
| <code>Slide.overlayChildren</code>       | Determina si las diapositivas secundarias de la diapositiva están visibles cuando se pasa de una diapositiva secundaria a la siguiente.                      |
| <code>Slide.parentIsSlide</code>         | Devuelve un valor booleano que indica si el objeto principal de la diapositiva también es una diapositiva ( <code>true</code> ) o no ( <code>false</code> ). |

| Propiedad                        | Descripción                                                                     |
|----------------------------------|---------------------------------------------------------------------------------|
| <code>Slide.playHidden</code>    | Determina si la diapositiva continúa reproduciéndose o no mientras está oculta. |
| <code>Slide.previousSlide</code> | Devuelve la diapositiva del nodo hoja anterior.                                 |
| <code>Slide.revealChild</code>   | Devuelve la raíz del árbol de diapositivas que contiene la diapositiva.         |

Hereda todas las propiedades de [UIObject](#), [UIComponent](#), [View](#), [Componente Loader](#) y [Clase Screen](#) (sólo para Flash Professional).

## Resumen de eventos de la clase Slide

| Evento                         | Descripción                                                                                 |
|--------------------------------|---------------------------------------------------------------------------------------------|
| <code>Slide.hideChild</code>   | Se difunde cuando se ocultan todas las diapositivas secundarias ocultas de una diapositiva. |
| <code>Slide.revealChild</code> | Se difunde cuando se muestran todas las diapositivas secundarias de una diapositiva.        |

Hereda todos los eventos de [UIObject](#), [UIComponent](#), [View](#), [Componente Loader](#) y [Clase Screen](#) (sólo para Flash Professional).

## Slide.autoKeyNav

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`mySlide.autoKeyNav`

### Descripción

Propiedad; determina si la diapositiva utiliza o no las funciones del teclado predeterminado para desplazarse a la diapositiva siguiente/anterior cuando se ha seleccionado `mySlide`. Esta propiedad acepta uno de los valores de cadena siguientes: "true", "false" o "inherit". También puede sustituir el comportamiento de la utilización del teclado predeterminado mediante la propiedad [Slide.defaultKeyDownHandler](#).

También puede establecer esta propiedad mediante el inspector de propiedades.

Si se ha establecido el valor en "true" y presiona la flecha derecha (`Key.RIGHT`) o la barra espaciadora (`Key.SPACE`) cuando se ha seleccionado `mySlide`, avanzará hasta la diapositiva siguiente; si presiona la flecha izquierda (`Key.Left`) se desplazará a la diapositiva anterior.

Si se ha establecido en "false", las funciones del teclado predeterminado no tendrán efecto alguno si se ha seleccionado `mySlide`.

Si se ha establecido en "inherit", *mySlide* comprueba la propiedad `autoKeyNav` de su diapositiva principal. Si la diapositiva principal de *mySlide* también se ha establecido en "inherit", se examinará la diapositiva principal de la diapositiva principal de *mySlide* de forma sucesiva hasta encontrar una diapositiva principal cuya propiedad `autoKeyNav` se haya establecido en "true" o "false".

Si *mySlide* no tiene ninguna diapositiva principal, es decir, si (`mySlide.parentIsSlide == false`) es true se comportará como si `autoKeyNav` se hubiera establecido en true.

### Ejemplo

En este ejemplo se desactiva la navegación automática mediante teclado para la diapositiva llamada `loginSlide`.

```
_root.presentación.loginSlide.autoKeyNav = "false";
```

### Véase también

[Slide.defaultKeydownHandler](#)

## Slide.currentSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.currentSlide

### Descripción

Propiedad (sólo lectura); devuelve la diapositiva que se encuentra activa. Se trata siempre de una diapositiva "hoja", es decir una diapositiva que no contiene diapositivas secundarias.

### Ejemplo

El código siguiente, asociado con un botón de la diapositiva raíz presentación, avanza por la presentación de diapositivas hasta la diapositiva siguiente cada vez que se presiona el botón.

```
// Asociado con la instancia de botón contenida en la diapositiva presentación:  
on(press){  
    _parent.currentSlide.gotoNextSlide();  
}
```

### Véase también

[Slide.gotoNextSlide\(\)](#)

## Slide.currentChildSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.currentChildSlide

### Descripción

Propiedad (sólo lectura); devuelve la diapositiva secundaria de *mySlide* que contiene la diapositiva que está activa; devuelve *null* si no se encuentra seleccionada ninguna de las diapositivas secundarias de *mySlide*.

### Ejemplo

Observe el contorno de pantalla siguiente:

```
presentación
  diapositiva_1
    Viñeta1_1
      SubViñeta1_1_1
    Viñeta1_2
      SubViñeta1_2_1
  diapositiva_2
```

Si se toma *SubViñeta1\_1\_1* como la diapositiva actual, las sentencias siguientes serán verdaderas:

```
presentación.currentChildSlide == diapositiva_1;
diapositiva_1.currentChildSlide == Viñeta_1_1;
SubViñeta_1_1_1.currentChildSlide == null;
diapositiva_2.currentChildSlide == null;
```

### Véase también

[Slide.currentSlide](#)

## Slide.currentFocusedSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mx.screens.Slide*.currentFocusedSlide

## Descripción

Propiedad (sólo lectura); devuelve la última diapositiva hoja que está seleccionada. Puede seleccionarse una diapositiva, un objeto de texto, un clip de película o el componente de dicha diapositiva; devuelve `null` si no se ha seleccionado ningún elemento.

## Ejemplo

```
var focusedSlide = mx.screens.Slide.currentFocusedSlide;
```

## Slide.defaultKeyDownHandler

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
mySlide.defaultKeyDownHandler = function (eventObj) {  
    // aquí código propio  
}
```

### Parámetros

*eventObj* Objeto de evento con las propiedades siguientes:

- **type** Cadena que indica el tipo de evento. Los valores posibles son "keyUp" y "keyDown".
- **ascii** Número entero que representa el valor ASCII de la última tecla presionada; corresponde al valor devuelto por `Key.getAscii()`.
- **code** Número entero que representa el código de tecla de la última tecla presionada; corresponde al valor devuelto por `Key.getCode()`.
- **shiftKey** Valor booleano (`true` o `false`) que indica si la tecla Mayús está presionada actualmente (`true`) o no (`false`).
- **ctrlKey** Valor booleano (`true` o `false`) que indica si la tecla Control está presionada actualmente (`true`) o no (`false`).

### Valor devuelto

Ninguno.

### Descripción

Controlador callback; permite sustituir la navegación mediante teclado predeterminado por un controlador de teclado personalizado creado por el usuario. Por ejemplo, en lugar de utilizar las teclas de flecha izquierda y derecha para desplazarse a las diapositivas anterior y siguiente de una presentación respectivamente, puede hacer que las teclas de flecha arriba y abajo lleven a cabo dichas funciones. Para más información sobre el comportamiento de las funciones del teclado predeterminado, consulte [Slide.autoKeyNav](#).

Las funciones automáticas del teclado se activan cuando la propiedad [Slide.autoKeyNav](#) de la diapositiva actual está establecida en "true", o bien si está establecida en "inherit" y la diapositiva inmediatamente anterior a la diapositiva actual (que no está establecida en "inherit") es la diapositiva raíz de la presentación, o bien el valor `autoKeyNav` de la misma está establecido en "true".

## Ejemplo

En este ejemplo se modifican las funciones predeterminadas del teclado para las diapositivas secundarias de la diapositiva con la que se encuentra asociado el controlador `on(load)`. Este controlador utiliza las flechas arriba/abajo para el desplazamiento en lugar de las flechas izquierda/derecha.

```
on(load){
    this.defaultKeyDownHandler = function(eventObj:Object) {
        switch (eventObj.code) {
            case Key.DOWN :
                this.currentSlide.gotoNextSlide();
                break;
            case Key.UP :
                this.currentSlide.gotoPreviousSlide();
                break;
            default :
                break;
        }
    };
}
```

## Véase también

[Slide.autoKeyNav](#)

## Slide.firstSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.firstSlide

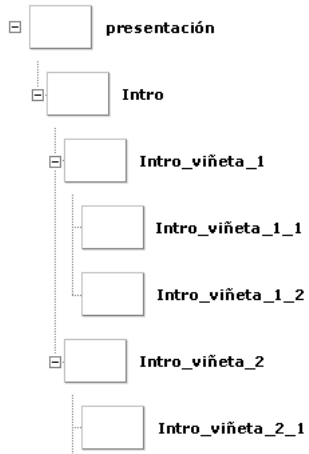
### Descripción

Propiedad (sólo lectura); devuelve la primera diapositiva secundaria de *mySlide* que no tenga diapositivas secundarias.

## Ejemplo

Por ejemplo, en la jerarquía de diapositivas que se muestra a continuación, todas las sentencias siguientes son verdaderas:

```
presentación.Intro.firstSlide == Intro_víñeta_1_1;  
presentación.Intro_víñeta_1.firstSlide == Intro_víñeta_1-1;
```



## Slide.getChildSlide()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
mySlide.getChildSlide(childIndex)
```

### Parámetros

*childIndex* Índice basado en cero de la diapositiva secundaria que se devuelve.

### Valor devuelto

Un objeto de diapositiva.

### Descripción

Método; devuelve la diapositiva secundaria de *mySlide* cuyo índice es *childIndex*. Este método resulta útil, por ejemplo, para repetir un conjunto de diapositivas secundarias cuyos índices son conocidos, tal como se muestra en el ejemplo siguiente.

## Ejemplo

En este ejemplo se muestran los nombres de todas las diapositivas secundarias de la diapositiva raíz presentación en el panel Salida.

```
var numSlides = _root.presentación.numChildSlides;
for(var slideIndex=0; slideIndex < numSlides; slideIndex++) {
    var childSlide = _root.presentación.getChildSlide(slideIndex);
    trace(childSlide._name);
}
```

## Véase también

[Slide.numChildSlides](#)

## Slide.gotoSlide()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.gotoSlide(*newSlide*)

### Parámetros

*newSlide* Diapositiva hasta la que se desplaza.

### Valor devuelto

Valor booleano (*true* o *false*) que indica si se ha desplazado correctamente (*true*) o no (*false*).

### Descripción

Método; se desplaza hasta la diapositiva especificada por *newSlide*. Para desplazarse correctamente, deben darse las condiciones siguientes:

- La diapositiva actual debe ser secundaria de *mySlide*.
- La diapositiva especificada por *newSlide* y la diapositiva actual comparten una diapositiva anterior común, es decir, la diapositiva actual y *newSlide* deben residir en el subárbol de la misma diapositiva.

Si no se cumple una de estas condiciones, el desplazamiento no se llevará a cabo correctamente y el método devolverá *false*; de lo contrario, el método se desplazará hasta la diapositiva especificada y devolverá *true*.

Por ejemplo, observe la siguiente jerarquía de diapositivas:

```
presentación
  diapositiva1
    diapositiva1_1
    diapositiva1_2
  diapositiva2
    diapositiva2_1
    diapositiva2_2
```



Si la diapositiva actual es `diapositiva1_2`, la llamada al método `gotoSlide()` siguiente no se podrá realizar, puesto que la diapositiva actual no depende de `diapositiva2`:

```
diapositiva2.gotoSlide(diapositiva2_1);
```

Observe también la siguiente jerarquía de pantallas, en la que un objeto `Form` es la pantalla principal de dos árboles de diapositiva distintos.

```
Formulario_1
  diapositiva1
    diapositiva1_1
    diapositiva1_2
  diapositiva2
    diapositiva2_1
    diapositiva2_2
```

Si la diapositiva actual es `diapositiva1_2`, la llamada al método siguiente tampoco se podrá realizar, puesto que `diapositiva1` y `diapositiva2` se encuentran en subárboles de diapositivas diferentes.

```
diapositiva1_2.gotoSlide(diapositiva2_2);
```

### Ejemplo

El código siguiente, asociado con el componente `Button`, utiliza la propiedad `Slide.currentSlide` y el método `gotoSlide()` para enviar la presentación a la diapositiva siguiente de la presentación.

```
on(click){
  _parent.gotoSlide(_parent.currentSlide.nextSlide);
}
```

Tenga en cuenta que esto equivale al código siguiente, que utiliza el método `Slide.gotoNextSlide()`.

```
on(click){
  _parent.currentSlide.gotoNextSlide();
}
```

### Véase también

`Slide.currentSlide`, `Slide.gotoNextSlide()`

## Slide.gotoFirstSlide()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
mySlide.gotoFirstSlide()
```

### Valor devuelto

Ninguno.

## Descripción

Método; se desplaza hasta la primera diapositiva de hoja del árbol de las diapositivas secundarias que se encuentran debajo de *mySlide*. Este método se pasa por alto cuando se le llama desde un controlador de eventos `on(hide)` o `on(reveal)` de una diapositiva, en caso de que dicho evento se haya creado al desplazarse por las diapositivas.

Para ir a la primera diapositiva de una presentación, llame a `mySlide.rootSlide.gotoFirstSlide()`. Para más información sobre `rootSlide`, consulte [Slide.revealChild](#).

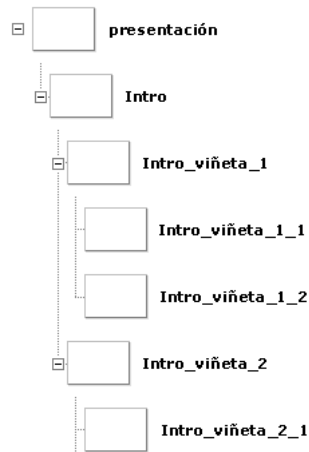
## Ejemplo

En la jerarquía de diapositivas que aparece a continuación, todas las llamadas de método desplazan hasta la diapositiva denominada `Intro_viñeta_1_1`.

```
presentación.gotoFirstSlide();  
presentación.Intro.gotoFirstSlide();  
presentación.Intro.Intro_viñeta_1.gotoFirstSlide();
```

Esta llamada de método desplaza hasta la diapositiva denominada `Intro_viñeta_2_1`.

```
presentación.Intro.Intro_viñeta_2.gotoFirstSlide();
```



## Véase también

[Slide.firstSlide](#), [Slide.revealChild](#)

## Slide.gotoLastSlide()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
mySlide.gotoLastSlide()
```

## Valor devuelto

Ninguno.

## Descripción

Método; desplaza a la última diapositiva de hoja del árbol de diapositivas secundarias debajo de *mySlide*. Este método se pasa por alto cuando se llama desde un controlador de eventos `on(hide)` o `on(reveal)` de una diapositiva, en caso de que dicho evento sea resultado de otro desplazamiento entre diapositivas.

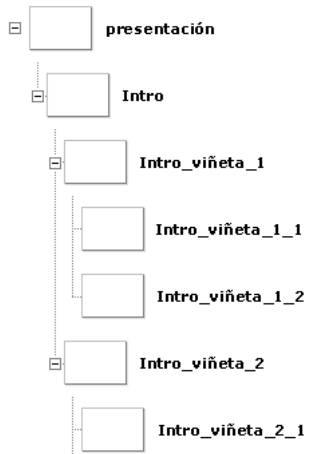
## Ejemplo

En la jerarquía de diapositivas que aparece a continuación, las llamadas de método siguientes desplazan hasta la diapositiva denominada `Intro_viñeta_1_2`.

```
presentación.Intro.gotoLastSlide();  
presentación.Intro.Intro_viñeta_1.gotoLastSlide();
```

Estas llamadas de método navegan hasta la diapositiva denominada `Intro_viñeta_2_1`.

```
presentación.gotoLastSlide();  
presentación.Intro.gotoLastSlide();
```



## Véase también

[Slide.gotoSlide\(\)](#), [Slide.lastSlide](#)

## Slide.gotoNextSlide()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
mySlide.gotoNextSlide()
```

### Valor devuelto

Valor booleano (`true` o `false`) o `null`; devuelve `true` si el método ha desplazado satisfactoriamente hasta la diapositiva siguiente; devuelve `false` si la presentación ya está en la última diapositiva cuando se invoca al método (es decir, si `currentSlide.nextSlide` es `null`); devuelve `null` si se invoca a una diapositiva que no contiene la diapositiva actual.

### Descripción

Método; desplaza hasta la diapositiva siguiente de la presentación de diapositivas. Cuando se pasa de una diapositiva a la siguiente, la diapositiva de salida se oculta y aparece la diapositiva de entrada. Si las diapositivas de salida y entrada se encuentran en subárboles diferentes, todas las diapositivas ascendientes, empezando por la diapositiva de salida y hasta la diapositiva ascendiente común de ambas diapositivas, se ocultan y se recibe un evento `hide`. Inmediatamente después, aparecen todas las diapositivas ascendientes de la diapositiva de entrada, hasta la diapositiva ascendiente común de las diapositivas de entrada y de salida, y se recibe el evento `reveal`.

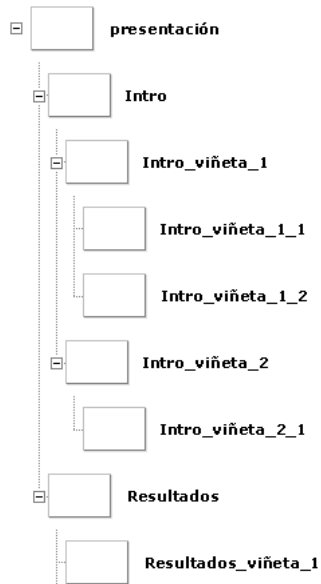
Normalmente, se llama a `gotoNextSlide()` en el nodo hoja que representa a la diapositiva actual. Si se llama a `someNode` en un nodo que no es hoja, `someNode.gotoNextSlide()` avanzará hasta el primer nodo hoja de la diapositiva o “sección” siguiente. Para más información, consulte el ejemplo que se muestra a continuación.

Este método no tiene ningún efecto cuando se invoca en una diapositiva que no contiene la diapositiva actual (véase el ejemplo siguiente).

Asimismo, el método tampoco tiene efecto cuando se llama desde un controlador de eventos `on(hide)` o `on(reveal)` asociado con una diapositiva, en caso de que se haya invocado a dicho controlador debido a los desplazamientos entre diapositivas.

## Ejemplo

Suponga que, en la jerarquía de la diapositiva siguiente, la diapositiva denominada `Intro_viñeta_1_1` es la diapositiva actual que aparece en pantalla; es decir, `_root.presentación.currentSlide._name == Intro_viñeta_1_1`.



En este caso, si se llama a `Intro_viñeta_1_1.gotoNextSlide()` accederá a `Intro_viñeta_1_2`, que es la diapositiva que se encuentra en el mismo nivel que `Intro_viñeta_1_1`.

Sin embargo, si invoca `Intro_viñeta_1.gotoNextSlide()` accederá a `Intro_viñeta_2_1`, que es la primera diapositiva hoja de `Intro_viñeta_2`, la cual, a su vez, es la siguiente diapositiva de `Intro_viñeta_1` que se encuentra al mismo nivel. Del mismo modo, si llama a `Intro.gotoNextSlide()` accederá a `Resultados_viñeta_1`, que es la primera diapositiva hoja contenida en la diapositiva `Resultados`.

Asimismo, siendo la diapositiva actual `Intro_viñeta_1_1`, si llama a `Resultados.gotoNextSlide()`, éste no tendrá ningún efecto, puesto que la diapositiva `Resultados` no contiene la diapositiva actual (es decir, `Resultados.currentSlide` es `null`).

## Véase también

[Slide.currentSlide](#), [Slide.gotoPreviousSlide\(\)](#), [Slide.nextSlide](#)

## Slide.gotoPreviousSlide()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
mySlide.gotoPreviousSlide()
```

### Valor devuelto

Valor booleano (`true` o `false`), o `null`; devuelve `true` si el método ha desplazado satisfactoriamente hasta la diapositiva anterior; devuelve `false` si la presentación se encuentra en la primera diapositiva cuando se invoca al método (es decir, si `currentSlide.nextSlide` es `null`); devuelve `null` si se invoca en una diapositiva que no contiene la diapositiva actual.

### Descripción

Método; accede a la diapositiva anterior de la presentación de diapositivas. Al pasar de una diapositiva a la anterior, la diapositiva de salida queda oculta y aparece la diapositiva de entrada. Si las diapositivas de salida y entrada se encuentran en subárboles diferentes, todas las diapositivas ascendientes, empezando por la diapositiva de salida y hasta la diapositiva ascendiente común de ambas diapositivas, se ocultan y se recibe un evento `hide`. Inmediatamente después, aparecen todas las diapositivas ascendientes de la diapositiva de entrada, hasta la diapositiva ascendiente común de las diapositivas de entrada y salida, y se recibe el evento `reveal`.

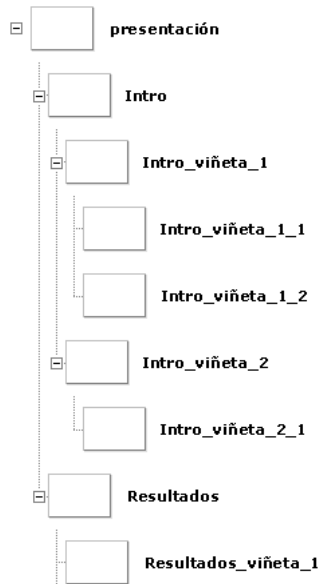
Normalmente se llama al método `gotoPreviousSlide()` en el nodo hoja que representa la diapositiva actual. Si se le llama en un nodo que no es de hoja, `someNode`, `someNode.gotoPreviousSlide()` avanzará hasta el primer nodo hoja de la siguiente diapositiva o "sección". Para más información, consulte el ejemplo que se muestra a continuación.

Este método no tiene ningún efecto cuando se invoca en una diapositiva que no contiene la diapositiva actual (véase el ejemplo siguiente).

Tenga en cuenta también que este método no tiene efecto alguno cuando se llama desde un controlador de eventos `on(hide)` o `on(reveal)` asociado con una diapositiva, en caso de que dicho controlador se haya invocado debido al desplazamiento por las diapositivas.

## Ejemplo

Suponga que, en la siguiente jerarquía de diapositivas, la diapositiva denominada Intro\_viñeta\_1\_2 es la diapositiva actual que aparece en pantalla (es decir, `_root.presentación.currentSlide._name == Intro_viñeta_1_2`).



En este caso, si llama a `Intro_viñeta_1_2.gotoPreviousSlide()` accederá a `Intro_viñeta_1_1`, que es la diapositiva anterior del mismo nivel que `Intro_viñeta_1_2`.

Sin embargo, si invoca a `Intro_viñeta_2.gotoPreviousSlide()` accederá a `Intro_viñeta_1_1`, la primera diapositiva hoja contenida en `Intro_viñeta_1`, que es la diapositiva anterior del mismo nivel que `Intro_viñeta_2`. De forma similar, si llama a `Resultados.gotoPreviousSlide()` accederá a `Intro_viñeta_1_1`, que es la primera diapositiva hoja contenida en la diapositiva `Intro`.

Asimismo, si la diapositiva actual es `Intro_viñeta_1_1`, si llama a `Resultados.gotoPreviousSlide()`, éste no tendrá ningún efecto, puesto que `Resultados` no contiene la diapositiva actual (es decir, `Resultados.currentSlide` es `null`).

## Véase también

[Slide.currentSlide](#), [Slide.gotoNextSlide\(\)](#), [Slide.previousSlide](#)

## Slide.hideChild

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(hideChild) {  
    // aquí código propio  
}
```

### Descripción

Evento; se difunde cada vez que el secundario de un objeto de diapositiva pasa de ser visible a no visible. Este evento sólo lo difunden los objetos de diapositiva, no los objetos Form. El uso principal del evento `hideChild` es aplicar transiciones a todas las diapositivas secundarias de una diapositiva determinada.

### Ejemplo

Al asociarlo con la diapositiva raíz (por ejemplo, la diapositiva presentación), este código mostrará el nombre de cada una de las diapositivas secundarias que dependan de la diapositiva raíz a medida que vayan apareciendo.

```
on(revealChild) {  
    var child = eventObj.target._name;  
    trace(child + " has just appeared");  
}
```

### Véase también

[Slide.revealChild](#)

## Slide.indexInParentSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
mySlide.indexInParent
```

### Descripción

Propiedad (sólo lectura); devuelve el índice (basado en cero) de *mySlide* en la lista de diapositivas secundarias de su diapositiva principal.



## Ejemplo

El código siguiente utiliza las propiedades `indexInParent` y `Slide.numChildSlides` para visualizar el índice de la diapositiva actual que aparece en pantalla y el número total de diapositivas que contiene la diapositiva principal. Para utilizar este código, asócielo a una diapositiva principal que contenga una o varias diapositivas secundarias.

```
on (revealChild) {  
    trace("Displaying "+(currentSlide.indexInParentSlide+1)+" of  
        "+currentSlide._parent.numChildSlides);  
}
```

Tenga en cuenta que, puesto que esta propiedad es un índice basado en cero, su valor aumenta uno (`currentSlide.indexInParent+1`) para visualizar valores más significativos.

## Véase también

[Slide.numChildSlides](#), [Slide.revealChild](#)

## Slide.lastSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.lastSlide

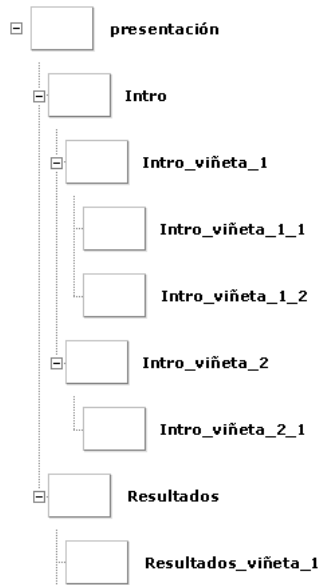
### Descripción

Propiedad (sólo lectura); devuelve la última diapositiva secundaria de *mySlide* que no tiene diapositivas secundarias.

## Ejemplo

Todas las sentencias siguientes relativas a la jerarquía de diapositivas que se muestra a continuación son true:

```
presentación.lastSlide._name == Resultados_víñeta_1;  
Intro.lastSlide._name == Intro_víñeta_1_2;  
Intro_víñeta_1.lastSlide._name == Intro_víñeta_1_2;  
Resultados.lastSlide._name == Resultados_víñeta_1;
```



## Slide.nextSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.nextSlide

### Descripción

Propiedad (sólo lectura); devuelve la diapositiva a la que se llega si se llama a *mySlide.gotoNextSlide()*; sin embargo no se accede a dicha diapositiva en realidad. Por ejemplo, puede utilizar esta propiedad para mostrar el nombre de la diapositiva siguiente de una presentación y permitir a los usuarios que seleccionen si desean navegar hasta dicha diapositiva.

## Ejemplo

En este ejemplo, la etiqueta del componente Button denominado `nextButton` muestra el nombre de la diapositiva siguiente de la presentación. Si no hay más diapositivas, es decir si `mySlide.nextSlide` es `null`, la etiqueta del botón se actualizará para indicar al usuario que se encuentra al final de la presentación de diapositivas.

```
if (mySlide.nextSlide != null) {
    nextButton.label = "Next slide: " + mySlide.nextSlide._name + " > ";
} else {
    nextButton.label = "End of this slide presentation.";
}
```

## Véase también

[Slide.gotoNextSlide\(\)](#), [Slide.previousSlide](#)

## Slide.numChildSlides

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.numChildSlides

### Descripción

Propiedad (sólo lectura); devuelve el número de diapositivas secundarias que contiene *mySlide*. Tenga en cuenta que una diapositiva puede contener formularios u otras diapositivas. Si *mySlide* contiene diapositivas y formularios, esta propiedad sólo devolverá el número de diapositivas y no contará los formularios.

## Ejemplo

En este ejemplo se utiliza `Slide.numChildSlide` y el método `Slide.getChildSlide()` para repetir todas las diapositivas secundarias de la diapositiva raíz de la presentación y los nombres de las mismas aparecen en el panel Salida.

```
var numSlides = _root.presentación.numChildSlides;
for(var slideIndex=0; slideIndex < numSlides; slideIndex++) {
    var childSlide = _root.presentación.getChildSlide(slideIndex);
    trace(childSlide._name);
}
```

## Véase también

[Slide.getChildSlide\(\)](#)

## Slide.overlayChildren

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.overlayChildren

### Descripción

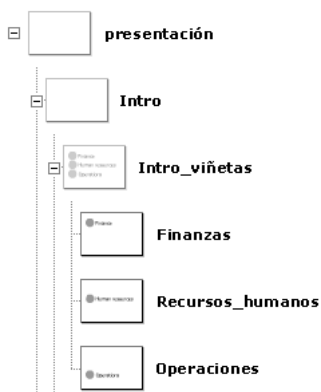
Propiedad; determina si las diapositivas secundarias de *mySlide* permanecen visibles al desplazarse de una diapositiva secundaria a la siguiente. Si está establecida en `true`, la diapositiva anterior permanecerá en pantalla cuando se pase a la diapositiva siguiente del mismo nivel; si está establecida en `false`, la diapositiva anterior quedará oculta cuando se pase a la diapositiva siguiente del mismo nivel.

Establecer esta propiedad en `true` resulta útil, por ejemplo cuando una diapositiva determinada contiene varias diapositivas de “punto de viñeta” que se muestran de forma separada (posiblemente mediante transiciones); sin embargo, todas deben estar visibles cuando aparezcan los puntos de viñeta nuevos.

**Nota:** esta propiedad se aplica sólo a los descendientes de *mySlide*, no a todas las diapositivas secundarias (anidadas).

### Ejemplo

Por ejemplo, la diapositiva Intro\_viñetas de la ilustración siguiente contiene tres diapositivas secundarias (Finanzas, Recursos\_humanos y Operaciones), cada una de las cuales muestra un punto de viñeta diferente. Al establecer `Intro_viñetas.overlayChildren` en `true`, cada diapositiva de viñeta permanecerá en el escenario a medida que vayan apareciendo el resto de puntos de viñetas.



## Slide.parentIsSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.parentIsSlide

### Descripción

Propiedad (sólo lectura); valor booleano (*true* o *false*) que indica si el objeto principal de *mySlide* también es una diapositiva. Si el objeto principal de *mySlide* es una diapositiva o una subclase de diapositiva, esta propiedad devolverá *true*; de lo contrario, devolverá *false*.

Si *mySlide* es la diapositiva raíz de una presentación, esta propiedad devolverá *false* puesto la diapositiva de la presentación depende de la línea de tiempo principal (*\_level0*), no de una diapositiva. Esta propiedad también devolverá *false* si *mySlide* depende de un formulario.

### Ejemplo

El código siguiente determina si el objeto principal de la diapositiva *mySlide* es, en sí, una diapositiva. En tal caso, se supone que *mySlide* es la diapositiva raíz o maestra de la presentación y, por lo tanto, no tiene diapositivas del mismo nivel. De lo contrario, si *mySlide.parentIsSlide* es *true*, el número de diapositivas del mismo nivel de *mySlide* aparecerá en el panel Salida.

```
if (mySlide.parentIsSlide) {  
    trace("I have " + mySlide._parent.numChildSlides+ " sibling slides");  
} else {  
    trace("I am the root slide and have no siblings");  
}
```

### Véase también

[Slide.numChildSlides](#)

## Slide.playHidden

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide*.playHidden

## Descripción

Propiedad; valor booleano que especifica si *mySlide* debe continuar reproduciéndose una vez que se haya ocultado. Si está propiedad está establecida en *true*, *mySlide* continuará reproduciéndose cuando esté oculta. Si está establecida en *false*, *mySlide* dejará de reproducirse cuando se oculte; cuando aparezca de nuevo, volverá a reproducirse en el fotograma 1 de *mySlide*.

También puede establecer esta propiedad en el inspector de propiedades del entorno de edición de Flash.

## Slide.previousSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*mySlide.previousSlide*

### Descripción

Propiedad (sólo lectura); devuelve la diapositiva a la que se llega si se llama a *mySlide.gotoPreviousSlide()*, pero realmente no se accede a dicha diapositiva. Por ejemplo, puede utilizar esta propiedad para mostrar el nombre de la diapositiva anterior de la presentación y permitir a los usuarios seleccionar si desean navegar hasta dicha diapositiva.

### Ejemplo

En este ejemplo, la etiqueta del componente Button denominado *previousButton* muestra el nombre de la diapositiva anterior de la presentación. Si no hay ninguna diapositiva anterior, es decir, si *mySlide.previousSlide* es *null*, la etiqueta del botón se actualizará para indicar que el usuario se encuentra al inicio de la presentación de las diapositivas.

```
if (mySlide.previousSlide != null) {  
    previousButton.label = "Previous slide: " + mySlide.previous._name + " >  
    ";  
} else {  
    previousButton.label = "You're at the beginning of this slide  
presentation.";
```

### Véase también

[Slide.gotoPreviousSlide\(\)](#), [Slide.nextSlide](#)

## Slide.revealChild

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
on(revealChild) {  
    // aquí código propio  
}
```

### Descripción

Evento; se ha difundido cada vez que una diapositiva secundaria de un objeto de diapositiva pasa de estar oculta a visible. Este evento se utiliza principalmente para asociar transiciones de entrada con todas las diapositivas secundarias de una diapositiva determinada.

### Ejemplo

Si se asocia con la diapositiva raíz, por ejemplo, la diapositiva presentación, este código mostrará el nombre de todas las diapositivas secundarias a medida que vayan apareciendo.

```
on(revealChild) {  
    var child = eventObj.target._name;  
    trace(child + " has just appeared");  
}
```

### Véase también

[Slide.hideChild](#)

## Slide.rootSlide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
mySlide.rootSlide
```

### Descripción

Propiedad (sólo lectura); devuelve la diapositiva raíz del árbol o subárbol de diapositivas que contiene *mySlide*.

### Ejemplo

Suponga que tiene un clip de película en una diapositiva y que al hacer clic en el mismo se accede a la primera diapositiva de la presentación. Para que esto suceda, debe asociar el código siguiente con el clip de película:

```
on(press){  
    _parent.rootSlide.gotoFirstSlide();  
}
```

En este caso, *\_parent* se refiere a la diapositiva que contiene el objeto del clip de película.

# Clase StyleManager

**Nombre de clase de ActionScript** `mx.styles.StyleManager`

La clase StyleManager lleva el seguimiento de los colores y estilos que se heredan. Sólo necesitará utilizar esta clase si ha creado componentes y desea añadir un color o un estilo heredados.

Para determinar qué estilos se heredan, consulte el [sitio Web W3C](#).

## Resumen de métodos de la clase StyleManager

| Método                                              | Descripción                                         |
|-----------------------------------------------------|-----------------------------------------------------|
| <code>StyleManager.registerColorName()</code>       | Registra un nombre de color nuevo con StyleManager. |
| <code>StyleManager.registerColorStyle()</code>      | Registra un estilo de color nuevo con StyleManager. |
| <code>StyleManager.registerInheritingStyle()</code> | Registra un estilo heredado nuevo con StyleManager. |

## StyleManager.registerColorName()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

`StyleManager.registerColorName(colorName, value)`

### Parámetros

*colorName* Cadena que indica el nombre del color (por ejemplo, "gray", "darkGrey", y así sucesivamente).

*value* Número hexadecimal que indica el color (por ejemplo, 0x808080, 0x404040, y así sucesivamente).

### Valor devuelto

Ninguno.

### Descripción

Método; asocia un nombre de color con un valor hexadecimal y lo registra con StyleManager.

### Ejemplo

En el ejemplo siguiente se registra "gray" como el nombre de color representado por el valor hexadecimal 0x808080:

```
StyleManager.registerColorName("gray", 0x808080 );
```



## StyleManager.registerColorStyle()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
StyleManager.registerColorStyle(colorStyle)
```

### Parámetros

*colorStyle* Cadena que indica el nombre del color (por ejemplo, "highlightColor", "shadowColor", "disabledColor", y así sucesivamente).

### Valor devuelto

Ninguno.

### Descripción

Método; añade un estilo de color nuevo a StyleManager.

### Ejemplo

En el ejemplo siguiente se registra "highlightColor" como un estilo de color:

```
StyleManager.registerColorStyle("highlightColor");
```

## StyleManager.registerInheritingStyle()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
StyleManager.registerInheritingStyle(propertyName)
```

### Parámetros

*propertyName* Cadena que indica el nombre de la propiedad de estilo (por ejemplo, "newProp1", "newProp2", y así sucesivamente).

### Valor devuelto

Ninguno.

### Descripción

Método; marca la propiedad de estilo como heredada. Utilice este método para registrar las propiedades de estilo que no aparecen en la lista de la especificación CSS. No utilice este método para cambiar propiedades de estilo no heredadas a heredadas.

## Ejemplo

En el ejemplo siguiente se registra `newProp1` como un estilo heredado:

```
StyleManager.registerInheritingStyle("newProp1");
```

## Componente TextArea

El componente `TextArea` ajusta el objeto `TextField` nativo de `ActionScript`. Puede utilizar estilos para personalizar el componente `TextArea`; cuando se desactiva una instancia, su contenido se muestra en un color representado por el estilo `disabledColor`. El componente `TextArea` también se puede formatear con HTML o como un campo de contraseña que disfraza el texto.

Es posible activar o desactivar el componente `TextArea` en una aplicación. Si está desactivado, no recibe la entrada del ratón ni del teclado. Cuando está activado, sigue las mismas reglas de selección y navegación que un objeto `TextField` de `ActionScript`. Cuando la instancia de `TextArea` esté seleccionada, podrá controlarla con las teclas siguientes:

| Tecla                  | Descripción                                                                        |
|------------------------|------------------------------------------------------------------------------------|
| Teclas de flecha       | Desplaza el punto de inserción una línea hacia arriba, abajo, izquierda o derecha. |
| Av Pág                 | Avanza el punto de inserción una pantalla.                                         |
| Re Pág                 | Retrocede el punto de inserción una pantalla.                                      |
| Mayúsculas + Tabulador | Desplaza la selección al objeto anterior.                                          |
| Tabulador              | Desplaza la selección al objeto siguiente.                                         |

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase FocusManager” en la página 292](#).

La vista previa dinámica de cada instancia de `TextArea` refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes. Si se necesita una barra de desplazamiento, ésta aparece en la vista previa dinámica, aunque no funciona. No es posible seleccionar texto en la vista previa, de igual modo que tampoco se puede introducir texto en la instancia del componente en el escenario.

Cuando se añade el componente `TextArea` a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo.

## Utilización del componente TextArea

Puede utilizar un componente `TextArea` siempre que necesite un campo de texto de varias líneas. Si necesita un campo de texto con una sola línea, utilice [“Componente TextInput” en la página 554](#). Por ejemplo, suponga que precisa un componente `TextArea` como campo de comentarios de un formulario. En tal caso, puede definir un detector que compruebe si el campo está vacío cuando un usuario utiliza el tabulador para salir del campo. El detector podría mostrar un mensaje de error indicando que debe introducirse un comentario en el campo.

## Parámetros del componente TextArea

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente TextArea en el inspector de propiedades o en el panel Inspector de componentes:

**text** indica el contenido de TextArea. No es posible introducir retornos de carro en el inspector de propiedades ni en el panel Inspector de componentes. El valor predeterminado es "" (cadena vacía).

**html** indica si el texto está formateado con HTML (true) o no (false). El valor predeterminado es false.

**editable** indica si el componente TextArea es editable (true) o no (false). El valor predeterminado es true.

**wordWrap** indica si el texto se ajusta (true) o no (false). El valor predeterminado es true.

Puede escribir código ActionScript para controlar éstas y otras opciones adicionales para los componentes de TextArea utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase TextArea](#).

## Creación de aplicaciones con el componente TextArea

El procedimiento siguiente explica cómo añadir un componente TextArea a una aplicación durante la edición. En este ejemplo, el componente es un campo Comment con un detector de eventos que determina si un usuario ha introducido texto.

**Para crear una aplicación con el componente TextArea, siga este procedimiento:**

- 1 Arrastre un componente TextArea desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de la instancia **comment**.
- 3 En el inspector de propiedades, defina los parámetros deseados. No obstante, deberá dejar el parámetro text vacío, el parámetro editable establecido en true y el parámetro password en false.
- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
textListener = new Object();
textListener.handleEvent = function (evt){
    if (comment.length < 1) {
        Alert(_root, "Error", "Debe introducir al menos un comentario en este
        campo", mxModal | mxOK);
    }
}
comment.addEventListener("focusOut", textListener);
```

Este código establece un controlador de eventos focusOut en la instancia comment de TextArea para comprobar que el usuario ha introducido algo en el campo de texto.

- 5 Una vez introducido el texto en la instancia, puede obtener su valor como se muestra a continuación:

```
var login = comment.text;
```

## Personalización del componente TextArea

El componente TextArea puede transformarse horizontal o verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()` o cualquiera de las propiedades y métodos aplicables de [Clase TextArea](#).

Cuando se cambia el tamaño de un componente TextArea, el borde se ajusta a los límites del cuadro nuevo. Si es necesario, las barras de desplazamiento se sitúan en los bordes inferior y derecho. A continuación, se ajusta el tamaño del campo de texto al interior del área restante; el componente TextArea carece de elementos con tamaño fijo. Si el componente TextArea es demasiado pequeño para mostrar el texto, éste se recorta.

## Utilización de estilos con el componente TextArea

Si bien el componente TextArea admite un conjunto de estilos de componente para todo el texto del campo, sin embargo, también puede mostrar código HTML compatible con la representación HTML de Flash Player. Para mostrar texto HTML, defina `TextArea.html` en `true`.

El componente TextArea tiene sus propiedades de estilo `backgroundColor` y `borderStyle` definidas en una declaración de estilo de clase. Los estilos de clase anulan los estilos `_global`; por tanto, si desea definir las propiedades de estilo `backgroundColor` y `borderStyle`, debe crear una declaración distinta de estilo predeterminado en la instancia.

Si el nombre de una propiedad de estilo termina por “Color”, significa que es una propiedad de estilo de color y se comporta de forma diferente a las que no lo son. Para más información, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

Un componente TextArea admite los siguientes estilos:

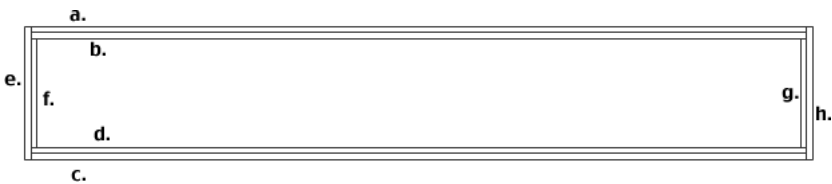
| Estilo                      | Descripción                                                 |
|-----------------------------|-------------------------------------------------------------|
| <code>color</code>          | Color predeterminado del texto.                             |
| <code>embedFonts</code>     | Fuentes que incorporar al documento.                        |
| <code>fontFamily</code>     | Nombre de la fuente del texto.                              |
| <code>fontSize</code>       | Tamaño de la fuente en puntos.                              |
| <code>fontStyle</code>      | Estilo de la fuente; puede ser “normal” o “italic”.         |
| <code>fontWeight</code>     | Grosor de la fuente; puede ser “normal” o “bold”.           |
| <code>textAlign</code>      | Alineación del texto; puede ser “left”, “right” o “center”. |
| <code>textDecoration</code> | Decoración del texto; puede ser “none” o “underline”.       |

## Utilización de aspectos con el componente TextArea

El componente TextArea utiliza la clase RectBorder para dibujar su borde. Puede utilizar el método `setStyle()` (véase [UIObject.setStyle\(\)](#)) para cambiar las propiedades de estilo RectBorder siguientes:

| Estilos RectBorder          | Letra |
|-----------------------------|-------|
| <code>borderColor</code>    | a     |
| <code>highlightColor</code> | b     |
| <code>borderColor</code>    | c     |
| <code>shadowColor</code>    | d     |
| <code>borderCapColor</code> | e     |
| <code>shadowCapColor</code> | f     |
| <code>shadowCapColor</code> | g     |
| <code>borderCapColor</code> | h     |

Las propiedades de estilo definen las siguientes posiciones en el borde:



## Clase TextArea

**Herencia** UIObject > UIComponent > View > ScrollView > TextArea

**Nombre de clase de ActionScript** mx.controls.TextArea

Las propiedades de la clase TextArea permiten definir el contenido y el formato del texto, así como las posiciones vertical y horizontal en tiempo de ejecución. También puede indicar si el campo es editable y si se trata de un campo “contraseña”, o restringir los caracteres que el usuario puede introducir.

Si una propiedad de la clase TextArea se define con ActionScript, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

El componente TextArea anula el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas.

El componente TextArea admite estilos CSS y cualquier otro estilo adicional de HTML admitido por Flash Player.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.TextArea.version);
```

**Nota:** el código siguiente devuelve `undefined`: `trace(myTextAreaInstance.version);`.

## Resumen de propiedades de la clase `TextArea`

| Propiedad                           | Descripción                                                                                                                                                                        |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>TextArea.editable</code>      | Valor booleano que indica si el campo es editable ( <code>true</code> ) o no ( <code>false</code> ).                                                                               |
| <code>TextArea.hPosition</code>     | Define la posición horizontal del texto incluido en el panel de desplazamiento.                                                                                                    |
| <code>TextArea.hScrollPolicy</code> | Indica si la barra de desplazamiento horizontal está activada siempre ( <code>"on"</code> ), nunca ( <code>"off"</code> ) o se activa cuando es necesario ( <code>"auto"</code> ). |
| <code>TextArea.html</code>          | Etiqueta que indica si el campo de texto admite formato HTML.                                                                                                                      |
| <code>TextArea.length</code>        | Número de caracteres de un campo de texto. Es una propiedad de sólo lectura.                                                                                                       |
| <code>TextArea.maxChars</code>      | Número máximo de caracteres que puede contener el campo de texto.                                                                                                                  |
| <code>TextArea.maxHPosition</code>  | Valor máximo de <code>TextArea.hPosition</code> .                                                                                                                                  |
| <code>TextArea.maxVPosition</code>  | Valor máximo de <code>TextArea.vPosition</code> .                                                                                                                                  |
| <code>TextArea.password</code>      | Valor booleano que indica si el campo es un campo de contraseña ( <code>true</code> ) o no ( <code>false</code> ).                                                                 |
| <code>TextArea.restrict</code>      | Conjunto de caracteres que puede introducir un usuario en el campo de texto.                                                                                                       |
| <code>TextArea.text</code>          | Contenido de texto de un componente <code>TextArea</code> .                                                                                                                        |
| <code>TextArea.vPosition</code>     | Número que indica la posición de desplazamiento vertical.                                                                                                                          |
| <code>TextArea.vScrollPolicy</code> | Indica si la barra de desplazamiento vertical está activada siempre ( <code>"on"</code> ), nunca ( <code>"off"</code> ) o se activa cuando es necesario ( <code>"auto"</code> ).   |
| <code>TextArea.wordWrap</code>      | Valor booleano que indica si el texto se ajusta ( <code>true</code> ) o no ( <code>false</code> ).                                                                                 |

## Resumen de eventos de la clase `TextArea`

| Evento                       | Descripción                                         |
|------------------------------|-----------------------------------------------------|
| <code>TextArea.change</code> | Notifica a los detectores que el texto ha cambiado. |

# TextArea.change

## Disponibilidad

Flash Player 6 versión 79.

## Edición

Flash MX 2004.

## Sintaxis

Sintaxis 1:

```
on(change){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    ...  
}  
textAreaInstance.addEventListener("change", listenerObject)
```

## Descripción

Evento; notifica a los detectores que el texto ha cambiado. Este evento se difunde una vez modificado el texto. No se puede utilizar para impedir que ciertos caracteres se añadan al campo de texto del componente; para ello, deberá utilizar [TextArea.restrict](#).

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `TextArea`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myTextArea`, envía “\_level0.myTextArea” al panel Salida.

```
on(change){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*textAreaInstance*) distribuye un evento (en este caso, *change*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 602](#).

## Ejemplo

En este ejemplo se señala el número total de veces que ha cambiado el campo de texto:

```
myTextArea.changeHandler = function(obj) {  
    this.changeCount++;  
    trace(obj.target);  
    trace("text has changed " + this.changeCount + " times now! it now contains  
    " +  
    this.text);  
}
```

## Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## TextArea.editable

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance*.editable

### Descripción

Propiedad; valor booleano que indica si el componente es editable (*true*) o no (*false*). El valor predeterminado es *true*.

## TextArea.hPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance*.hPosition

### Descripción

Propiedad; define la posición horizontal del texto dentro del campo. El valor predeterminado es 0.

## Ejemplo

El código siguiente muestra los caracteres situados en el extremo izquierdo del campo:

```
myTextArea.hPosition = 0;
```



## TextArea.hScrollPolicy

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.hScrollPolicy*

### Descripción

Propiedad; determina si la barra de desplazamiento horizontal está presente siempre ("on"), nunca ("off") o aparece automáticamente en función del tamaño del campo ("auto"). El valor predeterminado es "auto".

### Ejemplo

El código siguiente mantiene siempre activas las barras de desplazamiento horizontal:

```
text.hScrollPolicy = "on";
```

## TextArea.html

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.html*

### Descripción

Propiedad; valor booleano que indica si el campo de texto tiene formato HTML (*true*) o no (*false*). Si el valor de la propiedad *html* es *true*, el campo de texto es un campo de texto HTML. Si el valor de *html* es *false*, el campo de texto no es un campo de texto HTML. El valor predeterminado es *false*.

### Ejemplo

En el ejemplo siguiente se convierte el campo *myTextArea* en un campo de texto HTML y luego se formatea el texto con etiquetas HTML:

```
myTextArea.html = true;  
myTextArea.text = "The <b>Royal</b> Nonesuch"; // muestra "The Royal Nonesuch"
```

## TextArea.length

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.length*

### Descripción

Propiedad (sólo lectura); indica el número de caracteres de un campo de texto. Esta propiedad devuelve el mismo valor que `text.length` de ActionScript, pero es más rápida. El carácter de tabulador (“\t”) cuenta como un carácter. El valor predeterminado es 0.

### Ejemplo

En el ejemplo siguiente se obtiene la longitud del campo de texto y se copia en la variable `length`:

```
var length = myTextArea.length; // hallar la longitud de la cadena de texto
```

## TextArea.maxChars

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.maxChars*

### Descripción

Propiedad; indica el número máximo de caracteres que puede contener el campo de texto. Un script puede insertar más texto del que permite `maxChars`; la propiedad `maxChars` sólo indica cuánto texto puede introducir el usuario. Si el valor de esta propiedad es null, no hay límite en cuanto a la cantidad de texto que puede introducirse. El valor predeterminado es null.

### Ejemplo

En el ejemplo siguiente, se limita a 255 el número de caracteres que puede introducir el usuario:

```
myTextArea.maxChars = 255;
```

## TextArea.maxHPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.maxHPosition*

### Descripción

Propiedad (sólo lectura); valor máximo de [TextArea.hPosition](#). El valor predeterminado es 0.

### Ejemplo

El código siguiente desplaza el texto hasta el extremo derecho:

```
myTextArea.hPosition = myTextArea.maxHPosition;
```

### Véase también

[TextArea.vPosition](#)

## TextArea.maxVPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.maxVPosition*

### Descripción

Propiedad (sólo lectura); indica el valor máximo de [TextArea.vPosition](#). El valor predeterminado es 0.

### Ejemplo

El código siguiente desplaza el texto hasta la parte inferior del componente:

```
myTextArea.vPosition = myTextArea.maxVPosition;
```

### Véase también

[TextArea.hPosition](#)

## TextArea.password

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.password*

### Descripción

Propiedad; valor booleano que indica si el campo de texto es un campo de contraseña (`true`) o no (`false`). Si el valor de `password` es `true`, el campo de texto es un campo de contraseña y oculta los caracteres de entrada. Si su valor es `false`, el campo de texto no es de contraseña. El valor predeterminado es `false`.

### Ejemplo

El código siguiente convierte el campo de texto en un campo de contraseña y muestra todos los caracteres como asteriscos (\*):

```
myTextArea.password = true;
```

## TextArea.restrict

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.restrict*

### Descripción

Propiedad; indica el conjunto de caracteres que los usuarios pueden introducir en el campo de texto. El valor predeterminado es `undefined`. Si el valor de la propiedad `restrict` es `null`, el usuario puede introducir cualquier carácter. Si el valor de la propiedad `restrict` es una cadena vacía, no se puede introducir ningún carácter. Si el valor de la propiedad `restrict` es una cadena de caracteres, sólo podrán introducirse en el campo de texto; el texto se explora de izquierda a derecha. Puede especificarse un rango utilizando un guión (-).

La propiedad `restrict` sólo limita la interacción del usuario; un script puede introducir cualquier texto en el campo de texto. Esta propiedad no se sincroniza con las casillas de verificación de Incorporar contornos de fuentes del inspector de propiedades.

Si la cadena empieza por “^”, inicialmente se aceptan todos los caracteres; los caracteres posteriores de la cadena se excluyen del conjunto de caracteres aceptados. Si la cadena no empieza por “^”, inicialmente no se acepta ningún carácter; los caracteres posteriores de la cadena se incluyen en el conjunto de caracteres aceptados.

### Ejemplo

En el ejemplo siguiente, la primera línea del código limita el campo de texto a letras en mayúsculas, números y espacios. La segunda línea del código admite todos los caracteres excepto letras en minúsculas.

```
my_txt.restrict = "A-Z 0-9"; // limitar control a letras en mayúsculas, números  
y espacios  
my_txt.restrict = "^a-z"; // permitir todos los caracteres, excepto letras en  
minúsculas
```

## TextArea.text

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.text*

### Descripción

Propiedad; contenido de texto de un componente TextArea. El valor predeterminado es "" (cadena vacía).

### Ejemplo

El código siguiente coloca una cadena en la instancia myTextArea y luego traza dicha cadena en el panel Salida:

```
myTextArea.text = "The Royal Nonesuch";  
trace(myTextArea.text); // traza "The Royal Nonesuch"
```

## TextArea.vPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.vPosition*

### Descripción

Propiedad; define la posición vertical del texto de un campo de texto. La propiedad scroll es útil para dirigir a los usuarios a un párrafo específico en un pasaje largo, o para crear campos de texto con desplazamiento. Es posible obtener y definir esta propiedad. El valor predeterminado es 0.

### Ejemplo

El código siguiente muestra los caracteres situados en el extremo superior del campo:

```
myTextArea.vPosition = 0;
```

## TextArea.vScrollPolicy

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.vScrollPolicy*

### Descripción

Propiedad; determina si la barra de desplazamiento vertical está presente siempre ("on"), nunca ("off") o aparece automáticamente en función del tamaño del campo ("auto"). El valor predeterminado es "auto".

### Ejemplo

El código siguiente mantiene siempre activas las barras de desplazamiento vertical:

```
text.vScrollPolicy = "off";
```

## TextArea.wordWrap

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textAreaInstance.wordWrap*

### Descripción

Propiedad; valor booleano que indica si el texto se ajusta (true) o no (false). El valor predeterminado es true.

## Componente TextInput

TextInput es un componente de una sola línea que ajusta el objeto TextField nativo de ActionScript. Puede utilizar estilos para personalizar el componente TextInput; cuando se desactiva una instancia, su contenido se muestra en un color representado por el estilo "disabledColor". El componente TextInput se puede formatear también con HTML o como un campo de contraseña que disfraza el texto.

Es posible activar o desactivar el componente `TextInput` en una aplicación. Si está desactivado, no recibe la entrada del ratón ni del teclado. Cuando está activado, sigue las mismas reglas de selección y navegación que un objeto `TextField` de `ActionScript`. Cuando la instancia de `TextInput` está seleccionada, puede controlarla con las teclas siguientes:

| Tecla                  | Descripción                                                       |
|------------------------|-------------------------------------------------------------------|
| Teclas de flecha       | Desplaza el carácter un carácter hacia la izquierda o la derecha. |
| Mayúsculas + Tabulador | Desplaza la selección al objeto anterior.                         |
| Tabulador              | Desplaza la selección al objeto siguiente.                        |

Para más información sobre el control de la selección, consulte [“Creación de un desplazamiento personalizado de la selección” en la página 27](#) o [“Clase `FocusManager`” en la página 292](#).

La vista previa dinámica de cada instancia de `TextInput` refleja los cambios de parámetros realizados durante la edición en el inspector de propiedades o en el panel Inspector de componentes. No es posible seleccionar texto en la vista previa, de igual modo que tampoco se puede introducir texto en la instancia del componente en el escenario.

Cuando se añade el componente `TextInput` a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo.

## Utilización del componente `TextInput`

Puede utilizar un componente `TextInput` siempre que necesite un campo de texto de una sola línea. Si necesita un campo de texto con varias líneas, utilice [“Componente `TextArea`” en la página 542](#). Por ejemplo, si desea utilizar un componente `TextInput` como campo de contraseña en un formulario, puede definir un detector que compruebe si el campo tiene caracteres suficientes cuando el usuario utiliza el tabulador para salir del campo. El detector podría mostrar un mensaje de error indicando que se debe introducir el número de caracteres correcto.

## Parámetros del componente `TextInput`

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente `TextInput` en el inspector de propiedades o en el panel Inspector de componentes:

**text** especifica el contenido de `TextInput`. No es posible introducir retornos de carro en el inspector de propiedades ni en el panel Inspector de componentes. El valor predeterminado es "" (cadena vacía).

**editable** indica si el componente `TextInput` es editable (`true`) o no (`false`). El valor predeterminado es `true`.

**password** indica si el campo es un campo de contraseña (`true`) o no (`false`). El valor predeterminado es `false`.

Puede escribir código `ActionScript` para controlar éstas y otras opciones adicionales para los componentes de `TextInput` utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase `TextInput`](#).

## Creación de aplicaciones con el componente TextInput

El procedimiento siguiente explica cómo añadir un componente `TextInput` a una aplicación durante la edición. En este ejemplo, el componente es un campo de contraseña con un detector de eventos que determina si se ha introducido el número de caracteres correcto.

**Para crear una aplicación con el componente `TextInput`, siga este procedimiento:**

- 1 Arrastre un componente `TextInput` desde el panel Componentes al escenario.
- 2 En el inspector de propiedades, introduzca el nombre de instancia `passwordField`.
- 3 En el inspector de propiedades, siga este procedimiento:

- Deje el parámetro `text` vacío.
- Defina el parámetro `editable` en `true`.
- Defina el parámetro `password` en `true`.

- 4 Seleccione el fotograma 1 de la línea de tiempo, abra el panel Acciones e introduzca el código siguiente:

```
textInputer = new Object();
textInputer.handleEvent = function (evt){
    if (evt.type == "enter"){
        trace("You must enter at least 8 characters");
    }
}
passwordField.addEventListener("enter", textListener);
```

Este código define un controlador de eventos `enter` en la instancia `passwordField` de `TextInput` para comprobar que el usuario ha introducido el número de caracteres correcto.

- 5 Una vez introducido el texto en la instancia `passwordField`, puede obtener su valor como se muestra a continuación:

```
var login = passwordField.text;
```

## Personalización del componente TextInput

El componente `TextInput` puede transformarse horizontalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()` o cualquiera de las propiedades y métodos aplicables de la [Clase TextInput](#).

Cuando se cambia el tamaño de un componente `TextInput`, el borde se ajusta al nuevo recuadro de delimitación. El componente `TextInput` no utiliza barras de desplazamiento, si bien el punto de inserción se desplaza automáticamente a medida que el usuario interactúa con el texto. Así, el tamaño del campo de texto se ajusta al interior del área restante; el componente `TextInput` carece de elementos con tamaño fijo. Si el componente `TextInput` es demasiado pequeño para mostrar el texto, éste se recorta.

## Utilización de estilos con el componente TextInput

El componente `TextInput` tiene sus propiedades de estilo `backgroundColor` y `borderStyle` definidas en una declaración de estilo de clase. Los estilos de clase anulan los estilos `_global`, por tanto, si desea definir las propiedades de estilo `backgroundColor` y `borderStyle`, debe crear una declaración distinta de estilo predeterminado en la instancia.



Un componente `TextInput` admite los siguientes estilos:

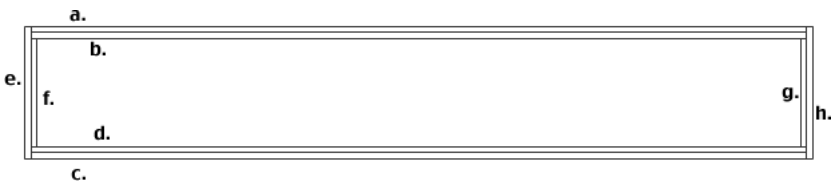
| Estilo                      | Descripción                                                 |
|-----------------------------|-------------------------------------------------------------|
| <code>color</code>          | Color predeterminado del texto.                             |
| <code>embedFonts</code>     | Fuentes que incorporar al documento.                        |
| <code>fontFamily</code>     | Nombre de la fuente del texto.                              |
| <code>fontSize</code>       | Tamaño de la fuente en puntos.                              |
| <code>fontStyle</code>      | Estilo de la fuente; puede ser “normal” o “italic”.         |
| <code>fontWeight</code>     | Grosor de la fuente; puede ser “normal” o “bold”.           |
| <code>textAlign</code>      | Alineación del texto; puede ser “left”, “right” o “center”. |
| <code>textDecoration</code> | Decoración del texto; puede ser “none” o “underline”.       |

**Utilización de aspectos con el componente `TextInput`**

El componente `TextInput` utiliza la clase `RectBorder` para dibujar su borde. Puede utilizar el método `setStyle()` (véase [UIObject.setStyle\(\)](#)) para cambiar las propiedades de estilo `RectBorder` siguientes:

| estilos <code>RectBorder</code> | Letra |
|---------------------------------|-------|
| <code>borderColor</code>        | a     |
| <code>highlightColor</code>     | b     |
| <code>borderColor</code>        | c     |
| <code>shadowColor</code>        | d     |
| <code>borderCapColor</code>     | e     |
| <code>shadowCapColor</code>     | f     |
| <code>shadowCapColor</code>     | g     |
| <code>borderCapColor</code>     | h     |

Las propiedades de estilo definen las siguientes posiciones en el borde:



**Clase `TextInput`**

**Herencia**    `UIObject > UIComponent > TextInput`  
**Nombre de clase de `ActionScript`**    `mx.controls.TextInput`

Las propiedades de la clase `TextInput` permiten definir el contenido, el formato y la posición horizontal del texto en tiempo de ejecución. También puede indicar si el campo es editable y si se trata de un campo “contraseña”, o restringir los caracteres que el usuario puede introducir.

Si una propiedad de la clase `TextInput` se define con `ActionScript`, sustituye al parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

El componente `TextInput` utiliza `FocusManager` para anular el rectángulo de selección predeterminado de Flash Player y dibuja uno personalizado con esquinas redondeadas. Para más información, consulte “Clase `FocusManager`” en la página 292.

El componente `TextInput` admite estilos CSS y cualquier otro estilo adicional de HTML admitido por Flash Player. Para información sobre la compatibilidad con CSS, consulte la especificación W3C.

Puede manipular la cadena de texto utilizando la cadena devuelta por el objeto de texto.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.controls.TextInput.version);
```

**Nota:** el código siguiente devuelve `undefined`: `trace(myTextInputInstance.version);`.

## Resumen de métodos de la clase `TextInput`

Hereda todos los métodos de `UIObject` y `UIComponent`.

## Resumen de propiedades de la clase `TextInput`

| Propiedad                           | Descripción                                                                                                                 |
|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>TextInput.editable</code>     | Valor booleano que indica si el campo es editable ( <code>true</code> ) o no ( <code>false</code> ).                        |
| <code>TextInput.hPosition</code>    | Posición del desplazamiento horizontal del campo de texto.                                                                  |
| <code>TextInput.length</code>       | Número de caracteres de un campo de texto de <code>TextInput</code> . Esta propiedad es de sólo lectura.                    |
| <code>TextInput.maxChars</code>     | Número máximo de caracteres que el usuario puede introducir en un campo de texto de <code>TextInput</code> .                |
| <code>TextInput.maxHPosition</code> | Valor máximo posible para <code>TextField.hPosition</code> . Esta propiedad es de sólo lectura.                             |
| <code>TextInput.password</code>     | Valor booleano que indica si el campo de texto de entrada es un campo de contraseña que oculta los caracteres introducidos. |
| <code>TextInput.restrict</code>     | Indica los caracteres que el usuario puede introducir en un campo de texto.                                                 |
| <code>TextInput.text</code>         | Define el contenido de texto de un campo de texto de <code>TextInput</code> .                                               |

Hereda todos los métodos de `UIObject` y `UIComponent`.

## Resumen de eventos de la clase TextInput

| Evento                           | Descripción                                   |
|----------------------------------|-----------------------------------------------|
| <a href="#">TextInput.change</a> | Se difunde cuando se modifica el campo Input. |
| <a href="#">TextInput.enter</a>  | Se difunde cuando se presiona la tecla Intro. |

Hereda todos los métodos de [UIObject](#) y [UIComponent](#).

### TextInput.change

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.

#### Sintaxis

Sintaxis 1:

```
on(change){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.change = function(eventObject){  
    ...  
}  
textInputInstance.addEventListener("change", listenerObject)
```

#### Descripción

Evento; notifica a los detectores que el texto ha cambiado. Este evento se difunde una vez modificado el texto. No se puede utilizar para impedir que ciertos caracteres se añadan al campo de texto del componente; para ello, deberá utilizar [TextInput.restrict](#). Este evento se activa sólo por una entrada del usuario, no por un cambio provocado por `ActionScript`.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `TextInput`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myTextInput`, envía “\_level0.myTextInput” al panel Salida:

```
on(change){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*textInputInstance*) distribuye un evento (en este caso, *change*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método *UIEventDispatcher.addEventListener()* de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en [la página 602](#).

### Ejemplo

En este ejemplo se define una etiqueta en la aplicación que indica si el contenido del campo *TextInput* ha sufrido cambios:

```
form.change = function(eventObj){
    // eventObj.target es el componente que ha generado el evento change,
    // por ejemplo, el componente Input.
    myFormChanged.visible = true; // definir un indicador de cambios si se
    cambia el contenido;
}
myInput.addEventListener("change", form);
```

### Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## TextInput.editable

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textInputInstance*.editable

### Descripción

Propiedad; valor booleano que indica si el componente es editable (*true*) o no (*false*). El valor predeterminado es *true*.

## TextInput.enter

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(enter){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.enter = function(eventObject){  
    ...  
}  
textInputInstance.addEventListener("enter", listenerObject)
```

### Descripción

Evento; notifica al detector que se ha presionado la tecla Intro.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `TextInput`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myTextInput`, envía “\_level0.myTextInput” al panel Salida:

```
on(enter){  
    trace(this);  
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*textInputInstance*) distribuye un evento (en este caso, `enter`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. Cada objeto `Event` tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En este ejemplo se define una etiqueta en la aplicación que indica si el contenido del campo `TextInput` ha sufrido cambios:

```
form.enter = function(eventObj){
    // eventObj.target es el componente que genera el evento enter,
    // por ejemplo, el componente Input.
    myFormChanged.visible = true;
    // definir un indicador de cambios si el usuario presiona la tecla Intro;
}
myInput.addEventListener("enter", form);
```

## Véase también

`UIEventDispatcher.addEventListener()`

## TextInput.hPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textInputInstance.hPosition*

### Descripción

Propiedad; define la posición horizontal del texto dentro del campo. El valor predeterminado es 0.

### Ejemplo

El código siguiente muestra los caracteres situados en el extremo izquierdo del campo:

```
myTextInput.hPosition = 0;
```

## TextInput.length

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*inputInstance.length*

### Descripción

Propiedad (sólo lectura); valor que indica el número de caracteres de un componente `TextInput`. El carácter de tabulador (“\t”) cuenta como un carácter. El valor predeterminado es 0.

### Ejemplo

El código siguiente determina el número de caracteres de la cadena `myTextInput` y lo copia en la variable `length`:

```
var length = myTextInput.length;
```

## TextInput.maxChars

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textInputInstance*.maxChars

### Descripción

Propiedad; indica el número máximo de caracteres que puede contener el campo de texto. Un script puede insertar más texto del que permite `maxChars`; la propiedad `maxChars` sólo indica cuánto texto puede introducir el usuario. Si el valor de esta propiedad es `null`, no hay límite en cuanto a la cantidad de texto que puede introducirse. El valor predeterminado es `null`.

### Ejemplo

En el ejemplo siguiente, se limita a 255 el número de caracteres que puede introducir el usuario:

```
myTextInput.maxChars = 255;
```

## TextInput.maxHPosition

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textInputInstance*.maxHPosition

### Descripción

Propiedad (sólo lectura); indica el valor máximo de `TextInput.hPosition`. El valor predeterminado es 0.

### Ejemplo

El código siguiente se desplaza hasta el extremo derecho:

```
myTextInput.hPosition = myTextInput.maxHPosition;
```

## TextInput.password

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textInputInstance.password*

### Descripción

Propiedad; valor booleano que indica si el campo de texto es un campo de contraseña (`true`) o no (`false`). Si el valor de `password` es `true`, el campo de texto es un campo de contraseña y oculta los caracteres de entrada. Si su valor es `false`, el campo de texto no es de contraseña. El valor predeterminado es `false`.

### Ejemplo

El código siguiente convierte el campo de texto en un campo de contraseña y muestra todos los caracteres como asteriscos (\*):

```
myTextInput.password = true;
```

## TextInput.restrict

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textInputInstance.restrict*

### Descripción

Propiedad; indica el conjunto de caracteres que los usuarios pueden introducir en el campo de texto. El valor predeterminado es `undefined`. Si el valor de la propiedad `restrict` es `null` o una cadena vacía (""), el usuario puede introducir cualquier carácter. Si el valor de la propiedad `restrict` es una cadena de caracteres, sólo podrán introducirse en el campo de texto; el texto se explora de izquierda a derecha. Puede especificarse un rango utilizando un guión (-).

La propiedad `restrict` sólo limita la interacción del usuario; un script puede introducir cualquier texto en el campo de texto. Esta propiedad no se sincroniza con las casillas de verificación de Incorporar contornos de fuentes del inspector de propiedades.

Si la cadena empieza por "^", inicialmente se aceptan todos los caracteres; los caracteres posteriores de la cadena se excluyen del conjunto de caracteres aceptados. Si la cadena no empieza por "^", inicialmente no se acepta ningún carácter; los caracteres posteriores de la cadena se incluyen en el conjunto de caracteres aceptados.



Puede utilizar el carácter barra invertida para introducir los caracteres “-”, “^” y “\”, como en el ejemplo siguiente:

```
\^  
\-  
\\
```

Cuando se introduce el carácter \ en el panel Acciones entre " " (comillas dobles), tiene un significado especial para el intérprete de comillas dobles del panel Acciones. Significa que el carácter después del carácter \ debe tratarse como tal. Por ejemplo, el siguiente código se utiliza para introducir una comilla sencilla:

```
var leftQuote = "\"";
```

El intérprete .restrict del panel Acciones también utiliza \ como un carácter de escape. Por tanto, es posible que crea que la siguiente expresión debe funcionar:

```
myText.restrict = "0-9\-\^\\\";
```

Sin embargo, el hecho de que esta expresión se encuentre entre comillas dobles, hace que el valor siguiente se envíe al intérprete .restrict: 0-9-^ y el intérprete .restrict no comprende este valor.

Esta expresión se debe incluir entre comillas dobles, pero no por ello debe proporcionarla sólo al intérprete .restrict, sino que también debe anular el intérprete incorporado del panel Acciones para las comillas dobles. Para enviar el valor 0-9\-\^\\ al intérprete .restrict, debe introducir el siguiente código:

```
myText.restrict = "0-9\\-\^\\\\\";
```

## Ejemplo

En el ejemplo siguiente, la primera línea del código limita el campo de texto a letras en mayúsculas, números y espacios. La segunda línea del código admite todos los caracteres excepto letras en minúsculas.

```
my_txt.restrict = "A-Z 0-9";  
my_txt.restrict = "^a-z";
```

El código siguiente permite al usuario introducir los caracteres “0 1 2 3 4 5 6 7 8 9 - ^ \” en la instancia myText. Debe utilizar dos barras invertidas para anular los caracteres “-, ^ y \”. La primera “\” anula el “-”, la segunda “\” le indica al intérprete que el carácter siguiente no debe ser tratado como un carácter especial, como en el ejemplo siguiente:

```
myText.restrict = "0-9\\-\^\\\\\";
```

## TextInput.text

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*textInputInstance.text*

### Descripción

Propiedad; contenido de texto de un componente TextInput. El valor predeterminado es "" (cadena vacía).

### Ejemplo

El código siguiente coloca una cadena en la instancia myTextInput y luego traza dicha cadena en el panel Salida:

```
myTextInput.text = "The Royal Nonesuch";  
trace(myTextInput.text); // traza "The Royal Nonesuch"
```

## Interfaz TransferObject

**Nombre de clase de ActionScript** mx.data.to.TransferObject

La interfaz TransferObject define un conjunto de métodos que deben implementar los elementos gestionados por el componente DataSet. La propiedad DataSet.itemClassName especifica el nombre de la clase del objeto de transferencia del que se crea una instancia cada vez que se necesita un elemento nuevo. También puede especificar esta propiedad mediante el inspector de propiedades para un componente DataSet determinado.

### Resumen de métodos para la interfaz TransferObject

| Método                                           | Descripción                                            |
|--------------------------------------------------|--------------------------------------------------------|
| <a href="#">TransferObject.clone()</a>           | Crea una instancia del objeto de transferencia.        |
| <a href="#">TransferObject.getPropertyData()</a> | Devuelve los datos para este objeto de transferencia.  |
| <a href="#">TransferObject.setPropertyData()</a> | Establece los datos para este objeto de transferencia. |

### TransferObject.clone()

#### Disponibilidad

Flash Player 7.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

```
class itemClass implements mx.data.to.TransferObject  
{  
    function clone() {  
        // aquí código propio  
    }  
}
```

#### Valor devuelto

Una copia del objeto de transferencia.

#### Descripción

Método; crea una instancia del objeto de transferencia. La implementación de este método crea una copia del objeto de transferencia existente y de sus propiedades y, a continuación, devuelve dicho objeto.

### Ejemplo

```
class itemClass implements mx.data.to.TransferObject {
    function clone():Object {
        var b:ContactClass = new ContactClass();
        for (var p in this) {
            b[p] = this[p];
        }
        return b;
    }
}
```

## TransferObject.getPropertyData()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
class itemClass implements mx.data.to.TransferObject {
    function getPropertyData() {
        // aquí código propio
    }
}
```

### Valor devuelto

Un objeto.

### Descripción

Método; devuelve los datos para este objeto de transferencia. La implementación de este método puede devolver un objeto ActionScript anónimo con las propiedades y los valores correspondientes.

### Ejemplo

```
class Contact implements mx.data.to.TransferObject {
    function getPropertyData():Object {
        var internalData:Object = { name:name, readOnly:_readOnly, phone:phone,
            zip:zip.zipPlus4 };
        return( internalData );
    }
}
```

## TransferObject.setPropertyData()

### Disponibilidad

Flash Player 7.

### Edición

Flash MX 2004.

### Sintaxis

```
class yourClass implements TransferObject {
    function setPropertyData(propData) {
        // aquí código propio
    }
}
```

### Parámetros

*propData* Objeto que contiene los datos asignados a este objeto de transferencia.

### Valor devuelto

Ninguno.

### Descripción

Método; establece los datos para este objeto de transferencia. El parámetro *propData* es un objeto cuyos campos contienen los datos asignados por el componente DataSet a este objeto de transferencia.

### Ejemplo

```
class Contact implements mx.data.to.TransferObject {

    function setPropertyData( data: Object ):Void {
        _readOnly = data.readOnly;
        phone = data.phone;
        zip = new mx.data.types.ZipCode( data.zip );
    }

    public var name:String;
    public var phone:String;
    public var zip:ZipCode;
    private var _readOnly:Boolean; // indica si es inmutable
}
```

## Componente Tree (sólo para Flash Professional)

El componente Tree permite a un usuario visualizar datos jerárquicos. El árbol aparece dentro de un cuadro como el componente List, pero cada elemento de un árbol se denomina *nodo* y puede ser una *hoja* o una *rama*. De forma predeterminada, una hoja se representa con una etiqueta de texto junto a un icono de archivo, mientras que una rama se representa con una etiqueta de texto junto a un icono de carpeta con un triángulo de difusión que el usuario puede abrir para exponer los secundarios. Los secundarios de una rama pueden ser hojas u otras ramas.

Los datos de un componente Tree deben proporcionarse desde una fuente de datos SML. Para más información, consulte la sección siguiente.

Si se selecciona una instancia Tree al hacer clic en ella o mediante la tabulación, podrá utilizar las teclas siguientes para controlar dicha instancia:

| Tecla            | Descripción                                                                                                                     |
|------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Flecha abajo     | La selección desciende un elemento.                                                                                             |
| Flecha arriba    | La selección asciende un elemento.                                                                                              |
| Flecha derecha   | Abre un nodo de rama seleccionado. Si una rama ya está abierta, se desplaza al primer nodo secundario.                          |
| Flecha izquierda | Cierra un nodo de rama seleccionado. Si se encuentra en un nodo hoja de un nodo de rama cerrado, se desplaza al nodo principal. |
| Espacio          | Abre o cierra un nodo de rama seleccionado.                                                                                     |
| Fin              | La selección se desplaza al final de la lista.                                                                                  |
| Inicio           | La selección se desplaza al principio de la lista.                                                                              |
| Av Pág           | La selección avanza una página.                                                                                                 |
| Re Pág           | La selección retrocede una página.                                                                                              |
| Control          | Permite varias selecciones no consecutivas.                                                                                     |
| Mayús            | Permite varias selecciones consecutivas.                                                                                        |

Los lectores de pantalla no tienen acceso al componente Tree.

## Utilización del componente Tree (sólo para Flash Professional)

El componente Tree se puede utilizar para representar datos jerárquicos como las carpetas del cliente de correo electrónico, los paneles del explorador de archivos o los sistemas de exploración por categorías para el inventario. En muchas ocasiones, los datos de un árbol se recuperan desde un servidor con formato de archivo XML, pero también puede ser un XML con formato correcto creado mientras se edita en Flash. El mejor modo de crear XML para el árbol es utilizar la [Interfaz TreeDataProvider \(sólo para Flash Professional\)](#). También puede utilizar la clase XML de ActionScript o crear una cadena XML. Después de crear una fuente de datos XML (o cargar una desde una fuente externa), debe asignarla a [Tree.dataProvider](#).

El componente Tree está formado por dos conjuntos de API: la clase Tree y la interfaz TreeDataProvider. La clase Tree contiene los métodos y las propiedades de configuración visual. La interfaz TreeDataProvider permite crear XML y añadirlo a varias instancias de árbol. Un objeto TreeDataProvider difunde los cambios a todos los árboles que lo utilicen. Asimismo, se asignarán los métodos y las propiedades de TreeDataProvider a cualquier objeto XML o XMLNode que se encuentre en el mismo fotograma que un árbol o un menú. Para más información, consulte [“Interfaz TreeDataProvider \(sólo para Flash Professional\)” en la página 588](#).

## Formato XML para el componente Tree

El componente Tree se ha diseñado para mostrar estructuras de datos jerárquicos. XML es el modelo de datos para el componente Tree. Es importante comprender la relación existente entre la fuente de datos XML y el componente Tree.

Observe el siguiente ejemplo de fuente de datos XML:

```
<node>
  <node label="Mail">
    <node label="INBOX"/>
    <node label="Personal Folder">
      <node label="Business" isBranch="true" />
      <node label="Demo" isBranch="true" />
      <node label="Personal" isBranch="true" />
      <node label="Saved Mail" isBranch="true" />
      <node label="bar" isBranch="true" />
    </node>
    <node label="Sent" isBranch="true" />
    <node label="Trash"/>
  </node>
</node>
```

**Nota:** el atributo `isBranch` es de sólo lectura; no se puede configurar directamente. Para configurarlo, llame al método `Tree.setIsBranch()`.

Los nodos de la fuente de datos XML pueden tener cualquier nombre. Observe en el ejemplo anterior que cada nodo recibe el nombre genérico "node". El árbol lee el objeto XML y crea la jerarquía de visualización basándose en la relación anidada de los nodos.

Cada uno de los nodos XML se puede visualizar como uno de los dos tipos que hay en el árbol: rama u hoja. Los nodos rama pueden contener varios nodos secundarios y aparecer como un icono de carpeta con un triángulo de difusión, que permite a los usuarios abrir y cerrar la carpeta. Los nodos hoja aparecen como un icono de archivo y no pueden contener nodos secundarios. Tanto las hojas como las ramas pueden ser raíces; los nodos raíz aparecen en el nivel superior del árbol y no tienen un nodo principal. Los iconos son personalizables; para más información, consulte [“Utilización de aspectos con el componente Tree” en la página 574](#).

XML se puede estructurar de varias maneras. El componente Tree no está diseñado para utilizar todos los tipos de estructuras XML, por lo que es importante utilizar XML que el componente Tree puede interpretar. No deben anidarse atributos de nodo en un nodo secundario; cada nodo debe contener todos sus atributos necesarios. Asimismo, los atributos de cada nodo deben ser coherentes para resultar de utilidad. Por ejemplo, para describir una estructura de buzón de correo con un componente Tree, deben utilizarse los mismos atributos en cada nodo (mensaje, datos, hora, archivos adjuntos, etc.). Esto permite al árbol conocer lo que va a representar y al usuario desplazarse desde el último elemento de la jerarquía al primero y viceversa para comparar datos.

Cuando un árbol muestra un nodo, utiliza de forma predeterminada el atributo `label` del nodo como etiqueta de texto. Si existen otros atributos, se convierten en propiedades adicionales de los atributos del nodo dentro del árbol.

El nodo raíz real se considera como el propio componente Tree. Esto significa que `firstChild` (en el ejemplo, `<node label="Mail">`), se representa como el nodo raíz en la vista de árbol. Esto demuestra que un árbol puede tener varias carpetas raíz. En este ejemplo, el árbol sólo muestra una carpeta raíz: "Mail". Sin embargo, si añadiera nodos colaterales a ese nivel en el XML, se visualizarían varios nodos raíz en el árbol.

## Parámetros del componente Tree

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Tree en el inspector de propiedades o en el panel Inspector de componentes:

**multipleSelection** Valor booleano que indica si se permite la selección de varios valores (*true*) o no se permite (*false*). El valor predeterminado es *false*.

**rowHeight** Altura de cada una de las filas, en píxeles. El valor predeterminado es 20.

Puede escribir código ActionScript para controlar éstas y otras opciones del componente Tree utilizando sus propiedades, métodos y eventos. Para más información, consulte [“Clase Tree \(sólo para Flash Professional\)” en la página 574](#).

No se pueden especificar parámetros de datos en el inspector de propiedades ni en el panel Inspector de componentes para el componente Tree como se puede hacer con otros componentes. Para más información, consulte [“Utilización del componente Tree \(sólo para Flash Professional\)” en la página 569](#) y [“Creación de una aplicación con el componente Tree” en la página 571](#).

## Creación de una aplicación con el componente Tree

En este ejemplo, un desarrollador crea una aplicación de correo electrónico y decide utilizar un componente Tree para mostrar los buzones.

No se pueden especificar parámetros de datos en el inspector de propiedades ni en el panel Inspector de componentes como se puede hacer con otros componentes. Dado que la estructura de los datos es más compleja para los componentes Tree, debe importar un objeto XML durante el tiempo de ejecución o crear uno en Flash durante la edición. Para crear XML en Flash, puede utilizar `TreeDataProvider` o el objeto XML de ActionScript o crear una cadena XML. Cada una de estas opciones se describe en los siguientes procedimientos.

### Para añadir un componente Tree a una aplicación:

- 1 En Flash, seleccione Archivo > Nuevo y, a continuación, Documento de Flash.
- 2 En el panel Componentes, haga doble clic en el componente Tree para añadirlo al escenario.
- 3 En el inspector de propiedades, introduzca el nombre de instancia **myTree**.
- 4 En el panel de Acciones, en el fotograma 1 especifique el código siguiente, que crea un controlador de eventos `change`:

```
listenerObject = new Object();
listenerObject.change = function(evtObject){
    trace(evtObject.target.selectedItem.attributes.label + " was selected");
}
myTree.addEventListener("change", listenerObject);
```

La acción `trace` que se encuentra dentro del controlador envía un mensaje al panel Salida cada vez que se selecciona un elemento del árbol.

- 5 Siga uno de los siguientes procedimientos para cargar o crear una fuente de datos XML para el árbol.

**Para cargar el objeto XML desde un archivo externo, siga este procedimiento:**

- 1 Siga los pasos anteriores para añadir un componente Tree a una aplicación y crear un controlador de eventos `change`.
- 2 En el panel Acciones del Fotograma 1, introduzca el código siguiente:

```
myTreeDP = new XML();
myTreeDP.ignoreWhite = true;
myTreeDP.load("http://myServer.myDomain.com/source.xml");
myTreeDP.onLoad = function(){
    myTree.dataProvider = myTreeDP;
}
```

Este código crea un objeto XML de ActionScript denominado `myTreeDP` y llama al método `XML.load()` para cargar una fuente de datos XML. A continuación, el código define un controlador de eventos `onLoad` que establece la propiedad `dataProvider` de la instancia `myTree` en el nuevo objeto XML cuando se carga XML. Para más información sobre el objeto XML, consulte su entrada en el apartado Diccionario de ActionScript de la Ayuda.

- 3 Seleccione Control > Probar película.

En el archivo SWF, puede visualizar la estructura de XML en el componente Tree. Haga clic en el componente Tree para ver cómo las acciones `trace` del controlador de eventos `change` envían los valores de datos al panel Salida.

**Para utilizar la clase `TreeDataProvider` para crear XML en Flash durante la edición, siga este procedimiento:**

- 1 Siga los pasos del primer procedimiento que se muestra más arriba, añada un componente Tree a una aplicación y cree un controlador de eventos `change`.
- 2 En el panel Acciones del Fotograma 1, introduzca el código siguiente:

```
var myTreeDP = new XML();
myTreeDP.addTreeNode("Local Folders", 0);

// Utilizar XML.firstChild para anidar nodos secundarios en Local Folders
var myTreeNode = myTreeDP.firstChild;
myTreeNode.addTreeNode("Inbox", 1);
myTreeNode.addTreeNode("Outbox", 2);
myTreeNode.addTreeNode("Sent Items", 3);
myTreeNode.addTreeNode("Deleted Items", 4);

// Asignar la fuente de datos myTreeDP con el componente myTree
myTree.dataProvider = myTreeDP;

// Configurar cada uno de los 4 nodos secundarios para que sean ramas
for(var i=0; i<myTreeNode.childNodes.length; i++){
    var node = myTreeNode.getTreeNodeAt(i);
    myTree.setIsBranch(node, true);
}
```

Este código crea un objeto XML denominado `myTreeDP`. Cualquier objeto XML que se encuentre en el mismo fotograma que un componente Tree recibe automáticamente todas las propiedades y los métodos de la interfaz API `TreeDataProvider`. La segunda línea de código crea un único nodo raíz denominado Local Folders. Para información detallada sobre el resto del código, vea los comentarios (líneas precedidas por `//`) que se encuentran en todo el código.



### 3 Seleccione Control > Probar película.

En el archivo SWF, puede visualizar la estructura de XML en el componente Tree. Haga clic en el componente Tree para ver cómo las acciones `trace` del controlador de eventos `change` envían los valores de datos al panel Salida.

**Para utilizar la clase XML de ActionScript para crear XML, siga este procedimiento:**

- 1 Siga los pasos del primer procedimiento que se muestra más arriba, añada un componente Tree a una aplicación y cree un controlador de eventos `change`.
- 2 En el panel Acciones del Fotograma 1, introduzca el código siguiente:

```
// Crear un objeto XML
var myTreeDP = new XML();
// Crear los valores de nodo
var myNode0 = myTreeDP.createElement("node");
myNode0.attributes.label = "Local Folders";
myNode0.attributes.data = 0;

var myNode1 = myTreeDP.createElement("node");
myNode1.attributes.label = "Inbox";
myNode1.attributes.data = 1;

var myNode2 = myTreeDP.createElement("node");
myNode2.attributes.label = "Outbox";
myNode2.attributes.data = 2;

var myNode3 = myTreeDP.createElement("node");
myNode3.attributes.label = "Sent Items";
myNode3.attributes.data = 3;

var myNode4 = myTreeDP.createElement("node");
myNode4.attributes.label = "Deleted Items";
myNode4.attributes.data = 4;
// Asignar nodos a la jerarquía del árbol XML
myTreeDP.appendChild(myNode0);
myTreeDP.firstChild.appendChild(myNode1);
myTreeDP.firstChild.appendChild(myNode2);
myTreeDP.firstChild.appendChild(myNode3);
myTreeDP.firstChild.appendChild(myNode4);
// Asignar la fuente de datos myTreeDP con el control de árbol
myTree.dataProvider = myTreeDP;
```

Por favor, lea los comentarios que se incluyen en el código (líneas que empiezan por `//`) para obtener una descripción del código. Para más información sobre el objeto XML, consulte su entrada en el apartado Diccionario de ActionScript de la Ayuda.

### 3 Seleccione Control > Probar película.

En el archivo SWF, puede visualizar la estructura de XML en el componente Tree. Haga clic en el componente Tree para ver cómo las acciones `trace` del controlador de eventos `change` envían los valores de datos al panel Salida.

**Para utilizar una cadena con formato correcto que permita crear un objeto XML en Flash durante la edición, siga este procedimiento:**

- 1 Siga los pasos del primer procedimiento que se muestra más arriba, añada un componente Tree a una aplicación y cree un controlador de eventos `change`.
- 2 En el panel Acciones del Fotograma 1, introduzca el código siguiente:

```
myTreeDP = new XML("<node label='Local Folders'><node label='Inbox' data='0' /><node label='Outbox' data='1' /></node>");  
myTree.dataProvider = myTreeDP;
```

El código anterior crea un objeto XML `myTreeDP` y lo asigna a la propiedad `dataProvider` de `myTree`.

- 3 Seleccione Control > Probar película.

En el archivo SWF, puede visualizar la estructura de XML en el componente Tree. Haga clic en el componente Tree para ver cómo las acciones `trace` del controlador de eventos `change` envían los valores de datos al panel Salida.

## Personalización del componente Tree (sólo para Flash Professional)

El componente Tree puede transformarse horizontal o verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice el método `setSize()` (véase `UIObject.setSize()`). Cuando un árbol no es suficientemente ancho para mostrar el texto de los nodos, el texto se recorta.

## Utilización de estilos con el componente Tree

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

## Utilización de aspectos con el componente Tree

Para obtener información reciente sobre esta función, haga clic en el botón Actualizar en la parte superior de la ficha Ayuda.

## Clase Tree (sólo para Flash Professional)

**Herencia** UIObject > UIComponent > View > ScrollView > ScrollSelectList > List > Tree

**Nombre de clase de ActionScript** mx.controls.Tree

## Resumen de métodos de la clase Tree

| Método                                 | Descripción                                                                                       |
|----------------------------------------|---------------------------------------------------------------------------------------------------|
| <code>Tree.addTreeNode()</code>        | Añade un nodo a la instancia de un árbol.                                                         |
| <code>Tree.addTreeNodeAt()</code>      | Añade un nodo en una ubicación específica de la instancia de un árbol.                            |
| <code>Tree.getDisplayIndex()</code>    | Devuelve el índice de visualización de un nodo determinado.                                       |
| <code>Tree.getIsBranch()</code>        | Especifica si la carpeta es una rama (cuenta con un icono de carpeta y una flecha de ampliación). |
| <code>Tree.getIsOpen()</code>          | Indica si una rama está abierta o cerrada.                                                        |
| <code>Tree.getNodeDisplayedAt()</code> | Devuelve el índice de visualización de un nodo determinado.                                       |
| <code>Tree.getTreeNodeAt()</code>      | Devuelve un nodo en la raíz del árbol.                                                            |
| <code>Tree.removeAll()</code>          | Elimina todos los nodos de la instancia de un árbol y actualiza el árbol.                         |
| <code>Tree.removeTreeNodeAt()</code>   | Elimina un nodo en una posición específica y actualiza el árbol.                                  |
| <code>Tree.setIsBranch()</code>        | Indica si un nodo es una rama (recibe un icono de carpeta y una flecha de ampliación).            |
| <code>Tree.setIcon()</code>            | Especifica si un nodo está abierto o cerrado.                                                     |
| <code>Tree.setIsOpen()</code>          | Especifica un símbolo que va a utilizarse como icono para un nodo.                                |

Hereda todos los métodos de `UIComponent`, `UIObject`, `View`, `ScrollView`, `ScrollSelectList` y `List`.

## Resumen de propiedades de la clase Tree

| Propiedad                          | Descripción                                                    |
|------------------------------------|----------------------------------------------------------------|
| <code>Tree.dataProvider</code>     | Especifica una fuente de datos XML.                            |
| <code>Tree.firstVisibleNode</code> | Especifica el primer nodo en la parte superior de la pantalla. |
| <code>Tree.selectedNode</code>     | Especifica el nodo seleccionado en una instancia de árbol.     |
| <code>Tree.selectedNodes</code>    | Especifica los nodos seleccionados en una instancia de árbol.  |

Hereda todas las propiedades de `UIComponent`, `UIObject`, `View`, `ScrollView`, `ScrollSelectList` y `List`.

## Resumen de eventos de la clase Tree

| Evento                      | Descripción                                  |
|-----------------------------|----------------------------------------------|
| <code>Tree.nodeClose</code> | Se difunde cuando un usuario cierra un nodo. |
| <code>Tree.nodeOpen</code>  | Se difunde cuando un usuario abre un nodo.   |

Hereda todos los eventos de `UIComponent`, `UIObject`, `View`, `ScrollView`, `ScrollSelectList` y `List`.

## Tree.addTreeNode()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
myTree.addTreeNode(label [, data])
```

Sintaxis 2:

```
myTree.addTreeNode(child)
```

### Parámetros

*label* Cadena que muestra el nodo o un objeto con un campo “label” (o con cualquier nombre de campo de etiqueta que especifique la propiedad `labelField`).

*data* Objeto de cualquier tipo que está asociado con el nodo. Este parámetro es opcional.

*child* Cualquier objeto XMLNode.

### Valor devuelto

El nodo XML añadido.

### Descripción

Método; añade un nodo secundario al árbol. El nodo se crea a partir de la información proporcionada en los parámetros `label` y `data` (Sintaxis 1) o a partir del nodo secundario creado previamente, que es un objeto XMLNode (Sintaxis 2). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

**Nota:** al llamar a este método se actualiza la vista.

### Ejemplo

El código siguiente añade un nuevo nodo a la raíz de `myTree`. La segunda línea de código desplaza un nodo de la raíz de `mySecondTree` a la raíz de `myTree`:

```
myTree.addTreeNode("Inbox", 3);  
myTree.addTreeNode(mySecondTree.getTreeNodeAt(3));
```

## Tree.addNodeAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
myTree.addNodeAt(index, label [, data])
```

Sintaxis 2:

```
myTree.addNodeAt(index, child)
```

### Parámetros

*index* Orden (en los nodos secundarios) que debe seguirse para añadir un nodo.

*label* Cadena que muestra el nodo.

*data* Objeto de cualquier tipo que está asociado con el nodo. Este parámetro es opcional.

*child* Cualquier objeto XMLNode.

### Valor devuelto

El nodo XML añadido.

### Descripción

Método; añade un nodo a la ubicación especificada del árbol. El nodo se crea a partir de la información proporcionada en los parámetros *label* y *data* (Sintaxis 1) o a partir del objeto XMLNode creado previamente (Sintaxis 2). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

**Nota:** al llamar a este método se actualiza la vista.

### Ejemplo

En el ejemplo siguiente se añade un nuevo nodo como segundo nodo secundario de la raíz de *myTree*. La segunda línea desplaza un nodo de *mySecondTree* para convertirlo en el cuarto nodo secundario de la raíz de *myTree*:

```
myTree.addNodeAt(1, "Inbox", 3);  
myTree.addNodeAt(3, mySecondTree.getTreeNodeAt(3));
```

## Tree.dataProvider

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myTree.dataProvider`

### Descripción

Propiedad; la propiedad `dataProvider` puede ser XML o una cadena. Si `dataProvider` es un objeto XML, se añade directamente al árbol. Si `dataProvider` es una cadena, debe contener XML válido que el árbol pueda leer y convertir en un objeto XML.

Puede cargar XML desde una fuente externa en tiempo de ejecución o crearlo en Flash durante la edición. Para crear XML, puede utilizar los métodos de `TreeDataProvider` o los métodos y las propiedades de la clase XML de ActionScript incorporados. También puede crear una cadena que contenga XML.

Los objetos XML que se encuentran en el mismo fotograma que un componente Tree contienen automáticamente los métodos y las propiedades de `TreeDataProvider`. Puede utilizar los objetos XML o `XMLNode` de ActionScript.

### Ejemplo

En el ejemplo siguiente se importa un archivo XML y se asigna a la instancia `myTree` del componente Tree:

```
myTreeDP = new XML();
myTreeDP.ignoreWhite = true;
myTreeDP.load("http://myServer.myDomain.com/source.xml");
myTreeDP.onLoad = function(){
    myTree.dataProvider = myTreeDP;
}
```

**Nota:** la mayoría de los archivos XML contienen espacios en blanco. Flash no ignora dichos espacios a menos que el valor de la propiedad `ignoreWhite` sea `true`.

## Tree.firstVisibleNode

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`myTree.firstVisibleNode`

## Descripción

Propiedad; primer nodo en la parte superior de la pantalla. Si el nodo se encuentra bajo un nodo que no se ha ampliado, la definición de `firstVisibleNode` no tiene ningún efecto. El valor predeterminado es el primer nodo visible o `undefined`, en el caso de que no haya ningún nodo visible. El valor de esta propiedad es un objeto `XMLNode`.

**Nota:** establecer esta propiedad tiene los mismos efectos que establecer `List.vPosition`.

## Ejemplo

En el ejemplo siguiente se establece la posición de desplazamiento en la parte superior de la visualización:

```
myTree.firstVisibleNode = myTree.getTreeNodeAt(0);
```

## Tree.getIsBranch()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.getIsBranch(node)
```

### Parámetros

*node*    Objeto `XMLNode`.

### Valor devuelto

Valor booleano que indica si el nodo es una rama (`true`) o no (`false`).

## Descripción

Método; indica si el nodo especificado cuenta con un icono de carpeta y una flecha de ampliación (es una *rama*). Esto se establece de forma automática al añadir nodos secundarios al nodo. Sólo debe llamar a `setIsBranch()` para crear carpetas vacías. Para más información, consulte [Tree.setIsBranch\(\)](#).

## Ejemplo

El código siguiente asigna el estado del nodo a una variable:

```
var open = myTree.getIsBranch(myTree.getTreeNodeAt(1));
```

## Véase también

[Tree.setIsBranch\(\)](#)

## Tree.getIsOpen()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.getIsOpen(node)
```

### Parámetros

*node* Objeto XMLNode.

### Valor devuelto

Valor booleano que indica si el árbol está abierto (`true`) o no (`false`).

### Descripción

Método; indica si el nodo especificado está abierto o cerrado.

### Ejemplo

El código siguiente asigna el estado del nodo a una variable:

```
var open = myTree.getIsOpen(myTree.getTreeNodeAt(1));
```

## Tree.getDisplayIndex()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.getDisplayIndex(node)
```

### Parámetros

*node* Objeto XMLNode.

### Valor devuelto

El índice del nodo especificado o `undefined` si el nodo no se está mostrando en ese momento.

### Descripción

Método; devuelve el índice de visualización del nodo especificado en el parámetro *node*.

El índice de visualización es una matriz de elementos que se pueden visualizar en la ventana del árbol. Por ejemplo, ningún nodo secundario de un nodo cerrado se incluye en el índice de visualización. El índice de visualización empieza por 0 y sigue por los elementos visibles independientemente de su nodo principal. En otras palabras, el índice de visualización es el número de fila, empezando por 0, de las filas que se está visualizando.



## Ejemplo

El código siguiente obtiene el índice de visualización de `myNode`:

```
var x = myTree.getDisplayIndex(myNode);
```

## Tree.getNodeDisplayedAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.getNodeDisplayedAt(index)
```

### Parámetros

*index* Entero que representa la posición de visualización en el área visualizable del árbol. La serie de números está basada en cero; el nodo de la primera posición es 0, la segunda posición es 1, etc.

### Valor devuelto

Objeto XMLNode especificado.

### Descripción

Método; asigna un índice de visualización del árbol al nodo que se muestra en esa posición. Por ejemplo, si la quinta fila del árbol muestra un nodo que se encuentra en el octavo nivel de la jerarquía, dicho nodo se devolverá especificando `getNodeDisplayedAt(4)`.

El índice de visualización es una matriz de elementos que se pueden visualizar en la ventana del árbol. Por ejemplo, ningún nodo secundario de un nodo cerrado se incluye en el índice de visualización. El índice de visualización empieza por 0 y sigue por los elementos visibles independientemente de su nodo principal. En otras palabras, el índice de visualización es el número de fila, empezando por 0, de las filas que se está visualizando.

**Nota:** los índices de visualización cambian cada vez que los nodos se abren y se cierran.

## Ejemplo

El código siguiente obtiene una referencia del nodo XML, que es la segunda fila que se muestra en `myTree`:

```
myTree.getNodeDisplayedAt(1);
```

## Tree.getTreeNodeAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.getTreeNodeAt(index)
```

### Parámetros

*index*    Número de índice de un árbol.

### Valor devuelto

Un objeto XMLNode.

### Descripción

Método; devuelve el nodo especificado en la raíz de myTree.

### Ejemplo

El código siguiente obtiene el segundo nodo en el primer nivel del árbol myTree:

```
myTree.getTreeNodeAt(1);
```

## Tree.nodeClose

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();  
listenerObject.nodeClose = function(eventObject) {  
    // introduzca aquí el código propio  
}  
myTreeInstance.addEventListener("nodeClose", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario cierra los nodos de un componente Tree.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. El componente Tree difunde un evento `nodeClose` cuando se cierra uno de sus nodos con un clic. El evento se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (*eventObject*) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `Tree.nodeClose` tiene una propiedad adicional: `node` (el nodo XML que se ha cerrado). Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, se define un controlador denominado `myTreeListener` y se pasa al método `myTree.addEventListener()` como segundo parámetro. El objeto del evento es capturado por el controlador `nodeClose` en el parámetro `evtObject`. Cuando se difunde el evento `nodeClose`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
myTreeListener = new Object();
myTreeListener.nodeClose = function(evtObject){
    trace(evtObject.node + " node was closed");
}
myTree.addEventListener("nodeClose", myTreeListener);
```

## Tree.nodeOpen

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
listenerObject = new Object();
listenerObject.nodeOpen = function(eventObject){
    // introducir aquí el código propio
}
myTreeInstance.addEventListener("nodeOpen", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando un usuario abre un nodo de un componente Tree.

Los componentes V2 utilizan un modelo de evento distribuidor/detector. El componente Tree distribuye un evento `nodeOpen` cuando un usuario abre un nodo con un clic. El evento se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (`listenerObject`) que crea el usuario. Llame al método `addEventListener()` y pase el nombre del controlador como parámetro.

Cuando se activa el evento, éste pasa automáticamente un objeto de evento (`eventObject`) al controlador. Cada objeto de evento tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. El objeto del evento `Tree.nodeOpen` tiene una propiedad adicional: `node` (el nodo XML que se ha abierto). Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

En el ejemplo siguiente, se define un controlador denominado `myTreeListener` y se pasa al método `myTree.addEventListener()` como segundo parámetro. El objeto del evento es capturado por el controlador `nodeOpen` en el parámetro `evtObject`. Cuando se difunde el evento `nodeOpen`, se envía una sentencia `trace` al panel Salida, como se indica a continuación:

```
myTreeListener = new Object();
myTreeListener.nodeOpen = function(evtObject){
    trace(evtObject.node + " node was opened");
}
myTree.addEventListener("nodeOpen", myTreeListener);
```

## Tree.removeAll()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.removeAll()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; elimina todos los nodos y actualiza el árbol.

## Ejemplo

El código siguiente vacía `myTree`:

```
myTree.removeAll();
```

## Tree.removeTreeNodeAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.removeTreeNodeAt(index)
```

### Parámetros

*index* Número de índice de un nodo secundario de un árbol. Cada nodo secundario de un árbol tiene asignado un índice basado en cero en el orden en que se ha creado.

**Valor devuelto**

Un objeto `XMLNode` o `undefined` si se produce algún error.

**Descripción**

Método; elimina un nodo (especificado por su posición de índice) en la raíz del árbol y lo actualiza.

**Ejemplo**

El código siguiente elimina el cuarto nodo secundario de la raíz del árbol `myTree`:

```
myTree.removeTreeNodeAt(3);
```

**Tree.setIsBranch()****Disponibilidad**

Flash Player 6 versión 79.

**Edición**

Flash MX Professional 2004.

**Sintaxis**

```
myTree.setIsBranch(node, isBranch)
```

**Parámetros**

*node*   Nodo XML.

*isBranch*   Valor booleano que indica si el nodo es una rama (`true`) o no (`false`).

**Valor devuelto**

Ninguno.

**Descripción**

Método; especifica si el nodo tiene un icono de carpeta y una flecha de ampliación y si tiene nodos secundarios o puede tenerlos. Un nodo se configura automáticamente como rama si tiene nodos secundarios; sólo tiene que llamar a `setIsBranch()` si desea crear una carpeta vacía. Puede crear ramas que aún no cuenten con nodos secundarios; por ejemplo, si desea que se carguen nodos secundarios sólo cuando un usuario abre una carpeta.

Al llamar al método `setIsBranch()` se actualizan todas las vistas.

**Ejemplo**

El código siguiente convierte un nodo de `myTree` en rama;

```
myTree.setIsBranch(myTree.getTreeNodeAt(1), true);
```

## Tree.setIcon()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.setIcon(node, linkID[, linkID2])
```

### Parámetros

*node*    Nodo XML.

*linkID*    Identificador de vinculación de un símbolo que va a utilizarse como icono junto al nodo. Este parámetro se utiliza para los nodos hoja y para el estado cerrado de los nodos rama.

*linkID2*    Identificador de vinculación de un símbolo que va a utilizarse como icono junto al nodo. Este parámetro se utiliza para el icono que representa el estado abierto de los nodos rama.

### Valor devuelto

Ninguno.

### Descripción

Método; especifica un icono para el nodo especificado. Este método toma un parámetro (*linkID*) para los nodos hoja y dos parámetros (*linkID* y *linkID2*) para las ramas (los iconos de cerrado y abierto). El segundo parámetro se ignora para los nodos hoja (los que no son rama) y en caso de que sólo se especifique un parámetro para un nodo rama, el icono se utiliza tanto para el estado cerrado como para el abierto.

### Ejemplo

El código siguiente especifica que se va a utilizar un símbolo con el identificador de vinculación “imageIcon” junto al segundo nodo myTree:

```
myTree.setIcon(myTree.getTreeNodeAt(1), "imageIcon");
```

## Tree.setIsOpen()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.setIsOpen(node, isOpen[, noEvent])
```

## Parámetros

*node*    Nodo XML.

*isOpen*    Valor booleano que abre un nodo (*true*) o lo cierra (*false*).

*noEvent*    Valor booleano que anima la transición que se está abriendo (*true*) o no (*false*). Este parámetro es opcional.

## Valor devuelto

Ninguno.

## Descripción

Método; abre o cierra un nodo.

## Ejemplo

El código siguiente abre un nodo de *myTree*:

```
myTree.setIsOpen(myTree.getTreeNodeAt(1), true);
```

## Tree.selectedNode

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.selectedNode
```

### Descripción

Propiedad; especifica el nodo seleccionado en una instancia de árbol.

### Ejemplo

En el ejemplo siguiente se especifica el primer nodo secundario de *myTree*:

```
myTree.selectedNode = myTree.getTreeNodeAt(0);
```

### Véase también

[Tree.selectedNodes](#)

## Tree.selectedNodes

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myTree.selectedNodes
```

## Descripción

Propiedad; especifica los nodos seleccionados en una instancia de árbol.

## Ejemplo

En el ejemplo siguiente se seleccionan los nodos primero y tercero de myTree:

```
myTree.selectedNodes = [myTree.getTreeNodeAt(0), myTree.getTreeNodeAt(2)];
```

## Véase también

[Tree.selectedNode](#)

# Interfaz TreeDataProvider (sólo para Flash Professional)

TreeDataProvider es una interfaz, por lo que no es necesario que se creen instancias de ella para utilizarla. Si una clase Tree se empaqueta en un archivo SWF, todas las instancias XML del archivo SWF contienen la interfaz API TreeDataProvider. Todos los nodos de un árbol son objetos XML que contienen la interfaz API TreeDataProvider.

Es recomendable utilizar los métodos de la interfaz API TreeDataProvider para crear XML para la propiedad `Tree.dataProvider`, ya que sólo TreeDataProvider difunde eventos a los componentes Tree que actualizan la visualización del árbol. Los métodos de la clase XML incorporados se pueden utilizar para crear XML, pero no difunden los eventos que actualizarán la visualización.

Se pueden utilizar los métodos de la interfaz API TreeDataProvider para controlar el modelo de datos y la visualización de datos. Se pueden utilizar los métodos de la clase XML incorporados para realizar tareas de sólo lectura como la navegación por la jerarquía del árbol.

La propiedad que contiene el texto que debe mostrarse puede seleccionarse especificando las propiedades `labelField` o `labelFunction`. Por ejemplo, `code myTree.labelField = "fred";` indica que se consulte la propiedad `myTreeDP.attributes.fred` para obtener el texto que se va a visualizar.

## Resumen de métodos para la interfaz TreeDataProvider

| Evento                                              | Descripción                                                                             |
|-----------------------------------------------------|-----------------------------------------------------------------------------------------|
| <a href="#">TreeDataProvider.addNode()</a>          | Añade un nodo secundario al final de un nodo principal.                                 |
| <a href="#">TreeDataProvider.addNodeAt()</a>        | Añade un nodo secundario en una ubicación especificada del nodo principal.              |
| <a href="#">TreeDataProvider.getTreeNodeAt()</a>    | Devuelve el nodo secundario especificado de un nodo.                                    |
| <a href="#">TreeDataProvider.removeTreeNode()</a>   | Elimina un nodo y todos sus descendientes del nodo principal de dicho nodo.             |
| <a href="#">TreeDataProvider.removeTreeNodeAt()</a> | Elimina un nodo y todos sus descendientes de la posición de índice del nodo secundario. |



## Resumen de propiedades para la interfaz `TreeDataProvider`

| Propiedad                                      | Descripción                                             |
|------------------------------------------------|---------------------------------------------------------|
| <code>TreeDataProvider.attributes.data</code>  | Especifica los datos que van a asociarse con un nodo.   |
| <code>TreeDataProvider.attributes.label</code> | Especifica el texto que va a mostrarse junto a un nodo. |

### `TreeDataProvider.addNode()`

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

Sintaxis 1:

```
someNode.addTreeNode(label, data)
```

Sintaxis 2:

```
someNode.addTreeNode(child)
```

#### Parámetros

*label* Cadena que muestra el nodo.

*data* Objeto de cualquier tipo que está asociado con el nodo.

*child* Cualquier objeto `XMLNode`.

#### Valor devuelto

El nodo XML añadido.

#### Descripción

Método; añade un nodo secundario a la raíz del árbol. El nodo se crea a partir de la información proporcionada en los parámetros `label` y `data` (Sintaxis 1) o a partir del nodo secundario creado previamente, que es un objeto `XMLNode` (Sintaxis 2). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

La llamada a este método actualiza la visualización de la instancia de árbol.

#### Ejemplo

La primera línea de código del ejemplo siguiente localiza el nodo al que debe añadirse un nodo secundario. La segunda línea añade un nuevo nodo a un nodo especificado, como se muestra a continuación:

```
var myTreeNode = myTreeDP.firstChild.firstChild;  
myTreeNode.addTreeNode("Inbox", 3);
```

El código siguiente traslada un nodo de un árbol a la raíz de otro árbol:

```
myTreeNode.addTreeNode(mySecondTree.getTreeNodeAt(3));
```

## TreeDataProvider.addTreeNodeAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

Sintaxis 1:

```
someNode.addTreeNodeAt(index, label, data)
```

Sintaxis 2:

```
someNode.addTreeNodeAt(index, child)
```

### Parámetros

*index* Entero que indica la posición de índice entre los nodos secundarios en la que debe añadirse el nodo.

*label* Cadena que muestra el nodo.

*data* Objeto de cualquier tipo que está asociado con el nodo.

*child* Cualquier objeto XMLNode.

### Valor devuelto

El nodo XML añadido.

### Descripción

Método; añade un nodo secundario a la ubicación especificada del nodo principal. El nodo se crea a partir de la información proporcionada en los parámetros label y data (Sintaxis 1) o a partir del nodo secundario creado previamente, que es un objeto XMLNode (Sintaxis 2). Si se añade un nodo que ya existía, éste se elimina de su ubicación anterior.

La llamada a este método actualiza la visualización de la instancia de árbol.

### Ejemplo

El código siguiente localiza el nodo al que se añadirá un nodo y añade un nuevo nodo como segundo nodo secundario de la raíz:

```
var myTreeNode = myTreeDP.firstChild.firstChild;  
myTreeNode.addTreeNodeAt(1, "Inbox", 3);
```

El código siguiente desplaza un nodo de un árbol para que sea el cuarto nodo secundario de la raíz de otro árbol:

```
myTreeNode.addTreeNodeAt(3, mySecondTree.getTreeNodeAt(3));
```

## TreeDataProvider.attributes.data

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*someNode.attributes.data*

### Descripción

Propiedad; especifica los datos que van a asociarse con el nodo. Esto añade el valor como un atributo del objeto del nodo XML. Al definir esta propiedad no se actualizan las visualizaciones del árbol. Esta propiedad puede ser de cualquier tipo de datos.

### Ejemplo

El código siguiente localiza el nodo que va a ajustarse y define su propiedad `data`:

```
var myTreeNode = myTreeDP.firstChild.firstChild;  
myTreeNode.attributes.data = "hi"; // tiene como resultado <node data = "hi">;
```

### Véase también

[TreeDataProvider.attributes.label](#)

## TreeDataProvider.attributes.label

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*someNode.attributes.label*

### Descripción

Propiedad; cadena que especifica el texto que se visualiza para el nodo. El texto se escribe en un atributo del objeto del nodo XML. Al definir esta propiedad no se actualizan las visualizaciones de ningún árbol.

### Ejemplo

El código siguiente localiza el nodo que va a ajustarse y define su propiedad `label`. El resultado del código siguiente es “<node label="Mail">”:

```
var myTreeNode = myTreeDP.firstChild.firstChild;  
myTreeNode.attributes.label = "Mail";
```

### Véase también

[TreeDataProvider.attributes.data](#)

## TreeDataProvider.getTreeNodeAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
someNode.getTreeNodeAt(index)
```

### Parámetros

*index* Entero que representa la posición del nodo secundario en el nodo actual.

### Valor devuelto

El nodo especificado.

### Descripción

Método; devuelve el nodo secundario especificado del nodo.

### Ejemplo

El código siguiente localiza un nodo y obtiene el segundo nodo secundario de `myTreeNode`:

```
var myTreeNode = myTreeDP.firstChild.firstChild;  
myTreeNode.getTreeNodeAt(1);
```

## TreeDataProvider.removeTreeNode()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
someNode.removeTreeNode()
```

### Parámetros

Ninguno.

### Valor devuelto

El nodo XML eliminado o undefined si se produce un error.

### Descripción

Método; elimina el nodo especificado, y los descendientes, de su nodo principal.

### Ejemplo

El código siguiente elimina un nodo:

```
myTreeDP.firstChild.removeTreeNode();
```

## TreeDataProvider.removeTreeNodeAt()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
someNode.removeTreeNodeAt(index)
```

### Parámetros

*index* Entero que indica la posición del nodo que debe eliminarse.

### Valor devuelto

El nodo XML eliminado o undefined si se produce un error.

### Descripción

Método; elimina un nodo (y todos sus descendientes) especificado por el nodo actual y la posición de índice del nodo secundario. Al llamar a este método se actualiza la vista.

### Ejemplo

El código siguiente elimina el cuarto nodo secundario de un nodo determinado:

```
myTreeDP.firstChild.removeTreeNodeAt(3);
```

## UIComponent

**Herencia** UIObject > UIComponent

**Nombre de clase de ActionScript** mx.core.UIComponent

Todos los componentes de la v2 amplían UIComponent; no es un componente visual. La clase UIComponent contiene funciones y propiedades que permiten a los componentes de Macromedia compartir algunos comportamientos comunes. La clase UIComponent permite realizar lo siguiente:

- Recibir selecciones y entradas del teclado
- Activar y desactivar componentes
- Cambiar el tamaño mediante el diseño

Para utilizar los métodos y propiedades de UIComponent, puede llamarlos directamente desde cualquier componente en uso. Por ejemplo, para llamar al método `UIComponent.setFocus()` desde el componente `RadioButton`, deberá escribir el código siguiente:

```
myRadioButton.setFocus();
```

Sólo necesita crear una instancia de UIComponent si está utilizando la arquitectura de componentes de Macromedia V2 para crear un componente nuevo. Incluso en tal caso, UIComponent se crea de forma implícita por medio de otras subclases, por ejemplo `Button`. Si necesita crear una instancia de UIComponent, utilice el código siguiente:

```
class MyComponent extends UIComponent;
```

## Resumen de métodos de la clase `UIComponent`

| Método                              | Descripción                                        |
|-------------------------------------|----------------------------------------------------|
| <code>UIComponent.setFocus()</code> | Devuelve una referencia al objeto seleccionado.    |
| <code>UIComponent.setFocus()</code> | Define la selección en la instancia de componente. |

Hereda todos los métodos de la clase `UIObject`.

## Resumen de propiedades de la clase `UIComponent`

| Propiedad                         | Descripción                                                                  |
|-----------------------------------|------------------------------------------------------------------------------|
| <code>UIComponent.enabled</code>  | Indica si el componente puede recibir selecciones y entradas.                |
| <code>UIComponent.tabIndex</code> | Número que indica el orden de tabulación para un componente de un documento. |

Hereda todas las propiedades de la clase `UIObject`.

## Resumen de eventos de la clase `UIComponent`

| Evento                            | Descripción                                        |
|-----------------------------------|----------------------------------------------------|
| <code>UIComponent.focusIn</code>  | Se difunde cuando se selecciona un objeto.         |
| <code>UIComponent.focusOut</code> | Se difunde cuando un objeto deja de seleccionarse. |
| <code>UIComponent.keyDown</code>  | Se difunde cuando se presiona una tecla.           |
| <code>UIComponent.keyUp</code>    | Se difunde cuando se suelta una tecla.             |

Hereda todos los eventos de la clase `UIObject`.

## `UIComponent.focusIn`

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
on(focusIn){
    ...
}
listenerObject = new Object();
listenerObject.focusIn = function(eventObject){
    ...
}
componentInstance.addEventListener("focusIn", listenerObject)
```

## Descripción

Evento; notifica al detector que el objeto se ha seleccionado con el teclado.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `focusIn`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

El código siguiente desactiva un botón mientras el usuario escribe en el campo de texto `txt`:

```
txtListener.handleEvent = function(eventObj) {  
    form.button.enabled = false;  
}  
txt.addEventListener("focusIn", txtListener);
```

## Véase también

`UIEventDispatcher.addEventListener()`

## UIComponent.focusOut

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
on(focusOut){  
    ...  
}  
listenerObject = new Object();  
listenerObject.focusOut = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("focusOut", listenerObject)
```

## Descripción

Evento; notifica a los detectores que el objeto ya no tiene la selección del teclado.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `focusOut`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

El código siguiente activa un botón cuando el usuario abandona el campo de texto `txt`:

```
txtListener.handleEvent = function(eventObj) {  
    if (eventObj.type == focusOut){  
        form.button.enabled = true;  
    }  
}  
txt.addEventListener("focusOut", txtListener);
```

## Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## UIComponent.enabled

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*componentInstance.enabled*

### Descripción

Propiedad; indica si el componente puede aceptar selecciones y entradas del ratón. Si el valor es `true`, puede recibir selecciones y entradas; si el valor es `false`, no puede. El valor predeterminado es `true`.



### Ejemplo

En el ejemplo siguiente se establece la propiedad `enabled` de un componente `CheckBox` en `false`:

```
checkBoxInstance.enabled = false;
```

## UIComponent.setFocus()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.setFocus();
```

### Parámetros

Ninguno.

### Valor devuelto

Referencia al objeto que recibe actualmente la selección.

### Descripción

Método; devuelve una referencia al objeto que recibe la selección del teclado.

### Ejemplo

El código siguiente devuelve una referencia al objeto que recibe la selección que asigna a la variable `tmp`:

```
var tmp = checkbox.setFocus();
```

## UIComponent.keyDown

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
on(keyDown){  
    ...  
}  
listenerObject = new Object();  
listenerObject.keyDown = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("keyDown", listenerObject)
```

## Descripción

Evento; notifica a los detectores que se ha presionado una tecla. Éste es un evento de nivel muy bajo que no debería utilizarse a menos que sea necesario, ya que puede afectar al rendimiento del sistema.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `keyDown`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

El código siguiente hace que un icono parpadee cuando se presiona una tecla:

```
formListener.handleEvent = function(eventObj)
{
    form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyDown", formListener);
```

## UIComponent.keyUp

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
on(keyUp){
    ...
}
listenerObject = new Object();
listenerObject.keyUp = function(eventObject){
    ...
}
componentInstance.addEventListener("keyUp", listenerObject)
```

## Descripción

Evento; notifica a los detectores cuando se suelta una tecla. Éste es un evento de nivel muy bajo que no debería utilizarse a menos que sea necesario, ya que puede afectar al rendimiento del sistema.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `keyUp`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

El código siguiente hace que un icono parpadee cuando se suelta una tecla:

```
formListener.handleEvent = function(eventObj)
{
    form.icon.visible = !form.icon.visible;
}
form.addEventListener("keyUp", formListener);
```

## UIComponent.setFocus()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.setFocus();
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; define la selección en esta instancia del componente. La instancia con la selección recibe todas las entradas del teclado.

### Ejemplo

El código siguiente define la selección sobre la instancia `checkbox`:

```
checkbox.setFocus();
```

## UIComponent.tabIndex

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
instance.tabIndex
```

### Descripción

Propiedad; número que indica el orden de tabulación para un componente en un documento.

### Ejemplo

El código siguiente define el valor de `tmp` en la propiedad `tabIndex` de la instancia `checkbox`:

```
var tmp = checkbox.tabIndex;
```

## UIEventDispatcher

**Nombre de clase de ActionScript**    `mx.events.EventDispatcher`; `mx.events.UIEventDispatcher`

Los eventos permiten conocer el momento en el que el usuario interactúa con un componente, así como los cambios importantes producidos en el aspecto o el ciclo de vida de un componente, como la creación o eliminación de un componente o sus cambios de tamaño.

Cada componente difunde distintos eventos, que aparecen en la entrada de cada componente. Hay varias formas de utilizar eventos de componentes en un código ActionScript. Para más información, consulte [“Eventos de componentes” en la página 25](#).

Utilice `UIEventDispatcher.addEventListener()` para registrar un detector con una instancia de componente. El detector se invoca cuando se activa un evento del componente.

## UIEventDispatcher.addEventListener()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004 y Flash MX Professional 2004.

## Sintaxis

```
componentInstance.addEventListener(event, listener)
```

## Parámetros

*event* Cadena que es el nombre del evento.

*listener* Referencia a un objeto detector o a una función.

## Valor devuelto

Ninguno.

## Descripción

Método; registra un objeto detector con una instancia de componente que difunde un evento. Cuando el evento se activa, se notifica al objeto detector o a la función. Se puede invocar este método desde cualquier instancia de componente. Por ejemplo, el código siguiente registra un detector para la instancia de componente `myButton`:

```
myButton.addEventListener("click", myListener);
```

Es necesario definir el detector (tanto si es un objeto como una función) antes de llamar a `addEventListener()` para registrar el detector con la instancia de componente. Si el detector es un objeto, debe tener definida una función callback que se invoque al activarse el evento. Por lo general, la función callback tiene el mismo nombre que el evento con el que se ha registrado el detector. Si el detector es una función, ésta se invoca cuando se activa el evento. Para más información, consulte [“Utilización de detectores de eventos de componente” en la página 26](#).

Es posible registrar varios detectores para una sola instancia de componente, pero es preciso utilizar una llamada individual al `addEventListener()` de cada detector. Asimismo, se puede registrar un detector para varias instancias de componente, pero hay que utilizar una llamada individual al `addEventListener()` de cada instancia. Por ejemplo, el siguiente código define un objeto detector y lo asigna a dos instancias de componente `Button`:

```
lo = new Object();
lo.click = function(evt){
    if (evt.target == button1){
        trace("button 1 clicked");
    } else if (evt.target == button2){
        trace("button 2 clicked");
    }
}
button1.addEventListener("click", lo);
button2.addEventListener("click", lo);
```

El orden de ejecución no está garantizado. No espere que se llame a un detector antes que a otro.

Un objeto `Event` pasa al detector como un parámetro. El objeto `Event` tiene propiedades que contienen información sobre el evento producido. Es posible utilizar el objeto `Event` dentro de la función callback del detector para acceder a la información sobre el tipo de evento que tuvo lugar y qué instancia difundió el evento. En el ejemplo anterior, el objeto de evento es `evt` (se puede utilizar cualquier identificador como nombre del objeto `Event`) y se utiliza en las sentencias `if` para determinar sobre qué instancia de botón se ha hecho clic. Para más información, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

En el ejemplo siguiente se define un objeto detector, `myListener`, y el clic de la función `callback`. A continuación, llama a `addEventListener()` para registrar el objeto detector `myListener` con la instancia del componente `myButton`. Para probar este código, coloque un componente `Button` en el escenario con el nombre de instancia `myButton` e introduzca el código siguiente en el fotograma 1:

```
myListener = new Object();
myListener.click = function(evt){
    trace(evt.type + " triggered");
}
myButton.addEventListener("click", myListener);
```

### Objetos Event

Un objeto `Event` pasa al detector como un parámetro. El objeto `Event` es un objeto de `ActionScript` que tiene propiedades donde se incluye información acerca del evento que ha tenido lugar. El objeto `Event` se puede utilizar dentro de la función `callback` del detector para acceder al nombre del evento que se difundió o al nombre de instancia del componente que difundió el evento. Por ejemplo, el código siguiente utiliza la propiedad `target` del objeto de evento `evtObj` para acceder a la propiedad `label` de la instancia `myButton` y enviar el valor al panel Salida:

```
listener = new Object();
listener.click = function(evtObj){
    trace("The " + evtObj.target.label + " button was clicked");
}
myButton.addEventListener("click", listener);
```

Algunas propiedades de objetos `Event` se definen en la [especificación W3C](#), pero no están implementadas en la versión 2 (v2) de la arquitectura de componentes de Macromedia. Cada objeto `Event` de la v2 tiene las propiedades que se especifican en la tabla que se muestra a continuación. Algunos eventos tienen propiedades adicionales definidas y, en este caso, las propiedades se enumeran en la entrada del evento.

### Propiedades del objeto Event

| Propiedad           | Descripción                                                     |
|---------------------|-----------------------------------------------------------------|
| <code>type</code>   | Cadena que indica el nombre del evento.                         |
| <code>target</code> | Referencia a la instancia del componente que difunde el evento. |

# UIObject

**Herencia** MovieClip > UIObject

**Nombre de clase de ActionScript** mx.core.UIObject

UIObject es la clase base para todos los componentes de v2; no es un componente visual. La clase UIObject ajusta el objeto MovieClip de ActionScript y contiene funciones y propiedades que permiten a los componentes de la v2 de Macromedia compartir algunos comportamientos comunes. El ajuste de la clase MovieClip permite a Macromedia añadir nuevos eventos y ampliar funciones en el futuro sin que ello afecte al contenido. Asimismo, el ajuste de la clase MovieClip permitirá a los usuarios que no conocen los conceptos tradicionales de Flash como “película” y “fotograma” utilizar la interfaz API para crear aplicaciones basadas en componentes sin conocer dichos conceptos.

La clase UIObject incorpora lo siguiente:

- Estilos
- Eventos
- Cambiar el tamaño mediante la escala

Para utilizar los métodos y propiedades de UIObject, puede llamarlos directamente desde cualquier componente en uso. Por ejemplo, para llamar al método `UIObject.setSize()` desde el componente `RadioButton`, deberá escribir el código siguiente:

```
myRadioButton.setSize(30, 30);
```

Sólo debe crear una instancia de UIObject si está utilizando la arquitectura de componentes de Macromedia V2 para crear un componente. Incluso en este caso, UIObject se crea con frecuencia de forma implícita por medio de otras subclases, por ejemplo `Button`. Si necesita crear una instancia de UIObject, utilice el código siguiente:

```
class MyComponent extends UIObject;
```

## Resumen de métodos de la clase UIObject

| Método                                    | Descripción                                                                                        |
|-------------------------------------------|----------------------------------------------------------------------------------------------------|
| <code>UIObject.createObject()</code>      | Crea un subobjeto en un objeto.                                                                    |
| <code>UIObject.createClassObject()</code> | Crea un objeto en la clase especificada.                                                           |
| <code>UIObject.destroyObject()</code>     | Elimina una instancia de componente.                                                               |
| <code>UIObject.invalidate()</code>        | Marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.    |
| <code>UIObject.move()</code>              | Mueve el objeto a la posición indicada.                                                            |
| <code>UIObject.redraw()</code>            | Fuerza la validación del objeto, de forma que se pueda volver a dibujar sobre el fotograma actual. |
| <code>UIObject.setSize()</code>           | Cambia el tamaño del objeto al indicado.                                                           |
| <code>UIObject.setSkin()</code>           | Define un aspecto en el objeto.                                                                    |

## Resumen de propiedades de la clase UIObject

| Propiedad                     | Descripción                                                                                                          |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------|
| <code>UIObject.bottom</code>  | Posición del borde inferior del objeto con respecto al borde inferior de su principal correspondiente. Sólo lectura. |
| <code>UIObject.height</code>  | Altura del objeto, expresada en píxeles. Sólo lectura.                                                               |
| <code>UIObject.left</code>    | Posición izquierda del objeto, expresada en píxeles. Sólo lectura.                                                   |
| <code>UIObject.right</code>   | Posición del borde derecho del objeto con respecto al borde derecho de su principal correspondiente. Sólo lectura.   |
| <code>UIObject.scaleX</code>  | Número que indica el factor de escala en la dirección x del objeto con respecto a su principal correspondiente.      |
| <code>UIObject.scaleY</code>  | Número que indica el factor de escala en la dirección y del objeto con respecto a su principal correspondiente.      |
| <code>UIObject.top</code>     | Posición del borde superior del objeto con respecto a su principal correspondiente. Sólo lectura.                    |
| <code>UIObject.visible</code> | Valor booleano que indica si el objeto es visible ( <code>true</code> ) o no ( <code>false</code> ).                 |
| <code>UIObject.width</code>   | Anchura del objeto, expresada en píxeles. Sólo lectura.                                                              |
| <code>UIObject.x</code>       | Posición izquierda del objeto, expresada en píxeles. Sólo lectura.                                                   |
| <code>UIObject.y</code>       | Devuelve la posición del borde superior del objeto con respecto a su principal correspondiente. Sólo lectura.        |

## Resumen de eventos de la clase UIObject

| Evento                       | Descripción                                                               |
|------------------------------|---------------------------------------------------------------------------|
| <code>UIObject.draw</code>   | Se difunde cuando un objeto está a punto de dibujar sus gráficos.         |
| <code>UIObject.hide</code>   | Se difunde cuando el estado de un objeto pasa de ser visible a invisible. |
| <code>UIObject.load</code>   | Se difunde cuando se han creado subobjetos.                               |
| <code>UIObject.move</code>   | Se difunde cuando se ha movido el objeto.                                 |
| <code>UIObject.resize</code> | Se difunde cuando se ha redimensionado el objeto.                         |
| <code>UIObject.reveal</code> | Se difunde cuando el estado de un objeto pasa de ser invisible a visible. |
| <code>UIObject.unload</code> | Se difunde cuando se han descargado los subobjetos.                       |

### UIObject.bottom

#### Disponibilidad

Flash Player 6 versión 79.

#### Edición

Flash MX 2004.



## Sintaxis

*componentInstance.bottom*

## Descripción

Propiedad (sólo lectura); número expresado en píxeles que indica la posición inferior del objeto con respecto a la parte inferior de su principal correspondiente. Para establecer esta propiedad, llame al método `UIObject.move()`.

## Ejemplo

En este ejemplo se desplaza la casilla de verificación para alinearla debajo del borde inferior del cuadro de lista:

```
myCheckbox.move(myCheckbox.x, form.height - listbox.bottom);
```

## UIObject.createObject()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*componentInstance.createObject(linkageName, instanceName, depth, initObject)*

## Parámetros

*linkageName* Cadena que indica el identificador de vinculación de un símbolo en el panel Biblioteca.

*instanceName* Cadena que indica el nombre de instancia de la instancia nueva.

*depth* Número que indica la profundidad de la nueva instancia.

*initObject* Objeto que contiene propiedades de inicialización para la nueva instancia.

## Valor devuelto

UIObject que es una instancia del símbolo.

## Descripción

Método; crea un subobjeto en un objeto. Por lo general, sólo lo utilizan los desarrolladores avanzados o de componentes.

## Ejemplo

En el ejemplo siguiente se crea una instancia `CheckBox` en el objeto `form`:

```
form.createObject("CheckBox", "sym1", 0);
```

## UIObject.createClassObject()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.createClassObject(className, instanceName, depth, initObject)
```

### Parámetros

*className*    Objeto que indica la clase de la nueva instancia.

*instanceName*    Cadena que indica el nombre de instancia de la instancia nueva.

*depth*    Número que indica la profundidad de la nueva instancia.

*initObject*    Objeto que contiene propiedades de inicialización para la nueva instancia.

### Valor devuelto

UIObject que es una instancia de la clase especificada.

### Descripción

Método; crea un subobjeto de un objeto. Por lo general, sólo lo utilizan los desarrolladores avanzados o de componentes. Este método permite crear componentes en tiempo de ejecución.

Debe especificar el nombre de paquete de clase. Siga uno de estos procedimientos:

```
import mx.controls.Button;  
createClassObject(Button,"button2",5,{label:"Test Button"});
```

o bien

```
createClassObject(mx.controls.Button,"button2",5,{label:"Test Button"});
```

### Ejemplo

En el ejemplo siguiente se crea un objeto CheckBox:

```
form.createClassObject(CheckBox, "cb", 0, {label:"Check this"});
```

## UIObject.destroyObject()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.destroyObject(instanceName)
```

## Parámetros

*instanceName* Cadena que indica el nombre de instancia del objeto que se va a eliminar.

## Valor devuelto

Ninguno.

## Descripción

Método; elimina una instancia de componente.

## UIObject.draw

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
on(draw){  
    ...  
}  
listenerObject = new Object();  
listenerObject.draw = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("draw", listenerObject)
```

### Descripción

Evento; notifica a los detectores que el objeto está a punto de dibujar sus gráficos. Se trata de un evento de nivel muy bajo que no debería utilizarse a menos que sea necesario, ya que puede afectar al rendimiento del sistema.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, *draw*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

## Ejemplo

El código siguiente vuelve a dibujar el objeto `form2` cuando se ha dibujado el objeto `form`:

```
formListener.draw = function(eventObj){
    form2.redraw(true);
}
form.addEventListener("draw", formListener);
```

## Véase también

[UIEventDispatcher.addEventListener\(\)](#)

## UIObject.height

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*componentInstance.height*

### Descripción

Propiedad (sólo lectura); número que indica la altura del objeto, expresada en píxeles. Para cambiar la propiedad `height` llame a la propiedad [UIObject.setSize\(\)](#).

## Ejemplo

En el ejemplo siguiente se aumenta la altura de la casilla de verificación:

```
myCheckbox.setSize(myCheckbox.width, myCheckbox.height + 10);
```

## UIObject.hide

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
on(hide){
    ...
}
listenerObject = new Object();
listenerObject.hide = function(eventObject){
    ...
}
componentInstance.addEventListener("hide", listenerObject)
```

### Descripción

Evento; se difunde cuando la propiedad `visible` del objeto cambia de `true` a `false`.

## Ejemplo

El controlador siguiente muestra un mensaje en el panel Salida cuando el objeto con el que está asociado queda oculto.

```
on(hide) {  
    trace("I've become invisible.");  
}
```

## Véase también

[UIObject.reveal](#)

## UIObject.getStyle()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.getStyle(propertyName)
```

### Parámetros

*propertyName* Cadena que indica el nombre de la propiedad del estilo (por ejemplo, "fontWeight", "borderStyle", y así sucesivamente).

### Valor devuelto

Valor de la propiedad del estilo. El valor puede ser cualquier tipo de datos.

### Descripción

Método; obtiene la propiedad del estilo procedente de styleDeclaration o del objeto. Si la propiedad del estilo es un estilo heredado, los principales del objeto pueden ser el origen del valor del estilo.

Para ver una lista de los estilos admitidos por cada componente, consulte sus entradas individuales.

## Ejemplo

El código siguiente define la propiedad de estilo `fontWeight` de la instancia `ib` en negrita cuando la propiedad de estilo `fontWeight` de la instancia `cb` sea negrita:

```
if (cb.getStyle("fontWeight") == "bold")  
{  
    ib.setStyle("fontWeight", "bold");  
};
```

## UIObject.invalidate()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.invalidate()
```

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; marca el objeto de forma que se pueda volver a dibujar en el siguiente intervalo de fotogramas.

### Ejemplo

En el ejemplo siguiente se marca la instancia pBar de ProgressBar para volver a dibujarla:

```
pBar.invalidate();
```

## UIObject.left

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.left
```

### Descripción

Propiedad (sólo lectura); número que indica en píxeles el borde izquierdo del objeto con respecto a su principal correspondiente. Para establecer esta propiedad, llame al método [UIObject.move\(\)](#).

## UIObject.load

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(load){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.load = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("load", listenerObject)
```

### Descripción

Evento; notifica a los detectores que se ha creado el subobjeto de este objeto.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, `load`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

En el ejemplo siguiente se crea una instancia de `MySymbol` una vez cargada la instancia `form`:

```
formListener.handleEvent = function(eventObj)  
{  
    form.createObject("MySymbol", "sym1", 0);  
}  
form.addEventListener("load", formListener);
```

## UIObject.move

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(move){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.move = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("move", listenerObject)
```

### Descripción

Evento; notifica a los detectores que el objeto se ha movido.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso *move*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

En el ejemplo siguiente se llama al método `move()` para mantener a `form2` 100 píxeles por debajo y a la derecha de `form1`:

```
formListener.handleEvent = function(){  
    form2.move(form1.x + 100, form1.y + 100);  
}  
form1.addEventListener("move", formListener);
```



## UIObject.move()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.move(x, y)
```

### Parámetros

*x* Número que indica la posición de la esquina superior izquierda del objeto con respecto a su principal correspondiente.

*y* Número que indica la posición de la esquina superior izquierda del objeto con respecto a su principal correspondiente.

### Valor devuelto

Ninguno.

### Descripción

Método; mueve el objeto hasta la posición indicada. Debe pasar sólo valores enteros a `UIObject.move()` o de lo contrario el componente puede aparecer borroso.

### Ejemplo

En este ejemplo la casilla de verificación se desplaza hacia la derecha 10 píxeles:

```
myCheckbox.move(myCheckbox.x + 10, myCheckbox.y);
```

## UIObject.redraw()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.redraw(always)
```

### Parámetros

*always* Valor booleano. Si es `true`, dibuja el objeto aunque no se haya llamado a `invalidate()`. Si es `false`, dibuja el objeto sólo si se ha llamado a `invalidate()`.

### Valor devuelto

Ninguno.

## Descripción

Método; fuerza la validación del objeto, de forma que se pueda volver a dibujar sobre el fotograma actual.

## Ejemplo

En el ejemplo siguiente se crean una casilla de verificación y un botón y se dibujan ya que no se espera que otros scripts modifiquen el formulario:

```
form.createClassObject(mx.controls.CheckBox, "cb", 0);
form.createClassObject(mx.controls.Button, "b", 1);
form.redraw(true)
```

## UIObject.resize

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(resize){
    ...
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.resize = function(eventObject){
    ...
}
componentInstance.addEventListener("resize", listenerObject)
```

### Descripción

Evento; notifica a los detectores que se ha cambiado el tamaño del objeto.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, *resize*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. Cada objeto Event tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

En el ejemplo siguiente se llama al método `setSize()` para que `sym1` sea la mitad de ancho y un cuarto de alto cuando `form` se mueva:

```
formListener.handleEvent = function(eventObj) {  
    form.sym1.setSize(sym1.width / 2, sym1.height / 4);  
}  
form.addEventListener("resize", formListener);
```

## UIObject.reveal

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
on(reveal){  
    ...  
}  
listenerObject = new Object();  
listenerObject.reveal = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("reveal", listenerObject)
```

### Descripción

Evento; se difunde cuando la propiedad `visible` del objeto cambia de `false` a `true`.

### Ejemplo

El controlador siguiente muestra un mensaje en el panel Salida cuando el objeto con el que está asociado aparece en pantalla.

```
on(reveal) {  
    trace("I've become visible.");  
}
```

### Véase también

[UIObject.hide](#)

## UIObject.right

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*componentInstance.right*

## Descripción

Propiedad (sólo lectura); número expresado en píxeles que indica la posición derecha del objeto con respecto a la parte derecha de su principal. Para establecer esta propiedad, llame al método `UIObject.move()`.

## Ejemplo

En el ejemplo siguiente se desplaza la casilla de verificación para alinearla debajo del borde derecho del cuadro de lista:

```
myCheckbox.move(form.width - listbox.right, myCheckbox.y);
```

## UIObject.scaleX

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*componentInstance.scaleX*

## Descripción

Propiedad; número que indica el factor de escala en la dirección *x* del objeto con respecto a su principal correspondiente.

## Ejemplo

En el ejemplo siguiente se hace que la casilla de verificación doble su anchura y se define la variable `tmp` en el factor de escala horizontal:

```
checkboxbox.scaleX = 200;  
var tmp = checkbox.scaleX;
```

## UIObject.scaleY

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

## Sintaxis

*componentInstance.scaleY*

## Descripción

Propiedad; número que indica el factor de escala en la dirección *y* del objeto con respecto a su principal.

## Ejemplo

En el ejemplo siguiente se hace que la casilla de verificación doble su altura y se define la variable `tmp` en el factor de escala vertical:

```
checkbox.scaleY = 200;  
var tmp = checkbox.scaleY;
```

## UIObject.setSize()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.setSize(width, height)
```

### Parámetros

*width*    Número que indica la anchura del objeto, expresada en píxeles.

*height*    Número que indica la altura del objeto, expresada en píxeles.

### Valor devuelto

Ninguno.

### Descripción

Método; cambia el tamaño del objeto por el tamaño especificado. Debe pasar sólo valores enteros a `UIObject.setSize()` o de lo contrario el componente puede aparecer borroso. Este método (y todos los métodos y propiedades de `UIObject`) está disponible desde cualquier instancia de componente.

La llamada a este método desde una instancia de `ComboBox` hace que se ajuste el tamaño del cuadro combinado y cambie la propiedad `rowHeight` de la lista que contiene.

## Ejemplo

En este ejemplo se cambia el tamaño de la instancia del componente `pBar` a 100 píxeles de ancho y 100 píxeles de alto:

```
pBar.setSize(100, 100);
```

## UIObject.setSkin()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.setSkin(id, linkageName)
```

## Parámetros

*id* Número que indica la variable. Este valor suele ser una constante definida en la definición de la clase.

*linkageName* Cadena que indica un elemento de la biblioteca.

## Valor devuelto

Ninguno.

## Descripción

Método; define un aspecto en la instancia de componente. Este método se utiliza cuando se desarrollan componentes. No es posible utilizar este método para definir los aspectos de un componente en tiempo de ejecución.

## Ejemplo

En este ejemplo se define un aspecto en la instancia de `checkbox`:

```
checkbox.setSkin(CheckBox.skinIDCheckMark, "MyCustomCheckMark");
```

## UIObject.setStyle()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
componentInstance.setStyle(propertyName, value)
```

### Parámetros

*propertyName* Cadena que indica el nombre de la propiedad del estilo (por ejemplo, "fontWeight", "borderStyle", y así sucesivamente).

*value* Valor de la propiedad.

### Valor devuelto

UIObject que es una instancia de la clase especificada.

### Descripción

Método; define la propiedad de estilo en la declaración de estilos o en el objeto. Si la propiedad de estilo es un estilo heredado, la notificación del nuevo valor se enviará a los elementos secundarios.

Para ver una lista de los estilos admitidos por cada componente, consulte sus entradas individuales.

### Ejemplo

El código siguiente define la propiedad de estilo `fontWeight` de la instancia de la casilla de verificación `cb` en negrita:

```
cb.setStyle("fontWeight", "bold");
```

## UIObject.top

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*componentInstance.top*

### Descripción

Propiedad (sólo lectura); número que indica en píxeles el borde superior del objeto con respecto a su principal correspondiente. Para establecer esta propiedad, llame al método `UIObject.move()`.

## UIObject.unload

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(unload){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.unload = function(eventObject){  
    ...  
}  
componentInstance.addEventListener("unload", listenerObject)
```

### Descripción

Evento; notifica a los detectores que se están descargando los subobjetos de este objeto.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente.

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*componentInstance*) distribuye un evento (en este caso, *unload*), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto *Event* (*eventObject*) al método del objeto detector. Cada objeto *Event* tiene un conjunto de propiedades que contiene información sobre el evento. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

En el ejemplo siguiente se elimina `sym1` cuando se activa el evento `unload`:

```
formListener.handleEvent = function(eventObj) {  
    // eventObj.target es el componente que ha generado el evento change,  
    form.destroyObject(sym1);  
}  
form.addEventListener("unload", formListener);
```

## UIObject.visible

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*componentInstance.visible*

### Descripción

Propiedad; valor booleano que indica si el objeto es visible (`true`) o no (`false`).

### Ejemplo

En el ejemplo siguiente se hace que la instancia de *Loader* `myLoader` sea visible:

```
myLoader.visible = true;
```

## UIObject.width

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.



### Sintaxis

*componentInstance.width*

### Descripción

Propiedad (sólo lectura); número que indica la anchura del objeto, expresada en píxeles. Para modificar la anchura, llame al método `UIObject.setSize()`.

### Ejemplo

En el ejemplo siguiente se aumenta la anchura de la casilla de verificación:

```
myCheckbox.setSize(myCheckbox.width + 10, myCheckbox.height);
```

## UIObject.x

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*componentInstance.x*

### Descripción

Propiedad (sólo lectura); número que indica el borde izquierdo del objeto, expresado en píxeles. Para establecer esta propiedad, llame al método `UIObject.move()`.

### Ejemplo

En el ejemplo siguiente la casilla de verificación se desplaza hacia la derecha 10 píxeles:

```
myCheckbox.move(myCheckbox.x + 10, myCheckbox.y);
```

## UIObject.y

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*componentInstance.y*

### Descripción

Propiedad (sólo lectura); número que indica el borde superior del objeto, expresado en píxeles. Para establecer esta propiedad, llame al método `UIObject.move()`.

### Ejemplo

En el ejemplo siguiente la casilla de verificación se desplaza hacia abajo 10 píxeles:

```
myCheckbox.move(myCheckbox.x, myCheckbox.y + 10);
```

## Clases de servicios Web (sólo para Flash Professional)

Las clases que se encuentran en el paquete `mx.services` sirven para acceder a servicios Web que utilizan el protocolo SOAP (Simple Object Access Protocol). Esta API de `WebService` no es la misma que la interfaz API del componente `WebServiceConnector`. La primera es un conjunto de clases que se pueden utilizar sólo en el código `ActionScript` y suele incluirse en varios productos Macromedia. La segunda es una API exclusiva de Flash MX 2004 que proporciona una interfaz de `ActionScript` para la herramienta de edición visual del componente `WebServiceConnector`.

### Disponibilidad de clases de servicios Web en tiempo de ejecución (sólo para Flash Professional)

Para que las clases de servicios Web estén disponibles en tiempo de ejecución, el componente `WebServiceClasses` debe encontrarse en la biblioteca del archivo FLA. Este componente contiene las clases de tiempo de ejecución que permiten trabajar con servicios Web. Para más información sobre cómo añadir estas clases a FLA, consulte “Trabajo con vinculación de datos y servicios Web en tiempo de ejecución (sólo Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

**Nota:** estas clases están disponibles automáticamente en un documento de Flash cuando se añade el componente `WebServiceConnector` al archivo FLA.

### Clases del paquete `mx.services` (sólo para Flash Professional)

En la tabla siguiente se enumeran las clases del paquete `mx.services`. Estas clases funcionan de modo muy integrado, de modo que para aprender sobre las clases de este paquete, debe leer la información en el orden en el que aparecen las clases en la lista.

| Clase                                                                         | Descripción                                                                                                                                                                                 |
|-------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Clase <code>WebService</code> (sólo para Flash Professional)</a>  | Con un archivo WSDL que define el servicio Web, crea un objeto <code>WebService</code> para llamar a los métodos de servicios Web y para controlar las funciones callback del servicio Web. |
| <a href="#">Clase <code>PendingCall</code> (sólo para Flash Professional)</a> | Objeto devuelto de una llamada de método de servicio Web que se ha implementado para controlar los resultados y los fallos de la llamada.                                                   |
| <a href="#">Clase <code>Log</code> (sólo para Flash Professional)</a>         | Objeto opcional que sirve para registrar la actividad relacionada con un objeto <code>WebService</code> .                                                                                   |
| <a href="#">Clase <code>SOAPCall</code> (sólo para Flash Professional)</a>    | Clase avanzada que contiene información sobre el funcionamiento del servicio Web y que permite controlar determinados comportamientos.                                                      |

### Clase `Log` (sólo para Flash Professional)

La clase `Log` forma parte del paquete `mx.services` y está pensada para utilizarse con la clase `WebService` (véase “[Clase `WebService` \(sólo para Flash Professional\)](#)” en la página 638). Para ver información general sobre las clases del paquete `mx.data.services`, consulte “[Clases de servicios Web \(sólo para Flash Professional\)](#)” en la página 622.

Puede crear un objeto `Log` para registrar la actividad relacionada con un objeto `WebService`. Para ejecutar códigos mientras se envían mensajes al objeto `Log`, utilice la función callback `Log.onLog()`. No hay archivo de registro; el mecanismo de registro es el que haya utilizado durante la función callback `onLog()`, como por ejemplo el envío de mensajes de registro a un comando `trace`.

El constructor de este objeto crea un objeto Log que puede pasarse como un argumento opcional al constructor de WebService (véase “Clase WebService (sólo para Flash Professional)” en la página 638).

**Nombre de clase de ActionScript**    mx.services.Log

## Resumen de funciones callback del objeto Log

| Función callback            | Descripción                                   |
|-----------------------------|-----------------------------------------------|
| <a href="#">Log.onLog()</a> | Envía un mensaje de registro a un objeto Log. |

## Constructor de la clase Log

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myWebSvcLog = new Log([logLevel] [, logName]);
```

### Parámetros

*logLevel*    Nivel de registro que indica los tipos de información que desea registrar en el registro. En el código de servicios Web, los mensajes de registro suelen dividirse en categorías o niveles. El parámetro *logLevel* del constructor del objeto Log está relacionado con estas categorías. Existen tres tipos de *logLevel* disponibles:

- **Log.BRIEF:** Registra el evento de ciclo de vida y las notificaciones de error principales.
- **Log.VERBOSE:** Registra todos los eventos de ciclo de vida y las notificaciones de error.
- **Log.DEBUG:** Registra las métricas y los eventos y errores pormenorizados.

El parámetro *logLevel* predeterminado es *log.BRIEF*.

*logName*    Nombre opcional que se incluye con cada mensaje de registro. Si utiliza varios objetos de registro, puede utilizar *logName* para determinar en qué registro ha quedado almacenado un mensaje determinado.

### Valor devuelto

Ninguno.

### Descripción

Constructor; crea un objeto Log. Utilice este constructor para crear un registro. Una vez que lo haya creado, puede pasarlo a un servicio Web para recibir los mensajes.

## Ejemplo

Puede llamar al constructor `new Log`, el cual devolverá un objeto `Log` para que lo pase al servicio Web:

```
// crea un objeto Log
myWebSvcLog = new Log();
myWebSvcLog.onLog = function(txt)
{
    myTrace(txt)
}
```

A continuación este objeto `Log` se pasa en forma de parámetro al constructor de `WebService`:

```
myWebSvc = new WebService("http://www.myco.com/info.wsdl", myWebSvcLog);
```

Cuando se ejecuta el código de los servicios Web y se envían los mensajes al objeto `Log`, se realiza la llamada a la función `onLog()` de dicho objeto. Aquí es donde debe colocar el código que muestra los mensajes de registro si quiere visualizarlos en tiempo real.

A continuación, se muestran ejemplos de mensajes de registro:

```
7/30 15:22:43 [INFO] SOAP: Decoding PendingCall response
7/30 15:22:43 [DEBUG] SOAP: Decoding SOAP response envelope
7/30 15:22:43 [DEBUG] SOAP: Decoding SOAP response body
7/30 15:22:44 [INFO] SOAP: Decoded SOAP response into result [16 millis]
7/30 15:22:46 [INFO] SOAP: Received SOAP response from network [6469 millis]
7/30 15:22:46 [INFO] SOAP: Parsed SOAP response XML [15 millis]
7/30 15:22:46 [INFO] SOAP: Decoding PendingCall response
7/30 15:22:46 [DEBUG] SOAP: Decoding SOAP response envelope
7/30 15:22:46 [DEBUG] SOAP: Decoding SOAP response body
7/30 15:22:46 [INFO] SOAP: Decoded SOAP response into result [16 millis]
```

## Log.onLog()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myWebSvcLog.onLog = function(message)
```

### Parámetros

*message* Mensaje de registro que se pasa al controlador. Para más información sobre mensajes de registro, consulte [“Clase Log \(sólo para Flash Professional\)” en la página 622](#).

### Valor devuelto

Ninguno.

## Descripción

Función callback de registro; Flash Player llama a esta función cuando se envía un mensaje de registro a un archivo de registro. Este controlador es un buen lugar donde colocar el código que registra o muestra los mensajes de registro, como por ejemplo un comando `trace`. La construcción del registro se describe en [“Clase Log \(sólo para Flash Professional\)” en la página 622](#).

## Ejemplo

En el ejemplo siguiente se crea un objeto de registro, se pasa a un objeto `WebService` nuevo y se controlan los mensajes de registro.

```
// crea un objeto Log
myWebSvcLog = new Log();
// pasa el objeto de registro al servicio Web
myWebService = new WebService(wsdlURI, myWebSvcLog);
// controla los mensajes de registro de entrada
myWebSvcLog.onLog = function(message)
{
    mytrace("Log Event:\r myWebSvcLog.message="+message+);
}
```

## Clase PendingCall (sólo para Flash Professional)

La clase `PendingCall` forma parte del paquete `mx.services` y está pensada para utilizarse con la clase `WebService` (véase [“Clase WebService \(sólo para Flash Professional\)” en la página 638](#)). Para ver información general sobre las clases del paquete `mx.data.services`, consulte [“Clases de servicios Web \(sólo para Flash Professional\)” en la página 622](#).

Cuando se llama a un método para un objeto `WebService`, dicho objeto devuelve un objeto `PendingCall`. Los desarrolladores no crean el objeto `WebService`. Utilice las funciones callback `onResult` y `onFault` del objeto `PendingCall` para controlar la respuesta asíncrona del método de servicio Web. Si el método de servicio Web devuelve un fallo, Flash Player llama a la función callback `PendingCall.onFault` y pasa un objeto `SOAPFault` que representa el fallo del protocolo SOAP de XML devuelto por el servidor/servicio Web. Si la invocación del servicio Web se finaliza correctamente, Flash Player llama a la función callback `PendingCall.onResult` y pasa un objeto de resultado. El objeto de resultado es la respuesta XML del servicio Web descodificada y deserializada en `ActionScript`. Para más información sobre el objeto `WebService`, consulte [“Clase WebService \(sólo para Flash Professional\)” en la página 638](#).

El objeto `PendingCall` también ofrece acceso a los parámetros de salida cuando existe más de uno. Muchos servicios Web devuelven un solo resultado, otros devuelven varios. El “valor devuelto” al que se hace referencia en esta API es simplemente el primer (o único) resultado. Las funciones `PendingCall.getOutputXXX` facilitan el acceso a todos los resultados, no sólo al primero. Por lo tanto, mientras que el “valor devuelto” se entrega en el argumento de la función callback `onResult()`, en el caso de que existan otros parámetros de salida a los que desea acceder, debe utilizar `getOutputValues()` (devuelve una matriz) y `getOutputValue(index)` (devuelve un solo valor) para obtener los valores descodificados de `ActionScript`.

También puede acceder directamente al objeto SOAPParameter. El objeto SOAPParameter es un objeto de ActionScript con dos propiedades: `value` contiene el valor de ActionScript de un parámetro de salida y `element` contiene el valor de XML del parámetro de salida. Las funciones siguientes devuelven un objeto SOAPParameter, o una matriz de objetos SOAPParameter, que contiene el valor (`param.value`), además del elemento XML (`param.element`):  
`getOutputParameters()`, `getOutputParameterByName(name)` y  
`getOutputParameter(index)`.

**Nombre de clase de ActionScript**    `mx.services.PendingCall`

## Resumen de funciones del objeto PendingCall

| Función                                             | Descripción                                                                                     |
|-----------------------------------------------------|-------------------------------------------------------------------------------------------------|
| <code>PendingCall.getOutputParameter()</code>       | Obtiene un objeto SOAPParameter basado en el valor de <code>index</code> que se ha pasado.      |
| <code>PendingCall.getOutputParameterByName()</code> | Obtiene un objeto SOAPParameter basado en el valor de <code>localName</code> que se ha pasado.. |
| <code>PendingCall.getOutputParameters()</code>      | Obtiene una matriz de objetos SOAPParameter.                                                    |
| <code>PendingCall.getOutputValue()</code>           | Obtiene el valor de salida basado en el índice que se ha pasado.                                |
| <code>PendingCall.getOutputValues()</code>          | Obtiene una matriz de todos los valores de salida.                                              |

## Resumen de propiedades del objeto PendingCall

| Propiedad                         | Descripción                                                   |
|-----------------------------------|---------------------------------------------------------------|
| <code>PendingCall.myCall</code>   | Descriptor de operación SOAPCall de la operación PendingCall. |
| <code>PendingCall.request</code>  | Petición SOAP en formato XML sin procesar.                    |
| <code>PendingCall.response</code> | Respuesta SOAP en formato XML sin procesar.                   |

## Resumen de funciones callback del objeto PendingCall

| Función callback                    | Descripción                                                                                             |
|-------------------------------------|---------------------------------------------------------------------------------------------------------|
| <code>PendingCall.onFault()</code>  | Llamada realizada por un servicio Web si el método falla.                                               |
| <code>PendingCall.onResult()</code> | Se realiza esta llamada cuando un método se ha llevado a cabo correctamente y ha devuelto un resultado. |

## Constructor de la clase PendingCall

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Descripción

Los desarrolladores no crean el objeto PendingCall. Cuando se llama a una función para un objeto WebService, el objeto WebService devuelve un objeto PendingCall.

## PendingCall.getOutputParameter()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myPendingCall.getOutputParameter(var index)
```

### Parámetros

*index* Índice del parámetro.

### Valor devuelto

Objeto SOAPParameter con los elementos siguientes:

| Elemento | Descripción                                                             |
|----------|-------------------------------------------------------------------------|
| value    | Valor de ActionScript del parámetro.                                    |
| element  | Código XML sin procesar del parámetro en el sobre SOAP (SOAP envelope). |

### Descripción

Función; obtiene un parámetro de salida adicional del objeto SOAPParameter, el cual contiene el valor y el elemento XML. Las llamadas RPC de SOAP pueden devolver varios parámetros de salida. El primer (o único) valor devuelto siempre se proporciona en el argumento de los resultados de la función callback `onResult()`, pero para poder acceder al resto de los valores debe utilizar funciones como ésta o bien `getOutputValue()`. La función `getOutputParameter()` devuelve el parámetro de salida *nth* en forma de objeto SOAPParameter.

Véanse también [PendingCall.getOutputValue\(\)](#), [PendingCall.getOutputValues\(\)](#), [PendingCall.getOutputParameterByName\(\)](#) y [PendingCall.getOutputParameters\(\)](#).

## Ejemplo

Un archivo descriptor SOAP como el siguiente, `getOutputParameter(1)` devolverá un objeto `SOAPParameter` con `value="Hi there!"` y `element=the <outParam2> XmlNode`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Hi there!</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

## PendingCall.getOutputParameterByName()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myPendingCall.getOutputParameterByName(var localName)
```

### Parámetros

*localName* Nombre local del parámetro. En otras palabras, el nombre de un elemento XML, sin información del espacio de nombres. Por ejemplo, el nombre local de los dos elementos siguientes es `bob`:

```
<bob abc="123">
<xsd:bob def="ghi">
```

### Valor devuelto

Objeto `SOAPParameter` con los elementos siguientes:

| Elemento             | Descripción                                                             |
|----------------------|-------------------------------------------------------------------------|
| <code>value</code>   | Valor de <code>ActionScript</code> del parámetro.                       |
| <code>element</code> | Código XML sin procesar del parámetro en el sobre SOAP (SOAP envelope). |

### Descripción

Función; obtiene los parámetros de salida en forma de objetos `SOAPParameter`, los cuales contienen el valor y el elemento XML. Las llamadas RPC de SOAP pueden devolver varios parámetros de salida. El primer (o único) valor devuelto siempre se proporciona en el argumento de los resultados de la función `callback onResult()`, pero para poder acceder al resto de los valores debe utilizar API como ésta. La llamada de `getOutputParameterByName()` devuelve el parámetro de salida llamado *localName*.

Véanse también `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`, `PendingCall.getOutputParameter()` y `PendingCall.getOutputParameters()`.



## Ejemplo

Un archivo descriptor SOAP como el siguiente, `getOutputParameterByName("outParam2")` devolverá un objeto `SOAPParameter` con `value="Hi there!"` y `element=the <outParam2> XMLNode`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Hi there!</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

## PendingCall.getOutputParameters()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myPendingCall*.getOutputParameterByName()

### Parámetros

Ninguno.

### Valor devuelto

Matriz de objetos `SOAPParameter` con los elementos siguientes:

| Elemento | Descripción                                                             |
|----------|-------------------------------------------------------------------------|
| value    | Valor de ActionScript del parámetro.                                    |
| element  | Código XML sin procesar del parámetro en el sobre SOAP (SOAP envelope). |

### Descripción

Función; obtiene parámetros de salida adicionales del objeto `SOAPParameter`, el cual contiene los valores y los elementos XML. Las llamadas RPC de SOAP pueden devolver varios parámetros de salida. El primer (o único) valor devuelto siempre se proporciona en el argumento de los resultados de la función callback `onResult()`, pero para poder acceder al resto de los valores debe utilizar API como ésta o bien `getOutputValues()`.

Véanse también `PendingCall.getOutputValue()`, `PendingCall.getOutputValues()`, `PendingCall.getOutputParameterByName()` y `PendingCall.getOutputParameter()`.

## PendingCall.getOutputValue()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myPendingCall.getOutputValue(var index)
```

### Parámetros

*index* Índice de un parámetro de salida. El primer parámetro es el índice 0.

### Valor devuelto

Parámetro de salida *nth*.

### Descripción

Función; obtiene el valor de ActionScript descodificado de un parámetro de salida individual. Las llamadas RPC de SOAP pueden devolver varios parámetros de salida. El primer (o único) valor devuelto siempre se proporciona en el argumento de los resultados de la función callback `onResult()`, pero para poder acceder al resto de los valores debe utilizar API como ésta o bien `getOutputParameter()`. La llamada de `getOutputValue()` devuelve el parámetro de salida *nth*.

Véanse también `PendingCall.getOutputParameter()`, `PendingCall.getOutputValues()`, `PendingCall.getOutputParameterByName()` y `PendingCall.getOutputParameters()`.

### Ejemplo

Un archivo descriptor SOAP como el siguiente, `getOutputValue(2)` devolverá `true`.

```
...
<SOAP:Body>
  <rpcResponse>
    <outParam1 xsi:type="xsd:int">54</outParam1>
    <outParam2 xsi:type="xsd:string">Hi there!</outParam2>
    <outParam3 xsi:type="xsd:boolean">true</outParam3>
  </rpcResponse>
</SOAP:Body>
...
```

## PendingCall.getOutputValues()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myPendingCall.getOutputValues()
```

## Parámetros

Ninguno.

## Valor devuelto

Matriz de los valores descodificados de todos los parámetros de salida.

## Descripción

Función; obtiene el valor de ActionScript descodificado de todos los parámetros de salida. Las llamadas RPC de SOAP puede devolver varios parámetros de salida. El primer (o único) valor devuelto siempre se proporciona en el argumento de los resultados de la función callback `onResult()`, pero para poder acceder al resto de los valores debe utilizar API como ésta o bien `getOutputParameters()`.

Véanse también `PendingCall.getOutputValue()`, `PendingCall.getOutputParameter()`, `PendingCall.getOutputParameterByName()` y `PendingCall.getOutputParameters()`.

## PendingCall.myCall

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*PendingCall.myCall*

### Descripción

Propiedad; objeto SOAPCall correspondiente a la operación de PendingCall. El objeto SOAPCall contiene información sobre la operación del servicio Web y permite controlar determinados comportamientos. Para más información, consulte [“Clase SOAPCall \(sólo para Flash Professional\)” en la página 634](#).

### Ejemplo

La función callback `onResult` siguiente rastrea el nombre de la operación SOAPCall.

```
callback.onResult = function(result)
{
    // Comprobar el nombre de la operación
    trace("My operation name is " + this.myCall.name);
}
```

## PendingCall.onFault()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

```
myPendingCallObj.onFault = function(fault)
{
    // controla todos los fallos, por ejemplo indicando al
    // usuario que el servidor no está disponible o que contacte
    // con el servicio de asistencia técnica
}
```

## Parámetros

*fault* Versión del objeto de ActionScript decodificado del error con propiedades. Si la información sobre el error proviene de un servidor con el formato XML, el objeto SOAPFault será la versión de ActionScript decodificada de dicho XML.

El tipo de objeto de error devuelto a `PendingCall.onfault()` es un objeto SOAPFault. No lo crean directamente los desarrolladores, sino que se devuelve como resultado de un error. Este objeto es una asignación de ActionScript del tipo XML de fallo SOAP.

| Propiedad SOAPFault | Descripción                                                                                                                                                                  |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| faultcode           | Cadena; cadena breve que describe el error.                                                                                                                                  |
| faultstring         | Cadena; descripción inteligible del error.                                                                                                                                   |
| detail              | Cadena; información específica de la aplicación asociada con el error, como por ejemplo un seguimiento de la pila u otra información que devuelve el motor del servicio Web. |
| element             | XML; objeto XML que representa la versión XML del fallo.                                                                                                                     |
| faultactor          | Cadena; origen del fallo (opcional si no interviene ningún intermediario).                                                                                                   |

## Valor devuelto

Ninguno.

## Descripción

Función callback del objeto PendingCall; debe utilizar este controlador que Flash Player llama cuando un método de servicio Web ha fallado y ha devuelto un error. El parámetro erróneo es un objeto SOAPFault de ActionScript.

## Ejemplo

En el ejemplo siguiente se controlan los errores devueltos desde el método de servicio Web.

```
// controla todos los errores devueltos debido al uso de un método de servicio
// Web
myPendingCallObj = myWebService.methodName(params)
myPendingCallObj.onFault = function(fault)
{
    // detecta el fallo de SOAP
    DebugOutputField.text = fault.faultstring;

    // añadir código para controlar los fallos, por ejemplo indicando
    // al usuario que el servidor no está disponible o que contacte
    // con el servicio de asistencia técnica
}
```

## PendingCall.onResult()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myPendingCallObj.onResult = function(result)
{
    // detecta el resultado y lo controla para esta aplicación
}
```

### Parámetros

*result* Versión del objeto de ActionScript descodificado del resultado XML devuelto por un método de servicio Web llamado con `myPendingCallObj = myWebService.methodName(params)`.

### Valor devuelto

Ninguno.

### Descripción

Función callback de PendingCall; proporcione este controlador que Flash Player llama cuando un método de servicio Web funciona correctamente y devuelve un resultado. El resultado es la versión del objeto de ActionScript descodificado del XML devuelto por la operación. Para obtener el XML sin procesar en lugar de un resultado descodificado, utilice la propiedad `PendingCall.response` (véase [PendingCall.response](#)).

### Ejemplo

En el ejemplo siguiente se controlan los resultados devueltos del método de servicio Web.

```
// controla los resultados devueltos debido al uso de un método de servicio Web
myPendingCallObj = myWebService.methodName(params)
myPendingCallObj.onResult = function(result)
{
    // detecta el resultado y lo controla para esta aplicación
    ResultOutputField.text = result;
}
```

## PendingCall.request

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
rawXML = myPendingCallback.request;
```

### Descripción

Propiedad PendingCall; contiene el formato XML sin procesar de la petición actual enviada con `myPendingCallback = myWebService.methodName()`. Normalmente, no se utiliza `PendingCall.request`, pero puede resultar útil si está interesado en el SOAP que se envía por red. Utilice esta propiedad para acceder al XML sin procesar de la petición. Utilice `myPendingCallback.onResult()` para obtener la versión de ActionScript de los resultados de la petición.

## PendingCall.response

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
rawXML = myPendingCallback.response;
```

### Descripción

Propiedad PendingCall; contiene formato XML sin procesar de la respuesta a la llamada del método de servicio Web más reciente enviada con `myPendingCallback = myWebService.methodName()`. Normalmente, no se utiliza `PendingCall.response`, pero puede resultar útil si está interesado en el SOAP que se envía por cable. Utilice `myPendingCallback.onResult()` para obtener la versión de ActionScript correspondiente de los resultados de la petición.

## Clase SOAPCall (sólo para Flash Professional)

La clase SOAPCall forma parte del paquete `mx.services` y se ha concebido como una función avanzada que debe utilizarse con la clase `WebService` (véase [“Clase WebService \(sólo para Flash Professional\)” en la página 638](#)). Para ver información general sobre las clases del paquete `mx.data.services`, consulte [“Clases de servicios Web \(sólo para Flash Professional\)” en la página 622](#).

Cuando se crea un objeto `WebService`, éste contiene los métodos correspondientes a las operaciones del URL WSDL que se pasen. En segundo plano, también se crea un objeto `SOAPCall` para cada operación en WSDL. `SOAPCall` es el descriptor de la operación y como tal contiene toda la información sobre dicha operación determinada; es decir, el aspecto del XML en la red, el estilo de la operación, etc. También permite controlar determinados comportamientos. Para obtener el objeto `SOAPCall` para una operación determinada utilice la función `getCall(operationName)`. Existe un solo objeto `SOAPCall` para cada operación, que comparten todas las llamadas activas a dicha operación. Una vez que disponga del objeto `SOAPCall`, puede personalizar el descriptor realizando las acciones siguientes:

- Activar/desactivar la decodificación de la respuesta XML.
- Activar/desactivar el retardo de la conversión de matrices SOAP en objetos de `ActionScript`.
- Cambiar la configuración de simultaneidad para una operación determinada.
- Añadir un encabezado al objeto `SOAPCall`.

**Nombre de clase de `ActionScript`**    `mx.services.SOAPCall`

## Resumen de las funciones del objeto `SOAPCall`

| Function                          | Descripción                                           |
|-----------------------------------|-------------------------------------------------------|
| <code>SOAPCall.addHeader()</code> | Añade un encabezado al objeto <code>SOAPCall</code> . |

## Resumen de propiedades del objeto `SOAPCall`

| Propiedad                            | Descripción                                                                                             |
|--------------------------------------|---------------------------------------------------------------------------------------------------------|
| <code>SOAPCall.concurrency</code>    | Modifica la configuración de simultaneidad de una operación determinada.                                |
| <code>SOAPCall.doDecoding</code>     | Activa/desactiva la decodificación de la respuesta XML.                                                 |
| <code>SOAPCall.doLazyDecoding</code> | Activa/desactiva el retardo de la conversión de matrices SOAP en objetos de <code>ActionScript</code> . |

## Constructor de la clase `SOAPCall`

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Descripción

Los desarrolladores no crean objetos `SOAPCall`, sino que cuando se llama a un método para un objeto `WebService`, dicho objeto devuelve un objeto `PendingCall`. Para acceder al objeto `SOAPCall` asociado, utilice `myPendingCall.myCall`.

## SOAPCall.addHeader()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
SOAPCall.addHeader(var header)
```

### Parámetros

*header* Encabezado que debe añadirse.

### Valor devuelto

Ninguno.

### Descripción

Función; añade un encabezado al objeto SOAPCall.

### Ejemplo

En el ejemplo siguiente se crea un encabezado de SOAP y se asocia con el objeto SOAPCall. Mediante el código siguiente:

```
import mx.services.QName;

var qname = new QName("bar", "http://foo");
var value = "hi there!";
var header = new SOAPHeader(qname, value);
soapCall.addHeader(header);

se crea el encabezado SOAP siguiente:

...
<SOAP:Header>
  <ns1:bar
    xmlns:ns1="http://foo"
    xsi:type="xsd:string">hi there!</ns1:bar>
</SOAP:Header>
...
```

## SOAPCall.concurrency

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
SOAPCall.concurrency
```



## Descripción

Propiedad; número de peticiones simultáneas. Los valores posibles aparecen enumerados en la tabla siguiente:

| Valor                                      | Descripción                                                                        |
|--------------------------------------------|------------------------------------------------------------------------------------|
| <code>SOAPCall.Multiple_Concurrency</code> | Permite varias llamadas activas.                                                   |
| <code>SOAPCall.Single_Concurrency</code>   | Permite una sola llamada cada vez e indica un error cuando hay una llamada activa. |
| <code>SOAPCall.Last_Concurrency</code>     | Permite una sola llamada mediante la cancelación de las anteriores.                |

## SOAPCall.doDecoding

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`SOAPCall.doDecoding`

### Descripción

Propiedad; activa/desactiva la descodificación de la respuesta XML (de forma predeterminada la respuesta XML se convierte, es decir se descodifica, en objetos de ActionScript). Si sólo desea el código XML, puede establecer `SOAPCall.doDecoding = false`.

## SOAPCall.doLazyDecoding

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

`SOAPCall.doLazyDecoding`

### Descripción

Propiedad; activa/desactiva la descodificación diferida de las matrices. De forma predeterminada, se utiliza un algoritmo de descodificación diferida para retardar hasta el último momento la conversión de las matrices SOAP en objetos de ActionScript; de este modo, las operaciones se devuelven mucho más rápido cuando se devuelven grandes conjuntos de datos. Esto significa que las matrices que se devuelven del extremo remoto son objetos ArrayProxy. Después, cuando se accede a un índice determinado (`foo[5]`) dicho elemento se descodifica automáticamente, si es necesario. Este comportamiento puede desactivarse estableciendo `SOAPCall.doLazyDecoding = false`; ello hará que se descodifiquen completamente todas las matrices.

## Clase WebService (sólo para Flash Professional)

La clase WebService forma parte del paquete mx.services y está pensada para utilizarse con las clases siguientes:

- Clase Log (sólo para Flash Professional)
- Clase PendingCall (sólo para Flash Professional)
- Clase SOAPCall (sólo para Flash Professional)

**Nota:** esta API de WebService no es la misma que la interfaz API del componente WebServiceConnector. La primera es un conjunto de clases que se pueden utilizar sólo en el código ActionScript y suele incluirse en varios productos Macromedia. La segunda es una API exclusiva de Flash MX 2004 que proporciona una interfaz de ActionScript para la herramienta de edición visual del componente WebServiceConnector.

Para ver información general de las clases del paquete mx.services, consulte [“Clases de servicios Web \(sólo para Flash Professional\)” en la página 622](#).

El objeto WebServices actúa como una referencia local a un servicio Web remoto. Cuando se crea un objeto WebService, el archivo WSDL que define el servicio Web se descarga, se analiza y se coloca en el objeto. Después, se llaman directamente a los métodos del servicio Web para el objeto WebService y se controlan todas las funciones callback del servicio Web. Si WSDL se ha procesado correctamente y el objeto WebService está listo, se invoca la función callback `onLoad()`. Si se detecta un problema durante la carga del archivo WSDL, se invoca la función callback `onFault()`.

Cuando se llama a un método para un objeto WebService, el valor devuelto es un objeto de función callback. El tipo de objeto de la función callback devuelto de todas las invocaciones de método de servicio Web es `PendingCall`. Estos objetos no suelen estar creados por los desarrolladores, sino que se crean automáticamente como resultado del comando `WebServiceObject.webServiceMethodName()`. Estos objetos no son el resultado de la llamada de WebService, que aparece posteriormente. Sino que el objeto `PendingCall` representa la llamada en curso. Una vez completada la operación de WebService (generalmente segundos después de realizar una llamada de método), se rellenan los diferentes campos de datos de `PendingCall` y se llama a la función callback `onResult` u `onFault` proporcionada por el usuario. Para más información sobre el objeto `PendingCall`, consulte [“Clase PendingCall \(sólo para Flash Professional\)” en la página 625](#).

El reproductor pone en cola las llamadas que se hayan realizado antes de haber analizado el archivo WSDL e intenta ejecutarlas después de analizar el archivo WSDL. Esto se debe a que el archivo WSDL contiene información necesaria para que las peticiones SOAP se codifiquen y envíen correctamente. No es necesario colocar en la cola las llamadas de función que se realicen una vez que se haya analizado el archivo WSDL; éstas se llevan a cabo de forma inmediata. Si una llamada en cola no coincide con el nombre de ninguna de las operaciones definidas en el archivo WSDL, Flash Player devuelve un error al objeto de función callback que se proporcionó al usuario al realizar la llamada originalmente.

**Nombre de clase de ActionScript**    mx.services.WebService

## Utilización de la interfaz API WebServices

La interfaz API WebServices, que se incluye con el paquete mx.services, consta de las clases WebService, Log y PendingCall.

## Tipos compatibles

La función `WebService` es compatible con un subconjunto de tipos de esquema XML, como se define en las tablas siguientes.

Los tipos complejos y el tipo de matriz codificada con SOAP también son compatibles y pueden estar formados por otros tipos complejos, matrices o tipos de esquema XML incorporados:

### Tipos numéricos simples

| Tipo de esquema XML             | Vinculación de ActionScript |
|---------------------------------|-----------------------------|
| <code>decimal</code>            | Number                      |
| <code>integer</code>            | Number                      |
| <code>negativeInteger</code>    | Number                      |
| <code>nonNegativeInteger</code> | Number                      |
| <code>positiveInteger</code>    | Number                      |
| <code>long</code>               | Number                      |
| <code>int</code>                | Number                      |
| <code>short</code>              | Number                      |
| <code>byte</code>               | Number                      |
| <code>unsignedLong</code>       | Number                      |
| <code>unsignedShort</code>      | Number                      |
| <code>unsignedInt</code>        | Number                      |
| <code>unsignedByte</code>       | Number                      |
| <code>float</code>              | Number                      |
| <code>double</code>             | Number                      |

### Tipos simples de fecha y hora

| Tipo de esquema XML     | Vinculación de ActionScript |
|-------------------------|-----------------------------|
| <code>date</code>       | Objeto Date                 |
| <code>datetime</code>   | Objeto Date                 |
| <code>duration</code>   | Objeto Date                 |
| <code>gDay</code>       | Objeto Date                 |
| <code>gMonth</code>     | Objeto Date                 |
| <code>gMonthDay</code>  | Objeto Date                 |
| <code>gYear</code>      | Objeto Date                 |
| <code>gYearMonth</code> | Objeto Date                 |
| <code>time</code>       | Objeto Date                 |

## Tipos simples de nombre y cadena

| Tipo de esquema XML | Vinculación de ActionScript |
|---------------------|-----------------------------|
| string              | Cadena de ActionScript      |
| normalizedString    | Cadena de ActionScript      |
| QName               | Objeto mx.services.Qname    |

## Tipo booleano

| Tipo de esquema XML | Vinculación de ActionScript |
|---------------------|-----------------------------|
| Boolean             | Booleano                    |

## Tipos de objeto

| Tipo de esquema XML | Vinculación de ActionScript                                                   |
|---------------------|-------------------------------------------------------------------------------|
| Any                 | Objeto XML                                                                    |
| Complex Type        | Objeto de ActionScript compuesto por propiedades de cualquier tipo compatible |
| Array               | Matriz de ActionScript compuesta por cualquier objeto o tipo compatible       |

## Elementos compatibles del objeto del esquema XML

```
schema
  complexType
    complexContent
      restriction
    sequence | simpleContent
      restriction
  element
    complexType | simpleType
```

## Seguridad de WebService

La interfaz API de WebService cumple el modelo de seguridad de Flash Player.

## Autenticación y autorización de usuarios

Las reglas de autenticación y autorización son las mismas para la interfaz API de WebService que para cualquier operación de red XML de Flash. El propio protocolo SOAP no especifica ningún método de autenticación ni autorización. Por ejemplo, cuando el transporte HTTP subyacente devuelve una respuesta HTTP BASIC en los encabezados HTTP, el navegador responde presentando un diálogo al usuario y adjuntando seguidamente la entrada del usuario a los encabezados HTTP en mensajes posteriores. Este mecanismo se produce en un nivel inferior al de SOAP y forma parte del diseño de autenticación HTTP de Flash.

## Integridad de los mensajes

La seguridad de los mensajes implica el cifrado de los propios mensajes SOAP en una capa conceptual sobre los paquetes de red en los que se entregan los mensajes SOAP.

**Seguridad de transporte** El transporte de red subyacente para los servicios Web SOAP de Flash Player siempre es HTTP POST. Por lo tanto, los medios de seguridad que se pueden aplicar en la capa de transporte HTTP de Flash, como SSL, se admiten a través de invocaciones de servicios Web desde Flash. SSL/HTTPS proporciona el tipo más común de seguridad de transporte para la mensajería SOAP; además, la utilización de autenticación HTTP BASIC, junto con SSL en la capa de transporte, es el tipo de seguridad más común para sitios Web en la actualidad.

## Resumen de funciones del objeto `WebService`

| Función                                | Descripción                                                                   |
|----------------------------------------|-------------------------------------------------------------------------------|
| <code>WebService.myMethodName()</code> | Invoca una operación de servicio Web específica definida por el archivo WSDL. |
| <code>WebService.getCall()</code>      | Obtiene el objeto SOAPCall de una operación determinada.                      |

## Resumen de las funciones callback del objeto `WebService`

| Función callback                  | Descripción                                                                           |
|-----------------------------------|---------------------------------------------------------------------------------------|
| <code>WebService.onLoad()</code>  | Se llama cuando el servicio Web ha cargado y analizado correctamente su archivo WSDL. |
| <code>WebService.onFault()</code> | Se llama cuando se ha producido un error durante el análisis de WSDL.                 |

## Constructor de la clase `WebService`

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
myWebServiceObject = new WebService(wsdURI [, logObject]);
```

### Parámetros

Los parámetros de constructor son los siguientes:

*wsdURI* URL del archivo WSDL del servicio Web.

*logObject* Parámetro opcional que especifica el nombre del objeto Log de este servicio Web (véase “Clase Log (sólo para Flash Professional)” en la página 622).

### Valor devuelto

Ninguno.

## Descripción

Para crear un objeto `WebService`, realice una llamada a `new WebService()` y proporcione una URL de WSDL. Flash Player devuelve un objeto `WebService`. El constructor del objeto `WebService` puede aceptar opcionalmente un objeto `Log` y una URL de proxy:

```
myWebServiceObject = new WebService(wsdlURI [, logObject]);
```

Si lo desea, puede utilizar dos funciones callback para el objeto `WebService`. Flash Player realiza una llamada a la función `WebServiceObject.onLoad(WSDLDocument)` cuando acaba de analizar el archivo WSDL y el objeto se ha completado. Es una buena ubicación para incluir código que se desea ejecutar sólo después de que el archivo WSDL se haya analizado completamente. Por ejemplo, puede incluir la primera llamada a método de servicio Web en esta función.

Flash Player realiza una llamada a `WebServiceObject.onFault(fault)` cuando se produce un error al buscar o analizar el archivo WSDL. Es una buena ubicación para incluir código de depuración y código que indique al usuario que el servidor no está disponible, que se vuelva a intentar la acción más tarde o información similar. Para más información, consulte las secciones particulares de estas funciones.

**Invocación de una operación de servicio Web:** las operaciones de servicio Web se invocan como método directamente disponible en el servicio Web. Por ejemplo, si el servicio Web tiene el método `getCompanyInfo(tickerSymbol)`, invoque el método de la manera siguiente:

```
myPendingCallObject = myWebServiceObject.getCompanyInfo(tickerSymbol);
```

En el ejemplo anterior, el objeto de función callback se denomina `myPendingCallObject`. Todas las invocaciones de métodos son asíncronas y devuelven un objeto de función callback del tipo `PendingCall`. Asíncrono significa que el resultado de la llamada al servicio Web no está disponible de forma inmediata.

Cuando se realiza una llamada

```
x = stockService.getQuote("macr");
```

el objeto `x` no es el resultado de `getQuote` (es un objeto `PendingCall`). El resultado real sólo está disponible más tarde (normalmente unos segundos después) cuando finaliza la operación de servicio Web. El código de `ActionScript` se notifica mediante una llamada a la función callback `onResult`.

**Gestión del objeto `PendingCall`:** este objeto de función callback es un objeto `PendingCall` que se utiliza para gestionar los resultados y los errores del método de servicio Web llamado (véase [“Clase `PendingCall` \(sólo para Flash Professional\)” en la página 625](#)). Por ejemplo:

```
MyPendingCallObject = myWebServiceObject.myMethodName(param1, ..., paramN);
MyPendingCallObject.onResult = function(result)
{
    OutputField.text = result
}
MyPendingCallObject.onFault = function(fault)
{
    DebugField.text = fault.faultCode + "," + fault.faultstring;

    // añadir código para controlar los fallos, por ejemplo indicando
    // al usuario que el servidor no está disponible o que contacte
    // con el servicio de asistencia técnica
}
```

## WebService.getCall()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
getCall(var operationName)
```

### Parámetros

*operationName* Operación de servicio Web del objeto SOAPCall correspondiente que desea recuperar.

### Valor devuelto

Objeto SOAPCall.

### Descripción

Cuando se crea un objeto `WebService`, éste contiene los métodos correspondientes a las operaciones del URL WSDL que se pasen. En segundo plano, también se crea un objeto `SOAPCall` para cada operación en WSDL. `SOAPCall` es el descriptor de la operación y como tal contiene toda la información sobre dicha operación determinada; es decir, el aspecto del XML en la red, el estilo de la operación, etc. También permite controlar determinados comportamientos. Puede obtener el objeto `SOAPCall` de una operación determinada utilizando el método `getCall(operationName)`. Existe un solo objeto `SOAPCall` para cada operación, que comparten todas las llamadas activas a dicha operación. Una vez que dispone del objeto `SOAPCall`, puede cambiar el descriptor de operador utilizando la interfaz API de `SOAPCall`. Para más información, consulte [“Clase SOAPCall \(sólo para Flash Professional\)” en la página 634](#).

### Ejemplo

Para ver un ejemplo de cómo utilizar esta llamada, consulte [“Clase SOAPCall \(sólo para Flash Professional\)” en la página 634](#).

## WebService.onFault()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
MyWebServiceObject.onFault
```

## Parámetros

*fault* Versión del objeto de ActionScript descodificado del error con propiedades. Si la información sobre el error proviene de un servidor con el formato XML, el objeto SOAPFault será la versión de ActionScript descodificada de dicho XML.

El tipo de error que se devuelve a los métodos `webservice.onFault()` es un objeto `SOAPFault`. No lo crean directamente los desarrolladores, sino que se devuelve como resultado de un error. Este objeto es una asignación de ActionScript del tipo XML de fallo SOAP.

| Propiedad SOAPFault      | Descripción                                                                                                                                                                  |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>faultcode</code>   | Cadena; QName estándar más corto que describe el error.                                                                                                                      |
| <code>faultstring</code> | Cadena; descripción inteligible del error.                                                                                                                                   |
| <code>detail</code>      | Cadena; información específica de la aplicación asociada con el error, como por ejemplo un seguimiento de la pila u otra información que devuelve el motor del servicio Web. |
| <code>element</code>     | XML; objeto XML que representa la versión XML del fallo.                                                                                                                     |
| <code>faultactor</code>  | Cadena; origen del fallo, es opcional si no interviene ningún intermediario.                                                                                                 |

## Valor devuelto

Ninguno.

## Descripción

Función callback de `WebService`; Flash Player realiza una llamada a esta función cuando el nuevo método `webService(WSDLUrl)` no se ha podido ejecutar y ha devuelto un error. Esto puede suceder cuando el archivo WSDL no se puede analizar o no se puede encontrar. El parámetro `fault` es un objeto `SOAPFault` de ActionScript.

## Ejemplo

En el ejemplo siguiente se gestionan los errores devueltos en la creación del objeto `WebService`.

```
MyWebServiceObject.onFault = function(fault)
{
    // captura el fallo
    DebugOutputField.text = fault.faultstring;

    // añadir código para controlar los fallos, por ejemplo indicando
    // al usuario que el servidor no está disponible o que contacte
    // con el servicio de asistencia técnica
}
```



## WebService.onLoad()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*myService.onLoad*

### Parámetros

*wSDLDocument* Documento WSDL XML.

### Valor devuelto

Ninguno.

### Descripción

Función callback de WebService; Flash Player llama a esta función callback cuando el objeto WebService ha cargado y analizado correctamente su archivo WSDL. Se pueden invocar operaciones en una aplicación antes de que se produzca este evento, pero si se hace, se colocarán en una cola interna y no se transmitirán realmente hasta que se haya cargado el archivo WSDL.

### Ejemplo

En el ejemplo siguiente se especifica la URL de WSDL, se crea un objeto de servicio Web y se recibe el documento WSDL después de cargarlo.

```
// especificar la URL de WSDL
var wsdURI = "http://www.flash-db.com/services/ws/companyInfo.wsdl";

// crea un objeto de servicio Web
stockService = new WebService(wsdURI);

// recibe el documento WSDL después de cargarlo
stockService.onLoad = function(wsdDocument);
{
    // código que debe ejecutarse cuando la carga de WSDL finalice y el
    // objeto se haya creado
}
```

## WebService.myMethodName()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*callbackObj* = *myWebServiceObject.myMethodName(param1, ... paramN)*;

## Parámetros

Los parámetros necesarios dependen del método de servicio que se llame.

## Valor devuelto

*callbackObj* Objeto PendingCall con el que se puede asociar una función para gestionar los resultados y los errores de la invocación. Para más información, consulte [“Clase PendingCall \(sólo para Flash Professional\)” en la página 625](#).

La función callback que se invoca cuando la respuesta regresa del método WebService es PendingCall.onResult() u onFault(). Si identifica de forma exclusiva los objetos de función callback, puede gestionar varias funciones callback onResult, como en el ejemplo siguiente:

```
myWebService = new WebService("http://www.myCompany.com/myService.wsdl");
callback1 = myWebService.getWeather("02451");
callback1.onResult = function(result)
{
    //realizar alguna acción
}
callback2 = myWebService.getDetailedWeather("02451");
callback2.onResult = function(result)
{
    //realizar alguna acción
}
```

## Descripción

Para invocar una operación de servicio Web, invóquela como método directamente disponible en el servicio Web. Por ejemplo, si el servicio Web tiene el método getCompanyInfo(tickerSymbol), realice una llamada a:

```
myCallbackObject.myservice.getCompanyInfo(tickerSymbol);
```

Todas las invocaciones son asíncronas y devuelven un objeto de función callback o el tipo de objeto PendingCall.

## WebServiceConnector (sólo para Flash Professional)

El componente WebServiceConnector le permite acceder a métodos remotos presentados por un servidor mediante el protocolo estándar SOAP (Simple Object Access Protocol). Un servicio Web puede aceptar parámetros y devolver un resultado. Utilizando la herramienta de edición de Flash MX Professional 2004 y el componente WebServiceConnector, puede introspeccionar, acceder y vincular datos entre un servidor Web remoto y su aplicación Flash. Una sola instancia de componente WebServiceConnector puede utilizarse para hacer diversas llamadas a la misma operación. Tiene que utilizar una instancia distinta de WebServiceConnector para cada operación distinta a la que desee realizar una llamada.

Un servicio Web define los métodos (a veces denominados operaciones) disponibles para utilizar mediante un archivo XML con el formato WSDL (Web Service Description Language). El archivo WSDL especifica una lista de operaciones, parámetros y resultados (lo que se denomina esquema) que el servicio Web presenta.

Los archivos WSDL son accesibles a través de una URL. En Flash MX Professional 2004, puede ver el esquema de cualquier servicio Web introduciendo la URL de su archivo WSDL en el panel Servicios Web. Una vez ha identificado un archivo WSDL, el servicio Web está disponible para cualquier aplicación que cree.

Sólo el autor del archivo WSDL puede cambiar el archivo WSDL o el parámetro de funcionamiento. Cuando el autor cambia el archivo WSDL, se actualizan los esquemas params y results. Estos cambios sobrescribirán las modificaciones que el desarrollador haya realizado en el esquema. Para obtener un archivo WSDL actualizado, puede seleccionar Actualizar servicios Web en el menú del panel Servicio Web.

El componente WebServiceConnector y el componente XMLConnector implementan la interfaz API del componente RPC (Remote Procedure Call), un conjunto de métodos, propiedades y eventos que definen una forma sencilla de enviar parámetros a una fuente de datos externa y recibir parámetros.

Se puede utilizar una sola instancia de componente WebServiceConnector para hacer diversas llamadas a la misma operación. Sin embargo, tiene que utilizar una instancia de WebServiceConnector distinta para cada operación distinta que desee llamar.

Un desarrollador puede editar el esquema para personalizarlo a fin de utilizarlo en una aplicación (por ejemplo, para proporcionar otros valores de formato o de validación). Véase “Trabajo con esquemas en la ficha Esquema (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

## Utilización de WebServiceConnector (sólo para Flash Professional)

Puede utilizar WebServiceConnector para conectarse a un servicio Web y permitir que las propiedades puedan vincularse a propiedades de componentes de interfaz de usuario de la aplicación. Para conectarse a un servicio Web, debe introducir primero la URL del servicio Web. WebServiceConnector aparece en el escenario durante la edición de la aplicación, pero no aparece visualmente en la aplicación en tiempo de ejecución.

Puede introducir la URL de un servicio Web en el panel Inspector de componentes o el panel Servicios Web. Véase “El componente WebServiceConnector” en el apartado Utilización de Flash de la Ayuda.

Para más información sobre cómo trabajar con el componente WebServiceConnector, consulte “Vinculación de datos (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

## Parámetros de WebServiceConnector

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente WebServiceConnector en la ficha Parámetros del Inspector de componentes:

`multipleSimultaneousAllowed` (tipo booleano) indica si se pueden realizar varias llamadas a la vez; el valor predeterminado es `false`. Si su valor es `false`, la función `trigger()` no realizará una llamada si ya se está realizando otra llamada. Se emitirá un evento `status`, con el código `CallAlreadyInProgress`. Si su valor es `true`, la llamada se realizará.

`operation` (tipo cadena) es el nombre de una operación que aparece en el puerto SOAP en un archivo WSDL.

`suppressInvalidCalls` (tipo booleano) indica si debe suprimirse una llamada si los parámetros no son válidos; el valor predeterminado es `false`. Si su valor es `true`, la función `trigger()` no realizará una llamada si los parámetros databound no superan la validación. Se emitirá un evento de estado, con el código `InvalidParams`. Si su valor es `false`, la llamada se realizará con los datos no válidos, como sea necesario.

`WSDLURL` (tipo cadena) es la URL del archivo WSDL que define la operación de servicio Web. Cuando se define esta URL durante la edición, el archivo WSDL se recoge y se analiza de forma inmediata. Los parámetros resultantes y la información de resultados se pueden consultar en la ficha Esquema del panel Inspector de componentes. La descripción del servicio también se añade al panel Servicio Web. Por ejemplo, véase [www.xmethods.net/sd/2001/TemperatureService.wsdl](http://www.xmethods.net/sd/2001/TemperatureService.wsdl).

## Clase `WebServiceConnector` (sólo para Flash Professional)

Herencia RPC > `WebServiceConnector`

Nombre de clase de ActionScript `mx.data.components.WebServiceConnector`

### Resumen de propiedades de la clase `WebServiceConnector`

| Propiedad                                                    | Descripción                                                                                                                      |
|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| <code>WebServiceConnector.multipleSimultaneousAllowed</code> | Indica si se pueden realizar varias llamadas a la vez.                                                                           |
| <code>WebServiceConnector.multipleSimultaneousAllowed</code> | Indica el nombre de una operación que aparece en el puerto SOAP en un archivo WSDL.                                              |
| <code>WebServiceConnector.params</code>                      | Especifica los datos que se enviarán al servidor cuando se ejecute la siguiente operación <code>trigger()</code> .               |
| <code>WebServiceConnector.results</code>                     | Identifica los datos que se han recibido del servidor como resultado de la operación <code>trigger()</code> .                    |
| <code>WebServiceConnector.suppressInvalidCalls</code>        | Indica si debe suprimirse una llamada si hay parámetros no válidos.                                                              |
| <code>WebServiceConnector.timeout</code>                     | Especifica un período de tiempo (en segundos) en el que la conexión de servicio Web dará un error si no se devuelven resultados. |
| <code>WebServiceConnector.WSDLURL</code>                     | Especifica la URL del archivo WSDL que define la operación de servicio Web.                                                      |

## Resumen de métodos de la clase `WebServiceConnector`

| Método                                     | Descripción                                |
|--------------------------------------------|--------------------------------------------|
| <code>WebServiceConnector.trigger()</code> | Inicia una llamada a procedimiento remoto. |

## Resumen de eventos de la clase `WebServiceConnector`

| Evento                                  | Descripción                                                                                                                                                                        |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>WebServiceConnector.result</code> | Se difunde cuando se realiza de forma correcta una llamada a un servicio Web.                                                                                                      |
| <code>WebServiceConnector.send</code>   | Se difunde cuando la función <code>trigger()</code> está en curso, después de que se hayan recogido los datos pero antes de que se validen y se inicie la llamada al servicio Web. |
| <code>WebServiceConnector.status</code> | Se difunde cuando se inicia una llamada a un servicio Web, para informar al usuario del estado de la operación.                                                                    |

## `WebServiceConnector.multipleSimultaneousAllowed`

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.multipleSimultaneousAllowed;
```

### Descripción

Propiedad; indica si se pueden realizar varias llamadas al mismo tiempo. Si su valor es `false`, la función `trigger()` realizará una llamada si ya se está realizando otra llamada. Se emitirá un evento `status`, con el código `CallAlreadyInProgress`. Si su valor es `true`, la llamada se realizará.

Cuando se están realizando varias llamadas de forma simultánea, no hay ninguna garantía de que finalicen en el mismo orden en el que se han activado. Además, Flash Player puede limitar el número de operaciones de red simultáneas. Este límite varía para cada versión y plataforma.

### Ejemplo

En el ejemplo siguiente se permite realizar varias llamadas simultáneas a `myXmlUrl`:

```
myXmlUrl.multipleSimultaneousAllowed = true;
```

## WebServiceConnector.operation

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*componentInstance.operation;*

### Descripción

Propiedad; nombre de una operación que aparece en el puerto SOAP en un archivo WSDL.

## WebServiceConnector.params

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

*componentInstance.params;*

### Descripción

Propiedad; especifica datos que se enviarán al servidor cuando se ejecute la siguiente operación `trigger()`. La descripción WSDL del servicio Web determina el tipo de datos.

Cuando se realiza una llamada a métodos de servicio Web, el tipo de datos de la propiedad `params` debe ser un objeto o matriz de `ActionScript`, como se indica a continuación:

Si el servicio Web tiene formato de documento, el tipo de datos de `params` es un documento XML de algún tipo.

Si utiliza el inspector de propiedades o el panel Inspector de componentes para definir WSDLURL y `operation` durante la edición, puede proporcionar `params` como matriz de parámetros en el mismo orden que exige el método de servicio Web, por ejemplo `[1, "hello", 2432]`.

### Ejemplo

En el ejemplo siguiente se define la propiedad `params` de un componente de servicio Web denominado `wsc`:

```
wsc.params = [param_txt.text];
```

## WebServiceConnector.result

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.addEventListener("result", myListenerObject);
```

### Descripción

Evento; se difunde cuando una operación Remote Procedure Call finaliza correctamente.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena "result"
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente `WebServiceConnector`)

Puede recuperar el valor de resultado real mediante la propiedad `results`.

### Ejemplo

En el ejemplo siguiente se define una función `res` para el evento `result` y se asigna la función al controlador de eventos `addEventListener`:

```
var res = function (ev) {  
    trace(ev.target.results);  
};  
wsc.addEventListener("result", res);
```

## WebServiceConnector.results

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.results;
```

## Descripción

Propiedad; identifica los datos que se han recibido del servidor como resultado de una operación `trigger()`. Cada componente RPC define cómo se recogen estos datos y qué tipos son válidos. Estos datos aparecen cuando la operación RPC ha finalizado correctamente, como señala el evento `result`. Está disponible hasta que el componente se descarga o hasta la siguiente operación RPC.

Es posible que los datos que se devuelvan sean de gran tamaño. Puede gestionarse de dos maneras:

- Seleccione un clip de película, una línea de tiempo o una pantalla adecuada como elemento principal para el componente RPC. El almacenamiento del componente estará disponible para la eliminación de datos innecesarios cuando el elemento principal no esté presente.
- En ActionScript, puede asignar `null` a esta propiedad en cualquier momento.

## WebServiceConnector.send

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.addEventListener("send", myListenerObject);
```

### Descripción

Evento; se difunde durante el proceso de una operación `trigger()`, después de que se hayan recogido los datos de parámetros pero antes de que se validen y se inicie la operación Remote Procedure Call. Es una buena ubicación para incluir código que modifique los datos de parámetros antes de la llamada.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena "send"
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente `WebServiceConnector`)

Puede recuperar o modificar los valores de parámetros reales mediante la propiedad `params`.

### Ejemplo

En el ejemplo siguiente se define una función `sendFunction` para el evento `send` y se asigna la función al controlador de eventos `addEventListener`:

```
var sendFunction = function (sendEnv) {  
    sendEnv.target.params = [newParam_txt.text];  
};  
wsc.addEventListener("send", sendFunction);
```



## WebServiceConnector.status

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.addEventListener("status", myListenerObject);
```

### Descripción

Evento; se difunde cuando se inicia una operación Remote Procedure Call para informar al usuario del estado de la operación.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- **type:** cadena "status"
- **target:** referencia al objeto que ha emitido el evento (por ejemplo, un componente WebServiceConnector)
- **code:** cadena que da el nombre de la condición específica que se ha producido.
- **data:** objeto cuyo contenido depende del código.

A continuación se detallan los códigos y los datos asociados disponibles para el evento status:

| Código                | Datos                 | Descripción                                                                                                                                                                                                                                                                                                                                        |
|-----------------------|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StatusChange          | {callsInProgress:nnn} | Este evento se emite siempre que se inicia o finaliza una llamada de servicio Web. El elemento "nnn" es el número de llamadas que se están realizando en este momento.                                                                                                                                                                             |
| CallAlreadyInProgress | sin datos             | Este evento se emite si (a) se realiza una llamada a la función <code>trigger()</code> , (b) el valor de <code>multipleSimultaneousAllowed</code> es <code>false</code> y (c) ya se está realizando una llamada. Después de que se produzca este evento, el intento de llamada se considera realizado y no habrá ningún evento "result" ni "send". |
| InvalidParams         | sin datos             | Este evento se emite si la función <code>trigger()</code> ha detectado que la propiedad "params" no contenía datos válidos. Si el valor de la propiedad "suppressInvalidCalls" es <code>true</code> , el intento de llamada se considera realizado y no habrá ningún evento "result" ni "send".                                                    |

Éstos son los posibles fallos de los servicios Web:

| <b>faultcode</b>              | <b>faultstring</b>                                                                       | <b>detail</b>                                                                                 |
|-------------------------------|------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| Timeout                       | Se ha agotado el tiempo de espera al llamar al método xxx.                               | p                                                                                             |
| MustUnderstand                | No hay ninguna función callback para el encabezado xxx.                                  | p                                                                                             |
| Server.Connection             | Unable to connect to endpoint: xxx.                                                      | p                                                                                             |
| VersionMismatch               | Request implements version: xxx Response implements version yyy                          | p                                                                                             |
| Client.Disconnected           | Could not load WSDL.                                                                     | Unable to load WSDL, if currently online, please verify the URI and/or format of the WSDL xxx |
| Server                        | Faulty WSDL format.                                                                      | Definitions must be the first element in a WSDL document                                      |
| Server.NoServicesInWSDL       | Could not load WSDL.                                                                     | No elements found in WSDL at xxx                                                              |
| WSDL.UnrecognizedNamespace    | The WSDL parser had no registered document for the namespace xxxx.                       | p                                                                                             |
| WSDL.UnrecognizedBindingName  | The WSDL parser couldn't find a binding named xxx in namespace yyy.                      | p                                                                                             |
| WSDL.UnrecognizedPortTypeName | The WSDL parser couldn't find a portType named xxx in namespace yyy.                     | p                                                                                             |
| WSDL.UnrecognizedMessageName  | El analizador WSDL no ha podido encontrar un mensaje denominado xxx en el namespace yyy. | p                                                                                             |
| WSDL.BadElement               | El elemento xxx no se puede resolver.                                                    | p                                                                                             |
| WSDL.BadType                  | El tipo xxx no se puede resolver.                                                        | p                                                                                             |
| Client.NoSuchMethod           | No se ha podido encontrar el método 'xxx' en el servicio.                                | p                                                                                             |

| faultcode            | faultstring                                                                                             | detail                                                                                                                   |
|----------------------|---------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| yyy.                 | yyy - se han notificado errores del servidor; esto depende del servidor con el que se esté comunicando. | p                                                                                                                        |
| No.WSDLURL.Defined   | El componente WebServiceConnector no tenía ninguna URL de WSDL definida.                                | p                                                                                                                        |
| Unknown.Call.Failure | La invocación a WebService no se ha podido realizar por motivos desconocidos.                           | p                                                                                                                        |
| Client.Disconnected  | No se ha podido cargar el esquema importado.                                                            | No se puede cargar el esquema; si está conectado en este momento, verifique la URI y/o el formato del esquema en (XXXX)) |

### Ejemplo

En el ejemplo siguiente se define una función `statusFunction` para el evento `status` y se asigna la función al controlador de eventos `addEventListener`:

```
var statusFunction = function (stat) {
    trace(stat.code);
    trace(stat.data.faultcode);
    trace(stat.data.faultstring);
};
wsc.addEventListener("status", statusFunction);
```

## WebServiceConnector.suppressInvalidCalls

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.suppressInvalidCalls;
```

### Descripción

Propiedad; indica si debe suprimirse una llamada en caso de que los parámetros no sean válidos. Si su valor es `true`, la función `trigger()` no realizará una llamada en caso de que los parámetros vinculados no superen la validación. Se emitirá un evento "status", con el código `InvalidParams`. Si su valor es `false`, la llamada se realizará con los datos no válidos, como sea necesario.

## WebServiceConnector.timeout

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.timeout;
```

### Descripción

Propiedad; período de tiempo (en segundos) en el que la conexión de servicio Web dará un error si no se devuelven resultados. Se emite un evento *status* (heredado del componente RPC) con el código *WebServiceFault*, código de fallo *Timeout*.

## WebServiceConnector.trigger()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.trigger();
```

### Descripción

Método; inicia una llamada a un servicio Web. Cada servicio Web define exactamente lo que esto implica. Si la operación es correcta, el resultado de la operación aparecerá en la propiedad *results* del servicio Web.

El método *trigger()* sigue estos pasos:

- 1 Si hay datos vinculados a la propiedad *params*, el método ejecuta todas las vinculaciones para garantizar que haya datos actualizados disponibles. Esto también provoca una validación de datos.
- 2 Si los datos no son válidos y el valor de *suppressInvalidCalls* es *true*, la operación se interrumpe.
- 3 Si la operación continúa, se emite el evento *send*.
- 4 La llamada remota real se inicia a través del método de conexión indicado (por ejemplo, HTTP).

## WebServiceConnector.WSDLURL

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

## Sintaxis

```
componentInstance.WSDLURL;
```

## Descripción

Propiedad; URL del archivo WSDL que define la operación de servicio Web. Cuando se define esta URL durante la edición, el archivo WSDL se recoge y se analiza de forma inmediata. Los parámetros resultantes y los resultados aparecen en la ficha Esquema del panel Inspector de componentes. La descripción del servicio también aparece en el panel Servicio Web.

## Componente Window

El componente Window muestra el contenido de un clip de película en una ventana provista de una barra de título, un borde y un botón de cierre opcional.

El componente Window puede ser modal o amodal. Una ventana modal impide que las entradas realizadas con el ratón o el teclado se apliquen a otros componentes situados fuera de la ventana. También admite la función de arrastre, es decir, el usuario puede hacer clic en la barra de título y arrastrar la ventana y su contenido a otro lugar. El arrastre de los bordes no cambia el tamaño de la ventana.

La vista previa dinámica de cada instancia de Window refleja los cambios realizados en todos los parámetros (excepto en `contentPath`) en el inspector de propiedades o en el panel Inspector de componentes durante la edición.

Cuando se añade el componente Window a una aplicación, es posible utilizar el panel Accesibilidad para que los lectores de pantalla puedan acceder al mismo. En primer lugar, debe añadir la línea de código siguiente para activar la accesibilidad:

```
mx.accessibility.WindowAccImpl.enableAccessibility();
```

La accesibilidad de un componente sólo se activa una vez, sea cual sea su número de instancias. Para más información, consulte “Creación de contenido accesible” en el apartado Utilización de Flash de la Ayuda.

## Utilización del componente Window

Puede utilizar una ventana en una aplicación siempre que desee ofrecer al usuario información o una opción que tenga prioridad sobre cualquier otro elemento de la aplicación. Por ejemplo, puede necesitar que un usuario rellene los datos de una ventana de conexión o de una ventana en la que se cambia una contraseña y se confirma una nueva.

Hay varias formas de añadir una ventana a una aplicación. Puede arrastrar un componente Window desde el panel Componentes al escenario. También puede llamar a `createClassObject()` (véase [UIObject.createClassObject\(\)](#)) para añadir una ventana a una aplicación. La tercera forma de añadir ventanas a una aplicación consiste en utilizar la [Clase PopUpManager](#). El uso de `PopUpManager` permite crear ventanas modales que se superponen a otros objetos del escenario. Para más información, consulte [Clase Window](#).

Si se utiliza `PopUpManager` para añadir un componente Window a un documento, la instancia Window tendrá su propio `FocusManager`, distinto al resto del documento. Si no utiliza `PopUpManager`, el contenido de la ventana comparte el orden de selección con el resto de los componentes del documento. Para más información sobre el control de la selección, consulte “[Creación de un desplazamiento personalizado de la selección](#)” en la [página 27](#) o “[Clase FocusManager](#)” en la [página 292](#).

## Parámetros del componente Window

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente Window en el inspector de propiedades o en el panel Inspector de componentes:

**contentPath** especifica el contenido de la ventana. Puede ser el identificador de vinculación del clip de película o el nombre de símbolo de una pantalla, formulario o diapositiva que incluya el contenido de la ventana. También puede ser una URL absoluta o relativa a un archivo SWF o JPG que se carga en la ventana. El valor predeterminado es "". El contenido cargado se recorta para que quepa en la ventana.

**title** indica el título de la ventana.

**closeButton** indica si la ventana va a mostrar un botón de cierre (true) o no (false). Al hacer clic en el botón de cierre se difunde un evento `click`, pero no se cierra la ventana. Debe escribir un controlador que invoque a `Window.deletePopUp()` para cerrar la ventana explícitamente. Para más información sobre el evento `click`, consulte [Window.click](#).

Puede escribir código `ActionScript` para controlar éstas y otras opciones adicionales para los componentes de Window utilizando sus propiedades, métodos y eventos. Para más información, consulte [Clase Window](#).

## Creación de aplicaciones con el componente Window

El procedimiento siguiente explica cómo añadir un componente Window a una aplicación. En este ejemplo se solicita al usuario que cambie su contraseña y confirme la nueva.

**Para crear una aplicación con el componente Window, siga este procedimiento:**

- 1 Cree un clip de película nuevo que contenga campos para insertar y confirmar una contraseña, además de botones Aceptar y Cancelar. Asigne al clip de película el nombre **PasswordForm**.  
Es el contenido que llenará la ventana. El contenido debe alinearse a 0,0 porque se encuentra en la esquina superior izquierda de la ventana.
- 2 En la biblioteca, seleccione el clip de película PasswordForm y elija Vinculación en el menú Opciones.
- 3 Seleccione Exportar para `ActionScript`.  
El identificador de vínculo **PasswordForm** se introduce de forma automática en el cuadro Identificador.
- 4 Introduzca **mx.core.View** en el campo de clase y haga clic en Aceptar.
- 5 Arrastre un componente Window desde el panel Componentes hasta el escenario y elimine el componente del escenario. De esta manera, añadirá el componente a la biblioteca.
- 6 En la biblioteca, seleccione el archivo SWC de Window y elija Vinculación en el menú Opciones.
- 7 Marque Exportar para `ActionScript`, si no está marcado.
- 8 Arrastre un componente de botón del panel Componentes al escenario y en el panel Inspector de propiedades, asígnele el nombre de instancia **button**.

- 9 Abra el panel Acciones e introduzca el controlador de acciones del ratón siguiente en el fotograma 1:

```
buttonListener = new Object();
buttonListener.click = function(){
    myWindow = mx.managers.PopupManager.createPopup(_root,
        mx.containers.Window, true, { title:"Change Password",
        contentPath:"PasswordForm"});
    myWindow.setSize(240.110);
}
button.addEventListener("click", buttonListener);
```

Este controlador llama a `PopupManager.createPopup()` para crear una instancia del componente `Window` con la barra de título “Change Password” que muestre el contenido del clip de película `PasswordForm` cuando se hace clic en el botón. Para cerrar la ventana cuando se haga clic en el botón Aceptar o Cancelar, tendrá que escribir otro controlador.

## Personalización del componente `Window`

El componente `Window` puede transformarse horizontal o verticalmente durante la edición y en tiempo de ejecución. Durante la edición, seleccione el componente en el escenario y utilice la herramienta Transformación libre o cualquiera de los comandos Modificar > Transformar. En tiempo de ejecución, utilice `UIObject.setSize()`.

Cuando se cambia el tamaño de la ventana, no se modifica el tamaño del botón de cierre ni del título. El título se alinea a la izquierda y la barra de cierre a la derecha.

## Utilización de estilos con el componente `Window`

La declaración de estilos de la barra de título de un componente `Window` se indica mediante la propiedad `Window.titleStyleDeclaration`.

Un componente `Window` admite los siguientes estilos de Halo:

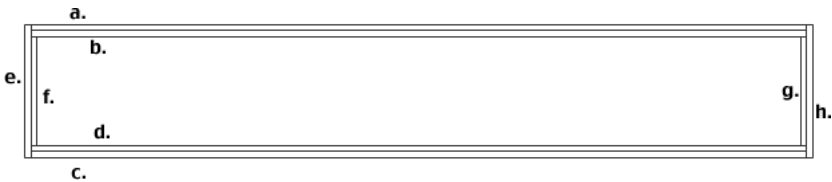
| Estilo                   | Descripción                                                                                |
|--------------------------|--------------------------------------------------------------------------------------------|
| <code>borderStyle</code> | Borde del componente; “none”, “inset”, “outset” o “solid”. Este estilo no hereda su valor. |

## Utilización de aspectos con el componente Window

El componente Window utiliza la clase `RectBorder` que se vale de la función de dibujo API de ActionScript para dibujar los bordes. Puede utilizar el método `setStyle()` (véase [UIObject.setStyle\(\)](#)) para cambiar las propiedades de estilo `RectBorder` siguientes:

| Estilos <code>RectBorder</code> | Letra |
|---------------------------------|-------|
| <code>borderColor</code>        | a     |
| <code>highlightColor</code>     | b     |
| <code>borderColor</code>        | c     |
| <code>shadowColor</code>        | d     |
| <code>borderCapColor</code>     | e     |
| <code>shadowCapColor</code>     | f     |
| <code>shadowCapColor</code>     | g     |
| <code>borderCapColor</code>     | h     |

Las propiedades de estilo definen las siguientes posiciones en el borde:



Si utiliza `UIObject.createClassObject()` o `PopUpManager.createPopUp()` para crear una instancia de `Window` de forma dinámica (en tiempo de ejecución), podrá aplicar también el aspecto de forma dinámica. Para asignar aspectos a un componente en tiempo de ejecución, defina las propiedades de aspecto en el parámetro `initObject` que pasa al método `createClassObject()`. Estas propiedades de aspecto definen los nombres de los símbolos que se utilizarán como estados del botón, ya sea con o sin icono. Para más información, consulte [UIObject.createClassObject\(\)](#) y [PopUpManager.createPopUp\(\)](#).

Un componente `Window` utiliza las propiedades de aspecto siguientes:

| Propiedad                        | Descripción                                                                                          |
|----------------------------------|------------------------------------------------------------------------------------------------------|
| <code>skinTitleBackground</code> | Barra de título. El valor predeterminado es <code>TitleBackground</code> .                           |
| <code>skinCloseUp</code>         | Botón de cierre. El valor predeterminado es <code>CloseButtonUp</code> .                             |
| <code>skinCloseDown</code>       | Botón de cierre en estado Presionado. El valor predeterminado es <code>CloseButtonDown</code> .      |
| <code>skinCloseDisabled</code>   | Botón de cierre en estado Desactivado. El valor predeterminado es <code>CloseButtonDisabled</code> . |
| <code>skinCloseOver</code>       | Botón de cierre en estado Sobre. El valor predeterminado es <code>CloseButtonOver</code> .           |



## Clase Window

**Herencia** UIObject > UIComponent > View > ScrollView > Window

**Nombre de clase de ActionScript** mx.containers.Window

Las propiedades de la clase Window permiten definir el título, añadir un botón de cierre y definir el contenido que deberá aparecer en tiempo de ejecución. Si una propiedad de la clase Window se define con ActionScript, sustituye el parámetro del mismo nombre definido en el inspector de propiedades o en el panel Inspector de componentes.

La mejor manera de crear una instancia de Window es llamando a `PopupManager.createPopup()`. Este método crea una ventana que puede ser modal (que solapa y desactiva los objetos existentes de la aplicación) o amodal. Por ejemplo, el código siguiente crea una instancia Window modal (el último parámetro indica modalidad):

```
var newWindow = PopupManager.createPopup(this, Window, true);
```

La modalidad se simula creando una ventana transparente grande por debajo del componente Window. Debido a la forma en la que se representan las ventanas transparentes, es posible que advierta una ligera atenuación de los objetos situados debajo. Para definir una transparencia efectiva, cambie el valor de `_global.style.modalTransparency` de 0 (completamente transparente) a 100 (opaco). Si hace que la ventana sea parcialmente transparente, también podrá definir el color de la ventana cambiando el aspecto Modal del tema predeterminado.

Si utiliza `PopupManager.createPopup()` para crear una ventana modal, deberá llamar a `Window.deletePopup()` para eliminarlo y eliminar también la ventana transparente. Por ejemplo, si utiliza `closeButton` en la ventana, deberá escribir el código siguiente:

```
obj.click = function(evt){
    this.deletePopup();
}
window.addEventListener("click", obj);
```

**Nota:** la ejecución del código no se interrumpe cuando se crea una ventana modal. En otros entornos, como Microsoft Windows, si crea una ventana modal, las líneas de código siguientes a la creación de la ventana no se ejecutan hasta que ésta se cierra. En Flash, las líneas de código siguen ejecutándose después de crear la ventana y antes de que se cierre.

Cada clase de componente tiene una propiedad `version` que es una propiedad de clase. Las propiedades de clase sólo están disponibles en la clase de que se trate. La propiedad `version` devuelve una cadena que indica la versión del componente. Para acceder a la propiedad `version`, utilice el código siguiente:

```
trace(mx.containers.Window.version);
```

**Nota:** el código siguiente devuelve undefined: `trace(myWindowInstance.version);`.

## Resumen de métodos de la clase Window

| Método                            | Descripción                                                                          |
|-----------------------------------|--------------------------------------------------------------------------------------|
| <code>Window.deletePopup()</code> | Elimina una instancia de Window creada por <code>PopupManager.createPopup()</code> . |

Hereda todos los métodos de `UIObject`, `UIComponent` y `View`.

## Resumen de propiedades de la clase Window

| Propiedad                                 | Descripción                                                                                                |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------|
| <code>Window.closeButton</code>           | Indica si la barra de título incluye un botón de cierre ( <code>true</code> ) o no ( <code>false</code> ). |
| <code>Window.content</code>               | Referencia al contenido especificado en la propiedad <code>contentPath</code> .                            |
| <code>Window.contentPath</code>           | Ruta de acceso al contenido que se muestra en la ventana.                                                  |
| <code>Window.title</code>                 | Texto que aparece en la barra de título.                                                                   |
| <code>Window.titleStyleDeclaration</code> | Declaración de estilos que asigna formato al texto de la barra de título.                                  |

Hereda todas las propiedades de [UIObject](#), [UIComponent](#) y [ScrollView](#).

## Resumen de eventos de la clase Window

| Evento                               | Descripción                                                                 |
|--------------------------------------|-----------------------------------------------------------------------------|
| <code>Window.click</code>            | Se difunde cuando se suelta el botón de cerrar.                             |
| <code>Window.mouseDownOutside</code> | Se difunde cuando se presiona el botón del ratón fuera de la ventana modal. |

Hereda todos los eventos de [UIObject](#), [UIComponent](#), [View](#) y [ScrollView](#).

## Window.click

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(click){  
    ...  
}
```

Sintaxis 2:

```
listenerObject = new Object();  
listenerObject.click = function(eventObject){  
    ...  
}  
windowInstance.addEventListener("click", listenerObject)
```

## Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta) sobre el botón de cierre.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `Window`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myWindow` del componente `Window`, envía “\_level0.myWindow” al panel Salida:

```
on(click){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*windowInstance*) distribuye un evento (en este caso, `click`), y éste se controla mediante una función, denominada también *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto `Event` (*eventObject*) al método del objeto detector. El objeto `Event` tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte “Objetos Event” en la página 602.

## Ejemplo

En el ejemplo siguiente se crea una ventana modal y se define un controlador de acciones del ratón que elimina la ventana. Se debe añadir un componente `Window` al escenario y, a continuación, eliminarlo para añadir el componente a la biblioteca del documento y añadir el código siguiente al fotograma 1:

```
import mx.managers.PopUpManager
import mx.containers.Window
var myTW = PopUpManager.createPopUp(_root, Window, true, {closeButton: true,
    title:"My Window"});
windowListener = new Object();
windowListener.click = function(evt){
    _root.myTW.deletePopUp();
}
myTW.addEventListener("click", windowListener);
```

## Véase también

`UIEventDispatcher.addEventListener()`, `Window.closeButton`

## Window.closeButton

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*windowInstance.closeButton*

### Descripción

Propiedad; valor booleano que indica si la barra de título debe llevar un botón de cierre (*true*) o no (*false*). La propiedad debe definirse en el parámetro *initObject* del método [PopUpManager.createPopUp\(\)](#). El valor predeterminado es *false*.

### Ejemplo

El código siguiente crea una ventana en la que se muestra el contenido del clip de película “LoginForm” y que tiene un botón de cierre en la barra de título:

```
var myTW = PopUpManager.createPopUp(_root, Window, true,
    {contentPath:"LoginForm", closeButton:true});
```

### Véase también

[Window.click](#), [PopUpManager.createPopUp\(\)](#)

## Window.content

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*windowInstance.content*

### Descripción

Propiedad; referencia al contenido (clip de película raíz) de la ventana. Esta propiedad devuelve un objeto *MovieClip*. Cuando se asocia un símbolo de la biblioteca, el valor predeterminado es una instancia del símbolo asociado. Si se carga el contenido de una URL, el valor predeterminado está sin definir hasta que comienza la operación de carga.

### Ejemplo

Define el valor de la propiedad de texto desde el interior del contenido del componente *Window*:

```
loginForm.content.password.text = "secret";
```

## Window.contentPath

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*windowInstance.contentPath*

### Descripción

Propiedad; define el nombre del contenido que ha de aparecer en la ventana. Este valor puede ser el identificador de vinculación de un clip de película de la biblioteca o la URL absoluta o relativa de un archivo SWF o JPG que se carga. El valor predeterminado es "" (cadena vacía).

### Ejemplo

El código siguiente crea una instancia de Window que muestra el clip de película con el identificador de vinculación "LoginForm":

```
var myTW = PopUpManager.createPopUp(_root, Window, true,  
    {contentPath:"LoginForm"});
```

## Window.deletePopUp()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*windowInstance.deletePopUp()*;

### Parámetros

Ninguno.

### Valor devuelto

Ninguno.

### Descripción

Método; elimina la instancia de Window y quita el estado modal. Este método sólo se puede llamar para instancias de Window creadas por medio de [PopUpManager.createPopUp\(\)](#).

## Ejemplo

El código siguiente crea una ventana modal y luego genera un detector que elimina la ventana cuando se hace clic en el botón de cierre:

```
var myTW = PopUpManager.createPopUp(_root, Window, true);
twListener = new Object();
twListener.click = function(){
    myTW.deletePopUp();
}
myTW.addEventListener("click", twListener);
```

## Window.mouseDownOutside

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

Sintaxis 1:

```
on(mouseDownOutside){
    ...
}
```

Sintaxis 2:

```
listenerObject = new Object();
listenerObject.mouseDownOutside = function(eventObject){
    ...
}
windowInstance.addEventListener("mouseDownOutside", listenerObject)
```

### Descripción

Evento; se difunde a todos los detectores registrados cuando se hace clic con el ratón (se suelta) fuera de la ventana modal. Este evento se utiliza muy raramente, pero puede servir para descartar una ventana si el usuario trata de interactuar con algún elemento situado fuera de la misma.

El primer ejemplo de sintaxis utiliza un controlador `on()` que debe asociarse directamente con una instancia del componente `Window`. La palabra clave `this`, utilizada en un controlador `on()` asociado con un componente, hace referencia a la instancia del componente. Por ejemplo, el código siguiente, asociado con la instancia `myWindowComponent` del componente `Window`, envía “\_level0.myWindowComponent” al panel Salida:

```
on(click){
    trace(this);
}
```

El segundo ejemplo de sintaxis utiliza un modelo de eventos distribuidor/detector. Una instancia de componente (*windowInstance*) distribuye un evento (en este caso, `mouseDownOutside`), y éste se controla mediante una función, también denominada *controlador*, asociada con el objeto detector (*listenerObject*) que crea el usuario. Debe definirse un método con el mismo nombre que el evento del objeto detector; se llama al método cuando se activa el evento. Cuando se activa el evento, éste pasa automáticamente un objeto Event (*eventObject*) al método del objeto detector. El objeto Event tiene un conjunto de propiedades que contiene información acerca del mismo. Estas propiedades sirven para escribir el código que gestiona el evento. Finalmente, se llama al método `UIEventDispatcher.addEventListener()` de la instancia del componente que difunde el evento para registrar el detector con la instancia. Cuando la instancia distribuya el evento, se llamará al detector.

Para más información acerca de los objetos de eventos, consulte [“Objetos Event” en la página 602](#).

### Ejemplo

En el ejemplo siguiente se crea una instancia de Window y se define un controlador `mouseDownOutside` que llama al método `beep()` cuando el usuario hace clic fuera de la ventana:

```
var myTW = PopUpManager.createPopUp(_root, Window, true, undefined, true);
// crear un detector
twListener = new Object();
twListener.mouseDownOutside = function()
{
    beep(); // emitir un sonido si el usuario hace clic fuera
}
myTW.addEventListener("mouseDownOutside", twListener);
```

### Véase también

`UIEventDispatcher.addEventListener()`

## Window.title

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

*windowInstance.title*

### Descripción

Propiedad; cadena que indica el texto de la barra de título. El valor predeterminado es "" (cadena vacía).

### Ejemplo

El código siguiente establece el título de la ventana en “Hello World”:

```
myTW.title = "Hello World";
```

## Window.titleStyleDeclaration

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX 2004.

### Sintaxis

```
windowInstance.titleStyleDeclaration
```

### Descripción

Propiedad; cadena que indica la declaración de estilos que asigna formato a la barra de título de una ventana. El valor predeterminado es undefined, que significa texto blanco, negrita.

### Ejemplo

El código siguiente crea una ventana que muestra el contenido del clip de película con el identificador de vinculación “ChangePassword” y utiliza la titleStyleDeclaration “MyTWStyles”:

```
var myTW = PopUpManager.createPopUp(_root, Window, true,  
    {contentPath:"LoginForm",  
      titleStyleDeclaration:"MyTWStyles"});
```

Para más información sobre estilos, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

## Componente XMLConnector (sólo para Flash Professional)

El componente XMLConnector es un componente de Flash MX 2004 v2 cuyo objetivo es leer o escribir documentos XML mediante las operaciones HTTP `get` o `post`. Actúa como conector entre otros componentes y fuentes de datos XML externas. XMLConnector se comunica con los componentes de la aplicación utilizando funciones de vinculación de datos del entorno de edición de Flash MX Professional 2004 o código ActionScript. El componente XMLConnector tiene propiedades, métodos y eventos pero no aparece en pantalla en tiempo de ejecución.

El componente XMLConnector y el componente WebServiceConnector implementan la interfaz API del componente RPC (Remote Procedure Call), un conjunto de métodos, propiedades y eventos que define una forma sencilla de enviar parámetros a una fuente de datos externa y recibirlos.

### Utilización del componente XMLConnector (sólo para Flash Professional)

El componente XMLConnector proporciona a su aplicación acceso a cualquier fuente de datos externa que devuelve o recibe XML a través de HTTP. La forma más fácil de conectarse con una fuente de datos XML externa y utilizar los parámetros y resultados de esa fuente de datos para su aplicación es especificar un *esquema*, la estructura del documento XML que identifica los elementos de datos del documento al que quiere vincularse.

El esquema aparece en la ficha Esquema del panel Inspector de componentes. El esquema identifica los campos del documento XML que se pueden vincular a las propiedades del componente de interfaz de usuario de la aplicación. Puede crear el esquema manualmente a través del panel Inspector de componentes o utilizar el entorno de edición para crear uno de forma automática.



**Nota:** el entorno de edición aceptará como modelo para el esquema una copia del documento XML externo al que se va a conectar. Si está familiarizado con la creación de scripts XML, puede crear un archivo XML de muestra que puede utilizarse para generar un esquema.

Aunque el componente XMLConnector tiene propiedades y eventos (como otros componentes), no aparece en pantalla en tiempo de ejecución. Para más información sobre cómo trabajar con el componente XMLConnector, consulte “El componente XMLConnector (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

## Parámetros del componente XMLConnector

A continuación se indican los parámetros de edición que se pueden definir para cada instancia del componente XMLConnector en la ficha Parámetros del Inspector de componentes:

`direction` (Enumeración) indica si los datos se están enviando o recibiendo o ambas operaciones.

`ignoreWhite` (Tipo booleano) cuando el valor es true, ignora los espacios en blanco al recuperar el código XML.

`multipleSimultaneousAllowed` (Tipo booleano) indica si se pueden realizar varias llamadas a la vez; el valor predeterminado es false.

`suppressInvalidCalls` (Tipo booleano) indica si debe suprimirse una llamada si los parámetros no son válidos; el valor predeterminado es false.

`URL` (Tipo cadena) es la URL del documento XML externo que se utiliza en las operaciones HTTP.

## Clase XMLConnector (sólo para Flash Professional)

**Herencia**   RPC > XMLConnector

**Nombre de clase de ActionScript**   mx.data.components.XMLConnector

## Resumen de propiedades de la clase XMLConnector

| Propiedad                                             | Descripción                                                                                                        |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>XMLConnector.direction</code>                   | Indica si los datos se están enviando o recibiendo o ambas operaciones.                                            |
| <code>XMLConnector.multipleSimultaneousAllowed</code> | Indica si se pueden realizar varias llamadas a la vez.                                                             |
| <code>XMLConnector.params</code>                      | Especifica los datos que se enviarán al servidor cuando se ejecute la siguiente operación <code>trigger()</code> . |
| <code>XMLConnector.results</code>                     | Identifica los datos que se han recibido del servidor como resultado de la operación <code>trigger()</code> .      |
| <code>XMLConnector.suppressInvalidCalls</code>        | Indica si debe suprimirse una llamada si hay parámetros no válidos.                                                |
| <code>XMLConnector.URL</code>                         | URL que utiliza el componente en operaciones HTTP.                                                                 |

## Resumen de métodos de la clase XMLConnector

| Método                              | Descripción                                |
|-------------------------------------|--------------------------------------------|
| <code>XMLConnector.trigger()</code> | Inicia una llamada a procedimiento remoto. |

## Resumen de eventos de la clase XMLConnector

| Evento                           | Descripción                                                                                                                                                                          |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>XMLConnector.result</code> | Se difunde cuando una operación Remote Procedure Call se realiza correctamente.                                                                                                      |
| <code>XMLConnector.send</code>   | Se difunde cuando hay una función <code>trigger()</code> en marcha, después de que los datos se hayan recopilado pero antes de que queden invalidados y se inicie la llamada remota. |
| <code>XMLConnector.status</code> | Se difunde cuando se inicia una llamada a procedimiento remoto para informar al usuario del estado de la operación.                                                                  |

## XMLConnector.direction

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.direction;
```

### Descripción

Propiedad; indica si los datos se están enviando o recibiendo o ambas operaciones. Los valores son los siguientes:

- `send` Los datos XML de la propiedad `params` se envían a través de HTTP POST a la URL del documento XML. Los datos que se devuelven se ignoran. No se establece ningún valor para la propiedad `results` y no hay ningún evento `result`.

**Nota:** las propiedades `params` y `results` y el evento `result` se heredan de la interfaz API del componente RPC.

- `receive` No se envían datos de la propiedad `params` a la URL. Se accede a la URL del documento XML a través de HTTP GET y se espera recibir datos XML válidos de la URL.
- `send/receive` Los datos de la propiedad `params` se envían a la URL y se espera recibir datos XML válidos de la URL.

## Ejemplo

En el ejemplo siguiente se establece el valor `receive` para la dirección del documento `mysettings.xml`:

```
myXMLConnector.direction = "receive";  
myXMLConnector.URL = "mysettings.xml";  
myXMLConnector.trigger();
```

## XMLConnector.multipleSimultaneousAllowed

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.multipleSimultaneousAllowed;
```

### Descripción

Propiedad; indica si se pueden realizar varias llamadas al mismo tiempo. Si su valor es `false`, la función `trigger()` realizará una llamada si ya se está realizando otra llamada. Se emitirá un evento `status`, con el código `CallAlreadyInProgress`. Si su valor es `true`, la llamada se realizará.

Cuando se están realizando varias llamadas de forma simultánea, no hay ninguna garantía de que finalicen en el mismo orden en el que se han activado. Además, Flash Player puede limitar el número de operaciones de red simultáneas. Este límite varía para cada versión y plataforma.

## XMLConnector.params

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.params;
```

### Descripción

Propiedad; especifica datos que se enviarán al servidor cuando se ejecute la siguiente operación `trigger()`. Cada componente RPC define cómo se utilizan estos datos y qué tipos son válidos.

## Ejemplo

En el ejemplo siguiente se definen los parámetros `name` y `city` de `myXMLConnector`:

```
myXMLConnector.params = new XML("<mydoc><name>Bob</name><city>Oakland</city></mydoc>");
```

## XMLConnector.result

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.addEventListener("result", myListenerObject);
```

### Descripción

Evento; se difunde cuando una operación Remote Procedure Call se realiza correctamente.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena "result"
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente `WebServiceConnector`)

Puede recuperar el valor de resultado real mediante la propiedad `results`.

### Ejemplo

En el ejemplo siguiente se define una función `res` para el evento `result` y se asigna la función al controlador de eventos `addEventListener`:

```
var res = function (ev) {  
    trace(ev.target.results);  
};  
xcon.addEventListener("result", res);
```

## XMLConnector.results

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.results;
```

### Descripción

Propiedad; identifica los datos que se han recibido del servidor como resultado de una operación `trigger()`. Cada componente RPC define cómo se recogen estos datos y qué tipos son válidos. Estos datos aparecen cuando la operación RPC ha finalizado correctamente, como señala el evento `result`. Está disponible hasta que el componente se descarga o hasta la siguiente operación RPC.

Es posible que los datos que se devuelvan sean de gran tamaño. Puede gestionarse de dos maneras:

- Seleccione un clip de película, una línea de tiempo o una pantalla adecuada como elemento principal para el componente RPC. El almacenamiento del componente estará disponible para la eliminación de datos innecesarios cuando el elemento principal no esté presente.
- En ActionScript, puede asignar null a esta propiedad en cualquier momento.

### Ejemplo

En el ejemplo siguiente se rastrea la propiedad `results` de `myXMLConnector`:

```
trace(myXMLConnector.results);
```

## XMLConnector.send

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.addEventListener("send", myListenerObject);
```

### Descripción

Evento; se difunde durante el proceso de una operación `trigger()`, después de que se hayan recogido los datos de parámetros pero antes de que se validen y se inicie la operación Remote Procedure Call. Es una buena ubicación para incluir código que modifique los datos de parámetros antes de la llamada.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- `type`: cadena "send"
- `target`: referencia al objeto que ha emitido el evento (por ejemplo, un componente `WebServiceConnector`)

Puede recuperar o modificar los valores de parámetros reales mediante la propiedad `params`.

### Ejemplo

En el ejemplo siguiente se define una función `sendFunction` para el evento `send` y se asigna la función al controlador de eventos `addEventListener`:

```
var sendFunction = function (sendEnv) {  
    sendEnv.target.params = [newParam_txt.text];  
};  
xcon.addEventListener("send", sendFunction);
```

## XMLConnector.status

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.addEventListener("status", myListenerObject);
```

### Descripción

Evento; se difunde cuando se inicia una operación Remote Procedure Call para informar al usuario del estado de la operación.

El parámetro para el controlador de eventos es un objeto con los campos siguientes:

- **type:** cadena "status".
- **target:** referencia al objeto que ha emitido el evento (por ejemplo, un componente WebServiceConnector).
- **code:** cadena que da el nombre de la condición específica que se ha producido.
- **data:** objeto cuyo contenido depende del código.

Se establece el valor Fault en el campo de código del evento Status si se producen problemas con la llamada, de la manera siguiente:

| Código | Datos                                                                                                       | Descripción                                                                                                                                                                                                                                                |
|--------|-------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Fault  | {faultcode: código,<br>faultstring: cadena,<br>detail: detalle,<br>element: elemento,<br>faultactor: actor} | Este evento se emite si se producen otros problemas durante el proceso de la llamada. Los datos son un objeto SOAPFault. Después de que se produzca este evento, el intento de llamada se considera realizado y no habrá ningún evento "result" ni "send". |

A continuación se indican los fallos que se pueden producir con el evento Status:

| FaultCode                | FaultString                               | Notas                                                                                                                                                                        |
|--------------------------|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XMLConnector.Not.XML     | params no es un objeto XML                | params debe ser un objeto XML de ActionScript.                                                                                                                               |
| XMLConnector.Parse.Error | params tenía el error NN de análisis XML. | La propiedad "status" del objeto XML de params tenía un valor NN distinto de cero. Véase la información sobre XML.status de la Ayuda de Flash para ver los posibles errores. |

| <b>FaultCode</b>                              | <b>FaultString</b>                                            | <b>Notas</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------------------|---------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>XMLConnector.No.Data.Received</code>    | No se han recibido datos del servidor.                        | RESTRICCIÓN: debido a diversas limitaciones de navegador, este mensaje puede significar (a) que la URL del servidor no era válida, no respondía o ha devuelto un código de error HTTP o (b) que la solicitud del servidor se ha ejecutado correctamente pero la respuesta generada tiene 0 bytes de datos. La solución recomendada es: diseñar la aplicación para que el servidor no devuelva NUNCA 0 bytes de datos. Si obtiene el error "XMLConnector.No.Data.Received", sabrá seguro que se ha producido un error en el servidor y podrá notificárselo al usuario final. |
| <code>XMLConnector.Results.Parse.Error</code> | Los datos recibidos tenían un error NN de análisis XML.       | El código XML recibido no era válido, como ha determinado el analizador XML integrado de Flash Player. Véase la información sobre XML.status de la Ayuda de Flash para ver los posibles errores NN.                                                                                                                                                                                                                                                                                                                                                                         |
| <code>XMLConnector.Params.Missing</code>      | La dirección es 'send' o 'send/receive', pero params es null. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

### Ejemplo

En el ejemplo siguiente se define una función `statusFunction` para el evento `status` y se asigna la función al controlador de eventos `addEventListener`:

```
var statusFunction = function (stat) {
    trace(stat.code);
    trace(stat.data.faultcode);
    trace(stat.data.faultstring);
};
xcon.addEventListener("status", statusFunction);
```

## XMLConnector.suppressInvalidCalls

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.suppressInvalidCalls;
```

### Descripción

Propiedad; indica si debe suprimirse una llamada en caso de que los parámetros no sean válidos. Si su valor es `true`, la función `trigger()` no realizará una llamada en caso de que los parámetros vinculados no superen la validación. Se emitirá un evento "status", con el código `InvalidParams`. Si su valor es `false`, la llamada se realizará con los datos no válidos, como sea necesario.

## XMLConnector.trigger()

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.trigger();
```

### Descripción

Método; inicia una Remote Procedure Call. Cada componente RPC define exactamente lo que esto implica. Si la operación es correcta, el resultado de la operación aparecerá en la propiedad `results` del componente RPC.

El método `trigger()` sigue estos pasos:

- 1 Si hay datos vinculados a la propiedad `params`, el método ejecuta todas las vinculaciones para garantizar que haya datos actualizados disponibles. Esto también provoca una validación de datos.
- 2 Si los datos no son válidos y el valor de `suppressInvalidCalls` es `true`, la operación se interrumpe.
- 3 Si la operación continúa, se emite el evento `send`.
- 4 La llamada remota real se inicia a través del método de conexión indicado (por ejemplo, HTTP).

## XMLConnector.URL

### Disponibilidad

Flash Player 6 versión 79.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
componentInstance.URL;
```

### Descripción

Propiedad; URL que este componente utiliza al realizar operaciones HTTP. Esta URL puede ser una URL absoluta o relativa. La URL está sujeta a todas las protecciones de seguridad de Flash Player estándar.



## Componente XUpdateResolver (sólo para Flash Professional)

Los componentes resolver se utilizan junto con el componente DataSet (parte de la función de administración de datos de la arquitectura de datos de Macromedia Flash). Los componentes resolver le permiten convertir los cambios realizados en los datos de la aplicación en un formato que sea apropiado para la fuente de datos externa que está actualizando. Estos componentes no tienen aspectos visuales en tiempo de ejecución.

Si se utiliza un componente DataSet en la aplicación, éste genera un conjunto optimizado de instrucciones (DeltaPacket) que describe los cambios realizados en los datos en tiempo de ejecución. Los componentes resolver convierten este conjunto de instrucciones al formato apropiado (paquete de actualización). Cuando se envía una actualización al servidor, éste envía una respuesta (paquete de resultados) que contiene otras actualizaciones o errores que se producen como consecuencia de la operación de actualización. Los componentes resolver pueden convertir esta información de nuevo en un DeltaPacket que se puede aplicar al componente DataSet para que esté sincronizado con la fuente de datos externa. Los componentes resolver le permiten mantener su aplicación y una fuente de datos externa sincronizados sin escribir código ActionScript adicional.

XUpdate es un estándar para describir los cambios que se realizan en un documento XML, y es compatible con diversas bases de datos XML, como Xindice o XHive. El componente XUpdateResolver convierte el DeltaPacket en sentencias XUpdate. Una fuente de datos externa puede procesar estas sentencias XUpdate. El documento XML contiene la información y el formato necesarios para actualizar cualquier base de datos XUpdate estándar.

Para información sobre el borrador de trabajo de la especificación de lenguaje XUpdate, consulte [www.xmldb.org/xupdate/xupdate-wd.html](http://www.xmldb.org/xupdate/xupdate-wd.html). Existe un componente resolver paralelo, RDBMSResolver (véase “Componente RDBMSResolver (sólo para Flash Professional)” en la página 470), para devolver datos a un servidor basado en XML. Para más información sobre componentes DataSet, consulte “Componente DataSet (sólo para Flash Professional)” en la página 211. Para más información sobre conectores, consulte “WebServiceConnector (sólo para Flash Professional)” en la página 646 y “XMLConnector (sólo para Flash Professional)” en la página 668. Para más información sobre la arquitectura de datos de Flash, consulte “Componentes resolver (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

**Nota:** también puede utilizar el componente XUpdateResolver para enviar actualizaciones de datos a una fuente de datos externa que pueda analizar el lenguaje XUpdate; por ejemplo, una página ASP, un servlet Java o un componente de ColdFusion.

Las actualizaciones del componente XUpdateResolver se envían en forma de paquete de datos XUpdate, que se comunica a la base de datos o al servidor a través de un objeto de conexión. El paquete XUpdate consta de un conjunto optimizado de instrucciones que describe las inserciones, modificaciones y eliminaciones realizadas en el componente DataSet. El componente resolver obtiene un DeltaPacket de un componente DataSet, envía su propio paquete XUpdate a un conector, recibe de la conexión errores de servidor y los transfiere de nuevo al componente DataSet. Todo esto mediante propiedades vinculables.

## Utilización del componente XUpdateResolver (sólo para Flash Professional)

Utilice este componente XUpdateResolver sólo cuando la aplicación Flash contenga un componente DataSet y tenga que devolver una actualización a la fuente de datos. Este componente resuelve los datos que desea devolver a una fuente de datos con formato XML.

Para más información sobre cómo trabajar con el componente XUpdateResolver, consulte “Componentes resolver (sólo para Flash Professional)” en el apartado Utilización de Flash de la Ayuda.

### Parámetros del componente XUpdateResolver

**includeDeltaPacketInfo** *Booleano*; si su valor es `true`, XUpdate incluye información adicional del `deltaPacket` en los atributos de los nodos XUpdate. Esta información incluye los ID de transacción y de operación.

En el ejemplo siguiente se muestra cómo se crea el paquete de actualización si el valor booleano de esta propiedad es `false`:

```
<xupdate:modifications
  version="1.0"
  xmlns:xupdate="http://www.xmldb.org/xupdate">

  <xupdate:remove select="/datapacket/row[@id='100']"/>

</xupdate:modifications>
```

En el ejemplo siguiente se muestra cómo se crea el paquete de actualización si el valor booleano de esta propiedad es `true`:

```
<xupdate:modifications
  version="1.0"
  xmlns:xupdate="http://www.xmldb.org/xupdate"
  transId="46386292065:Wed Jun 25 15:52:34 GMT-0700 2003">

  <xupdate:remove select="/datapacket/row[@id='100']" opId="0123456789"/>

</xupdate:modifications>
```

### Resumen de propiedades del componente XUpdateResolver

| Propiedad                                           | Descripción                                                                                       |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------|
| <code>XUpdateResolver.deltaPacket</code>            | Contiene una descripción de los cambios realizados en el componente DataSet.                      |
| <code>XUpdateResolver.includeDeltaPacketInfo</code> | Incluye información adicional del <code>deltaPacket</code> en los atributos de los nodos XUpdate. |
| <code>XUpdateResolver.updateResults</code>          | Describe los resultados de la actualización.                                                      |
| <code>XUpdateResolver.xupdatePacket</code>          | Contiene la conversión XUpdate de los cambios realizados en el componente DataSet.                |

## Resumen de eventos del componente XUpdateResolver

| Evento                                          | Descripción                                                                                                                                                                       |
|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>XUpdateResolver.beforeApplyUpdates</code> | Llamado por el componente resolver para realizar modificaciones personalizadas inmediatamente después de haber creado el paquete XML e inmediatamente antes enviar dicho paquete. |
| <code>XUpdateResolver.reconcileResults</code>   | Llamado por el componente resolver para comparar dos paquetes.                                                                                                                    |

### XUpdateResolver.beforeApplyUpdates

#### Disponibilidad

Flash Player 7.

#### Edición

Flash MX Professional 2004.

#### Sintaxis

```
resolveData.beforeApplyUpdates(eventObject)
```

#### Parámetros

*eventObject* Objeto de evento resolver; describe las personalizaciones del paquete XML antes de que la actualización se envíe a través del conector a la base de datos. Este objeto de evento resolver debe contener las propiedades siguientes:

| Propiedad                 | Descripción                                           |
|---------------------------|-------------------------------------------------------|
| <code>target</code>       | Objeto; resolver que activa este evento.              |
| <code>type</code>         | Cadena; nombre del evento.                            |
| <code>updatePacket</code> | Objeto XML; objeto XML que está a punto de aplicarse. |

#### Valor devuelto

Ninguno.

#### Descripción

Evento; llamado por el componente resolver para realizar modificaciones personalizadas inmediatamente después de haber creado el paquete XML para un nuevo `deltaPacket` e inmediatamente antes de enviar dicho paquete mediante la vinculación de datos. Puede utilizar este controlador de eventos para realizar modificaciones personalizadas en el código XML antes de enviar los datos actualizados a un conector.

#### Ejemplo

En el ejemplo siguiente se añaden los datos de autenticación de usuario al paquete XML:

```
on (beforeApplyUpdates) {  
    // añadir datos de autenticación de usuarios  
    var userInfo = new XML( ""+getUserId()+" "+getPassword()+" " );  
    xupdatePacket.firstChild.appendChild( userInfo );  
}
```

## XUpdateResolver.deltaPacket

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.deltaPacket
```

### Descripción

Propiedad; propiedad del tipo `deltaPacket` que recibe un `deltaPacket` que debe convertirse en `xupdatePacket` y produce un `deltaPacket` a partir de los resultados de cualquier servidor incluidos en la propiedad `updateResults`. Este controlador de eventos proporciona un método para realizar modificaciones personalizadas en el código XML antes de enviar los datos actualizados a un conector.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade un delta con mensajes al `deltaPacket` de nuevo para que se pueda reenviar la próxima vez que el `deltaPacket` se envíe al servidor. Debe escribir código que maneje deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente `deltaPacket`.

## XUpdateResolver.includeDeltaPacketInfo

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.includeDeltaPacketInfo
```

### Descripción

Propiedad; propiedad de tipo `booleano` que, si su valor es `true`, incluye información adicional del `deltaPacket` en los atributos de los nodos `XUpdate`. Esta información incluirá los ID de transacción y operación.

Para un ejemplo de XML resultante, consulte [“Parámetros del componente XUpdateResolver” en la página 678](#).

## XUpdateResolver.reconcileResults

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.reconcileResults(eventObject)
```

### Parámetros

*eventObject* Objeto ResolverEvent; describe el objeto de evento utilizado para comparar dos *updatePacket*. Este objeto de evento resolver debe contener las propiedades siguientes:

| Propiedad | Descripción                              |
|-----------|------------------------------------------|
| target    | Objeto; resolver que activa este evento. |
| type      | Cadena; nombre del evento.               |

### Valor devuelto

Ninguno.

### Descripción

Evento; utilice esta función callback para insertar código después de que se hayan recibido los resultados del servidor e inmediatamente antes de la transmisión, a través de una vinculación de datos, del *deltaPacket* que contiene los resultados de la operación. Es una buena ubicación para incluir código que gestione mensajes del servidor.

### Ejemplo

En el ejemplo siguiente se igualan dos *updatePacket* y se borran las actualizaciones si han sido correctas:

```
on (reconcileResults) {  
    // examinar resultados  
    if( examine( updateResults ))  
        myDataSet.purgeUpdates();  
    else  
        displayErrors( results );  
}
```

## XUpdateResolver.updateResults

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.updateResults
```

## Descripción

Propiedad; propiedad del tipo `deltaPacket` que contiene los resultados de una actualización devuelta por el servidor mediante un conector. Utilice esta propiedad para transmitir errores y datos actualizados del servidor a un componente `DataSet`; por ejemplo, cuando el servidor asigna nuevos ID para un campo asignado automáticamente. Vincule esta propiedad a la propiedad `Results` de un conector para que pueda recibir los resultados de una actualización y transmitirlos de nuevo al componente `DataSet`.

Los mensajes de la propiedad `updateResults` se tratan como errores. Esto significa que se añade un delta con mensajes al `deltaPacket` de nuevo para que se pueda reenviar la próxima vez que el `deltaPacket` se envíe al servidor. Debe escribir código que maneje deltas que tengan mensajes a fin de que los mensajes se presenten al usuario y se modifiquen antes de añadirlos al siguiente `deltaPacket`.

## XUpdateResolver.xupdatePacket

### Disponibilidad

Flash Player 7.

### Edición

Flash MX Professional 2004.

### Sintaxis

```
resolveData.xupdatePacket
```

### Descripción

Propiedad; propiedad de tipo `xml` que contiene la conversión `XUpdate` de los cambios realizados en el componente `DataSet`. Vincule esta propiedad a la propiedad del componente conector que transmite el paquete de actualización convertido de cambios de nuevo al componente `DataSet`.

# CAPÍTULO 5

## Creación de componentes

En este capítulo se describe cómo puede crear sus propios componentes, permitir que los utilicen otros desarrolladores y empaquetarlos para su despliegue.

### Novedades

La versión actual (versión 2) de la arquitectura de componentes de Macromedia es muy diferente de la versión Macromedia Flash MX (versión 1). Macromedia ha realizado modificaciones para mejorar la escalabilidad y la extensibilidad de los componentes para los desarrolladores. La lista siguiente proporciona información general sobre algunas de las modificaciones:

- Panel Inspector de componentes que reconoce metadatos de ActionScript
- Administradores y clases básicas ampliables
- Vista previa dinámica incorporada
- Mensajes de compilador mejorados
- Nuevo modelo de eventos
- Gestión de la selección
- Estilos basados en CSS

### Trabajo con el entorno de Flash

El entorno de Macromedia Flash MX 2004 y Flash MX Professional 2004 está configurado para crear una estructura lógica de clases y componentes. En esta sección se describe dónde debe almacenar los archivos de componentes.

### Activos de los archivos FLA

Cuando crea un componente, debe empezar con un archivo FLA y añadirle aspectos, gráficos y otros activos. Puede almacenar estos activos en cualquier lugar del archivo FLA, ya que los usuarios de los componentes de Flash sólo necesitan un archivo de componente recopilado y no los activos de origen.

Cuando cree componentes de Flash, debe utilizar archivos SWF de dos capas y dos fotogramas. La primera capa es una capa de acciones, que apunta al archivo de clase ActionScript del componente. La segunda es la capa de los activos, que contiene gráficos, símbolos y otros activos que utiliza el componente.

## Archivos de clase

El archivo FLA incluye una referencia al archivo de clase ActionScript para el componente. Esto se conoce como vinculación del componente al archivo de clase.

El código ActionScript especifica las propiedades y los métodos del componente y define qué clases (si las hay) hereda el componente. Debe utilizar la convención de asignación de nombres de archivo \*.as para el código fuente ActionScript y asignar al archivo de código fuente el nombre del propio componente. Por ejemplo, MyComponent.as contiene el código fuente del componente MyComponent.

Los archivos .as de clase principal de Flash MX 2004 residen en una única carpeta denominada Clases/mx/Core. Otros archivos de clase ActionScript se organizan por su nombre de paquete en sus propias carpetas en /Classes.

Para un componente personalizado, cree un nuevo directorio en /Classes y almacene allí el archivo de clase ActionScript.

## La ruta de clases

En esta sección se describe la ruta de clases de Flash.

### Aspectos básicos de la ruta de clases

La ruta de clases es una lista ordenada de directorios en la que Flash busca archivos de clase durante la exportación de componentes o la generación de archivos SWF. El orden de las entradas de la ruta de clases es importante porque Flash utiliza las clases por orden de llegada. En el momento de la exportación, las clases que se encuentran en la ruta de clases y que coinciden con los identificadores de vínculo del archivo FLA se importan al archivo FLA y se registran con sus símbolos.

Una ruta de clase global hace referencia a todos los archivos FLA que se generan mediante Flash. Una ruta de clase local sólo se aplica al archivo FLA actual.

La ruta de clases local predeterminada está vacía. La ruta de clases global predeterminada consta de las dos rutas siguientes:

- \$(UserConfig)/Classes (Macintosh); \$(LocalData)/Classes (Windows)
- .

El punto (.) indica el directorio de trabajo actual. Flash busca clases de ActionScript en el directorio actual del archivo FLA.

Las rutas \$(UserConfig)/Classes y \$(LocalData)/Classes indican el directorio de configuración del usuario. Este directorio apunta a las ubicaciones siguientes:

- En Windows, este directorio es c:\Documents and Settings\*nombre de usuario*\Datos de programa\Macromedia\Flash MX 2004es\Configuration.
- En Macintosh, este directorio es *volumen:Users:nombre de usuario:Library:Application Support:Macromedia:Flash MX 2004:es:configuration*.

Los directorios UserConfig y LocalData reflejan los directorios ubicados en *Flash\_root/es/Configuration*. Sin embargo, la ruta de clases no incluye estos directorios directamente, y es relativa al directorio UserConfig o LocalData.



## Modificación de la ruta de clases

Puede cambiar la ruta de clases de un archivo FLA independiente (ruta de clases local) o la de todos los archivos FLA con los que trabaja en Flash (ruta de clases global).

### Para cambiar la ruta de clases global:

- 1 Seleccione Edición > Preferencias.  
Aparecerá el cuadro de diálogo Preferencias.
- 2 Seleccione la ficha ActionScript.
- 3 Haga clic en el botón Configuración de ActionScript 2.0.  
Aparecerá el cuadro de diálogo Configuración de ActionScript.
- 4 Añada, elimine o edite entradas en el cuadro Ruta de clases.
- 5 Guarde los cambios.

### Para cambiar la ruta de clases local:

- 1 Seleccione Archivo > Configuración de publicación.  
Aparecerá el cuadro de diálogo Configuración de publicación.
- 2 Seleccione la ficha Flash.
- 3 Pulse el botón Configuración.  
Aparecerá el cuadro de diálogo Configuración de ActionScript.
- 4 Añada, elimine o edite entradas en el cuadro Ruta de clases.
- 5 Guarde los cambios.

## Búsqueda de los archivos de origen de los componentes

Cuando desarrolle un componente, puede almacenar los archivos de origen en cualquier directorio. Sin embargo, debe incluir ese directorio en la configuración de la ruta de clases de Flash MX 2004 para asegurarse de que Flash encuentre los archivos de clase necesarios cuando exporte el componente. Asimismo, para probar el componente, debe almacenarlo en el directorio Flash Components. Para más información sobre el almacenamiento de archivos SWC, consulte [“Utilización de archivos SWC” en la página 708](#).

## Edición de símbolos

Cada símbolo tiene su propia línea de tiempo. Así como puede añadir fotogramas, fotogramas clave y capas a la línea de tiempo principal, también puede efectuar la misma operación en la línea de tiempo de un símbolo.

Cuando crea componentes, siempre se inicia con un símbolo. Flash proporciona los tres métodos siguientes para editar símbolos:

- Editar el símbolo en el contexto de los otros objetos del escenario mediante el comando Editar en contexto. Otros objetos aparecen atenuados para distinguirlos del símbolo que se está editando. El nombre del símbolo que se está editando se muestra en una barra de edición situada en la parte superior del escenario, a la derecha del nombre de la escena actual.
- Editar el símbolo en una ventana distinta mediante el comando Editar en nueva ventana. La edición de un símbolo en una ventana independiente le permite ver a la vez el símbolo y la línea de tiempo principal. El nombre del símbolo que se está editando se muestra en la barra de edición situada en la parte superior del escenario.
- Editar el símbolo cambiando la ventana de la vista del escenario a una vista de sólo el símbolo, con el modo de edición de símbolos. El nombre del símbolo que se está editando se muestra en la barra de edición situada en la parte superior del escenario, a la derecha del nombre de la escena actual.

## Ejemplos de código de componentes

Flash MX 2004 y Flash MX Professional 2004 incluyen los siguientes archivos de origen de componentes para ayudarle a desarrollar sus propios componentes:

- Código fuente del archivo FLA: *Flash MX 2004\_install\_dir*/en/First Run/ComponentFLA/StandardComponents.flc
- Archivos de clase ActionScript: *Flash MX 2004\_install\_dir*/en/First Run/Classes/mx

## Creación de componentes

En esta sección se describe cómo crear un componente que será una subclase de una clase existente de Flash MX 2004. En las secciones posteriores se describe cómo escribir el archivo de clase ActionScript del componente y cómo editar dicho componente para mejorar su calidad y su facilidad de uso.

### Creación de un símbolo de un nuevo componente

Todos los componentes son objetos MovieClip, que son un tipo de símbolo. Para crear un nuevo componente, primero debe insertar un nuevo símbolo en un nuevo archivo FLA.

#### Para añadir el símbolo de un nuevo componente:

- 1 En Flash, cree un documento Flash vacío.
- 2 Seleccione Insertar > Nuevo símbolo.  
Aparecerá el cuadro de diálogo Crear un nuevo símbolo.
- 3 Especifique un nombre de símbolo. Asigne un nombre al componente poniendo en mayúscula la primera letra de cada una de las palabras del componente (por ejemplo, MyComponent).
- 4 Seleccione el botón de opción Clip de película para el comportamiento.  
Un objeto MovieClip dispone de su propia línea de tiempo de varios fotogramas, que se reproduce de forma independiente de la línea de tiempo principal.
- 5 Haga clic en el botón Avanzado.  
El cuadro de diálogo mostrará la configuración avanzada.

- 6 Seleccione Exportar para ActionScript. Esto hace que Flash empaquete el componente de forma predeterminada con cualquier contenido de Flash que utilice el componente.
- 7 Introduzca un identificador de vínculo.  
El identificador se utiliza como nombre de símbolo, nombre de vínculo y nombre de clase asociado.
- 8 En el cuadro de texto Clase de AS 2.0, introduzca la ruta completa en la clase ActionScript 2.0, relativa a la configuración de su ruta de acceso.  
**Nota:** no incluya la extensión del nombre de archivo; el cuadro de texto Clase de AS 2.0 apunta a la ubicación del paquete de la clase y no al nombre que ha otorgado al archivo el sistema de archivos.  
  
Si el archivo ActionScript se encuentra en un paquete, debe incluir el nombre del paquete. El valor de este campo puede ser relativo a la ruta de clases o también una ruta de paquetes absoluta (por ejemplo, myPackage.MyComponent).  
Para más información sobre la configuración de la ruta de clases de Flash MX 2004 consulte [“Aspectos básicos de la ruta de clases” en la página 684](#).
- 9 En la mayoría de los casos, debe anular la selección de Exportar en primer fotograma (está seleccionada de forma predeterminada). Para más información, consulte [“Recomendaciones para el diseño de un componente” en la página 710](#).
- 10 Haga clic en Aceptar.  
Flash añade los símbolos a la biblioteca y cambia al modo de edición de símbolos. En este modo, el nombre del símbolo aparece encima de la esquina superior izquierda del escenario y una cruz filar indica el punto de registro del símbolo.  
Ahora puede editar este símbolo y añadirlo al archivo FLA de su componente.

## Edición de capas de símbolos

Una vez que haya creado el nuevo símbolo y definido sus vínculos, puede definir los activos del componente en la línea de tiempo del símbolo.

El símbolo de un componente debe tener dos capas. En esta sección se describe qué capas deben insertarse y qué debe añadirse a estas capas.

Para obtener más información sobre cómo editar los símbolos, consulte [“Edición de símbolos” en la página 685](#).

### Para editar las capas de los símbolos:

- 1 Introduzca el modo de edición de símbolos.
- 2 Asigne un nuevo nombre a una capa vacía o cree una capa llamada Acciones.
- 3 En el panel Acciones, añada una línea que importe el archivo de clase ActionScript completo del componente.

Esta sentencia se basa en la configuración de la ruta de clases de Flash MX 2004. Para más información, consulte [“Aspectos básicos de la ruta de clases” en la página 684](#). El ejemplo siguiente importa el archivo MyComponent.as que se encuentra en el paquete myPackage:

```
import myPackage.MyComponent;
```

**Nota:** utilice la sentencia `import` y no la sentencia `include` cuando importe un archivo de clase ActionScript. No coloque comillas en el nombre de clase o en el paquete.

- 4 Asigne un nuevo nombre a una capa vacía o cree una capa llamada Activos.  
La capa Activos incluye todos los activos que utiliza este componente.
- 5 En el primer fotograma, añada una acción `stop()` en el panel Acciones, como se muestra en el ejemplo siguiente:  
`stop();`  
No añada ningún activo gráfico a este fotograma. Flash Player se detendrá antes del segundo fotograma, en el que podrá añadir los activos.
- 6 Si está ampliando un componente existente, ubique este componente y todas las demás clases base que utilice y coloque una instancia de ese símbolo en el segundo fotograma de su capa. Para hacerlo, seleccione el símbolo en el panel Componentes y arrástrelo hasta el escenario en el segundo fotograma de la capa Activos de su componente.  
Todos los activos que el componente utilice (ya sea otro componente o medios como mapas de bits) deben disponer de una instancia dentro del componente.
- 7 Añada los activos gráficos que utilice este componente en el segundo fotograma de la capa Activos de su componente. Por ejemplo, si está creando un botón personalizado, añada los gráficos que representan los estados del botón (arriba, presionado, etc.).
- 8 Cuando haya terminado de crear el contenido del símbolo, siga uno de estos procedimientos para volver al modo de edición de documentos:
  - Haga clic en el botón Atrás situado en la parte izquierda de la barra de edición, encima del escenario.
  - Seleccione Edición > Editar documento.
  - Haga clic en el nombre de escena, en la barra de información, encima del escenario.

## Adición de parámetros

El siguiente paso en el desarrollo de componentes es definir los parámetros del componente. Los parámetros son el método principal mediante el que los usuarios modifican las instancias de los componentes que ha creado.

En ediciones anteriores de Flash, solían definirse los parámetros mediante el panel Inspector de componentes. En Flash MX 2004 y Flash MX Professional 2004, se definen los parámetros en el archivo de clase ActionScript y el panel Inspector de componentes identifica los que son públicos para mostrarlos a los usuarios.

La sección siguiente trata la escritura del archivo ActionScript externo del componente, que incluye información sobre la adición de parámetros de componentes.

## Escritura de ActionScript del componente

La mayoría de los componentes incluyen algún tipo de código ActionScript. El tipo de componente determina la ubicación en la que escribirá el ActionScript y la cantidad que debe escribirse. Existen dos enfoques básicos para desarrollar componentes:

- Crear nuevos componentes sin clases principales
- Ampliar las clases de los componentes existentes

Esta sección se centra en la ampliación de los componentes existentes. Si está creando un componente que se basa en el archivo de clase de otro componente, debe escribir un archivo de clase ActionScript externo como se describe en esta sección.

## Ampliación de las clases de los componentes existentes

Cuando se crea el símbolo de un componente que procede de una clase principal, se vincula a un archivo de clase ActionScript 2.0 externo. Para más información sobre la definición de este archivo, consulte [“Creación de un símbolo de un nuevo componente” en la página 686](#).

La clase ActionScript externa amplía otra clase, añade métodos, captadores y definidores y define controladores de eventos para el componente. Para editar los archivos de clase ActionScript, puede utilizar Flash, cualquier editor de texto o cualquier IDE (entorno de desarrollo integrado, Integrated Development Environment).

Sólo puede heredar de una clase. ActionScript 2.0 no permite la herencia múltiple.

## Ejemplo sencillo de un archivo de clase

A continuación se muestra un ejemplo sencillo de un archivo de clase denominado MyComponent.as. Este ejemplo contiene un conjunto mínimo de importaciones, métodos y declaraciones para un componente que hereda de la clase UIObject.

```
import mx.core.UIObject;

class myPackage.MyComponent extends UIObject {
    static var symbolName:String = "MyComponent";
    static var symbolOwner:Object = Object(myPackage.MyComponent);
    var className:String = "MyComponent";
    #include "../core/ComponentVersion.as"
    function MyComponent() {
    }
    function init(Void):Void {
        super.init();
    }
    function size(Void):Void {
        super.size();
    }
}
```

## Proceso general para escribir un archivo de clase

Utilice el siguiente proceso general cuando escriba el ActionScript de un componente. Algunos de los pasos pueden ser opcionales, en función del tipo de componente que haya creado.

Este proceso se analiza más detalladamente en el resto de este capítulo.

### Para escribir el archivo ActionScript de un componente:

- 1 Importe todas las clases necesarias.
- 2 Defina la clase mediante la palabra clave `class`; si es necesario, amplíe una clase principal.
- 3 Defina las variables `symbolName` y `symbolOwner`; éstos son, respectivamente, el nombre del símbolo de la clase ActionScript y el nombre completo del paquete de la clase.
- 4 Defina el nombre de la clase como la variable `className`.
- 5 Añada la información sobre el control de versiones.
- 6 Introduzca las variables de miembro predeterminadas.
- 7 Cree variables para los elementos y los vínculos de cada aspecto que se utilizan en el componente. Esto permite a los usuarios configurar un elemento de aspecto distinto mediante la modificación de un parámetro del componente.

- 8 Añada las constantes de las clases.
- 9 Añada una palabra clave y una declaración de metadatos para cada variable que disponga de un captador/definidor.
- 10 Defina las variables de miembros no inicializadas.
- 11 Defina los captadores y los definidores.
- 12 Escriba un constructor. Normalmente debería estar vacío.
- 13 Añada un método de inicialización. Este método se llama cuando se crea la clase.
- 14 Añada un método de delimitación del tamaño.
- 15 Añada los métodos personalizados o sustituya los métodos heredados.

## Importación de clases

La primera línea del archivo de clase ActionScript externo debe importar los archivos de clase necesarios que utiliza la clase. Esto incluye la clases que proporcionan la funcionalidad, así como la superclase que la clase amplía, si la hay.

Cuando utilice la sentencia `import`, debe importar el nombre de clase completo en lugar del nombre de archivo de la clase, como se muestra en el ejemplo siguiente:

```
import mx.core.UIObject;  
import mx.core.ScrollView;  
import mx.core.ext.UIObjectExtensions;
```

También puede utilizar el carácter comodín (\*) para importar todas las clases de un paquete determinado. Por ejemplo, la sentencia siguiente importa todas las clases del paquete `mx.core`:

```
import mx.core.*;
```

Si una clase importada no se utiliza en un script, la clase no se incluirá en el código de bytes del archivo SWF resultante. Como resultado, la importación de un paquete entero con un carácter comodín no crea un archivo SWF innecesariamente grande.

## Selección de una clase principal

La mayoría de los componentes tienen algunas funciones y comportamientos comunes. Flash incluye dos clases básicas para proporcionar estas características comunes. Si se convierten estas clases en subclases, los componentes empiezan con un conjunto básico de métodos, propiedades y eventos.

La tabla siguiente describe brevemente las dos clases base:

| Clase completa                   | Amplía                 | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------------------------|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>mx.core.UIObject</code>    | <code>MovieClip</code> | <code>UIObject</code> es la clase base para todos los objetos gráficos. Puede tener forma, dibujarse a sí misma y ser invisible. <code>UIObject</code> proporciona las funciones siguientes: <ul style="list-style-type: none"><li>• Edición de estilos</li><li>• Gestión de eventos</li><li>• Cambio del tamaño mediante la escala</li></ul>                                                                                                                                                                               |
| <code>mx.core.UIComponent</code> | <code>UIObject</code>  | <code>UIComponent</code> es la clase base para todos los componentes. Puede participar en la tabulación, aceptar eventos de bajo nivel como las entradas de teclado y de ratón y estar desactivada para no tener que recibir entradas de teclado y de ratón. <code>UIComponent</code> proporciona las funciones siguientes: <ul style="list-style-type: none"><li>• Creación de desplazamientos de la selección</li><li>• Activación y desactivación de componentes</li><li>• Cambio de tamaño de los componentes</li></ul> |

## Aspectos básicos de la clase `UIObject`

Los componentes que se basan en la versión 2 de la arquitectura de componentes de Macromedia proceden de la clase `UIObject`, que ajusta la clase `MovieClip`. La clase `MovieClip` es la clase base de todas las clases de Flash que pueden realizar dibujos en la pantalla. Muchas de las propiedades y los métodos de `MovieClip` están relacionados con la línea de tiempo, que es una herramienta con la que no están familiarizados los desarrolladores que empiezan a trabajar con Flash. La clase `UIObject` fue creada para extraer muchos de estos detalles. Las subclases de `MovieClip` no documentan las propiedades y los métodos de `MovieClip` innecesarios. Sin embargo, puede acceder a estas propiedades y métodos si lo desea.

`UIObject` intenta ocultar la gestión del ratón y de los fotogramas en `MovieClip`. Envía eventos a sus detectores justo antes de dibujar (el equivalente de `onEnterFrame`), cuando se carga y se descarga y cuando su diseño cambia (`move`, `resize`).

`UIObject` proporciona variables de sólo lectura alternativas para determinar la posición y el tamaño del clip de película. Puede utilizar los métodos `move()` y `setSize()` para modificar la posición y el tamaño de un objeto.

## Aspectos básicos de la clase `UIComponent`

`UIComponent` es una clase secundaria de `UIObject`. Es la clase base de todos los componentes que interactúan con el usuario (entradas de teclado y de ratón).

## Ampliación de otras clases

Para facilitar la construcción de componentes, puede crear una subclase para cualquier clase; no es obligatorio ampliar las clases `UIObject` o `UIComponent` directamente. Si amplía la clase de cualquier otro componente, ampliará estas clases de forma predeterminada. Cualquier clase de componente que aparezca en el diccionario de componentes puede ampliarse para crear una nueva clase de componente.

Flash incluye un grupo de clases que dibujan en la pantalla y se heredan desde `UIObject`. Por ejemplo, la clase `Border` dibuja bordes alrededor de otros objetos. Otro ejemplo es `RectBorder`, que es una subclase de `Border` y sabe cómo modificar correctamente el tamaño de sus elementos visuales. Todos los componentes que admiten bordes deben utilizar una de las clases o subclases de borde. Para obtener una descripción detallada de estas clases, consulte el [Capítulo 4, “Diccionario de componentes”](#), en la página 49.

Por ejemplo, si desea crear un componente que se comporte de forma casi exacta que el componente `Button`, puede ampliar la clase `Button` en lugar de volver a crear todas las funciones de la clase `Button` desde las clases base.

## Escritura del constructor

Los constructores son métodos que tienen una única función: configurar propiedades y llevar a cabo otras tareas cuando se crea una instancia nueva para un componente. Puede reconocer un constructor porque tiene el mismo nombre que la clase del componente. Por ejemplo, el código siguiente muestra el constructor del subcomponente `ScrollBar`:

```
function ScrollBar() {  
}
```

En este caso, cuando se crea una nueva instancia para el componente `ScrollBar`, se llama al constructor `ScrollBar()`.

Normalmente, los constructores de componentes deben estar vacíos, de modo que el objeto pueda personalizarse con la interfaz de propiedades. Asimismo, la configuración de propiedades en los constructores puede conducir a menudo a la sobrescritura de los valores predeterminados, en función del orden de las llamadas de inicialización.

Una clase sólo puede contener una función constructora; `ActionScript 2.0` no admite funciones constructoras sobrecargadas.

## Control de versiones

Cuando cree nuevos componentes, debe definir un número de versión. Esto permite a los desarrolladores saber si deben actualizarlo y les ayuda en caso de consultas al servicio de asistencia técnica. Cuando establezca el número de versión de un componente, utilice la variable estática `version`, como se muestra en el ejemplo siguiente:

```
static var version:String = "1.0.0.42";
```

Si crea muchos componentes como parte de un paquete de componentes, puede incluir el número de versión en un archivo externo. Así, actualiza el número de versión en una sola ubicación. Por ejemplo, el código siguiente importa el contenido de un archivo externo que almacena el número de versión en una sola ubicación:

```
#include "../myPackage/ComponentVersion.as"
```

El contenido del archivo `ComponentVersion.as` es idéntico a la declaración de variables anterior, como muestra el siguiente ejemplo:

```
static var version:String = "1.0.0.42";
```



## Nombres de clase, símbolo y propietario

Para ayudar a Flash a encontrar las clases y los paquetes ActionScript y para conservar el nombre del componente, debe establecer las propiedades `symbolName`, `symbolOwner` y `className` en el archivo de clase ActionScript del componente.

La tabla siguiente describe estas variables:

| Variable                 | Descripción                                                                                                                                                                                                                                                     |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>symbolName</code>  | Nombre del símbolo para el objeto. Esta variable es estática y de tipo <code>String</code> .                                                                                                                                                                    |
| <code>symbolOwner</code> | Clase que se utiliza en la llamada interna al método <code>createClassObject()</code> . Este valor debería ser el nombre de clase completo, que incluye la ruta del paquete. Esta variable es estática y de tipo <code>Object</code> .                          |
| <code>className</code>   | Nombre de la clase del componente. Esta variable también se utiliza para calcular los valores de estilo. Si <code>_global.styles[className]</code> existe, establece valores predeterminados para un componente. Esta variable es de tipo <code>String</code> . |

El ejemplo siguiente muestra la denominación de un componente personalizado:

```
static var symbolName:String = "MyComponent";
static var symbolOwner:Object = custom.MyComponent;
var className:String = "MyComponent";
```

## Definición de captadores y definidores

Los captadores y los definidores permiten visualizar las propiedades de los componentes y controlan el acceso de otros objetos a estas propiedades.

La convención para definir métodos de captadores y definidores es colocar antes del nombre del método `get` o `set`, seguido de un espacio y el nombre de la propiedad. Es recomendable utilizar letras en mayúsculas al inicio de las palabras que vayan a continuación de `get` o `set`.

La variable que almacena el valor de la propiedad no puede tener el mismo nombre que el captador o el definidor. Normalmente, antes del nombre de las variables captador o definidor se sitúan dos guiones bajos.

El ejemplo siguiente muestra la declaración de `initialColor`, así como los métodos de captador o definidor que obtienen y establecen el valor de esta propiedad:

```
...
public var __initialColor:Color = 42;
...
public function get initialColor():Number {
    return __initialColor;
}
public function set initialColor(newColor:Number) {
    __initialColor = newColor;
}
```

Los captadores y los definidores suelen utilizarse junto con palabras claves de metadatos para definir las propiedades que son visibles, que se pueden vincular y que cuentan con otras propiedades.

## Metadatos de componente

Flash identifica las sentencias de metadatos de componente en los archivos de clase ActionScript externos. Las etiquetas de metadatos pueden definir los atributos de los componentes, las propiedades de las vinculaciones de datos y los eventos. Flash interpreta estas sentencias y actualiza el entorno de desarrollo en consecuencia. Esto permite definir estos miembros una sola vez, en lugar de en el código ActionScript y en los paneles de desarrollo.

Las etiquetas de metadatos sólo pueden utilizarse en los archivos de clase ActionScript externos. No puede utilizar etiquetas de metadatos en los fotogramas de acción de los archivos FLA.

### Utilización de palabras clave de metadatos

Los metadatos se asocian con una declaración de clases o con un único campo de datos. Si el valor de un atributo es del tipo String, debe ponerlo entre comillas.

Las sentencias de metadatos están vinculadas a la siguiente línea del archivo ActionScript. Cuando defina la propiedad de un componente, añada la etiqueta de metadatos en la línea antes de la declaración de la propiedad. Cuando defina eventos de los componentes, añada la etiqueta de metadatos fuera de la definición de la clase, de modo que el evento se vincule a toda la clase.

En el ejemplo siguiente, las palabras clave de metadatos Inspectable se aplican a los parámetros flavorStr, colorStr y shapeStr:

```
[Inspectable(defaultValue="strawberry")]
public var flavorStr:String;
[Inspectable(defaultValue="blue")]
public var colorStr:String;
[Inspectable(defaultValue="circular")]
public var shapeStr:String;
```

En el inspector de propiedades y en la ficha Parámetros del panel Inspector de componentes, Flash muestra todos estos parámetros como de tipo String.

### Etiquetas de metadatos

La tabla siguiente describe las etiquetas de metadatos que puede utilizar en los archivos de clase ActionScript:

| Etiqueta        | Descripción                                                                                                                                                                                                                                                                                                   |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Inspectable     | Define un atributo expuesto a los usuarios de componentes en el panel Inspector de componentes. Véase <a href="#">“Inspectable” en la página 695</a> .                                                                                                                                                        |
| InspectableList | Identifica qué subconjunto de parámetros Inspectable deben enumerarse en el inspector de propiedades. Si no añade un atributo InspectableList a la clase del componente, todos los parámetros Inspectable aparecen en el inspector de propiedades. Véase <a href="#">“InspectableList” en la página 697</a> . |
| Evento          | Define los eventos de los componentes. Véase <a href="#">“Event” en la página 697</a> .                                                                                                                                                                                                                       |
| Bindable        | Identifica una propiedad en la ficha Vinculaciones del panel Inspector de componentes. Véase <a href="#">“Bindable” en la página 698</a> .                                                                                                                                                                    |

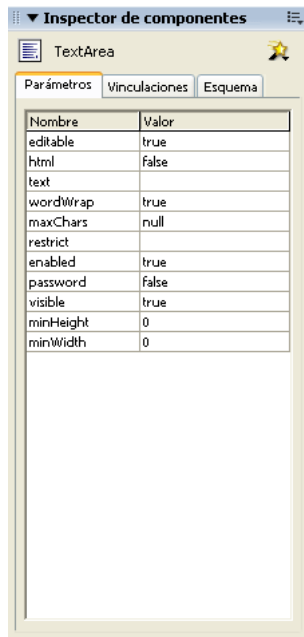
| Etiqueta    | Descripción                                                                                                                                                 |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ChangeEvent | Identifica los eventos que provocan la vinculación de datos. Véase <a href="#">“ChangeEvent” en la página 699</a> .                                         |
| IconFile    | Nombre de archivo del icono que representa este componente en el panel Componentes de Flash. Véase <a href="#">“Adición de un icono” en la página 709</a> . |

Las secciones siguientes describen con mayor detalle las etiquetas de los metadatos de componente.

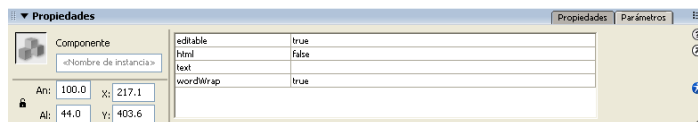
## Inspectable

Se especifican los parámetros que puede editar el usuario (o "Inspectable") de un componente en la definición de su clase. Dichos parámetros aparecen en el panel Inspector de componentes. Esto permite mantener las propiedades Inspectable y el código ActionScript subyacente en la misma ubicación. Para visualizar las propiedades de los componentes, arrastre una instancia del componente hasta el escenario y seleccione la ficha Parámetros del panel Inspector de componentes.

La siguiente ilustración muestra la ficha Parámetros del panel Inspector de componentes para el control TextArea:



De forma alternativa, puede visualizar un subconjunto de propiedades del componente en la ficha Parámetros del inspector de propiedades, como muestra la ilustración siguiente:



Cuando se determinan los parámetros que deben mostrarse en el entorno de edición, Flash utiliza la palabra clave de metadatos `Inspectable`. La sintaxis de esta palabra clave es la siguiente:

```
[Inspectable(value_type=value[,attribute=value,...])]  
property_declaration name:type;
```

El ejemplo siguiente define el parámetro `enabled` como `Inspectable`:

```
[Inspectable(defaultValue=true, verbose=1, category="Other")]  
var enabled:Boolean;
```

La palabra clave `Inspectable` también es compatible con atributos sueltos como el siguiente:

```
[Inspectable("danger", 1, true, maybe)]
```

La sentencia de metadatos debe preceder a la declaración de variables de la propiedad que debe vincularse a dicha propiedad.

La tabla siguiente describe los atributos de la palabra clave de metadatos `Inspectable`:

| Atributo     | Tipo            | Descripción                                                                                                                                                                                                                                                                                                          |
|--------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name         | Cadena          | (Opcional) Un nombre de visualización de la propiedad. Por ejemplo, "Font Width". Si no lo especifica, utilice el nombre de la propiedad, como por ejemplo "_fontWidth".                                                                                                                                             |
| type         | Cadena          | (Opcional) Un especificador del tipo. Si se omite, utilice el tipo de la propiedad. Los valores siguientes son aceptables: <ul style="list-style-type: none"><li>• Matriz</li><li>• Objeto</li><li>• Lista</li><li>• Cadena</li><li>• Número</li><li>• Booleano</li><li>• Nombre de fuente</li><li>• Color</li></ul> |
| defaultValue | Cadena o número | (Obligatorio) Valor predeterminado para la propiedad <code>Inspectable</code> .                                                                                                                                                                                                                                      |
| enumeration  | Cadena          | (Opcional) Especifica una lista separada por comas de valores legales de la propiedad.                                                                                                                                                                                                                               |
| verbose      | Número          | (Opcional) Indica que esta propiedad <code>Inspectable</code> sólo debe mostrarse cuando el usuario indica que deben incluirse propiedades detalles. Si no se especifica este atributo, Flash presupone que la propiedad debe mostrarse.                                                                             |
| category     | Cadena          | (Opcional) Agrupa la propiedad en una subcategoría en el inspector de propiedades.                                                                                                                                                                                                                                   |
| listOffset   | Número          | (Opcional) Se añade para que sea compatible con los componentes de Flash MX. Se utiliza como el índice predeterminado en un Valor de lista.                                                                                                                                                                          |
| variable     | Cadena          | (Opcional) Se añade para que sea compatible con los componentes de Flash MX. Se utiliza para especificar la variable con la que está vinculado este parámetro.                                                                                                                                                       |

## InspectableList

Utilice la palabra clave de metadatos `InspectableList` para especificar exactamente qué subconjunto de parámetros `Inspectable` debe aparecer en el inspector de propiedades. Utilice `InspectableList` junto con `Inspectable`, de modo que pueda ocultar los atributos heredados para los componentes que se han convertido en subclases. Si no añade una palabra clave de metadatos `InspectableList` a la clase del componente, todos los parámetros `Inspectable`, incluidos los de las clases principales del componente, aparecen en el inspector de propiedades.

La sintaxis de `InspectableList` es la siguiente:

```
[InspectableList("attribute1"[,...])]  
// definición de clase
```

La palabra clave `InspectableList` debe preceder a la definición de la clase, ya que se aplica a toda la clase.

El ejemplo siguiente permite mostrar las propiedades `flavorStr` y `colorStr` en el inspector de propiedades, pero excluye otras propiedades `Inspectable` de la clase `DotParent`:

```
[InspectableList("flavorStr","colorStr")]  
class BlackDot extends DotParent {  
    [Inspectable(defaultValue="strawberry")]  
    public var flavorStr:String;  
    [Inspectable(defaultValue="blue")]  
    public var colorStr:String;  
    ...  
}
```

## Event

Utilice la palabra clave de metadatos `Event` para definir los eventos que emite este componente.

La sintaxis de esta palabra clave es la siguiente:

```
[Event("event_name")]
```

Por ejemplo, el código siguiente define un evento `click`:

```
[Event("click")]
```

Añada las sentencias de `Event` fuera de la definición de la clase en el archivo `ActionScript`, de modo que se vinculen a la clase y no a un miembro concreto de dicha clase.

Los ejemplos siguientes muestran los metadatos `Event` para la clase `UIObject`, que gestiona los eventos `resize`, `move` y `draw`:

```
...  
import mx.events.UIEvent;  
[Event("resize")]  
[Event("move")]  
[Event("draw")]  
class mx.core.UIObject extends MovieClip {  
    ...  
}
```

## Bindable

La vinculación de datos conecta los componentes entre sí. La vinculación de datos visual se consigue a través de la ficha Vinculaciones del panel Inspector de componentes. Desde esta ficha puede añadir, visualizar y eliminar vinculaciones para un componente.

Aunque la vinculación de datos funciona con cualquier componente, su objetivo principal es conectar los componentes de la interfaz de usuario a los orígenes de datos externos, como los servicios Web o los documentos XML. Estos orígenes de datos están disponibles como componentes con propiedades, que se pueden vincular con las propiedades de otros componentes. El panel Inspector de componentes es la principal herramienta que se utiliza en Flash MX Professional 2004 para realizar la vinculación de datos.

Utilice la palabra clave de metadatos Bindable para que las propiedades y las funciones de captadores/definidores de las clases ActionScript aparezcan en la ficha Vinculaciones del panel Inspector de componentes.

La palabra clave de metadatos Bindable tiene la sintaxis siguiente:

```
[Bindable[readonly|writeonly[,type="datatype"]]]
```

La palabra clave Bindable debe preceder a una propiedad, a una función de captador/definidor o a otra palabra clave de metadatos que preceda a una propiedad o a una función de captador/definidor.

El ejemplo siguiente define la variable `flavorStr` como una variable pública e Inspectable a la que también se puede acceder desde la ficha Vinculaciones del panel Inspector de componentes:

```
[Bindable]
[Inspectable(defaultValue="strawberry")]
public var flavorStr:String = "strawberry";
```

La palabra clave de metadatos Bindable utiliza tres opciones que especifican el tipo de acceso a la propiedad y el tipo de datos de dicha propiedad. La tabla siguiente describe estas opciones:

| Opción                       | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>readonly</code>        | Indica a Flash que permita que la propiedad sea sólo el origen de una vinculación, como se muestra en el ejemplo:<br><code>[Bindable("readonly")]</code>                                                                                                                                                                                                                                                                                                                             |
| <code>writeonly</code>       | Indica a Flash que permita que la propiedad sea sólo el destino de una vinculación, como se muestra en el ejemplo:<br><code>[Bindable("writeonly")]</code>                                                                                                                                                                                                                                                                                                                           |
| <code>type="datatype"</code> | Especifica el tipo de datos de la propiedad que se está vinculando.<br>Si no especifica esta opción, la vinculación de datos utiliza el tipo de datos de la propiedad como se declara en el código ActionScript.<br>Si <code>datatype</code> es un tipo de datos registrado, puede utilizar la función del menú emergente Tipo de datos de la ficha Esquema.<br>El ejemplo siguiente establece el tipo de datos de la propiedad en String:<br><code>[Bindable(type="String")]</code> |

Puede combinar la opción de acceso y la opción de tipo de datos, como se muestra en el ejemplo siguiente:

```
[Bindable(param1="writeonly",type="DataProvider")]
```

La palabra clave `Bindable` es obligatoria cuando utilice la palabra clave de metadatos `ChangeEvent`. Para más información, consulte [“ChangeEvent” en la página 699](#).

Para obtener información sobre la creación de vinculaciones de datos en el entorno de edición de Flash, consulte [“Vinculación de datos \(sólo en Flash Professional\)”](#) en el apartado Utilización de Flash de la Ayuda.

## ChangeEvent

Utilice la palabra clave de metadatos `ChangeEvent` para generar uno o varios eventos de componentes cuando se efectúen cambios en las propiedades vinculables.

La sintaxis de esta palabra clave es la siguiente:

```
[Bindable]
[ChangeEvent("event"[,...])]
property_declaration or get/set function
```

Como `Bindable`, esta palabra clave sólo puede utilizarse con declaraciones de variables o con funciones de captadores o definidores.

En el ejemplo siguiente, el componente genera el evento `change` cuando el valor de la propiedad vinculable `flavorStr` se modifica:

```
[Bindable]
[ChangeEvent("change")]
public var flavorStr:String;
```

Cuando tiene lugar el evento especificado en los metadatos, Flash informa al elemento que está vinculado a la propiedad de que la propiedad se ha modificado.

También puede indicar al componente que genere un evento cuando se llame a una función de captador o definidor, como se muestra en el ejemplo siguiente:

```
[Bindable]
[ChangeEvent("change")]
function get_selectedDate():Date
...

```

En la mayoría de los casos, el evento `change` se establece en el captador y se distribuye en el definidor.

Puede registrar varios eventos `change` en los metadatos, como se muestra en el ejemplo siguiente:

```
[ChangeEvent("change1", "change2", "change3")]
```

Cualquiera de estos eventos indica un cambio en la variable. No es necesario que tengan lugar todos para indicar un cambio.

## Definición de parámetros de componentes

Cuando crea un componente, puede añadir parámetros que definan su aspecto y su comportamiento. Las propiedades utilizadas más habitualmente aparecen como parámetros de edición en el panel Inspector de componentes. Estas propiedades se definen mediante la palabra clave `Inspectable` (consulte [“Inspectable” en la página 695](#)). También puede establecer todos los parámetros `Inspectable` con `ActionScript`. Los valores de los parámetros establecidos mediante `ActionScript` sustituyen cualquier otro valor que se haya establecido durante la edición.

El ejemplo siguiente establece varios parámetros de componentes en el archivo de clase `JellyBean` y los expone con la palabra clave de metadatos `Inspectable` en el panel Inspector de componentes:

```
class JellyBean{
    // un parámetro de cadena
    [Inspectable(defaultValue="strawberry")]
    public var flavorStr:String;
    // un parámetro de lista de cadenas
    [Inspectable(enumeration="sour,sweet,juicy,rotten",defaultValue="sweet")]
    public var flavorType:String;
    // un parámetro de matriz
    [Inspectable(name="Flavors", defaultValue="strawberry,grape,orange",
        verbose=1, category="Fruits")]
    var flavorList:Array;
    // un parámetro de objeto
    [Inspectable(defaultValue="belly:flop,jelly:drop")]
    public var jellyObject:Object;
    // un parámetro de color
    [Inspectable(defaultValue="#ffffff")]
    public var jellyColor:Color;
}
```

Los parámetros pueden ser de cualquier de los siguientes tipos compatibles:

- Matriz
- Objeto
- Lista
- Cadena
- Número
- Booleano
- Nombre de fuente
- Color

## Implementación de métodos principales

Existen dos métodos principales que todos los componentes deben implementar: el método de tamaño y el método de inicialización. Si no suplanta estos dos métodos en un componente personalizado, puede producirse un error en Flash Player.



## Implementación del método de inicialización

Flash llama al método de inicialización cuando se crea la clase. Como mínimo, el método de inicialización debe llamar al método de inicialización de la superclase. Los parámetros `width`, `height` y `clip` no están configurados correctamente hasta que se llama a este método.

El siguiente método de inicialización de ejemplo de la clase `Button` llama al método de inicialización de la superclase, configura la escala y otros valores predeterminados y obtiene el valor del atributo de color del objeto `UIObject`:

```
function init(Void):Void {
    super.init();
    labelField.selectable = false;
    labelField.styleName = this;
    useHandCursor = false;
    // marca como si se utilizara color "color"
    _color = UIObject.textColorList;
}
```

## Implementación del método de tamaño

Flash llama al método de tamaño del componente desde el método `setSize()` para disponer el contenido del componente. Como mínimo, el método de tamaño debe llamar al método de tamaño de la superclase, como se muestra en el ejemplo siguiente:

```
function size(Void):Void {
    super.size();
}
```

## Gestión de eventos

Los eventos permiten a un componente conocer el momento en el que un usuario ha interactuado con la interfaz, así como los cambios importantes producidos en el aspecto o el ciclo de vida de un componente, como la creación o eliminación de un componente o el cambio de su tamaño.

El modelo de eventos es un modelo distribuidor/detector basado en la especificación Eventos de XML. El usuario escribe un código que registra a los detectores con el objeto de destino, de modo que cuando el objeto de destino distribuya un evento se llame a los detectores.

Los detectores pueden ser funciones u objetos, pero no métodos. El detector recibe un único objeto de evento como el parámetro que contiene el nombre del evento e incluye toda la información relevante sobre éste.

Los componentes generan y distribuyen eventos y consumen (detectan) otros eventos. Un objeto que desea conocer los eventos de otro objeto se registra con este último. Cuando tiene lugar un evento, el objeto distribuye el evento a todos los detectores registrados mediante la llamada de una función solicitada durante el registro. Para recibir varios eventos desde el mismo objeto, debe registrarse para cada evento.

Flash MX 2004 amplía el controlador `on()` `ActionScript` para admitir los eventos de componentes. Se admite cualquier componente que declare eventos en su archivo de clase e implemente el método `addEventListener()`.

## Eventos frecuentes

A continuación, se muestra una lista de difusión de eventos frecuentes para varias clases. Cada componente debe intentar difundir estos eventos si tienen algún significado para ese componente. Esta lista no incluye los eventos de todos los componentes, sino solamente los que otros componentes pueden utilizar. Aunque algunos eventos no especifican ningún parámetro, todos los eventos cuentan con un parámetro implícito: una referencia al objeto que está difundiendo el evento.

| Evento   | Parámetros                                                                                                                                                     | Utilización                                                                                                                        |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| click    | Ninguno                                                                                                                                                        | Utilizado por Button o siempre que un clic del ratón no tenga otro significado.                                                    |
| scroll   | Scrollbar.lineUp, lineDown, pageUp, pageDown, thumbTrack, thumbPosition, endScroll, toTop, toBottom, lineLeft, lineRight, pageLeft, pageRight, toLeft, toRight | Utilizado por ScrollBar y por otros controles de desplazamiento (puntos de desplazamiento en un menú emergente de desplazamiento). |
| cambiar  | Ninguno                                                                                                                                                        | Utilizado por List, ComboBox y otros componentes de entrada de texto.                                                              |
| maxChars | Ninguno                                                                                                                                                        | Se utiliza cuando el usuario intenta especificar demasiados caracteres en los componentes de entrada de texto.                     |

Asimismo, debido a la herencia de `UIComponent`, todos los componentes difunden los eventos siguientes:

| Evento<br><code>UIComponent</code> | Descripción                                                                                                                                                                                                  |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| load                               | El componente está creando o cargando sus subobjetos.                                                                                                                                                        |
| unload                             | El componente está descargando sus subobjetos.                                                                                                                                                               |
| focusIn                            | Ahora el componente tiene la selección de entrada. Algunos componentes equivalentes a HTML (ListBox, ComboBox, Button, Text) también pueden emitir la selección, pero todos emiten <code>DOMFocusIn</code> . |
| focusOut                           | El componente ha perdido la selección de entrada.                                                                                                                                                            |
| mover                              | El componente se ha desplazado a otra ubicación.                                                                                                                                                             |
| resize                             | Se ha cambiado el tamaño del componente.                                                                                                                                                                     |

La tabla siguiente describe eventos clave frecuentes:

| Eventos clave | Descripción                                                                                                                                                                                                                                                                            |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| keyDown       | Se ha presionado una tecla. La propiedad <code>code</code> contiene el código clave y la propiedad <code>ascii</code> contiene el código ASCII de la tecla que se ha presionado. No consulte el objeto <code>Key</code> de bajo nivel, ya que éste tal vez no haya generado el evento. |
| keyUp         | Se ha soltado una tecla.                                                                                                                                                                                                                                                               |

## Utilización del objeto Event

Un objeto Event pasa al detector como un parámetro. El objeto Event es un objeto de ActionScript cuyas propiedades contienen información sobre el evento que ha tenido lugar. El objeto Event se puede utilizar dentro de la función callback del detector para acceder al nombre del evento que se difundió o al nombre de instancia del componente que difundió el evento.

Por ejemplo, el código siguiente utiliza la propiedad `target` del objeto Event `evtObj` para acceder a la propiedad `label` de la instancia `myButton` y rastrear el valor:

```
listener = new Object();
listener.click = function(evtObj){
    trace("clic en el botón " + evtObj.target.label + " ");
}
myButton.addEventListener("click", listener);
```

La tabla siguiente enumera las propiedades más frecuentes en los objetos de evento:

| Propiedad           | Descripción                                                                                                                                                       |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>type</code>   | Cadena que indica el nombre del evento. Esta propiedad es obligatoria.                                                                                            |
| <code>target</code> | Referencia a la instancia del componente que está difundiendo el evento. En general, no es obligatorio que describa este objeto de referencia de forma explícita. |

Los eventos más frecuentes, como `click` y `change`, no tienen ninguna propiedad obligatoria a excepción de `type`.

Puede crear de forma explícita un objeto Event antes de distribuir el evento, como se muestra en el ejemplo siguiente:

```
var eventObj = new Object();
eventObj.type = "myEvent";
eventObj.target = this;
dispatchEvent(eventObj);
```

También puede utilizar una sintaxis de método abreviado que establezca el valor de la propiedad `type` y distribuya el evento en una sola línea:

```
ancestorSlide.dispatchEvent({type:"revealChild", target:this});
```

En el ejemplo anterior, no es obligatorio definir la propiedad `target`, ya que es implícita.

La descripción de cada evento en la documentación de Flash MX 2004 muestra una lista con las propiedades de eventos opcionales y las obligatorias. Por ejemplo, el evento `ScrollBar.scroll` utiliza una propiedad `detail` además de las propiedades `type` y `target`. Para más información, consulte las descripciones de eventos en el [Capítulo 4, “Diccionario de componentes”](#), en la [página 49](#).

## Distribución de eventos

En el cuerpo del archivo de clase `ActionScript` del componente, difunde eventos mediante el método `dispatchEvent()`. La firma para el método `dispatchEvent()` es la siguiente:

```
dispatchEvent(eventObj)
```

El parámetro `eventObj` es el objeto `Event` que describe el evento (consulte [“Utilización del objeto Event” en la página 703.](#))

## Identificación de controladores de eventos

Debe definir el objeto del controlador de eventos o la función del controlador de eventos que detecta los eventos de componente en el `ActionScript` de la aplicación.

En el ejemplo siguiente se crea un objeto detector, se gestiona un evento `click` y se añade como un detector de eventos a `myButton`:

```
listener = new Object();
listener.click = function(evtObj){
    trace("clíc en el botón " + evtObj.target.label + " ");
}
myButton.addEventListener("click", listener);
```

Además de utilizar un objeto detector, también puede utilizar una función como detector. Un detector es una función si no pertenece a ningún objeto. Por ejemplo, el código siguiente crea la función de detección `myHandler()` y la registra en `myButton`:

```
function myHandler(eventObj){
    if (eventObj.type == "click"){
        // aquí código propio
    }
}
myButton.addEventListener("click", myHandler);
```

Para más información sobre la utilización del método `addEventListener()`, consulte [“Utilización de detectores de eventos de componente” en la página 26.](#)

Cuando descubre que un objeto concreto es el único detector de un evento, puede aprovechar que el nuevo modelo de eventos siempre llama a un método en la instancia de componente. Este método es el nombre del evento más la palabra `Handler`. Por ejemplo, para gestionar el evento `click`, escriba el código siguiente:

```
myComponentInstance.clickHandler = function(o){
    // introduzca aquí el código propio
}
```

En el código anterior, la palabra clave `this`, si se utiliza en la función callback, tiene el ámbito de `myComponentInstance`.

También puede utilizar objetos detectores que admitan el método `handleEvent()`. Con independencia del nombre del evento, se llama al método `handleEvent()` del objeto detector. Para gestionar varios eventos, es preciso utilizar una sentencia `if...else` o `switch`, por lo que la sintaxis es complicada. Por ejemplo, el código siguiente utiliza una sentencia `if...else` para gestionar los eventos `click` y `enter`:

```
myObj.handleEvent = function(o){
    if (o.type == "click"){
        // aquí código propio
    } else if (o.type == "enter"){
        // aquí código propio
    }
}
target.addEventListener("click", myObj);
target2.addEventListener("enter", myObj);
```

## Utilización de metadatos de evento

Añada metadatos de evento al archivo de clase `ActionScript` para cada detector de eventos. El valor de la palabra clave `Event` es el primer argumento en las llamadas al método `addEventListener()`, como se muestra en el ejemplo siguiente:

```
[Event("click")] // declaración de evento
...
class FCheckBox{
    function addEventListener(eventName:String, eventHandler:Object) {
        ... //eventName es String
    }
}
```

Para más información sobre la palabra clave de metadatos `Event`, consulte [“Event” en la página 697](#).

## Aplicación de aspectos

Un control de interfaz de usuario está formado en su totalidad por clips de película adjuntos. Esto significa que todos los activos de un control de interfaz de usuario pueden ser externos al clip de película del control de interfaz de usuario, de modo que otros componentes puedan utilizarlos. Por ejemplo, si el componente necesita funcionalidad de botón, puede reutilizar los activos del componente `Button` existente.

El componente `Button` utiliza un clip de película independiente para representar todos sus estados (`FalseDown`, `FalseUp`, `Disabled`, `Selected`, etc.). Sin embargo, puede asociar los clips de película personalizados, llamados *aspectos*, con estos estados. Durante el tiempo de ejecución, los clips de película antiguos y nuevos se exportan al archivo SWF. Los estados antiguos pasan a ser invisibles para dejar paso a los nuevos clips de película. Esta capacidad para cambiar los aspectos durante la edición y durante el tiempo de ejecución se denomina *aplicar aspectos*.

Para utilizar la aplicación de aspectos en los componentes, cree una variable para los elementos/vinculaciones de cada aspecto que se utilizan en el componente. Esto permite establecer un elemento de aspecto distinto simplemente modificando un parámetro del componente, como se muestra en el ejemplo siguiente:

```
var falseUpSkin = "mySkin";
```

El nombre `“mySkin”` se utiliza como el nombre de la vinculación del símbolo `MovieClip` para mostrar el aspecto `FalseUp`.

El ejemplo siguiente muestra las variables de aspecto para varios estados del componente Button:

```
var falseUpSkin:String = "ButtonSkin";
var falseDownSkin:String = "ButtonSkin";
var falseOverSkin:String = "ButtonSkin";
var falseDisabledSkin:String = "ButtonSkin";
var trueUpSkin:String = "ButtonSkin";
var trueDownSkin:String = "ButtonSkin";
var trueOverSkin:String = "ButtonSkin";
var trueDisabledSkin:String = "ButtonSkin";
var falseUpIcon:String = "";
var falseDownIcon:String = "";
var falseOverIcon:String = "";
var falseDisabledIcon:String = "";
var trueUpIcon:String = "";
var trueDownIcon:String = "";
var trueOverIcon:String = "";
var trueDisabledIcon:String = "";
```

## Adición de estilos

Añadir estilos es el proceso que permite registrar todos los elementos gráficos del componente con una clase; esa clase controla entonces los esquemas de color de los gráficos durante el tiempo de ejecución. No es necesario ningún código especial en las implementaciones del componente para admitir los estilos. Los estilos se implementan en su totalidad en las clases y los aspectos base.

Para más información sobre estilos, consulte [“Utilización de estilos para personalizar el texto y el color de un componente” en la página 31](#).

## Cómo facilitar el acceso a los componentes

Cada vez resulta más importante que el contenido Web sea accesible para las personas con discapacidades. Las personas con deficiencias visuales pueden utilizar el contenido visual de las aplicaciones Flash mediante un software de lector de pantalla, que proporciona una descripción en audio del material que aparece en pantalla.

Flash MX 2004 incluye las siguientes funciones de accesibilidad:

- Desplazamiento personalizado de la selección
- Métodos abreviados de teclado personalizados
- Documentos basados en pantallas y el entorno de edición de pantallas
- La clase Accessibility

Al crear un componente podrá incluir ActionScript que permita que el componente y el lector de pantalla se comuniquen. Cuando los desarrolladores utilicen el componente para crear una aplicación en Flash, emplearán el panel Accesibilidad para configurar la instancia de cada componente.

Añada la línea siguiente al archivo FLA de su componente, en la misma capa en la que va a añadir otras llamadas de ActionScript:

```
mx.accessibility.ComponentName.enableAccessibility();
```

Por ejemplo, la línea siguiente permite la accesibilidad al componente MyButton:

```
mx.accessibility.MyButton.enableAccessibility();
```

Cuando los desarrolladores añaden el componente MyButton a una aplicación, pueden utilizar el panel Accesibilidad para hacerlo accesible a los lectores de pantalla.

Para obtener información sobre el panel Accesibilidad y otras funciones de accesibilidad de Flash, consulte “Creación de contenido accesible”, en el apartado Utilización de Flash de la Ayuda.

## Exportación del componente

Flash MX 2004 exporta componentes como paquetes de componentes (archivos SWC). Cuando distribuye un componente, sólo necesita entregar a los usuarios el archivo SWC. Este archivo contiene todo el código, los archivos SWF, las imágenes y los metadatos asociados con el componente, de modo que los usuarios puedan colocarlo fácilmente en su entorno de Flash.

En esta sección se describe un archivo SWC y se explica cómo importar y exportar archivos SWC en Flash.

## Aspectos básicos de los archivos SWC

Un archivo SWC es un archivo zip (empaquetado y ampliado mediante el formato de archivos PKZip) que genera la herramienta de edición de Flash.

En la tabla siguiente se describen los contenidos de un archivo SWC.

| Archivo                               | Descripción                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| catalog.xml                           | (Obligatorio) Muestra una lista con el contenido del paquete de componentes y cada uno de sus componentes y funciona como directorio para los otros archivos del archivo SWC.                                                                                                                                                                                                                                                                                                                                                                                                |
| Código fuente                         | Si el componente se crea mediante Flash MX 2004, el código fuente es uno o varios archivos ActionScript que contienen una declaración de clase para el componente.<br>El código fuente sólo se utiliza para la verificación de tipos cuando se crean subclases para los componentes y la herramienta de edición no lo compila, ya que el código de bits ya se encuentra en el archivo SWF que se implementa.<br>El código fuente puede contener definiciones de clases intrínsecas que no contienen cuerpos de función y sólo se proporcionan para la verificación de tipos. |
| Implementación de archivos SWF        | (Obligatorio) Archivos SWF que implementan los componentes. En un solo archivo SWF pueden definirse uno o varios componentes. Si el componente se crea mediante Flash MX 2004, sólo se exporta un componente por archivo SWF.                                                                                                                                                                                                                                                                                                                                                |
| Archivos SWF de vista previa dinámica | (Opcional) Si se especifica, estos archivos SWF se utilizan para realizar obtener una vista previa dinámica en la herramienta de edición. Si se omiten, se utilizarán los archivos SWF que se van a implementar para realizar la vista previa dinámica. El archivo SWF de vista previa dinámica puede omitirse en casi todas las clases; debe incluirse sólo si la apariencia del componente depende de datos dinámicos (por ejemplo, un campo de texto que muestra el resultado de una llamada de servicio Web).                                                            |
| Información de depuración             | (Opcional) Archivo SWD que corresponde al archivo SWF que se va a implementar. El nombre de archivo siempre es el mismo que el del archivo SWF, pero con la extensión .swd. Si se incluye en el archivo SWC, se permite la depuración del componente.                                                                                                                                                                                                                                                                                                                        |

| Archivo                  | Descripción                                                                                                                                                                                                                                                                                                           |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Icono                    | (Opcional) Archivo PNG que contiene el icono de 18 x 18 de 8 bits por píxel que se utiliza para mostrar un componente en la interfaz de usuario de la herramienta de edición. Si no se proporciona ningún icono, se muestra un icono predeterminado. (Véase <a href="#">“Adición de un icono” en la página 709.</a> ) |
| Inspector de propiedades | (Opcional) Si se especifica, este archivo SWF se utiliza como el inspector de propiedades personalizado en la herramienta de edición. Si se omite, se muestra al usuario el inspector de propiedades predeterminado.                                                                                                  |

Para visualizar el contenido de un archivo SWC, puede abrirlo mediante cualquier utilidad de compresión que admita el formato PKZip (incluido WinZip).

De forma opcional, puede incluir otros archivos en el archivo SWC una vez que lo haya generado en el entorno Flash. Por ejemplo, es posible que desee incluir un archivo Readme (Léame) o el archivo FLA si desea que los usuarios puedan acceder al código fuente del componente.

Los diversos archivos SWC se expanden en un único directorio, por lo que cada componente debe disponer de un nombre de archivo exclusivo para evitar conflictos.

## Utilización de archivos SWC

En esta sección se describe cómo crear e importar archivos SWC. Debe proporcionar instrucciones para importar archivos SWC a los usuarios de sus componentes.

### Creación de archivos SWC

Flash MX 2004 y Flash MX Professional 2004 permiten crear archivos SWC mediante la exportación de un clip de película como archivo SWC. Cuando crea un archivo SWC, Flash informa de los errores en el tiempo de compilación como si estuviera probando una aplicación Flash.

#### Para exportar un archivo SWC:

- 1 Seleccione un elemento en la biblioteca Flash.
- 2 Haga clic con el botón derecho del ratón (Windows) o haga clic con la tecla Control presionada (Macintosh) sobre el elemento y seleccione Exportar archivo SWC.
- 3 Guarde el archivo SWC.

### Importación de archivos SWC de componente a Flash

Cuando distribuya sus componentes a otros desarrolladores, puede incluir las siguientes instrucciones para que puedan instalarlos y utilizarlos inmediatamente.

#### Para utilizar un archivo SWC en el entorno de edición de Flash, haga lo siguiente:

- 1 Cierre el entorno de edición de Flash.
- 2 Copie el archivo SWC en el directorio *flash\_root/en/First Run/Components*.
- 3 Inicie el entorno de edición de Flash o vuelva a cargar el panel Componentes.  
El icono del componente debe aparecer en el panel Componentes.



## Cómo facilitar la utilización del componente

Una vez que haya creado el componente y lo haya preparado para su empaquetado, puede facilitar su utilización por parte de los usuarios. En esta sección se describen algunas técnicas para facilitar la utilización del componente.

### Adición de un icono

Puede añadir un icono que represente el componente en el panel Componentes del entorno de edición de Flash.

**Para añadir un icono del componente, haga lo siguiente:**

- 1 Cree una imagen nueva.  
La imagen debe medir 18 píxeles cuadrados y guardarse en formato PNG. Debe ser de 8 bits y con transparencia alfa, y el píxel superior izquierdo debe ser transparente para admitir máscaras.
- 2 Añada la definición siguiente al archivo de clase ActionScript del componente antes de la definición de la clase:  

```
[IconFile("component_name.png")]
```
- 3 Añada la imagen al mismo directorio en el que se encuentra el archivo FLA. Cuando exporte el archivo SWC, Flash incluirá la imagen en la raíz del archivo.

### Utilización de la vista previa dinámica

La función Vista previa dinámica, que se activa de forma predeterminada, permite ver los componentes del escenario tal como aparecerán en el contenido de Flash publicado, con su tamaño aproximado.

Ya no es necesario añadir una vista previa dinámica cuando se crean componentes mediante la arquitectura de la versión 2. Los archivos SWC del componente incluyen el archivo SWF que va a implementarse y el componente utiliza dicho archivo SWF en el escenario de Flash.

### Adición de sugerencias

Las sugerencias aparecen cuando un usuario pasa el ratón por encima del nombre o del icono del componente en el panel Componentes del entorno de edición de Flash.

Para añadir sugerencias al componente, utilice la palabra clave `tiptext` fuera de la definición de la clase en el archivo de clase ActionScript del componente. Debe marcar esta palabra clave mediante un asterisco (\*) y precederla del símbolo @ para que el compilador la reconozca adecuadamente.

En el ejemplo siguiente se muestra la sugerencia para el componente `CheckBox`:

```
* @tiptext Componente CheckBox básico. Amplía Button.
```

## Recomendaciones para el diseño de un componente

Utilice las siguientes recomendaciones cuando diseñe un componente:

- Intente que el tamaño del archivo sea lo más reducido posible.
- Intente que su componente sea lo más reutilizable posible mediante la generalización de sus funciones.
- Utilice el nuevo modelo de eventos en lugar de la sintaxis `on(event)`.
- Utilice la clase `Border`, en lugar de los elementos gráficos, cuando dibuje los bordes de los objetos.
- Utilice la aplicación de aspectos basadas en etiquetas.
- Utilice la propiedad `symbolName`.
- Asuma un estado inicial. Puesto que las propiedades de estilo se encuentran ahora en el objeto, puede establecer la configuración inicial para los estilos y las propiedades para que su código de inicialización no tenga que establecerlo cuando se cree el objeto, a no ser que el usuario sustituya el estado predeterminado.
- Cuando defina el símbolo no seleccione la opción Exportar en primer fotograma, a no ser que sea absolutamente necesario. Flash carga el componente justo antes de que se utilice en la aplicación Flash, por lo que si selecciona esta opción Flash precargará el componente en el primer fotograma de su principal. La razón por la que habitualmente no se precarga el componente en el primer fotograma tiene que ver con la Web: el componente se carga antes de que se inicie el precargador, lo que inutiliza el objetivo de este último.

# ÍNDICE ALFABÉTICO

## A

- accesibilidad
  - editar 15
  - para componentes personalizados 706
  - y componentes 15
- Accordion, componente 52
  - Accordion, clase 58
  - crear aplicaciones 54
  - parámetros 53
  - personalizar 56
  - utilizar 53
  - utilizar aspectos 57
  - utilizar estilos 56
- ActionScript
  - escribir para un nuevo componente 688
  - flujo de trabajo para escribir para un nuevo componente 689
- activadores de menú 423
- Activos de los archivos FLA, almacenamiento para los archivos de componentes 683
- addEventListener 704
- Alert, clase
  - métodos 69
  - propiedades 70
- Alert, componente 66
  - Alert, clase 69
  - crear aplicaciones 67
  - eventos 70
  - parámetros 67
  - personalizar 68
  - utilizar 67
  - utilizar aspectos 68
  - utilizar estilos 68
- ampliar clases 691
- añadir componentes mediante ActionScript 23
- API de componente RPC
  - para WebServiceConnector 647

- y XMLConnector 668
- aplicar aspectos 41
  - para componentes personalizados 705
- archivo WSDL
  - obtener una actualización para 647
  - para servicio Web 646
- archivos de componentes, almacenar 685
- archivos de origen de componentes 686
- archivos SWC 14
  - crear 708
  - explicación del formato del archivo 707
  - importar 708
  - trabajar 20
  - y clips compilados 14
- arquitectura de componentes de la versión 1, diferencias con la versión 2 683
- arquitectura de componentes de la versión 2
  - modificaciones de la versión 1 683
  - utilizar el archivo SWC para la vista previa dinámica 709
- aspectos 41
  - aplicar 43
  - aplicar a subcomponentes 44
  - editar 42
- atributos XML 396

## B

- Biblioteca, panel 18
- Binding, clase 132
- Button, componente 75
  - Button, clase 79
  - crear aplicaciones 76
  - eventos 80
  - métodos 80
  - parámetros 76, 515
  - personalizar 77
  - propiedades 80

- utilizar 76
  - utilizar aspectos 78
  - utilizar estilos 77
- C**
- cambiar el tamaño de los componentes 23
  - capas de símbolos, editar para un nuevo componente 687
  - captadores, definir para propiedades 693
  - categorías
    - administradores 51
    - controles de la IU 49
    - datos 50
    - medios 51
    - pantallas 52
  - CellRenderer
    - métodos 88, 89
    - propiedades 89
    - utilizar 88
  - CellRenderer, API 87
  - CellRenderer, componente 87
  - Centro de soporte de Macromedia Flash 10
  - CheckBox, componente 94
    - CheckBox, clase 97
    - crear aplicaciones 95
    - eventos 98
    - métodos 97
    - parámetros 95
    - propiedades 98
    - utilizar 94
    - utilizar aspectos 96
    - utilizar estilos 96
  - clase
    - archivos, almacenamiento para componentes 684
    - hojas de estilos 31
    - nombre, para un componente personalizado 693
  - clase DateField, métodos 272
  - clase PopUpManager, métodos 443
  - clase principal, seleccionar para un nuevo componente 690
  - clase StyleManager, métodos 540
  - clase UIComponent
    - y herencia de componente 13
  - Clase UIObject, definida 691
  - clases
    - Accordion 58
    - Alert 69
    - ampliar 689, 691
    - Binding 132
    - Button 79
    - CheckBox 97
    - ComboBox 107
    - ComponentMixins 147
    - crear subclases 691
    - CustomFormatter 135
    - CustomValidator 138
    - DataGrid 171
    - DataGridColumn 191
    - DataSet 214
    - DataType 153
    - DateChooser 259
    - DateField 272
    - elementos multimedia 365
    - EndPoint 142
    - FocusManager 294
    - importar 690
    - Label 307
    - List 314
    - Loader 342
    - Log 622
    - Menu 403
    - MenuBar 427
    - NumericStepper 438, 574
    - PendingCall 625
    - ProgressBar 449
    - RadioButton 463
    - ScrollPane 501
    - seleccionar una clase principal 690
    - SOAPCall 634
    - TextArea 545
    - TextInput 557
    - UIComponent 691
    - UIObject 691
    - WebService 638
    - y herencia de componente 13
  - className 693
  - clickHandler 27
  - clips compilados 14
    - del panel Biblioteca 18
    - trabajar 20
  - colores
    - definir propiedades de estilo 36
  - ComboBox, componente 102
    - ComboBox, componente 107
    - crear aplicaciones 104
    - métodos 108
    - parámetros 104
    - propiedades 108
    - utilizar 104

- utilizar aspectos 106
  - utilizar estilos 105
  - ComboBox, eventos 109
  - componentes
    - añadir a documentos de Flash 21
    - añadir dinámicamente 23
    - arquitectura 13
    - cambiar el tamaño 23
    - categorías 49
    - categorías, descripción 12
    - compatibilidad con Flash Player 13
    - DateField 269
    - DepthManager 286
    - disponibles en Flash MX 2004 8
    - disponibles en Flash MX Professional 2004 8
    - eliminar 24
    - FocusManager, clase 292
    - herencia 13
    - instalar 17
    - medios 350
  - Componentes de datos 50
  - componentes multimedia 51, 350
    - aspectos básicos 352
    - comportamientos, asociar MediaController y MediaDisplay 360
    - comportamientos, asociar MediaDisplay y MediaController 359
    - comportamientos, utilizar Navegación por cuepoints en diapositivas 360
    - comportamientos, utilizar Navegación por cuepoints en fotogramas con nombre 360
    - crear aplicaciones 364
    - interacción con 351
    - parámetros 361
    - personalizar 364
    - utilizar 355
    - utilizar aspectos 364
    - utilizar el Inspector de componentes 358
    - utilizar estilos 364
  - Componentes, panel 17
  - ComponentMixins, clase 147
  - comportamientos
    - vídeo, control de la reproducción 359
  - constructor, escribir para un nuevo componente 692
  - controles de la interfaz de usuario (IU) 49
  - convenciones tipográficas, en la documentación de componentes 10
  - crear componentes
    - accesibilidad 706
    - añadir eventos 704
    - añadir texto de información 709
    - aplicar aspectos 705
    - crear archivos SWC 708
    - definir parámetros 700
    - estilos 706
    - eventos frecuentes 702
    - exportar 707
    - gestionar eventos 701
    - implementar métodos principales 700
    - importar archivos SWC 708
    - metadatos de evento 705
    - seleccionar el propietario de un símbolo 693
    - seleccionar un nombre de clase 693
    - seleccionar un nombre de símbolo 693
    - utilizar sentencias de metadatos 694
    - vista previa dinámica con el archivo SWC 709
  - crear un componente
    - ampliar la clase de un componente 689
    - añadir parámetros 688
    - añadir un icono 709
    - Clase UIComponent definida 691
    - Clase UIObject definida 691
    - crear una subclase de una clase 691
    - definir un número de versión 692
    - editar capas de símbolos 687
    - ejemplo de código para el archivo de clase 689
    - escribir ActionScript 688
    - escribir un constructor 692
    - proceso para escribir ActionScript 689
    - seleccionar una clase principal 690
    - símbolo del componente 686
  - CSSStyleDeclaration 33
  - CustomFormatter, clase 135
  - CustomValidator, clase 138
- ## D
- DataGrid, clase
    - métodos 171
    - propiedades 172
  - DataGrid, componente 166
    - aspectos básicos 168
    - clase 171
    - crear aplicaciones 169
    - interacción con 166
    - modelo de datos 168
    - parámetros 169
    - personalizar 170
    - utilizar 167

- utilizar aspectos 171
- utilizar estilos 170
- vista 168
- DataGridColumn, clase 191
  - métodos 192
- DataHolder, componente 198
- DataProvider, clase
  - métodos 201
- DataSet, clase 214
- DataSet, componente 211
- DataType, clase 153
- DateChooser, clase
  - métodos 260
  - propiedades 260
- DateChooser, componente 257
  - crear aplicaciones 258
  - DateChooser, clase 259
  - eventos 260
  - parámetros 257
  - personalizar 258
  - utilizar 257
  - utilizar aspectos 259
  - utilizar estilos 258
- DateTimeField, componente 269
  - crear aplicaciones 270
  - DateTimeField, clase 272
  - eventos 273
  - parámetros 269
  - propiedades 272
  - utilizar 269
  - utilizar aspectos 271
  - utilizar estilos 271
- declaraciones de estilo
  - clase predeterminada 35
  - crear personalizadas 33
  - definir clase 35
  - global 33
- defaultPushButton 28
- definidores, definir para propiedades 693
- DeltaPacket
  - información 677
  - uso con componentes 677
- DepthManager 28
  - clase 286
  - métodos 287
- desplazamiento de la selección, crear 27
- detail 632, 644
- detector
  - funciones 27
- detectores 26

- documentación
  - guía de términos 10
  - información general 9

## E

- editar símbolos, para componentes 685
- ejemplo de código del archivo de clase del componente 689
- ejemplos de código para desarrollar componentes 686
- element 632, 644
- elementos multimedia
  - componentes
    - utilizar comportamientos 359
- EndPoint, clase 142
- estilos 31
  - admitidos 38
  - definir en una instancia 32
  - definir global 33
  - definir personalizados 33
  - determinar precedencia 35
  - establecer 31, 37
  - herencia, seguir 540
  - para componentes personalizados 706
- estilos de instancia 31
- etiquetas 23
- etiquetas para metadatos 694
- evento
  - detectores 26
  - metadatos 697, 705
  - objetos 26
- eventos 25
  - añadir 704
  - difundir 26
  - eventos frecuentes 702
  - gestión 701
- exportar componentes personalizados 707

## F

- faultactor 632, 644
- faultcode 632, 644
- faultstring 632, 644
- Flash MX 2004, componentes disponibles 8
- Flash MX Professional 2004, componentes disponibles 8
- Flash Player
  - soporte 29
  - y componentes 13
- Flash Professional
  - RDBMSResolver, componente 470

- tipos de componente 470, 677
- XUpdateResolver, componente 677
- FragmentManager 27
- FragmentManager, clase 292
- FragmentManager, componente
  - crear aplicaciones 293
  - FragmentManager, clase 294
  - parámetros 293
  - personalizar 293
  - utilizar 292
- Form, clase 299

## G

- global
  - ruta de clases 684

## H

- handleEvent 27
- herencia
  - en componentes de la versión 2 13
- hojas de estilos personalizadas 31

## I

- icono para un componente personalizado 709
- identificadores de vínculos para aspectos 42
- Importar clases 690
- Inspectable en sentencias de metadatos,
  - propiedades 695
- Inspector de componentes, panel 18
- Inspector de propiedades 18
- instalación
  - instrucciones 9
  - verificar 9
- instalar componentes 8
- instancias
  - definir estilos 32
- instancias, definir estilos 32
- interfaces
  - TransferObject 566
- interfaz
  - TreeDataProvider 588
- Interfaz API de DataProvider 200
  - eventos 201
  - propiedades 201
- interfaz API de pantalla 52
- Interfaz TreeDataProvider
  - métodos 588
  - propiedades 589

## L

- Label, clase 307
- Label, componente 306
  - crear aplicaciones 306
  - eventos 308
  - Label, clase 307
  - métodos 308
  - parámetros 306
  - personalizar 306
  - propiedades 308
  - utilizar 305
  - utilizar estilos 307
- lectores de pantalla, accesibilidad 15
- List, clase 314
  - composición 87
  - desplazar 87
- List, componente 310
  - aspectos básicos 87
  - crear aplicaciones 311
  - eventos 317
  - List, clase 314
  - métodos 315
  - parámetros 311
  - personalizar 312
  - propiedades 316
  - utilizar 311
  - utilizar estilos 313
- Loader, componente 339
  - crear aplicaciones 340
  - eventos 343
  - Loader, clase 342
  - métodos 342
  - parámetros 340
  - personalizar 341
  - propiedades 342
  - utilizar 340
- Log, clase 622

## M

- Macromedia DevNet 10
- Media, clase 365
  - eventos 367
  - métodos 365
  - propiedades 366
- MediaController, componente
  - aspectos básicos 354
  - parámetros 362
  - utilizar 357

- MediaDisplay, componente
  - aspectos básicos 353
  - parámetros 361
  - utilizar 357
- MediaPlayer, componente
  - aspectos básicos 355
  - parámetros 363
  - utilizar 355
- Menu, clase 403
  - métodos 404
  - propiedades 404
- Menu, componente 392
  - añadir menús jerárquicos 395
  - atributos XML 396
  - clase 403
  - crear aplicaciones 400
  - elementos de menú, tipos 397
  - exposición de elementos en ActionScript 398
  - interacción con 393
  - parámetros 400
  - personalizar 402
  - propiedades de objeto de inicialización 399
  - utilizar 394
  - utilizar aspectos 403
  - utilizar estilos 403
- MenuBar, clase
  - métodos 427
  - propiedades 427
- MenuBar, componente 423
  - clase 427
  - crear aplicaciones 424
  - interacción con 424
  - parámetros 424
  - personalizar 425
  - utilizar 424
  - utilizar aspectos 427
  - utilizar estilos 426
- metadatos 694–699
  - etiquetas 694
  - evento 697, 705
  - explicación 694
  - Inspectable, propiedades 695
  - sintaxis 694
- método de inicialización, implementar 701
- método de tamaño, implementar 701
- método handleEvent 27
- métodos
  - definir captadores y definidores 693
  - implementar 700

- inicialización, implementar 701
- tamaño, implementar 701
- modelo de datos
  - Menu, componente 395

## N

- name
  - símbolo, para un componente personalizado 693
- nombre
  - clase 693
- NumericStepper, clase
  - métodos 59, 419, 438
  - propiedades 59, 272, 404, 438
- NumericStepper, componente 257, 434
  - crear aplicaciones 258, 435, 571
  - eventos 59, 69, 171, 192, 272, 404, 419, 427, 438
- NumericStepper, clase 438
  - parámetros 257, 435
  - personalizar 258, 436
  - utilizar 257, 435
  - utilizar aspectos 259, 437
  - utilizar estilos 56, 258, 436
- números de versión para componentes 692

## O

- on() 25
- onFault 633, 644
- orden de tabulación, para componentes 292

## P

- paquetes 13
- parámetros
  - añadir a un nuevo componente 688
  - definir 24, 700
  - establecer 18
  - Inspectable, en sentencias de metadatos 695
  - visualizar 18
- PendingCall, clase 625
- personalizar
  - color 31
  - texto 31
- PopUpManager, clase 443
- profundidad, gestionar 28
- ProgressBar, componente 445
  - crear aplicaciones 446
  - eventos 450
  - métodos 449



- parámetros 445
- personalizar 447
- ProgressBar, clase 449
- propiedades 450
- utilizar 445
- utilizar aspectos 448
- utilizar estilos 448
- propiedades de aspecto
  - cambiar en el prototipo 47
  - establecer 42
- propiedades de estilo
  - color 36
  - establecer 37
  - obtener 37
- prototipo 47

## R

- RadioButton, componente 460
  - crear aplicaciones 461
  - eventos 464
  - métodos 464
  - parámetros 461
  - personalizar 462
  - propiedades 464
  - RadioButton, clase 463
  - utilizar 461
  - utilizar aspectos 463
  - utilizar estilos 462
- RDBMSResolver, componente 470
  - eventos 473
  - métodos 473
  - parámetros 471
  - propiedades 472
  - utilizar 471
- recursos, adicionales 10
- Remote Procedure Call (RPC), para
  - WebServiceConnector 647
- Remote Procedure Call, componente 481
- requisitos del sistema 8
- ruta de clase
  - global 684
  - local 684
- ruta de clases
  - aspectos básicos 684
  - cambiar 685
  - UserConfig, directorio 684
- ruta de clases local 684

## S

- ScrollPane, componente 499
  - crear aplicaciones 500
  - eventos 503
  - métodos 502
  - parámetros 500
  - personalizar 501
  - propiedades 502
  - ScrollPane, clase 501
  - utilizar 499
  - utilizar aspectos 501
  - utilizar estilos 501
- selección 27
- separator 397
- servicio Web, archivo WSDL 646
- servicios Web, clases
  - clases
  - servicios Web 622
- setSize() 23
- símbolo
  - nombre, para un componente personalizado 693
  - propietario, para un componente personalizado 693
- símbolo del componente, crear 686
- sintaxis, para sentencias de metadatos 694
- SOAPCall, clase 634
- SOAPFault 632, 644
- StyleManager, clase 540
- subclases, utilizar para sustituir aspectos 47
- subcomponentes, aplicar aspectos 44
- sugerencias para el código, activar 24

## T

- tabIndex 28
- tema Halo 39
- tema Sample 39
- temas 39
  - aplicar 40
  - crear 40
- términos utilizados en la documentación 10
- TextArea, componente 542
  - crear aplicaciones 543
  - eventos 546
  - parámetros 543
  - personalizar 544
  - propiedades 546
  - TextArea, clase 545
  - utilizar aspectos 545
  - utilizar estilos 544

- TextInput, componente 554
  - crear aplicaciones 556
  - eventos 559
  - métodos 558
  - parámetros 555
  - personalizar 556
  - propiedades 558
  - TextInput, clase 557
  - utilizar 555
  - utilizar estilos 556
- texto de información, para un componente
  - personalizado 709
- tipos de componente
  - Accordion 52
  - Administradores 51
  - Alert 66
  - Botón 75
  - CellRenderer 87
  - CheckBox 94
  - ComboBox 102
  - controles de la IU 49
  - DataGrid 166
  - DataHolder 198
  - DataSet 211
  - DateChooser 257
  - DateField 269
  - datos 50
  - Flash Professional 470, 677
  - Label 305
  - List 310
  - Loader 339
  - medios 51
  - Menu 392
  - MenuBar 423
  - NumericStepper 66, 257, 269, 392, 434
  - PopUpManager, clase 443
  - ProgressBar 445
  - Proveedor de datos (DataProvider) 200
  - RadioButton 460
  - RDBMSResolver 470
  - Remote Procedure Call 481
  - ScrollPane 499
  - StyleManager, clase 540
  - TextArea 542
  - TextInput 554
  - TransferObject 566
  - Tree 568
  - WebServiceConnector 646

- XMLConnector 668
- XUpdateResolver 677
- TransferObject, componente 566
  - métodos 566
- Tree, clase
  - propiedades 575
- Tree, componente 568
  - clase 574
  - crear aplicaciones 571
  - Formato XML 570
  - parámetros 571
  - personalizar 574
  - utilizar 569
  - utilizar aspectos 574
  - utilizar estilos 574

## U

- UIComponent, clase
  - definida 691

## V

- versión 1, componentes 29
  - actualizar 29
- versión 2, componentes
  - ventajas y descripción 12
  - y Flash Player 13
- vinculación de datos, clases 131
- vista
  - Menu, componente 395
- vista previa de componentes 20
- vista previa dinámica 20
  - para un componente personalizado 709

## W

- WebService, clase 638
- WebServiceConnector
  - multipleSimultaneousAllowed, parámetro 647
  - operation, parámetro 647
  - parámetros 647
  - resumen de eventos 649
  - resumen de métodos 649
  - resumen de propiedades 648
  - suppressInvalidCalls, parámetro 648
  - utilizar 647
  - WSDLURL, parámetro 648
- WebServiceConnector, componente 646

## **X**

### XML

- aplicación de formato para el componente Tree 570

### XMLConnector

- clase 669
- parámetros 669
- resumen de eventos 670
- resumen de métodos 670
- resumen de propiedades 669
- y esquemas 668

XMLConnector, componente 668

### XUpdate 677

XUpdateResolver, componente 677

- eventos 679
- parámetros 678
- propiedades 678
- utilizar 678

