CSE417: WEB ENGINEERING

Daffodil International University

You Will Be Able To

- ✓ Know about Server Side Programming
- ✓ PHP and its workings.
- ✓ Apply PHP syntax
  - ✓ variables, operators, if...else...and switch, while, do while, and for.
  - ✓ Some useful PHP functions

Contents

- Basic PHP

Server-Side Dynamic Web Programming

- **CGI is one of the most common approaches to server-side programming**
  - Universal support: (almost) Every server supports CGI programming. A great deal of ready-to-use CGI code. Most APIs (Application Programming Interfaces) also allow CGI programming.
  - Choice of languages: CGI is extremely general, so that programs may be written in nearly any language. Perl is by far the most popular, with the result that many people think that CGI means Perl. But C, C++, Ruby or Python are also used for CGI programming.
  - Drawbacks: A separate process is run every time the script is requested. A distinction is made between HTML pages and code.

- **Other server-side alternatives try to avoid the drawbacks**
  - Server-Side Includes (SSI): Code is embedded in HTML pages, and evaluated on the server while the pages are being served. Add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program.
  - Active Server Pages (ASP, Microsoft) : The ASP engine is integrated into the web server so it does not require an additional process. It allows programmers to mix code within HTML pages instead of writing separate programs. (Drawback(?) Must be run on a server using Microsoft server software.)
  - Java Servlets (Sun): As CGI scripts, they are code that creates documents. These must be compiled as classes which are dynamically loaded by the web server when they are run.
  - Java Server Pages (JSP): Like ASP, another technology that allows developers to embed Java in web pages.

PHP

- **developed in 1995 by Rasmus Lerdorf (member of the Apache Group)**
  - originally designed as a tool for tracking visitors at Lerdorf's Web site
  - within 2 years, widely used in conjunction with the Apache server
  - developed into full-featured, scripting language for server-side programming
  - free, open-source
  - server plug-ins exist for various servers
  - now fully integrated to work with mySQL databases

- **PHP is similar to JavaScript, only it's a server-side language**
  - PHP code is embedded in HTML using tags
  - when a page request arrives, the server recognizes PHP content via the file extension (`.php` or `.phtml`)
  - the server executes the PHP code,
  - the resulting page is then downloaded to the client
  - user never sees the PHP code, only the output in the page

What do You Need?

- **Our server supports PHP**
  - You don't need to do anything special! *
  - You don't need to compile anything or install any extra tools!
  - Create some .php files in your web directory - and the server will parse them for you.

    * Slightly different rules apply when dealing with an SQL database (as will be explained when we get to that point).

- **Most servers support PHP**
  - Download PHP for free here: http://www.php.net/downloads.php
  - Download MySQL for free here: http://www.mysql.com/downloads/index.html
  - Download Apache for free here: http://httpd.apache.org/download.cgi

  - INSTALL XAMPP/WAMPP and get it all!!!

Basic PHP syntax

A PHP scripting block always starts with `<?php` and ends with `?>`. A PHP scripting block can be placed (almost) anywhere in an HTML document.

```
<html>
<!-- hello.php  -->
<head><title>Hello World</title></head>
<body>
 <p>This is going to be ignored by the PHP interpreter.</p>

   <?php echo '<p>While this is going to be parsed.</p>'; ?>

  <p>This will also be ignored by PHP.</p>

   <?php print('<p>Hello and welcome to <i>my</i> page!</p>');
    ?>

  <?php

   //This is a comment

  /*
   This is
   a comment
   block
  */
  ?>

</body>
</html>
```

`print` and `echo` for output

`a semicolon (;)` at the end of each statement

// for a single-line comment

/* and */ for a large comment block.

The server executes the print and echo statements, substitutes output.

Scalars

All variables in PHP start with a $ sign symbol. A variable's type is determined by the context in which that variable is used (i.e. there is no strong-typing in PHP).

```
<html><head></head>
<!-- scalars.php  -->
<body>  <p>
<?php
$foo = true; if ($foo) echo "It is TRUE! <br /> \n";
$txt='1234'; echo "$txt <br /> \n";
$a = 1234; echo "$a <br /> \n";
$a = -123;
echo "$a <br /> \n";
$a = 1.234;
echo "$a <br /> \n";
$a = 1.2e3;
echo "$a <br /> \n";
$a = 7E-10;
echo "$a <br /> \n";
echo 'Arnold once said: "I\'ll be back"', "<br /> \n";
$beer = 'Heineken';
echo "$beer's taste is great <br /> \n";
$str = <<<EOD
Example of string
spanning multiple lines
using "heredoc" syntax.
EOD;
echo $str;
?>
</p>
</body>
</html>
```

Four scalar types:

boolean
  true or false

integer,

float,
 floating point numbers

string
 single quoted
 double quoted

Arrays

An array in PHP is actually an ordered map. A map is a type that maps values to keys.

```
<?php
$arr = array("foo" => "bar", 12 =>
true);
echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

`array()` = creates arrays

key = either an integer or a string.

value = any PHP type.

```
<?php
array(5 => 43, 32, 56, "b" => 12);
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

if no key, the maximum of the integer indices + 1.

if an existing key, its value will be overwritten.

```
<?php
$arr = array(5 => 1, 12 => 2);
$arr[] = 56;     // the same as $arr[13] = 56;
$arr["x"] = 42; // adds a new element
unset($arr[5]); // removes the element
unset($arr);     // deletes the whole array
$a = array(1 => 'one', 2 => 'two', 3 => 'three');
unset($a[2]);
$b = array_values($a);
?>
```

can set values in an array

`unset()` removes a key/value pair

`array_values()` makes reindexing effect (indexing numerically)

*Find more on arrays

Constants

A constant is an identifier (name) for a simple value. A constant is case-sensitive by default. By convention, constant identifiers are always uppercase.

```
<?php

// Valid constant names
define("FOO",     "something");
define("FOO2",     "something else");
define("FOO_BAR", "something more");

// Invalid constant names  (they shouldn't start
//      with a number!)

define("2FOO",     "something");

// This is valid, but should be avoided:
// PHP may one day provide a "magical" constant
// that will break your script
define("__FOO__", "something");

?>
```

You can access constants anywhere in your script without regard to scope.

Operators

- **Arithmetic Operators:** +, -, *,/ , %, ++, --
- **Assignment Operators:** =, +=, -=, *=, /=, %=

Example	Is the same as
x+=y	x=x+y
x-=y	x=x-y
x*=y	x=x*y
x/=y	x=x/y
x%=y	x=x%y

- **Comparison Operators:** ==, !=, >, <, >=, <=
- **Logical Operators:** &&, ||, !
- **String Operators**: . and .= (for string concatenation)

```
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";
```

Conditionals: if else

Can execute a set of code depending on a condition

```
<html><head></head>
<!-- if-cond.php  -->
<body>

<?php
$d=date("D");
if ($d=="Fri")
     echo "Have a nice weekend! <br/>";
else
     echo "Have a nice day! <br/>";

$x=10;
if ($x==10)
{
     echo "Hello<br />";
     echo "Good morning<br />";
}

?>

</body>
</html>
```

if (condition)
code to be executed if condition is true;
else
code to be executed if condition is false;

date() is a built-in PHP function that can be called with many different parameters to return the date (and/or local time) in various formats

In this case we get a three letter string for the day of the week.

Conditionals: switch

Can select one of many sets of lines to execute

```
<html><head></head>
<body>
<!-- switch-cond.php  -->
<?php
$x = rand(1,5);  // random integer
echo "x = $x <br/><br/>";
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
  break;
}
?>

</body>
</html>
```

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

Looping: while and do-while

Can loop depending on a condition

```
<html><head></head>
<body>

<?php
$i=1;
while($i <= 5)
{
  echo "The number is $i <br
/>";
  $i++;
}
?>

</body>
</html>
```

loops through a block of code if, and as long as, a specified condition is true

```
<html><head></head>
<body>

<?php
$i=0;
do
{
  $i++;
  echo "The number is $i <br
/>";
}
while($i <= 10);
?>

</body>
</html>
```

loops through a block of code once, and then repeats the loop as long as a special condition is true (so will always execute at least once)

Looping: for and foreach

Can loop depending on a "counter"

```php
<?php
for ($i=1; $i<=5; $i++)
{
echo "Hello World!<br />";
}
?>
```

loops through a block of code a specified number of times

```php
<?php
$a_array = array(1, 2, 3, 4);
foreach ($a_array as $value)
{
   $value = $value * 2;
   echo “$value <br/> \n”;
}
?>
```

loops through a block of code for each element in an array

User Defined Functions

Can define a function using syntax such as the following:

```php
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Example function.\n";
    return $retval;
}
?>
```

Can return a value of any type

```php
<?php
function square($num)
{
    return $num * $num;
}
echo square(4);
?>
```

```php
<?php
function small_numbers()
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
echo $zero, $one, $two;
?>
```

```php
<?php
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
 takes_array(array(1,2));
?>
```

PHP Functions

```
<?php
 function addNumbers(int $a, int $b) {
        return $a + $b;
}
echo addNumbers(5, "5 days");
 // since strict is NOT enabled "5 days" is changed to int(5), and it will return 10
?>
```

Variable Scope

The scope of a variable is the context within which it is defined.

```
<?php
$a = 1; /* limited variable scope */
function Test()
{
    echo $a;
  /* reference to local scope variable
*/
}
Test();
?>
```

The scope is local within functions, and hence the value of $a is undefined in the “echo” statement.

```
<?php
$a = 1;
$b = 2;
function Sum()
{
    global $a, $b;
    $b = $a + $b;
}
Sum();
echo $b;
?>
```

`global`

refers to its global version.

```
<?php
function Test1()
{
    static $a = 0;
    echo $a;
    $a++;
}
Test1();
Test1();
Test1();
?>
```

`static`

does not lose its value.

Exercise

- Design a form which calculate sum of two integers given by the user.
- **READINGS/Practice**
  - M Schafer: Ch. 29, 30
  - http://uk.php.net/tut.php
  - http://www.w3schools.com/php/default.asp
  - http://php.resourceindex.com/

Acknowledgement

- This module is designed and created with the help from following sources-
  - https://cgi.csc.liv.ac.uk/~ullrich/COMP519/
  - http://www.csc.liv.ac.uk/~martin/teaching/comp519/