PHP Scripting Introduction

Making your Webpages Smarter

Limitations of HTML

- It is a “markup” language
  - i.e. can only render the small set (<100 tags) that are defined for it
- Is therefore inherently “stupid”
  - It cannot make decisions, or dynamically create page content
- Since the WWW earliest inceptions this was considered unacceptable
  - What was the solution?

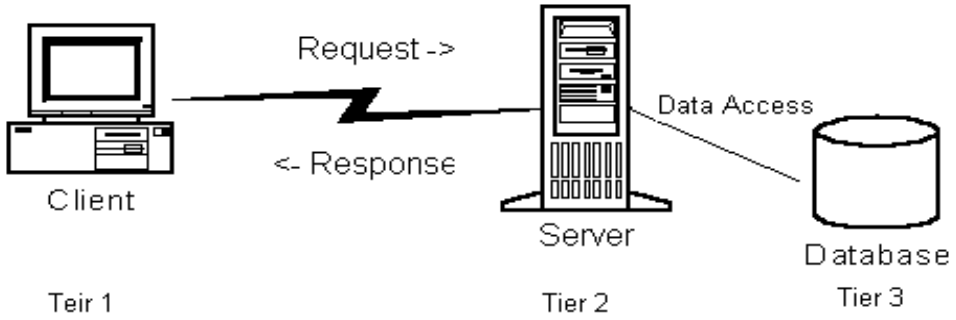
Web scripting/programming Languages?

- Developers came up with the technologies to incorporate decision making with web page content
- Takes two different models:
  - Client side
  - Server side

    The “side” refers to where the code is processed
- Also take on two different flavours:
  - Interpreted (scripting)
  - Compiled (higher level programming languages)

3-Tier Architecture Review

- Most of the Internet is based on a 3-tier architecture: including client, server database
  - Client side
    - Local machine running a web browser (IE, Netscape, Safari, Mozilla, Opera…)
    - Big scripting language: JavaScript
  - Server side
    - High-powered computer running web server (Apache or IIS are the big ones)
    - server can be configured to run different web applications/technologies
    - Several popular/powerful scripting technologies (PHP, JSP, ASP to name a few)



What is PHP?

- Originally named: *Personal Home Page Tools*
- Now stands for: *PHP: Hypertext Preprocessor*
- Open-source, server-side, HTML-embedded Web scripting
- Free, full-featured, stable, fast, cross-platform, easy to learn (looks like C++)
- Very popular, it compensates for HTML inherent “stupidity”

PHP Facts

- According to wikipedia: http://en.wikipedia.org/wiki/PHP
  - As of January 2013, PHP was used in more than 240 million websites (39% of those sampled) and was installed on 2.1 million web servers
  - As of February 2014, 82% of websites (whose server-side programming language was known) used PHP.
  - Recognized as the one of the most popular Apache Web Server add-ons
  - Parts of significant websites are written in PHP, including :
    - Facebook
    - Tumblr
    - Content management systems (CMS): Droopal, Joomla, WordPress
    - YouTube (originally)

What can PHP do?

- Perform system functions
- Gather information from forms
- Access databases
- Access/Modify cookies
- Start/Use sessions
- User authentication
- Encrypt Data
- Create/modify images

PHP Tags

- The web browser, when it comes across files with a *.php extension looks for special tags

```
<?      ?>
<?php   ?>  //use these in this course
<script language="php"> </script>
```

- Processes what is inside the tags, then sends html (i.e. text) to the requesting browser

PHP Syntax

- Variables can be of three (3) basic types:
  - Strings
  - Integers
  - Floating point precision numbers
- Variables are declared always in the format:

  $*variable_name* = "*intial_value*";

  - Notice: No data type required
- Variable naming rules:
  - A variable name must start with a letter or an underscore "_"
  - A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _)
  - A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my_name) C++, or with capitalization ($myName)
  - Same as C++ naming

PHP Arithmetic Operators

- `+ $b = $a + 3;` adds values
- `- $b = $a – 3;` subtracts values
- `* $b = $a * 3;` multiplies values
- `/ $b = $a / 3;` divides values
- `% $b = $a % 3;` modulus operator
  - modulus determines the remainder

PHP Assignment Operators

=

- assignment operator, place the value of the right-hand operand into the memory spot of the left-hand operand

+=

- addition assignment operator

-=

- subtraction assignment operator

.=

- string concatenation (the period will append the right-hand operand to the end of the left-hand operand

PHP Comparison Operators a.k.a. Relational Operators

- == logical equal to
- != not equal to
- > greater than
- < less than
- >= greater than or equal to
- <= less than or equal to

PHP Logical Operators

- && Logical AND
  - Implies true if and only if both operands are true
- || Logical OR
  - Implies true if either operand is true

PHP Provided Functions

- PHP comes with a vast collection of pre-defined functions:
  - http://ca3.php.net/manual/en/funcref.php
- One that we will use extensively is echo() which will output text to a page (similar to cout in a C++ console application
  - http://ca3.php.net/manual/en/function.echo.php

simple_example.php

```
<html>
  <head>
     <title>First PHP Page</title>
  </head>
  <body>
     <h1><?php echo “Hello World!”; ?></h1>
  </body>
</html>
```

`require()` and `include()`

- Allow you to place external file content into the calling page
- Main difference: if an “include”d file is not found the page will still load, whereas a “require”d file that is not loaded will cause the page to stop executing
- Both give you the option of centralizing functionality (PHP functions) and/or page content

header.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
  lang="en">
     <head>
  <meta http-equiv="content-type" content="text/html;
  charset=UTF-8"/>
  <title>A Page Built from Separate Files</title>
  <link rel="stylesheet" type="text/css"
  href=“css/webd2201.css"/>
</head>
<body>
```

footer.html

```
  <a
  href="http://validator.w3.org/check?uri=referer">
  <img src="http://www.w3.org/Icons/valid-xhtml10"
                alt="Valid XHTML 1.0 Strict" />
  </a>
</body>
</html>
```

built_page1.php

```
<?php
include(“header.html");
echo "<h1>First Build Page</h1>";
include(“footer.html");
?>
```

Effectively

- What is sent to the browser is:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   <head>
   <meta http-equiv="content-type" content="text/html; charset=UTF-
   8"/>
   <title>A Page Built from Separate Files</title>
   <link rel="stylesheet" type="text/css" href=“css/webd2201.css"/>
</head>
<body>
<h1>First Build Page</h1>
   <a href="http://validator.w3.org/check?uri=referer">
   <img src="http://www.w3.org/Icons/valid-xhtml10"
                alt="Valid XHTML 1.0 Strict" />
   </a>
</body>
</html>
```

What about the Title?

- For each page, there should be/could be page specific details
  - E.g. title, comments, banners etc
- If the pages that are “include”d need to make decisions on the fly, they CANNOT be HTML

header.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
  lang="en">
  <head>
  <meta http-equiv="content-type" content="text/html;
  charset=UTF-8"/>
      <title><?php echo $title; ?></title>
<link rel="stylesheet" type="text/css" href="css/webd2201.css"/>
</head>
<body>
```

footer.php (no change)

```
  <a
  href="http://validator.w3.org/check?uri=referer">
  <img src="http://www.w3.org/Icons/valid-xhtml10"
                alt="Valid XHTML 1.0 Strict" />
  </a>
</body>
</html>
```

built_page2.php

```
<?php
/*
$title has to be declared before the include or it
will
not be defined when the echo tries to display it in
header.php
*/
$title = “A Built Page with a Dynamic Title”;
include(“header.php");
echo "<h1>Second Build Page</h1>";
include(“footer.php");
?>
```

Effectively

- What is sent to the browser is:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
   <head>
   <meta http-equiv="content-type" content="text/html; charset=UTF-
   8"/>
        <title> A Built Page with a Dynamic Title</title>
   <link rel="stylesheet" type="text/css" href="css/webd2201.css"/>
</head>
<body>
<h1>Second Build Page</h1>
   <a href="http://validator.w3.org/check?uri=referer">
   <img src="http://www.w3.org/Icons/valid-xhtml10"
                alt="Valid XHTML 1.0 Strict" />
   </a>
</body>
</html>
```