BIT 2nd Year
Semester 3
IT 3505

Web Application Development II

Server Side Web Development (PHP & MySQL) – Part 3

Duration: 30 hours

Instructional Objectives

- Install PHP in a windows environment
- Install PHP in Linux environment
- Explain basic features of PHP
- Articulate MVC architecture
- Differentiate available PHP frameworks
- Explain MVC
- Use web services with PHP
- Develop a web application with PHP

Sub Topics

1.1 Introduction to PHP (Ref 01 Pg:271-278)
1.2. Configuring the environment (Ref 01 Pg : 76 - 85)
1.3. PHP Syntax and Semantics
- 1.3.1. Variables (Ref 01 Pg:281-287)
- 1.3.2. Constants (Ref 01 pg:287 - 296)
- 1.3.3. Conditional Statements (Ref 01 pg:320-335)
- 1.3.4. Loops (Ref 01 Pg:335-346)
- 1.3.5. Functions (Ref 01 Pg: 346-357)

1.4. Arrays and data processing with arrays (Ref 01 Pg: 296-307)
1.5. Form processing with PHP (Ref 02)
1.6. Session control and Cookies (Ref 01 Pg:437-446)
1.7. File system management (Ref 01 Pg: 366-389)
1.8. Email sending using PHP (Ref 03)
1.9. Object Orientation with PHP (Ref 01 pg :397-423)
1.10. Working with MySQL database (Ref 01 PG:515-528)
1.11. Introduction to PHP frameworks (Ref 5)
1.12. Fundamentals of MVC (Ref 6)
1.13. How to call web service using PHP (Ref 01 pg:541-553)

Compound Data Types

Arrays

- One of the compound data types provided by PHP is arrays.
- In PHP an array is an ordered collection of data items where each item in the collection is associated with a key. A PHP array can be considered as an ordered map. Individual data items of an array can be of any type.

Construction of an array

- An array is constructed by using the language construct *array(array_elements)*
- The *array_elements* comprises of a comma-separated *index,value* pairs, where each pair is represented as *index => value.*

Syntax :

```
array(
  index_1 =>value_1,
  index_2 => value_2,
 ……………
 index_n => value_n,
)
```

The comma after the last array element is optional and can be omitted

Construction of an array…….

- The *index* of an element can be of type *integer or string*.
- When the index is omitted, an integer index is automatically generated, starting at highest integer index + 1 or 0 if no item is given an integer index.

example :

```
$a = array(
        1=> "First Item",
        "item2" => "Second
Item",
        5 => "Third item",
        "Forth item"
        );
```

PHP would automatically generate the index 6 for the last item in the array.

Accessing an element in an array.

- An element of an array can be accessed by using the syntax array_variable[index]

example :

```
$a = array(
        1=> "First Item",
        "item2" => "Second Item",
        5 => "Third item",
        "Forth item"
        );
```

Each element in the above array can be accessed as below;

```
$a[1]
$a["item2"]
$a[5]
$a[6]
```

Changing the value of an array element.

- The following syntax can be used to change the value of an array element.

$array_variable[index] = new_value;

example:

```
$a = array(
        1=> "First Item",
        "item2" => "Second Item",
        5 => "Third item",
        "Forth item"
        );

$a[1] = “abc”;
$a[“item2”] = 25;
```

Adding a new element to an array.

- The following syntax can be used to add a new value to an array.

$array_variable[new_index] = new_value;

The *new_index* should not exist as an index in the array.

Appending elements to the end of an array.

array_push command can be used to add one or more elements to the end of the array.

Syntax :

```
  array_push(array_variable, value1,value2,......)
```

Example :

```
$a = array("Nimal","Saman");
 array_push($a,"Kamal","Waruna");
```

Array of arrays

- Elements of an array can also be arrays.

example :

```
$a = array(
        "males" => array("a" => "Nimal","b" => "Amara","c"
=>"Kamal"),
        "females" => array("a" => "Kumari", "b" => "Nirmala", "c" =>
"Kamala"),
         "fees" => array (2500,1500,500)
        );
```

Accessing array elements example :

```
  echo  $a["females"]["b"]
```

Looping through array elements

foreach looping construct can be used to loop through the elements of an array.

Syntax :

foreach (array_expression as $value) statement

Or

foreach (array_expression as $key => $value) statement

Looping through array elements - Example

```
<?php
 $a = array(
        1=> "First Item",
        "item2" => "Second Item",
        5 => "Third item",
        "Forth item"
        );

 foreach ($a as $key => $value){
   echo $key," - ",$value,"\n";
 }
?>
```

Combining Script Files

Including Files

- PHP allows content in separate files to be inserted into a single file just before the execution of the file. This is a very useful when the same content (PHP, HTML, text) to be included on multiple pages of a website.
- File inclusion can be done by using two commands.
  - include ‘filename’
  - require ‘filename’
- The above commands can be included in the locations of a file where the content of the specified files are to be inserted.

include and require commands

- The semantics of the two commands are identical except how PHP handles the error resulting in not finding the specified file specified to be included.
  - Command “require” will produce a fatal error and stop the execution of the script.
  - Command “include” will only produce a warning error and the execution of the script will continue.

include and require commands – Example 1

The following script is in a file named *example1.php*

```
<?php
include "example2.php";
echo "Executing the file example1.php\n";
?>
```

The following script is in a file named *example2.php* in the same folder

```
<?php
echo "Executing the file example2.php\n";
?>
```

include and require commands – Example 2

The following script is in a file named *example1.php*

```
<?php
echo "Executing the file example1.php\n";
require "example2.php";
?>
```

The following script is in a file named *example2.php* in the same folder

```
<?php
echo "This is the footer to be included in every web page \n";
?>
```

Including PHP Scripts in HTML Pages

How to include PHP scripts in HTML Pages.

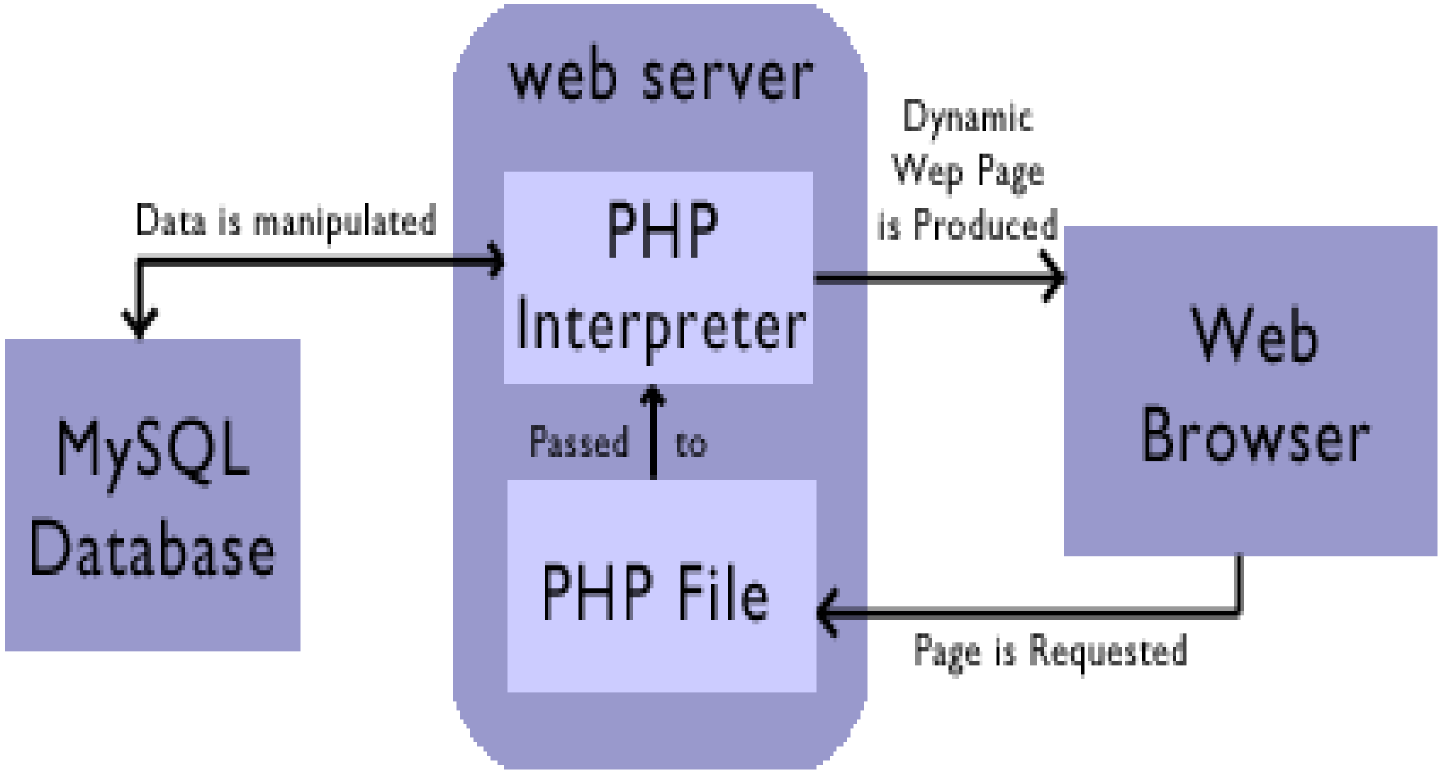
- Any number of PHP scripts can be included at different places in a HTML page.
- When including a PHP script, it should be enclosed in “<?php” and “?>” tag pair to designate what is in between these tag pairs as a PHP script.
- A HTML file with PHP scripts should be saved with the extension “.php” to indicate that the file needs to be interpreted by PHP.
- To execute PHP codes embedded in a HTML page, the page must be loaded through a web server which has configured to execute embedded PHP scripts – Example Apache web server.

Example

```
<!DOCTYPE html>

<html>
    <head>
        <title>Including PHP in web pages</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>
    <body>
        <div><?php echo "This is the first script"; ?></div>
         <div>
         <?php echo "This is the second script<br />";
         echo "With many lines<br />";
         ?>
         </div>
    </body>
</html>
```

How web servers execute PHP scripts



Processing Form data

Forms in HTML

- HTML forms is a powerful feature that enables an application on a web server to interact with users.
- HTML forms allow users to send data to the web site.
- An HTML Form is typically made of a collection of widgets such as text fields, buttons, checkboxes, radio buttons or select boxes.
- The “post” or “get” HTTP methods can be used to send data to the server.

action and method attributes

- Two important attributes of the "form" element are "action" and "method".
  - action : specifies the URL of the web resource designated to receive the data collected from the form element.
  - method : specifies which HTTP method (get or post) should be used to send data to the receiving URL.
  - If the receiving URL is a PHP program, then depending on the ***method*** used in the HTML form either the PHP superglobal $_GET or $_POST can be used to access form data at the servers end.

Example 1- Form with text inputs

```
<html>
<body>
<form action=“example.php" method="post">
Name: <input type="text" name="name"><br>
<input type="submit">
</form>
</body>
</html>
```

The **name** attribute specifies the key value of the $_POST global array element from which the value of this input item can be retrieved

The content of the example.php script is given below

```
<!DOCTYPE html>

<html>
    <body>
        <div> Hello <?php echo $_POST["name"]?></div>
    </body>
</html>
```

Example 2- Form with check boxes

```
<html>
<body>

<form action=“example.php" method=“post">
Do you have an email?
<input type=“checkbox" name=“emailOption“ value =
“Yes”><br>
<input type="submit">
</form>

</body>
</html>
```

When the user checked the checkbox, the value “Yes” is send to the server as the value of the attribute “emailOption”.

Example 2- check boxes ….

```
 <html>
   <body>
      <div>
         <?php if($_POST["emailOption"]
== "Yes"){
            echo "Option is checked";
         } else {
            echo "Option is not-checked";
         }
         ?>
      </div>
   </body>
</html>
```

The data send to the server can be accessed by using the $_POST global array element with the key value "emailOption"

Example 3- Form with a check box group

```
<html>
<body>
<form action=“example.php" method=“post">
Which fruits do you like?
<input type=“checkbox" name=“fruits[]“ value = “Apples”>Apples<br>
<input type=“checkbox" name=“fruits[]“ value = “Oranges”>Oranges<br>
<input type=“checkbox" name=“fruits[]“ value = “Grapes”>Grapes<br>
<input type="submit">
</form>
</body>
</html>
```

Note that the checkboxes have the same name “fruits” and each name ends in [ ].

- The same name indicates that these checkboxes are all related and forms a group.
- [ ] indicates that the selected values will be provided to PHP script as an array. This means That the $_POST[‘fruitsr'] is an array not a single string.

Example 3- Form with a check box group ....

```
<html>
    <body>
        <div>
            <?php
            $fruits = $_POST["fruits"];
            if(!empty($fruits)){
                echo "You like ";
                for($i=0; $i < count($fruits);$i++){
                    echo "<br>". $fruits[$i];
                }
            } else {
                echo "You do not like any fruits";
            }
            ?>
        </div>
    </body>
</html>
```

Example 4- Form with a Radio button

```
 <html>
   <body>
      <form action="example.php" method="post">
         Please specify your sex :<br>
         <input type="radio" name="sex" value =
"male">male<br>
         <input type="radio" name="sex" value =
"female">female<br>
         <input type="submit">
      </form>

   </body>
</html>
```

Note that all radio buttons should have the same value for the attribute "name".

Example 4- Form with a Radio button ….

```
<html>
  <body>
     <div>
        <?php
         echo "you are a ". $_POST["sex"];
        ?>
     </div>
  </body>
</html>
```

Scope of Variables

Scope of Variables

- The scope of a variable is the range of statements in which the variable can be used(referenced).
- The scope rules of a language specify how a particular occurrence of a name should be associated with a variable.

PHP Scope Rules

1) Scope of a variable used inside a used defined function is limited to the body of the function by default.

Example 1:

```
<?php
function test(){
  $a = 1;
}

echo $a;
?>
```

In the above script $a variable is local to the function ***test***. Thus, it is unknown outside the function.

PHP Scope Rules

1) Scope of a variable used inside a used defined function is limited to the body of the function by default.

Example 2:

```
<?php
$a = 1;
function test(){
  echo $a;
}
test()
?>
```

In the above script an error is raised as the variable $a defined outside the function is not visible inside the function.

PHP Scope Rules …..

2) Scope of a variable in a script spans over the files included in a script.

Example :

```
<?php
 $a = 1;
 include 'example1.php';

?>
```

The variable $a is visible in the included file “example1.php”.

The file example1.php contains the following code

```
<?php
echo $a;
?>
```

global keyword

The keyword **global** can be used inside functions to refer to variable define outside the function(in the global context).

Syntax:

global $a,$b,……;

```
<?php
$a = 1;
function test(){
  global $a;
  echo $a;
}

test();
?>
```

In the above script the variable $a inside the function binds to the variable $a defined outside the function.

Superglobals

Superglobals are built-in variables that are always available in all scopes. This means the superglobal variables are provided by PHP and are visible everywhere in a script. The PHP superglobals are given in the following list.

- $GLOBALS
- $_SERVER
- $_POST
- $_FILES
- $_SESSION

$_ENV

$_REQUEST

$_GET

$_COOKIE

$GLOBALS variable.

- The $GLOBALS variable defines an associative array with the name of the global variable being the key and the contents of that variable being the value of the array element. This variable can be used to access global variables from anywhere in the PHP script.

```
<?php
$a = 1;
function test(){
  echo
$GLOBALS['a'];
}

test();
?>
```

Static Variables

- A variable inside a function can be defined as static to retain its value between function calls. This means that a static variable does not lose its value when program execution leaves the scope.

```
<?php
function test(){
    static $a = 0;
    $a = $a + 1;
    echo $a;
}

for($i=0; $i <10; $i++){
   test();
}
?>
```

Static Variables ………

- A static variable can be initialize with a constant value. However, PHP does not allow a static variable to be initialized with the final value of an expression.

```
<?php
function test(){
    static $a = 2 + 3; //
incorrect

}
```

Cookies

Cookies

- A cookie is a file with small amount of data that a website embeds on the user's computer through a web browser. This cookie is send back to the website by the browser every time the user is accessing the same website by using the same browser.
- The browsers can either enable or disable cookies.
- In PHP data stored in cookies can be accessed by using the global array $_COOKIE

Setting a cookie – setcookie()

- Syntax:
- setcookie($name [, $value [, $expire]]);
- Semantic:
- Sends a cookie to the browser.
- $name - The name of the cookie
- $value – The value of the cookie
- $expire – The time (in seconds) the cookie expires. If this value is set to 0 or omitted, the cookie will expire when the browser closes.
- The function returns the Boolean value TRUE on success or FALSE on failure.

Cookies must be sent before producing any output in a script. This requires setcookie() function to be used prior to any output, including <html> and <head> tags as well as any whitespace.

Checking whether cookies are enabled or not.

- <?php
- setcookie("name","saman",time()+3600);
- if(count($_COOKIE) > 0){
-     echo "Cookies are enabled<br>";
- } else {
-   echo "Cookies are disabled <br>";
- }

Modifying the value of a cookie

- To modify the value of a cookie call the same function setcookie() with the new value.

```
<?php
setcookie("name",“Kamal",tim
e()+3600);
?>
```

Deleting a cookie.

- To delete a cookie execute the same setcookie() function with an expiration date in the past.

```
<?php
setcookie("name",“Kamal",tim
e()-3600);
?>
```

File Handling

Typical operations on files

- Opening a file
- Adding data/ Accessing data
- Closing the file

Opening a File

Syntax:

fopen ($filename , $mode [,$use_include_path = false [, $context]])

fopen() binds the resource named as $filename, to a stream.

File opening modes
w – write
r – reading
a – appending

fopen returns a file pointer resource on success or FALSE on failure

- If a file named “mydata.txt” exists then the content of the file is deleted.
- If there is no file with the name “mydata.txt” then a new file with the name “mydata.txt” is created.

Writing data to a file

```
<?php
$f = fopen("data.txt","w");
fwrite($f,"My name is
saman\n");
fwrite($f,"My age is 90");
fclose($f);
?>
```

fwrite returns the number of bytes written to the file or FALSE on failure.

Syntax of fwrite :
fwrite ($handle , $string [, $length])
fwrite() writes the content of **$string** to the file stream pointed to by **$handle**. If the optional length argument is given, writing will stop after **$length** number of bytes is written or the end of string is reached, whichever comes first.

Appending data to a file

```
<?php
$f = fopen("data.txt",“a");
fwrite($f,"My name is Sunil\n");
fclose($f);
?>
```

Reading data from a file – fgets()

Syntax:

file ($filename)

Semantics:

- Reads the entire file $filename into an array.
- The command returns
  - The file in an array. Each element of the array corresponds to a line in the file or
  - FALSE if an error occurs.

```
<?php
$lines= file(“data.txt");
foreach($lines as
$line_no => $line){
 echo
$line_no,$line,"<br>";
}
?>
```

Reading data from a file – file()

Syntax:

fgets ($handle [,$length])

Semantics:

- Reads a line from the file pointed to by the file pointer $handle.
- The command returns
  - A line of symbols (including the end of line marker) from the file as a string when the $length parameter is not specified or
  - A string of up to length - 1 bytes from the file when $length parameter is specified or
  - The Boolean value FALSE when there is no more data to read in the file or
  - The Boolean value FALSE if an error occurred while reading the file.

Reading data from a file – fscanf()

Syntax:

fscanf($handle, $format)

Semantics:

- Reads a line of the file pointed to by the file pointer $handle according to the format specified by the string $format.
- The command returns
  - the values parsed as an array.

```
<?php
$f = fopen("data.txt","r");
while ($line =
fscanf($f,"%s\t%d\n")){
 echo $line[0],"-
",$line[1],"<br>";
}
?>
```

Reading data from a file - Example

```
<?php
$f =
fopen("data.txt","r");
while (! feof($f)){
  $line = fgets($f);
  echo $line, "<br>";
}
fclose($f);
?>
```

Checking the existence of a file/directory

Command : file_exists()

Syntax : file_exists($filename)

Semantics :

Checks the existence of a file or directory.

It returns the Boolean value TRUE when the file/Directory exists, otherwise it returns FALSE.

```
<?php
if(
!file_exists("data.txt")
){
 echo "File does not
exists";
 exit;
}
echo "File Exists";
?>
```