Part 2: JavaScript, PHP, SQL, Advanced PHP

EE4727/IM4727 Web Application Design
PHP

Lecturer :
Dr Karim
Dr Wesley Tan Chee Wah
Dr Hu Xiao

PHP+MySQL

- Many websites on the internet are implemented using PHP and MySQL.
- Recommended Book:

  Title : PHP and MySQL Web development – 4th ed.

  Authors: Luke Welling, Laura Thomson.

  ISBN 978-0-672-32916-6

  QA76.73.P224W45 2008

  Published by Addison Wesley.

What have you learned so far?

- HTML
- CSS
- Javascript

- What is coming up next?
  - Using PHP and SQL Database for Dynamic content.

What is PHP?

- PHP is server-side open source scripting language
- PHP script is case-sensitive
- PHP code is interpreted at the web server and dynametically generates HTML or other output that the visitor will see
- PHP can create, open, read, write, and close files on the server
- PHP can collect form data
- PHP supports a wide range of databases
- PHP originally stood for Personal Home Page, and now stands for PHP Hypertext Preprocessor.

What is MySQL?

- MySQL is a fast robust relational database management system (RDBMS) used on the web
- MySQL is a multiuser, multithreaded server using *Structured Query Language (SQL)*
- MySQL store data in tables.
- MySQL is ideal for both small and large applications
- MySQL compiles on a number of platforms

Why PHP+MySQL?

- PHP can add, delete, modify data in a database
- MySQL compiles on a number of platforms
- PHP works on major operating system and many minor ones
- MySQL is the most popular database system used with PHP
- PHP+MySQL are cross-platform

PHP Script and HTML

hello.php

- Embedding PHP in HTML.
  - PHP scripts are embedded between PHP tags in HTML
  - PHP code begins with "<?php" and ended with "?>".
    - Any text between the tags is interpreted by the web server as PHP.
    - Any text outside these tags is treated as normal HTML
    - You can embed JavaScript in PHP

File: hello.php

```
<!DOCTYPE html>
<html>
<body>
<?php
echo "Hello, my first PHP script!";
?>
</body>
</html>
```

File: js_hello.php

```
<?php
echo '<script type="text/javascript">
alert("Hello...from javascript");
</script>';
?>
```

Bob's Auto Parts (1) – The Order Form

orderform..html

Item	Quantity
Tires	
Oil	
Spark Plugs	

How did you find Bob's? I'm a regular customer

Submit Order

- Form action is implemented using the `action` attribute of the form tag
- The `action` is handled by PHP script `processorder.php`
- Form inputs in html are passed into PHP script by sending through the current URL via the `post` method

```html
<html>
<body>
<form action="processorder.php" method="post">
<table border="0">
<tr bgcolor="#cccccc">
  <td width="150">Item</td>
  <td width="15">Quantity</td>
</tr>
<tr>
  <td>Tires</td>
  <td align="center"><input type="text" name="tireqty" size="3"
    maxlength="3"></td>
</tr>
<tr>
  <td>Oil</td>
  <td align="center"><input type="text" name="oilqty" size="3" maxlength="3"></td>
</tr>
<tr>
  <td>Spark Plugs</td>
  <td align="center"><input type="text" name="sparkqty" size="3"
    maxlength="3"></td>
</tr>
```

Bob's Auto Parts (2) -- The Order Form

- The form `action` is registered through the `submit` attribute of the form
- On submission, the form event `action` attribute is activated and PHP code in `processor.php` run at the web server

Item	Quantity
Tires	
Oil	
Spark Plugs	
How did you find Bob's?	I'm a regular customer

Submit Order

```
<tr>
  <td>How did you find Bob's?</td>
  <td><select name="find">
        <option value = "a">I'm a regular customer</option>
        <option value = "b">TV advertising</option>
        <option value = "c">Phone directory</option>
        <option value = "d">Word of mouth</option>
      </select>
  </td>
</tr>
<tr>
  <td colspan="2" align="center"><input type="submit" value="Submit Order"></td>
</tr>
</table>
</form>
</body>
</html>
```

Bob's Auto Parts (3) – Processing the Form

processorder.php

- Form inputs are passed into script via a `post` method and declared as local variables
- The form input name attributes `tireqty`, `oilqty`, and `sparkqty` are used to get the form inputs
- Output is generated in HTML

Bob's Auto Parts

Order Results

Order processed at 03:02, 16th September 2014

Your order is as follows:

Items ordered: 6
1 tires
2 bottles of oil
3 spark plugs
Subtotal: $132.00
Total including tax: $145.20

Regular customer.

```php
<?php
  // create short variable names
  $tireqty = $_POST['tireqty'];
  $oilqty = $_POST['oilqty'];
  $sparkqty = $_POST['sparkqty'];
  $find = $_POST['find'];
?>
<html>
<head>
  <title>Bob's Auto Parts - Order Results</title>
</head>
<body>
<h1>Bob's Auto Parts</h1>
<h2>Order Results</h2>
```

Bob's Auto Parts (4) – processorder.php

- The `date()` function is called in the script to produce the date and time content
- `echo` is a PHP function to output onto the web page
- PHP codes embedded in HTML tags will not be visible as PHP script, as it is interpreted and replaced it with output from the script in clean HTML

Order processed at 03:02, 16th September 2014

Your order is as follows:

Items ordered: 6
1 tires
2 bottles of oil
3 spark plugs
Subtotal: $132.00
Total including tax: $145.20

Regular customer.

```php
<?php

    echo "<p>Order processed at ".date('H:i, jS F Y')."</p>";

    echo "<p>Your order is as follows: </p>";

    $totalqty = 0;
    $totalqty = $tireqty + $oilqty + $sparkqty;
    echo "Items ordered: ".$totalqty."<br />";


    if ($totalqty == 0) {

      echo "You did not order anything on the previous page!<br />";

    } else {

      if ($tireqty > 0) {
        echo $tireqty." tires<br />";
      }

      if ($oilqty > 0) {
        echo $oilqty." bottles of oil<br />";
      }

      if ($sparkqty > 0) {
        echo $sparkqty." spark plugs<br />";
      }
    }
```

Bob's Auto Parts (5) – processorder.php

Bob's Auto Parts

Order Results

Order processed at 03:02, 16th September 2014

Your order is as follows:

Items ordered: 6
1 tires
2 bottles of oil
3 spark plugs
Subtotal: $132.00
Total including tax: $145.20

Regular customer.

```
$totalamount = 0.00;

define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);

$totalamount = $tireqty * TIREPRICE
             + $oilqty * OILPRICE
             + $sparkqty * SPARKPRICE;

echo "Subtotal: $".number_format($totalamount,2)."<br />";

$taxrate = 0.10;  // local sales tax is 10%
$totalamount = $totalamount * (1 + $taxrate);
echo "Total including tax: $".number_format($totalamount,2)."<br />";

if($find == "a") {
  echo "<p>Regular customer.</p>";
} elseif($find == "b") {
  echo "<p>Customer referred by TV advert.</p>";
} elseif($find == "c") {
  echo "<p>Customer referred by phone directory.</p>";
} elseif($find == "d") {
  echo "<p>Customer referred by word of mouth.</p>";
} else {
  echo "<p>We do not know how this customer found us.</p>";
}

?>
</body>
</html>
```

PHP Tags

quotation_marks.php

- Four styles of PHP tags.
  1. XML style

     ```
     <?php echo '<p>Order processed.</p>'; ?>
     ```

     – used here by default
  2. Short style

     ```
     <? echo '<p>Order processed.</p>'; ?>
     ```

     – need to enable short_open_tag setting
  3. Script style

     ```
     <script language='php'>
         echo '<p>Order processed. </p>'; </script>
     ```

  4. ASP style

     ```
     <% echo '<p>Order processed.</p>'; %>
     ```

     – need to enable asp_tags setting

Note: string quotation marks are in single (′) or double prime (″).

These are not the same as open and close quotation marks (‘ ’ “ ”).

Character codes:
Prime: 2032,
Double prime: 2033
Quotes: 2018, 2019,
Double quotes: 2016, 201D

Basic PHP Syntax

- PHP Statements are terminated by semicolon (;).

```
echo '<p>Order processed. </p>' ;
```

  - It prints (or echoes) the string passed to it to the browser.
- Whitespace
  - Spacing characters such as newline (carriage returns), spaces, and tabs are ignored by browser and PHP engine.
- Comments
  - PHP supports C, C++, and shell script-style comments.

```
/* Author: Bob Smith
This script processes the customer orders.
*/
```

  - Single-line comments, in the C++ style:

```
echo '<p>Order processed.</p>' ; // Start printing order
```

  - or in the shell script style:

```
echo '<p>Order processed. </p>'; # Start printing order
```

Adding Dynamic Content

date.php

- Use PHP's built-in date() function to tell date and time when an order was processed

```
<?php
  echo "<p>0rder processed at";
  echo date('H:i, jS F Y');
  echo "</p>";
?>
```

- Using the concatenation operator (.)

```
<?php
  echo "<p>0rder processed at". date('H:i, jS F Y'). "</p>";
?>
```

  - H is in 24-hour format, i is minutes, j is day of month, S represents the ordinal suffix (in this case th), F is the full name of the month, Y is the year.

Accessing Form Variables

- Variable name in PHP start with a dollar sign ($).
- Three ways of accessing form data via variables

```
$tireqty                          // short style
$_post['tireqty']                 //medium style
$HTTP_POST_VARS['tireqty']        //long style
```

- Short style is convenient but requires the register_globals configuration setting be turned on.
  - for security reasons, this setting is turned off by default.
- Medium style is the recommended approach
  - Medium style involves retrieving form variables from one of the arrays $_POST, $_GET, or $_REQUEST.
- Long style is most verbose and can be turned off by the register_long_arrays configuration directive.

Creating (Declaring) PHP Variables(1) orderform_get..html

- PHP script is loosely typed
  - A variable is created the moment you first assign a value to it

    e.g. $tireqty = $_POST['tireqty'];
- Variables are passed into script via the respective methods
  - Data in the tireqty box will be stored in $_POST['tireqty'] and $_GET['tireqty'] when the form is submitted via the POST and GET method, respectively
  - POST method hides data from URL, while GET displays it on URL
  - In either case, the data will also be available in $_REQUEST['tireqty']

Creating (Declaring) PHP Variables(2)

Processorder_get.php

- New variables are generally created in a block of code at the start of the processing script
- Example, this code creates variables via the POST method

```
<?php
    // create short variable names
    $tireqty = $_POST['tireqty'];
    $oilqty = $_POST['oilqty'];
    $sparkqty = $_POST['sparkqty'];
?>
```

Generating PHP Output

processorder.php

- PHP has a build-in function `echo` to output text onto a web page
  - the text can contain HTML markup
- To make the script in Bob's Auto Parts start doing something visible, add the following lines

```
echo '<p> Your order is as follows: </p>';
echo $tireqty. 'tires<br />';
echo $oilqty. 'bottles of oil<br />';
echo $sparkqty. 'spark plugs<br />';
```

Bob's Auto Parts

Order Results

Order processed at 03:02, 16th September 2014

Your order is as follows:

Items ordered: 6
1 tires
2 bottles of oil
3 spark plugs
Subtotal: $132.00
Total including tax: $145.20

Regular customer.

String Concatenation

string_interpolation.php

- String concatenation operator is a period (.)

```
echo $tireqty . 'tires<br />';
```

- This is equivalent to the above statement

```
echo "$tireqty . tires<br />";
```

  - Replacing a variable with its contents within a string, is known as interpolation.

- You cannot place variable name inside a single-quoted string.

```
echo '$tireqty. tires<br />';
```

  - This simply sends "$tireqty. tires
" to the browser
  - Single quoted strings are treated as true string literals

PHP's Data Types

- PHP supports the following basic data types:
  - Integer—Used for whole numbers
  - Float (also called a double)—Used for real numbers
  - String—Used for strings of characters
  - Boolean—Used for true or false values
  - Array—Used to store multiple data items
  - Object—Used for storing instances of classes
- Two special types are also available: NULL and resource.
  - Null variables are those that have not been given a value
  - Resource is external variable (e.g. a database connection) is not directly manipulatednd and is mostly returned by a function as a parameter to another.
- PHP variables are loosely typed or dynamically typed language.
  - a variable is determined by the value assigned to it.

Type Casting

typecasting.php

- Standard type casting

```
$totalqty = 0
$totalamount = (float) $totalqty;
```

  - After type casting, $totalamounrt stores the float value of $totalqty, while $totalqty remains of type integer.
- Variable variables

```
$varname = 'tireqty';
$$varname = 5;
```

  - This is exactly equivalent to

```
$tireqty = 5;
```

  - Variable variables enable you to change the name of a variable dynamically.

Declaring and Using Constants

constants.php

- Constants are defined using the define function

```
Define ('TIREPRICE', 100);
Define ('OILPRICE', 10);
Define ('SPARKPRICE', 4);
```

  - Constants are in uppercase simply for convention and easier reference
  - Constant need not have $ prefix

```
echo TIREPRICE;
```

Variable Scope

- The six basic scope rules in PHP are as follows:
  1. Built-in superglobal variables are visible everywhere within a script
  2. Constants, once declared, are always visible globally; that is, they can be used inside and outside function.
  3. Global variables declared (outside a function) in a script are visible throughout that script, but not inside functions.
  4. Variables inside functions that are declared as global refer to the global variables of the same name.
  5. Variables created inside functions and declared as static are invisible from the outside the function but keep their value in between one execution of the function and the next.
  6. Variables created inside functions are local to the function and cease to exist when the function terminates.

Superglobals

superglobal.php

- The arrays $_GET and $_POST and some other special variables have their own scope rules.
  - They are known as superglobals or autoglobals and can be seen everywhere, both inside and outside functions.
- The complete list of superglobals is as follows:
  - $GLOBALS - An array of all global variables (Like the global keyword, this allows you to access global variables inside a function—for example, as $GLOBALS['myvariable'].)
  - $_SERVER—An array of server environment variables
  - $_GET—An array of variables passed to the script via the GET method
  - $_POST—An array of variables passed to the script via the POST method
  - $_COOKIES—An array of cookie variables
  - $_FILES –An array of variables related to file uploads
  - $_ENV—An array of environment variables
  - $_REQUEST—An array of all user input including contents of $_GET, $_POST, and $_COOKIE (but not including $_FILES)
  - $_SESSION—An array of session variables

PHP's Operators(1)

- Arithmetic Operators

Operator	Name	Example
+	Addition	$a + $b
-	Subtraction	$a - $b
*	Multiplication	$a * $b
/	Division	$a / $b
%	modulus	$a % $b

- String Operator

```
$a = "Bob's";
$b =" Auto Parts";
$result =$a.$b;
```

  - The $result variable now contains the string "Bob's Auto Parts".

PHP's Operators(2)

reference.php

- Combined Assigned Operators

Operator	Use	Equivalent to
+=	$a += $b	$a = $a + $b
-=	$a -= $b	$a = $a - $b
*=	$a *= $b	$a = $a * $b
/=	$a /= $b	$a = $a / $b
%=	$a %=$b	$a = $a % $b
.=	$a .= $b	$a = $a . $b

- Reference Operator

```
$a =5;
$b =&$a;
$a =7; // $a and $b are now both 7
```

  - A reference is like an alias rather than like a pointer.

PHP's Operators(3)

- Comparison Operators

Operator	Name	Use
==	Equals	$a == $b
===	Identical	$a === $b
!=	Not equal	$a != $b
!==	Not identical	$a !== $b
<>	Not equal (comparison operator)	$a <>$b
<	Less than	$a < $b
>	Greater than (comparison operator)	$a > $b
<=	Less than or equal to	$a <= $b
>=	Greater than or equal to	$a >= $b

- Logical Operators

Operator	Name	Use	Result
!	NOT	!$b	Returns true if $b is false and vice versa.
&&	AND	$a && $b	Returns true if both $a and $b are true; otherwise false.
||	OR	$a || $b	Returns true if either $a or $b or both are true; otherwise false.
and	AND	$a and $b	Same as &&, but with lower precedence.
or	OR	$a or $b	Same as ||, but with lower precedence.
xor	XOR	$a xor $b	Returns true if either $a or $b is true, and false if they are both true or both false.

PHP's Operators(5)

dir.php

- The Execution Operator
  - Is a pair of backticks (` `).
  - PHP attempts to execute whatever is contained between the backticks as a command at the server's command line.

```
$out = `dir c`;
echo '<pre>'. $out.'</pre>';
```

  - It means obtain a directory listing and stores it in $out. It can then be echoed to the browser or dealt with in any other way.

- Other Operators
  - The comma operator (,) separates function arguments and other lists of items.
  - Two special operators, new and -> are used to instantiate a class and access class members.

Array Operators

array_op.php

- Array Operators

Operator	*Name*	*Use*	*Result*
+	*Union*	*$a + $b*	*Returns an array containing everything in $a and $b*
==	*Equality*	*$a == $b*	*Returns true if $a and $b have the same key and pairs*
===	*Identity*	*$a === $b*	*Returns true if $a and $b have the same key and value pairs the same order*
!=	*Inequality*	*$a != $b*	*Returns true if $a and $b are not equal*
<>	*Inequality*	*$a <> $b*	*Returns true if $a and $b are not equal*
!==	*Non-identity*	*$a !== $b*	*Returns true if $a and $b are not identical*

  - All array operators have equivalent operators that work on scalar variables.

Control Statements

- *if, else, elseif* Statements

```
e.g. if ($totalqty == 0)
        echo 'You did not order anything on the previous page!<br />';
```

  - A Code Block

```
if ($totalqty == 0) {
  echo '<p style="color:red">';
  echo 'You did not order anything on the previous page!';
  echo'</p>';
}
```

- *switch* Statements
  - allow the control condition to take more than two values.

```
e.g. switch ($find)  {
        case "a":
            echo "<p>Regular customer.<p/>";
            break;
        case "b":
        ……
```

Control Statements(2)

- Loops

```
while (control_expression)
        statement or compound;
        increment/decrement counter;

for (init; control; increment/decrement)
        statement or compound;

do
    statement or compound;
while (control_expression)
```

Numerically Indexed Arrays

- Initializing Numerically Indexed Arrays

  e.g. $products = array('Tires', 'Oil', 'Spark Plugs');

  - This code creates an array called $products containing the three values given: 'Tires', 'Oil' and 'Spark Plugs'.

- The following code creates the same $products array :

  $products [0] = 'Tires'

  $products [1] = 'Oil'

  $products [2] ='Spark Plugs'

Numerically Indexed Arrays(2)

array1.php

- Using *for* loop to access the Array

```
for ($i = 0; $i<3; $i++) {
  echo $products[$i].” ”;
}
```

- You can also use the *foreach* loop, specially designed for use with arrays.

```
foreach ($products as $current) {
  echo $current.” ”;
}
```

  - This code assigns the value of each element in $product[] to $current and the array pointer is moved by one till the end of the array.

Associative Arrays with Named Keys

- Initializing an Associative Array
  - The following code creates an array with product names as keys and prices as values:

e.g. $prices = array('Tires' =>100, 'Oil' =>10, 'Spark Plugs'=>4);

- The following code creates the same $prices array with one element and adds two more:

```
$prices = array('Tires' =>100 );
$prices ['Oil'] =10;
$prices['Spark Plugs'] =4;
```

- The array is created when the first element is added to it:

```
$prices['Tires'] =100;
$prices ['Oil'] =10;
$prices['Spark Plugs'] =4;
```

Associative Arrays with Named Keys(2)

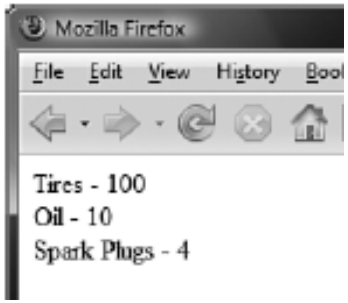
array2.php

- Accessing the Array Elements using keys
  - Keys are used to access an array when the indices are not numbers hence cannot use a simple counter in a for loop
  - Can use foreach loop and each() construct
- Using *foreach* loop to access *keys* in an array

```
e.g.  foreach ($prices as $key => $value) {
            echo $key." - ".$value."<br />";
      }
```

- The following code lists the contents of the $prices array using *each()* construct:

```
while ($element = each($prices)) {
    echo $element['key'];
    echo " - ";
    echo $element['value'];
    echo "<br />";  }
```



Multidimensional Arrays

- Each location in an array can hold another array, creating a two-dimensional array.

Code	Description	Price
TIR	Tires	100
OIL	Oil	10
SPK	Spark Plugs	4

product

product attribute

- Using PHP, you would write the following code to set up the data in the array:

```
$products = array( array('TIR', 'Tires', 100),
                   array('OIL', 'Oil',10),
                   array('SPK', 'Spark Plugs',4));
```

- To display the contents of this array, you could manually access each element in order like this:

```
echo'|'.$products[0][0].'|'.$products[0][1].'|'.$products[0][2].'|<br/> ';
echo'|'.$products[1][0].'|'.$products[1][1].'|'.$products[1][2].'|<br/>';
echo'|'.$products[2][0].'|'.$products[2][1].'|'.$products[2][2].'|<br/>';
```

Multidimensional Arrays(2)

multiarray.php

- Alternatively, you could place a for loop inside another for loop to achieve the same result:

```
for ($row = 0; $row < 3; $row++) {
  for ($column = 0; $column < 3; $column++){
    echo '|'.$products[$row][$column];
  }
  echo '|<br/>';
}
```

Code	Description	Price
TIR	Tires	100
OIL	Oil	10
SPK	Spark Plugs	4

product

product attribute

- Both version produces the same output in the browser, but code in second method is shorter:

```
|TIR|Tires|100|
|OIL|Oil|10|
|SPK|Spark Plugs|4|
```

Sorting Arrays

- Using *sort()*
  - The following code showing the sort() function results in the array being sorted into ascending alphabetical order:

    e.g. $products = array(‘Tires’, ‘Oil’, ‘Spark Plugs’);

    sort($products);
  - The array elements will now appear in the order Oil, Spark Plugs, Tires.
  - You can sort values by numerical order, too.

    e.g. $prices = array(100, 10, 4)

    sort($prices);
  - The prices will now appear in the order 4, 10, 100.

Using Functions in PHP

- Calling Functions

```
function_name();
```

- Passing data and name of variable to functions

```
function_name('parameters');
function_name(2);
function_name(7.993);
function_name($variables);
```

  - In the last line, $variable might be any type of PHP variable, including an array or object.

- Here is the declaration of a trivial function:

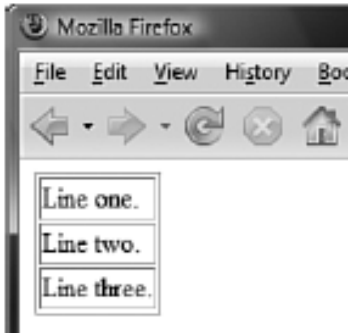
```
function my_function() {
  echo 'My function was called';
}
```

Using Parameters

function.php

- A sample function that takes a one-dimensional array and displays it as a table

```
function create_table($data) {
        echo "<table border=\"1\">";
        reset($data);   // Remember this points to the beginning
        $value=current($data);   // current element of array
        while ($value) {
                echo "<tr><td>".$value."</td></tr>\n";
         $value = next($data);
        }
         echo"</table>";
}
```



- Figure shows output of calling the create_table() function:

```
$my_array = array('Line one. ',' Line two. ',' Line three.');
create_table($my_array);
```

Passing by Reference vs. Passing by Value

passbyval.php

- Passing by value

```
function increment($value, $amount = 1) {
    $value = $value + $amount;
}
```

  - This code is of no use. The output from the following test code will be 10:

```
$value = 10;
increment ($value);
echo $value;
```

  - The contents of $value have not changed because of the scope rules.

Passing by Reference vs. Passing by Value(2)

passbyref.php

- Passing by reference
  - You can modify the preceding increment() example to have one parameter passed by reference and it will work correctly:

```
function increment(&$value, $amount = 1) {
        $value = $value +$amount;
}
```

  - You now have a working function and are free to name the variable to increment anything you like. The following test code now echoes 10 before the call to increment() and 11 after:

```
$a = 10;
echo $a.'<br/>';
increment ($a);
echo $a. '<br/>';
```

Returning Values from Functions

return.php

- You can write the larger() function as follows:

```
function larger ($x, $y) {
    if ((!isset($x) || (!isset($y))) {
        return false;
    } else if ($x>$y) {
        return $x;
    } else {
        return $y;
    } }
```

  - The built-in function isset() tells if a variable is created and given a value.

- The code

```
$a = 1; $b= 2.5; $c = 1.9;
echo larger($a, $b).'<br/>;
echo larger($c, $a).'<br/>;
echo larger($d, $a).'<br/>;
```

  - produces this output because $d does not exist and false is not visible:

    2.5
    1.9