By icarus

This article copyright Melonfire 2000−2002. All rights reserved.

Table of Contents

Extreme Reactions.....1

Welcome To The Matrix.....2

Access Granted.....5

Alien Invasion.....7

Loading Up.....10

Coming Back For More.....12

A Fine Balance.....14

Extreme Reactions

"Flash forms!", yelled the Developer, tugging at his goatee in exasperation. "Flash forms? Flash? Forms? Are you nuts? Has you brain gone AWOL? Do you know how long they'll take to load up? And how slow they'll be? And...listen, go away and bug someone else, willya? Some of us have work to do!"

If your reaction to the thought of building Web−based forms in Flash is something similar, you probably don't want to read the rest of this article. Close your browser, take a pill (in fact, take a couple − you'll sleep better) and consider signing up for a short course in yoga.

If, on the other hand, the thought of using Flash to collect user input intrigues you, keep reading. Over the next few pages, I'll be showing you how you can build Flash movie clips that also have the ability to communicate with server−side scripts, both to send and retrieve data. And no, it won't give you heartburn.

Welcome To The Matrix

Before we get started with building a Flash form, you need to know a little theory.

Most HTML forms send the data they collect from the user to a server−side script, which is actually responsible for processing the data. This server−side script, usually written in a language like Perl or PHP, actually takes care of doing something useful with the data − for example, using it to build a database query, writing it to a file, and/or generating a result page.

Flash forms work much like HTML forms, except that the constructs used to build them are a little different. A Flash form can be programmed to connect to a specific server−side script, and to pass form variables to it via either GET or POST methods.

In order to better understand this, let's consider a simple example. Pop open Flash, create a new movie and then create a new Button symbol. Name it "textBox".

In the Symbol Editor, define a rectangular textbox,

and then use the Windows −> Panels −> Text Properties panel to turn it into an input text box. Set a variable name for this text box as well − I've called mine "name".

This is the form variable which will store the user's input, and which will be passed forward to the server−side script.

Back in the Stage, drag and drop a copy of this symbol into the workspace. Add some descriptive text above it if you like.

Next, it's time to connect the text box with a server−side script. Since I want the server−side script to be invoked when the Flash form is submitted (or, in this case, when the user enters some data into the text box and hits the Enter key), I need to add some ActionScript to the symbol. Right−click the symbol instance, pop up the ActionScript editor and add the following code to it:

```
on (keyPress "<Enter>") {
getURL ("matrix.php", "", "GET");
}
```

This is fairly self−explanatory, even if you've never programmed in ActionScript before. When the form input box receives an Enter keypress, it will invoke the URL "matrix.php", and submit the form data to that URL via the GET method. It is then up to the server−side script to decide what to do next.

Let's see what "matrix.php" looks like:

```
<html>
<body>
Welcome to the Matrix, <? echo $_GET['name'];?>
</body>
</html>
```

Extremely simple, this − all it does is accept the form data from the Flash form, and print a simple Web page with a message incorporating the form data. Note that the script references the data entered by the user into the form input box via its variable name − you'll remember that I defined this variable, "name", a couple steps back when first creating the symbol.

Export the movie as a Flash SWF file, place it in an appropriate location under your Web server root, and access it via your browser. Here's what you should see:

Enter some data, and hit the Enter key. The Flash form will submit your entered data to the "matrix.php" script, and you'll see something like this:

```
Welcome to the Matrix, john
```

Access Granted

Obviously, that was a very primitive example, though one that did demonstrate the concept of integrating user−supplied data with a server−side script via a Flash movie. Now, how about a twist?

This next example uses the data entered into a Flash form to query a database and return a list of matches. In order to create the form, create a new Button symbol, in much the same way as you created the one on the previous page. Name the form variable within the symbol "q", and add a copy of the instance to the Flash movie.

Once that's done, add some ActionScript to the symbol instance, referencing the server−side script "search.php" − this time, use POST to send the form variables to the script.

```
on (keyPress "<Enter>") {
getURL ("search.php", "", "POST");
}
```

All that's left now is to write the PHP script "search.php". Here's what mine looks like:

```
<html>
<head><basefont face="Arial"></head>
<body>

<h2>Search Results</h2>
<ol>
<?

// search.php

// get query term
$query = trim($_POST['q']);

// open connection to database
$connection = mysql_connect("localhost", "user", "pass") or
die ("Unable
to connect!");
```

```
mysql_select_db("db20139a") or die ("Unable to select
database!");

// formulate and execute query
$query = "SELECT * FROM data WHERE tag LIKE '%$query%' OR
keyword LIKE
'%$query%'"; $result = mysql_query($query) or die("Error in
query: " .
mysql_error());

// get record
while($row = mysql_fetch_object($result))
{
echo "<li>";
echo "<a href=go.php?id=$row->id>$row->tag</a>";
echo "<br>$row->title";
echo "<p>";
}

// clean up
mysql_close($connection);
?>

</ol>
</body>
</html>
```

Here's what the search box looks like,

and here's a sample results page.

Alien Invasion

If you're using the latest version of Flash, Flash MX, you'll be happy to hear that Macromedia's included a bunch of new form controls in it. These controls allow you to build ever more complex forms, including forms containing list boxes, radio buttons and check boxes.

To access these, use the Window −> Components command to pop up a panel containing these new form components.

Let's build a simple form using some of these components. Pop open a new Flash movie, and add a couple of static text boxes to it

Next, add form input fields next to each of these − name the corresponding variables "name" and "species" respectively.

How about checking the species type as well? Drag a couple of radio button instances from the Components panel into your form,

and click on each one to set the properties for it.

In the Properties panel that opens up, click the Parameters tab and set the label for each radio button.

The label attached to the radio button instance on the Stage should change simultaneously.

Also set the group name for the radio buttons. Since this is an either−or choice, you should make sure that both radio buttons have the same group name − I've used "speciesType".

Next, how about adding a list box? Drag and drop a copy of the Combo Box symbol from the Components panel into the Stage. Name it "residence".

At this point, the box doesn't have any values in it − but you can fix that by popping open its Properties panel, going to the Parameters section, and filling up the Labels and Data sections. The Labels section contains the items to be displayed within the box, while the Data section contains the corresponding values to be sent to the server−side script.

And here's what the combo box should look like once you're done.

Finally, we need a button to actually submit the form. Pull out a Push Button symbol instance from the Components panel, and add it to your almost−complete form. Alter the label via the Properties panel, so that it says something a little more helpful.

While in the Properties section, add a click event handler to the form. This is an ActionScript function that is invoked whenever the button is clicked.

At the end of all this activity, here's what your form should look like:

Loading Up

Next, we need to write a little ActionScript to submit the form data to a server−side script. Select the first keyframe in the Timeline, pop open the Actions panel, and add the following code to it.

```
function doSubmit()
{
userData = new LoadVars();
userData.name = name;
userData.species = species;
userData.speciesType = speciesType.getValue();
userData.residence = residence.getValue();
userData.send("register.php", "", "post");
}
```

Here's what this all means:

The LoadVars() object is what actually packages the data for transmission to the server−side script...and so, the first task is to create an instance of this object.

```
userData = new LoadVars();
```

Once that's done, I've created object properties corresponding to the various form elements, and assigned values to them from the various input fields from the form.

```
userData.name = name;
userData.species = species;
userData.speciesType = speciesType.getValue();
userData.residence = residence.getValue();
```

Note my usage of the getValue() methods to acquire the value of the selected items in both the radio button set and the list box. ActionScript also offers the getSelectedItems() function, which can be used to retrieve the selected items in a multiple−select list box (not demonstrated here).

Once all the form data has been loaded into the object, the send() method is used to submit this data to a server−side script − in this case, "register.php".

```
userData.send("register.php", "", "POST");
```

Pretty simple − the variables from the form are sent to the script "register.php" via POST. Let's see what that script does.

```
<?

$name = $_POST['name'];
$species = $_POST['species'];
$speciesType = $_POST['speciesType'];
$residence = $_POST['residence'];

// open connection to database
$connection = mysql_connect("localhost", "user", "pass") or
die ("Unable
to connect!");
mysql_select_db("db20139a") or die ("Unable to select
database!");

// formulate and execute query
$query = "INSERT INTO aliens (alien_name, species_name,
species_type,
residence_type) VALUES ('$name', '$species', '$speciesType',
'$residence')"; $result = mysql_query($query) or die("Error in
query: "
. mysql_error());

// clean up
mysql_close($connection);

?>
```

This should be familiar to you if you've programmed in PHP before − the data entered by the user into the form is extracted from the special $_POST array and plugged into an SQL query template. A connection to the database server is initiated, this SQL query is then executed, and the user data is inserted into the database.

Coming Back For More

The example you just saw submitted the user's input to a PHP script, thereby redirecting the browser to a fresh page. In case you'd like this result page to also be generated in Flash, Flash MX includes a new ActionScript function named sendAndLoad(), which allows you to submit form data to a server−side script and read the response back into the Flash movie for further processing. This response can then be used to build a new Flash movie on the fly.

If this seems like alien technology to you, don't be afraid − it's not very difficult to do. First, you need to set up your server−side script so that it returns a set of variable−value pairs as URL−encoded form data − as the following example demonstrates:

```
alpha=1&beta=45
```

These variables can then be read into Flash and used to assign values to variables within the Flash movie.

Consider the following revision of the ActionScript code above, which demonstrates the process:

```
function doSubmit()
{
userData = new LoadVars();
userData.name = name;
userData.species = species;
userData.speciesType = speciesType.getValue();
userData.residence =
residence.getValue();

response = new LoadVars();
response.onLoad = getResponse();

userData.sendAndLoad("register.php", response, "post");

}

function getResponse(result)
{
if(result == true)
{
// use result values
}
else
{
// display error
}
}
```

In this case, a new instance of the LoadVars() object has been created, this one designed to hold the variable package returned by the server−side script. When this response is received by the Flash movie, the onLoad event handler is triggered, and the getResponse() function invoked.

Note the difference in the technique used to actually submit the form data to the server − the previous example used the send() method, which only submitted the data, while this one uses the sendAndLoad() method, which both submits data and accepts a response.

The getResponse() function returns an object, whose properties correspond to the variables returned by the server−side script. So, if the server−side script returned the string,

```
color=red&shape=round
```

which contains the variables "color" and "red", I could modify the getResponse() function above to use them like this:

```
function getResponse(result)
{
if(result == true)
{
set("itemInstance.color", result.color);
set("itemInstance.shape", result.shape);
}
else
{
// display error
}
}
```

In this case, variables within the Flash symbol "itemInstance" are being set to the values returned by the server−side script "register.php", via the set() method.

A Fine Balance

And that's about it for the moment. In this article, I demonstrated how Flash could be used to build simple Web forms to collect user data on your Web site or within your Flash application. I showed you how to build simple text input boxes, and link them to a form processing script that does something useful with the information submitted. I also gave you a crash course in the new form controls that ship with Flash MX, illustrating how they can be used to build ever more complicated forms, and how they can be combined with new ActionScript functions to both submit form data and process the response to dynamically build a results page.

On a concluding note, it's important to remember that building a form in Flash is a double−edged sword. On the one hand, you get access to all of Flash's cool animation functions, which make it possible to build graphically−rich, user−friendly forms rapidly and efficiently; on the other, the more complicated your forms are, the longer they'll take to download and render. Flash offers creative developers an immense amount of power − it's wise to remember that it should be used carefully.

Till next time...be good!

Note: All examples in this article have been tested on Macromedia Flash 5.0. Examples are illustrative only, and are not meant for a production environment. Melonfire provides no warranties or support for the source code described in this article. YMMV!