|  |  |  |
| --- | --- | --- |
| **BESE 16-A/B** | **CS340 Web Technologies** | **Dr Naima Iltaf** |

**Lab 8 – Advanced PHP**

**PHP Arrays:**

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

In PHP, there are three kind of arrays:

* Numeric array - An array with a numeric index
* Associative array - An array where each ID key is associated with a value
* Multidimensional array - An array containing one or more arrays

**Numeric Arrays**

A numeric array stores each array element with a numeric index.

There are two methods to create a numeric array.

In the following example the index are automatically assigned (the index starts at 0):

$cars=array ("Saab","Volvo","BMW","Toyota");

In the following example we assign the index manually:

<?php

$cars[0]="Saab";  
$cars[1]="Volvo";  
$cars[2]="BMW";  
$cars[3]="Toyota";

echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";

?>

**Associative Arrays**

An associative array, each ID key is associated with a value.

With associative arrays we can use the values as keys and assign values to them.

$ages = array ("Peter"=>32, "Quagmire"=>30, "Joe"=>34);

This example is the same as example 1, but shows a different way of creating the array:

The ID keys can be used in a script:

<?php  
$ages['Peter'] = "32";  
$ages['Quagmire'] = "30";  
$ages['Joe'] = "34";  
  
echo "Peter is " . $ages['Peter'] . " years old.";  
?>

**Multidimensional Arrays**

In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

<?php

$products = array(

'paper' => array(

'copier' => "Copier & Multipurpose",

'inkjet' => "Inkjet Printer",

'laser' => "Laser Printer",

'photo' => "Photographic Paper"),

'pens' => array(

'ball' => "Ball Point",

'hilite' => "Highlighters",

'marker' => "Markers"),

'misc' => array(

'tape' => "Sticky Tape",

'glue' => "Adhesives",

'clips' => "Paperclips") );

echo "<pre>";

foreach ($products as $section => $items)

foreach ($items as $key => $value)

echo "$section:\t$key\t($value)<br>"; echo "</pre>";

?>

**PHP Forms:**

The PHP $\_GET and $\_POST variables are used to retrieve information from forms, like user input.

**Form Handling:**

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts. The Method attribute is used to tell the browser how the form information should be sent. The two most popular methods you can use are GET and POST.

**Get Example:**

<html>

<head>

<title>A BASIC HTML FORM</title>

</head>

<body>

<FORM NAME ="form1" METHOD =" GET">

<INPUT TYPE = "TEXT" VALUE ="username">

<INPUT TYPE = "Submit" Name = "Submit1" VALUE = "Login">

</FORM>

</body>

</html>

The resultant URL is

http://localhost/basicForm.php?Submit1=Login

This is a consequence of using the GET method. The data from the form ends up in the address bar. Submit1 was the NAME of the button, and Login was the VALUE of the button (the text on the button). This is what is being returned by the GET method. You use the GET method when the data you want returned is not crucial information that needs protecting.

**Post Example:**

<html>

<?php

if(isset($\_POST['submit1']))

{

echo ($\_POST["name"]." is ");

echo ($\_POST["age"]. " years old.");

}

?>

<body>

<form action="index.php" method="post">

Name: <input type="text" name="name" />

Age: <input type="text" name="age" />

<input type="submit" name="submit1"/>

</form>

</body>

</html>

When the user fills in this form and click on the submit button, the form data is sent to the "index.php" which in this case is the same in which php script is written.

**Form Validation**

User input should be validated whenever possible. Client side validation is faster, and reduces server load. However, any site that gets enough traffic to worry about server resources, may also need to worry about site security. You should always use server side validation if the form accesses a database.

PHP provides a number of functions to help us validate form input:

* isset() is variable null?
* is\_numeric() is variable a numeric type?
* is\_string() is variable a string?

<?php

// multiply a number by two

// get parameter “num” from url string

$num = $\_GET[“num”];

// is $num a number? if not, throw an error

if (!is\_numeric($num)) {

echo “Error!”;

}

else {

echo $num\*2;

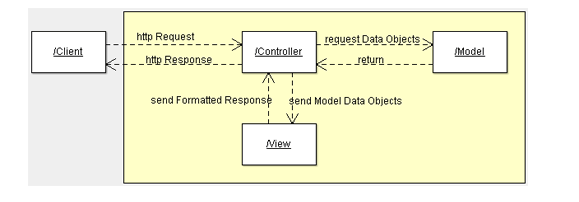
}

?>

**MVC Introduction:**

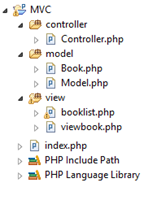
The Model-View-Control (MVC) pattern, originally formulated in the late 1970s, is a software architecture pattern built on the basis of keeping the presentation of data separate from the methods that interact with the data. The MVC pattern separates an application in 3 modules: Model, View and Controller:

* The model is responsible to manage the data; it stores and retrieves entities used by an application, usually from a database, and contains the logic implemented by the application.
* The view (presentation) is responsible to display the data provided by the model in a specific format. It has a similar usage with the template modules present in some popular web applications, like WordPress, joomla, …
* The controller handles the model and view layers to work together. The controller receives a request from the client, invokes the model to perform the requested operations and sends the data to the View. The view formats the data to be presented to the user, in a web application as an html output.



**A simple example of MVC in PHP:**

1. Create the following hierarchy in Zend studio project.



1. Write the code below in index.php file.

<?php

include\_once("controller/Controller.php");

$controller = new Controller();

$controller->invoke();

?>

The controller is the first thing which takes a request, parses it, initializes and invokes the model and takes the model response and sends it to the presentation layer. The application entry point is index.php. The index php file delegates all the requests to the controller

1. Write the code below in Controller.php file.

<?php

include\_once("model/Model.php");

class Controller {

public $model;

public function \_\_construct()

{

$this->model = new Model();

}

public function invoke()

{

if (!isset($\_GET['book']))

{

// no special book is requested, we'll show a list of

// all available books

$books = $this->model->getBookList();

include 'view/booklist.php';

}

else

{

// show the requested book

$book = $this->model->getBook($\_GET['book']);

include 'view/viewbook.php';

}

}

}

?>

Controller class has only one function and the constructor. The constructor instantiates a model class and when a request is done, the controller decides which data is required from the model. Then it calls the model class to retrieve the data. After that it calls the corresponding view script passing the data coming from the model.

1. The model layer consists of two classes, a Book class which is visible to the view layer and a Model class. Write the code below in Model.php file.

<?php

include\_once("model/Book.php");

class Model

{

public function getBookList()

{

return array

(

"Jungle Book" => new Book("Jungle Book", "R.

Kipling", "A classic book."),

"Moonwalker" => new Book("Moonwalker", "J.

Walker", ""),

"PHP for Dummies" => new Book("PHP for Dummies",

"Some Smart Guy", "")

);

}

public function getBook($title)

{

$allBooks = $this->getBookList();

return $allBooks[$title];

}

}

?>

Now write the below code in Book.php file.

<?php

class Book {

public $title;

public $author;

public $description;

public function \_\_construct($title, $author, $description)

{

$this->title = $title;

$this->author = $author;

$this->description = $description;

}

}

?>

1. In our example the view contains only 2 files one for displaying one book and the other one for displaying a list of books. Write the following code in viewbook.php file.

<html>

<head></head>

<body>

<?php

echo 'Title:' . $book->title . '<br/>';

echo 'Author:' . $book->author . '<br/>';

echo 'Description:' . $book->description . '<br/>';

?>

</body>

</html>

Write the following code in booklist.php file.

<html>

<head></head>

<body>

<table>

<tr><td>Title</td><td>Author</td><td>Description</td></tr>

<?php

foreach ($books as $title => $book)

{

echo '<tr><td><a href="index.php?book='.$book->title.'">'.$book->title.'</a></td><td>'.$book->author.'</td><td>'.$book->description.'</td></tr>';

}

?>

</table>

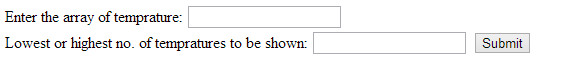
</body>

</html>

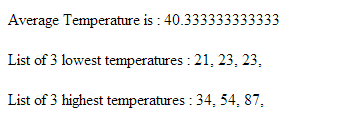
Now run the above code and observe the output.

**Lab Tasks**

1. **Write a PHP script to calculate and display average temperature, seven lowest and highest temperatures. (3)**



**Expected output:**



**Solution:**

<html>

<head>

<title></title>

</head>

<body>

<form action=*"index.php"* method=*"post"*>

<label for=*"arr"* >Enter Array of Temperature: </label>

<input type=*"text"* id=*"arr"* name = *"Temps"*/> <br />

<label for=*"num"*>Lowest or Highest Temperature to be shown: </label>

<input type=*"text"* id=*"num"* name =*"Qty"* />

<input type=*"submit"* value=*"Submit"* name = *"submit1"*/>

</form>

</body>

</html>

<?php

**if** (**isset**($\_POST['submit1']))

{

$str = $\_POST['Temps'];

$num = $\_POST['Qty'];

$ar = explode(",",$str);

sort($ar);

$av = 0;

**for** ($i=0;$i<count($ar);$i++)

{

$av = $av + $ar[$i];

}

**echo** "Average Temperature is: ".$av/count($ar);

**if** ($num != "")

{

**print** "<br /> List of $num Lowest Temperture is: ";

**for** ($i=0;$i<$num;$i++)

{

**echo** $ar[$i];

**echo** " ";

}

rsort($ar);

**print** "<br /> List of $num Highest Temperture is: ";

**for** ($i=0;$i<$num;$i++)

{

**echo** $ar[$i];

**echo** " ";

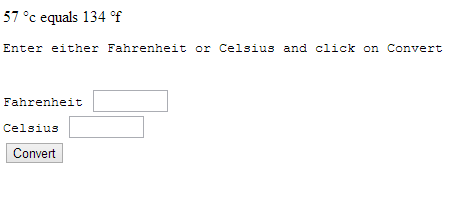
}

}

}

?>

1. **Write a temperature conversion program which takes either Fahrenheit or Celsius and converts into the other one. (2)**



**Solution:**

<html>

<body>

<form action=*"index.php"* name=*"form1"* method=*"post"*>

Enter either Fahrenheight or Celsius temperature:<br>

Fahrenheight:<input type=*"text"* name=*"fah"*><br>

Celsius:<input type=*"text"* name=*"cel"*><br>

<input type=*"submit"* name=*"submit1"* value=*"Convert"* />

</form>

</body>

<?php

**if**(**isset**($\_POST['submit1']))

{

$f = $\_POST["fah"];

$c = $\_POST["cel"];

**if**($f!=**null** && $c!=**null**)

{

**echo** "<br> Enter value in one text box not both";

}

**else if**($c!=**null**)

{

$fah =((9/5)\*$c)+32;

**echo** "<br>".$c." C equals ".$fah." F";

}

**else if**($f!=**null**)

{

$cel =(($f-32)\*(5/9));

**echo** "<br>".$f." F equals ".$cel." C";

}

**else if**($f!=**null** && $c!=**null**)

{

**echo** "<br> Enter value in either text box..";

}

**else**

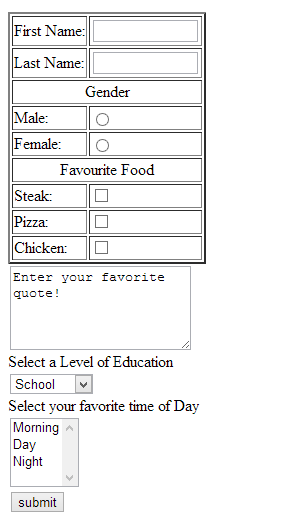
**echo** "<br> Enter value in either text box..";

}

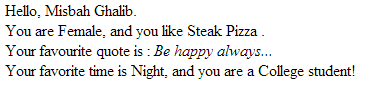
?>

</html>

1. **Write a php script to create the following form. (5)**



**Expected output:**



**Solution:**

<html>

<head>

<title></title>

</head>

<body>

<form action=*"index.php"* method=*"post"*>

<table border=*"1"*>

<tbody>

<tr>

<td>First Name: </td>

<td><input type=*"text"* name=*"fName"* /></td>

</tr>

<tr>

<td>Last Name: </td>

<td><input type=*"text"* name=*"lName"* /></td>

</tr>

<tr>

<td align=*"center"* colspan=*"2"*>Gender</td>

</tr>

<tr>

<td>Male:</td>

<td><input type=*"radio"* name=*"gender"* value=*"Male"*/></td>

</tr>

<tr>

<td>Female:</td>

<td><input type=*"radio"* name=*"gender"* value=*"Female"*/></td>

</tr>

<tr>

<td align=*"center"* colspan=*"2"*>Favorite Food</td>

</tr>

<tr>

<td>Steak: </td>

<td><input type=*"checkbox"* name=*"Food[]"* value=*"Steak"*/></td>

</tr>

<tr>

<td>Pizza: </td>

<td><input type=*"checkbox"* name=*"Food[]"* value=*"Pizza"*/></td>

</tr>

<tr>

<td>Chicken: </td>

<td><input type=*"checkbox"* name=*"Food[]"* value=*"Chicken"*/></td>

</tr>

</tbody>

</table>

<textarea name=*"tArea"*>

Enter Your Favorite Quote!!

</textarea><br />

<label for=*"edu"*>Select Your Level of Education</label> <br />

<select id=*"edu"* name=*"education"*>

<option>School</option>

<option>High School</option>

<option>University</option>

</select><br />

<label for=*"TOD"*>Select Your Favorite Time of Day</label> <br />

<select size=*"4"* id=*"TOD"* name=*"TimeOf"*>

<option>Morning</option>

<option>Day</option>

<option>Night</option>

</select><br />

<input type=*"submit"* name=*"Submit1"* />

</form>

</body>

</html>

<?php

**if** (**isset**($\_POST['Submit1']))

{

$firstN = $\_POST['fName'];

$LastN = $\_POST['lName'];

$Education = $\_POST['education'];

$T = $\_POST['TimeOf'];

$Quote = $\_POST['tArea'];

$Gender = $\_POST['gender'];

$food = $\_POST['Food'];

$foodcat = implode(" ", $food);

**print** "Hello, $firstN $LastN <br />";

**print** "You are $Gender & you like $foodcat .<br />";

**print** "Your Favorite Quote is: <i>$Quote</i><br />";

**print** "Your Favorite Time is $T, and you are a $Education Student <br />";

}

?>