FLEX - IMAGE CONTROL

Introduction

The Image control lets you import JPEG, PNG, and GIF files at runtime. You can also embed any of these files at compile time by using @Embed*source = 'filename'*.

Embedded images load immediately, because they are compiled with of the Flex SWF file. However, they add to the size of the application and slow down the application initialization process. Embedded images also require you to recompile your applications whenever your image files change.

You can load an image from the local file system in which the SWF file runs, or you can access a remote image, typically though an HTTP request over a network. These images are independent of your Flex application, so you can change them without causing a recompile operation as long as the names of the modified images remain the same. The referenced images add no additional overhead to an application's initial loading time.

Class declaration

Following is the declaration for **spark.components.Image** class:

```
public class Image
   extends SkinnableComponent
```

Public properties

S.N.	Property & Description
1	**bitmapData : BitmapData** [read-only] Returns a copy of the BitmapData object representing the currently loaded image content *unscaled*.
2	**bytesLoaded : Number** [read-only] The number of bytes of the image already loaded.
3	**bytesTotal : Number** [read-only] The total image data in bytes loaded or pending load.
4	**clearOnLoad : Boolean** Denotes whether or not to clear previous image content prior to loading new content.
5	**contentLoader : IContentLoader** Optional custom image loader
6	**contentLoaderGrouping : String**

Optional content grouping identifier to pass to the an associated IContentLoader instance's load method.

7

fillMode : String

Determines how the bitmap fills in the dimensions.

8

horizontalAlign : String

The horizontal alignment of the content when it does not have a one-to-one aspect ratio and scaleMode is set to mx.graphics.BitmapScaleMode.LETTERBOX.

9

preliminaryHeight : Number

Provides an estimate to use for height when the "measured" bounds of the image is requested by layout, but the image data has yet to complete loading.

10

preliminaryWidth : Number

Provides an estimate to use for width when the "measured" bounds of the image is requested by layout, but the image data has yet to complete loading.

11

scaleMode : String

Determines how the image is scaled when fillMode is set to mx.graphics.BitmapFillMode.SCALE.

12

smooth : Boolean

Specifies whether to apply a smoothing algorithm to the bitmap image.

13

source : Object

The source used for the bitmap fill.

14

sourceHeight : Number

[read-only] Provides the unscaled height of the original image data.

15

sourceWidth : Number

[read-only] Provides the unscaled width of the original image data.

16

trustedSource : Boolean

[read-only] A read-only flag denoting whether the currently loaded content is considered loaded form a source whose security policy allows for cross domain image access.

17

verticalAlign : String

The vertical alignment of the content when it does not have a one-to-one aspect ratio and scaleMode is set to mx.graphics.BitmapScaleMode.LETTERBOX.

Public methods

S.N.	Method & Description
1	**Image** Constructor.

Events

S.N.	Events & Description
1	**complete** Dispatched when content loading is complete.
2	**httpStatus** Dispatched when a network request is made over HTTP and Flash Player can detect the HTTP status code.
3	**ioError** Dispatched when an input or output error occurs.
4	**progress** Dispatched when content is loading.
5	**ready** Dispatched when content loading is complete.
6	**securityError** Dispatched when a security error occurs.

Methods inherited

This class inherits methods from the following classes:

- spark.components.supportClasses.SkinnableComponent

- mx.core.UIComponent
- mx.core.FlexSprite
- flash.display.Sprite
- flash.display.DisplayObjectContainer
- flash.display.InteractiveObject
- flash.display.DisplayObject
- flash.events.EventDispatcher
- Object

Flex Image Control Example

Let us follow the following steps to check usage of Image control in a Flex application by creating a test application:

Step	Description
1	Create a project with a name *HelloWorld* under a package *com.tutorialspoint.client* as explained in the *Flex - Create Application* chapter.
2	Create a foler *assets* in HelloWorld application root folder *HelloWorld*.
3	Download a sample image flex-mini.png to a folder *assets* under HelloWorld folder.
4	Modify *HelloWorld.mxml* as explained below. Keep rest of the files unchanged.
5	Compile and run the application to make sure business logic is working as per the requirements.

Following is the content of the modified mxml file **src/com.tutorialspoint/HelloWorld.mxml**.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
   xmlns:s="library://ns.adobe.com/flex/spark"
   xmlns:mx="library://ns.adobe.com/flex/mx"
   width="100%" height="100%" minWidth="500" minHeight="500"
   applicationComplete="init(event)" >
   <fx:Style source="/com/tutorialspoint/client/Style.css"/>
   <fx:Script>
   <![CDATA[
      import mx.controls.Alert;
      import mx.events.FlexEvent;

      [Bindable]
      [Embed(source="assets/flex-mini.jpg")]
      private var flexImage:Class;

      protected function init(event:FlexEvent):void
      {
         dynamicImage.source =
         "http://www.tutorialspoint.com/images/flex-mini.png";
      }

      ]]>
   </fx:Script>
   <s:BorderContainer width="550" height="600"
   styleName="container">
      <s:VGroup width="100%" height="100%" gap="50" horizontalAlign="center"
      verticalAlign="middle">
```

```
         <s:Label
         fontSize="40" color="0x777777" styleName="heading"/>
         <s:Panel title="Using Embeded Image" width="420" height="200" >
         <s:Image source="{flexImage}" />
         </s:Panel>
         <s:Panel title="Using Image from URL" width="420" height="200" >
            <s:Image  />
         </s:Panel>
      </s:VGroup>
   </s:BorderContainer>
</s:Application>
```

Once you are ready with all the changes done, let us compile and run the application in normal mode as we did in Flex - Create Application chapter. If everything is fine with your application, this will produce following result: [Try it online]