

Lab Manual for Introduction to Database

Lab 03: PHP Introduction and Forms Handling

Table of Contents

1. Introduction	105
2. Activity Time boxing	105
3. Objective of the experiment	105
4. Concept Map	105
4.1 PHP Variables	106
4.2 PHP Form Handling	107
4.3 PHP Data Types	107
4.4. PHP Operators	108
4.4 PHP Conditional Statements	108
4.5 Loops	109
5. Procedure& Tools	112
5.1 Tools	112
5.2 Setting-up and Setting up XAMPP (MySQL, Apache)	112
5.3 Walkthrough Task [Expected time = 30mins]	112
6. Evaluation Criteria	114
7. Practice Tasks	114
7.1 Task 01	114
7.2 Task 02	114
7.3 Task 03	114
7.4 Outcome	115
7.5 Testing	115
8. Evaluation Task (Unseen) [Expected time = 60 mins]	115
9. Evaluation Criteria	115
10. Further Reading	115

Lab 03: PHP Introduction and Forms Handling

1. Introduction

What is PHP (Personal Home Page)?

PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. It is a server side scripting language. The PHP code is ran on the webserver and then the output is returned to the user through a web browser.

The main reasons for its popularity are:

- It is open-source and free!
- Easy to use.

It has a very simple syntax unlike other languages such as Perl or C. Rather than writing lots of code to create a webpage, we create HTML documents and embed simple PHP codes into them. It has multi-platform support. It supports all major operating systems. Moreover, the syntax is consistent among different platforms. You can create PHP codes in Windows and easily switch to UNIX. All PHP commands are enclosed within special start and end tags:

```
<?php
...PHP code...
?>
```

For instance, if the PHP code is embedded into an HTML document, the PHP interpreter reads and executes only the PHP code enclosed within the start and end tags.

2. Activity Time boxing

Table 1: Activity Time Boxing

Task No.	Activity Name	Activity time	Total Time
6.2	Setting-up and Setting Up XAMPP (MySQL, Apache)	20mins	20mins
6.3	Walkthrough Tasks	30mins	60mins
7	Practice tasks	20 to 30mins for each task	50mins
8	Evaluation Task	40mins for all assigned task	40mins

3. Objective of the experiment

- To Introduce Hypertext Markup Language Forms
- To learn about the PHP language and its usage with HTML

4. Concept Map

Now first we study about the syntax of a php followed by the interaction of form with php.

4.1 PHP Variables

A variable in PHP can be used to store both numeric and nonnumeric data.

- 1) Every variable has a name, which is preceded by a dollar (\$) symbol.
- 2) Variable names are case sensitive and they must begin with a letter or underscore character.

We can replace the PHP code above with:

```
<?php
//define variable
$answer = 'A: Chameleon';
//print output
Echo "<h2><i>$answer</i></h2>";
?>
```

This will produce the same result as before.

- 1) To assign a value to a variable, use the equality (=) symbol (ex: \$answer = 'A: Chameleon' ;).
- 2) To use a variable value in your script, call the variable by its name. PHP will substitute its value when the code is executed (ex: Echo "<h2><i>\$answer</i></h2>" ;).

4.2 PHP Data Types

There are four basic data types in PHP. PHP can automatically determine the variable type by the context in which it is being used.

Data Type	Description	Example
Boolean	Specifies a true or false value.	\$auth = true;
Integer	Integers like -98, 2000.	\$age = 28;
Floating-point	Fractional numbers such as 12.8 or 3.149391	\$temp = 76;
String	Sequence of characters. May be enclosed in either double quotes or single quotes.	\$name = 'Ismail';

- 1) The data type of a variable can be retrieved by the function `gettype($variable_name)`.
- 2) If a string variable is enclosed within double quotes, the variables are automatically replaced by their values.

```
<?php
$identity = 'James Bond';// this would contain the string "My name is James Bond"
$sentence = "My name is $identity";// this would contain the string "My name is $identity"
$sentence = 'My name is $identity';?>
```

4.3 PHP Operators

There are over 15 operators in PHP that can be used to perform operations on the variables:

Operator	What It Does
=	Assignment
+	Addition
-	Subtraction
*	Multiplication
/	Division, returns quotient
%	Division, returns modulus
.	String concatenation
==	Equal to
===	Equal to and of the same type
!=	Not equal to or not of the same type
<>	Not equal to
<, <=, >, >=	Less than, Less than or equal to etc.
&&	Logical AND
	Logical OR
Xor	Logical XOR
!	Logical NOT

1) PHP has its own set of rules about which operators have precedence over others (Operators on the same line have the same level of precedence):

- “!”
- “*”, “/”, “%”
- “+”, “-”, “.”
- “<”, “<=”, “>”, “>=”
- “==”, “!=”, “===”, “!==”
- “&&”
- “||”

4.4 PHP Conditional Statements

A conditional statement enables you to test whether a specific condition is true or false, and to perform different actions on the basis of the test result. We will use the if() statement to create conditional statements:

```
<?php
if (conditional test)
{
    do this;
}
else
{
    do this;
}
?>
```

1) If the conditional expression after “if” evaluates to true, all PHP code within the following curly brackets is executed. If not, the code coming after the “else” is executed.

- 2) The “else” part of the above code can be removed. In that case, if the conditional expression is false, the code within the curly braces is skipped and the lines following the “if” construct are executed.

```
<?php
if ($temp >= 100)
{
    echo 'Very hot!';
}
else
{
    echo 'Within tolerable limits';
}
?>
```

PHP also provides you with a way of handling multiple possibilities:

```
<?php
if ($country == 'UK')
{$capital = 'London';}
elseif ($country == 'US')
{$capital = 'Washington';}
elseif ($country == 'FR')
{$capital = 'Paris';}
else
{
    $capital = 'unknown';
}
?>
```

4.5 Loops

A loop is a control structure that enables you to repeat the same set of commands over and over again. The actual number of repetitions may be dependent on a number you specify, or on the fulfillment of a certain condition.

The simplest loop in PHP is the *while* loop. With this loop type, so long as the conditional expression specified evaluates to true, the loop will continue to execute. When the condition is false, the loop will be broken and the statements following it will be executed.

```
<?php
$num = 11; // define number and limits for multiplication tables
$upperLimit = 10;
$lowerLimit = 1;
while ($lowerLimit <= $upperLimit) // loop and multiply to create table
{
    echo "$num x $lowerLimit = " . ($num*$lowerLimit);
    $lowerLimit++;
}
?>
```

This script uses a while loop to create a multiplication table for the given table. It starts with “11 x 1 = 11” and continues until “11 x 10 = 110”.

- 1) “\$lowerLimit++,” does the same job as “\$lowerLimit = \$lowerLimit + 1;”

If the loop condition evaluates as false on the first iteration of the loop, the loop will never be

executed. However, sometimes you might need to execute a set of commands at least once. Regardless of how the conditional expression evaluates. For such situations, PHP offers the do-while loop. The construction of the do-while() loop is such that the statements within the loop are executed first, and the condition to be tested is checked after.

The structure of the do-while loop is as follows:

```
<?php
do
{
    do this
} while (condition is true)
?>
```

Let's now revise the previous PHP script so that it runs at least once, regardless of how the conditional expression evaluates the first time.

```
<?php
// define number and limits for multiplication tables
$num = 11;
$upperLimit = 10;
$lowerLimit = 12;// loop and multiply to create table
do
{
    echo "$num x $lowerLimit =" . ($num*$lowerLimit);
    $lowerLimit++;
} while ($lowerLimit<= $upperLimit)
?>
```

Both while and do-while loops continue to iterate for so long as the specified conditional expression remains true. But there often arises a need to execute a certain set of statements a fixed number of times. We use the for() loop for this purpose.

```
<?php
for (initialize counter; conditional test; update counter)
{
    do this
}
?>
```

The for loop uses a counter that is initialized to a numeric value, and keeps track of the number of times the loop is executed. Before each execution of the loop, a conditional statement is tested. If it evaluates to true, the loop will execute once more and the counter will be incremented by 1 (or more). If it evaluates to false, the loop will be broken and the lines following it will be executed instead.

To see how this loop can be used, create the following script, which lists all the numbers between 2 and 100:

```
<?php
for ($x = 2; $x <=100; $x++)
{
    echo "$x";
}
?>
```

4.6 PHP | Arrays

Arrays in PHP is a type of data structure that allows us to store multiple elements of similar data type under a single variable thereby saving us the effort of creating a different variable for every data. The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key. Suppose we want to store five names and print them accordingly. This can be easily done by the use of five different string variables. But if instead of five, the number rises to hundred, then it would be really difficult for the user or developer to create so much different variables. Here array comes into play and helps us to store every element within a single variable and also allows easy access using an index or a key. An array is created using an `array()` function in PHP.

There are basically three types of arrays in PHP:

- **Indexed or Numeric Arrays:** An array with a numeric index where values are stored linearly.
- **Associative Arrays:** An array with a string index where instead of linear storage, each value can be assigned a specific key.
- **Multidimensional Arrays:** An array which contains single or multiple array within it and can be accessed via multiple indices.

```
<?php
```

```
// One way to create an indexed array
$name_one= array("Zack", "Anthony", "Ram", "Salim", "Raghav");
```

```
// Accessing the elements directly
echo"Accessing the 1st array elements directly:\n";
echo$name_one[2], "\n";
echo$name_one[0], "\n";
echo$name_one[4], "\n";
```

```
// Second way to create an indexed array
$name_two[0] = "ZACK";
$name_two[1] = "ANTHONY";
$name_two[2] = "RAM";
$name_two[3] = "SALIM";
$name_two[4] = "RAGHAV";
```

```
// Accessing the elements directly
echo"Accessing the 2nd array elements directly:\n";
echo$name_two[2], "\n";
echo$name_two[0], "\n";
echo$name_two[4], "\n";
```

```
//Also Access like that in loop
for($n= 0; $n< $round; $n++){
    echo$name_two[$n], "\n";
}
```

4.7 Form Interaction with PHP:

To see how PHP works with HTML, create the code below using notepad.


```

<html>
<head><basefont face= "Arial "></head>
<body>
<h2>Q: This creature can change color to blend in with its surroundings. What is its name?</h2>

<?php
//print output
echo "<h2><i>A: Chameleon </i></h2>";
?>

</body>
</html>

```

A PHP script consists of one or more statements, with each statement ending in a semicolon (ex: echo ‘<h2><i>A: Chameleon </i></h2>’;).

For greater readability, you should add comments to your code. Comments can be written after “//” (ex: //print output).

Save this script as question.php and browse to it. View the source code of the web page you have created by clicking “Source” in the “View” tab in Internet Explorer. You will see:

```

<html>
<head><basefont face="Arial"></head>
<body>
<h2>Q: This creature can change color to blend in with its surroundings. What is its name?</h2>
<h2><i>A: Chameleon </i></h2>
</body>
</html>

```

When the code is executed, PHP converted the code inside the “<?php” and “?>” tags to regular HTML code! Everything outside these tags is ignored by PHP and returned as is.

4.8 Post Data from Forms using PHP

You can add interactivity to your web site using FORMS. A form enables your users to submit inputs to your web site. Create the HTML document below to get user input (save it as getinput.html). Then we will manipulate this input using a PHP script.

```

<html>
<head></head>
<body>
<form action= "message.php" method= "post">
Enter your message: <input type= "text" name= "msg" size= "30">
<input type="submit" value="Send">
</form>
</body>
</html>

```

- 1) “action” attribute specifies the name of the script that will process the information entered into the form. Here, the input entered into the form will be sent to message.php.
- 2) The value of the input entered is stored in the variable named msg.

Now create the script that will process the input and save it as message.php:

```
<?php
```

```
// retrieve form data in a variable
$input = $_POST['msg'];
// print it
Echo "You said: <i>$input</i>";
?>
```

3) To access the value of a form variable, use its name inside `$_POST` (ex: `$_POST['msg']`).

Now, run the `getinput.html` and enter some data into the form (“hello”) and submit it. `Message.php` should read it and display it back to you (“You said: hello”).

Now, in a walkthrough section we will design an application of Login authentication using Form and PHP.

5. Procedure& Tools

5.1 Tools

In this section tools installation and setup is defined.

- Desktop Computer
- Microsoft Windows 10 operating system
- Internet Browser(Internet Explorer, Mozilla Firefox or Google Chrome etc)
- Notepad

5.2 Setting-up and Setting up XAMPP (MySQL, Apache) [Expected time = 5mins]

Refer to Lab 1 sec 6.2.

For creating PHP file move to `C:\xampp\htdocs` and perform tasks of PHP using notepad++

5.3 Walkthrough Task [Expected time = 30mins]

In this task you can learn how to create a simple web application using PHP.

- 1) Move to <Drive>\xampp\htdocs
- 2) Create an Folder WalkthroughTaskon <Drive>\xampp\htdocs
- 3) Download **bootstrap.min.css**, and **ie10-viewpoint-bug-workaround.css** css file from <http://getbootstrap.com/css/> site and copy paste on WalkthroughTask\css
- 4) Open any text editor for example notepad
- 5) Write the following code in it

```

1 <html>
2 <head></head>
3 <body>
4 <form action="login_authentication.php">
5   <div >
6     <label type="text">Email address</label>
7     <input type="email" name="txtemail" placeholder="Email">
8   </div>
9   <div class="form-group">
10    <label type="text" name="txtpassword">Password</label>
11    <input type="password" placeholder="Password">
12  </div>
13  <button type="submit">Sign in</button>
14 </form>
15 </form>
16 </body>
17 </html>

```

Figure 1: Login Form Using Bootstrap

- 6) Save your document as index.php.
- 7) Open another text editor and write code

```

<?php
$email = $_POST['txtemail'];
$password = $_POST['txtpassword'];
if ($email == 'ali@gmail.com' and $password == '123') {
    session_start();
    $_SESSION['sid'] = session_id();
    header("location:LoginCheck.php");
} else {
    echo "<script>alert('Email or password is incorrect!')</script>";
}
?>

```

Figure 2: Login Authentication

- 8) Save your document as loginauthentication.php
- 9) Open another text editor and write code

```

<?php

session_start();
$sessionno = session_id();
if ($_SESSION['sid'] == session_id()) {
    echo "Welcome to you<br>";
    echo $sessionno."<br>";
    echo "<a href='logout.php'>Logout</a>";
} else {
    header("location:index.php");
}

?>

```

Figure 3: LoginCheck

10) Save your document as LoginCheck.php

11) Open another text editor and write code

```
<?php

echo "Logged out successfully";
session_start();
//session is a way to store information
//(in variables) to be used across multiple pages.
session_destroy();

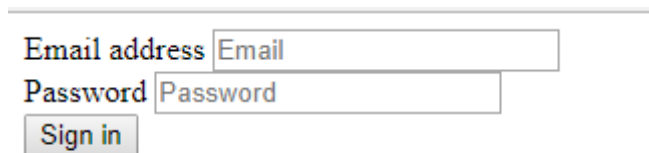
?>
```

Figure 4: Logout

12) Save your document as Logout.php

13) Open googleMozilla Firefox and write URL : <http://localhost:<Your's XAMPApache port>/WalkthroughTask/index.php>

14) Output Like



The image shows a simple web form. It has two text input fields. The first field is labeled 'Email address' and contains the placeholder text 'Email'. The second field is labeled 'Password' and contains the placeholder text 'Password'. Below these fields is a button labeled 'Sign in'.

Figure 5: Output

6. Evaluation Criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 2: Evaluation of the Lab

Sr. No	Task No.	Task Description	Grade
1	7	Practice task	30
2	-	Unseen Task	20

7. Practice Tasks

7.1 Task 01

Create a website that asks a simple question and retrieves the viewer's answer from an array of Q/As. Display messages on the screen depending on the answer:

- If the answer is correct, display “Congratulations!”
- Otherwise, display a message like “Your answer was -----. The correct answer is -----.

7.2 Task 02

Use the project created above in 7.1 and Put the Questions and answers into a session variable.

7.3 Task 03

Show all Question Answers from Session in tabular form, also add edit and delete button in next column for edit and delete the Question

7.4 Outcome

After completing this lab, students will be able to perform some functionality using PHP and also understand the concept of web application development using PHP.

7.5 Testing

TestCasesforPracticeTask-1 and Task-2

SampleInputs	SampleOutputs
Question) 1+1 <input type="text" value="2"/> <input type="button" value="Submit Answer"/>	answer is correct

Table2:ConfirmationofpracticetasksT1,T2,T3,T4

PracticeTask	Confirmatio	Comments
T1		
T2		
T3		

8. Evaluation Task (Unseen)

[Expected time = 60 mins]

The lab instructor will give you unseen task depending upon the progress of the class.

9. Evaluation Criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 3: Evaluation of the Lab

Sr. No.	TaskNo	Description	Marks
1	6	ProceduresandTools	0
2	7.1	Practicetask1withTesting	5
3	7.2	Practicetask2withTesting	5

4	7.3	Practicetask3withTesting	5
5	8	EvaluationTasks(Unseen)	80
6		GoodProgrammingPractices	5
TotalMarks			100

10. Further Reading

HTML W3 Tutorial

- <http://www.w3schools.com/html/>

HTML code Tutorial

- <http://www.htmlcodetutorial.com/>