Getting started

Download and install Flex Builder 2 if you haven't done so already. To start, you must create the database. I've called mine *sample*, but you can call yours whatever you like. Next, create a table that will hold the user data. Here is the SQL script you can use to create the table:

```
CREATE TABLE 'users' (
  'userid' int(10) unsigned NOT NULL auto_increment,
  'username' varchar(255) collate latin1_general_ci NOT NULL,
  'emailaddress' varchar(255) collate latin1_general_ci NOT NULL,
  PRIMARY KEY  ('userid')
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci
AUTO_INCREMENT=3 ;
```

Next, you'll create the PHP script that adds users and exports the XML that the Flex application will consume. The script is relatively simple and consists of 25 lines of code. Note the use of the `quote_smart` function as a best practice to help verify user input, according to the PHP.Net website.

```
<?php
Define( "DATABASE_SERVER", "localhost" );
Define( "DATABASE_USERNAME, "username" );
Define( "DATABASE_PASSWORD", "password" );
Define( "DATABASE_NAME", "sample" );

//connect to the database
$mysql = mysql_connect(DATABASE_SERVER, DATABASER_USERNAME, DATABASE_PASSWORD);

mysql_select_db( DATABASE_NAME );

// Quote variable to make safe
function quote_smart($value)
{
   // Stripslashes
   if (get_magic_quotes_gpc()) {
       $value = stripslashes($value);
   }
   // Quote if not integer
   if (!is_numeric($value)) {
       $value = "'" . mysql_real_escape_string($value) . "'";
   }
   return $value;
}

if( $_POST["emailaddress"] AND $_POST["username"])
{
      //add the user
      $Query = sprintf("INSERT INTO users VALUES ('', %s, %s)",
quote_smart($_POST["username"], quote_smart($_POST["emailaddress"])";

      $Result = mysql_query( $Query );
}

//return a list of all the users
$Query = "SELECT * from users";
$Result = mysql_query( $Query );

$Return = "<users>";

while ( $User = mysql_fetch_object( $Result ) )
{
      $Return .= "<user><userid>".$User->userid."</userid><username>".$User-
```

```
>username."</username><emailaddress>".$User-
>emailaddress."</emailaddress></user>";
}
$Return .= "</users>";
mysql_free_result( $Result );
print ($Return)
?>
```

Here is a quick explanation of the PHP code. The `$_POST` variable is populated with values from the Flex application with two required fields: `emailaddress` and `username`. If a user enters information for both of those, the PHP code adds the user to the database. After that, the PHP code returns a list of users in XML format.

Note: You cannot pass PHP variables to Flex applications directly. You must encode them in XML first. By abstracting the user interface from the data retrieval, you can easily change how you display data. For example, you could use this same PHP script to pass data to a mobile phone version of the same application. All you would need for that is to write the front end of the application, the back-end PHP script would remain the same.

Up until now, everything should be familiar to you. You have a PHP script and a MySQL database. Now it's time to start building the interface to the application.

Building the user interface

Flex applications use a combination of ActionScript 3.0 and MXML. ActionScript is based on ECMAScript (similar to JavaScript), so it should be familiar to web developers. MXML is an XML-based layout engine for Flex applications. Essentially, you lay out the user interface using XML, and script the user interface using ActionScript. The MXML for the interface is, again, very simple (only 26 lines):

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*"
layout="absolute" creationComplete="userRequest.send()">
   <mx:HTTPService id="userRequest" url="http://localhost/flex/php/request.php"
useProxy="false" method="POST">
      <mx:request xmlns="">
         <username>{username.text}</username><emailaddress>{emailaddress.text}</
emailaddress>
      </mx:request>
   </mx:HTTPService>
   <mx:Form x="22" y="10" width="493">
      <mx:HBox>
         <mx:Label text="Username"/>
         <mx:TextInput id="username"/>
      </mx:HBox>
      <mx:HBox>
         <mx:Label text="Email Address"/>
         <mx:TextInput id="emailaddress"/>
      </mx:HBox>
      <mx:Button label="Submit" click="userRequest.send()"/>
   </mx:Form>
   <mx:DataGrid id="dgUserRequest" x="22" y="128"
dataProvider="{userRequest.lastResult.users.user}">
      <mx:columns>
         <mx:DataGridColumn headerText="User ID" dataField="userid"/>
         <mx:DataGridColumn headerText="User Name" dataField="username"/>
      </mx:columns>
   </mx:DataGrid>
   <mx:TextInput x="22" y="292" id="selectedemailaddress"
text="{dgUserRequest.selectedItem.emailaddress}"/>
</mx:Application>
```

Let's examine each line in detail. These are the first two lines of each Flex application:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*"
layout="absolute" creationComplete="userRequest.send()">
```

The first line declares that this is an XML document. The second line declares that this is an Application, provides the namespace for MX components, declares the layout to be absolute (which means that you can position items to the exact x and y coordinates. Other options are horizontal layouts or vertical layouts), and finally `creationComplete="userRequest.send()"` specifies that on completion of loading the user interface, the application calls the function `send()` on the MXML element with the id `userRequest`.

The following section sets up HTTPService to send and receive data from the PHP script you created:

```
<mx:HTTPService id="userRequest" url="http://localhost/flex/php/request.php"
useProxy="false" method="POST">
      <mx:request xmlns="">
         <username>{username.text}</username><emailaddress>{emailaddress.text}</
emailaddress>
      </mx:request>
   </mx:HTTPService>
```

In this section, you set the id to `userRequest`, and provide a URL to the PHP script. You set the method of submit to `POST` (you could also use `GET`, but then you must change the variables in the PHP script). The request itself contains two variables, `username` and `emailaddress`. This code also sets the value for `username` to the `text` attribute of the element with id `username` (`username.text`) and the value for the PHP variable `_POST["emailaddress"]` is set to the text attribute of the element with id `emailaddress` (`emailaddress.text`). The `{` and `}` brackets bind the variables to the value of the user interface elements.

To be clear, if you changed `<username>` to `<user_name>`, you would have to change the PHP variable to `_POST["user_name"]`. If you change `{username.text}` to `{user_name.text}`, you would have to modify your MXML: you would have to change the element with the ID of `username user_name`.

Creating a form

Next, you will build a simple form:

```
<mx:Form x="22" y="10" width="493">
<mx:HBox>
      <mx:Label text="Username"/>
      <mx:TextInput id="username"/>
   </mx:HBox>
   <mx:HBox>
      <mx:Label text="Email Address"/>
      <mx:TextInput id="emailaddress"/>
   </mx:HBox>
   <mx:Button label="Submit" click="userRequest.send()"/>
</mx:Form>
```

Note that you can lay out the exact x and y coordinates of the form and set its exact width. Then, two HBoxes surround a label and text input, allowing them to flow from left to right, one above the other. Finally, your Submit button appears at the end of the form. When a user clicks the button, the application calls the `send()` function of the element with ID `userRequest` (in this case, it is the HTTPService element).

Now that you have created the functionality that submits new entries to the database, how do you display them? Use the following code:

```
<mx:DataGrid id="dgUserRequest" x="22" y="128"
dataProvider="{userRequest.result.users.user}">
      <mx:columns>
         <mx:DataGridColumn headerText="User ID" dataField="userid"/>
         <mx:DataGridColumn headerText="User Name" dataField="username"/>
      </mx:columns>
   </mx:DataGrid>
   <mx:TextInput x="22" y="292" id="selectedemailaddress"
text="{dgUserRequest.selectedItem.emailaddress}"/>
</mx:Application>
```

In this case, you have a DataGrid component that populates itself with the XML from the `userRequest` HTTPService. You return an XML document. In this case, you bind the DataGrid component to the user elements in the XML document that is returned. The returning XML looks something like the following:

```
<users>
<user>
<userid>1</userid>
<username>Joe Schmoe</username>
<emailaddress>joe@schmoe.com</emailaddress>
</user>
<user>
<userid>2</userid>
<username>Betty Schmoe</username>
<emailaddress>betty@schmoe.com</emailaddress>
</user>
</users>
```

Note that you bind to the actual elements that are returned, not to the wrapper element around them.

The DataGrid component displays the user id and user names of people in the database. I decided not to show the e-mail address in the datagrid, but you could add another column with that information in it. Note that the columnName element needs to map directly to the XML elements. The DataGrid element will take care of allowing your users to sort and highlight the rows as they are selected—you don't need to do anything for that!

Finally, you have a TextInput element that shows the e-mail address of the selected user, `dgUserRequest.selectedItem.emailaddress`, and then an XML tag that closes the application.

That's it. You have a simple Flash application that submits and retrieves data from a MySQL database, using PHP as a back end.

Where to go from here

I urge you to download Flex Builder 2 and build more complicated applications using PHP, MySQL, and Adobe Flex. Check out my blog for more information on Adobe Flex 2 and please provide suggestions for future articles and what other samples you'd like to see using this set of technologies. You may also want to read my article, Using Flex 2 and AMFPHP.