Full Stack Web development

Lecture 4: Introduction to PHP

Today:

- What is PHP / Server Side Programming?
- Why do we use it?
- Basics
  - Variables, Arrays, Control Structures
- LAB: How do we run a local PHP installation?
  - Installing WAMP or MAMP.

What is PHP?

- PHP is a server-side programming language
- PHP is run on the web server (magic pc in cloud ;))
- PHP formats manipulates and generates the resulting code
- That result is sent to the browser as <u>just</u> HTML, JavaScript and CSS.

Example:

```
<?php
echo “Hello ECC!!”;
?>

Output: Hello ECC!!
```

Why do we need server-side programming?

- JavaScript is ALWAYS exposed to the end user (hacker)
  - ˜ no security in your code
  - ˜ info like passwords are freely available to automated programs or even anyone who knows how to view the source.
  - ˜ Server-side programming fixes this issue by keeping private information and algorithms on the server (not exposed)
- Heavy client-side (in browser) processing slows down the site.
  - Server-side programming fixes this by doing some processing on the server, making the browser have to do much less work.

Basic PHP Syntax

- Files have a .php extension (not .html)
- You must run them from a web server
  - ˜ If you try to load them in the browser, they will not load!
- PHP is <u>**Case Sensitive**</u>!
- Comments in code:
  - // a single line comment
  - # another single line comment
  - /* a multiline comment starts with a slash asterisk and ends with a asterisk slash like this */

Basic PHP Syntax

- All statements (not a control structures like if, while, etc!) must end with a semicolon
- Variables must start with a $ character.
  - $test = 1 // set a variable named test to the number 1
  - $a = true // initialize a variable to the Boolean value true
  - $b = “testing” // initialize a variable to the string testing
  - $B = ”test test” // a completely different variable from $b
- Scope
  - Global cannot see local! Explained in a sec…

Variable Scope

```
<?php
$name = “David”;
function x() {
   echo “My name is $name.”; # this produces an error.
   $age = 33;
}
echo “I am $age.”; # this produces an error.
?>
```

Variable Scope (global keyword)

```
<?php
$name = “David”;
function x() {
   global $name;
   echo “My name is $name.”; # this works
}
?>
```

Note: this is bad practice but sometimes you gotta do what you gotta do...

Echoing variables

```
<?php
function x() {
  $name = “David”;
  echo “My name is $name.”;
  echo “My name is “.$name.”.”;
  echo ’My name is $name’; # does not work
  echo ‘My name is ‘.$name.’.’;
}
x();
```

Arrays

```
<?php
  $y = array(“David”,”Joe”,”Beth”,”Michele”);
  echo $y[2]; // beth
  $z = array( // associative array
    ”name” => “David”,
    “age” => 38,
    ”gender” => “male”
  );
  echo $z[‘age’]; // 38
?>
```

Objects

```
<?php
class House {
  $house_number = “123”;
  $street = ”Main Street”;
  function StreetAddress() {
    return $this->house_number . “ “ .   $this->street;
  }
}

$myHouse = new House();
echo $myHouse->StreetAddress(); // prints 123 Main Street
?>
```

Control Structures (If Statement)

```php
<?php
if ($x == True) {
  // do something here
} else if ($x == False) {
  // do something here too
} else {
  // do something else here
}
?>
```

Control Structures (While Loop)

```
<?php
$x = 0;
while ($x < 10) {
  echo $x;
  $x++; // same as $x = $x + 1
}
?>
```

Control Structures (For Loop)

```php
<?php
for ($x = 0; $x < 10; ++$x) {
  echo $x;
}
?>
```

Control Structures (Foreach Loop)

```
<?php
$x = array(“David”,”Mike”,”Emma”,”Sabrina”);
foreach ($x as $name) {
  echo “Hello “ . $name;
}
?>
```

Control Structures (Switch Statement)

```
<?php
$name = “Britney”;
switch($name) {
   case “David”:
      echo “Hello Mr. “.$name;
      break;
   case “Britney”:
   case “Allison”:
      echo “Hello Ms. “.$name;
      break;
   default:
      echo “Hello “ . $name;
}
?>
```

Superglobals

$_SERVER – A very important Superglobal variable that allows access to items such as the current domain name, URL, referrer, the users browser, the protocol, and the physical path on the server to the current file.

$_POST – The variables submitted to a given PHP page by way of the POST method inside a form.

$_GET – The variables submitted to a given PHP page by way of the GET method inside a form.

$_REQUEST – All variables from both $_POST and $_GET.

$_SESSION – Variables tied to a specific user session.

Includes

```
<body>
<?php include ‘header.php’; ?>
<h1>Homepage text.</h1>
</body>
```

header.php can contain something like a shared navigation bar or something like that.

JSON in PHP!

```
$obj = json_decode($json_string, true);
$obj2 = array(
  “test” => true,
  “dev” => false
);
$obj3 = json_encode($obj2); #obj3 is a json
string
{“test”:true,”dev”:false}
```

File Handling

```
$file = fopen($filename,”r”); # opens a file
echo fread($file); #prints out the file content
fwrite($file, “test”); #writes contents to a
file.
fclose($file); #closes a file
```

Sending email

```
<?php
// The message
$message = "Line 1\r\nLine 2\r\nLine 3";

// In case any of our lines are larger than 70 c
haracters, we should use wordwrap()
$message = wordwrap($message, 70, "\r\n");

// Send
mail(‘adkinsd@ecc.edu', ’A Subject', $message);
?>
```

Resources

https://www.w3schools.com/php/default.asp

https://www.php.net/manual/en/index.php

WARNING: There is a lot of bad, outdated information online with PHP. I highly recommend these two sites first. If you do search Google, always search for PHP7, not just PHP.

Demo

- Installing PHP / WAMP / MAMP on your own computer.
  - Windows: C:\wamp\www
  - Mac: /Applications/MAMP/htdocs

- Using some of these functions.

- Revisiting the HTML Form!

Homework

- Your homework for this class is to simply get WAMP or MAMP up and running on your computer and to build a basic form submission for your Project 1 page. Do some research on your own on how to do this and in the next class I'll show you show I would build it.