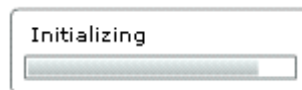


Mostrando el progreso de descarga de una aplicación

La clase [Application](#) soporta un [preloader](#) que utiliza una barra de progreso de descarga para mostrar la cantidad de bytes descargados o que quedan por descargar de un archivo SWF. Por defecto, el preloader de la aplicación está habilitado. El preloader muestra cuantos bytes ha sido descargados y actualiza constantemente la barra de progreso.

El progreso de la descarga muestra información de dos fases diferentes: la fase de descarga y la fase de inicialización de la aplicación. El evento “[Application.creationComplete](#)” finaliza el preloader.

Esta es la apariencia del progreso de descarga por defecto de las aplicaciones Flex.



La barra de progreso no se visualiza si el archivo SWF se encuentra en tu propio PC o si se encuentra “cacheado” previamente, de lo contrario, la barra de progreso se muestra si después de 700 milisegundos (0.7 segundos) se ha descargado menos de la mitad de la aplicación.

Para deshabilitar la barra de progreso establecemos a false la propiedad “[usePreloader](#)” del contenedor [Application](#).

```
<mx:Application
  xmlns:mx="http://www.adobe.com/2006/mxml"
  usePreloader="false">
```

Creando una barra de progreso propia

Por defecto, la precarga de la aplicación utiliza la clase [DownloadProgressBar](#) del paquete [mx.preloaders](#) para mostrar la barra de progreso. Para crear una barra de progreso propia, puedes crear una subclase de [DownloadProgressBar](#) o crear una subclase de [flash.display.Sprite](#) que implemente el interface [mx.preloaders.IPreloaderDisplay](#). Puedes implementar una barra de progreso como un componente SWC o un componente ActionScript. Una barra de progreso propia que extienda la clase [Sprite](#) no debería utilizar ninguno de los componentes estándar de Flex porque la carga sería muy lenta. Tampoco deberías implementar la barra de progreso como un componente MXML porque de igual forma la carga sería demasiado lenta.

Para utilizar una barra de progreso propia, debes asignar a la propiedad “[preloader](#)” del contenedor Application la vía del componente SWC o del componente ActionScript. Un componente SWC debe estar ubicado en el mismo directorio que el archivo MXML o en un directorio indicado en el classpath de tu aplicación. Un componente ActionScript puede estar ubicado en uno de esos directorios o en un subdirectorio de éstos.

El código fuente en el siguiente ejemplo especifica una barra de progreso propia llamada “CustomBar” que está ubicada en el directorio “mycomponents/mybars” por debajo del directorio raíz de la aplicación:

```
<mx:Application
  xmlns:mx="http://www.adobe.com/2006/mxml"
  preloader="mycomponents.mybars.CustomBar">
```

Eventos del progreso de la descarga

La operación de descarga se producen una serie de eventos, éstos son enviados por la clase [Preloader](#). Una barra de progreso propia debe manejar estos eventos.

La tabla siguiente describe los eventos:

ProgressEvent.PROGRESS	Enviado cuando el archive SWF comienza a ser descargado a nuestro ordenador.
Event.COMPLETE	Enviado cuando el archivo SWF ha sido descargado totalmente. Este evento lo vamos a recibir ninguna o una vez.
FlexEvent.INIT_COMPLETE	Enviado cuando la aplicación finaliza la fase de inicialización. Este evento solamente se envía una vez y es el último que envía la clase Preloader. La barra de progreso debe enviar un evento COMPLETE después de recibir un evento INIT_COMPLETE. El evento COMPLETE notifica al Preloader que la barra de progreso ha completado todas las operaciones y puede finalizar. La barra de progreso, después de recibir el evento INIT_COMPLETE y antes de enviar el evento COMPLETE, puede realizar tareas adicionales como mostrar una animación. El envío del evento COMPLETE debería ser la última acción de la barra de progreso.
FlexEvent.INIT_PROGRESS	Enviado cuando la aplicación completa la fase de inicialización.
RslEvent.RSL_ERROR	Enviado cuando falla la carga de una RSL (Runtime Shared Library).
RslEvent.RSL_LOADED	Enviado cuando finaliza la carga de una RSL .
RslEvent.RSL_PROGRESS	Enviado cuando una RSL comienza a descargarse a nuestro ordenador. El primero de los eventos indica el inicio de la descarga. La cantidad de bytes descargados se puede obtener de las propiedades rslIndex y rslTotal de la clase RSLEvent .

La clase [DownloadProgressbar](#) define un oyente de evento (event listener) para cada uno de éstos eventos. Cuando sobrescribes esta clase, puedes opcionalmente sobrescribir el comportamiento por defecto del oyente del evento. Si creas una barra de progreso propia como una subclase de la clase [Sprite](#), debes definir un oyente de evento de cada uno de éstos eventos.

Creando una subclase simple de la clase DownloadProgressBar

La manera más fácil de crear tu propia barra de progreso de descarga es crear una subclase de la clase [mx.preloaders.DownloadProgressBar](#) y modificarla en función de los requerimientos de tu aplicación.

Puedes definir textos propios o establecer el tiempo mínimo que la barra debe mostrarse. He aquí un ejemplo:

```
package myComponents
{
    import mx.preloaders.*;
    import flash.events.ProgressEvent;
    public class DownloadProgressBarSubClassMin extends DownloadProgressBar
    {
        public function DownloadProgressBarSubClassMin()
        {
            super();
            // Establecer el texto de la fase de descarga.
            downloadingLabel="Cargando aplicación..."
            // Establacer el texto de la fase de inicialización.
            initializingLabel="Inicializando aplicación..."
            // Tiempo mínimo que se visualiza el mensaje (2 segundos).
            MINIMUM_DISPLAY_TIME=2000;
        }
        // Sobreescribir método que indica si la barra ha de mostrarse
        // en el proceso de inicialización.
        override protected function showDisplayForInit(elapsedTime:int,
        count:int):Boolean {
            return true;
        }
        // Sobreescribir el método que indica si la barra de progreso
        // ha de mostrarse en la fase de descarga.
        override protected function showDisplayForDownloading(elapsedTime:int,
        event:ProgressEvent):Boolean {
            return true;
        }
    }
}
```

Puedes utilizar esta clase en la aplicación Flex como indica este ejemplo:

```
<?xml version="1.0"?>
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
    preloader="myComponents.DownloadProgressBarSubClassMin">

    <mx:Button/>
    <mx:TextInput
        text="sub class min" />
</mx:Application>
```

Creando una subclase de la clase DownloadProgressBar

En el siguiente ejemplo se crea una subclase de la clase [DownloadProgressBar](#) para mostrar textos que indican el estado de la descarga y de la inicialización de la aplicación.

Este ejemplo define oyentes de eventos enviados por la barra de progreso de descarga para escribir los textos en los objetos [flash.text.TextField](#).

```
package myComponents
{
import flash.display.*;
import flash.text.*;
import flash.utils.*;
import flash.events.*;
import mx.preloaders.*;
import mx.events.*;

    public class MyDownloadProgressBar extends DownloadProgressBar
    {
        // Definir un TextField para los mensajes que
        // describen el progreso de descarga de la aplicación
        private var progressText:TextField;
        // Definir un TextField para el texto final después
        // de la inicialización de la aplicación.
        private var msgText:TextField;
        public function MyDownloadProgressBar()
        {
            super();
            // Configurar el TextField para los mensajes de progreso
            progressText = new TextField();
            progressText.x = 10;
            progressText.y = 90;
            progressText.width = 400;
            progressText.height = 400;
            addChild(progressText);
            // Configurar el TextField para el texto final.
            msgText = new TextField();
            msgText.x = 10;
            msgText.y = 10;
            msgText.width = 400;
            msgText.height = 75;
            addChild(msgText);
        }
        // Definir los oyentes para los eventos del preloader
        override public function set preloader(preloader:Sprite):void {
            // Escuchar los eventos relevantes
            preloader.addEventListener(
                ProgressEvent.PROGRESS, myHandleProgress);
            preloader.addEventListener(
                Event.COMPLETE, myHandleComplete);
            preloader.addEventListener(
                FlexEvent.INIT_PROGRESS, myHandleInitProgress);
            preloader.addEventListener(
                FlexEvent.INIT_COMPLETE, myHandleInitEnd);
        }
        // Oyente para el evento ProgressEvent.PROGRESS.
        private function myHandleProgress(event:ProgressEvent):void {
            progressText.appendText("\n" + "Progreso cargados: " +
                event.bytesLoaded + " bytes de: " + event.bytesTotal);
        }
        // Oyente para el evento Event.COMPLETE.
        private function myHandleComplete(event:Event):void {
            progressText.appendText("\n" + "Completado");
        }
    }
}
```

```
// Oyente para el evento FlexEvent.INIT_PROGRESS.
private function myHandleInitProgress(event:Event):void {
    progressText.appendText("\n" + "Comienza la inicialización de la
aplicación.");
}
// Oyente para el evento FlexEvent.INIT_COMPLETE.
private function myHandleInitEnd(event:Event):void {
    msgText.appendText("\n" + "Inicialización finalizada");
    var timer:Timer = new Timer(2000,1);
    timer.addEventListener(TimerEvent.TIMER, dispatchComplete);
    timer.start();
}

// Oyente para el Timer, pausar el proceso
// para dar tiempo a leer los textos del progress bar.
private function dispatchComplete(event:TimerEvent):void {
    dispatchEvent(new Event(Event.COMPLETE));
}
}
```

Puedes utilizar esta clase en la aplicación Flex como indica este ejemplo:

```
<?xml version="1.0"?>
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
    preloader="myComponents.MyDownloadProgressBar">

    <mx:Button/>
    <mx:TextInput/>
</mx:Application>
```

Creando una subclase de Sprite

Implementando una subclase de la clase [Sprite](#) puedes modificar completamente la apariencia del proceso de descarga. Un uso común de éste método es mostrar un archivo SWF durante la fase de inicialización, por ejemplo podrías mostrar un SWF con un reloj con las manecillas desplazándose.

El siguiente ejemplo muestra un archivo SWF como progreso de descarga, esta clase debe implementar el interface [IPreloaderDisplay](#).

```
package myComponents
{
import flash.display.*;
import flash.utils.*;
import flash.events.*;
import flash.net.*;
import mx.preloaders.*;
import mx.events.*;
    public class MyDownloadProgressBarSWF extends Sprite implements IPreloaderDisplay
    {
        // Definir un control Loader para cargar el archive SWF.
        private var dpbImageControl:flash.display.Loader;
        public function MyDownloadProgressBarSWF() {
            super();
        }
    }
}
```

```
// Especificar los oyentes de eventos.
public function set preloader(preloader:Sprite):void {
    // Escuchar los eventos relevantes.
    preloader.addEventListener(
        ProgressEvent.PROGRESS, handleProgress);
    preloader.addEventListener(
        Event.COMPLETE, handleComplete);
    preloader.addEventListener(
        FlexEvent.INIT_PROGRESS, handleInitProgress);
    preloader.addEventListener(
        FlexEvent.INIT_COMPLETE, handleInitComplete);
}

// Inicializar el control Loader en el método initialize
// del interface IPreloaderDisplay
public function initialize():void {
    dpbImageControl = new flash.display.Loader();
    dpbImageControl.contentLoaderInfo.addEventListener(
        Event.COMPLETE, loader_completeHandler);
    dpbImageControl.load(new URLRequest("assets/dpbSWF.swf"));
}

// Después de cargar el SWF, establecer el tamaño.
private function loader_completeHandler(event:Event):void {
    addChild(dpbImageControl);
    dpbImageControl.width = 50;
    dpbImageControl.height = 50;
    dpbImageControl.x = 100;
    dpbImageControl.y = 100;
}

// Definir los oyentes vacíos.
private function handleProgress(event:ProgressEvent):void {
}
private function handleComplete(event:Event):void {
}
private function handleInitProgress(event:Event):void {
}
private function handleInitComplete(event:Event):void {
    var timer:Timer = new Timer(2000,1);
    timer.addEventListener(TimerEvent.TIMER, dispatchComplete);
    timer.start();
}
private function dispatchComplete(event:TimerEvent):void {
    dispatchEvent(new Event(Event.COMPLETE));
}

// Implementar el interface IPreloaderDisplay
public function get backgroundColor():uint {
    return 0;
}
public function set backgroundColor(value:uint):void {
}
public function get backgroundAlpha():Number {
    return 0;
}
public function set backgroundAlpha(value:Number):void {
}
public function get backgroundImage():Object {
    return undefined;
}
public function set backgroundImage(value:Object):void {
}
public function get backgroundSize():String {
    return "";
}
}
```

```
        public function set backgroundColor(value:String):void {  
        }  
        public function get stageWidth():Number {  
            return 200;  
        }  
        public function set stageWidth(value:Number):void {  
        }  
        public function get stageHeight():Number {  
            return 200;  
        }  
        public function set stageHeight(value:Number):void {  
        }  
    }  
}
```