PHP: Processing XHTML Forms

Introduction

A Web form is a common method for allowing the user to provide input. As described in *XHTML: Forms*, there are two parts to a form: the user interface, and a script to process the input and ultimately do something meaningful with it. This document explains how PHP can be used to process form-based input. Consult the *XHTML: Forms* document for details about form syntax, submission method types (i.e., `GET` and `POST`), and the types of form widgets.

Form setup

The first step in handling user input via Web forms is to create the user interface. The form is typically defined in an XHTML document (e.g., a file that ends with `.html`) unless the form itself has dynamic content. The `action` attribute of the `form` element should be the URL of the PHP script that will be processing the form data.

Consider the following XHTML markup that defines a form with selection widgets, text input widgets, and a hidden field. Note that the `action` attribute of the `form` element designates the PHP script `processform.php` as the recipient of the form data.

```
<form method="post" action="processform.php">
  <!-- Radio buttons, none pre-selected -->
  <p>How would you rate your skill in programming?<br />
    <input type="radio" name="skill" value="beg" />Beginner
    <input type="radio" name="skill" value="int" />Intermediate
    <input type="radio" name="skill" value="adv" />Advanced
    <input type="radio" name="skill" value="sup" />Super-hacker</p>

  <!-- Radio buttons, one pre-selected -->
  <p>How many hours do you spend programming each week?<br />
    <input type="radio" name="hours" value="0-10" />0-10<br />
    <input type="radio" name="hours" value="11-20" checked="checked" />11-20<br />
    <input type="radio" name="hours" value="21-30" />21-30<br />
    <input type="radio" name="hours" value="30+" />30+</p>

<!-- Checkboxes, several pre-selected -->
  <p>I agree to...<br />
    <input type="checkbox" name="cheaplabor" value="yes" checked="checked" />work for $1.50/hour.<br />
    <input type="checkbox" name="longdays" value="yes" checked="checked" />work 12 hours per day.<br />
    <input type="checkbox" name="late" value="yes" />show up late every day.<br />
    <input type="checkbox" name="usecomments" value="yes" checked="checked" />comment my code.
    <br /></p>

  <!-- Menu, one selected, multiple selections allowed -->
  <select name="state[]" size="5" multiple="multiple">
    <option value="al">Alabama</option>
    <option value="ak">Alaska</option>
    <option value="as">American Samoa</option>
    <option value="az">Arizona</option>
    <option value="ar">Arkansas</option>
    <option value="ca" selected="selected">California</option>
    <option value="other">Some other state</option>
  </select>

  <!-- Text box and password box -->
  <p>Username: <input type="text" name="username" /></p>
  <p>Password: <input type="password" name="passwd" /></p>
```

```
  <!-- Text area -->
  <textarea name="comments" rows="5" cols="40"></textarea>

  <!-- Hidden field -->
  <input type="hidden" name="promotion_code" value="x3g9kf43" />

  <!-- Submit button -->
  <p><input type="submit" value="Submit the Data" /></p>
</form>
```

Now that the user interface has been established, the PHP script can be created to process the data.

Receiving form data

Recall that form data is submitted in name-value pairs, which are derived from the form widgets' `name` and `value` attributes. The standard method for accessing this data is by accessing one of the predefined associative arrays named `$_POST` and `$_GET`, depending on the form submission method used. The syntax is `$_POST['name']`, where *name* corresponds to the name attribute of a given form widget. Here is a complete PHP script that lists the name-value pairs in a table.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>Programming is Fun!</title>
  </head>
  <body>

    <h1>Form Results</h1>
    <table border="1">
      <tr><th>Field</th><th>Value</th></tr>
    <?php>
      print "<tr><td>Skill</td><td>{$_POST['skill']}</td></tr>\n";
      print "<tr><td>Hours spent programming</td><td>{$_POST['hours']}</td></tr>\n";
      print "<tr><td>Programming languages used</td><td>{$_POST['proglang']}</td></tr>\n";
      print "<tr><td>Work for $1.50/hr?</td><td>{$_POST['cheaplabor']}</td></tr>\n";
      print "<tr><td>Work 12-hour days?</td><td>{$_POST['longdays']}</td></tr>\n";
      print "<tr><td>Show up late?</td><td>{$_POST['late']}</td></tr>\n";
      print "<tr><td>Comment your code?</td><td>{$_POST['usecomments']}</td></tr>\n";
      print "<tr><td>State of residence</td><td>{$_POST['state']}</td></tr>\n";
      print "<tr><td>Username</td><td>{$_POST['username']}</td></tr>\n";
      print "<tr><td>Password</td><td>{$_POST['passwd']}</td></tr>\n";
      print "<tr><td>Comments</td><td>{$_POST['comments']}</td></tr>\n";
      print "<tr><td>Promotion code</td><td>{$_POST['promotion_code']}</td></tr>\n";
    ?>
    </table>
  </body>
</html>
```

If a value for a form widget was not specified – for example, if no text was supplied for the text box named `username` – the value for that entry in the `$_GET` or `$_POST` associative array will be the empty string.

Menus with multiple selections

When using a menu widget that allows multiple selections, an empty pair of brackets must be added to the `name` attribute of the `select` element. Recall the menu definition from the previous XHTML markup:

```
<select name="state[]" size="5" multiple="multiple">
  ...
</select>
```

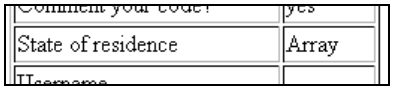
If the brackets are omitted, only the bottom-most selected menu choice will be available via the `$_GET` or `$_POST` associative arrays. For example, suppose that three menu options were selected: Alabama, Alaska, and Arkansas.

Without the brackets, the value of `$_POST['state']` is the string `"Arkansas"`. However, if the `name` attribute's value for the `select` element is changed to `state[]`, the value stored in `$_POST['state']` is an array with three string elements. There are several things to note here:

- When multiple menu options are selected, the form data sent to the server actually contains multiple name-value pairs. For example, the URL encoding for a GET request may look something like the following:

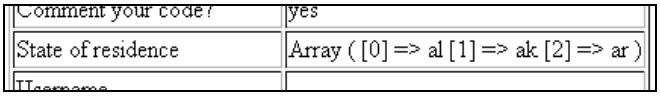
```
...&state=alabama&state=alaska&state=arkansas...
```

- The value of `$_POST['state']` will be an array even if only one item in the menu is selected. In this case `$_POST['state']` will be an array with one element.
- Arrays will be covered in a later document. However, there are two concepts worth mentioning at this point.
    1. If you call `print()` with an array, the output will be `Array`. Here is how the output may appear in a browser:



    2. To display the contents of an array, use `print_r()` – meaning "print recursively". The output format is rather crude, but it allows you to see the contents of the array. The modified line of PHP code and example browser output follows:

```
print "<tr><td>State of residence</td><td>";
print_r($_POST['state']);
print "</td></tr>\n";
```



- The requirement of modifying the `name` attribute seems to be unique to PHP. For example, Perl's CGI interface automatically converts the multiple selections into an array.