print this article | close this window

Flash* and .NET* Integration Using ASP.NET

Introduction

By Vinod Raghupathy,
Sr. Application Developer

Learn how to integrate these two powerful technologies, adding database connectivity and more robust interactivity to Flash movies and applications.

Macromedia Flash* has been widely used on the Internet to enhance the content presentation of Web sites. The last releases of Macromedia Flash (Flash 6 and Flash MX*) have made it clear that Macromedia is serious about making Flash the interface of choice for Web application development. For an interface, Flash can be a great option; however, it is not an option to replace business logic, query management, and security. For robust performance and enterprise capabilities, you need .NET*. Flash, with rich presentation support, together with Microsoft's .NET technology can make a potent combination for Web developers.

Flash includes ActionScript*, a scripting language to add interactivity to movies. Beginning with ActionScript's introduction in Flash 5, Macromedia advanced it from a simple scripting language to one with solid but still limited programming features. With Flash MX 2004, Macromedia introduced ActionScript 2.0, which supports new capabilities and new ways in structuring and working with code. To get the most from this article, you should first read about and understand the basics of ActionScript* for building a Flash site.

Macromedia has introduced ways to easily integrate external data into Flash applications to allow for more complex and robust data-driven applications. Both Flash and .NET promote themselves as cross-platform solutions and, as such, are natural partners. Still, connecting the two technologies is no simple task. This article talks about some of the methodologies for integrating Flash and .NET applications.

Integrating Flash and ASP.NET

Embedding Flash Movies in ASP.NET Pages

Macromedia Flash movies are placed inside .aspx pages using a combination of **OBJECT** and **EMBED** tags. **EMBED** tags were initially developed by Netscape* to work with Netscape 2.0 or higher. Later, the tags were adopted by Microsoft Internet Explorer* (MSIE). For browser compatibility, use both the MSIE-specific **OBJECT** tag and the **EMBED** tag. This HTML code demonstrates embedding a Flash movie (*.swf file) with both the **OBJECT** and **EMBED** tags:

```
<body MS_POSITIONING="GridLayout">

  <form id="Form1" method="post" runat="server">
      <OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
          codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=5,0,0,0"
          id="id1" WIDTH=100% HEIGHT=100% VIEWASTEXT>
          <PARAM NAME=movie VALUE="test.swf">
          <PARAM NAME=quality VALUE=high>
          <PARAM NAME="menu" VALUE="false" />
          <EMBED src="test.swf" quality=high WIDTH=100% HEIGHT=100% TYPE="application/x-shockwave-flash"
           PLUGINSPAGE="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash">
          </EMBED>
      </OBJECT>
  </form>

</body>
```

Click here for larger image

Connecting Flash to External Data

There are three ways to integrate external data with Flash movies: **FlashVars**, **LoadVars**, and the new FlashRemoting technology. The simplest of these approaches is to pass external data from an .aspx page to a Flash movie embedded in a Web page by passing variables in the HTML portion of the code using **FlashVars**. The variable values set using **FlashVars** are accessible in the root level of the Flash movie. The **FlashVars** property must be assigned in both the **OBJECT** and **EMBED** tags in order to work for all browsers.

This shows **FlashVars** with an object tag:

```
<PARAM Name="FlashVars" Value="init=yes&check=true">
```

This shows **FlashVars** being used with the **EMBED** tag:

```
<EMBED src="test.swf" FlashVars="init=yes&check=true"></EMBED>
```

For more information on **FlashVars** see this article from Macromedia:

http://www.macromedia.com/cfusion/knowledgebase/index.cfm?id=tn_16417*

Why FlashVars

Without **FlashVars**, variables can be passed to a Flash movie through HTML tags by appending them to the URL of the .SWF file as a query string. This method can be a problem if the variables are needed immediately on load of the .SWF file, because an .SWF file should be downloaded and played before data is requested from the server. Variable values can also be set in Flash through the “Load Variables” method and through JavaScript*. Each of these methods has drawbacks in terms of sluggish response; that’s why developers should use **FlashVars** to pass data from ASP.NET to the Flash movie. This figure shows the code for passing variables as a query string:

```
<form id="Form1" method="post" runat="server">
  <OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=5,0,0,0"
    id="id1" WIDTH=100% HEIGHT=100% VIEWASTEXT>
    <PARAM NAME=movie VALUE="test.swf">
    <PARAM NAME=quality VALUE=high>
    <PARAM NAME="menu" VALUE="false" />
    <PARAM NAME=FlashVars VALUE="init=yes&check=true">
    <EMBED src="test.swf" FlashVars="init=yes&check=true" quality=high WIDTH=100% HEIGHT=100%
    TYPE="application/x-shockwave-flash"
    PLUGINSPAGE="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash">
    </EMBED>
  </OBJECT>
</form>
```

Click here for larger image

Any variables passed using **FlashVars** can be accessed at the root level (**_level0**) of the Flash movie. To retrieve the values within a Flash movie, create a dynamic variable. This ActionScript code assigns the “**initvalue**” in Flash and the value for the Flash variable:

```
_level0.initvalue = init;
```

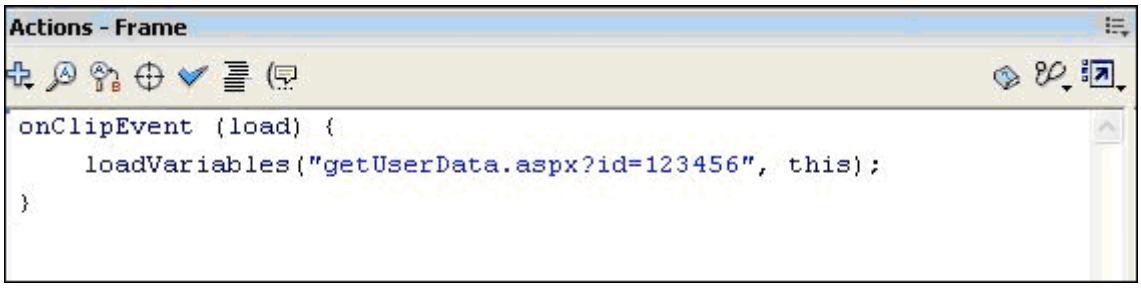
Communication From Flash to a Database

A Flash movie cannot access a database directly. However, Flash can communicate to database via any server-side technology like ASP or ASP.NET. There are several methodologies to accomplish this. In this article we will explain the **LoadVariables** method and the **sendAndLoad** method.

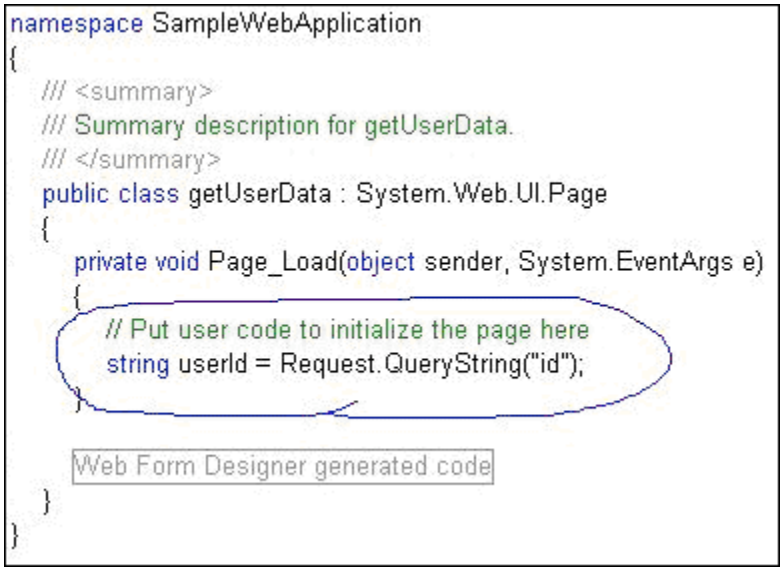
Using the loadVariables method from a movie clip

The movie clip's **loadVariables** method reads data from an external file or URL and sets the values for the variables of the movie clip. The **load** action of the movie clip's **onClipEvent** event handler is initiated as soon as the movie clip is instantiated. The **loadVariables** method can be called to load the variables when the movie clip is initiated. The **loadVariables** method is an asynchronous call and Flash will not wait for the results of the call. The **data** action of the movie clip's **onClipEvent** event handler is initiated when data is received in a **loadVariables()** action.

To demonstrate the usage of the method, this example retrieves the user's name and age from a database depending on the ID selected in the Flash movie. In the figure below, the movie's **loadVariables** method is used to load data into the movie clip from an .aspx page, **getUserData.aspx**. Here, the ActionScript passes the **id** of the user whose **name** and **age** are to be displayed in the Flash movie as a query string to the .aspx page.



The **getUserData.aspx** page retrieves the value of **id** from the query string as shown in figure below:



ASP.NET can now pass this value of **id** to .NET business components. This example retrieves the user's **name** and **age**:

```
getUserData.aspx.cs
SampleWebApplication.getUserData
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace SampleWebApplication
{
    /// <summary>
    /// Summary description for getUserData.
    /// </summary>
    public class getUserData : System.Web.UI.Page
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
            // Put user code to initialize the page here
            string userId = Request.QueryString("id");

            //Call the .NET Business component to retrieve User Data
            UserData userdata = new UserData();
            User = userdata.getData(userId);

            //Set data to flash movie
            Response.Write("name=" & User.name);
            Response.Write("age=" & User.age);

        }
```

As mentioned earlier in this section, the **data** action of the movie clip's **onClipEvent** event handler is initiated when data is received in a **loadVariables()** action. Declare the following dynamic variables in the Flash movie: **flashName** and **flashAge**. The following figure shows the ActionScript code that is used in the Flash movie's **onClipEvent** handler to retrieve the values passed back from the .aspx page.

Note: It is very important that the variables accessed within the Flash movie are the same as the ones being set from the .aspx page. In the example demonstrated here, the variable value that is accessed within the Flash movie is "**name**", which is the same as the one being set from the .aspx page (**Response.Write("name=" & User.name)**).

```
onClipEvent (data) {
    //name - variable set from ASP.NET page
    //age - age variable set from ASP.NET page
    flashName = name;
    flashAge = age;
}
Layer 1 : 1
```

These values can now be used within the Flash movie.

Using the sendAndLoad method from a movie clip

In today's Internet, XML's potential as a universal transfer format allowing even applications of different types to exchange data smoothly holds as much promise as its role as a document system. XML provides a gateway for communication between applications – even applications on wildly different systems. As long as applications can share data (through HTTP, file sharing, or another mechanism), and have an XML parser, they can share structured information that is easily processed.

Flash MX supports XML Data Transfer and has several ways of exchanging XML with server technologies like ASP and ASP.NET. Flash MX can send and receive XML data packets using the XML classes **load**, **send**, and **sendAndLoad** or Flash MX's new **XMLConnector**. The last of these can be used to send and receive packets over HTTP and bind them to UI components on the Flash stage.

This section discusses the Flash XML object's **sendAndLoad** method. This is the preferred approach in terms of the architecture oriented around XML data communication, exception handling, and performance. We use this simple example: inserting a user's score details into the database from a Flash client and passing back a success or failure message to the Flash client.

Creating an XML Packet in Flash

An XML data packet can be easily created in Flash as shown below:

var sendXML:XML = new XML("<userid>123456</userid>");

Sending an XML packet to a Server Side page that uses ASP/ASP.NET and handling the return XML data packet

Usage:

> **xmlObject.sendAndLoad (URL: String, targetXMLObject: XML): Void**
>
> The '**xmlObject**' is the XML object instance that encapsulates the XML that has to be passed on to the .aspx page.
>
> The '**URL**' is the destination URL for the specified XML object.
>
> The '**targetXMLObject**' is the XML object that will receive the return information from server.

The **sendAndLoad** method works by sending an XML packet to a server-side template and receiving an XML packet as a response, which is different from the XML class **send** method which just sends an XML packet and doesn't expect any sort of response. Each of these methods is useful in their own way but have one significant difference: the XML class **send** method takes a target as a parameter which allows you to open a new browser window or replace the content in the current browser window, whereas the **sendAndLoad** method takes a target XML packet as a parameter which won't launch a new browser window. Because the **send** method opens a new browser window, we can see the output in the browser window itself which could be useful for debugging.

Note: If the Flash file issuing the **sendAndLoad** call is running in a Web browser, the destination URL should be in the same domain as the Flash movie. If you still require to communicate with an application from a different domain, check for more information on "Cross Domain Data Loading" at www.macromedia.com*.

The following figure shows how the sendAndLoad method of an XML object is used to communicate with an .aspx page and how the XML returned from the server can be handled:

```
Actions - Frame
//Creating the XML object to pass the userscore to the database
var userid_str:String = "<user userid=\""+userid_txt.text+"\" score=\""+score_txt.text+"\" />";
var userscore_xml:XML = new XML(userid_str);

//Creating the responsexml object to handle the return information from server
var ResponseXMLObj_xml:XML = new XML();
ResponseXMLObj_xml.ignoreWhite = true;
//Set the function that would check for response
ResponseXMLObj_xml.onLoad = myOnLoad;

//Calling the sendAndLoad method to pass information to the server
userscore_xml.sendAndLoad("http://localhost/MyTestApp/scoring.aspx", ResponseXMLObj_xml);

function myOnLoad(success:Boolean) {
   if (success) {
      //Logic for parsing the XML return string from server
      //Return a status node in the XML for success and failure and
      //--handle it accordingly in flash. Any database exceptions from server
      // side pages can also be handled by returning a proper failure code in the return XML.
   }
}
```

Click here for larger image

The above code does the following:

- Loads the XML string to an XML object "**userscore_xml**". This instance holds the XML that is being sent to the remote server-side ASP.NET page.
- Sets the **ignoreWhite** property for the XML object "**ResponseXMLObj_xml**". This XML object instance holds the XML packet returned from the server.
- Calls the XML object's ("userscore_xml") **sendAndLoad** method to send the XML to the server-side .aspx page, **http://localhost/MyTestApp/scoring.aspx** and store the returned XML packet into the XML object instance "ResponseXMLObj_xml".
- Sets the onLoad method of the XML object "**ResponseXMLObj_xml**". Flash receives the returned XML response from the server-side .aspx page, which is handled in the XML object's **onLoad** method.

It then parses the received XML packet to display the necessary message within the Flash client.

Receiving the XML packet send from Flash in an ASPX page

The following figure shows the code to achieve this:

```
/// Summary description for GetDataFromFlash.
/// </summary>
public class GetDataFromFlash : System.Web.UI.Page
{
    private void Page_Load(object sender, System.EventArgs e)
    {
        // Put user code to initialize the page here
        XmlDocument xmlDoc = null;
        Result result = new Result();
        xmlDoc = new XmlDocument();
        //Load XML from the flash movie. This XML document object should be parsed
        //to retrieve the information sent from flash
        xmlDoc.Load(Request.InputStream);

        //Pass the XML document to an XML parsing component so that it could be inserted
        //to the database.
        FlashXmlProcesor flashXmlProcessor = new FlashXmlProcesor();
        flashXmlProcessor.ProcessXml(xmlDoc);

        //Sending the result back to flash.
        Response.ContentType = "text/xml";
        //The XmlHelper.Serialize method is a function that would
        //serialize an input object to return back an XML string.
        //The Result class defines the XML structure of the return XML
        //that has to be returned back to flash.
        Response.Write(XmlHelper.SerializeXml(typeof(Result),result));
    }
```

Click here for larger image

The above code does the following:

- Receives the XML sent from the Flash client. (**xmlDoc.Load(Request.InputStream)**)
- Parses the XML received and passes it to the business component to insert into the database.
- Forms the XML message to be passed back to the Flash client. The **Success** and **Exception** conditions raised in the business layer are handled accordingly and transformed to respective XML messages, which are passed back to the Flash client. The Flash client receives these XML messages and displays them accordingly.

Conclusion

Flash and .NET technologies continue to gain increasing popularity in the market and together will very likely make a successful partnership in a growing number of application architectures. The two technologies tackle some of the same issues, but are essentially different. Flash MX makes Web site experiences interactive while .NET helps build enterprise applications. As explored in this article, it is possible to integrate Flash and .NET applications. The first step is making a decision on how to connect the two together and to strategize the right architecture.

Though new architectures around FlashRemoting are gaining momentum in this area, the methodologies discussed in this article will remain viable for some years to come.

References and Related Links

References

http://www.macromedia.com/devnet*

http://www.macromedia.com/support/flash/action_scripts/actionscript_tutorial/*

http://www.macromedia.com/cfusion/knowledgebase/index.cfm?id=tn_16417*

Related Links

http://www.intel.com/cd/ids/developer/asmo-na/eng/57382.htm

http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/code/211657.htm

http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/code/19121.htm

About the Author

Vinod Raghupathy works as a Team Lead with the Intel® Learning Network. He has over 10 years of experience in industrial automation and technologies like Java*, .NET, and Flash. Vinod has extensive experience in strategizing/architecting solutions in .NET. Today, Vinod is completely focused on Microsoft .NET Technologies like C#, ASP.NET, and emerging technologies like FlashRemoting.