WEBD 236
Web Information Systems Programming

Week 1

Agenda

- Course overview
- This week's expected outcomes
- This week's topics
- This week's homework
- Upcoming deadlines
- Questions and answers

Introductions

- Prof. Scott Sharkey
  - Adjunct faculty @ Franklin
  - WEBD, ISEC Instructor
- Industry experience in software development, systems administration, web development, and networking

Course Overview

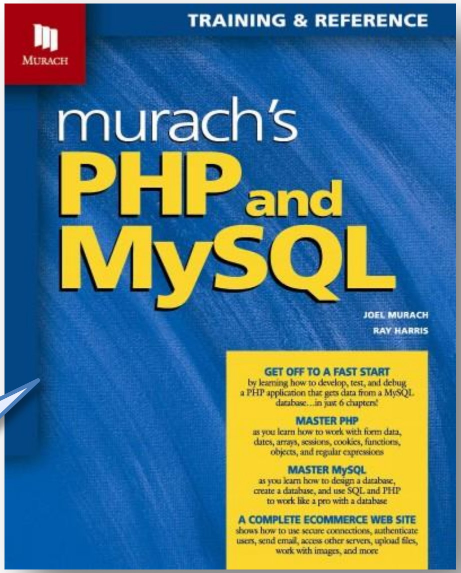
- Course Outcomes
  - Design, code, test, and debug programs using a server-based scripting language.
  - Persist objects in a relational database.
  - Compare and contrast Model 1 and Model 2 web-architectures.

Course Overview

- Course Outcomes
  - Implement object-oriented model, view, and controller components.
  - Implement basic security techniques for web information systems.

Course Overview

- Book
  - Primary: Murach's PHP and MySQL
- Additional
  - Safari
  - OhioLINK Electronic Book Center



TRAINING & REFERENCE

MURACH

murach's PHP and MySQL

JOEL MURACH
RAY HARRIS

GET OFF TO A FAST START
by learning how to develop, test, and debug a PHP application that gets data from a MySQL database...in just 6 chapters!

MASTER PHP
as you learn how to work with form data, dates, arrays, sessions, cookies, functions, objects, and regular expressions

MASTER MySQL
as you learn how to design a database, create a database, and use SQL and PHP to work like a pro with a database

A COMPLETE ECOMMERCE WEB SITE
shows how to use secure connections, authenticate users, send email, access other servers, upload files, work with images, and more

I expect that you will have read the chapters BEFORE the Franklin*Live* session for the week.

Course Overview

- Why is this course important?
  - The web as an information system
    - HTML/JavaScript front end
    - PHP (or Java/C#/Ruby) application logic
    - MySQL (or Postgres/Oracle) database back end
  - Primary programming model today

Why PHP/MySQL?

- Ubiquity
  - LAMP is a very popular stack
  - Free tools and environments
  - Similarity to other programming languages
  - No scaffolding to write web apps
- Could have used Ruby/Rails, etc.
- Changes
  - We'll use SQLite for our database (but everything will work with it).

Course Overview

- Course Structure
  - Lots of practice ("shampoo" method)
    - Reading
    - FranklinLive presentations
    - Homework Exercises
    - Lab Exercises
    - Midterm exam
    - Final exam

Increasing difficulty and point value

Course Overview

- Tools you will need
  - NetBeans Integrated Development Environment (IDE)
  - A standards-compliant web browser (Chrome, Firefox, IE9)
  - XAMPP (Apache, MySQL)
  - SQLite Expert Personal (SQL Editor)
  - Your textbooks
  - Patience and willingness to experiment!

FRANKLIN UNIVERSITY

What you should already know

- A substantive amount of HTML
  - Well formed documents
  - Tags and attributes
  - Forms, tables
- A basic amount of CSS
  - Fonts
  - Colors
  - Selectors

What you should already know

- A substantive amount of programming
  - JavaScript (no, we don't use this)
    - Documentation and style
    - Variables (scalar and array-based)
    - Selection/repetition structures
    - Functions (defining and calling)
    - Algorithms
    - Libraries
  - Problem solving, software lifecycle

What you should already know

- A substantive amount about databases
  - ERD modeling
  - Normalization (1st, 2nd, 3rd normal forms)
  - SQL
    - SELECT, INSERT, UPDATE, DELETE
    - Constraints
    - Primary and foreign keys

What you will learn

- 50,000 foot view
  - How to tie together HTML, programming, and databases to produce real working web applications!

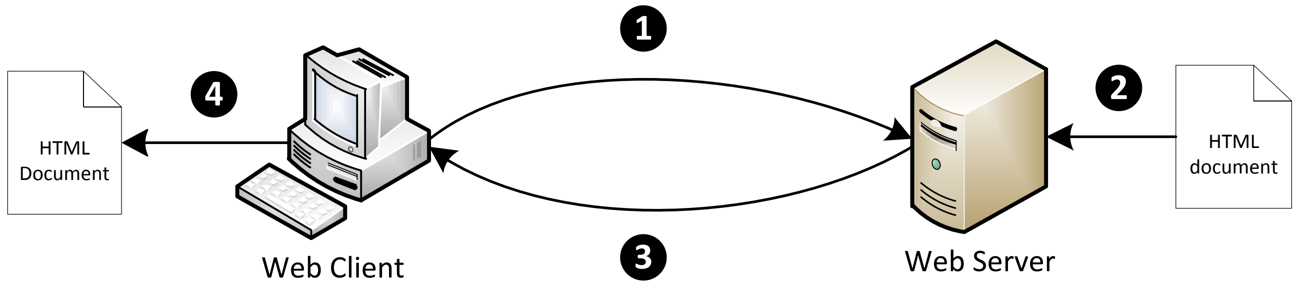
Let’s dive in!

Week 1 Outcomes

- Install and use a web database development environment.
- Describe the request/response cycle.
- Distinguish between POST and GET web methods.
- Employ a form and server-side processing to solve a simple programming problem.

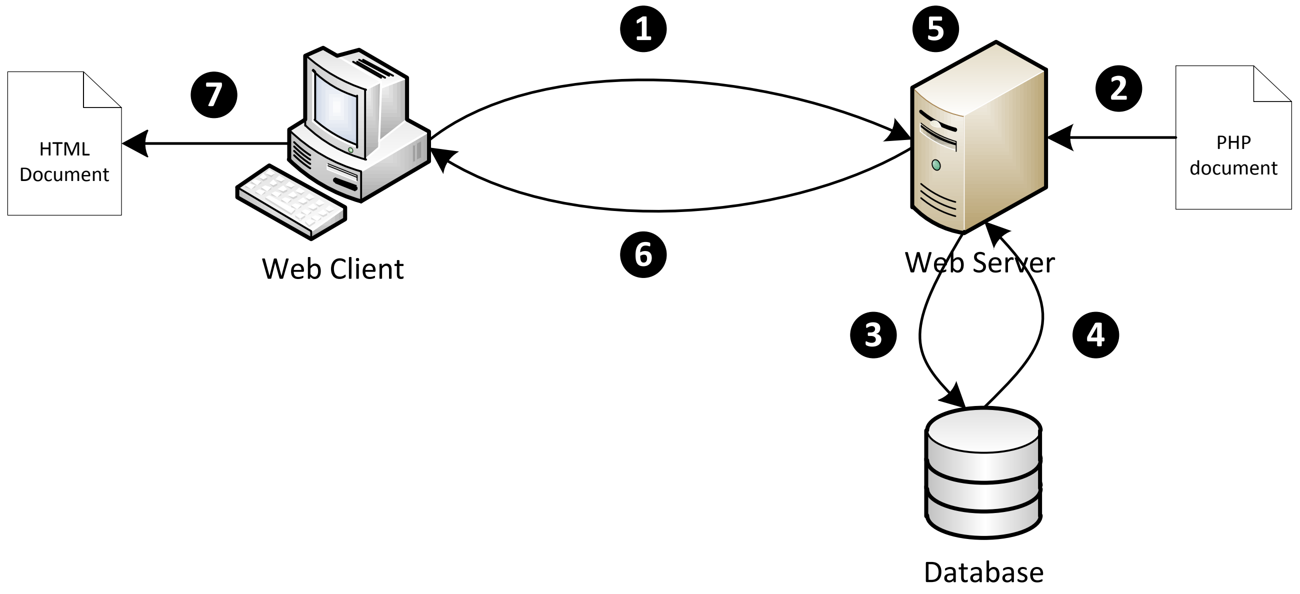
Request/Response Cycle

- Static content (HTML, CSS, JS, etc.)



Request/Response Cycle

- Dynamic content (PHP, CGI, etc.)



Protocols

- Protocol – a language
  - HTTP: hypertext transfer protocol – application layer, used by web servers and browsers.
  - TCP: transmission control protocol – transport layer, reliable communications between processes
  - IP: internet protocol – best effort communications between hosts

HTTP

- Request

```
GET / HTTP/1.1
Host: www.franklin.edu
```

- Response

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 136
Server: Apache/2.2.3

<html><head>...
```

HTTP

- Request

```
GET / HTTP/1.1
Host: www.frank
```

- Response

```
HTTP/1.1 200 O
Content-Type: text/html
Content-Length: 136
Server: Apache/2.2

<html><head>...
```

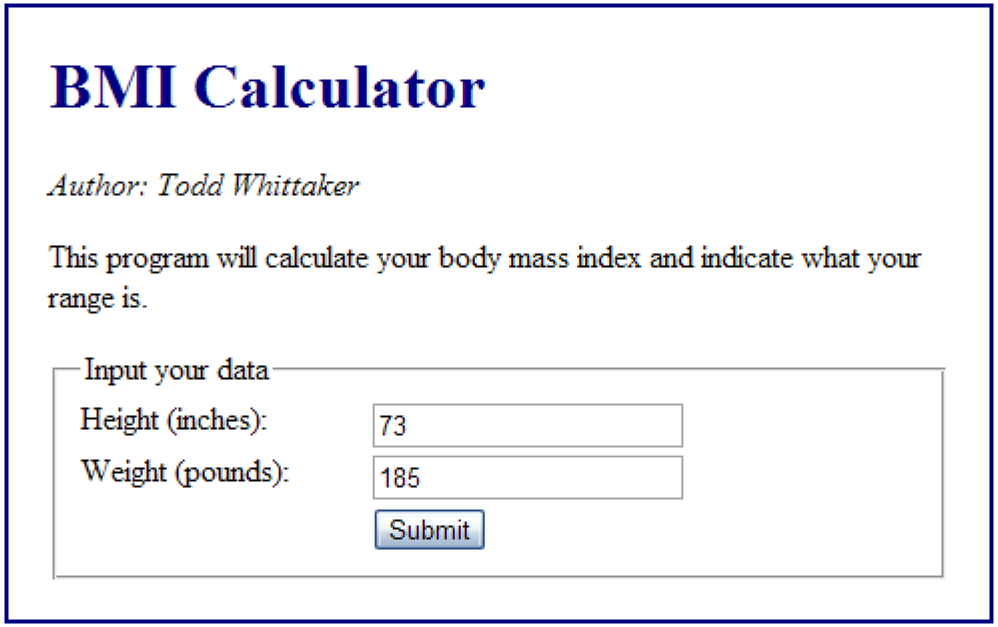
HTML is a language transmitted inside the HTTP protocol, which is inside the TCP protocol, which is inside the IP protocol, etc.

Simple PHP Workflow

- Workflow
  - HTML page has a form
  - Form submits to a PHP page for processing
  - PHP page then
    - Does some calculations (including DB access)
    - Produces HTML
  - HTML returned to browser

First Example: BMI Calculator

- User interface



BMI Calculator

Author: Todd Whittaker

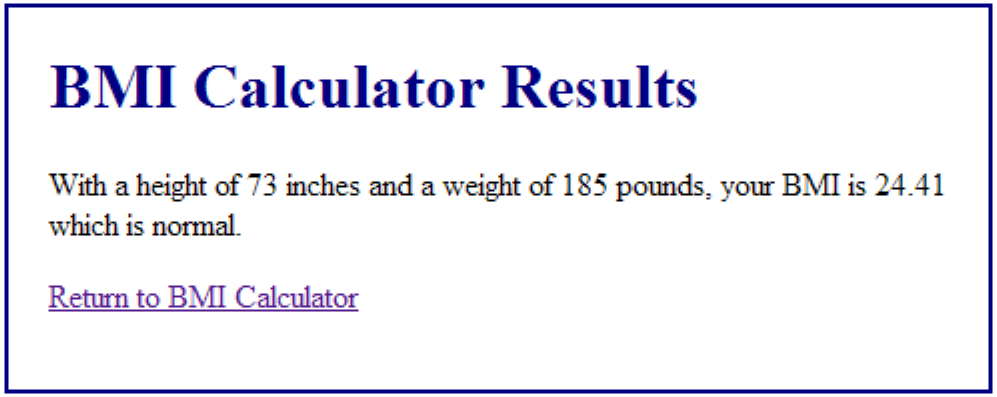
This program will calculate your body mass index and indicate what your range is.

Input your data

Height (inches):	73
Weight (pounds):	185

First Example: BMI Calculator

- User interface



BMI Calculator Results

With a height of 73 inches and a weight of 185 pounds, your BMI is 24.41 which is normal.

Return to BMI Calculator

First Example: BMI Calculator

```
<!DOCTYPE html>
<html>
  <head>
    <title>BMI Calculator</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <div id="content">
      <h1>BMI Calculator</h1>
      <p><em>Author: Todd Whittaker</em></p>
      <p>This program will calculate your body mass index
      and indicate what your range is.</p>
```

index.html

First Example: BMI Calculator

```
      <form action="bmi.php" method="post">
        <fieldset>
          <legend>Input your data</legend>
          <label for="height">Height (inches):</label>
          <input type="text" id="height" name="height" /><br />
          <label for="height">Weight (pounds):</label>
          <input type="text" id="weight" name="weight" /><br />
          <label>&nbsp;</label>
          <input type="submit" value="Submit" /><br />
        </fieldset>
      </form>
    </div>
  </body>
</html>
```

First Example: BMI Calculator

```css
#content {
  width: 450px;
  margin: 0 auto;
  padding: 0px 20px 20px;
  background: white;
  border: 2px solid navy;
}
h1 {
  color: navy;
}
label {
  width: 8em;
  padding-right: 1em;
  float: left;
}
```

style.css

First Example: BMI Calculator

```
<?php
function safeParam($key, $default) {
    if (isset($_POST[$key]) && $_POST[$key] != "") {
        return htmlentities($_POST[$key]);
    } else if (isset($_GET[$key]) && $_GET[$key] != "") {
        return htmlentities($_GET[$key]);
    } else {
        return $default;
    }
}
```

First Example: BMI Calculator

```
function categoryFor($bmi) {
    $result = "";
    if ($bmi < 16) {
        $result = "severely underweight";
    } else if ($bmi <= 18.5) {
        $result = "underweight";
    } else if ($bmi <= 25) {
        $result = "normal";
    } else if ($bmi <= 30) {
        $result = "overweight";
    } else {
        $result = "obese";
    }
    return $result;
}
```

bmi.php

First Example: BMI Calculator

```
$height = safeParam('height', 1);
$weight = safeParam('weight', 0);
$bmi = (703 * $weight) / ($height * $height);
$bmiCategory = categoryFor($bmi);
?>
<!DOCTYPE html>
<html>
  <head>
    <title>BMI Calculator Results</title>
    <link rel="stylesheet" href="style.css" />
  </head>
```

First Example: BMI Calculator

```
  <body>
    <div id="content">
      <h1>BMI Calculator Results</h1>
      <p>With a height of <?php echo $height ?>
      inches and a weight of <?php echo $weight ?>
      pounds, your BMI is <?php echo number_format($bmi,2) ?>
      which is <?php echo $bmiCategory ?>.</p>
      <p><a href="index.html">Return to BMI Calculator</a></p>
    </div>
  </body>
</html>
```

Quick Tip

- DRY Principle
  - To avoid repeated code, use includes

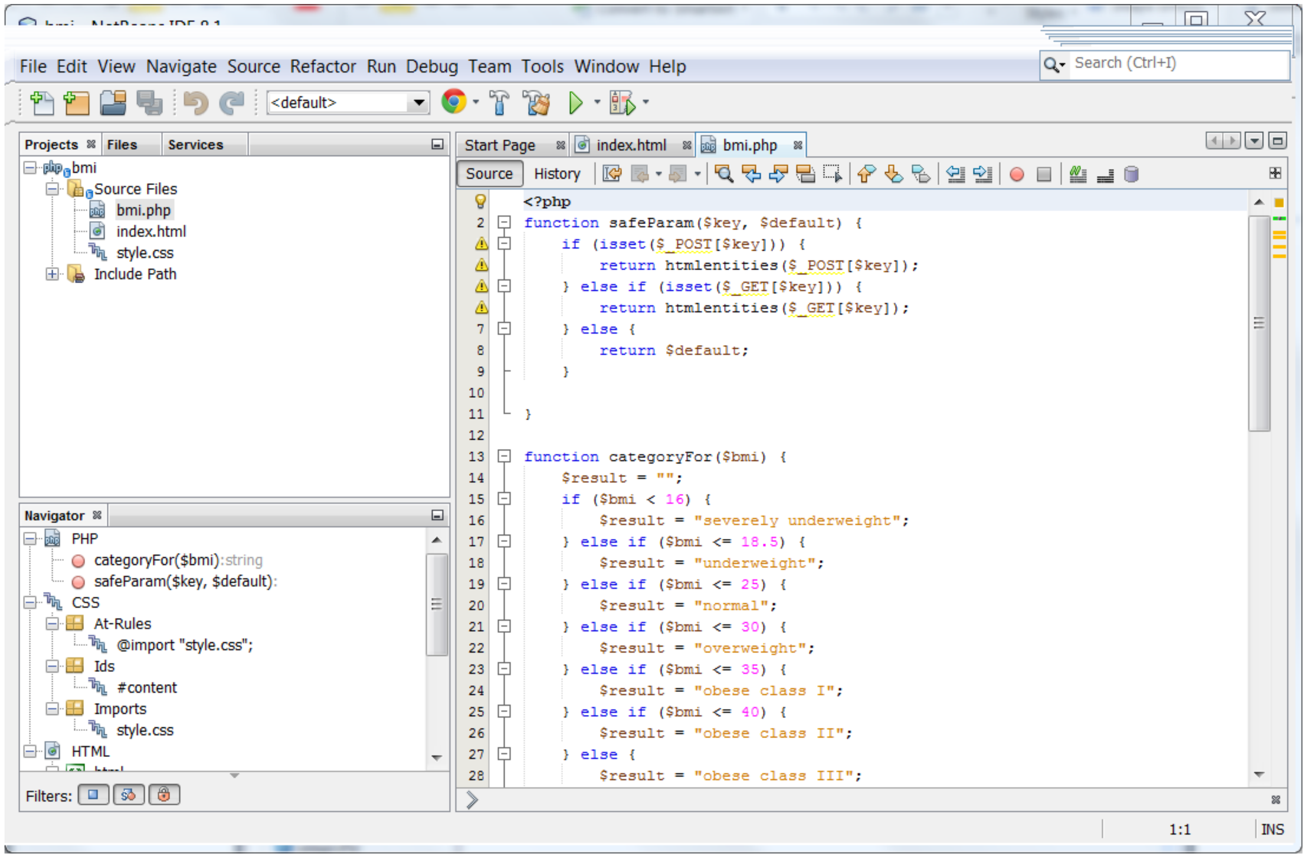
```php
<?php
function getMeaning() {
    return 42;
}
?>
```

In the file useful.inc

```php
<?php
include 'useful.inc';
# Now we can call getMeaning...
?>
```

In the file index.php

NetBeans for editing

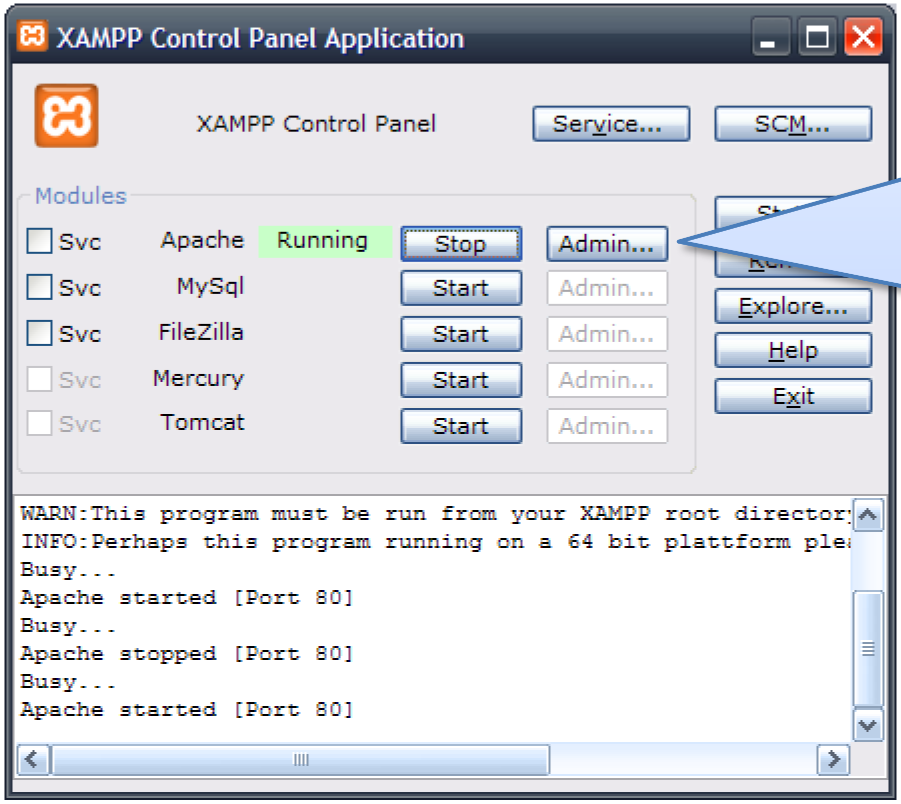


```php
<?php
function safeParam($key, $default) {
    if (isset($_POST[$key])) {
        return htmlentities($_POST[$key]);
    } else if (isset($_GET[$key])) {
        return htmlentities($_GET[$key]);
    } else {
        return $default;
    }

}

function categoryFor($bmi) {
    $result = "";
    if ($bmi < 16) {
        $result = "severely underweight";
    } else if ($bmi <= 18.5) {
        $result = "underweight";
    } else if ($bmi <= 25) {
        $result = "normal";
    } else if ($bmi <= 30) {
        $result = "overweight";
    } else if ($bmi <= 35) {
        $result = "obese class I";
    } else if ($bmi <= 40) {
        $result = "obese class II";
    } else {
        $result = "obese class III";
```

XAMPP


XAMPP Control Panel Application
XAMPP Control Panel
Service...
SCM...
Modules
Svc Apache Running Stop Admin...
Svc MySql Start Admin...
Svc FileZilla Start Admin...
Svc Mercury Start Admin...
Svc Tomcat Start Admin...
Explore...
Help
Exit

```
WARN:This program must be run from your XAMPP root director
INFO:Perhaps this program running on a 64 bit plattform ple
Busy...
Apache started [Port 80]
Busy...
Apache stopped [Port 80]
Busy...
Apache started [Port 80]
```

Files under c:\xampp\htdocs are served by Apache. If you make this your "workspace" directory in NetBeans, you can edit live apps and just refresh your browser for testing

Basic PHP - tags

- PHP is intermixed with HTML in a single file
  - Code is contained within **<?php** and **?>** tags.
    - HTML can be printed to the document within the tags.
  - Static HTML/JavaScript is outside those tags.

```
<?php
for ($i = 0; $i < 10; ++$i) {
    print "Hello <br />\n";
}
?>
```

```
<?php
for ($i = 0; $i < 10; ++$i) {
?>
  Hello <br />
<?php
}
?>
```

Basic PHP - comments

- Three kinds of comments

Like all PHP, comment's don't appear in the rendered output. HTML comments will.

```
<?php
/**
 * This is a multi-line comment.
 * Use this to document functions and files.
 */

 $x = 1;  // This is a comment to EOL.
 $y = 1;   # As is this kind of comment.
?>
<!-- This is an HTML comment. -->
```

Basic PHP - variables

- Variables
  - All variables start with a ‘$’ symbol.
  - Naming conventions apply
    - Avoid keywords (i.e. $if is confusing)
    - Names should reflect their use
  - Scope
    - Global scope vs. function scope
    - To access a global variable within a function, use the global keyword.

Basic PHP – data types

- Data types
  - Integer
  - Double
  - Boolean
  - String
  - Array
  - Object

Basic PHP – data types

- Data types
  - Integer
  - Double
  - Boolean
  - String
  - Array
  - Object

Two kinds of strings: single and double quoted strings.

```
<?php
 $x = 'World';
 print "Hello $x!<br />\n";
 print 'Hello $x!<br />\n';
?>
```

Double quoted strings expand special characters and variables. Single quoted do not.

Basic PHP – data types

- Type juggling
  - PHP data types are determined by context

```
<?php
$foo = "0";  // $foo is string (ASCII 48)
$foo += 2;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
?>
```

Source: http://php.net/manual/en/language.types.type-juggling.php

Basic PHP – data types

- Type casting
  - Can force manual type conversion

```
<?php
$foo = 7.7;
$bar = (boolean) $foo;
$baz = (integer) $foo;
print "foo=$foo, bar=$bar, baz=$baz";
?>
```

```
foo=7.7, bar=1, baz=7
```

Basic PHP – “truthiness”

- False values
  - What evaluates to false after type juggling?
    - null
    - 0
    - 0.0
    - "0"
    - false
    - Empty arrays
  - Everything else is true.

Basic PHP – undeclared variables

- Undeclared variables trigger warnings, but execution continues

```
<?php
if ($x) {
    print "Hello.";
}
?>
```

Notice: Undefined variable: x
in **C:\xampp\htdocs\Scratch\foo.php** on line **2**

Basic PHP – undeclared variables

- Undeclared variables trigger warnings, but execution continues

```
<?php
if (isset($x)) {
    print "Hello.";
}
?>
```

- Use isset() or empty() to determine if a variable has a value

Basic PHP – empty vs isset

- Juggling vs. empty() vs. isset()

value	if()	empty()	isset()
null	false	true	false
0	false	true	true
0.0	false	true	true
"0"	false	true	true
""	false	true	true
false	false	true	true
array()	false	true	true
other stuff	true	false	true

Source: http://phabricator.com/docs/phabricator/article/PHP_Pitfalls.html

FRANKLIN UNIVERSITY

Basic PHP – operators

- Operators
  - Just like in most descendents of C
  - Mathematical operators, remember precedence
    - +, -, *, /, %, ++, --
  - Assignment and augmented assignment
    - =, +=, -=, *=, /=, %=
  - Use parentheses to change precedence.

FRANKLIN UNIVERSITY

Basic PHP – operators

- Operators
  - String concatenation
    - . (that’s a period, **not** a ‘+’)
  - Logical operators
    - &&, ||, !
    - and, or, xor
  - Relational operators
    - <, >, <=, >=, ==, !=
    - ===, !==

These operators also compare types without juggling!

Basic PHP – control flow

- Control flow: selection
  - if, if/else just as with most languages.

```
<?php
$x = 5;
if ($x > 7) {
    print "foo";
} else if (x % 2 == 1) {
    print "bar";
} else {
    print "baz";
}
?>
```

Basic PHP – control flow

- Control flow: repetition
  - while just as with most languages.

```
<?php
$x = 0;
while ($x < 10) {
    print "foo $i<br />";
    ++$x;
}
?>
```

Basic PHP – control flow

- Control flow: repetition
  - for just as with most languages.

```
<?php
for ($i = 0; $i < 10; ++$i) {
   print "foo $i<br />";
}
?>
```

Note, for loops do not introduce scope. $i is visible and has value 10 after this loop.

Basic PHP – control flow

- Control flow: repetition
  - foreach similar to for/in in JavaScript.

```
<?php
$arr = array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
foreach ($arr as $value) {
    print "foo $value<br />";
}
?>
```

Basic PHP – modularization

- Modularization
  - Functions: similar to most language
    - Variables can be passed by value or reference (precede parameter by &).
    - Can return a single value.

```
<?php
function fibonacci($num) {
    $i = 0;
    $j = 1;
    while ($num > 0) {
        $sum = $i + $j;
        $i = $j;
        $j = $sum;
        --$num;
        print "$i, ";
    }
    return $i;
}
fibonacci(20);
?>
```

Basic PHP - modularization

- Modularization
  - Separate files
    - Group related functions and variables into a file
    - Name the file with the “.inc” extension instead of “.php” (by convention, not necessity)
    - Import the contents of one file into another with the include keyword:

```
<?php
include 'fibonacci.inc';
# now we have access to the function
?>
```

Could also use include_once to avoid redefinition.

Basic PHP - modularization

– Separate files
  • Also very useful for extracting common content:

```
<!DOCTYPE html>
<html>
   <head>
      <title>My web site</title>
      <link rel="style.css" type="text/css" href="style.css" />
      <script type="text/javascript" src="jquery-1.3.2.js"></script>
   </head>
   <body>
      <div id="container">
```

header.inc

```
      </div> <!-- container -->
      <div id="footer">
         <p>
            Copyright &copy; 2012-2017 Todd A. Whittaker
         </p>
      </div>
   </body>
</html>
```

footer.inc

Basic PHP – termination

- Termination
  - Can immediately stop processing a script using exit or die.
    - Nothing from that point down will execute
    - Server immediately returns whatever has been rendered so far. Useful for redirects:

```php
<?php
function redirect($url) {
    header("Location: $url");
    exit();
}
?>
```

Basic PHP – reserved words

- Key words

abstract	and	array	as	break
case	catch	class	clone	const
continue	declare	default	do	else
elseif	endswitch	endwhile	extends	final
for	foreach	function	global	goto
if	implements	interface	instanceof	namespace
new	or	private	protected	public
static	switch	throw	try	use
var	while	xor		

Basic PHP – reserved words

- Language constructs

die	echo	empty	exit
eval	include	include_once	isset
list	require	require_once	return
print	unset		

Basic PHP – reserved words

- Compile time constants

__CLASS__	__DIR__	__FILE__	__LINE__
__FUNCTION__	__METHOD__	__NAMESPACE__	

Basic PHP – GET and POST

- Submitting a form can use the method GET or POST for transmitting data
  - GET method appends values to the URL
    - Ex: http://localhost/bmi/bmi.php?height=73&weight=185
    - Used when the request doesn't change state on the server (i.e. no database writes)
  - Post method puts the values inside the HTTP request
    - Used when the request changes state on the server

Basic PHP – GET

- Contents of an HTTP GET request

```
GET http://localhost/bmi/bmi.php?height=72&weight=185 HTTP/1.1
Host: localhost
Connection: keep-alive
User-Agent: Chrome/16.0.912.63
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

Basic PHP – GET

- Accessing the GET parameters
  - Use the "superglobal" $_GET associative array.

```
<?php
foreach ($_GET as $key => $value) {
   print "Received parameter \"$key\" with value \"$value\"<br />";
}
$height = $_GET['height'];
$weight = $_GET['weight'];
?>
```

Really need to use isset() to check if they exist!

Basic PHP – POST

- Contents of an HTTP POST request

```
POST http://localhost/bmi/bmi.php HTTP/1.1
Host: localhost
Connection: keep-alive
Content-Length: 20
Cache-Control: max-age=0
Origin: http://localhost
User-Agent: Chrome/16.0.912.63
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: http://localhost/bmi/index.html
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

height=73&weight=185
```

Basic PHP – POST

- Accessing the POST parameters
  - Use the “superglobal” $_POST associative array.

```
<?php
foreach ($_POST as $key => $value) {
   print "Received parameter \"$key\" with value \"$value\"<br />";
}
$height = $_POST['height'];
$weight = $_POST['weight'];
?>
```

Really need to use isset() to check if they exist!

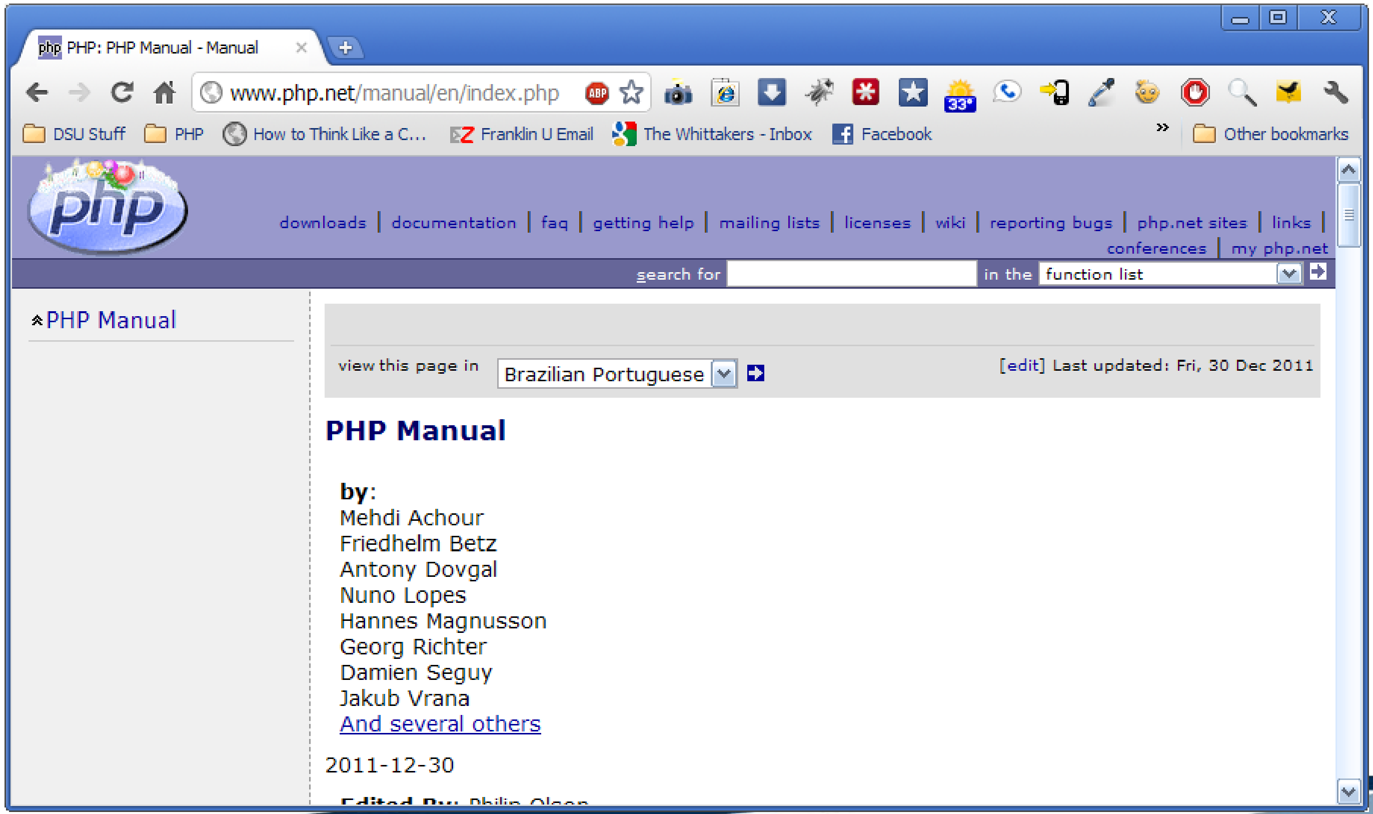
Basic PHP – superglobals

- Use $_REQUEST if method is irrelevant

```
<?php
function safeParam($key, $default) {
    if (isset($_REQUEST[$key]) && $_REQUEST[$key] != "") {
        return htmlentities($_REQUEST[$key]);
    } else {
        return $default;
    }
}
$height = safeParam('height', false);
$weight = safeParam('weight', false);
if (!($height && $weight)) {
    # generate an error message
}
?>
```

Basic PHP – documentation

- http://www.php.net/manual/en/index.php


PHP: PHP Manual - Manual
www.php.net/manual/en/index.php
DSU Stuff
PHP
How to Think Like a C...
Franklin U Email
The Whittakers - Inbox
Facebook
Other bookmarks
downloads | documentation | faq | getting help | mailing lists | licenses | wiki | reporting bugs | php.net sites | links | conferences | my php.net
search for
in the
function list

PHP Manual

view this page in Brazilian Portuguese

[edit] Last updated: Fri, 30 Dec 2011

PHP Manual

by:
Mehdi Achour
Friedhelm Betz
Antony Dovgal
Nuno Lopes
Hannes Magnusson
Georg Richter
Damien Seguy
Jakub Vrana
And several others

2011-12-30

FRANKLIN UNIVERSITY

Basic PHP – dive in!

- Maybe you're feeling like this

Basic PHP – dive in!

- Maybe you're feeling like this

"It's easy to play with PHP! Edit a script, navigate to the URL, click 'reload.'"
COOLFUNPICS.COM

Upcoming Deadlines

- Readings for next week
  - Chapters 3 and 4 in *PHP and MySQL*
- Assignments
  - Homework 1 due end of week 2
  - Homework 2 due end of week 3
  - Lab 1 due end of week 4
- Miscellaneous
  - Get your proctor information submitted!

General Q & A

- Questions?
- Comments?
- Concerns?