69/B Monwara Plaza (Level-5)
East Panthapath
Dhaka-1205
Cell: 01713 432003

Lecture Sheet on “PHP+MySQL”
Day-5: (PHP-2)

Objectives

- Handle HTML forms using PHP scripts
- Validate form data
- Manage arrays using various types of loops
- Work with magic quotes

Topics

- Variable Scope
- PHP SuperGlobals
- Variable Types
- Control Structures
- Arrays
- Using Forms

Variable Scope

The scope of a variable is the context within which it is defined. Basically you can not access a variable which is defined in different scope. The script below will not produce any output because the function Test() declares no $a variable. The echo statement looks for a local version of the $a variable, and it has not been assigned a value within this scope. Depending on error_reporting value in php.ini the script below will print nothing or issue an error message.

```
<?php
$a = 1; // $a is a global variable
function Test()
{
  echo $a; // try to print $a, but $a is not defined here
}
Test();
?>
```

If you want your global variable (variable defined outside functions) to be available in function scope you need to use the $global keyword. The code example below shows how to use the $global keyword.

```
<?php
$a = 1; // $a is defined in global scope ...
$b = 2; // $b too
function Sum()
{
  global $a, $b; // now $a and $b are available in Sum()
  $b = $a + $b;
}
Sum();
echo $b;
?>
```

PHP Superglobals

Superglobals are variables that available anywhere in the program code. They are :

$_SERVER

Variables set by the web server or otherwise directly related to the execution environment of the current script. One useful variable is $_SERVER['REMOTE_ADDR'] which you can use to know you website visitor's IP address

Your computer IP is:

```
<?php
echo $_SERVER['REMOTE_ADDR'];
?>
```

$_GET

Variables provided to the script via HTTP GET. You can supply GET variables into a PHP script by appending the script's URL like this : http://edunews24.com/tuition.php?page=16 or set the a form method as method="get"

```
<?php
echo "My name is {$_GET['name']} <br>";
echo "My friend's name is {$_GET['friend']}";
?>
```

Note that I put $_GET['name'] and $_GET['friend'] in curly brackets. It's necessary to use these curly brackets when you're trying to place the value of an array into a string.

You can also split the string like this :
echo "My name is " . {$_GET['name']} . "<br>"; but it is easier to put the curly brackets.

$_POST

Variables provided to the script via HTTP POST. These comes from a form which set method="post"

$_COOKIE

Variables provided to the script via HTTP cookies.

$_FILES

Variables provided to the script via HTTP post file uploads. You can see the example in "Uploading files to MySql Database"

$_ENV

Variables provided to the script via the environment.

$_REQUEST

Variables provided to the script via the GET, POST, and COOKIE input mechanisms, and which therefore cannot be trusted. It's use the appropriate $_POST or $_GET from your script instead of using $_REQUEST so you will always know that a variable comes from POST or GET.

$GLOBALS

Contains a reference to every variable which is currently available within the global scope of the script.
You usually won't need the last three superglobals in your script.

Variable Types

PHP supports eight primitive types.
Four scalar types:
boolean : expresses truth value, TRUE or FALSE. Any non zero values and non empty string are also counted as TRUE.
integer : round numbers (-5, 0, 123, 555, ...)
float : floating-point number or 'double' (0.9283838, 23.0, ...)
string : "Hello World", 'PHP and MySQL, etc

Two compound types: 1. array 2. object
And finally two special types:
1. resource (one example is the return value of mysql_connect() function)
2. NULL
In PHP an array can have numeric key, associative key or both. The value of an array can be of any type. To create an array use the array() language construct like this.

```
<?php
$numbers = array(1, 2, 3, 4, 5, 6);
$age = array("mom" => 45, "pop" => 50, "bro" => 25);
$mixed = array("hello" => "World", 2 => "It's two";

echo "numbers[4] = {$numbers[4]} <br>";
echo "My mom's age is {$age['mom']} <br>";
echo "mixed['hello'] = {$mixed['hello']} <br>";
echo "mixed[2] = {$mixed[2'}";
?>
```

When working with arrays there is one function I often used. The print_r() function. Given an array this function will print the values in a format that shows keys and elements

```
<?php
$myarray = array(1, 2, 3, 4, 5);
$myarray[5] = array("Hi", "Hello", "Konnichiwa", "Apa Kabar", “como estas”, “comment ca va”);
echo '<pre>';
print_r($myarray);
echo '</pre>';
?>
```

Don't forget to print the preformatting tag <pre> and </pre> before and after calling print_r(). If you don't use them then you'll have to view the page source to see a result in correct format.

Control Structures

The next examples will show you how to use control structures in PHP. I won't go through all just the ones that I will use in the code examples in this site. The control structures are

- ☑ if
- ☑ else
- ☑ while
- ☑ for

If-Else

The if statement evaluates the truth value of it's argument. If the argument evaluates as TRUE the code following the if statement will be executed. And if the argument evaluate as FALSE and there is an else statement then the code following the else statement will be executed.

```
<?php
$ip    = $_SERVER['REMOTE_ADDR'];
$agent = $_SERVER['HTTP_USER_AGENT'];
if(strpos($agent, 'Opera') !== false)
   $agent = 'Opera';
else if(strpos($agent, "MSIE") !== false)
   $agent = 'Internet Explorer';

echo "Your computer IP is $ip and you are using $agent";
?>
```

The strpos() function returns the numeric position of the first occurrence of it's second argument ('Opera') in the first argument ($agent). If the string 'Opera' is found inside $agent, the function returns the position of the string. Otherwise, it returns FALSE. When you're using Internet Explorer 6.0 on Windows XP the value of $_SERVER['HTTP_USER_AGENT'] would be something like: *Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)* and if you're using Opera the value the value may look like this : *Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5; Windows NT 5.1) Opera 7.0 [en]* So if i you use Opera the **strpos()** function will return value would be 61. Since 61 !== false then the first if statement will be evaluated as true and the value of $agent will be set to the string 'Opera'.
Note that I used the !== to specify inequality instead of != The reason for this is because if the string is found in position 0 then the zero will be treated as FALSE, which is not the behavior that I want here.

While Loop

The while() statement is used to execute a piece of code repeatedly as long as the while expresssion evaluates as true. For example the code below will print the number one to nine.

```
<?php
$number = 1;
while ($number < 10)
{
  echo $number . '<br>';
  $number += 1;
}
?>
```

You see that I make the code $number += 1; as bold. I did it simply to remind that even an experienced programmer can sometime forget that a loop will happily continue to run forever as long as the loop expression (in this case $number < 10) evaluates as true. So when you're creating a loop please make sure you already put the code to make sure the loop will end in timely manner.

Break

The break statement is used to stop the execution of a loop. As an example the while loop below will stop when $number equals to 6.

```
<?php
$number = 1;
while ($number < 10)
{
  echo $number . '<br>';
  if ($number == 6)
  {
    break;
  }

  $number += 1;
}
?>
```

For

The for loop syntax in PHP is similar to C. For example to print 1 to 10 the for loop is like this

```
<?php
for ($i = 1; $i <= 10; $i++) {
   echo $i . '<br>';
}
?>
```

A more interesting function is to print this number in a table with alternating colors. Here is the code

```
<table width="200" border="0" cellspacing="1" cellpadding="2">
<tr>
<td bgcolor="#CCCCFF">Alternating row colors</td>
</tr>
<?php
for ($i = 1; $i <= 10; $i++) {
   if ($i % 2) {
      $color = '#FFFFCC';
   } else {
      $color = '#CCCCCC';
   }
?>
<tr>
<td bgcolor="<?php echo $color; ?>"><?php echo $i; ?></td>
</tr>
<?php
}
?>
</table>
```

This code display different row colors depending on the value of $i. If $i is not divisible by two it prints yellow otherwise it prints gray colored rows.

```
<?php
// fill an array with all items from a directory
$handle = opendir('.');
while (false !== ($file = readdir($handle))) {
    $files[] = $file;
}
closedir($handle);  echo '<pre>';  print_r($files); echo '</pre>'
?>
```

This a very simple function on how to sort a multidimentional array in php

```
<?php function sort_multi_array($mult_array , $field , $sort_type="ASC_NUM")

{

$code = "";
switch($sort_type) {

case ASC_NUM:
$code .= 'return strcmp($a["'.$field.'"], $b["'.$field.'"]);';
break;
case DESC_NUM:
$code .= 'return (-1*strcmp($a["'.$field.'"], $b["'.$field.'"]));';
break;
}
```

```
$compare = create_function('$a, $b', $code);
usort($mult_array, $compare);
return $mult_array;
}

//How to use :
$data[] = array('volume' => 67, 'date' => '2008-12-30 21:20:56', 'name' => 'abc');
$data[] = array('volume' => 86, 'date' => '2007-12-25 21:20:56', 'name' => 'def' );
$data[] = array('volume' => 85, 'date' => '2007-12-27 21:20:56', 'name' => 'ghi');
$data[] = array('volume' => 05, 'date' => '2008-12-02 21:20:56', 'name' => 'jkl');
$data[] = array('volume' => 86, 'date' => '1975-12-05 21:20:56', 'name' => 'mno');
$data[] = array('volume' => 67, 'date' => '2007-12-30 20:20:56', 'name' => 'rto');

$data = sort_multi_array($data ,'date' , 'DESC_NUM');
var_dump($data);
echo "<pre>"; print_r($data); echo "</pre>";
?>
/*
it will sort the array according to ‘date’
array(6) {
[0]=>
array(3) {
["volume"]=>
int(67)
["date"]=>
string(19) “2008-12-30 21:20:56″
["name"]=>
string(3) “abc”
}
[1]=>
array(3) {
["volume"]=>
int(5)
["date"]=>
string(19) “2008-12-02 21:20:56″
["name"]=>
string(3) “jkl”
}
[2]=>
array(3) {
["volume"]=>
int(67)
["date"]=>
string(19) “2007-12-30 20:20:56″
["name"]=>
string(3) “rto”
}
[3]=>
array(3) {
["volume"]=>
int(85)
["date"]=>
string(19) “2007-12-27 21:20:56″
["name"]=>
string(3) “ghi”
}
[4]=>
```

```
array(3) {
["volume"]=>
int(86)
["date"]=>
string(19) “2007-12-25 21:20:56″
["name"]=>
string(3) “def”
}
[5]=>
array(3) {
["volume"]=>
int(86)
["date"]=>
string(19) “1975-12-05 21:20:56″
["name"]=>
string(3) “mno”
}
} */
```

Using Form

Using forms in a web based application is very common. Most forms are used to gather information like in a signup form, survey / polling, guestbook, etc. A form can have the method set as post or get. When using a form with method="post" you can use $_POST to access the form values. And when the form is using method="get" you can use $_GET to access the values. The $_REQUEST superglobal can be used to to access form values with method="post" and method="get" but it is recommended to use $_POST or $_GET instead so you will know from what method did the values come from.

Here is an example of HTML form :

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
Name : <input name="username" type="text"><br>
Password : <input name="password" type="password"><br>
<input name="send" type="submit" value="Send!">
</form>
```

The form above use $_SERVER['PHP_SELF'] as the action value. It is not required for the form to perform correctly but it's considered good programming practice to use it.

Below is the PHP code used to access form values :

```
<?php
if(isset($_POST['send']))
{
  echo "Accessing username using POST : " .      $_POST['username'] . "<br>";
  echo "Accessing username using REQUEST : " .      $_REQUEST['username'] . "<br>";

  $password = $_POST['password'];
  echo "Password is $password";
}
?>
```

The if statement is used to check if the send variable is set. If it is set then the form must have been submitted. The script then print the value of username using $_POST and $_REQUEST

Using Array As Form Values

Take a look at the code example below. The form have five input with the same name, language[]. Using the same input name is common for checkboxes or radio buttons.

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
Select the programming languages you can use<br>
```

```
<input name="language[]" type="checkbox" value="C++">
C++<br>
<input name="language[]" type="checkbox" value="Java">
Java<br>
<input name="language[]" type="checkbox" value="PHP">
PHP<br>
<input name="language[]" type="checkbox" value="ASP">
ASP<br>
<input name="send" type="submit" id="send" value="Send!">
</form>
```

The PHP code below print the value of language after the form is submitted. Go ahead and try the example. Try checking and unchecking the options to see the effect.

```
<?php
if(isset($_POST['language']))
{
  $language = $_POST['language'];
  $n        = count($language);
  $i        = 0;

  echo "The languages you selected are \r\n" .
       "<ol>";
  while ($i < $n)
  {
    echo "<li>{$language[$i]}</li> \r\n";
    $i++;
  }
  echo "</ol>";
}
?>
```

From the above code you will notice that $language is an array. This is because in the form code language is repeated several times. When you specify the same input name in a form, PHP will treat it as an array.

There is one important issue you should know when using forms, that is form validation. The code above does not check whether the input is correct such as checking if the user is entering any value into the textbox or is the user selecting any checkboxes. This is actually a bad practice because you should never put a web form without any validation.

Form validation with PHP & JavaScript

If you want to validate a form without submitting it, you can use two methods, one is pure client side JavaScript and the other being a mix of JavaScript and server-side scripting which can be PHP,ASP.JSP,Perl or anything for that matter.First I will give an example of pure JavaScript validation The HTML:

```
<html>
 JavaSrcipt Form Validation<head>
 <script language="JavaScript" type="text/javascript">
 function validateForm(oForm)
 {
        //oForm refers to the form which you want to validate
        oForm.onsubmit = function() // attach the function to onsubmit event
        {
                var regex = /^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/;
                if(oForm.elements['email'].value.length<1)
                {
                        alert("You cannot leave the email field empty");
                        return false;
```

```
			}
			else if(!regex.test(oForm.elements['email'].value))
			{
				alert("Invalid email address format");
				return false;
			}
			return true;
		}
}
</script>
</head>
<body>
<form name="email_check" method="post">
Enter email : <input type="text" name="email" />

<input type="submit" value="Check & Submit" />
</form>
<script language="JavaScript">
new validateForm(document.forms['email_check']);
</script>
</body>
</html>
```