Flash-PHP Forms

PHP (PHP Hypertext Preprocessor, not "Perl-Hating Programmers") is a server-side scripting language. It is typically run as a module to the Apache web server software. Our server uses the Linux operating system and we have the MySQL relational database system available. This combination of Linux, Apache, MySQL, and PHP is called "LAMP."

HTML forms have the virtue of being easily created and connected to server-side technologies like PHP. However, they are limited to the display characterictics offered by the browser and ways these features can be manipulated with Cascading Style Sheets.

Flash is traditionally thought of as a system for creating web animations. However, it is increasingly used to develop custom user interfaces. With Flash, if you can dream it, you can probably create it.

The purpose of this project is to develop a simple Flash form which can send data to PHP and receive a reply from PHP and display it in the Flash interface. Naturally, the interface can be made more attractive and the PHP actions more sophisticated, including database requests, etc., but this will serve as a proof of concept.

HTML Form

Below is a sample form in HTML to collect information from a web visitor along the lines of a simple feedback form. The data will be sent to PHP.

```
<html>
<body>
<form method="post" action="script.php">
Name:    <input type="text" name="visitor"><br>
E-Mail:  <input type="text" name="email"><br>
Comment: <br>
  <textarea name="message" wrap="virtual" rows=5 cols=80>
  <br>
<input type="submit" value="Go">
</form>
</body>
</html>
```

Of note here are the parameters in the <form> tag. The method property describes how the data collected in the form will be transmitted to the PHP script.

The two choices are POST and GET. The GET method collects the variables and values and sends them in the URL which calls the PHP script. There are certain times when this technique is very convenient. Variables and values can be sent to a program when the user clicks a link created with an <a> tag.

About 95% of the time POST is the better choice. The GET method is limited by the maximum length of a URL. Microsoft Internet Explorer, for example, limits a URL to 2048 characters. This would be a problem for this form where a visitor might give a comment in excess of 400 words.

The POST method sends the variables and values as part of the HTTP body rather than in the URL. Because of this, the length is not limited in the way it is with the GET method.

The action parameter of the <form> tag defines the location of the script which processes the data collected in the HTML form. Filenames on Unix and Linux servers are case sensitive so this value must be consistent with the spelling and capitalization used for the file on the server.

This form has three kinds of input elements, although there are many others. The first two examples are <input type="text" ...>. This is the basic default for an HTML form and allows a user to type in a line of alphanumeric characters. The name property defines the variable name which is associated with this user input.

The next type is the <textarea> input. As with the text input example, the name property defines the variable for the data. The rows and cols properites set the size of a scrolling text entry field which can contain many lines of input. The wrap property defines how the <textarea> field should behave. In this case, the value "virtual" will wrap the text but not change the actual value transmitted by inserting carriage return (\r) or newline (\n) characters.

The last input type in this sample form is <input type="submit" ...> which actually sends the collected data to the PHP script.

PHP Script

The next step is to create a PHP script to receive this data and do something with it. Our form asks for the visitor's name, e-mail, and a comment. Since PHP runs on the server, we can access the resources available on the server, including the ability to send e-mail messages.

Most installations of PHP have a control feature called `register_globals` in the `off` position; this provides more security than the `on` position. PHP receives POST form data and places it in an associative array called `$_POST`. It should be noted that variable names in PHP are case sensitive so `$XYZ` is different than `$xyz`.

An associative array is similar to what would be called a hash in the Perl language. It has keys which are words instead of numbers for traditional arrays. However, in PHP all arrays are associative arrays, even when the index keys are numbers.

PHP programs are typically embedded inside HTML documents like the sample code below:

```
<html>
<body>

<!-- HTML Comment: start PHP section -->
<?php
  # PHP comment: collect data from POST input
  $name  = $_POST['visitor'];
  $email = $_POST['email'];
  $msg   = $_POST['message'];

  # display data
  print "<b>From:   </b>$name<br>\n";
  print "<b>E-Mail: </b>$email<br>\n";
  print "<fieldset>\n";
  print nl2br($msg);
  print "</fieldset>\n";
?>

</body>
</html>
```

Notice that comments in the HTML section of the code are bound by <!-- ... --> but within the PHP section comments are lines which begin with the pound sign (#) or which are bound by /* ... */.

The three statements which use the $_POST variables assign values from the form to simpler variable names. They can be displayed with print or echo statements.

The
 tag is used to affect the way content is displayed in the browser by continuing the content on the next line. The newline (\n) character in a print statement will affect the HTML source code and continue it on the next line; it does not affect the display in the browser. In some of the print statements in the sample both
 and \n are used.

The nl2br() function converts newline (\n) characters from the <textarea> input field into
 tags so the content can be displayed properly in an HTML page.