Lab 1: Processing Web Forms

Due: at the end of the lab period in which it is issued

Objectives

Upon successful completion of this lab, you will have demonstrated the ability to:

- Use PHP Apache module to generate a web page with a form;
- Use PHP to process data in the web form when it is submitted

Reference

- See lecture notes.
- You may want to look at this sample program.

Required Materials

- zenit account

Lab Preparation

- none

Part I - Form submission using GET & POST

When a web page request is submitted, information about the *request* is passed through *environment variables*. These variables are often used to identify characteristics of the requesting application. The values of these variables are also often used to create logs of web requests (for example, if I am required to log the IPs of all submissions of a form, I can capture the information and store it in a DB). You can also detect client browser types, and this is often used to generate web pages differently because different browsers and different versions of the same browser work in slightly different ways. For example, IE supports some features FF does not, and vice versa. By detecting the browser type, we can modify the web page to display the same page properly in both.

In addition, you can also pass user variables to the web server through forms by using POST or GET. GET sends the variable names and their associated values as part of the URL (you can see it in the location bar), and POST sends them invisibly (but in text) as part of the request. POSTed values are considered slightly more secure since they are invisible to the casual user, but you can use Wireshark or a similar program to see these values. GET values also have a limited string length since they are sent as part of the URL and the browsers all have limits to the length of a URL they will process (max length varies from browser to browser). On the other hand, POSTed values can be much longer since they aren't sent via the URL.

Here's how you can examine any POST or GET variable in PHP:

```
$fieldNameValue = $_POST['fieldName']; $fieldNameValue =
$_GET['fieldName']; $fieldNameValue = $_REQUEST['fieldName']
```

`$_POST`,`$_GET` and `$_REQUEST` are three built-in arrays that use the name specified in the HTML form (`name="firstName"`) as the key while the value in the array is whatever was entered in the form input field. `$_REQUEST` works for both the GET and POST methods. In all three cases, keys are created based on the names given to input fields on the form, and any values in the input elements (text in a text field, the selected element from a drop down, etc) will be stored as that key's value. Note that you should avoid using $_REQUEST - use $_POST and $_GET since these will be empty if there is no GET or POST, while $_REQUEST may not be empty because it includes other things (such as cookies). Also, using $_POST and $_GET is less confusing to someone trying to follow your program logic because they know the method you've specified.

1. Here is a link to a web form: https://open.senecacollege.ca/cms/file.php/601/fsoss-register.html . Copy this URL and paste it on a new browser tab to access the web form. Write a PHP program called **fsoss-register.php** that displays this web page the first time the user comes to the page (either by clicking a link or typing in the URL). Hint: you do not need to code the HTML - view page source and copy and paste it into your program to start.
2. Modify the program so that it submits the form as a GET to another PHP program named **displayforminfo.php** which then displays all the information in the form. Label the information appropriately (for example, put "First name:" followed by the string entered in the first name field).
3. Redo the program using POST.

Part II - Self-Referential Web programming

The ACTION attribute in an HTML tag specifies what to do when the submit button is pressed. In the previous sections you called a second PHP script to display the values in the form fields. However, most forms actually call back to the same program when they are submitted. This is called **self-referential programming** (or **stateless programming**). A self-referential program has an ACTION that refers back to itself. When you run this kind of program, it needs to know if a script is being called for the first time (in which case it displays a blank form) or through the press of a Submit button (in which case the data entered in the form fields needs to be processed). To determine if a Submit button (named 'submit') was pressed, you can use the following logic:

```
# checks to see if the submit button was pressedif (
isset($_POST['submit']) ) { process form data (usually validates
data first before processing)} display form
```

`isset()` tests to see if there is a value set for this key. If there is no value, then this means the submit button wasn't pressed. Alternatively, you could have tested for the existence of the key "submit" using

```
if ( key_exists('submit', $_POST) )
```

or the existence of the $_POST superglobal array (which does not exist if it is not a POST)

```
if ( exists($_POST) )
```

An even shorter form would be

```
if ($_POST)
```

Revise the **fsoss-register.php** script so that it displays the form if it is **not** a submit and, **if the form is submitted, prints out ALL the information entered in the form**. The program should **not** call another program, only itself (i.e. merge the two programs from Part I).

Part III - Re-populating Form Fields

Modify **fsoss-register.php** so that when the submit button is pressed the information entered in the text fields is re-displayed in the text fields (i.e. repopulate the form fields). You do not have to repopulate the radio buttons or drop down menu.

Part IV - Simple Validation

Modify your script so that it prints an error message next to any text field that does not contain at least one character. You do not have to validate the radio buttons, checkboxes or drop down.

Final touches and submission

Demonstrate your **fsoss-register.php** script to me before the end of class.