Chapter1:- Server Side Scripting Basics in PHP

1.1 Overview of Client Side Script

- **Client-side scripting** generally refers to the class of computer programs on the web that are executed client-side, by the user's web browser
- Client-side scripts are often embedded within an HTML or XHTML document (hence known as an "embedded script"), but they may also be contained in a separate file, to which the document (or documents) that use it make reference (hence known as an "external script").
- instructions can be followed without further communication with the server
- By viewing the file that contains the script, users may be able to see its source code. Many web authors learn how to write client-side scripts partly by examining the source code for other authors' scripts.
- The client-side environment used to run scripts is usually a browser. The processing takes place on the end users computer. The source code is transferred from the web server to the user's computer over the internet and run directly in the browser.
- The scripting language needs to be enabled on the client computer. Sometimes if a user is conscious of security risks they may switch the scripting facility off. When this is the case a message usually pops up to alert the user when script is attempting to run.
- Includes :-
    - JavaScript, Action Script (used to create animated interactive web applications for Adobe Flash Player using Adobe Flash Pro), VBScript (NOTE: VBScript can also be used as Server-side so that processing is done on the server.), Python, html etc

1.2. Introduction to Server Side Script

Server-side scripting language, which means that the scripts are, executed on the server, the computer where the Web site is located.

Server-side scripting is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic web pages. It is usually used to provide interactive web sites that interface to databases or other data stores. This is different from client-side scripting where scripts are run by the viewing web browser.

The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores. From security point of view, server-side scripts are never visible to the browser as these scripts are executes on the server and emit HTML corresponding to user's input to the page.

In contrast, server-side scripts, written in languages such as PHP, ASP.NET, Java, ColdFusion, Perl, Ruby, Go, Python, and server-side JavaScript, are executed by the web server when the user requests a document. They produce output in a format understandable by web browsers (usually HTML), which is then sent to the user's computer. The user cannot see the script's source code (unless the author publishes

the code separately), and may not even be aware that a script was executed. Documents produced by server-side scripts may, in turn, contain client-side scripts.

Server-side Web scripting is mostly about connecting Web sites to back end servers, such as databases. This enables two-way communication:

- Client to server: Customer-entered information as request.
- Server to client: Web pages can be assembled from back end-server to give output.

Server-side scripting is about "programming" the behavior of the server while client-side scripting is about "programming" the behavior of the browser. Normally, when a browser requests an HTML file, the server returns the file. However, if the file contains a server-side script, the script is executed on the server before the file is returned to the browser as plain HTML.

A server script can do:-

- Dynamically edit, change or add any content to a Web page
- Respond to user queries or data submitted from HTML forms
- Access any data or databases and return the result to a browser
- Customize a Web page to make it more useful for individual users
- Provide security since server code cannot be viewed from a browser

In server side script, since the scripts are executed on the server, the browser that displays the file does not need to support scripting at all. The followings are server-side scripts:

- PHP (*.php)
- Active Server Pages (ASP)
- ANSI C scripts • Java via JavaServer Pages (*.jsp)
- JavaScript using Server-side JavaScript (*.ssjs)
- Lasso (*.lasso) etc

The main focus here is PHP, which is a server-side scripting language, which can be embedded in HTML or used as a standalone binary, and could be run with open source software like WAMP server.

- ✓ PHP can dynamically create the HTML code that generates the Web page.
- ✓ Web page visitors see the output from scripts, but not the scripts themselves.

1.3. PHP Basic Syntax

PHP stands for PHP: Hypertext Preprocessor. It is a server-side scripting language, which can be embedded in HTML. Over the past few years, PHP and server-side Java have gained momentum, while ASP has lost mindshare.

PHP is a server-side scripting language, which means that the scripts are executed on the server, the computer where the Web site is located. This is different than JavaScript, another popular language for dynamic Web sites. JavaScript is executed by the browser, on the user's computer. Thus, JavaScript is a client-side language.

Because PHP scripts execute on the server, PHP can dynamically create the HTML code that generates the Web page, which allows individual users to see customized Web pages. Web page visitors see the output from scripts, but not the scripts themselves.

PHP is particularly strong in its ability to interact with databases. PHP handles connecting to the database and communicating with it, so we don't need to know the technical details for connecting to a database or for exchanging messages with it, it is enough telling PHP the name of the database and where it is, and PHP handles the details. It connects to the database, passes our instructions to the database, and returns the database response to us.

Major databases currently supported by PHP include the following:

- dBASE ,Informix ,Ingres, Microsoft SQL Server ,mSQL ,MySQL ,Oracle ,PostgreSQL,Sybase and etc.

Hence, PHP scripts in the Web site can store data in and retrieve data from any supported database. PHP also can interact with supported databases outside a Web environment. Database use is one of PHP's best features.

Use of PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
- Using php, it is possible to add, delete, and modify elements within our database.
- It helps to assign sessions and cookies for privacy.
- Using PHP, we can restrict users to access some pages of the website.
- It can encrypt data and so mores

The syntax of PHP is: - There are four ways to write php syntaxes:-

1. **Canonical PHP tags:**
   - The most universally effective PHP tag style is:
     <?php statements written here..?>
   - This is the common one
2. **Short-open (SGML-style) tags**: Short or short-open tags look like this: <? Statements written here…?>
   - Short tags are, as one might expect, the shortest option
   - We must do one of two things to enable PHP to recognize the tags:
     - Choose the --enable-short-tags configuration option when building PHP.
     - Set the short_open_tag setting in php.ini file to on. This option must be disabled to parse XML with PHP because the same syntax is used for XML tags.
3. **ASP-style tags**: ASP-style tags mimic the tags used by Active Server Pages to delineate code blocks. ASP-style tags look like this: <%.. .%>
   - To use ASP-style tags, we should set the configuration option in your php.ini file.
4. **HTML script tags:** HTML script tags look like this:

<script language="php">.. . </script>

To work with server script, for example with PHP, we need to install a web server, programming itself, and database server. For PHP, we install PHP as a programming. There are many web servers to choose for PHP uses; the most commonly used web server is called Apache which we can download from the internet freely. Similarly, for database, there are many options to use; the most popular database server for web pages is **MySQL** which again can be downloaded freely from internet.

PHP is also available freely on the internet. In order to develop and run PHP Web pages three vital components need to be installed on computer system.

- **Web Server** - PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server.
- **Database** - PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.
- **PHP Parser** - In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser.

The official PHP website (PHP.net) has installation instructions for
PHP: http://php.net/manual/en/install.php

For the apache code to execute properly, it should be saved in web directory. The web directory depends on what web server is used. For example, for xampp web server, our web directory can be install in C:\xampp\htdocs". Hence, we should save PHP files in this folder, www.

For XAMPP server, we save the php code in C:\xampp\htdocs\your file here.

Writing PHP Comments and Output Statements

A comment is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP:

- ✓ **Single-line comments**: They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.
  <?php
  #This is a single line comment and
  // This also a single line comments too. Each style comments only print "An example with single line comments";
  ?>
- ✓ **Multi-lines comments**: They are generally used to provide pseudo code algorithms and more detailed explanations when necessary. Use /* and */ Here are the example of multi lines comments.
  <?php
  /* This is a comment with multiline
  ……
  ……

```
…….
*/
Print "An example with multi line comments";
?>
```

Output Statements:-

The two most basic constructs for displaying output in PHP are echo and print. Both can be used either with parentheses or without them.

Echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print. The echo or print statement can be used with or without parentheses: echo or echo().

The general format of the echo statement is as follows: echo outputitem1,outputitem2,outputitem3, . . .; echo (output);

The parameterized version of echo does not accept multiple arguments. The general format of the print statement is as follows:

print output;

print(output);

- Example: different ways of echo and print
  echo 123; //output: 123
  echo “Hello World!”; //output: Hello world!
  echo (“Hello World!”); //output: Hello world!
  echo “Hello”,”World!”; //output: Hello World!
  echo Hello World!; //output: error, string should be enclosed in quotes
  print (“Hello world!”); //output: Hello world!

The command print is very similar to echo, with two important differences:

- ✓ Unlike echo, print can accept only one argument.
- ✓ Unlike echo, print returns a value, which represents whether the print statement succeeded.

1.4. Variables and Constants

a. PHP Variables

A variable is a special container that can be defined to hold a value such as number, string, object, array, or a Boolean. The main way to store information in the middle of a PHP program is by using a variable. Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.

- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.

For example:-

- $distance = 2;
- $name = “stay home”;

As shown above, Numbers are not enclosed in quotes when they are assigned to variable. However, strings should be enclosed in either single or double quotes (“ or ‘). The quotes tell PHP that the characters are a string, handled by PHP as a unit. Without the quotes, PHP doesn’t know the characters are a string and won’t handle them correctly. PHP has a total of eight data types which we use to construct our variables:

1. **Integers**: are whole numbers, without a decimal point, like 4195.
2. **Doubles:** are floating-point numbers, like 3.14159 or 49.1.
3. Booleans: have only two possible values either true or false.
4. **NULL:** is a special type that only has one value: NULL. simply assign it like this: $my_var = NULL; The special constant NULL is capitalized by convention, but actually it is case insensitive. A variable that has been assigned NULL has the following properties:
    - It evaluates to FALSE in a Boolean context.
    - It returns FALSE when tested with IsSet() function.
5. **Strings:** are sequences of characters, like 'PHP supports string operations.'
6. **Arrays:** are named and indexed collections of other values.
7. **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
8. **Resources:** are special variables that hold references to resources external to PHP (such as database connections). The first five are simple types, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

Strings:

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";

$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php

$variable = "name";

$literally = 'My $variable will not print!\\n';

echo($literally);

$literally = "My $variable will print!\\n";

echo($literally);

?>
```

Find output to the above code.

There are no artificial limits on string length - within the bounds of available memory, we ought to be able to make arbitrarily long strings. Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP:

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are:

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

PHP provides a large number of predefined variables to all scripts. The variables represent everything from external variables to built-in environment variables, last error messages to last retrieved headers.

- ✓ **Superglobals** — Superglobals are built-in variables that are always available in all scopes
- ✓ **$GLOBALS** — References all variables available in global scope
- ✓ **$_SERVER** — Server and execution environment information
- ✓ **$_GET** — HTTP GET variables
- ✓ **$_POST** — HTTP POST variables
- ✓ **$_FILES** — HTTP File Upload variables
- ✓ **$_REQUEST** — HTTP Request variables, and can replace $_POST, $_GET and $_COOKIE variables
- ✓ **$_SESSION** — Session variables
- ✓ **$_COOKIE** — HTTP Cookies
- ✓ **$php_errormsg** — The previous error message

- ✓ **$HTTP_RAW_POST_DATA** — Raw POST data
- ✓ **$http_response_header** — HTTP response headers
- ✓ $**argc** — The number of arguments passed to script
- ✓ $**argv** — Array of arguments passed to script

Many of these variables, however, cannot be fully documented as they are dependent upon which server are running, the version and setup of the server, and other factors.

Removing Variables

- We can uncreated the variable by using this statement: **unset(VariableName**);
- After this statement, the variable $age no longer exists. If we try to echo it, you get an "undefined variable" notice. It is possible to unset more than one variable at once, as follows: **unset($age, $name, $address);**

Variable Scope:

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of three scope types:

Local variables:- The variable is only accessible from within the function (or method) that created it A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function:

```
<?php

$x = 4;

function assignx () {

$x = 0;

echo "\$x inside function is $x. ";

}

assignx();

echo "\$x outside of function is $x. ";

?>
```

The output will be:-

- o $x inside function is 0.
- o $x outside of function is 4.

Global variables:- The variable is accessible from anywhere in the script. Global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified. This is accomplished,

conveniently enough, by placing the keyword GLOBAL in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example:

```
<?php

$somevar = 15;

function addit() {

GLOBAL $somevar;

$somevar++ ;

echo "Somevar is $somevar";

 }

addit();

?>
```

The output will be:-

- Somevar is 16

Static variables:- this type of variables be either a global or local variable. Both are created by preceding the variable declaration with the keyword static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

```
<?php

function keep_track() {

 STATIC $count = 0;

$count++;

echo $count; print " ";

}

 keep_track();

keep_track();

keep_track();

?>
```

This will produce following result. 1

2

3

Variable Naming:

Rules for naming a variable are:

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , (,) . & , etc
- There is no size limit for variables.

b. PHP Constants

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If we have defined a constant, it can never be changed or undefined.

To define a constant we have to use define() function and to retrieve the value of a constant. Unlike with variables, you do not need to have a constant with a $. We can also use the function constant() to read a constant's value if we wish to obtain the constant's name dynamically.

constant () function is used to return the value of the constant. This is useful when we want to retrieve value of a constant, but we do not know its name, i.e. It is stored in a variable or returned by a function.

constant () example:

```
<?php

define("MINSIZE", 50);

echo MINSIZE;

echo”<br>”; echo constant("MINSIZE"); // same thing as the previous line

?>
```

Only scalar data (boolean, integer, float and string) can be contained in constants. PHP provides a large number of predefined constants to any script which it runs. There are five magical constants that change depending on where they are used. For example, the value of __LINE__ depends on the line that it's used on in script.

The name of a constant follows the same rules as any label in PHP. A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thusly: [a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*

1.5. PHP Operators (Reading Assignment)

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

The ternary operator

One especially useful construct is the ternary conditional operator, which plays a role somewhere between a Boolean operator and a true branching construct. Its job is to take three expressions and use the truth value of the first expression to decide which of the other two expressions to evaluate and return. The syntax looks like:

test-expression ? yes-expression: no-expression;

The value of this expression is the result of yes-expression if test-expression is true; otherwise, it is the same as no-expression. For example, the following expression assigns to $max_num either $first_num or $second_num, whichever is larger: $max_num = $first_num > $second_num ? $first_num : $second_num;

Example:-

<?php

$x=2;

$y=10; If $y%$x>3?"$y value is more than 3 folds of $x":"$y is not large enough";

?>

1.6. Manipulate Strings

PHP has many functions to work with strings. The most commonly used functions for searching and modifying strings are those that use regular expressions to describe the string. The followings are string manipulation operations:-

i. **String concatenation operation: -** To concatenate two string variables together, use the dot (.) operator like echo $string1 . " " . $string2;
ii. **strpos() function**:- used to search for a string or character within a string. If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.
Written as strops(orginal string, new string)
Example: the following code used to show from where the word "world" started.
<?php
echo strpos("Hello world!","world");

?>
The output will be 6. As seen the position of the string "world" in our string is position 6. The reason that it is 6, and not 7, is that the first position in the string is 0, and not 1.

iii. **The strrev( ) function:-** takes a string and returns a reversed copy of it.
Has a syntax:-
$string = strrev(string);
For example:
echo strrev("study hard");
drah yduts

iv. **The strlen() function**:- is used to find the length of a string. For example:- To find the length of "Hello world!", we can write as follows
<?ph
echo strlen("Hello world!");
?>
(Reading Assignment)

There are many string function in php.

Unit 2: PHP Statements and Form Creations

2.1. PHP statements: - Use Conditionals/Decision Making, LOOP, Arrays

2.1.1. Conditionals/Decision Making Statements

The if, elseif ...else and switch statements are used to take decision based on the different condition.

- **The If...Else Statement**

  If you want to execute some code if a condition is true and another code if a condition is false.

  We use this statement if we want to execute a set of code when a condition is true and another if the condition is not true

  Syntax

  if (condition)
  code to be executed if condition is true;
  else code to be executed if condition is false;

  Example

```
<?php
$x = "Amanuel";
 if ($x == "Dani")
 {
echo 'Hello Dani!';
 }
else {
echo 'You are not Dani ! are you Amanuel?';
 echo “yes, you are my friend”;
}
?>
```

- **The ElseIf Statement**

  If you want to execute some code if one of several conditions is true use the elseif statement ¬
  This statement uses to execute a set of code if one of several condition are true

  Syntax

  if (condition)
  code to be executed if condition is true;
  elseif (condition) code to be executed if condition is true;
  else code to be executed if condition is false;

  If more than one line should be executed in a condition of true/false, the lines should be enclosed within curly braces as shown below.

  Example

```
<?php
$d=date("D");
if ($d=="Fri"){
echo "Have a nice weekend!";
echo “have good refreshment”;
}
elseif ($d=="Sun")
echo "Have a nice Sunday!";
```

```
else
echo "Have a nice day!";
?>
```

- **The Switch Statement**

Avoid long blocks of if..elseif..else code.
Could execute one of many blocks of code / output
The switch statement is used to avoid long blocks of if..elseif..else code.
Syntax

```
switch (expression)
 {
case label1: code to be executed if expression = label1;
break;
case label2: code to be executed if expression = label2;
break;
.
.
 .
case label n:
default: code to be executed if expression is different from both label1 and label n;
 }
```

Example:

```
<?php
$name=”kal”;
switch ($name)
{
case "esayas" :
echo "you are esayas "; break;
case "Adane" :
echo "you are adane"; break;
case " eyob " :
echo "you are eyob ";
case "kal" :
echo "you are a fantastic person"; break;
default :
echo "Unknown Person”;"; break;
}
?>
```

2.1.2. PHP Loop Types

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following loop types with **continue** and **break** keywords which uses to control the loops execution.

- ✓ **for** - loops through a block of code a specified number of times.
- ✓ **while** - loops through a block of code if and as long as a specified condition is true.
- ✓ **do...while** - loops through a block of code once, and then repeats the loop as long as a special condition is true.
- ✓ **foreach** - loops through a block of code for each element in an array.

While Loop

- The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {
  code to be executed;
}
```

The example below first sets a variable $x to 1 ($x=1;). Then, the while loop will continue to run as long as $x is less than, or equal to 5. $x will increase by 1 each time the loop runs ($x++;):

- ✓ Example

```
<?php
$x=1;
while($x<=5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

Do…While Loop

- The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
  code to be executed;
} while (condition is true);
```

The example below first sets a variable $x to 1 ($x=1;). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

- ✓ Example

```
<?php
$x=1;
do {
  echo "The number is: $x <br>";
  $x++;
} while ($x<=5);
?>
```

Notice that in a do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition fails the first time.

PHP for Loops

- PHP for loops execute a block of code a specified number of times.

Syntax

```
for (init counter; test counter; increment counter) {
  code to be executed;
}
```

Parameters:

- init counter: Initialize the loop counter value
- test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- increment counter: Increases the loop counter value

Example

```
<?php
for ($x=0; $x<=10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

The PHP for each Loop

- The for each loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

For each ($*array* as $*value*) {
code to be executed;
}

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Example

```
<?php
$colors = array("red","green","blue","yellow");
for each ($colors as $value) {
  echo "$value <br>";
}
?>
```

Php Arrays

- ✓ An array stores multiple values in one single variable:

Example

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

What is an Array?

An array is a special variable, which can hold more than one value at a time.

Create an Array in PHP

In PHP, the array() function is used to create an array:

- array();

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

There are two ways to create indexed arrays:

- The index can be assigned automatically (index always starts at 0):

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

- or the index can be assigned manually:

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

Example

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

Get The Length of an Array - The count() Function

The count() function is used to return the length (the number of elements) of an array:

Example

```
<?php
$cars=array("Volvo","BMW","Toyota");
```

```
echo count($cars);
?>
```

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

Example

```
<?php
$cars=array("Volvo","BMW","Toyota");
$arrlength=count($cars);

for($x=0;$x<$arrlength;$x++) {
  echo $cars[$x];
  echo "<br>";
}
?>
```

PHP Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:
  $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");

Example

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

Example

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
foreach($age as $x=>$x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
?>
```

MULTIDIMENSIONAL ARRAYS (READING ASSIGNEMENTS)

2.2PHP GET, POST & REQUIST Methods

- Usually used when we create forms to make communication between interfaces and databases.

The GET Method:-

- ✓ Has restriction to send to server/ database parts up to 1024 characters only.
- ✓ GET can't be used to send binary data, like images or word documents, to the server because the GET method sends the encoded user information.
- ✓ The data sent by GET method can be accessed using QUERY_STRING environment variable.
- ✓ Never use GET method for systems which have password or other sensitive information to be sent to the server.
- ✓ The $_GET variable is used to collect values from a form with method="get".

Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) For example http://localhost/xy.php?name=bekele$age=39

Example:- let we save as xy.php

```
<?php
if( $_GET["name"] || $_GET["age"] )
{
echo "Welcome ". $_GET['name']. "<br />";
 echo "You are ". $_GET['age']. " years old.";
exit();
 }
?>
<html>
<body>
 <form action="<?php $_PHP_SELF ?>" method="GET">
Name: <input type="text" name="name" />
 Age: <input type="text" name="age" />
<input type="submit" />
 </form>
</body>
</html>
```

The above code's action attribute value can be represented as the file names itself like:-
<form action="xy.php" method="Get">

The POST Method

The POST method transfers information via HTTPs headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- ✓ The POST method does not have any restriction on data size to be sent.
- ✓ Relatively secured and could large data in requesting and responding data

- ✓ The POST method can be used to send ASCII as well as binary data.
- ✓ The data sent by POST method goes through HTTP header is secured enough on HTTP protocol. The PHP provides $_POST associative array to access all the sent information using POST method.
- ✓ Variables sent with HTTP POST are not shown in the URL
- ✓ The $_POST variable is used to collect values from a form with method="post".

Information sent from a form with the POST method is invisible to others For example http://localhost/xy.php

Example:- as example, we could change GET by post from example.

The $_REQUEST variable

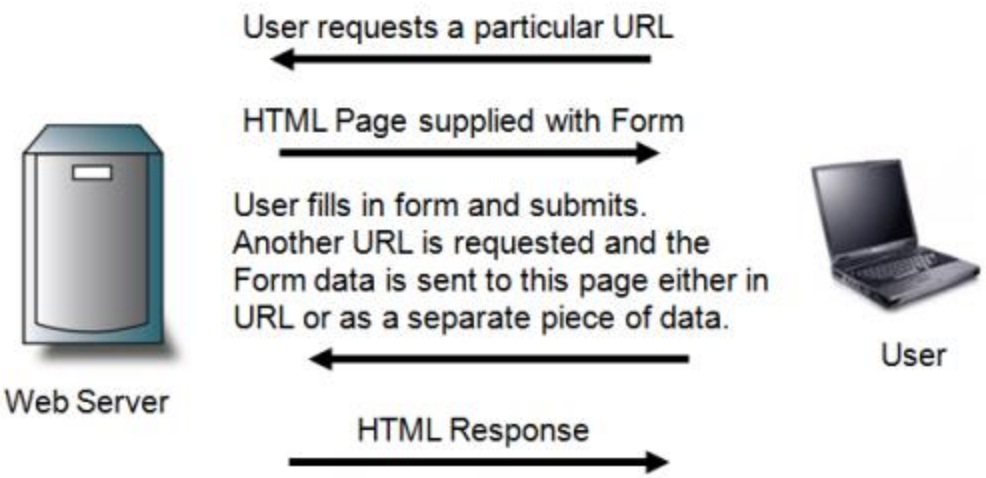
- ➢ The PHP $_REQUEST variable contains the contents of $_GET, $_POST, and $_COOKIE variables
- ➢ This variable can be used to get the result from form data sent with both the GET and POST methods.

Example:- <?php

```
if( $_REQUEST["name"] || $_REQUEST["age"] )
 {
 echo "Welcome ". $_REQUEST['name']. "<br />";
 echo "You are ". $_REQUEST['age']. " years old.";
 exit(); }
?><html>
<body>
 <form action="<?php $_PHP_SELF ?>" method="POST">
 Name: <input type="text" name="name" />
 Age: <input type="text" name="age" />
 <input type="submit" />
 </form>
</body>
</html>
```

2.3 Creating PHP Forms

- ➢ 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 PHP scripts.
- ➢ It works as follows:-



- The syntax could be written as -
  <form action="url to submit the form filled" method="get" or "post" or "request">
  <!–- form contents -->
  </form>

- Where
  - action="…" is the page that the form should submit its data to, and
  - method="…" is the method by which the form data is submitted. If the method is get the data is passed in the url string, if the method is post it is passed as a separate file.

The form variables are available to PHP in the page to which they have been submitted. The variables are available in two superglobal arrays created by PHP called $_POST and $_GET.

The basic concept that is important to understand is that any form element will automatically be available to PHP scripts. See the following example:

```
<form action="action.php" method="post">
Your name: <input type="text" name="name" />
Your age: <input type="text" name="age" />
<input type="submit" />
</form>
```

When the user fills in this form and hits the submit button, the action.php page is called. In this file we would write something like this:

```
<?php
echo ("you are ($_POST['name']");
echo $_POST['age']; years old.
?>
```

2.4 Form Validation

If a user who forgot to enter one of the fields or enter wrong input, we need to validate the form to make sure it's complete and filled out with valid information. We can use JavaScript for this validation. Validations can also be done with simple PHP if statements as shown below.

When the process is done, if it is done making validations, it will check to see if there is an error message. If there is, it displays the error message. If there are no errors, it displays a success message.

```
<Html>
<Body>
<form action="a.php" method="post">
Your Name: <input type="text" name="yourname" /><br />
 E-mail: <input type="text" name="email" /><br/>
<p>Do you like this website?
<input type="radio" name="likeit" value="Yes" checked="checked" /> Yes
<input type="radio" name="likeit" value="No" /> No
<input type="radio" name="likeit" value="Not sure" /> Not sure</p><br/>
<p>Your comments:<br />
<textarea name="comments" rows="10" cols="40"></textarea></p>
<p><input type="submit" value="Send!"></p>
</form>
</body>
</html>
```

Note that:- from the above code,

- All variables passed to the current script via the HTTP POST method are stored in associative array $_POST. For example, in PHP we can access data from each field using $_POST['NAME'], where NAME is the actual field name.
- If we submit the above form, we would have access to a number of $_POST array values inside the a.php file:

In php, we can check the validity of inputs such as URL, E-mail, digits, letters and other special characters etc using functions. For example preg_match() function used to match list of inputs with defined lists. For example, see the following rules:-

i. **URL Address:-** If there is an input field named "website" we can check for a valid URL address like this:

```
$url = htmlspecialchars($_POST['website']);
if (!preg_match("/^(https?:\/\/+[\w\-]+\.[\w\-]+)/i",$url)) {
 die("URL address not valid");
}
```

From the code given above, if the input held by $url is not match with the given string , then the die() function force the system to terminate the running .

ii. **Digits 0-9 only: -** This uses to check whether an input is digit/ number or not.

The following is a syntax to check if $age is a number or not. If not number, it display “Please enter numbers only for Age” .

```
$age= htmlspecialchars($_POST['age']);

if (!preg_match("/\D/",$age)) {

die("Please enter numbers only for Age");

}
```

iii. **Validate e-mail address:-** Used to check an email is valid, i.e to have valid forms.
There is a simple way to check if data entered into input field named "email" is an e-mail address without any unnecessary complications and fancy regular expressions.

```
$email = htmlspecialchars($_POST['email']);
if (!preg_match("/([\w\-]+\@[\w\-]+\.[\w\-]+)/",$email)) {
 die("E-mail address not valid");
}
```

iv. **Letters a-z and A-Z only:-** This code will check if $text is made of letters a-z and A-Z only (no spaces, digits or any other characters):

```
$name = test_input($_POST["name"]);
 if (preg_match("/[^a-zA-Z]/",$text)) {
die("Please enter letters a-z and A-Z only!");
}
```

Please read Lab tutorials more about the forms and form validations.

Chapter 3 Files and Directories in PHP

3.1. Reading files/ Directories

Files and directories have three levels of access: User, Group and Other. The three typical permissions for files and directories are: Read (r), Write (w) and Execute (x) A stream is a channel used for accessing a resource that we can read from and write to. The input stream reads data from a resource (such as a file) while the output stream writes data to a resource.

Once a file is opened using fopen() function it can be read with a function called fread(). This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes. The file's length can be found using the filesize() function which takes the file name as its argument and returns the size of the file expressed in bytes.

So here are the steps required to read a file with PHP.

- Open a file using fopen() function. Syntax:- variable= fopen("text file", "mode"); where mode mean r, r+, w, w+ etc as shown in the next page.
- Get the file's length using filesize() function. The syntax is filesize($filename);
- Read the file's content using fread() function. Has syntax variable = fread(filename, filesize);
- Close the file with fclose() function. It uses when finished working with a file stream to save space in memory.
  Syntax:- fclose(file name that contains the opened files);
  $handle from the following example.

Example:- Files modes can be specified as one of the six options in this table.

Mode	Descriptions /Purpose
r	Opens the file for reading only. Places the file pointer at the beginning of the file.
r+	Opens the file for reading and writing. Places the file pointer at the beginning of the file.
w	Opens the file for writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
w+	Opens the file for reading and writing only. Places the file pointer at the beginning of the file. And truncates the file to zero length. If files does not exist then it attempts to create a file.
A	Opens the file for writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.
a+	Opens the file for reading and writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.

Creating Directories in PHP

- A new directory can be created in PHP using the *mkdir()* function.
- This function takes a path to the directory to be created.

- To create a directory in the same directory as your PHP script simply provide the directory name.
- To create a new directory in a different directory specify the full path when calling *mkdir()*.

```
<?php

$result = mkdir ("/path/to/directory", "0777");

?>
```

Deleting a Directory

- Directories are deleted in PHP using the ***rmdir()*** function.
- ***rmdir()*** takes a single argument, the name of the directory to be deleted.
- The deletion will only be successful if the directory is empty.
- If the directory contains files or other sub-directories the deletion cannot be performed until those files and sub-directories are also deleted.

Read files in PHP

- The **fread()** function reads from an open file.
- The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.
- Example:-

```
<?php
$myFile = "sampleFile2.txt";
$fh = fopen($myFile, 'r');
$myFileContents = fread($fh,filesize("samplefile2.txt"));
fclose($fh);
echo $myFileContents;
?>
```

Write to File in PHP

- The **fwrite()** function is used to write to a file.
- The first parameter of **fwrite()** contains the name of the file to write to and the second parameter is the string to be written.
- Example:

```
<?php
$myFile2 = "sampleFile2.txt";
$myFileLink2 = fopen($myFile2, 'a') or die("Can't open file.");
$newContents = "I am a student";
fwrite($myFileLink2, $newContents);
fclose($myFileLink2);
?>
```

Open File in PHP

- A better method to open files is with the **fopen()** function.
- The first parameter of **fopen()** contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

3.2. Upload Files

- Web applications allow visitors to upload files to and from their local computer. The files that are uploaded and downloaded may be simple text files or more complex file types, such as images, documents, or spreadsheets
- Files are uploaded through an HTML form using the "post" method and enctype attribute with value of "multipart/form-data," which instructs the browser to post multiple sections – one for regular form data and one for the file contents.
- The file input field creates a browser button for the user to navigate to the appropriate file to upload <form method="post" action=" " enctype= multipart/form-data >
  <input type="file" name="picture_file" /> </form>
- The MAX_FILE_SIZE (uppercase) attribute of a hidden form field specifies the maximum number of bytes allowed in the uploaded file and it must appear before the file input field.
- When the form is posted, information for the uploaded file is stored in the $_FILES auto global array.
- Example:-The following HTM code below creates an uploaded form. This form is having method attribute set to post and enctype attribute is set to multipart/form-data

```
<html> <body>
<h3>File Upload:</h3>
Select a file to upload: <br />
<form action="<?php $_PHP_SELF ?>" method="post"
 enctype="multipart/form-data">
<input type="file" name="file" size="50" />
<br />
<input type="submit" value="Upload File" />
</form>
<?php
if( $_FILES['file']['name'] != "" ){
 copy( $_FILES['file']['name'], "C:\wamp\www\Leture\Test.php" ) or die( "Could
not copy file!");
}
else{
 die("No file specified!");}
?>
<html>
<head>
<title>Uploading Complete</title>
</head>
```

```
<body>
<h2>Uploaded File Info:</h2>
<ul>
<li>Sent file: <?php echo $_FILES['file']['name']; ?>
<li>File size: <?php echo $_FILES['file']['size']; ?> bytes
<li>File type: <?php echo $_FILES['file']['type']; ?>
</ul>
</body>
</html>
</body></html>
```

3. 3 PHP Cookies and Session

What is a Cookie?

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

- A cookie is created with the setcookie() function.

Syntax

- setcookie(name, value, expire, path, domain, secure, httponly);
- Only the name parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

- The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).
- We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set:

Note: The setcookie() function must appear BEFORE the <html> tag.

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>
```

```
<?php
if(!isset($_COOKIE[$cookie_name])) {
   echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
   echo "Cookie '" . $cookie_name . "' is set!<br>";
   echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

PHP Session

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- Session variables hold information about one single user, and are available to all pages in one application.
- If you need a permanent storage, you may want to store the data in a database.
- A session is started with the session_start() function.
- Session variables are set with the PHP global variable: $_SESSION.

Example

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>
</body>
</html>
```

- **Note:** The session_start() function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?>
</body>
</html>
```

3.4 PHP - File Inclusion

You can include the content of a PHP file into another PHP file before the server executes it. There are two PHP functions which can be used to included one PHP file into another PHP file.

- ✓ **The include() Function**
- ✓ **The require() Function**

This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of complete

website with minimal effort. If there is any change required then instead of changing thousand of files just change included file.

The include() Function

- ✓ The include() function takes all the text in a specified file and copies it into the file that uses the include function.
- ✓ If there is any problem in loading a file then the include() function generates a warning but the script will continue execution.
- ✓ Assume you want to create a common menu for your website. Then create a file menu.php with the following content.

```
<a href="http://www.tutorialspoint.com">Home</a>
<a href="http://www.w3schools.com">w3schools</a>
<a href="http://www.ajax.com">AJAX</a>
 <a href="http://www.mysql.com">MySQL</a> <br />
```

Now create as many pages as you like and include this file to create header. For example now your **test.php** file can have following content.

```
<html>
<body>
<?php
include("menu.php");
?>
<p>This is an example to show how to include PHP file!</p>
 </body>
</html>
```

The require() function

- ✓ The require() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the **require()** function generates a fatal error and halt the execution of the script.
- ✓ So there is no difference in require() and include() except they handle error conditions.
- ✓ It is recommended to use the require() function instead of include(), because scripts should not continue executing if files are missing or misnamed.
- ✓ You can try using above example with require() function and it will generate same result. But if you will try following two examples where file does not exist then you will get different results.

```
<html>
<body>
 <?php
include("xxmenu.php");
?>
<p>This is an example to show how to include wrong PHP file!</p>
</body>
```

```
</html>
```

Output:- This is an example to show how to include wrong PHP file!

```
<html>
<body>
    <?php
      require("xxmenu.php");
    ?>
<p>This is an example to show how to include wrong PHP file!</p>
</body>
</html>
```

Output:-This time file execution halts and nothing is displayed.

NOTE:-You may get plain warning messages or fatal error messages or nothing at all. This depends on your PHP Server configuration.

Chapter 4:- PHP Database / PHP with MYSQL

- With PHP, you can connect to and manipulate databases.
- MySQL is the most popular database system used with PHP.

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation

The data in a MySQL database are stored in tables. A table is a collection of related data, and it consists of columns and rows.

Database Queries

- ✓ A query is a question or a request.
- ✓ We can query a database for specific information and have a recordset returned.
- ✓ Example: SELECT LastName FROM Employees

Connecting to databases

To connect PHP with database, four important things must be taken place. Those are:-

- ✓ **Define constants**
- ✓ **Create connection using mysql_connect.**
- ✓ **Select database.**
- ✓ **Close connection.**

1. Define constants

To connect php with database, defining constants is very important. Constants that must be defined are:-

- ✓ *Define the server.*
- ✓ *Define user name of the server.*
- ✓ *Define password of the user.*
- ✓ *Define database name.*

To define the constant, we can use as follows:

```
<?php

define("db_server","localhost");// the server that we use

define("db_user","root");// the username of the server

define("db_pass","");//password of the server

define("db_name","schoolmgt");//the database that we use

?>
```

2. **Opening Database Connection**

After defining constants using php, opening or creating connection is very important. To open or create database connection, we use **mysql_connect** function**.**

This function takes three parameters. Those are:

- db_server,db_user,db_pass.
- ✓ Db_server:-The host name running database server
- ✓ Db_user:-The username accessing the database
- ✓ Db_pass:-The password of the user accessing the database.

From the above the host name running database server is "localhost", the username accessing the database is "user", and the password of the user accessing the database is empty. Connection can opened or created as follows:

$connection=mysql_connect(db_server,db_user,db_pass);

3. Select database

Once you establish a connection with a database server then it is required to select a particular database where your all the tables are associated.

This is required because there may be multiple databases residing on a single server and you can do work with a single database at a time.

To select database, we use **mysql_select_db** as follows**:-**

$db_select=mysql_select_db(db_name,$connection);

4. Closing database connection

Its simplest function **mysql_close** PHP provides to close a database connection. This function takes connection resource returned by **mysql_connect** function. For example, to close the connection that you use in the above; you use **mysql_close** function as follows:-

mysql_close($connection);

Create MySQL Database Using PHP

To create and delete a database you should have admin privilege. Its very easy to create a new MySQL database. PHP uses **mysql_query** function to create a MySQL database. For example to create the database test_db using php, you can write as follows:-

```
<?php

define("db_server","localhost");

define("db_user","root");

define("db_pass","");

$connection=mysql_connect(db_server,db_user,db_pass);

if(!$connection)

{

die("error connection to db server".mysql_error());

}

echo “Connected successfully”;

$sql = “CREATE Database test_db”;

$retval = mysql_query( $sql, $connection);

 if(! $retval ) {

die('Could not create database: ' . mysql_error());

}

echo "Database test_db created successfully\n";

mysql_close $connection);

?>
```

Creating Database Tables

To create tables in the new database you need to do the same thing as creating the database. First create the SQL query to create the tables then execute the query using **mysql_query**() function.

```
<?php

define("db_server","localhost");

define("db_user","root");
```

```
define("db_pass","");
define("db_name","test_db");
$con=mysql_connect(db_server,db_user,db_pass);
if(!$con){
die("not connected".mysql_query());
}
$sqldb=mysql_select_db(db_name,$con);
if (!$sqldb){
die("incorrectly selected".mysql_error());
}
$sql = 'CREATE TABLE employee( '.
    'emp_id INT NOT NULL AUTO_INCREMENT, '.
    'emp_name VARCHAR(20) NOT NULL, '.
    'emp_address  VARCHAR(20) NOT NULL, '.
    'emp_salary   INT NOT NULL, '.
    'join_date    timestamp(14) NOT NULL, '.
    'primary key ( emp_id ))';
$retval = mysql_query($sql,$con);
  if(!$retval ) {
    die('Could not create table: ' . mysql_error());
  }
  echo "Table employee created successfully\n";
?>
```

In case you need to create many tables then its better to create a text file first and put all the SQL commands in that text file and then load that file into $sql variable and execute those commands. Consider the following content in **sql_query.txt** file.

```
CREATE TABLE employee
```

```
(emp_id INT NOT NULL AUTO_INCREMENT,
emp_name VARCHAR(20) NOT NULL,
emp_address VARCHAR(20) NOT NULL,
emp_salary INT NOT NULL,
join_date timestamp(14) NOT NULL,
primary key ( emp_id ));

<?php
define("db_server","localhost");
define("db_user","root");
define("db_pass","");
define("db_name","test_db");
$con=mysql_connect(db_server,db_user,db_pass);
if(!$con){
die("not connected".mysql_query());
}
$sqldb=mysql_select_db(db_name,$con);
if (!$sqldb){
die("incorrectly selected".mysql_error());
}
$query_file = 'sql_query.txt';
$fp = fopen($query_file, 'r');
$sql = fread($fp, filesize($query_file));
fclose($fp);
$retval = mysql_query($sql,$con);
  if(!$retval ) {
```

```
    die('Could not create table: '.mysql_error());
  }
  echo "Table employee created successfully\n";
?>
```

Insert Data into MySQL Database

Data can be entered into MySQL tables by executing SQL INSERT statement through PHP function **mysql_query**. Below a simple example to insert a record into **employee** table.

```
sql = 'INSERT INTO employee '.
    '(emp_name,emp_address, emp_salary, join_date) '.
    'VALUES ( "guest", "XYZ", 2000, NOW() )';
```

Getting Data From MySQL Database

Data can be fetched from MySQL tables by executing SQL **SELECT** statement through PHP function mysql_query. You have several options to fetch data from MySQL.

The most frequently used option is to use function **mysql_fetch_array()**. This function returns row as an associative array, a numeric array, or both. This function returns FALSE if there are no more rows.

next is a simple example to fetch records from **employee** table.

```
$sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
$retval = mysql_query($sql,$con);
  if(!$retval ) {
    die('Could not create table: ' . mysql_error());
  }
  while($row = mysql_fetch_array($retval, MYSQL_ASSOC)) {
    echo "EMP ID :{$row['emp_id']}  <br> ".
      "EMP NAME : {$row['emp_name']} <br> ".
      "EMP SALARY : {$row['emp_salary']} <br> ".
      "--------------------------------<br>";
  }
```

The content of the rows are assigned to the variable $row and the values in row are then printed.Always remember to put curly brackets when you want to insert an array value directly into a string.

In above example the constant **MYSQL_ASSOC** is used as the second argument to **mysql_fetch_array**(), so that it returns the row as an associative array. With an associative array you can access the field by using their name instead of using the index.

PHP provides another function called **mysql_fetch_assoc()** which also returns the row as an associative array.

Using mysql_fetch_assoc()

```
while($row = mysql_fetch_assoc($retval)) {

    echo "EMP ID :{$row['emp_id']}  <br> ".

      "EMP NAME : {$row['emp_name']} <br> ".

      "EMP SALARY : {$row['emp_salary']} <br> ".

      "--------------------------------<br>";

  }
```

Using MYSQL_NUM

```
while($row = mysql_fetch_array($retval, MYSQL_NUM)) {

    echo "EMP ID :{$row[0]}  <br> ".

      "EMP NAME : {$row[1]} <br> ".

      "EMP SALARY : {$row[2]} <br> ".

      "--------------------------------<br>";

  }
```

Deleting Data from MySQL Database

Data can be deleted from MySQL tables by executing SQL **DELETE** statement through PHP function **mysql_query**.

Below is a simple example to delete records into **employee** table. To delete a record in any table it is required to locate that record by using a conditional clause. Below example uses primary key to match a record in employee table.

```
$emp_id = $_POST['emp_id'];

$sql = "DELETE FROM employee WHERE emp_id= $emp_id" ;
```

Deleting MySQL Database and table Using PHP

If a database is no longer required then it can be deleted forever. You can use pass an SQL command to mysql_query to delete a database.

$sql = 'DROP DATABASE test_db'; ---→deleting MySQL database

$sql = 'DROP TABLE employee'; ------→ deleting table

Updating Data into MySQL Database

Data can be updated into MySQL tables by executing SQL UPDATE statement through PHP function **mysql_query**.

Below is a simple example to update records into **employee** table. To update a record in any table it is required to locate that record by using a conditional clause. Below example uses primary key to match a record in employee table.

$emp_id=$_POST['emp_id'];

$emp_salary = $_POST['emp_salary'];

$sql = "UPDATE employee ". "SET emp_salary = $emp_salary ". "WHERE emp_id = $emp_id" ;