UNIT IV: Creating Dynamic Web Sites

4.1 Including Multiple Files

PHP provides the facility to combine one or more files together to create a single page.

It allows dividing the scripts and web page into two or more parts. So the commonly used processes or files can be designed separately & can be combined with any webpage.

Generally titles of the website, menus & some photos or some common information/contents are displayed on every page of website.

So this can be separately designed & included into every page instead of designing it on every page of website.

PHP Provides four functions for using external files/pages in webpage or including one page into another, these are:

1. include()
2. include_once()
3. require()
4. require_once().

To use them, your PHP script would have a line like

include ('filename.php');

include_once ('filename.php');

require ('/path/to/filename.html');

An important consideration with included files is that PHP treats the included code as HTML.

The included file may be .php, .html, or .txt. Also note that you can use either absolute or relative paths to the included file.

An absolute path is a path where a file is starting from the root directory of the computer. Such paths are always correct, For example, a PHP script can include a file using

include ('C:/php/includes/file.php');

A relative path uses the referencing (parent) file as the starting point. To move up one folder, use two periods together.

To move into a folder, use its name followed by a slash. So assuming the current script is in the wwwww/ex1 folder and you want to include something in www/ex2, the code would be:

include('../ex2/file.php');

A relative path will remain accurate, even if moved to another server, as long as the files keep their current relationship.

include() and require() functions are exactly the same when working properly but behave differently when they fail.

If an **include()** function doesn't work (means it cannot include the file for some reason), a warning will be printed to the Web browser, but the script will continue to run.

If **require()** fails, an error is printed and the script execution is stopped.

include_once() & require_once() functions guarantees that the file is included only once regardless of how many times a script may attempt to include it.

4.2 Handling HTML Forms with PHP Redux

PHP Redux is one of the facilities of PHP, used for handling the html form on the same PHP page.

Means form designing using HTML & code for handling or processing that form can be written on the same page using PHP Redux facility.

Generally we have to design two separate files: one for creating html form for getting input & other to read the input for handling or processing that input.

For designing and handling the form on same page, a conditional must check which action (display or handle) should be taken:

It is shown as follows:

```
if (/* form has been submitted */) {
// Handle it.
} else {
// Display it.
}
```

To determine if the form has been submitted, check if a $_POST variable is set

For example, create a submit button on form with a name “submitted” as follows:

<input type=”submit” name=”submitted” value=”Submit Form”> Then the condition testing for form submission would be

```
if (isset($_POST[‘submitted’])) {
// Handle it.
} else {
// Display it.
}
```

If you want a page to handle a form and then display it again (e.g., to add a record to a database and then give an option to add another), then don’t write the else clause:

```
if (isset($_POST[‘submitted’])) {
// Handle it.
}
```

Using that code, a script will handle a form if it has been submitted and display the form every time the page is loaded.

```
<form action="<?php echo $_SERVER[PHP_SELF]; ?>" method ="post">
<h1 align="center"> Cost Caculation </h1>
<table border="1" align="center" width="75%">
<tr> <td> Enter Distance in KM </td><td> <input type ="text" name="t1"
></td></tr>
<tr> <td> Enter Rate of Petrol </td><td> <input type ="text" name="t2"></td></tr>
<tr> <td> Enter Average of Vichicle </td><td> <input type ="text" name="t3"></td></tr>
<tr> <td> </td><td> <input type ="Submit" name="b1" value="Calculate"></td></tr>
</table>
</form>

<?php
$s=$_POST["b1"];
if(isset($s))
{
$dist=$_POST["t1"];
$rate=$_POST["t2"];
$avg=$_POST["t3"];
$p=$dist/$avg;
$cost=$p*$rate;
echo "<h1 align=center> Cost of Journey=" .$cost;
}
?>
```

4.3 Making Sticky Forms

A sticky form is simply a standard HTML form that remembers the information or data entered in the form.

This is a particularly nice feature for users, especially if you are requiring them to resubmit a form after filling it out incorrectly in the first time

To remember or to retain what's entered in a text box, use its value attribute:

```
<input type="text" name="city" size="20" value="Aurangabad">
```

This will set value Aurangabad in textbox previously.

To retain or remember that value which was entered before on the textbox, print the appropriate variable in front of value attributes of textbox as follows:

<input type=”text” name=”city” size=”20” value=”<?php echo $city; ?>” >

To retain or remember the value of a textarea, place the value between the textarea tags as follows:

<textarea name=”comments” ><?php echo $comments;?></textarea>

Similarly you can retain value of another control.

Following example shows the how the **sticky** forms can be created.

```
<form action="<?php echo $_SERVER[PHP_SELF]; ?>" method ="post">
<h1 align="center"> Cost Calculation </h1>
<table border="1" align="center" width="75%">
<tr> <td> Enter Distance in KM </td><td> <input type ="text" name="t1" value="<?php if(isset($_POST[t1])) echo $_POST[t1]; ?>"></td></tr>
<tr> <td> Enter Rate of Petrol </td><td> <input type ="text" name="t2" value="<?php if(isset($_POST[t2])) echo $_POST[t2]; ?>"></td></tr>
<tr> <td> Enter Average of Vichicle </td><td> <input type ="text" name="t3" value="<?php if(isset($_POST[t3])) echo $_POST[t3]; ?>"></td></tr>
<tr> <td> </td><td> <input type ="Submit" name="b1" value="Calculate"></td></tr>
</table>
</form>
<?php
$s=$_POST["b1"];
if(isset($s))
{
$dist=$_POST["t1"];
$rate=$_POST["t2"];
$avg=$_POST["t3"];
$p=$dist/$avg;
$cost=$p*$rate;
echo "<h1 align=center> Cost of Journey=" .$cost;
}
?>
```

4.4 Creating and Calling Your Own Functions

Functions are the self-contained block of statement written to solve a particular problem or to divide a program into set of units or blocks.

Using function readability of program increases. Means the program will be more readable & easy to understand.

With so many predefined functions, PHP provides the facility to define and use your own functions for any purpose.

The syntax for creating your own function is:

```
function function_name ()
{
// Function code.
}
```

The name of function can be any combination of letters, numbers, and the underscore, but it must begin with either a letter or the underscore and existing function name (print, echo, isset, and so on) cannot be used as a your function name.

Valid function definition is:

```
function display()
{
// Display data
}
```

In PHP function names are case-insensitive (but variables names are case sensitive), so you could call the above display function as:

Display(); DISPLAY(); display(); etc.

A simple function that writes my name when it is called:

```
<?php
function writeName()
{
echo "College of Computer Science & IT Latur";
}
writeName();?>
```

PHP Functions - Adding parameters

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

You may have noticed the parentheses after the function name, like: writeName(). The parameters are specified inside the parentheses.

```
<?php
function writeName($fname)
{
echo $fname . "<br />";
}
writeName("Kale Amol");
writeName("Sunil Mane");
writeName("Amit Mane");
?>
```

PHP Functions - Return values

Functions can also be used to return values.

Example:

```
<?php
function add($x,$y)
{
$total = $x + $y;
return $total;
}
echo "10 + 20 = " . add(10,20);
?>
```

The output of the code above will be:

10 + 20 = 30

Setting default argument values

This is another facility provided by PHP, through this you can specify the default value for the argument's value. If value not provided or passed while calling the function the default value will be taken.

Ex:

```
function Message ($name, $msg = 'Hello')
```

```
{
echo "$msg, $name!";
}
```

The above function called be called as

```
Message ('Amol'); // Hello Amol
Message ('Ram', 'Good evening'); // Good evening Ram
```

If a value is passed to it, the passed value is used; otherwise, the default value is used.

You can set default values for as many of the arguments as you want, as long as those arguments come last in the function definition.

In other words, the required arguments should always be listed first.

4.5 Variable Scope

Every variable in PHP has a scope to it, means the area or portion of program, in which the variable can be accessed.

Each variable declared in PHP page have the scope of that page in which they declared.

So if you define $var, the rest of the page can access $var, but other pages generally cannot.

Variables defined within the included file are available to the parent (including) script after the include () line.

User-defined functions have their own scope: variables defined within a function are not available outside of it, and variables defined outside of a function are not available within it.

For this reason, a variable inside of a function can have the same name as one outside of it but still be an entirely different variable with a different value. This is a confusing concept for many beginning programmers.

To alter the variable scope within a function, you can use the global statement.

```
function function_name()
{
global $var;
}
$var = 20;
function_name(); // Function call.
```

$var inside of the function is now the same as $var outside of it.

This means that the function $var already has a value of 20, and if that value

changes inside of the function, the external $var's value will also change.

The super-global variable like $_GET, $_POST, $_REQUEST, etc. are automatically accessible within any functions (because, they are super- global).

You can also add elements to the $GLOBALS array to make them available within a function.

4.6 Date and Time Functions

PHP provides date() & mktime() functions for manipulating date & time in php page or program.

The PHP date() function is used to format a time or a date.

The PHP date() function formats a timestamp to a more readable date and time.

Syntax of date function

date(format,timestamp)

Format -Required. Specifies the format of the timestamp

Timestamp -Optional. Specifies a timestamp. Default is the current date and time (as a timestamp)

A timestamp is the number of seconds since January 1, 1970 at 00:00:00 GMT. This is also known as the Unix Timestamp.

The first parameter in the date() function specifies how to format the date/time. It uses letters to represent date and time formats. Here are some of the letters that can be used as a first parameter

d - The day of the month (01-31)

D- The day of the month (Monday-Sunday)

m - The current month, as a number (01-12)

M - The current month, as a number (Jan-Dec)

F - The current month, as a number (January-December)

Y - The current year in four digits

y- The current year in two digits

h – Current hour in time.

i –minutes in current time

s - seconds in current time

Other characters, like"/", ".", or "-" can also be inserted between the letters to add additional formatting:

```
<?php
echo date("Y/m/d"); echo "<br />";
echo date("Y.m.d"); echo "<br />";
```

```
echo date("Y-m-d");
?>
```

The output

2006/07/11

2006.07.11

2006-07-11

PHP Date - Adding a Timestamp:

The second parameter in the date() function specifies a timestamp. This parameter is optional. If you do not supply a timestamp, the current time will be used.

In our next example we will use the mktime() function to create a timestamp for tomorrow.

The mktime() function returns the Unix timestamp for a specified date.

Syntax

mktime(hour,minute,second,month,day,year,is_dst);

To go one day in the future we simply add one to the day argument of mktime():

```
<?php
$tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y")); echo "Tomorrow is ".date("Y/m/d", $tomorrow);
?>
```

Output

Tomorrow is 2006/07/12

Ex.

```
<?php
$d=date("D");
if ($d=="Fri")
echo "Have a nice weekend!"; else
echo "Have a nice day!";
?>
$today = date("F j, Y, g:i a");
$today = date("D M j G:i:s T Y");
```

4.7 Sending Email

One of useful benefits of PHP is sending Email from PHP is easy. On a properly configured server, the process is simple. PHP provides mail() function for sending email the syntax of this function is:
mail (to, subject, body, [headers]);

The to value or parameter to this function should be an email address or a series of addresses, separated by commas.

Any of these are allowed:
darsh@rediff.com,
darsh@rediff.com,
harsh@gmail.com.

$to = “darsh@rediff.com”;
$subject = “This is the subject”;
$body = 'This is the body’.
“It goes over multiple lines”;

mail ($to, $subject, $body);

You can create an email message that goes over multiple lines by having the text do exactly that within the quotation marks. You can also use the newline character (\n) within double quotation marks to accomplish this:

$body = "This is the body.\n It goes over multiple lines.”;

The mail() function takes a fourth, optional parameter for additional headers. This is where you could set the From, Reply-To, Cc, Bcc, and similar settings. For example,

mail ($to, $subject, $body, 'From:reader@example.com');

To use multiple headers of different types in your email, separate each with \r\n:

$headers="From:amol@yahoo.com\r\n";
$headers= $headers ."Cc: darsh@rediff.com, harsh@gmail.com\r\n"; mail ($to, $subject, $body, $headers);

Although this fourth argument is optional, it is advised that you always include a from value (although that can also be established in PHP’s configuration file).

The End