Web Technologies (ICDT6004) – Lecture 7

Introduction to PHP

Aims of lecture

- At the end of today's lecture students should be able to
  - Install a web server
  - Write server-side script

What is PHP?

- It is a server-side scripting language
  - requires a web server
  - Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP- specific features thrown in.
  - Supports many different databases
  - Open source software - free to download and use

What needs to be installed?

- A web server - Apache in this case
- A WAMP (Windows Apache MySQL PHP) software bundle
  - Easy PHP Devserver 17.0 (Latest version)
- WAMP contains the web server - Apache
- WAMP contains a database – MySQL
- WAMP has PHP support
- Other examples include WAMPServer and XAMPP

Accessing the web server

- The web server can be accessed through localhost at either:
  - http://localhost/
  - or http://127.0.0.1
- By default, apache uses port 80
- If another web server, e.g. IIS, is also installed, then both cannot be used simultaneously on the same port number
  - either one of the web servers must be stopped
  - or if the 2 servers are to run simultaneously, they should use different port numbers. E.g. Apache **can** use port 8086 or 1111 Accessed by specifying port number - http://localhost:8086/ or http://127.0.0.1:8086/

Writing a php file

- A PHP file is saved as .php
- It can be written in any simple editor like notepad, notepad++, sublimeText or a web development application like Adobe Dreamweaver CC can be used
- It can contain text, HTML tags, JavaScript code, CSS code and PHP scripts
- However PHP code is never sent to browser, only the resulting HTML code is sent to the browser

PHP Delimiters

- PHP codes are enclosed between delimiters
- <? Some PHP code ?> - server must have shorthand support enabled
- <?php Some PHP code ?> - recommended
- A PHP scripting block can be placed anywhere in the document.
- Each line of code in PHP must end with a semicolon.
- Comments can be either:
  - //this is a comment
  - /* this is a comment block - can span
    more than one line
    */

Saving php files

- All files must be placed either in the directory “eds-www" or an alias that has been created, so that PHP can interpret the PHP pages.
- To view the PHP pages, open browser to http://127.0.0.1 (with port number if needed) or an Alias on the "Administration" page of EasyPHP.

Example 1

```
<!DOCTYPE html>
<html>
<head>
<title>My First PHP Page</title>
</head>
<body>
<?php
echo "Hello World!";
?>
</body>
</html>
```

Example 2

```
<!DOCTYPE html>
<html>
<head>
<title>My Second PHP Page</title>
</head> <body>
<?php
echo "Hello World!";
        echo "Hello World!";
?>
</body>
</html>
```

Line breaks and white spaces

- As with HTML, line breaks and white spaces are ignored.
- What is the output produced by the code in Example 2?

Variables in PHP

- All variables in PHP start with a $ sign symbol.
- $var_name = value;
- There is no need to define the name and type of the variable before using it.
- PHP automatically converts the variable to the correct data type, depending on how it is set.
- $name="John";
- $code=2003;

Variables in PHP(2)

- The $ sign is required for a variable, otherwise it does not work
- There are rules to naming a variable in PHP.
- From the php.net website:
  - A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
- A variable name cannot contain spaces.

Case-sensitivity

- In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive.
- However variable names are case-sensitive: **$a_number** and **$A_number** are different

Example 3

```
<?php
$var = "Bob";
$Var = "Joe";
echo $var ;//outputs Bob
echo $Var;//outputs Joe
$4site = "not yet";// invalid; starts with a number
$_4site = "not yet";// valid; starts with an underscore
?>
```

Example 4

```
<?php
$txt1="Hello World.";
$txt2="We are in 2008.";
echo $txt1. " ". $txt2 ;
/* note the use of '.' for concatenation */
?>
```

Qu: Investigate the strlen() and strpos() functions.

Arrays

- An array stores multiple values in one single variable and array names start with $.
- Arrays are divided into elements that behave as individual variables.
- <?php
  $cars = array("Volvo", "BMW", "Toyota");
  echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
  ?>

Conditional Statements

In PHP we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

Example 5 - if

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
?>
```

Example 6 – if...else

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

Example 7 - if...elseif....else

```
<?php
$t = date("H");

if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

Switch…case

switch (*n*)
{
case *label1:*
code to be executed if n=label1;
break;
case *label2:*
code to be executed if n=label2;
break;
default:
code to be executed if n is different from both label1 and label2;
}

Example 8 – Switch…case

```
<html><body>
<?php
$classsize = 2;
switch ($classsize) {
case -1:
echo "Poor";
break;
case 1:
echo "adequate";
break;
case 2:
echo "Good";
break;
default:
echo "None selected";
}
?> </body> </html>
```

PHP Loops

- Often, you want the same block of code to run over and over again in a row.
- Loops can be used instead of repeating the same code.
- In PHP, we have the following looping statements:
  - **while** - loops through a block of code while a specified condition is true
  - **do...while** - loops through a block of code once, and then repeats the loop as long as a specified condition is true
  - **for** - loops through a block of code a specified number of times
  - **foreach** - loops through a block of code for each element in an array

While loop

- Loops execute a block of code while a specified condition is true.
- while (*condition is true*) {
  code to be executed;
  }
- <?php
  $x = 1;

  while($x <= 5) {
  echo "The number is: $x
";
  $x++;
  }
  ?>

For loop

- Loops execute a block of code a specified number of times.

  for (*init; condition; increment*) {

  code to be executed;

  }

- ```
  <?php
  for ($x = 0; $x <= 10; $x++) {
      echo "The number is: $x <br>";
  }
  ?>
  ```
  ```

Foreach loop

- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
- foreach ($*array* as $*value*) {
  code to be executed;
  }
- <?php
  $colors = array("red", "green", "blue", "yellow");

  foreach ($colors as $value) {
  echo "$value
";
  }
  ?>

Functions

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.
- PHP has more than 1000 built-in functions.
- We can also create our own functions.

Example 9 - function

```
<!DOCTYPE html>
<html><body>
<?php
//creating the function printHelloWorld
function printHelloWorld(){
echo "Hello World";
}
//calling the function printHelloWorld
printHelloWorld();
?>
</body></html>
```

Example 10 – function with argument

```
<!DOCTYPE html>
<html>
<body>
<?php
function sayHi($fname) {
echo $fname . "<br />";
}
echo "Hi";
sayHi("John");
echo "Hi";
sayHi("Jane");
</body>
</html>
```

Example 11 – function returning a value

```
<!DOCTYPE html>
<html> <body>
<?php
function sum($num1,$num2) {
$total = $num1 + $num2;
//returning a value
return $total;
}
echo "1 + 2 = ".sum(1,2);
?>
 </body>
</html>
```

PHP Superglobals

- PHP Superglobals are built-in variables that are always available in all scopes.
- They can be accessed from any function, class or file without having to do anything special.
- $GLOBALS
- $_POST
- $_GET
- $_SESSION

Example 12

```
<?php
$x = 75;
$y = 25;

function addition() {
   $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

$_POST and $_GET

- PHP $_POST and $_GET are widely used to collect form data after submitting an HTML form with method="post“ or “get”.
- $_POST is also widely used to pass variables.

Example 13 – details.html

```
<!DOCTYPE html>
<html>
<body>
<form action="displaydetails.php" method="post">
Name: <input type="text" name="txt_name" />
Course: <input type="text" name="txt_course" />
<input type="submit" />
</form>
</body>
</html>
```

Example 14 – displaydetails.php

```
<!DOCTYPE html>
<html>
<body>
Hi <?php echo $_POST["txt_name"];
?>.<br />
You are following the course<?php echo
$_POST["txt_course"]; ?>.
</body>
</html>
```

If method="get" was used

```
<!DOCTYPE html>
<html>
<body>
Hi <?php echo $_GET["txt_name"];
?>.<br />
You are following the course<?php echo
$_GET["txt_course"]; ?>.
</body>
</html>
```

References

- http://www.html5tutorial4u.com
- Http://www.w3schools.com
- http://www.html5rocks.com

Questions?