CHAPTER 1-PHP Introduction & Basic Syntex

What is PHP:

The **PHP (PHP: Hypertext Preprocessor)** is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web-based software applications.

- It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
- It is deep enough to run large social networks!
- It is also easy enough to be a beginner's first server-side language!
- PHP is a widely-used, open-source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

History of PHP

- PHP was developed by Rasmus Lerdorf in 1994.
- PHP version 2 was released in 1995.
- PHP version 3.0 was released by Zeev Suraski and Andi Gutmans along with Rasmus in June 1998.
- PHP version 4.0 was launched in May 2000. This version included session handling features, output buffering, a richer core language, and support for a wider variety of Web server platforms.
- PHP version 5.0 was released in July 2004. It supports Object Oriented Programming (OOP).

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images or PDF files. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

Features of PHP:

- PHP is available at free of cost under PHP General Public License.
- PHP uses procedural as well as object-oriented programming or mixture of them.
- PHP is platform-independent. It can run across various operating systems such as Windows, Linux, Mac OSX, Solaris etc.
- PHP is server-independent. It can run on various servers such as Apache, IIS etc.
- PHP is database-independent. It supports almost all popular databases such as MySQL, SQL Server, Oracle etc.

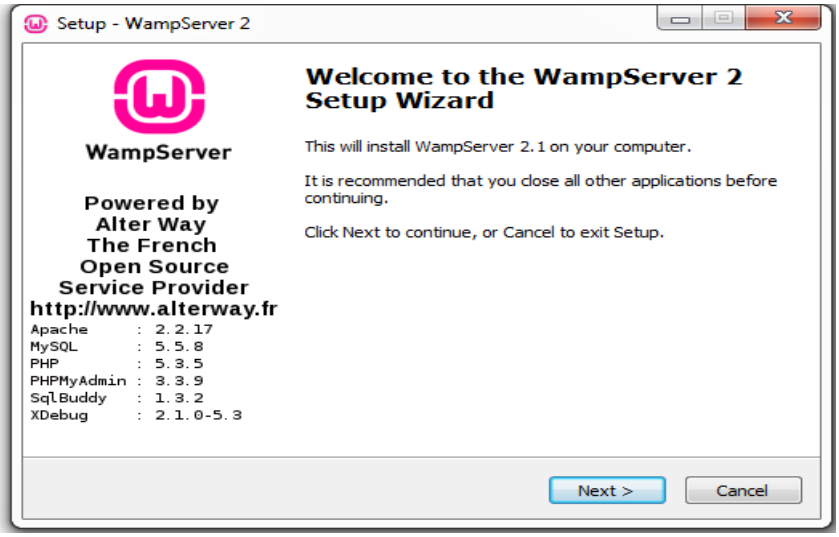
WAMP, LAMP & XAMPP:

Installing PHP

- PHP is a scripting language. It needs a platform (operating system) and it must be integrated with web server and database to be used for making web pages.
- Generally, Apache is used as a web server & MySQL is used as a database along with PHP on either Windows or Linux platform worldwide.
- There are two ways of doing this. Either you install each component separately or you can install an application which includes all of these components. Once you install the application, all components are installed together. The prior way is used with Linux platform and known as LAMP. For later way an application named WAMP Server or XAMPP Server is available for Windows platform.

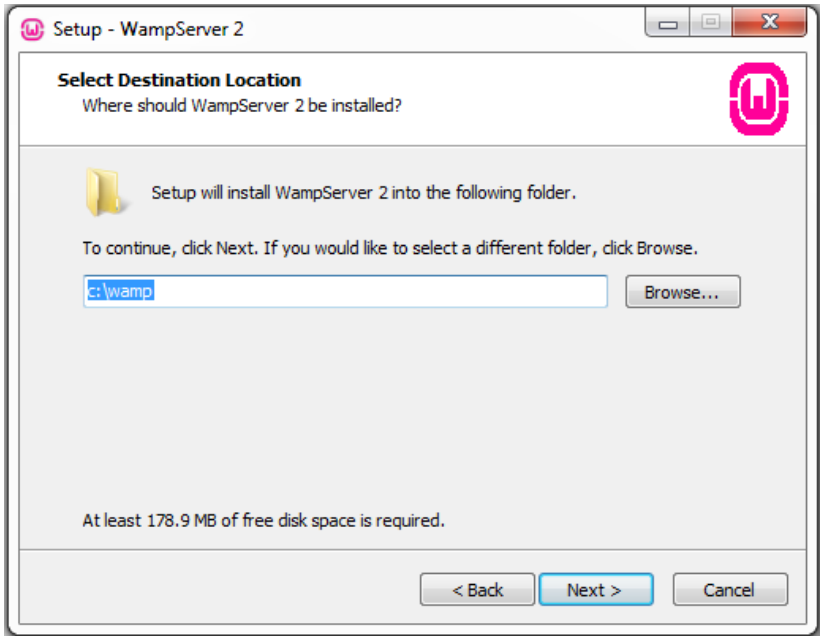
Installing PHP using WAMP on Windows
WAMP server is a server application to run PHP web pages. WAMP is an acronym for Windows, Apache, MySQL, PHP. Following are the steps for installing WAMP server on windows.

Step-1: Download the source of WAMP server for free from its website.
Step-2: Once downloaded, double click on it to run the executable file. You will see a setup wizard window as shown below:

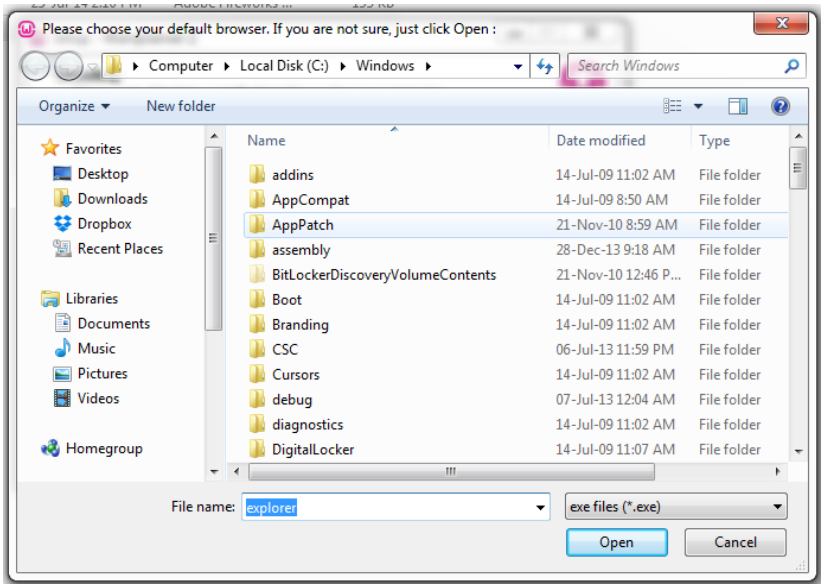


Step-3 Click on 'Next'. Accept the agreement on next screen and then again click on 'Next'.

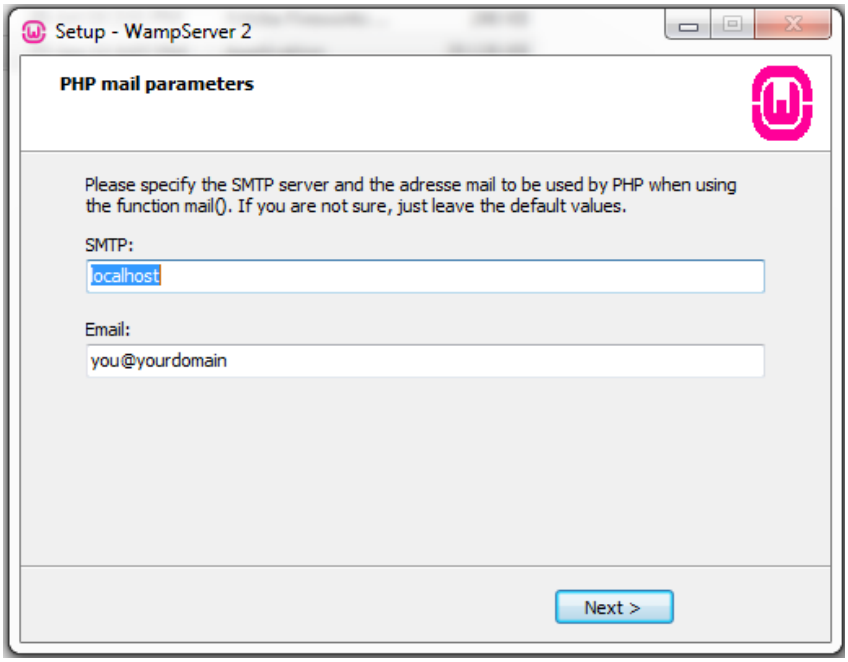
Step-4 Select the server location. Generally, it is installed into C drive as shown below.



Step-5 Click on 'Next' in following screen. The below screen appears. If you are not sure what name should be given leave it named as 'explorer'.

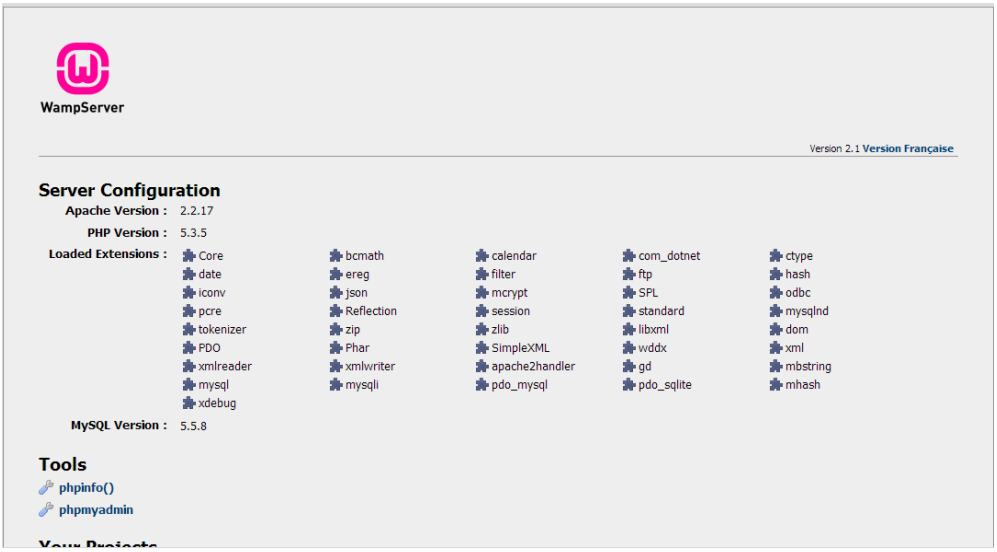


Step-6 Before finishing, it asks for SMTP and Email details. Leave the default values for it and click on 'Next'.



Step-7 Now, installation process is almost done. Click on 'Finish' to complete the installation.

You can verify the installation by opening any web browser. Type 'http://localhost' in the address bar. If following screen appears then WAMP server installation is successful.



Installing LAMP on Ubuntu

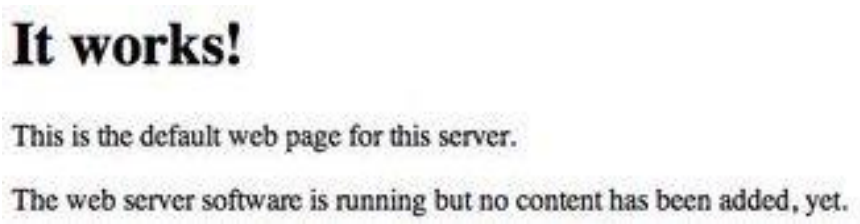
LAMP is an acronym for Linux, Apache, MySQL, PHP. It is a group of open source software used to install web server. Following are the steps to install LAMP on Ubuntu.

Step-1 First of all, we will install Apache. It is a free open source software. To install Apache, open terminal and type in following command:

sudo apt-get

sudo apt-get install apache2

By writing these two lines, Apache is installed. To verify the installation, type your server's IP address (e.g. http://136.56.45.87) and you can see the message as following:



Step-2 Next, we will install MySQL. MySQL is a database management system used to manage data. To install MySQL, open the terminal and type the following:

sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql

After executing this, you have to set the root password. Once it is done, MySQL installation is finished. But to start MySQL working, it is to be activated by following command:

sudo mysql_install_db

Finish the installation by running the MySQL set up script:

sudo /usr/bin/mysql_secure_installation

MySQL installation is complete now. But you are asked to enter the root password. Enter the correct password and agree to all following steps by typing 'Y'. Once, you do it, you are done with MySQL installation.

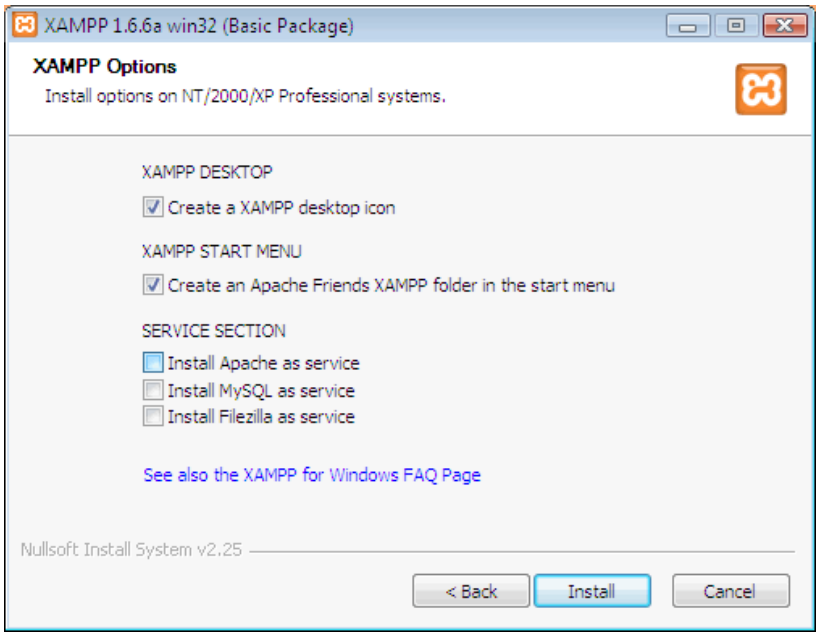
Step-3 PHP comes with a variety of useful libraries and modules that you can add. You can see all the available libraries by following command:

apt-cache search php5-
A list is displayed on terminal that looks like below:
php5-cgi - server-side, HTML-embedded scripting language (CGI binary)
php5-cli - command-line interpreter for the php5 scripting language
php5-common - Common files for packages built from the php5 source
php5-curl - CURL module for php5
php5-dbg - Debug symbols for PHP5
php5-dev - Files for PHP5 module development
php5-gd - GD module for php5
php5-gmp - GMP module for php5
php5-ldap - LDAP module for php5
[...]
Decide the module you want to install and type:
sudo apt-get install name of the module
Multiple libraries can be installed at once by separating the name of each module with a space.
And with this, installation of LAMP is complete.

Installing XAMPP on Windows
In XAMPP, 'X' stands for any operating system, either Windows or Linux. So,
XAMPP is for Cross platform, Apache, MySQL, PHP, Perl. It can be installed with following steps:

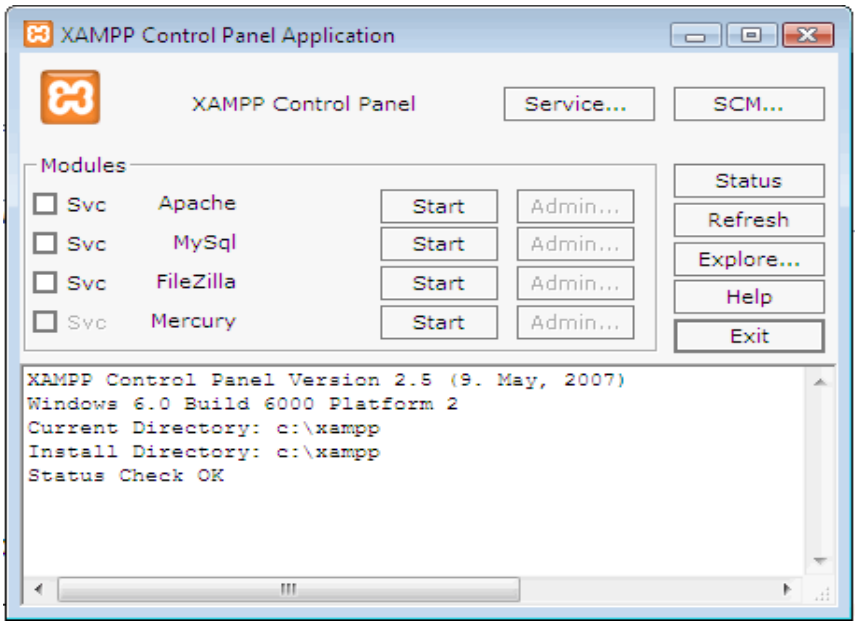
Step-1 Download the source file for installing XAMPP from its website. Once downloaded, start installing it by double-clicking the file.

Step-2 Choose a language from the menu and then click 'OK'. Click on 'Next' in next screens. The following screen appears.



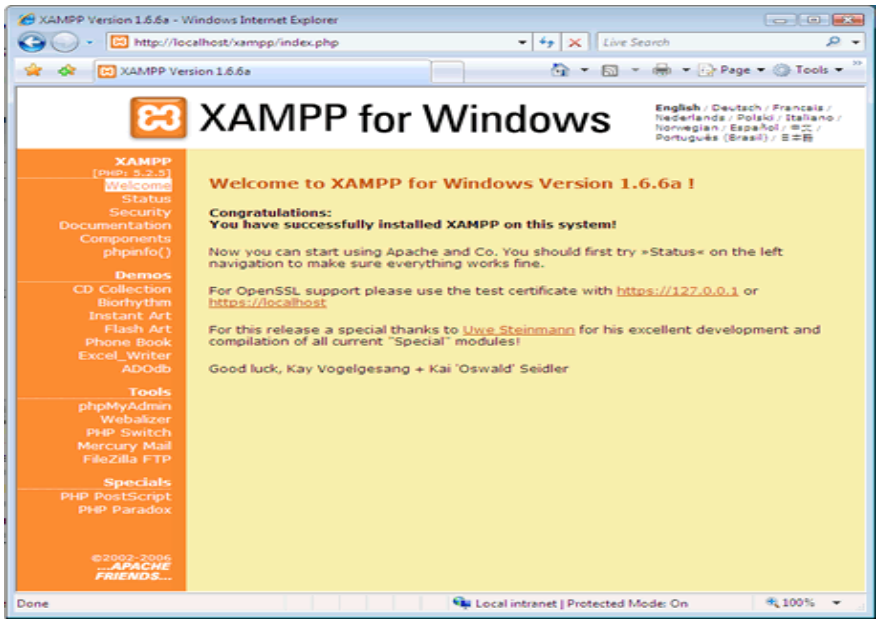
Step-3 Click on 'Install'. When installation is finished, click on 'Finish' to finish the installation.

Step-4 Once done, start the control panel of XAMPP. The following screen appears.



Step-5 If the buttons to the right of Apache & MySQL are visible, click them. This starts the Apache web server and MySQL database server.

Step-6 To confirm the successful installation, open the browser and type http://localhost in address bar.



The above screen confirms the successful installation.

PHP Syntax:

Start and End Tags

PHP is integrated with HTML code to generate dynamic web pages. This is done by using one of the following start and end tags:

1) *<?php*
//php code here
?>

2) *<script language = "php">*
//php code here
</script>

3) *<?*
//php code here
?>

4) *<%*
//php code here
%>

Among these styles first one is widely used. Third and forth are depends on environment.

Comments

Comments are not read and executed by web server. They are added with the code to specify the information about the document such as author name, purpose of writing the document etc.

There are Two types of comments:

a) Single Line
Single line comments are made with '//' or '#'

b) Multiline
Multiline comments start with '/*' and end with '*/'.

Example:

```
<?php
   // This is Single line comment
  # This is also a Single line comment

  /*
     This is multiline comment, that
     spans over multiple lines
  */
?>
```

Case Sensitivity

In PHP, some constructs are case-sensitive while some are not.

In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.

All functions, classes and keywords, whether built-in or user-defined, are NOT case-sensitive. Whereas, variables, constants, array keys, class properties are case-sensitive.

Example:

```
<?php
echo "Hello PHP<br>";
Echo "Hello PHP<br>";
ECHO "Hello PHP";
?>
```

Output:

Hello PHP

Hello PHP

Hello PHP

Note: However; all variable names are case-sensitive.

Look at the example below; only the first statement will display the value of the $var variable! This is because $var, $Var, and $VAR are treated as three different variables:

Example:

```
<?php
    $var = 'Nice';
    echo 'PHP is '. $var;
    echo 'PHP is '. $Var;
    echo 'PHP is '. $VAR;
?>
```

Output:

PHP is Nice

PHP is

PHP is

PHP Output statements

There are two basic statements in PHP for output:

- echo
- print

PHP echo statement can output one or more strings by using only once, while print statement can output only one string. PHP print statement always returns integer 1 value, whereas echo does not return any value at all. Because of this reason, echo statement is little faster than print.

Example:

```
<?php

echo "Hello World<br>";

echo 'Single quotes can also be used<br>';

echo "This", "is", "how ", "multiple strings", "can be displayed<br>";

print "This is the use of print statement.";

?>
```

Output:

Hello World

Single quotes can also be used

This is how multiple strings can be displayed

This is the use of print statement.

Instruction Separation

Each PHP instruction must be terminated with a semicolon at the end of each statement.

Example:

```
<?php
        echo "This is a statement";
        echo "This is another one"; echo "One more statement separated by semicolon";
?>
```

Embedding PHP with HTML

HTML is a mark-up language which can create static web pages only. PHP must be embedded (mixed) with HTML to make the page dynamic. It is done with the help of PHP start and end tag as shown below:

```
<html>
    <head>
        <title>Embedding PHP</title>
    </head>
    <body>
    <h3>Here is HTML</h3>
    <?php
        echo "Hello World!!<br>";//PHP code starts here
    ?>
    <b>PHP code ended and HTML started</b><br>
    <?php
        echo "Again php code<br>"//PHP code starts again and HTML ends
    ?>
    <p>More HTML code</p>
    </body>
</html>
```

Output:

Here is HTML

Hello World!!
PHP code ended and HTML started
Again php code

More HTML code

How PHP Works

PHP file is having an extension '.php'. It consists of mixture of code of HTML and PHP as discussed in previous topic. PHP is server-side scripting language and hence it is executed by server. On the other hand, HTML code is executed by browser. So, after the execution of PHP code on server, the result is returned to the browser as plain HTML. This pure HTML code is executed by browser and result is displayed to the user.

Consider the following PHP code:

```
<?php
    echo "This line is executed by server as it is part of PHP";
?>
<b>But this line is sent to the browser and browser executes it as it is part of HTML</b>
<?php echo "Again this part is executed by server"; ?>
```

Variables in PHP

Variables are the containers for data. Variables in PHP have some significant properties. They are discussed below.

- Variables are case-sensitive as described above.
- Variables are preceded with '$' sign.
- Variable naming rules are same as in C. $number, $cust_name, $student123 are all valid variable names.
- Remember, variable names are number, cust_name and student123. So they should not be referred as $number (DolorNumber) or $student123 (DolorStudent123).

The purpose of adding '$' sign is to inform the PHP interpreter about variable. PHP interpreter treats all the literals preceded with '$' as variables.

Declaring and initializing the variables

In PHP, variables do not need to be declared. You can directly initialize them by assigning them some value. PHP variables are automatically converted to the correct data type, depending on its value.

For example, $number = 100 and $name = 'php' are valid variable initializations.

Displaying the variable value

Variable value can be displayed by using echo statement.

Example:
<?php *$lang = 'PHP';* *echo "I love ".$lang;* *?>*
Output:
I love PHP

Here, '.' is a string concatenation operator. Variable values can also be displayed by putting them into double quotes, but single quotes will not work for this purpose.

Example:
<?php *$lang = 'PHP';* *echo "I love $lang ";* *echo 'I love $lang';* *?>*
Output:
I love PHP *I love $lang*

As you can see from example, value of a variable inside double quotes is parsed, while inside single quotes it is not parsed.

PHP Constants

Constants are just like variables, which are used to store data. But, their value cannot be changed or undefined. Constant name starts with a letter or an underscore. Dolor ($) sign is not added before constant name like in variable name we do.
Note: Unlike variables, constants are automatically global across the entire script.

To create a constant, use the define() function.
Syntax:
define(name, value, case-insensitive)
Parameters:
name: Specifies the name of the constant
value: Specifies the value of the constant
case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false

Example:

```
<?php
define ("message", "Nice PHP Language");
echo message; //outputs Nice PHP Language
?>
```

We can define case-sensitive constant by passing 'true' as a third parameter to define() function. By default, its value is false.

Example:

```
<?php
define("MESSAGE", "Nice PHP Language", true);
echo message; //outputs Nice PHP Language
?>
```

Data types in PHP:

There are Seven types of data in PHP: Integer, Floating point number, String, Boolean, Array, Object, NULL.

1) Integer
2) Floating point number
3) String
4) Boolean
5) Array
6) Object
7) NULL

1) Integer

Integer is a number without decimals.
For Example,

```
<?php
$no1 = 4343; $no2 = -2345;
echo $no1.'<br>'.$no2;
?>
```

Output

```
4343
-2345
```

2) Floating point number

It is a number with a decimal point or a number in exponential form.
For example,

```
<?php
$a = 9.007;
echo $a.'<br>';
$a = 4.3e3;
echo $a;
?>
```

Output

```
9.007
4300
```

3) String

A string data type is a sequence of characters. It is a text between single quotes or double quotes.
For example,

```
<?php
$str = 'PHP';
echo $str.',';
echo $str;
?>
```

Output

```
PHP, PHP
```

4) Boolean

Value of Boolean data type can either be TRUE or FALSE. It is used in conditional testing.
For example,

```
<?php
$a = true;
$b = false;
?>
```

5) Array

An array is a variable used to store multiple values. It is created using array( ) function, which will be discussed in detail in next section.

An array stores multiple values in one single variable.
In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

For example,

```
<?php
$cars = array("Volvo","BMW","Toyota");
echo $cars."<br>";
echo $cars[2]."<br>";
echo $cars[0]."<br>";
var_dump($cars);
?>
```

Output

```
Array
Toyota
Volvo
array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW" [2]=> string(6) "Toyota" }
```

Here, we have created an array named ‘fruit’ of three elements.

6) Object

An object is a data type that stores data and information on how to process that data.
In PHP, an object must be explicitly declared.

We must declare a class of object. For this, we use the class keyword. Then data type in the object class is defined, and then data type is used in instances of that class.

7) NULL

When variable has no value, it contains NULL value. Its only possible value is NULL itself.
Variables can be emptied by setting their value to NULL.
Null is a special data type which can have only one value: NULL.
A variable of data type NULL is a variable that has no value assigned to it.
Tip: If a variable is created without a value, it is automatically assigned a value of NULL.
Variables can also be emptied by setting the value to NULL:

For example,

```
<?php
$str = ‘Hello’;
$str = null;
?>
```

Trying to output the value of $str will not display anything as its value has been set to null.

Type casting in PHP

In PHP, type-casting is done automatically.
For example,

```
<?php
$digit = “100”;
$add = 100 + $digit;
echo $add;
?>
```

This will display the value 200. Here, string type is automatically converted to integer.
Though, PHP allows and supports manual casting, too.
For example,

```
<?php

$digit = “20”;
$val = 30 + $digit;
$val = (string)$val;

?>
```

This will convert the value of $val to string from integer and assign it to $val again.

Changing the type of a variable
We can change the value of a variable and set a new value for it by using settype() function.

Example:

```
<?php

$var = 123;
var_dump($var);
echo "<br>";
settype($var,"string");
var_dump($var);

?>
```

Output:

```
int(123)

string(3) "123"
```

Here, data type of $vat is converted from integer to string. ‘var_dump’ is a function that dumps (displays) every small information of a variable passed as an argument to it.

PHP Operators:

Arithmetic Operators

Arithmetic operators are used to perform mathematical operations. There are five such operators namely, Addition (+), Subtraction (-), Multiplication (*), Division (/) , Modulo (%) and Exponentiation(**). Division operator returns the quotient and modulo operator returns the remainder after division.

Operators	Meaning	Example	Result
+	Addition	4 + 2	6
–	Subtraction	4 – 2	2
*	Multiplication	4 * 2	8
/	Division	4 / 2	2
%	Modulus operator to get remainder in integer division	5 % 2	1
**	Exponent	5**2 = 5^2	25

Example:

```
<?php

$x = 26;
$y = 8;

echo ($x + $y);//outputs 34
echo ($x - $y); //outputs 18
echo ($x * $y); //outputs 208
echo ($x / $y); //outputs 3.25
echo ($x % $y); //outputs 2

echo 3**2; //outputs  9
?>
```

Assignment Operator:

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Operator	Example	Equivalent Expression (m=15)	Result
=	y = a+b	y = 10 + 20	30
+=	m +=10	m = m+10	25
-=	m -=10	m = m-10	5
*=	m *=10	m = m*10	150
/=	m /=10	m = m/10	1.5
%=	m %=10	m = m%10	5

Exmaple:

```
<?php

$a = 10;
echo $a;//outputs 10
echo $a += 5 //outputs 15
echo $a *= 5 //outputs 75

?>
```

String Operators

There is a string concatenation operator (.) used to join two or more strings.

Sequence	When	Example
""	Single variable	$myString = "Adding single variable $string to this string";
.	Single line	$myString = 'Combining multiple ' . $strings . ' on one line';
.=	Multiple lines	$myString = 'Combining multiple lines '; $myString .= $strings; $myString .= ' by using the assignment operator';

Example:

```
<?php

echo "Hello "."PHP"; //Outputs Hello PHP

$var = "Hello ";

echo $var."PHP"; //Outputs Hello PHP

$var .= "PHP";

echo $var;//Outputs Hello PHP ?>
```

Increment/Decrement Operators

Increment operator (++) adds 'one' to the current value of variable. Decrement operator (--) subtracts 'one' from value of the variable. There are two types for each type. Pre-increment increments the value of a variable and then returns it, while Post-increment returns the value first and then increments it. Same, applies to decrement operator.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
–$x	Pre-decrement	Decrements $x by one, then returns $x
$x–	Post-decrement	Returns $x, then decrements $x by one

Example:

```
<?php

$a = 10;
echo $a++;//Outputs 10
echo ++$a;//Outputs 12,( since, 10 is incremented to 11 in previous statement)
echo $a--;//Outputs 12
echo --$a;//Outputs 10, (since, 12 is decremented to 11 in previous statement)

?>
```

Comparison Operators

Comparison operators are used to compare two values.

Comparison Operators

Example	Name	Result
$a == $b	Equal	**TRUE** if *$a* is equal to *$b* after type juggling.
$a === $b	Identical	**TRUE** if *$a* is equal to *$b*, and they are of the same type.
$a != $b	Not equal	**TRUE** if *$a* is not equal to *$b* after type juggling.
$a <> $b	Not equal	**TRUE** if *$a* is not equal to *$b* after type juggling.
$a !== $b	Not identical	**TRUE** if *$a* is not equal to *$b*, or they are not of the same type.
$a < $b	Less than	**TRUE** if *$a* is strictly less than *$b*.
$a > $b	Greater than	**TRUE** if *$a* is strictly greater than *$b*.
$a <= $b	Less than or equal to	**TRUE** if *$a* is less than or equal to *$b*.
$a >= $b	Greater than or equal to	**TRUE** if *$a* is greater than or equal to *$b*.

Example:

```
<?php

$x=100;
$y="100";
var_dump($x == $y); // returns true because values are equal
echo "<br>";
var_dump($x === $y); // returns false because types are not equal
echo "<br>";
var_dump($x != $y); // returns false because values are equal
echo "<br>";
var_dump($x !== $y); // returns true because types are not equal
echo "<br>";
$a=50;
$b=90;
var_dump($a > $b);
echo "<br>";
var_dump($a <  $b);

?>
```

Output:

```
bool(true)
bool(false)
bool(false)
bool(true)
bool(false)
bool(true)
```

Logical Operators

AND (&&/and), OR (||/or), XOR (xor) and NOT (!) are logical operators.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

Example:

```
<?php
$a = 10;
$b = 20;
if($a>0 && $b<20){//returns false
}
if($a==10 && $b==20){//returns true
}
if($a>0 || $b<20){//returns true
}
?>
```

Array Operators

Array operators are used to compare arrays and for performing union operation. They are as per listed below.

Let us say than $a and $b are two different arrays then,

Union (+): Union of $a and $b

Equality (= =): True only if $a and $b have the same key-value pairs

Inequality (! = / < >): True if $a and $b are not equal

Identity (= = =): True only if $a and $b have the same key-value pairs in the same order and of same types.

Non-Identity (! = =): True if $a and $b are not identical.

Example:

```
<?php

$x = array("a" => "red", "b" => "green");
$y = array("c" => "blue", "d" => "yellow");
$z = $x + $y; // union of $x and $y
var_dump($z);
echo "<br>";
var_dump($x == $y);
echo "<br>";
var_dump($x === $y);
echo "<br>";
var_dump($x != $y);
echo "<br>";
var_dump($x <> $y);
echo "<br>";
var_dump($x !== $y);

?>
```

Output:
array(4) { ["a"]=> string(3) "red" ["b"]=> string(5) "green" ["c"]=> string(4) "blue" ["d"]=> string(6) "yellow" } *bool(false)* *bool(false)* *bool(true)* *bool(true)* *bool(true)*

Operator Precedence:

The precedence of an operator specifies how "tightly" it binds two expressions together. For example, in the expression 1 + 5 * 3, the answer is 16 and not 18 because the multiplication ("*") operator has a higher precedence than the addition ("+") operator. Parentheses may be used to force precedence, if necessary. For instance: (1 + 5) * 3 evaluates to 18.

When operators have equal precedence, their associativity decides how the operators are grouped. For example, "-" is left-associative, so 1 - 2 - 3 is grouped as (1 - 2) - 3 and evaluates to -4. "=" on the other hand, is right-associative, so $a = $b = $c is grouped as $a = ($b = $c).

Operators of equal precedence that are non-associative cannot be used next to each other, for example 1 < 2 > 1 is illegal in PHP. The expression 1 <= 1 == 1 on the other hand is legal, because the == operator has lesser precedence than the <= operator.

Associativity	**Operators**	**Additional Information**
non-associative	*clone new*	clone and new
Left	*[*	array()
Left	**	arithmetic
Right	++ -- *~ (int) (float) (string) (array) (object) (bool) @*	types and increment/decrement
non-associative	*Instanceof*	types
Right	*!*	logical
Left	** / %*	arithmetic
Left	+ - .	arithmetic and string
Left	<< >>	bitwise
non-associative	< <= > >=	comparison
non-associative	== *!*= === *!*== <>	comparison
Left	&	bitwise and references
Left	^	bitwise

Left	|	bitwise
Left	&&	logical
Left	||	logical
Left	? :	ternary
Right	= += -= *= /= .= %= &= |= ^= <<= >>= =>	assignment
Left	And	logical
Left	Xor	Logical
Left	Or	Logical

Flow Control Statements:

Flow control statements are used to control the flow of execution of code. PHP supports all the flow control statements available in C language. And fortunately, syntax is also C-like.

The if statement

The if statement is used to force some code to be executed only if some condition is true.

For example,

```
<?php
if($age>58)
{
   echo "You are retired";
}
?>
```

The above code will output the message if value of $age is greater than 58.

The if…else statement

The if…else statement is used to execute some fraction of code only if some condition is true else other code is executed.
For example,

```
<?php
$age=47;
if($age>58){
   echo "You are retired";
}
else{
   echo "Keep Working";
}
?>
```

The above code will output the message 'You are retired', if value of $age is greater than 58, else 'Keep Working'.

The else if clause:

The else if clause is used to execute code based on multiple conditions.

For example,

```
<?php
$grade=52;
if($grade >=66.66)
{
   echo "Distinction";
}
else if($grade<66.66 && $grade>=60){
   echo "First Class";
}
else if($grade<60 && $grade>=40){
   echo "Pass Class";
}
else{
   echo "Fail";
}
?>
```

The above code will test the multiple conditions and output the message depending upon the value of $grade.

The switch statement

The switch statement works like else if clause, but the difference is that, it can check only the value of a variable or an expression. Robust conditions like in else if clause cannot be executed.

Example:

```
<?php
switch($day)
{
case 1:
echo "It's Sunday";
break;
case 2:
echo "It's Monday";
break;
case 3:
echo "It's Tuesday";
break;
case 4:
```

```
echo "It's Wednesday";
break;
case 5:
echo "It's Thursday";
break;
case 6:
echo "It's Friday";
break;
default:
echo "It's Saturday";
}
?>
```

PHP Loops:

PHP loops are used when we want to execute the same code again and again till some condition is true.

The while Loop

In while loop, some condition is tested first. The code inside while block only gets executed when condition is true. After executing while block, control again goes back to check the condition. This is kept repeating until condition is false.

Example:

```
<?php
$no = 1;
while($no<=5){
echo $no.'<br>';
$no++;
}
?>
```

Output:

```
1
2
3
4
5
```

The do-while Loop

In do-while loop, code is executed first. Then after condition is checked. If condition is true, control goes back to execute the code in do-while block. This is kept repeating until condition is false.

Example:
<?php *$no = 1;* *do {* *echo $no.' ';* *$no++;* *} while ($no<=5);* *?>*
Output:
1 *2* *3* *4* *5*

The for Loop

In for loop, code is executed after variable initialization and condition checking. If condition is true, control executes the code inside for block. After that, control again goes to check the condition. This is kept repeating until condition is false.

Example:
<?php *for($no = 1;$no<=4;$no++){* *echo $no.' ';* *}* *?>*
Output:
1 *2* *3* *4*

The foreach Loop

The foreach loop is used to access the array keys and values. It has been discussed in next section.

Loops can be nested too in PHP as in C language.

Syntax:

```
foreach ($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.

Examples:
The following example will output the values of the given array ($colors):

PHP break and continue statements

In PHP, break statement is used to break the current loop and execute the further code.

Example:

```
<?php
for ($no = 1; $no <=5; $no++){
   echo $no.'<br>';
   if ($no==3){
      break;
   }
}
echo 'Outside of for loop';
?>
```

Output:

```
1
2
3
```

Sometimes, we want to take the control to the beginning of the loop (for example for, while, do while etc.) skipping the rest statements inside the loop which have not yet been executed.

The keyword 'continue' allows us to do this. When 'continue' is executed inside a loop the control is passes to the beginning of loop.

Example:

```
<?php
$x=1;
while ($x<=10)
{
        if (($x % 2)==0)  {
                $x++;
                continue;
        }
        else  {
                echo $x.'<br />';
                $x++;
        }
}
```

Output:
1 *3* *5* *7* *9*

Here, we see that if number is even, its value is incremented and loop continues with next iteration.

By this, we end this unit. In this unit we covered basics of PHP scripting language. In next unit we will see advanced concepts of PHP.

CHAPTER 2- Arrays and User defined Functions

PHP Arrays:

What is an Array?
An array is a data structure that stores one or more similar type of values in a single value. An array stores multiple values in one single variable.
An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is to create an array!
An array can hold many values under a single name, and you can access the values by referring to an index number.

Defining an Array:

Array is defined by array( ) function.

Syntax:

```
$array_name = array(key1=>value1, key2=>value2, key3=>value3… );
```

Here, $array_name is the name of array variable. Values are mandatory to be specified, while, keys are optional in array definition. This concept has been discussed later.

Types of Array:

There are three different kind of arrays. They differ in terms of keys they use.

- **Numeric array** - An array with a numeric index. They are same as C language arrays.
- **Associative array** - An array with strings as index. They contain numeric keys as well as string keys.
- **Multidimensional array** - An array containing one or more arrays and values are accessed using multiple indices.

Numeric Arrays

In numeric arrays, index is represented by numbers.

Example:

```
<?php
$subject = array('dwd', 'cns', 'java');
$marks = array(65,60,55);
var_dump($subject);
var_dump($marks);
?>
```

Output:

```
Array
  0 => string ''dwd'' (length=5)
  1 => string ''cns'' (length=5)
  2 => string ''java'' (length=6)
array
  0 => int 65
  1 => int 60
  2 => int 55
```

As, we can see from the example above, array function automatically assigns numeric keys starting from 0 in incremental order when keys are not defined.

Associative Arrays

Associative arrays may contain either digits or strings as keys.

Example:

```
<?php

$sub_marks = array('dwd'=>65, 'cns'=>60, 'java'=>55);
$test = array('one'=>1, 2=>'two', '3'=>'three');
var_dump($sub_marks);
var_dump($test);

?>
```

Output:

```
Array
  ''dwd'' => int 65
  ''cns'' => int 60
  ''java'' => int 55
array
  ''one'' => int 1
  2 => string ''two'' (length=5)
  ''3'' => string ''three'' (length=7)
```

If key is not specified, it is considered as integer by default in incremental order. This has been explained in below example.

Example:

```
<?php
        $test = array('one'=>1, 2, '3'=>'three', 4, 5=>'five',6);
        var_dump($test);
?>
```

Output:

```
Array
  ''one'' => int 1
  0 => int 2
  ''3'' => string ''three'' (length=7)
  1 => int 4
  5 => string ''five'' (length=6)
  6 => int 6
```

Now, you can have the idea that, index is automatically started from the one digit more than the previous integer index.

Multidimensional Arrays

Multidimensional arrays contain another array inside array.

Example:

```
<?php
  $marks = array( "sachin" => array
                    ("dwd" => 55,
                    "cns" => 50,
                    "java" => 49),
          "kartik" => array
                    ("dwd" => 60,
                     "cns" => 62,
                     "java" => 39),
           "nupur" => array
                    ("dwd" => 51,
                    "cns" => 42,
                    "java" => 39)
              );
  /* Accessing multi-dimensional array values */
  echo "Marks for Sachin in dwd : " ;
  echo $marks['sachin']['dwd'] . "<br />";
  echo "Marks for Kartik in cns : ";
  echo $marks['kartik']['cns'] . "<br />";
  echo "Marks for Nupur in java : " ;
  echo $marks['nupur']['java'] . "<br />";
?>
```

Output:
Marks for Sachin in dwd : 55 *Marks for Kartik in cns : 62* *Marks for Nupur in java : 39*

Adding Elements to Array

Once the array is declared, it is possible to insert the elements into it. PHP allows this by array_push( ) function.

Example:
<?php *$arr = array('bill', 'steve');* *array_push('mark', 'larry', 'sergey');* *var_dump($arr);* *?>*
Output:
Array *0 => string ''bill'' (length=6)* *1 => string ''steve'' (length=7)* *array* *0 => string ''bill'' (length=6)* *1 => string ''steve'' (length=7)* *2 => string ''mark'' (length=6)* *3 => string ''larry'' (length=7)* *4 => string ''sergey'' (length=8)*

Using FOREACH Loop

Once the array is defined, we can traverse through it using foreach loop. It provides a convenient way to access each element of an array with its key as well as value.

Syntax:

There are two different syntax for traversing through an array.

```
foreach ($array_name as $value)
{
//foreach block
}
```

Above syntax is used to access only values of each element. In every iteration, subsequent values of each element are assigned to $value variable automatically. Then we can use this value.

If we want to access keys along with values, below syntax is used.

```
foreach ($array_name as $key => $value)
{
//foreach block
}
```

This syntax works similarly as earlier one, but the difference is that along with values, keys are also stored in $key variable for each iteration.

Example:

```
<?php
$arr = array('one', 'two'=>2, 3=>'3', 4, '5'=>'five');
foreach($arr as $value)
{
        echo $value.'<br>';
}
?>
<br /><br />
<?php
foreach($arr as $key => $value)
{
        echo 'The key is '.$key.' and value is '.$value.'<br>';
}
?>
```

Output:

```
One
2
3
4
Five
The key is 0 and value is one
The key is two and value is 2
The key is 3 and value is 3
The key is 4 and value is 4
The key is 5 and value is five
```

User-defined Functions:

We can define user-defined functions in PHP. A function is a block of statements that can be used repeatedly in a program. A function will be executed by a call to the function.

Syntax:

```
function functionName (optional_arguements)
{
//function body
}
```

A function name can start with a letter or underscore (not a number). Function names are NOT case-sensitive.

Once the function is defined, it can be called by writing its name and providing arguments if any.

Example:

```
<?php
function sayHello( )
{
        echo 'Hello';
}
sayHello( ); //calling a function
?>
```

Output:

```
Hello
```

Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just seperate them with a comma.

Example:

```
<?php
function familyName($fname)
{
echo "$fname Patel.<br>";
}

familyName("Meet");
familyName("Bob");
familyName("Raj");
?>
```

Output:

```
Meet Patel
Bob Patel
Raj Patel
```

Example:

```
<?php

function familyName($fname, $lname)
{
echo "$fname $lname<br>";
}
familyName("Zara","Patel");
familyName("Bob","Patel");

?>
```

Output:

```
Zara Patel
Bob Patel
```

Default Arguments

Default arguments can be passed in PHP. Default arguments are the arguments which are passed when no argument is supplied in function call.

Example:

```
<?php
function makecoffee($type = "cappuccino")
{
        return "Making a cup of $type.\n";
}
echo makecoffee();
echo makecoffee(null);
echo makecoffee("espresso");
?>
```

Output:

```
Making a cup of cappuccino.
Making a cup of .
Making a cup of espresso.
```

All the examples discussed so far, are the passing arguments by value. Arguments can also be passed by reference.

Passing Argument by Reference

By default, function arguments are passed by value (so that if the value of the argument within the function is changed, it does not get changed outside of the function). To allow a function to modify its arguments, they must be passed by reference.

To have an argument to a function always passed by reference, prepend an ampersand (&) to the argument name in the function definition.

Example:

```
<?php

function add_some_extra(&$string)
{
$string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;    // outputs 'This is a string, and something extra.'

?>
```

Returning Values:

We can make functions return values. Below example show that.

Example:

```
<?php

function sum($x,$y)
{
  $z=$x+$y;
  return $z;
}
echo "5 + 10 = " . sum(5,10) . "<br>";
echo "7 + 13 = " . sum(7,13) . "<br>";
echo "2 + 4 = " . sum(2,4);

?>
```

Output:

```
5+ 10 = 15
7+ 13 = 20
2+ 4 = 6
```

CHAPTER 3- Built-in Functions

PHP contains hundreds of built-in functions, which can be used for different purpose. They are categorized in the context of they are used.

Array Functions:

count( ) : The count() function returns the number of elements in an array.

Syntax:

count(*array, mode*)

Parameter	Description
Array	Required. Specifies the array
Mode	Optional. Specifies the mode. Possible values: • 0 - Default. Does not count all elements of multidimensional arrays • 1 - Counts the array recursively (counts all the elements of multidimensional arrays)

Example 1:

```
<?php
$animals=array("Lion","Tiger","Elephant");
echo count($animals);
?>
```

Output:

```
3
```

Example 2:

```
<?php
$cars=array (
  "Volvo"=>array ("XC60", "XC90" ),
  "BMW"=>array ( "X3", "X5" ),
  "Toyota"=>array( "Highlander" )
  );
echo "Normal count: " . count($cars)."<br>";
echo "Recursive count: " . count($cars,1);
?>
```

Output:

```
Normal count: 3
Recursive count: 8
```

sort( ) : Sort an array in ascending order

Syntax:

sort(array &$array, int $flags = SORT_REGULAR) : bool

The optional second parameter flags may be used to modify the sorting behavior using these values:

Sorting type flags:

SORT_REGULAR - compare items normally; the details are described in the comparison operators section

SORT_NUMERIC - compare items numerically

SORT_STRING - compare items as strings

SORT_LOCALE_STRING - compare items as strings, based on the current locale. It uses the locale, which can be changed using setlocale()

SORT_NATURAL - compare items as strings using "natural ordering" like natsort()

SORT_FLAG_CASE - can be combined (bitwise OR) with SORT_STRING or SORT_NATURAL to sort strings case-insensitively

Example:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars,);
$clength = count($cars);
for($x = 0; $x < $clength; $x++) {
  echo $cars[$x];
  echo "<br>";
}
?>
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

Output:

BMW

Toyota

Volvo

array_count_values( ) : Counts all the values of an array

Syntax:

array_count_values(array $array) : array

Example:
<?php *$array = array(1, "hello", 1, "world", "hello");* *print_r(array_count_values($array));* *?>*
Output:
Array([1] => 2 [hello] => 2 [world] => 1)

array_key_exists( ) : Checks if the given key or index exists in the array

Syntax:

array_key_exists(string|int $key, array $array) : bool

Example:
<?php *$search_array = array('first' => 1, 'second' => 4);* *if (array_key_exists('first', $search_array)) {* *echo "The 'first' element is in the array";* *}* *?>*
Output:
The 'first' element is in the array

array_push( ) : Push one or more elements onto the end of array

Syntax:

array_push(array &$array, mixed ...$values) : int

Returns the new number of elements in the array.

Example:

```
<?php

$stack = array("orange", "banana");

array_push($stack, "apple", "raspberry");

print_r($stack);

?>
```

Output:

```
Array

(

    [0] => orange

    [1] => banana

    [2] => apple

    [3] => raspberry

)
```

array_search( ) : Searches the array for a given value and returns the first corresponding key if successful

Syntax:

array_search(mixed $needle, array $haystack, bool $strict = false): int | string | false

Returns the key for needle if it is found in the array, false otherwise.

If needle is found in haystack more than once, the first matching key is returned. To return the keys for all matching values, use array_keys() with the optional search_value parameter instead.

Example:

```
<?php

$array = array(0 => 'blue', 1 => 'red', 2 => 'green', 3 => 'red');
```

```
$key = array_search('green', $array); // $key = 2;

$key = array_search('red', $array);   // $key = 1;

?>
```

array_reverse( ) : Return an array with elements in reverse order

Syntax:

array_reverse(array $array, bool $preserve_keys = false): array

If preserve_keys set to true numeric keys are preserved. Non-numeric keys are not affected by this setting and will always be preserved.

Example:

```
<?php

$a=array("a"=>"Volvo","b"=>"BMW","c"=>"Toyota");

print_r(array_reverse($a));

?>
```

Output:

```
Array ( [c] => Toyota [b] => BMW [a] => Volvo )
```

array_unique( ) : Removes duplicate values from an array

Syntax:

array_unique(array $array, int $flags = SORT_STRING): array

The optional second parameter flags may be used to modify the sorting behavior using these values:

Sorting type flags:

SORT_REGULAR - compare items normally (don't change types)

SORT_NUMERIC - compare items numerically

SORT_STRING - compare items as strings

SORT_LOCALE_STRING - compare items as strings, based on the current locale.

Example:

```
<?php
$input = array("a" => "green", "red", "b" => "green", "blue", "red");
$result = array_unique($input);
print_r($result);
?>
```

Output:

```
Array
(
    [a] => green
    [0] => red
    [1] => blue
)
```

array_values( ) : returns all the values from the array and indexes the array numerically

Syntax:

array_values(array $array): array

Example:

```
<?php
$array = array("size" => "XL", "color" => "gold");
print_r(array_values($array));
?>
```

Output:

```
Array
(
    [0] => XL
```

```
    [1] => gold

)
```

in_array( ) : Checks if a value exists in an array

Syntax:

in_array(mixed $needle, array $haystack, bool $strict = false): bool

Example:

```
<?php

$os = array("Mac", "NT", "Irix", "Linux");

if (in_array("Irix", $os)) {

    echo "Got Irix";

}

if (in_array("mac", $os)) {

    echo "Got mac";

}

?>
```

Output:

```
Got Irix
```

array_walk( ) : Applies the user-defined function to each element of the array.

Syntax:

array_walk(array &$array, callable $callback, mixed $arg = null): bool

Example:

```
<?php

function myfunction($value,$key)

{
```

```
echo "The key $key has the value $value<br>";

}

$a=array("a"=>"red","b"=>"green","c"=>"blue");

array_walk($a,"myfunction");

?>
```

Output:

```
The key a has the value red

The key b has the value green

The key c has the value blue
```

String Functions

explode( ) : Returns an array of strings, each of which is a substring of string formed by splitting it on boundaries formed by the string separator.

Syntax:

explode(string $separator, string $string, int $limit = PHP_INT_MAX) : array

If limit is set and positive, the returned array will contain a maximum of limit elements with the last element containing the rest of string.

If the limit parameter is negative, all components except the last -limit are returned.

If the limit parameter is zero, then this is treated as 1.

Example:

```
<?php

$str = "Hello world. Have a nice day.";

print_r (explode(" ",$str));

?>
```

Output:

```
Array ( [0] => Hello [1] => world. [2] => Have [3] => a [4] => nice[5] => day. )
```

implode( ) : Returns a string containing a string representation of all the array elements in the same order, with the separator string between each element.

Syntax:

implode(array $array, string $separator) : string

Example:
<?php *$arr = array('Hello','World!','Nice','Day!');* *echo implode(" ",$arr);* *?>*
Output:
Hello World! Nice Day!

trim ( ) : Strip whitespace (or other characters) from the beginning and end of a string

Syntax:

trim (string *$str* [, string *$charlist*]) : string

Example:
<?php *$str = "Hello World!";* *echo $str . " ";* *echo trim($str,"Hed!");* *?>*
Output:
Hello World! *llo Worl*

md5 ( ) : Calculates the MD5 hash of string and returns that hash.

Syntax:

md5(string $string, bool $binary = false) : string

Example:
<?php

```
$str = "Hello";

echo md5($str);

?>
```

Output:

8b1a9953c4611296a827abf8c47804d7

str_contains ( ) : Determine if a string contains a given substring

Syntax:

str_contains(string $haystack, string $needle) : bool

Example:

```
<?php

$string = 'The lazy fox jumped over the fence';

if (str_contains($string, 'lazy')) {

    echo "The string 'lazy' was found in the string\n";

}
```

Output:

The string 'lazy' was found in the string

str_replace ( ) : Returns a string or an array with all occurrences of string in main string replaced with the given string.

Syntax:

str_replace (mixed $search , mixed $replace , mixed $subject) : mixed

Example:

```
<?php
 echo str_replace("world","Ankit","Hello world!");
?>
```

Output:

Hello Ankit!

str_split ( ) : Converts a string to an array

Syntax:

str_split(string $string, int $length = 1) : array

Example:
<?php *print_r(str_split("Hello"));* *?>*
Output:
Array ([0] => H [1] => e [2] => l [3] => l [4] => o)

strcmp ( ) : Compare two strings (case-sensitive). Returns 0 if two strings are equal, <0 if string1 is less than string2 and >0 if string1 is greater than string2.

Syntax:

strcmp (string `$str1` , string `$str2`) : int

Example:
<?php *echo strcmp("Hello world!","Hello world!"); // the two strings are equal* *echo strcmp("Hello world!","Hello"); // string1 is greater than string2* *echo strcmp("Hello world!","Hello world! Hello!"); // string1 is less than string2* *?>*
Output:
0 *7* *-7*

strlen ( ) : Returns the length of the string passed.

Syntax:

strlen (string $string) : int

Example:

```
<?php

$len = strlen("Hello WORLD!");

echo  $len;

?>
```

Output:

12

strpos ( ) : Returns the position of the first occurrence of a string inside another string (case-sensitive)

Syntax:

strpos (string `$haystack` , mixed `$needle)  : mixed`

Example:

```
<?php
echo strpos("I love php, I love php too!","php");
?>
```

Output:

7

strtolower ( ) : Returns $str with all alphabetic characters converted to lowercase.

Syntax:

strtolower (string $string) : string

Example:

```
<?php

echo strtolower("Hello WORLD.");

?>
```

Output:

hello world

strtoupper ( ) : Returns $str with all alphabetic characters converted to uppercase.

Syntax:

strtolower (string $string) : string

Example:
<?php echo strtoupper("Hello WORLD!"); ?>
Output:
HELLO WORLD

Date and Time Functions

date( ) : Formats a local date/time.

Syntax:

date (string `$format` [, int `$timestamp` = time()]) : string

Returns a string formatted according to the given format string using the given integer timestamp or the current time if no timestamp is given. In other words, timestamp is optional.

format	d - The day of the month (from 01 to 31) D - A textual representation of a day (three letters) j - The day of the month without leading zeros (1 to 31) l (lowercase 'L') - A full textual representation of a day N - The ISO-8601 numeric representation of a day (1 for Monday through 7 for Sunday) S - The English ordinal suffix for the day of the month (2 characters st, nd, rd or th. Works well with j) w - A numeric representation of the day (0 for Sunday through 6 for Saturday) z - The day of the year (from 0 through 365) W - The ISO-8601 week number of year (weeks starting on Monday) F - A full textual representation of a month (January through December) m - A numeric representation of a month (from 01 to 12) M - A short textual representation of a month (three letters)

	n - A numeric representation of a month, without leading zeros (1 to 12) t - The number of days in the given month L - Whether it's a leap year (1 if it is a leap year, 0 otherwise) o - The ISO-8601 year number Y - A four digit representation of a year y - A two digit representation of a year a - Lowercase am or pm A - Uppercase AM or PM B - Swatch Internet time (000 to 999) g - 12-hour format of an hour (1 to 12) G - 24-hour format of an hour (0 to 23) h - 12-hour format of an hour (01 to 12) H - 24-hour format of an hour (00 to 23) i - Minutes with leading zeros (00 to 59) s - Seconds, with leading zeros (00 to 59) e - The timezone identifier (Examples: UTC, Atlantic/Azores) I (capital i) - Whether the date is in daylights savings time (1 if Daylight Savings Time, 0 otherwise) O - Difference to Greenwich time (GMT) in hours (Example: +0100) T - Timezone setting of the PHP machine (Examples: EST, MDT) Z - Timezone offset in seconds. The offset west of UTC is negative, and the offset east of UTC is positive (-43200 to 43200) c - The ISO-8601 date (e.g. 2004-02-12T15:19:21+00:00) r - The RFC 2822 formatted date (e.g. Thu, 21 Dec 2000 16:01:07 +0200) U - The seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)

timestamp	Optional. This is an integer Unix timestamp that defaults to the current local time if a timestamp is not given. In other words, it defaults to the value of time().

Example:

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

Output:

```
Today is 2014/08/03
Today is 2014.08.03
Today is 2014-08-03
Today is Sunday
```

checkdate( ) : Validate a Gregorian date

Syntax:

checkdate (int *$month* , int *$day* , int *$year*) : bool

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Example:

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

Output:

```
bool(true)

bool(false)
```

date_parse( ): Returns associative array with detailed info about given date/time

Syntax:

date_parse(string $datetime) : array

Example:
<?php *print_r(date_parse("2022-05-01 12:30:45.5"));* *?>*
Output:
Array ([year] => 2022 [month] => 5 [day] => 1 [hour] => 12 [minute] => 30 [second] => 45 [fraction] => 0.5 [warning_count] => 0 [warnings] => Array ( ) [error_count] => 0 [errors] => Array ( ) [is_localtime] =>))

getdate( ): Returns an array that contains date and time information for a Unix timestamp.

Syntax:

getdate ([int `$timestamp` = time()]) : array

Returns an associative array containing the date information of the timestamp, or the current local time if no timestamp is given.

Example:
<?php *$today = getdate();* *print_r($today);* *?>*
Output:
Array *(* *[seconds] => 40* *[minutes] => 58*

```
    [hours]   => 21

    [mday]    => 17

    [wday]    => 2

    [mon]     => 6

    [year]    => 2003

    [yday]    => 167

    [weekday] => Tuesday

    [month]   => June

    [0]       => 1055901520

)
```

time( ): Returns current Unix timestamp

Syntax:

time (void) : int

Example:

```
<?php
$t=time();
echo($t . "<br>");
echo(date("Y-m-d",$t));
?>
```

Output:

```
1407124248
2014-08-03
```

mktime( ) : Returns the Unix timestamp for a date.

Syntax:

mktime(hour, minute, second, month, day, year, is_dst) : int | false

Parameter	Description
hour	Optional. Specifies the hour
minute	Optional. Specifies the minute
second	Optional. Specifies the second
month	Optional. Specifies the numerical month
day	Optional. Specifies the day
year	Optional. Specifies the year.
is_dst	Optional. Parameters always represent a GMT date so is_dst doesn't influence the result.

Example:
<?php *$lastday = mktime(0, 0, 0, 3, 0, 2000);* *echo strftime("Last day in Feb 2000 is: %d\n", $lastday);* *$lastday = mktime(0, 0, 0, 4, -31, 2000);* *echo strftime("Last day in Feb 2000 is: %d", $lastday);* *?>*
Output:
Last day in Feb 2000 is: 29 *Last day in Feb 2000 is: 29*

File Functions

fopen ( ) : Opens file or URL

Syntax:

fopen(string $filename, string $mode, bool $use_include_path = false, resource $context = null): resource | false

mode	Description
'r'	Open for reading only; place the file pointer at the beginning of the file.
'r+'	Open for reading and writing; place the file pointer at the beginning of the file.
'w'	Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
'w+'	Open for reading and writing; otherwise it has the same behavior as 'w'.
'a'	Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it. In this mode, fseek() has no effect, writes are always appended.
'a+'	Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it. In this mode, fseek() only affects the reading position, writes are always appended.
'x'	Create and open for writing only; place the file pointer at the beginning of the file. If the file already exists, the fopen() call will fail by returning false and generating an error of level E_WARNING. If the file does not exist, attempt to create it. This is equivalent to specifying O_EXCL|O_CREAT flags for the underlying open(2) system call.
'x+'	Create and open for reading and writing; otherwise it has the same behavior as 'x'.
'c'	Open the file for writing only. If the file does not exist, it is created. If it exists, it is neither truncated (as opposed to 'w'), nor the call to this function fails (as is the case with 'x'). The file pointer is positioned on the beginning of the file. This may be useful if it's desired to get an advisory lock (see flock()) before attempting to modify the file, as using 'w' could truncate the file before the lock was obtained (if truncation is desired, ftruncate() can be used after the lock is requested).
'c+'	Open the file for reading and writing; otherwise it has the same behavior as 'c'.
'e'	Set close-on-exec flag on the opened file descriptor. Only available in PHP compiled on POSIX.1-2008 conform systems.

Example:

```
<?php

$handle = fopen("/home//file.txt", "r");
```

```
$handle = fopen("/home//file.gif", "wb");

$handle = fopen("http://www.example.com/", "r");

$handle = fopen("ftp://user:password@example.com/somefile.txt", "w");

?>
```

fread ( ) : Binary-safe file read

Syntax:

fread(resource $stream, int $length): string|false

stream - A file system pointer resource that is typically created using fopen().

length - Up to length number of bytes read.

Example:

```
<?php

// get contents of a file into a string

$filename = "/usr/local/something.txt";

$handle = fopen($filename, "r");

$contents = fread($handle, filesize($filename));

fclose($handle);

?>
```

fwrite ( ) : Binary-safe file write

Syntax:

fwrite(resource $stream, string $data, ?int $length = null): int|false

fwrite() writes the contents of data to the file stream pointed to by stream.

Example:

```
<?php
```

```
$file = fopen("test.txt","w");

echo fwrite($file,"Hello World. Testing!");

fclose($file);

?>
```

Output:

21

fclose ( ) : Closes an open file pointer

Syntax:

fclose(resource $stream): bool

Example:

```
< ?php

$handle = fopen('somefile.txt', 'r');

fclose($handle);

?>
```

copy ( ) : Copies file

Syntax:

copy(string $from, string $to, ?resource $context = null): bool

Example:

```
<?php

$file = 'example.txt';

$newfile = 'example.txt.bak';

if (!copy($file, $newfile)) {

    echo "failed to copy $file...\n";

}
```

```
?>
```

file_exists ( ) : Checks whether a file or directory exists

Syntax:

file_exists(string $filename): bool

Example:

```
<?php
$filename = '/path/to/foo.txt';
if (file_exists($filename)) {
    echo "The file $filename exists";
} else {
    echo "The file $filename does not exist";
}
?>
```

file_size ( ) : Gets file size

Syntax:

filesize(string $filename): int | false

Example:

```
<?php
// outputs e.g.  somefile.txt: 1024 bytes
$filename = 'somefile.txt';
echo $filename . ': ' . filesize($filename) . ' bytes';
?>
```

mkdir ( ) : Makes directory

Syntax:

filesize(string $filename): int | false

Example:

```
<?php

// outputs e.g.  somefile.txt: 1024 bytes

$filename = 'somefile.txt';

echo $filename . ': ' . filesize($filename) . ' bytes';

?>
```

rmdir ( ) : Removes directory

Syntax:

rmdir(string $directory, ?resource $context = null): bool

Example:

```
<?php

if (!is_dir('examples')) {

    mkdir('examples');

}

rmdir('examples');

?>
```

unlink ( ) : Deletes a file

Syntax:

unlink(string $filename, ?resource $context = null): bool

Example:

```
<?php
```

```
$fh = fopen('test.html', 'a');

fwrite($fh, '<h1>Hello world!</h1>');

fclose($fh);

unlink('test.html');

?>
```

CHAPTER 4-Session, Cookie and Form Submission in PHP

Input through Form Controls

In PHP, static data can be stored in variables. However, when we want dynamic input from user, we must have some other way. Forms serve this purpose. They contain some elements which we can use to accept dynamic input from users. Form elements are designed by HTML only, but we need to use PHP for making them work.

All form elements are put inside <form> tag. They might not work properly if this is not done.

```
<form>
        All Form Elements Here…
</form>
```

Following are the form elements with necessary properties which are used for accepting different types of input.

Text box: It is used to accept short text input from user.

<input type="text" name="txtName" />

Here, 'type' property is used to create a text box. 'name' property uniquely identifies the element. It works as an ID of the element. Further, it should be given a name so that, we can identify the element and purpose of using it by just knowing a name.

Password Field: It is used to accept password-like text which is not readable when entered.

<input type="password" name="txtPassword" />

Text area: When text input is long e.g. address, we can use this input type.

<textarea name="tareaAddress"></textarea>

Hidden Field: When we want to pass some value along with other form values, this element is used. This element is not shown in browser but the value it contains is submitted like other form elements.

<input type="hidden" value="hiddenvalue" />

Radio Button: Radio button is used for single selection from multiple options e.g. Gender.

<input type="radio" name="rdoGender" value="male" /> MALE

<input type="radio" name="rdoGender" value="female" /> FEMALE

Here, ‘name’ property of all radio buttons from same group must be same. Further, ‘value’ property must be set. The value of this property is passed when user selects the particular option.

Combo box/Dropdown List: When number of options are more in case of single selection, e.g. City, we should use combo box instead of radio button.

```
<select name="cmbCity">
    <option value="ahmedabad"> Ahmedabad </option>
    <option value="vadodara">Vadodara</option>
    <option value="rajkot">Rajkot</option>
    <option value="surat">Surat</option>
</select>
```

Here, ‘option’ tags are used to display different options and the text between them is shown as a caption to user.

Check box: When multiple selection from give options is required, e.g. Hobby, check box is used.

```
<input type="checkbox" name="chkHobby[ ]" value="cricket" /> Cricket
<input type="checkbox" name="chkHobby[ ]" value="reading" /> Reading
<input type="checkbox" name="chkHobby[ ]" value="travelling" /> Travelling
```

‘name’ property is having square brackets. The reason for that is multiple options chosen by user are passed in form of array. If we omit brackets, only single value will be passed. Hence, bracket in ‘name’ property in check box is mandatory.

List box: When options are more in case of multiple selection, we should use List box instead of checkbox.

```
<select name="cmbCity" multiple="multiple">
    <option value="ahmedabad"> Ahmedabad </option>
    <option value="vadodara">Vadodara</option>
    <option value="rajkot">Rajkot</option>
```

```
    <option value="surat">Surat</option>
</select>
```

As you can see, by adding just 'multiple' property into combobox, it is displayed as a list box.

File upload Field: It is used to facilitate the user to upload a file.

<input type="file" name="fileImage" />

In addion to this, following property must be added to <form> to make file uploading work.

<form enctype="multipart/form-data">

By doing this, we can see a file upload button on a webpage.

Submit Button: It is used to submit a form.

<input type="submit" />

Some browsers display 'Submit Query' caption on button. We can use 'value' property to show the caption we want.

<input type="submit" value="Register" />

Reset Button: It is used to reset all the fields in form.

<input type="reset" />

Submitting and Handling a Form

Submitting a Form

Once the form is designed, it is necessary to write code for submitting and handling it. This is achieved by using **Two** files ideally. ONE FILE is used to design and show the entire form to the user. The SECOND FILE is used to receive the data passed from first file and process the received data.

We discussed in previous section that how to design a form in first file. Now, we will see how to pass the information entered by user to second file and how to process that information.

To pass the information to the second file, two properties are used in <form> tag namely 'action' and 'method'. 'Action' property indicates the name of the file to which data is to be passed, whereas 'method' property indicates the method by which the information is to be passed.

There are Two methods by which we can pass the information: GET and POST.

The GET Method

- GET requests can be cached.
- GET requests remain in the browser history.
- GET requests can be bookmarked.
- GET request data is shown in URL.
- GET request is less secure than POST request.
- GET requests should never be used when dealing with sensitive data.
- GET requests have length restrictions.
- GET requests should be used only to retrieve data.

The POST Method

- POST requests are never cached.
- POST requests do not remain in the browser history.
- POST requests cannot be bookmarked.
- POST request data is NOT shown in URL.
- POST request is more secure than GET request.
- POST requests have no restrictions on data length.

Now, let us assume that we have put our code for form in a file named "formDisplay.php" and the file name to which the data is submitted is "formSubmit.php", then we can write above properties in a <form> as shown below:

```
<form method="post" action="formSubmit.php">
        Form Element Code
</form>
```

Accessing Information Submitted by User

Once the form is submitted, the information is passed to second file specified in action property of <form> tag. This information can be accessed using $_POST, $_GET or $_REQUEST. The syntax for this is:

```
<?php
```

$_POST[‘name property of element whose value to be accessed’];//When method used is ‘post’

$_GET[‘name property of element whose value to be accessed’];//When method used is ‘get’

$_REQUEST[‘name property of element whose value to be accessed’];//Irrespective of method used

?>

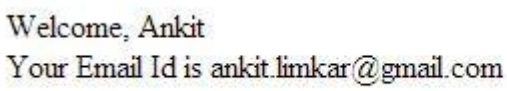
We can understand the concepts discussed so far by example.

Example:

```
//formdesign.php
<!DOCTYPE HTML>
<html>
<body>
<form action="formsubmit.php" method="post">
Name: <input type="text" name="txtName"><br>
E-mail: <input type="text" name="txtEmail"><br>
<input type="submit">
</form>
</body>
</html>
//formsubmit.php
<?php
echo ‘Welcome, ’.$_POST[‘txtName’];
?>
<br>
<?php
echo ‘Your Email Id is ’.$_POST[‘txtEmail’];
?>
```

Above code can be seen as an output in browser as shown below when executed.

Below output can be seen when form is submitted.



Validating User Input

We can validate the user input by using different PHP constructs. Below is an example in which two fields in previous example are checked for required field validation.

```
<?php
if(!$_POST[‘txtName’] || !$_POST[‘txtEmail’]){
echo “Name and Email are required”;
}
else{
echo ‘Welcome, ’.$_POST[‘txtName’];
?>
<br>
<?php
echo ‘Your Email Id is ’.$_POST[‘txtEmail’];
}
?>
```

Here, Name and Email are displayed only if both are entered by user.

Redirecting the User

We can redirect the user to front page (first file) rather than displaying message in back end file (second file). This can be done by header( ) function.

header(“location:path of file to be redirected to”);

Example:

```
<?php

if(!$_POST['txtName'] || !$_POST['txtEmail']){

header("location:formdesign.php"); //Redirects user to formdesign.php

}

?>
```

PHP File Upload

PHP allows you to upload single and multiple files through few lines of code only. PHP file upload features allow you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

PHP $_FILES

The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

Here, we are assuming that file name is *filename*.

$_FILES['filename']['name']

returns file name.

$_FILES['filename']['type']

returns MIME type of the file.

$_FILES['filename']['size']

returns size of the file (in bytes).

$_FILES['filename']['tmp_name']

returns temporary file name of the file which was stored on the server.

$_FILES['filename']['error']

returns error code associated with this file.

move_uploaded_file() function

The move_uploaded_file() function moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.

Syntax

bool move_uploaded_file (string $filename , string $destination)

PHP File Upload Example

HTML file:

```
<form action="uploader.php" method="post" enctype="multipart/form-data">
    Select File:
    <input type="file" name="fileToUpload"/>
    <input type="submit" value="Upload Image" name="submit"/>
</form>
```

uploader.php*:*

```
<?php

$target_path = "e:/";

$target_path = $target_path.basename( $_FILES['fileToUpload']['name']);

if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path)) {

  echo "File uploaded successfully!";

} else{

   echo "Sorry, file not uploaded, please try again!";

}

?>
```

PHP Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer.

There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser. For example, name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

With PHP, you can both create and retrieve cookie values.

Creating a Cookie

Syntax:

setcookie(name, value, expire, path, domain);

- **Name -** This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value -**This sets the value of the named variable and is the content that you actually want to store.
- **Expiry -** This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path -**This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain -** This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security -** This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Example:

```
<?php
setcookie("user", "Rahul Pancholi", time()+3600);
?>
<html>
.....
```

You can also set the expiration time of the cookie in another way. It may be easier than using seconds.

Example:

```
<?php

$expire=time()+60*60*24*30;

setcookie("user", "Alex Porter", $expire);

?>

<html>

.....
```

Accessing Cookie Values

PHP provides many ways to access cookies. Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables.

Example:

```
<?php

echo $_COOKIE["name"]. "<br />";

/* is equivalent to */

echo $HTTP_COOKIE_VARS["name"]. "<br />";



echo $_COOKIE["age"] . "<br />";

/* is equivalent to */

echo $HTTP_COOKIE_VARS["name"] . "<br />";

?>
```

You can use isset() function to check if a cookie is set or not.

Example:

```
<?php

 if( isset($_COOKIE["name"]))

   echo "Welcome " . $_COOKIE["name"] . "<br />";
```

```
  else

    echo "Sorry... Not recognized" . "<br />";

?>
```

Deleting a Cookie

```
<?php

  setcookie( "name", "", time()- 60, "/","", 0);

  setcookie( "age", "", time()- 60, "/","", 0);

?>

<html>

……
```

Session in PHP

A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

Why to Use Sessions

As a website becomes more sophisticated, so must the code that backs it. When you get to a stage where your website needs to pass along user data from one page to another, it might be time to start thinking about using PHP sessions. A normal HTML website will not pass data from one page to another. In other words, all information is forgotten when a new page is loaded. This makes it quite a problem for tasks like a shopping cart, which requires data(the user's selected product) to be remembered from one page to the next.

Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session. This is done using session_start() function.

Example:

```
<?php session_start(); ?>
```

```
<html>

……
```

It should be noted down that this statement should appear before <html> tag.

Storing a Session variable

When you want to store user data in a session use the $_SESSION associative array. This is where you both store and retrieve session data.

Example:

```
<?php

session_start();

$_SESSION['views'] = 1; // store session data

echo "Pageviews = ". $_SESSION['views']; //retrieve data

?>
```

Output:

```
Pageviews = 1
```

By editing above example we can set up a page count which displays the number of times the page has been visited.

Example:

```
<?php

session_start( );

if(isset($_SESSION['views']))

$_SESSION['views']=$_SESSION['views']+1;

else

$_SESSION['views']=1;

echo "This page has been visited ". $_SESSION['views'].” times.”;

?>
```

Here, isset( ) function checks whether session variable has been set to some value or not.

Cleaning and Destroying Session

We can use unset( ) function to destroy single session variable.

Example:

```
<?php

session_start( );

unset($_SESSION['views']);

?>
```

We can unset all the session variables also by session_destroy( ) function.

```
<?php

session_destroy( );

?>
```

CHAPTER 5-Database Operations using PHP

Introduction to MYSQL

With the PHP server side scripting language, you can connect to and view or manipulate data stored in the database tables. You can do so by simply using MYSQL. MYSQL is the most popular Database System used with the PHP.

MYSQL is the most popular Open Source SQL database management system. MYSQL is the relational database management system and this type of systems is fast, reliable and easy to use. It also supports standard SQL and can be suitable for both large and small system. MYSQL is free to download and use. The data in MySQL is stored in tables. A table is a collection of related data, and it consists of columns and rows.

MySQL is a client/server system. There is a database server called as MySQL and randomly many clients known by application programs, which communicate with the server; i.e. they query data, save changes, etc. The clients can run on the same computer as the server or on another computer

For the retrieval of the data from the MySQL database, you must be familiar with how to establish a connection with the MYSQL Database System from the PHP- server side scripting Language.

Performing Database Operations in the PHP-MYSQL is the very simple and easy task, but before performing any of the queries you should be familiar with how to connect to MYSQL Database Management System with the inbuilt functions. It’s very simple compared to other languages.

For Connecting with the Database and performing queries to view and manipulate database the following functions of the MYSQL can be used. Let's get started. The first thing to do is connect to the database.

Data Types of MYSQL

MYSQL Data Types are divided into three main categories as given below:

(1) Numeric or Number Data Type
(2) String or Text Data Type
(3) Date Time Data Type

(1) Numeric or Number Data Type

Data type	Description
NYINT(size)	his data type can be used to store integer values and it ranges from -128 to 127 normal and 0 to 255 is used for UNSIGNED. The maximum number of digits may be specified in parenthesis
MALLINT(size)	his data type can be used to store integer values and it ranges -32768 to 32767 normal and 0 to 65535 is used for UNSIGNED. The maximum number of digits may be specified in parenthesis
EDIUMINT(size)	his data type can be used to store integer values and it ranges from -8388608 to 8388607 normal and 0 to 16777215 UNSIGNED. The maximum number of digits may be specified in parenthesis
IT(size)	his data type can be used to store integer values and it ranges from -2147483648 to 2147483647 normal and 0 to 4294967295 UNSIGNED. The maximum number of digits may be specified in parenthesis
GINT(size)	his data type can be used to store integer values and it ranges from -9223372036854775808 to 9223372036854775807 normal and 0 to 18446744073709551615 UNSIGNED. The maximum number of digits may be specified in parenthesis
LOAT(size,d)	his data type can be used to store single precision floating point value. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
OUBLE(size,d)	his data type can be used to store double precision floating point value. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
ECIMAL(size,d)	his data type can be used to store decimal value. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

(2) String or Text Data Type

Data type	Description
CHAR(size)	This data type is used to store fixed length string. The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	This data type is used to store variable length string. The maximum size is specified in parenthesis. Can store up to 255 characters.
TINYTEXT	This data type is used to store string with a maximum length of 255 characters.
TEXT	This data type is used to store string with a maximum length of 65,535 characters.
BLOB	This data type is used For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data.
MEDIUMTEXT	This data type is used For string with a maximum length of 16,777,215 characters.
MEDIUMBLOB	This data type is used For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	This data type is used For string with a maximum length of 4,294,967,295 characters
LONGBLOB	This data type is used For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(a,b,c,etc.)	This data type lets you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them.You enter the possible values in this format: ENUM('A','B','C')

(3) Date Time Data Type

Data type	Description
DATE()	This data type is used to store date values in the format: YYYY-MM-DD and the supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	This data type is used to store date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59.
TIMESTAMP()	This data type is used to store TIMESTAMP values as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MM:

	SS. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	This data type is used to store time. Format: HH:MM:SS.
YEAR()	This data type is used to store year in two-digit or four-digit format. Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

MYSQL FUNCTIONS

mysqli_connect ():- This Function connects or establishes connection with the MYSQL server from the PHP Script. Prior to doing things of viewing or manipulating data in database table, we must first connect with the MYSQL Server. To do so From the PHP Script we simply use the function mysqli_connect (), which takes three arguments and returns connection on the success and false on failure.

Syntax*:*

$con =mysqli_connect("server","username","password");

Parameter	**Description**
Server	(1) If you are using Local Computer, Use localhost and (2) If you are using Remote Server, use either the hostname or the IP address.
Username	(1) If you are using Local Computer, Use root as the username (2) If you are using Remote Server, use the username you wish to logon.
Password	(1) If you are using Local Computer, Use ' ' as password and (2) If you are using Remote Server, use the password you wish to logon.

Example:

```
<?php

$host ="localhost";

$username="root";

$password="";

$con= mysqli_connect($host,$username,$password);

if (!$con)

{

die("Could not connect Server:" . mysqli_error());

}

else

{

echo "Connected Sucessfully To Server";
}
mysqli_close();
?>
```

The above example demonstrates the use of the mysqli_connect () function, which takes three parameters and returns connection on success and false on failure. To print the error message, use the die function in the block of if statement and check the condition of variable $con.

If the connection is established successfully $con contains true or else it contains false. You can also combine mysqli_error() function to get the exact error message. You can also combine both the statements like die function and mysqli_connect() function in a single statement like this.

Example:

```
<?php

$host ="localhost";

$username="root";

$password="";

$con= mysqli_connect($host,$username,$password) or die("Could not connect Server:" . mysqli_error());

if($con)

        echo "Sucessfully Connected To Server";

mysqli_close();

?>
```

mysqli_select_db() :-After connecting to server, you need to simply select the database. To do so use mysqli_select_db() function which takes only argument that is the name of the database you wish to select for the use and returns true on success and false on failure.

Syntax:

$db=mysqli_select_db("database_name");

Example:

```
<?php

$host ="localhost";

$username="root";

$password="";
```

```
$con= mysqli_connect($host,$username,$password);

 $db = mysqli_select_db("Employee");

if(!$db)

{

        die("Could not Select The Database".mysqli_error());

}

else

{

        echo "Successfully Selected Database";

}

mysqli_close();

?>
```

mysqli_query() :- After selecting the database, you might need to execute queries based on the selected database. To do so, use mysqli_query() function which executes a query on a MySQL selected database. This function takes two arguments one for the query that is to be executed and other for the connection, where the connection part is an optional part. If the link identifier is not specified, the last link opened by mysqli_connect () is assumed.

Syntax:

mysqli_query(String query [,resource link_identifier])

- Here query specifies the SQL query to send and it should not end with a semicolon and the link_identifier is optional part which indicates MYSQL connection.

- For SELECT, SHOW, DESCRIBE and other statements returning resultset, **mysqli_query()** returns a resource on success, or false on error.
- For other type of SQL statements, INSERT, UPDATE, DELETE, DROP, etc, **mysqli_query()** returns `TRUE` on success or `FALSE` on error.
- The returned result resource should be passed to mysqli_fetch_array() or mysqli_fetch_row, and other functions for dealing with result tables, to access the returned data.
- **mysqli_query()** will also fail and return `FALSE` if the user does not have permission to access the table used in the query.

Example:

```
<?php

$con = mysqli_connect("localhost","root","") or die("Could Not Connect Server");

mysqli_select_db("Employee") or   die("Could Not Select Database");

$sql = "select * from emp_details";

$result =mysqli_query($sql,$con);

$row = mysqli_fetch_row($result);

echo $row[0];

echo $row[1];

mysqli_close();

?>
```

mysqli_num_rows():- This functions gets the number of rows in the result set. This function is only valid for statements like SELECT or SHOW that return an actual result set. To retrieve the number of rows affected by a INSERT, UPDATE, REPLACE or DELETE query, use the function mysqli_affected_rows().This function takes one argument that is the resource which comes from the execution of mysqli_query() function. It returns the number of rows in a result set on success or

FALSE on failure.

Syntax:

int mysqli_num_rows (resource $result)

Example:

```
<?php

$con = mysqli_connect("localhost", "root", "");

mysqli_select_db("Employee", $con);

$result = mysqli_query("SELECT * FROM emp_details", $con);

$count = mysqli_num_rows($result);

echo "Number of Rows Returned:- ".$count;

?>
```

mysqli_fetch_array():-This function fetches a result row as an associative array and as a numeric array, or both. This function gets a row from the mysqli_query() function and returns an array on success, or false on failure or when there are no more rows. It takes two arguments; former is the result and later is to determine what kind of array to return.

Syntax:

array mysqli_fetch_array (resource $result [, int $result_type = MYSQL_BOTH])

- In above function first argument is the result resource that is being evaluated. This result comes by executing mysqli_query() function. The second argument is the result type which indicates the type of array to be fetched. It should be constant and can take the following values: `MYSQL_ASSOC`, `MYSQL_NUM`, and `MYSQL_BOTH`.
- MYSQL_ASSOC - Associative Array –you will get Associative Indices.
- MYSQL_NUM - Numeric Array-You will get Numeric Indices.

- MYSQL_BOTH - Default. You will get both Associative and Numeric array indices.

- Associative index means you can use the name of the column as per in the database table. Numeric index means you can use the column number as the index, and both means you can use any one of two or both.
- In Numeric Index, index starts with zero and not with one. Means column 1 in the database table can be retrieved as index zero in result set.
- After the data is retrieved, this function moves to the next row in the record set. Each subsequent call to mysqli_fetch_array() returns the next row in the record set. Field names returned by this function are case-sensitive in case of associative array.

Example:

```
<?php

mysqli_connect("localhost", "root", "");

mysqli_select_db("Employee");

$result = mysqli_query ("SELECT * FROM emp_details");

while ($row = mysqli_fetch_array($result, MYSQL_BOTH))

{

printf ("ID: %s  Name: %s", $row[0], $row["name"]);

}

mysqli_close();

?>
```

mysqli_fetch_row():-The mysqli_fetch_row() function is used to fetch a row of data from a result handle. It takes one argument that is resource which comes from the result as the execution of the mysqli_query(). It returns a numerical array that corresponds to the fetched row and moves the internal data pointer ahead and false if there are no more rows.

Syntax:

array **mysqli_fetch_row** (resource `$result`)

- In above function argument is the result resource that is being evaluated. This result comes by executing mysqli_query() function.
- **mysqli_fetch_row()** fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Example:

```
<?php

mysqli_connect("localhost", "root", "");

mysqli_select_db("Employee");

$result = mysqli_query("select * from emp_details");

$row = mysqli_fetch_row($result);

echo $row[0];

echo $row[1];

mysqli_close();

?>
```

mysqli_close():**-** This function closes the MYSQL connection that is currently open. It takes one argument that is link identifier returned by mysqli_connect() function and it is an optional part. if the link_identifier is not specified then last opened link is used. It returns true on success and false on failure.

Syntax:

bool mysqli_close ([resource $link_identifier = NULL])

Example:

```
<?php

$con = mysqli_connect ("localhost","root","");

$r =mysqli_close($con);

if($r)

echo " Sucessfully Closed";

mysqli_close();

?>
```

mysqli_error() : -If anything fails and we are not able to sort out ,exactly what's the error, we can use the mysqli_error() function. This function is used to get the error message from the last MySQL operation. Link_identifier specifies the MySQL connection and its an optional part, if you don't specify, the last link opened by mysqli_connect() is assumed and it returns the error text from the last MYSQL function or empty string if no error has occurred.

Syntax:

string mysqli_error ([resource $link_identifier = NULL])

Example:

```
<?php

$host ="localhost";

$username="root";

$password="";
```

```
$con= mysqli_connect($host,$username,$password) or die("Could not connect Server:" . mysqli_error());

$db = mysqli_select_db("Employee") or die("Could not select Database".mysqli_error());

mysqli_close();

?>
```

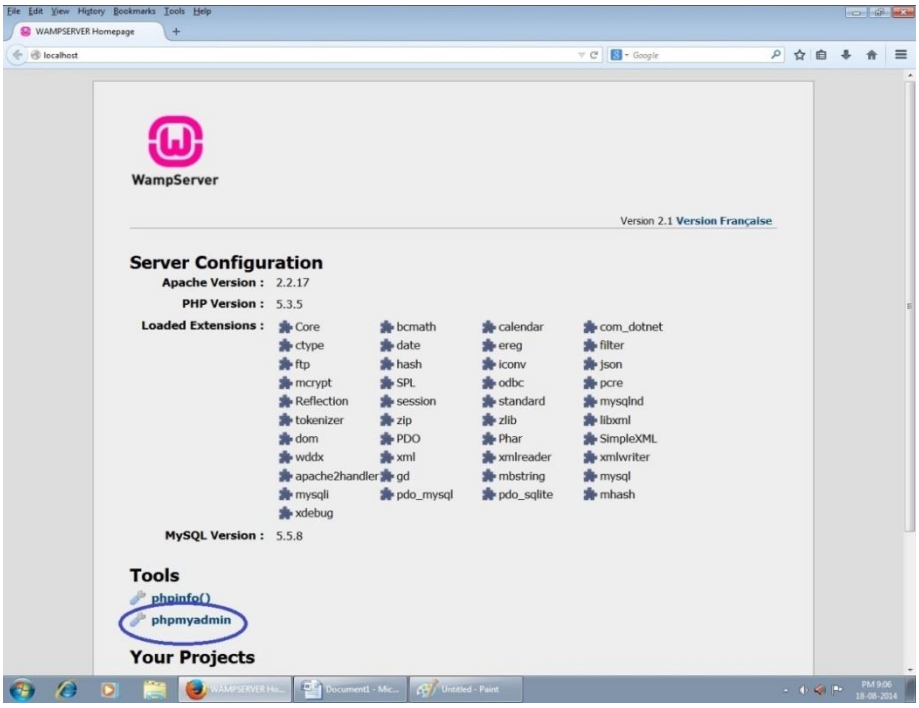
Creating Database using PhpMyAdmin & Console(using query, using Wamp server)

Creating a Database using PhpMyAdmin

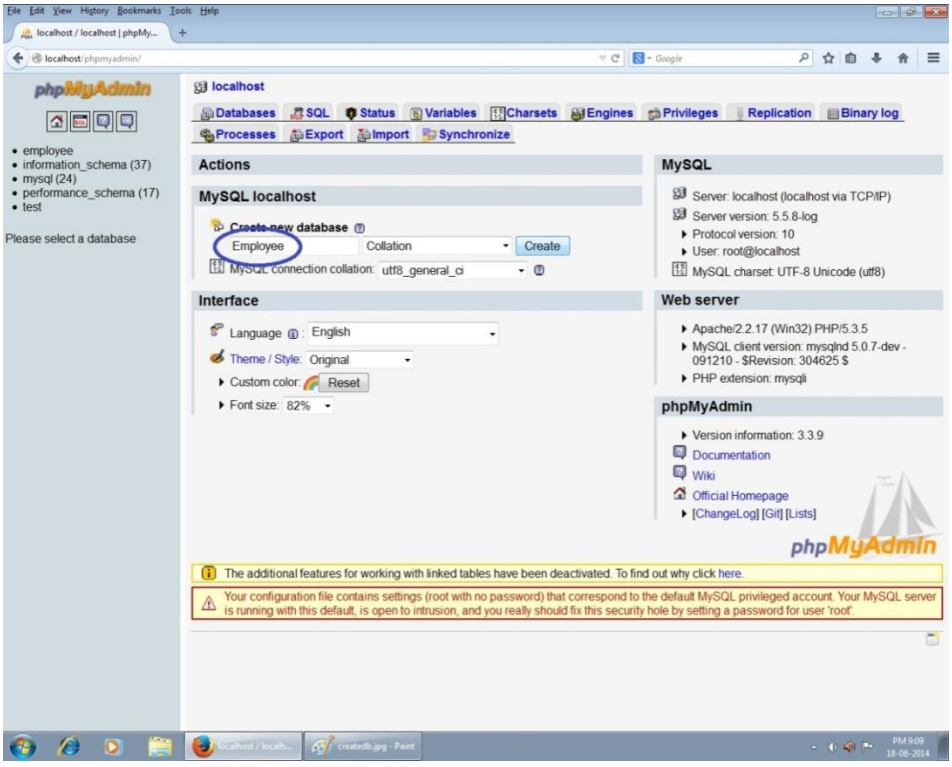
- In order to create a new database, below are the steps given:

(1) Launch PhpMyAdmin form the taskbar as shown below:

(2) Next Step is open the browser and write http://localhost/ and press Enter and you should get below screen. In the below screen there is option called as PhpMyAdmin, click on that.

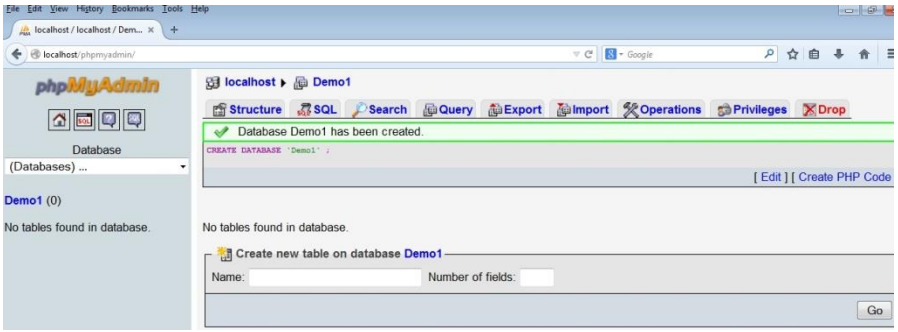


(3) By clicking on PhpMyAdmin you will get the below screen.



Enter the name of database you wish to create in the textbox given and click on create button as shown above.

(4) As soon as you click on create button, it will create new database and display the statement as shown below.



Creating New Database using PHP Script (Query)

```
<?php

$cn = mysqli_connect("localhost","root","") or die(" Could Not Connect Server");

$sql ="create database Demo2";

$r = mysqli_query($sql) or die("Query Execution Failed");

if($r)

        echo "<script>alert('Database Sucessfully Created');</script>";

mysqli_close();

?>
```

Connecting with PHP/ Database Connection, creating and executing queries using mysql_query()

mysqli_connect() Function connects or establishes connection with the MYSQL server from the PHP Script. Prior to doing things of viewing or manipulating data in database table, we must first connect with the MYSQL Server. To do so From the PHP Script we simply use the function

mysqli_connect (), which takes three arguments and returns connection on the success and false on failure.

Syntax:

$con =mysqli_connect("server","username","password");

Parameter	**Description**
Server	(1) If you are using Local Computer, Use localhost and (2) If you are using Remote Server, use either the hostname or the IP address.
Username	(1) If you are using Local Computer, Use root as the username (2) If you are using Remote Server, use the username you wish to logon.
Password	(1) If you are using Local Computer, Use ' ' as password and (2) If you are using Remote Server, use the password you wish to logon.

Example:

```
<?php

$host ="localhost";

$username="root";

$password="";

$con= mysqli_connect($host,$username,$password);

if (!$con)

{

    die("Could not connect Server:" . mysqli_error());

}

else
```

```
{
    echo "Connected Sucessfully To Server";
}
?>
```

The above example demonstrates the use of the mysqli_connect () function, which takes three parameters and returns connection on success and false on failure. To print the error message, use the die function in the block of if statement and check the condition of variable $con.

If the connection is established successfully $con contains true or else it contains false. You can also combine mysqli_error() function to get the exact error message.

Creating tables, inserting data in to table, inserting data through HTML Forms

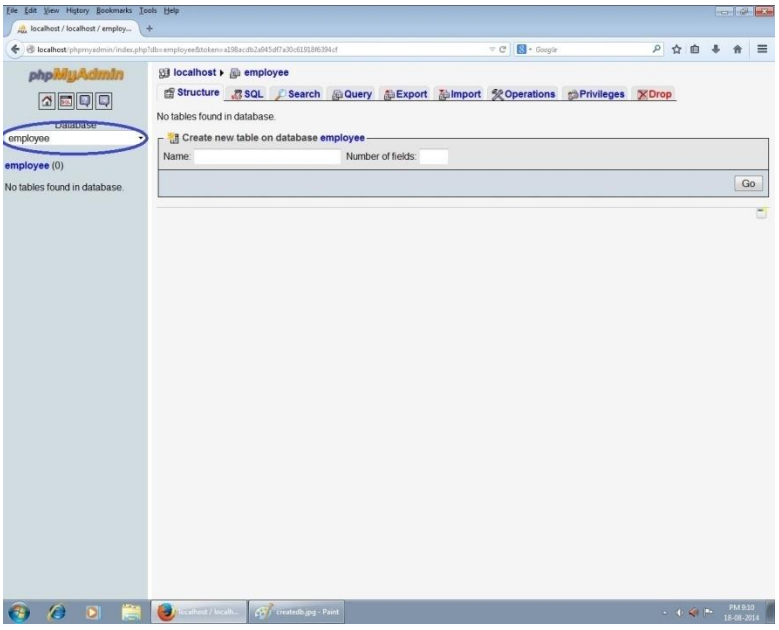
Creating Table in the Database using PhpMyAdmin

In order to create a new table using PhpMyAdmin, below are the steps given:

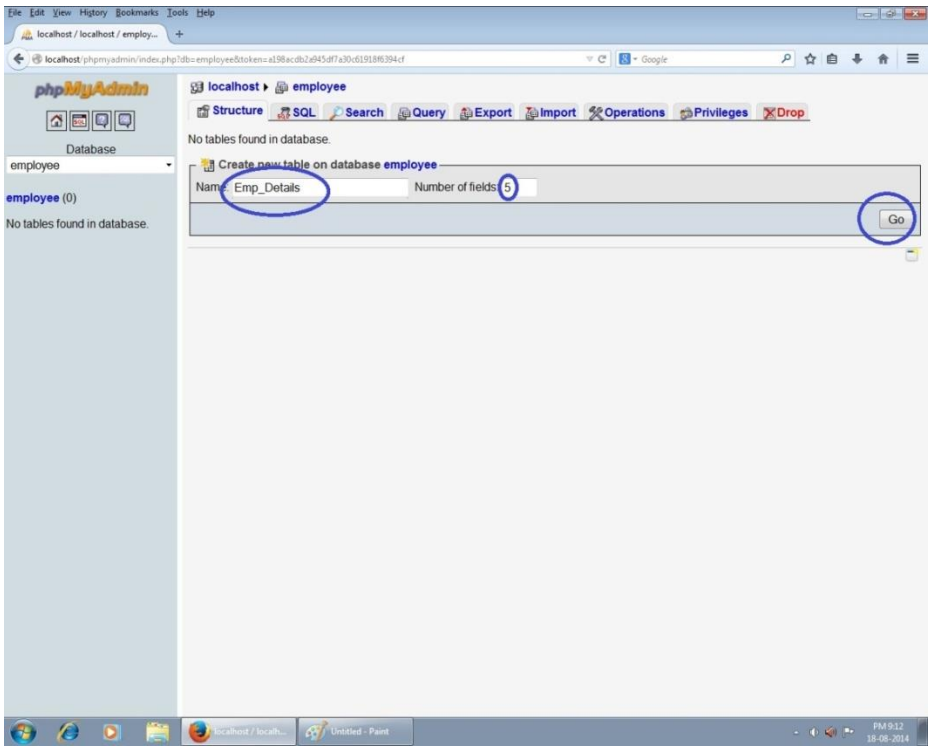
(1) Launch PhpMyAdmin form the taskbar.

(2) Next Step is open the browser and write http://localhost/ and press Enter and click on PhpMyAdmin.

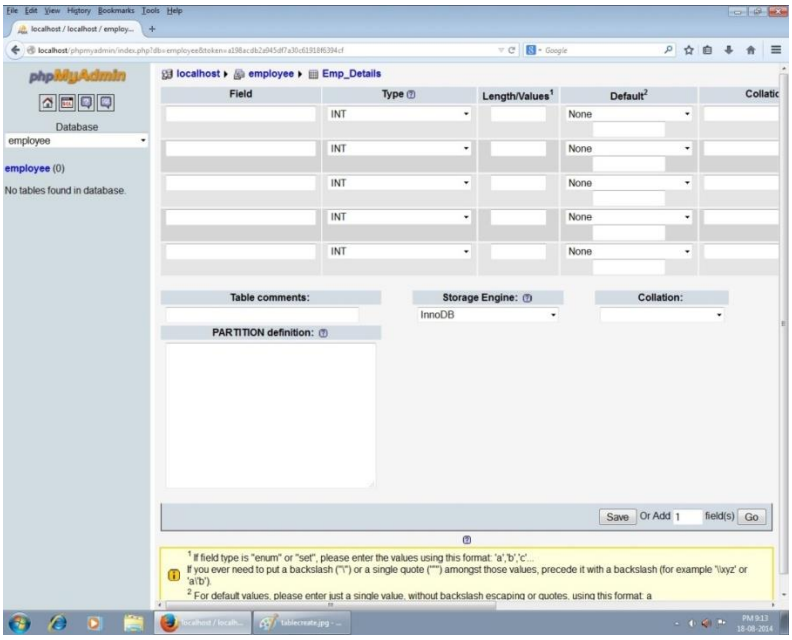
(3) Select the name of the database from the given list of databases.



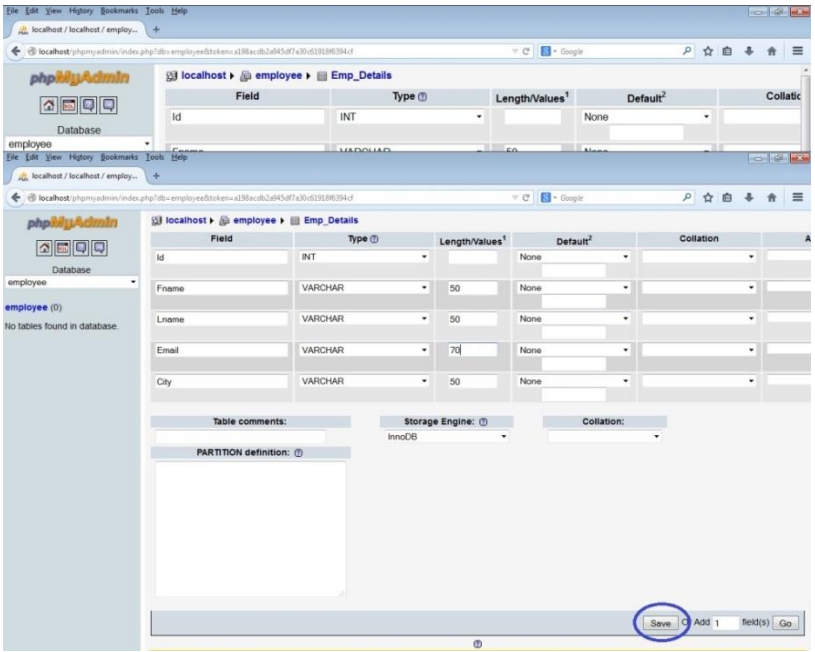
(4) Enter name of the new table you wish to create and specify how many fields you want to add in the table and then click on go button for creating new table.



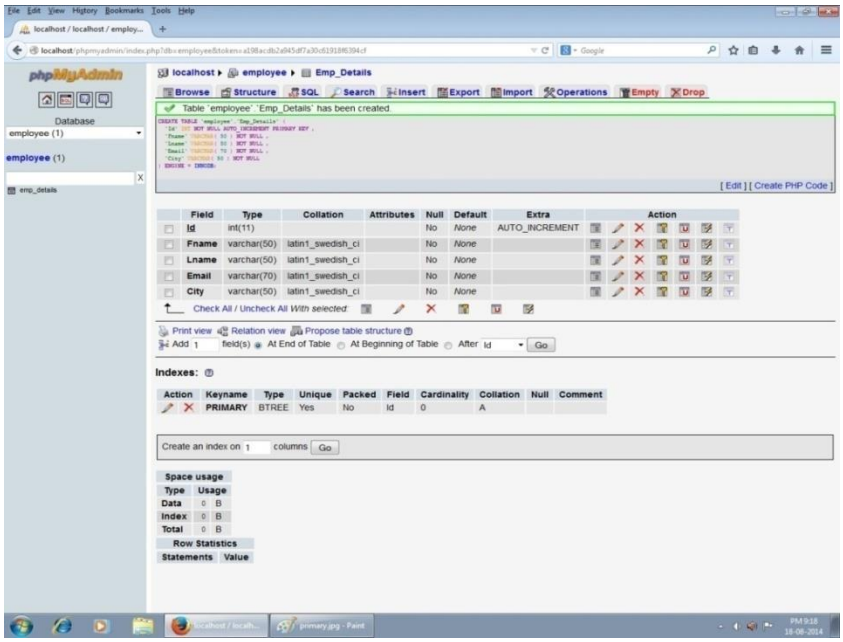
(5) As soon as you click on Go Button, a screen will appear demanding from you the name of the fields you wish to create and its relevant data types and all other criteria's like auto increment, primary key or any other key you wish to define on table.



(6) Enter the Field names and choose its data types and specify any other criteria's like primary key and auto increment if you wish and click on save button.



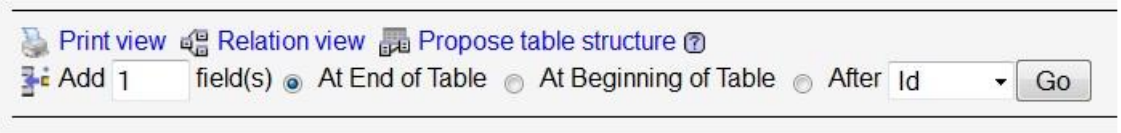
(7) As soon as you click on the save button it will create new table with specified field and display the statement to create new table as shown below:



After creating table, you can perform any of following operations on table as shown below:

Browse	Browse records from the table
Change or Alter	Alter the field specifications
Delete	Deletes the selected field from the table
Primary Key	Adds Primary key to selected Field
Unique key	Adds Unique key to selected Field
Index	It Defines index for selected Field

After Creating table, if you wish to add new columns in the table, you can do so by specifying numbers of fields to be added at the beginning, ending or after selected field as shown below:



Creating Table Using PHP Script

In order to create a new table using PHP script, below is the script given:

```
<?php

$cn =mysqli_connect("localhost","root","") or die(" Could Not Connect Server");
```

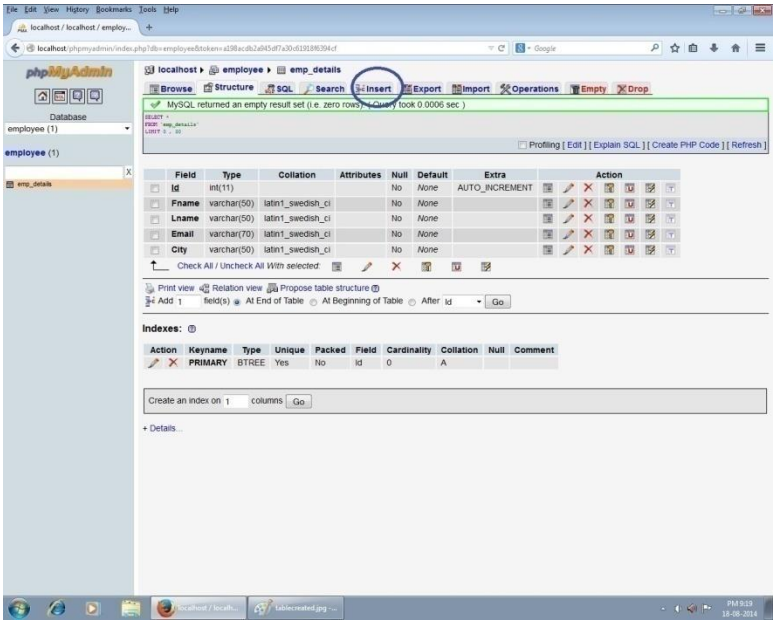
```
mysqli_select_db("Employee") or die("Database not selected");

$sql ="create table emp_details(Id int primary key,Fname varchar(50), Lname varchar(50),
city varchar(50), email varchar(50))";

$r = mysqli_query($sql) or die("Query Execution Failed");

if($r)

echo "<script>alert('Table  Created');</script>";

mysqli_close();

?>
```

Inserting Records

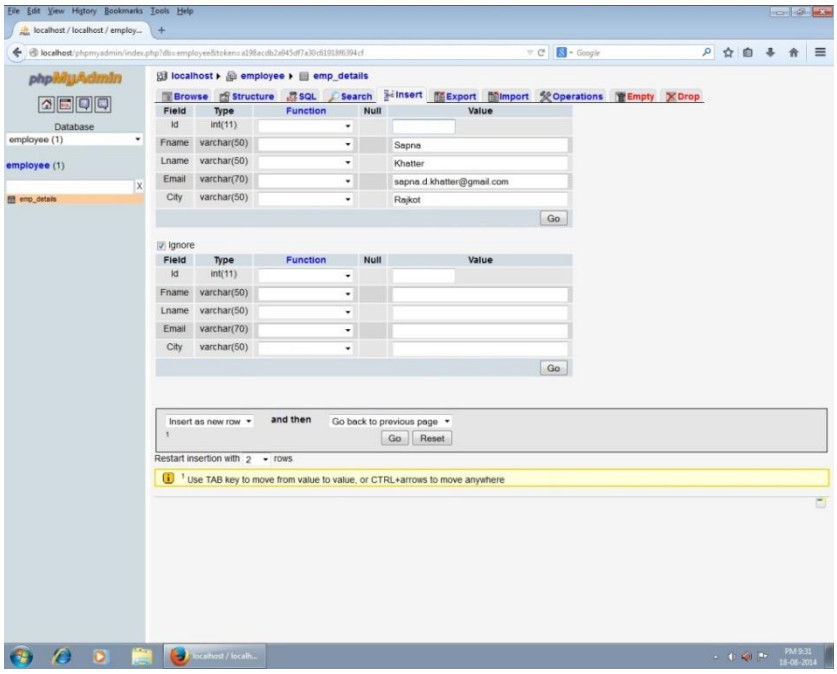
Inserting Records in the Database Table using PhpMyAdmin

In order to Insert data in database table using PhpMyAdmin, below are the steps given:

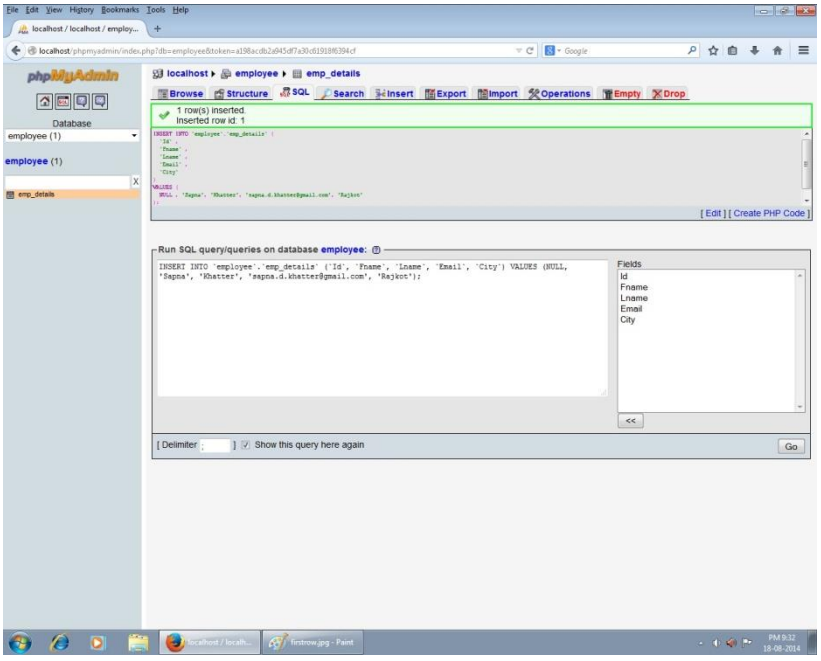
(1) Select the table from set of the tables in which you wish to enter records and then click on Insert link.



(2) It will display the below screen, fill it with data you need and then click on Go Button.



(3) As soon as you click on Go Button, Record will be inserted in the database table and it will display insert query as follows.



Inserting Records in the Database Table using PHP Script

In order to insert data in database table using Php script, below are the steps given. First of all, create the file named insert1.php.

Insert1.php

```
<html>
<head><title>Insert Data In The MySql Table</title></head>
<body>
<form method="post" action="insert2.php">
<table align="center">
	<tr>
		<th colspan="3">Insert Details</th>
	</tr>
	<tr>
		<td>First Name </td>
		<td><input type="text" name="fname"/></td>
	</tr>
	<tr>
	<td>Last Name </td>
		<td><input type="text" name="lname"/></td>
	</tr>
	<tr>
	<td>Email </td>
		<td><input type="text" name="email"/></td>
	</tr>
	<tr>
	<td>City </td>
		<td><input type="text" name="city"/></td>
	</tr>
	<tr>
		<td colspan="2" align="center">
		<input type="submit" name="insert" value="Insert"/>
		</td>
	</tr>
</table>
</form>
</body>
```

```
</html>
```

After creating an insert1.php, you need to write a Php script that will insert the data in the database table, when you click on the submit button, for that create another file called as insert2.php as shown below.

Insert2.php

```
<html>
<head><title>Insertion Query Code</title></head>
<body>
<?php
$cn =mysqli_connect("localhost","root","") or die(" Could Not Connect Server");
$db = mysqli_select_db("employee") or die("Could not Select Database");
$fname = $_POST['fname'];
$lname = $_POST['lname'];
$email = $_POST['email'];
$city = $_POST['city'];

$sql ="insert into
emp_details(Fname,Lname,Email,City)values('$fname','$lname','$email','$city')";
$r = mysqli_query($sql) or die("Query Execution Failed".mysqli_error());
if($r)
        echo "<script>alert('Sucessfully Inserted');</script>";
        echo "<script>window.location='insert1.php';</script>";
?>
</body>
</html>
```

Retrieving data from Table, using mysqli_num_rows() , Printing the output using PHP and HTML

- As soon as you successfully establish connection between PHP script and MySQL database, you can retrieve records from the table and display those records on the web page using the table tag of the html efficiently which is very simple

- Table tag permits you to display records from the database table in the form of rows and columns.
- To retrieve records from the database table using the php script follow the below steps:
  (1) Establish connection with MySQL server using the mysqli_connect() function.
  (2) Select the database of your choice using the mysqli_select_db() function.
  (3) Fire the select query to retrieve records from the database table using mysqli_query() function.
  (4) Use the Concept of the loop, to display all the records that are retrieved from the database table using the HTML table tag and using the function mysqli_fetch_array() or mysqli_fetch_row().
  (5) You can count the number of records retrieved using the mysqli_num_rows() function.

select.php

```
<html>

<head><title>Selection Query</title></head>

<body>

<?php

$cn =mysqli_connect("localhost","root","") or die(" Could Not Connect Server");

$db = mysqli_select_db("employee") or die("Could not Select Database");

$sql ="select * from emp_details";

$r = mysqli_query($sql) or die("Query Execution Failed".mysql_error());

$count = mysqli_num_rows($r);
```

```
echo "<table border=1 align='center'>";

echo
"<tr><th>Id</th><th>FirstName</th><th>LastName</th><th>Email</th><th>City</th></tr>
";

while($row = mysqli_fetch_array($r))

{

        echo "<tr>";

        echo "<td>$row[0]</td>";

        echo "<td>$row['Fname']</td>";

        echo "<td>$row['Lname']</td>";

        echo "<td>$row[3]</td>";

        echo "<td>$row[4]</td>";

        echo "</tr>";

}

echo "</table>";

echo "Number of Records Retrieved :-".$count;

?>

</body>

</html>
```

When using the mysqli_fetch_array function, as shown in the while loop you can use either the index number or the name of the column in the database table. Index 0 means first column in the database table

Searching a record, displaying record fields in HTML form controls, Updating and deleting records

Searching Records in the Database Table using PHP Script

In order to search data in database table using Php script, below are the steps given. First of all create the file named search1.php.

Search1.php

```
<html>
<head><title>Searching </title></head>
<body>
<form action="search2.php" method="post">
<table align="center" border="0">
<tr>
<th colspan="2">Search </th>
</tr>
<tr>
<td>Enter Word </td>
<td><input type="text" name="txtsearch"/></td>
</tr>
<tr>
<td colspan="2" align="center">
<input type="submit" name="search" value="Search"/>
</td>
</tr>
</table>
</form>
</body>
```

```
</html>
```

After creating a search1.php, you need to write a Php script that will search the data in the database table, when you click on the submit button, for that create another file called as search2.php as shown below.

Search2.php

```
<html>
<head><title>Searching Records</title></head>
<body>
<?php
$cn =mysqli_connect("localhost","root","") or die(" Could Not Connect Server");
$db = mysqli_select_db("employee") or die("Could not Select Database");
 $s = $_POST['txtsearch'];

$sql ="select * from emp_details WHERE  Fname LIKE '%" . $s . "%' or Lname LIKE '%" . $s . "%'
or city LIKE '%" . $s . "%' or email LIKE '%" . $s . "%'  ";
$r = mysqli_query($sql) or die("Query Execution Failed".mysqli_error());
 $n =mysqli_num_rows($r);
if($n!=0)
{
echo "<table border=1 align='center'>";
echo
"<tr><th>Id</th><th>FirstName</th><th>LastName</th><th>Email</th><th>City</th></tr>
";

while($row = mysqli_fetch_array($r))
{
echo "<tr>";
                        echo "<td>$row[0]</td>";
                        echo "<td>$row[1]</td>";
                        echo "<td>$row[2]</td>";
                        echo "<td>$row[3]</td>";
```

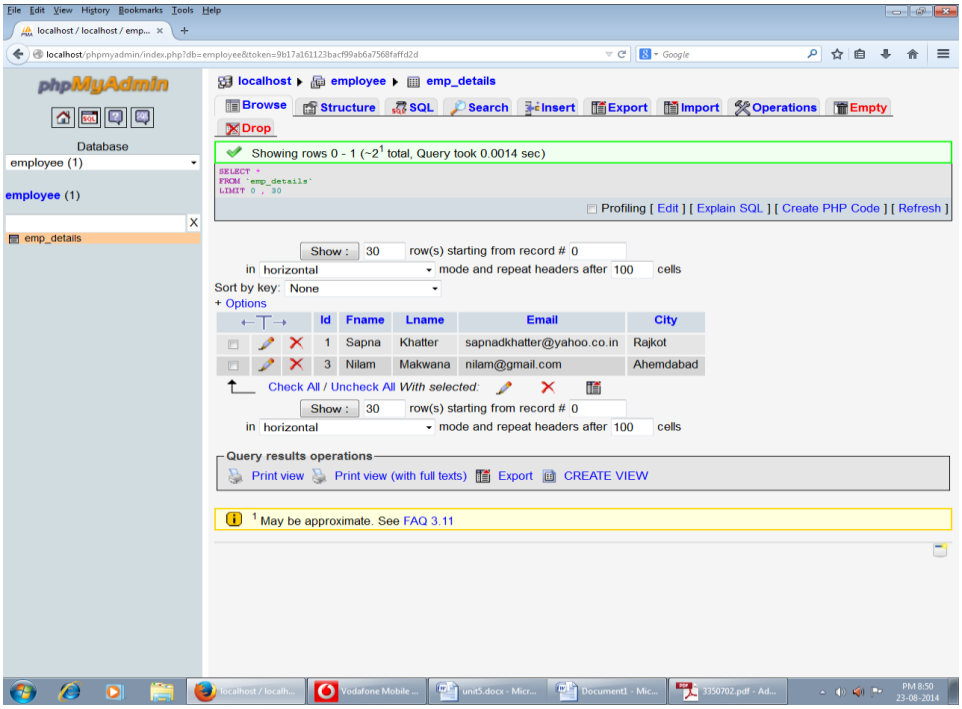
```
                    echo "<td>$row[4]</td>";
        echo "</tr>";
}
}
else
{
echo "<font color='red'> No Records Found</font>";
}
?>
<a href="search1.php">Search Again</a>
</body>
</html>
```

Updating Records

Updating Records in the database table using PhpMyAdmin:

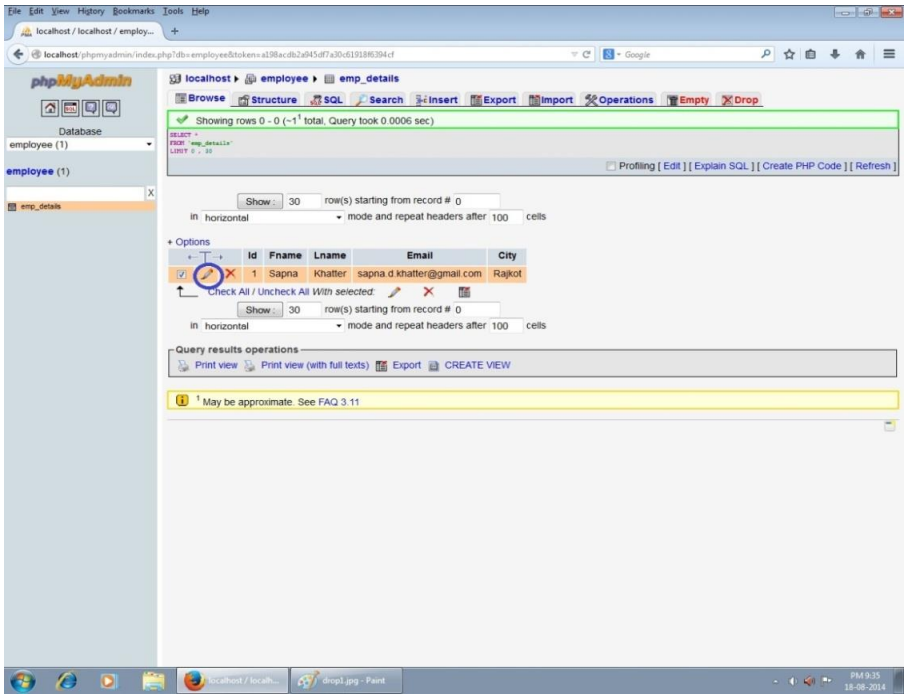
In order to update the record in the database table, below are the steps given:

(1) Select the table from the set of the tables in the database and click on Browse link and it will display all the records of the table selected as shown below.

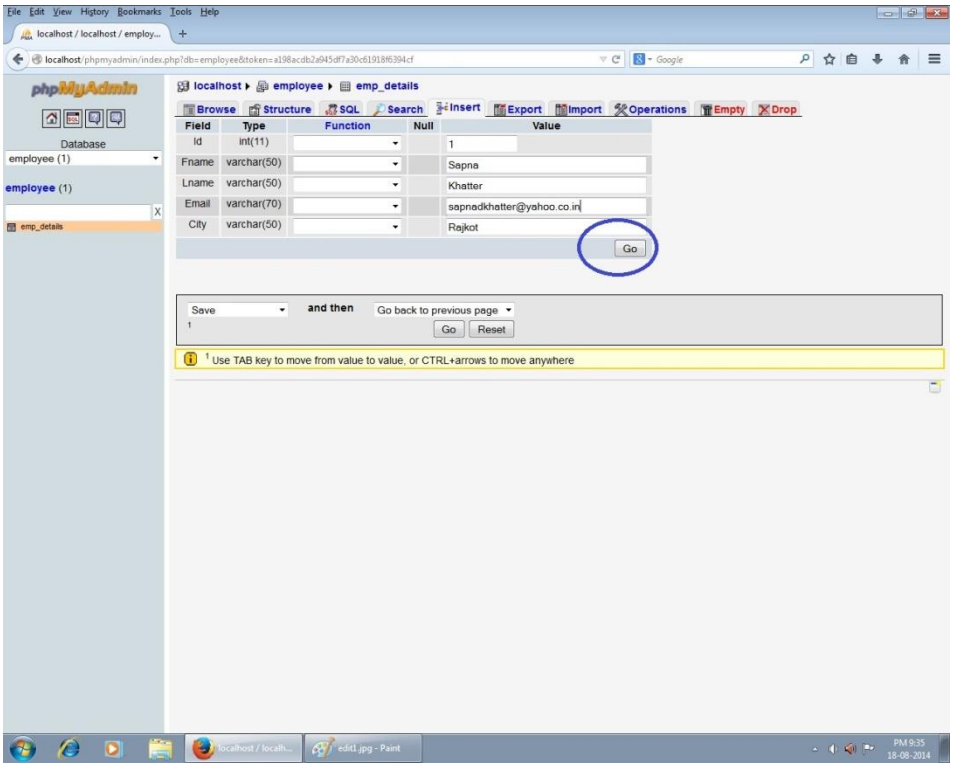


(2) Select the record you wish to update from the set of records and then click on

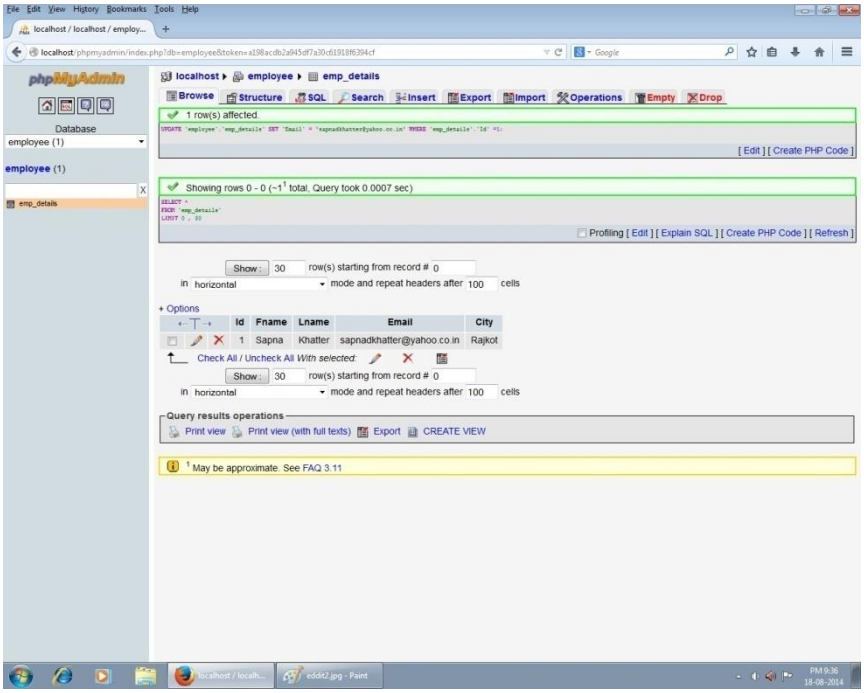
button to update the record.



(3) As soon as you click on button, it will display the selected record in textboxes so that you can make appropriate changes in it.



(4) Update the fields in the record as per requirement and then click on, go button to save changes in the database table. It will update the record after pressing the go button as shown below.



Updating Records in the Database Table using PHP Script

In order to update data in database table using Php script, below are the steps given. First of all, create the file named update1.php in which all the records are retrieved and in addition to that one column in HTML table is added which will contain a hyperlink to pass to next page with id of the record to be updated.

Update1.php

```
<html>
<head><title>Selection Query</title></head>
<body>
<?php
$cn =mysqli_connect("localhost","root","") or die(" Could Not Connect Server");
$db = mysqli_select_db("employee") or die("Could not Select Database");
$sql ="select * from emp_details";
$r = mysqli_query($sql) or die("Query Execution Failed".mysqli_error());
echo "<table border=1 align='center'>";
echo
"<tr><th>Id</th><th>FirstName</th><th>LastName</th><th>Email</th><th>City</th></tr>
";
while($row = mysqli_fetch_array($r))
{
        echo "<tr>";
            echo "<td>$row[0]</td>";
            echo "<td>$row[1]</td>";
            echo "<td>$row[2]</td>";
            echo "<td>$row[3]</td>";
            echo "<td>$row[4]</td>";
        echo "<td><a href='update2.php?id=$row[0]'>Update</a></td>";
        echo "</tr>";

}
echo "</table>";
?>
</body>
```

```
</html>
```

(2) After creating a file called as update1.php, create another file called as update2.php in which the selected record fields will be displayed in the textboxes where you can make changes as per your requirement and click on update button to reflect changes in the database table.

update2.php

```
<?php
$id = $_REQUEST['id'];
$cn =mysqli_connect("localhost","root","") or die(" Could Not Connect Server");
$db = mysqli_select_db("employee") or die("Could not Select Database");
$sql ="select * from emp_details where Id=$id";
$r = mysqli_query($sql) or die("Query Execution Failed".mysql_error());
$row = mysqli_fetch_array($r);
?>
<html>
<head><title>Update Fields Demonstration</title></head>
<body>
<form method="post" action="update3.php">
<table align="center">
<tr>
<th colspan="3">Update Fields</th>
</tr>
<tr>
<td>First Name </td>
<td><input type="text" name="fname" value="<?php echo $row[1];?>"/></td>
</tr>
<tr>
<td>Last Name </td>
<td><input type="text" name="lname" value="<?php echo $row[2];?>"/></td>
</tr>
<tr>
<td>Email </td>
```

```
<td><input type="text" name="email" value="<?php echo $row[3];?>"/></td>
</tr>
<tr>
<td>City </td>
<td><input type="text" name="city" value="<?php echo $row[4];?>"/></td>
</tr>
<tr>
<td colspan="2" align="center">
<input type="hidden" name="id" value="<?php echo $row[0];?>" />
<input type="submit" name="update" value="Update"/>
</td>
</tr>
</table>
</form>

</body>
</html>
```

(3) When user does the changes in the textboxes given in the web form, and then clicks on submit button, it is redirected to update3.php in which the code is written for actual updating in database table, and finally changes are reflected in the database table.

Update3.php

```
<html>
<head>
<title>Update Query</title>
</head>
<body>
<?php
$cn =mysqli_connect("localhost","root","") or die(" Could Not Connect Server");
$db = mysqli_select_db("employee") or die("Could not Select Database");
echo $id = $_REQUEST['id'];
$fname = $_POST['fname'];
```

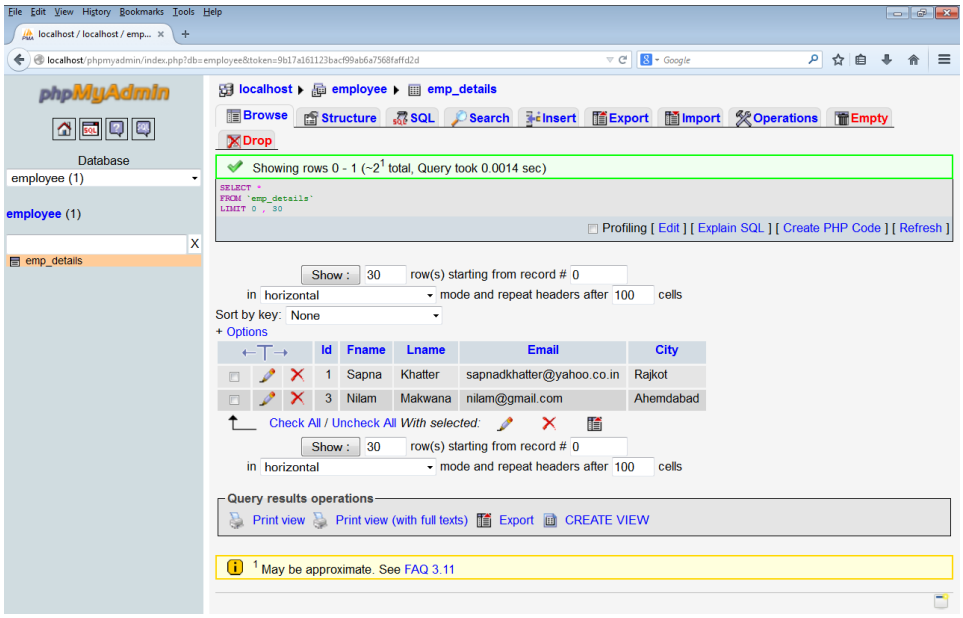
```
$lname = $_POST['lname'];
$email = $_POST['email'];
$city = $_POST['city'];

 $sql ="update emp_details set Fname='$fname' ,Lname='$lname', City='$city' , Email='$email' where Id=$id";
$r = mysqli_query($sql) or die("Query Execution Failed".mysql_error());
if($r)
        echo "<script>alert('Sucessfully Updated');</script>";
        echo "<script>window.location='update1.php';</script>";
?>
</body>
</html>
```

Deleting Records

Deleting Records in the database table using PhpMyAdmin:

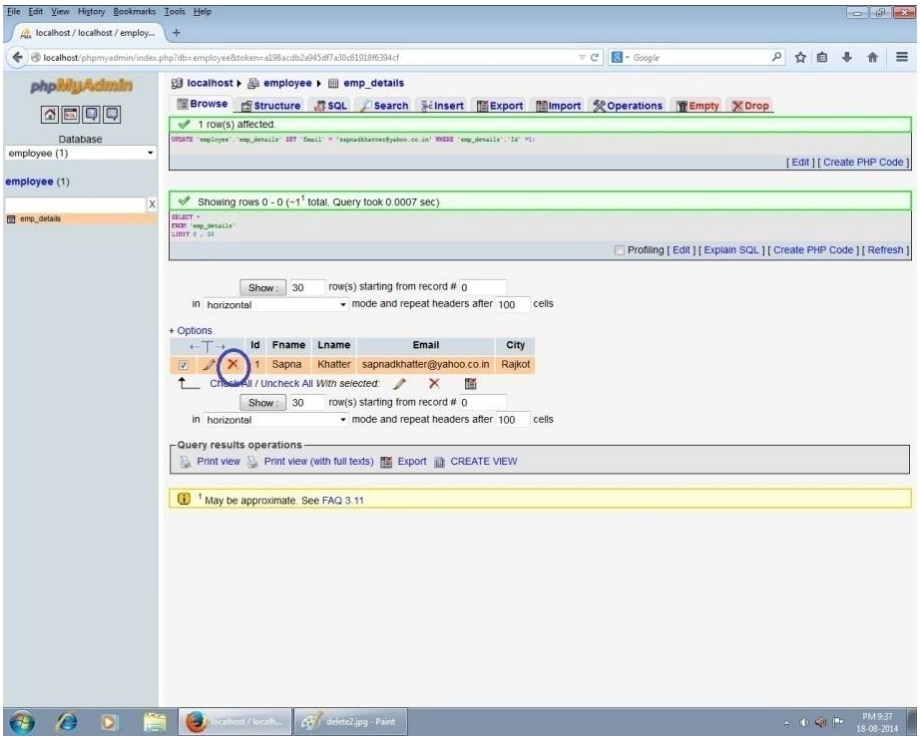
In order to delete the record in the database table, below are the steps given:

(1) Select the table from the set of the tables in the database and click on Browse link and it will display all the records of the table selected as shown below.

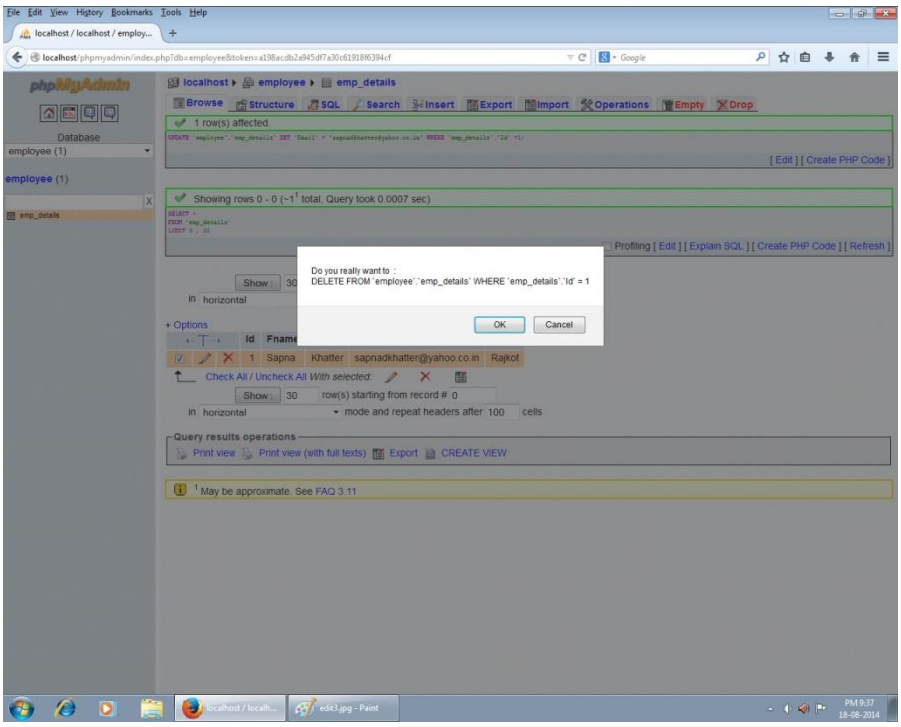


(2) Select the record you wish to delete from the set of records and then click on

button to delete the record.



(3) As soon as you click on button, it will display confirm message box asking for do you really want to delete the record, if you wish to delete the record, click on ok button.



(4) As soon as you press ok the record will be deleted from the database table.

Deleting Records in the Database Table using PHP Script

In order to delete data from database table using Php script, below are the steps given. First of all, create the file named delete1.php in which all the records are retrieved and in addition to that one column in HTML table is added which will contain a hyperlink to pass to next page with id of the record to be deleted.

Delete1.php

```
<html>
<head><title>Selection Query</title></head>
<body>
<?php
$cn =mysqli_connect("localhost","root","") or die(" Could Not Connect Server");
$db = mysqli_select_db("employee") or die("Could not Select Database");
$sql ="select * from emp_details";
$r = mysqli_query($sql) or die("Query Execution Failed".mysql_error());
echo "<table border=1 align='center'>";
echo "<tr><th>Id</th><th>FirstName</th><th>LastName</th><th>Email</th>
<th>City</th></tr>";
while($row = mysqli_fetch_array($r))
{
        echo "<tr>";
                        echo "<td>$row[0]</td>";
                        echo "<td>$row[1]</td>";
                        echo "<td>$row[2]</td>";
                        echo "<td>$row[3]</td>";
                        echo "<td>$row[4]</td>";
                        echo "<td>
<a href='delete2.php?id=$row[0]'>Delete</a></td>";
        echo "</tr>";
}
echo "</table>";
?>
```

```
</body>
</html>
```

(2) After creating a file called as delete1.php, create another file called as delete2.php that deletes the record selected from the first file delete1.php from the database table.

Delete2.php

```
<html>
<head><title>Delete Query</title></head>
<body>
<?php
$id = $_REQUEST['id'];
$cn =mysqli_connect("localhost","root","") or die(" Could Not Connect Server");
$db = mysqli_select_db("employee") or die("Could not Select Database");
$sql ="delete from emp_details where Id=$id";
$r = mysqli_query($sql) or die("Query Execution Failed".mysql_error());
if($r)
        echo "<script>alert('Sucessfully Deleted');</script>";
        echo "<script>window.location='delete1.php';</script>";

?>
</body>
</html>
```

Hosting Website (Using ‘C’ panel, Using FileZilla Software)

Hosting Website Using ‘C’ panel

cPanel is a web-based control panel tool which provides a graphical user interface and automation tools which permit’s you manage your web hosting account through a web interface instead of using the control. With the use of the cPanel you are able to accomplish your tasks faster and efficiently and even non-technical persons can easily set their websites via cPanel. cPanel offers a rich set of

features, ranging from adding an email address to managing sub-domain names and many more as described below.

Account Information and Statistics

You can find important information about your hosting account in this section. This Information includes Hosting package, Ip address, server name, Home directory, Disk space usage, monthly transfer, email accounts, FTP accounts, sub domains, SQL database, Last Login from, bandwidth, webalizer, Raw Access Logs etc. You can also monitor the resource usage of your account in this section.

cPanel Files Section

The Files section in cPanel includes the following tools - Backups, Backup Wizard, File Manager, FTP accounts, FTP Session Control, FTP Backup restore.

cPanel Domains Section

Through the Domains section you can manage your sub domains, add-on domains and other domains. You can also set redirects for your URLs. Additionally, you can purchase a domain name transfer and a domain ID protection. You can register a new domain, change the primary domain name for your account and manage the DNS configuration of your domains.

cPanel Mail Tutorial

The Mail area functions allow a user to do many different tasks with email accounts. This includes creating email accounts, removing accounts, forwarding email, etc. It can also include functionalities such as Email Accounts, Webmail, Auto Responders, Forwarders, Account Level Filtering, User Level Filtering, Email Authentication etc.

cPanel Security

The Security section in cPanel includes the following tools - Password Protect Directories, IP Deny Manager, and Site Security Check.

cPanel Databases

The Databases section allows you to create MySQL databases and users, to modify databases and access to them. SQL is an international standard in querying and retrieving information from databases. MySQL is essentially an SQL server - it responds to requests for information that are written in SQL. You can communicate with MySQL using a PHP. MySQL is Open Source software and free for use.

cPanel software section

This section allows you to install scripts, remove scripts, or upgrade scripts as per your requirement.

cPanel Advanced section

This section allows you to use tools such as Apache Handlers, Image Managers, Index Managers, and Error Pages.