CSC2233
Internet Programming

Lecture 6:

Forms

Nishanthi Dasanayaka
nishanthid@dcs.ruh.ac.lk

Outline

- HTML Forms
- Passing information between pages
  - GET Argument
  - POST Argument
- Validating the form data

HTML Forms

```
<html>
<head>
<title>A BASIC HTML FORM</title>
</head>
<body>
<FORM NAME ="form1" METHOD =" "  ACTION = "">
<INPUT TYPE = "TEXT“ NAME = “Name” VALUE ="username">
<INPUT TYPE = "Submit" NAME = "Submit1" VALUE = "Login“>
</FORM>
</body>
</html>
```

- Method attribute is used to tell the browser how the form information should be sent
- The two most popular methods
  - GET
  - POST

Ex:-

- <FORM NAME ="form1" METHOD ="GET" ACTION = ““>
- <FORM NAME ="form1" METHOD =“POST" ACTION = "">

GET Method

- Information sent from a form with the GET method is visible to everyone.
- Use the GET method when the data you want to returned is not crucial information that needs protecting.
- It has limits on the amount of information to send through GET method.
- The predefined $_GET variable is used to collect values in a form with method="get"

GET Method

```
<form action="welcome.php" method="get">
    Name: <input type="text" name="fname" />
    Age: <input type="text" name="age" />
    <input type="submit" />
</form>
```

Form.html

```
Welcome <?php
        echo $_GET["fname"];
    ?>.<br /> You are
    <?php
        echo $_GET["age"];
    ?> years old!
```

welcome.php

POST Method

- When Information sent from a form with the POST method
  - invisible to others
  - no limits on the amount of information being sent.

- The predefined $_POST variable is used to collect values from a form sent with method="post".

Using both GET and POST in the same page

```
<?php
if(isset($_POST['submit']) ){
        if($_GET['lang'] == "english"){
                echo("First name: " . $_POST['firstname'] . "<br />\n");
                echo("Last name: " . $_POST['lastname'] . "<br />\n");
        }
elseif($_GET['lang'] == "spanish") {
        echo("Nombre: " . $_POST['firstname'] . "<br />\n");
        echo("Apellido: " . $_POST['lastname'] . "<br />\n"); }
else{}    }
?>
```

Using both GET and POST in the same page

```
<form method="post" action="form.php?lang = spanish ">
<p>First name: <input type="text" name="firstname" /></p>
<p>Last name: <input type="text" name="lastname" /></p>
<input type="submit" name="submit" value="Submit" />
</form>
```

GET vs. POST

- Both GET and POST create an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)).

- This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

- Both GET and POST are treated as $_GET and $_POST. These are super globals, which means that they are always accessible, regardless of scope.

GET vs. POST

- $_GET is an array of variables passed to the current script via the URL parameters.

- $_POST is an array of variables passed to the current script via the HTTP POST method

$_REQUEST

- $_REQUEST is a super global variable which is widely used to collect data after submitting html forms
- Instead of using GET and POST arrays, you can also use the $_REQUEST array
- If GET and POST variables have the same name, POST will take priority

```
<?php
        $name=$_REQUEST['name'];
        echo $name;
?>
```

Action Property

- action property indicate
  - Where to send form data

EX:-

  - <form method = "post" action = "form.php">
  - The action attribute of the form element indicates that when the user clicks submit button, the form data will be posted to form.php

Using PHP_SELF in the action field of a form

- PHP_SELF is a variable that returns the current script being executed
- returns the name and path of the current file (from the root folder)
  - echo $_SERVER['PHP_SELF'];
- Using PHP_SELF variable you can write more generic code which can be used on any page

```
<form name="form1" method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>" >
```

Getting values from a Text Box with PHP

- To get at the text that a user entered into a text box, the text box needs a NAME attribute.

Ex:-

- <INPUT TYPE = "Text" VALUE ="username" NAME = "username">
- To return data from a HTML form element, you use the following strange syntax:

```
$username = $_POST['username'];
$username = $_GET['username'];
```

PHP and Radio Buttons

```
<FORM name ="form1" method ="post" action
="radioButton.php">
<Input type = 'Radio' Name ='gender' value= 'male'>Male
<Input type = 'Radio' Name ='gender' value= 'female'>Female
<Input type = "Submit" Name = "Submit1" VALUE = "Select a
Radio Button">
</FORM>
```

PHP and Radio Buttons

```
<?PHP
$male_status = 'unchecked'; $female_status = 'unchecked';
if (isset($_POST['Submit1'])) {
$selected_radio = $_POST['gender'];
        if ($selected_radio == 'male') {
                $male_status = 'checked';
                echo "male Selected"; }
        else if ($selected_radio == 'female') {
        $female_status = 'checked'; echo "female Selected";
} } ?>
```

PHP and Dropdown List

```
<form action=“dropdown.php" method="post">
<select name="manufacturer">
<option value="Lincoln">Lincoln</option>
<option value="Example">Example</option>
</select>
<input type="submit" name="btnsubmit" />
</form>
```

PHP and Dropdown List

```
<?php
if(isset($_POST['btnsubmit']))
echo "Manufacturer: ".$_POST['manufacturer'];
?>
```

Form Processing

- Confirm that valid information was entered
- extract function
  - Creates variables corresponding to each key-value pair in array
  - Easily retrieve all values sent to PHP page
- Regular expressions very helpful
- Ending a script
  - die function
  - Remember to close all HTML tags

```
<?php
        extract( $_POST );
if( isset($_POST['submit']) ) { }
// determine whether phone number is valid and print
// an error message if not
if ( !preg_match( "/^(\(\+[0-9]{2}\)[0-9]{9})$/",$phone) ){// Perform a regular
expression match
        print( "<p><span style = \"color: red; font-size: 2em\">
        INVALID PHONE NUMBER</span><br />
        A valid phone number must be in the form
        <strong>(+94)412278456</strong><br />
        </span></p></body></html>" );
die(); // terminate script execution }
else{ echo "Valid phone no"; }
?>
```

Form Validation

- User input should be validated on the browser by client scripts
  - JavaScript
- Server validation use if the user input will be inserted into a database
- Use **check_input** function and simply call this function whenever we need to validate simple input data

```
<?php
$yourname = check_input($_POST['yourname']);
$email = check_input($_POST['email']);
$likeit = check_input($_POST['likeit']);
$comments = check_input($_POST['comments']);
?>
<html> <body>
Your name is: <?php echo $yourname; ?><br />
Your e-mail: <?php echo $email; ?><br /> <br /> Do you like this website?
<?php echo $likeit; ?><br /> <br /> Comments:<br />
<?php echo $comments; ?>
</body> </html>
 <?php function check_input($data) { $data = trim($data); $data = stripslashes($data);
return $data; } ?>
```

Validating the input

- **empty()**
  - Determine whether a variable is empty

    if(empty($_POST["name"]))

- **Isset()**
  - Determine if a variable is set and is not NULL

    if(isset($_POST['submit']))

Validating the input

- Check if the name field only contains letters and whitespace. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
        $nameErr = "Only letters and white space allowed";
}
```

The preg_match() function searches a string for pattern, returning true if the pattern exists, and false otherwise.

Validating the input

- Validate email format.
- Use PHP's filter_var() function to check whether an email address is well-formed.

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $emailErr = "Invalid email format";
}
```

Form validation example

- Validate email format.
- Use PHP's filter_var() function to check whether an email address is well-formed.

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $emailErr = "Invalid email format";
}
```

Summary

- One of the most powerful features of PHP is the way it handles HTML forms.

- Learn on how to work with HTML Forms using HTML attributes METHOD, ACTION and SUBMIT