
Web Technology

Dr.K.Pazhanikumar

Asst.Prof., Dept of Computer Science

S.T.Hindu College

Nagercoil

Web Technology

Web technology is the establishment and use of mechanisms that make it possible for different computers and devices to communicate and share resources. Web technologies are infrastructural building blocks of any effective computer network.

Markup Languages

Markup languages like HTML, CSS and XML are part of Web technology. These languages tell computers in text how to format, layout and style Web pages and programs.

Different Sides of Web Technology

Client Side Scripting is utilized when the client's (user's) browser has all the code and the page is modified on the basis of the client's (user's) information. The Web Browser executes the client side scripting that locates within the user's computer. Client-side scripts are also known as embedded script (as they are often embedded within an HTML or XHTML document).

Example Java Script, VB Script

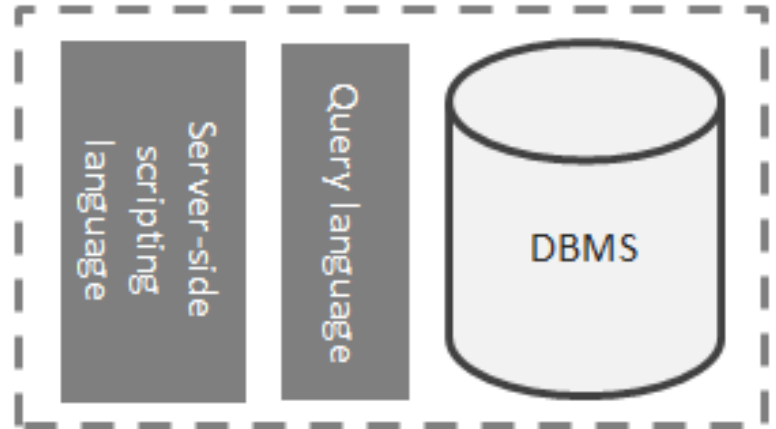
Server Side Scripting Technology

Server side scripting is a technique used in website designing to create dynamic web pages based various conditions when the user's (client's) browser makes a request to the server The Web Server executes the server side scripting which produces the page to be send to the browser.

Example PHP, JSP ,ASP.NET, PERL



Linux / Windows / IOS
Operating system running a
web-browser



Web Server with server-side scripting
language, query language and a Database
Management System (DBMS)

TOP 5 PROGRAMMING LANGUAGES FOR SERVER-SIDE WEB DEVELOPMENT

- Python
 - Ruby
 - Node.js
 - PHP
 - Go
-

PYTHON

Python is the most interesting and profitable programming language to learn in the current scenario. It is not just a server-side programming language, we can develop a complicated application using it. The syntaxes are programmer-friendly, especially for the beginners. It comes with a large library with so many different pre-coded functions.

RUBY

The popularity of Python is more than Ruby, and in comparison, Python is easier to learn. Ruby is perfect for small web application development and that too rapidly. Rails is the most popular framework for Ruby, and the framework is highly programmer-friendly.

Node.js

JavaScript is the most popular client-side programming language that every web developer needs to learn along with HTML and CSS. But the popularity of JavaScript increased in multifold when Node.js was introduced as it is a server-side programming language based on JavaScript. Therefore, there is going to be better sync of client side and server side programming language. As a matter of fact, Node.js is the fastest growing server-side web development programming language.

PHP

PHP is the oldest programming language on the list. Yet, it is one of the most sought after programming languages in the server-side. It is also the most used programming language in web development as far as the server-side goes. It dominated the Web 2.0 era, and the platform WordPress is made out of PHP on which most of the content and business presentation websites are made.

GO

GO is a programming language introduced by Google in 2009 and it is expected to lead the server-side web development in a few years. It is growing rapidly, and that is because most of the programmers consider it to be technically better than Java and equal with Python and Ruby. In fact, for large applications, GO performs better, and Google is pumping its popularity.

PHP

PHP Hypertext Preprocessor

PHP was conceived by Rasmus Lerdorf

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

Usage statistics and market share of PHP for websites

79% websites were developed using PHP

Popular sites using PHP

Facebook.com

Wikipedia.org

Qq.com

Taobao.com

Sina.com.cn

Wordpress.com

Vk.com

Weibo.com

Babylon.com

Mail.ru



PHP

PHP use has increased dramatically over the last 5 years.

The main reasons for its popularity are:

- It is open-source and free!
 - Easy to use. It has a very simple syntax unlike other languages such as Perl or C. Rather than writing lots of code to create a webpage, we create HTML documents and embed simple PHP codes into them.
 - It has multi-platform support. It supports all major operating systems. Moreover, the syntax is consistent among different platforms. We can create PHP codes in Windows and easily switch to Unix.
 - PHP supports many new technologies. In particular, it supports MySQL.
-

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
 - PHP code are executed on the server, and the result is returned to the browser as plain HTML
 - PHP files have extension ".php"
-

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 our database
- PHP can restrict users to access some pages on your website
- PHP can encrypt data

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, 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
-

Creating a Simple PHP file

All PHP commands are enclosed within special start and end tags:

```
<?php
```

```
...PHP code...
```

```
?>
```

SIMPLE PHP PROGRAM

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?php
```

```
echo "My first PHP script!";
```

```
?>
```

```
</body>
```

```
</html>
```

WampServer

WampServer is a Windows web development environment. It allows you to create web applications with PHP and a MySQL database. Alongside, PhpMyAdmin allows us to manage easily our databases.

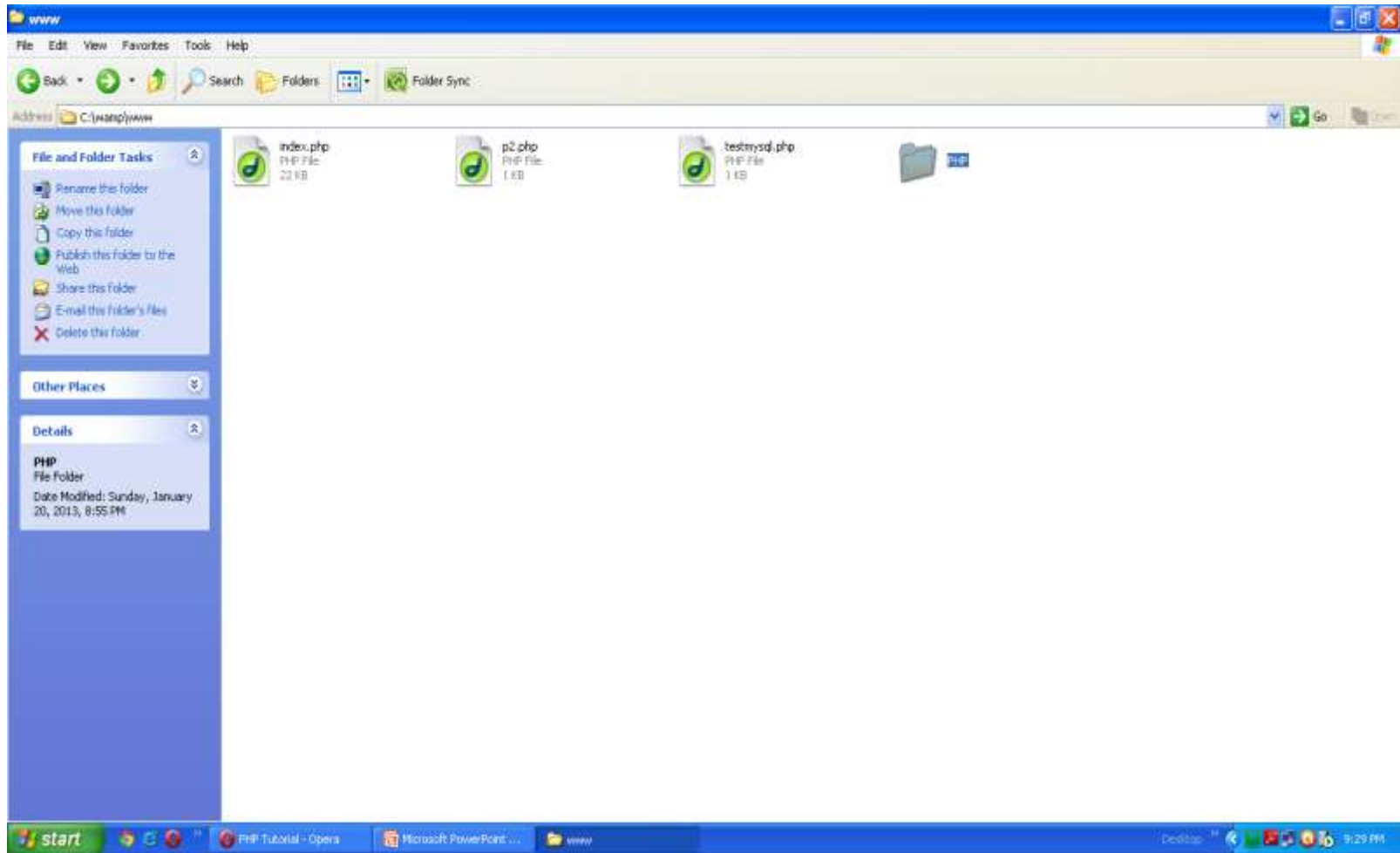
We can download it from the site

<http://www.wampserver.com/en/>

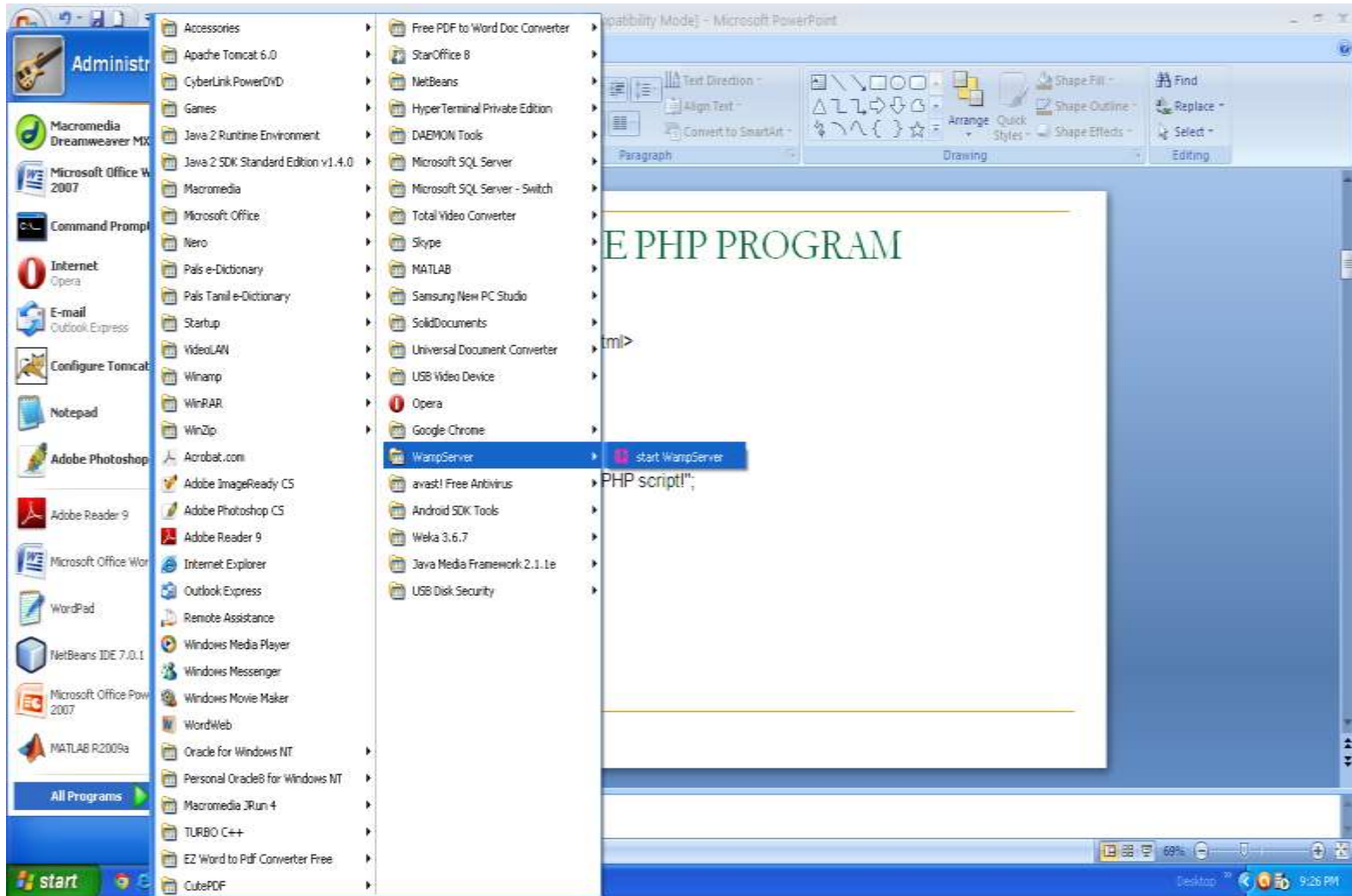
Installing WampServer

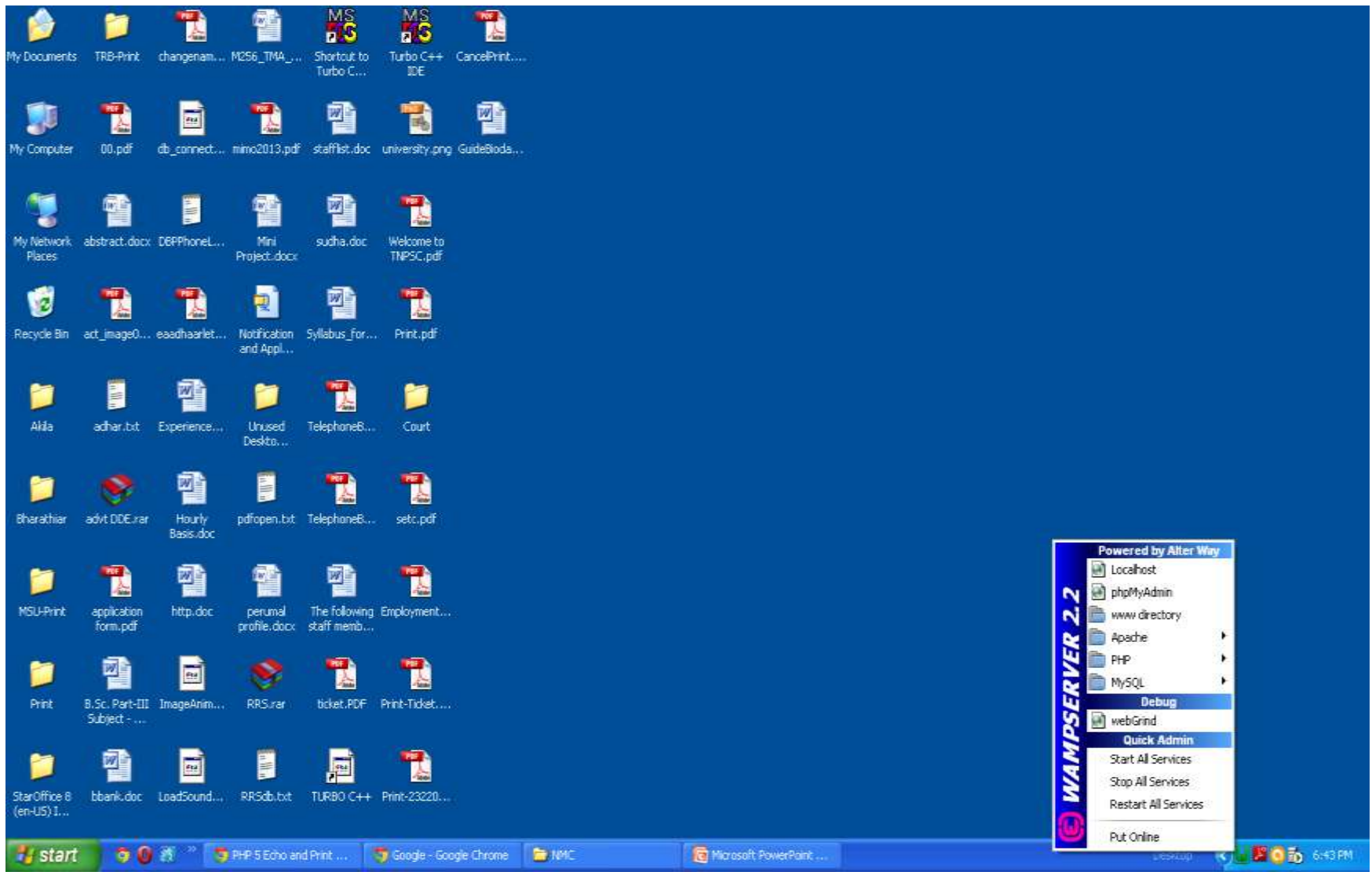
- Double click on the downloaded file and just follow the instructions. Everything is automatic. The WampServer package is delivered with the latest releases of Apache, MySQL and PHP.
 - The “www” directory will be automatically created (usually c:\wamp\www)
 - Create a subdirectory in “www” and put your PHP files inside.
 - Click on the “localhost” link in the WampServer menu or open your internet browser and go to the URL : <http://localhost>
-

Create a folder in WWW



Starting Wamp Server





Powered by Alter Way

- Localhost
- phpMyAdmin
- www directory
- Apache
- PHP
- MySQL
- Debug**
- webGrind
- Quick Admin**
- Start All Services
- Stop All Services
- Restart All Services
- Put Online



Version 2.2 Version Française

Server Configuration

Apache Version : 2.2.21

PHP Version : 5.3.10

Loaded Extensions :

- Core
- ereg
- mcrypt
- standard
- dom
- xmlreader
- mysqli

- bcmath
- filter
- SPL
- mysqlnd
- PDO
- xmlwriter
- pdo_mysql

- calendar
- ftp
- odbc
- tokenizer
- Phar
- apache2handler
- pdo_sqlite

- com_dotnet
- hash
- pcre
- zip
- SimpleXML
- mbstring
- mhash

- ctype
- iconv
- Reflection
- zlib
- wddx
- gd
- xdebug

- date
- json
- session
- libxml
- xml
- mysql

MySQL Version : 5.5.20

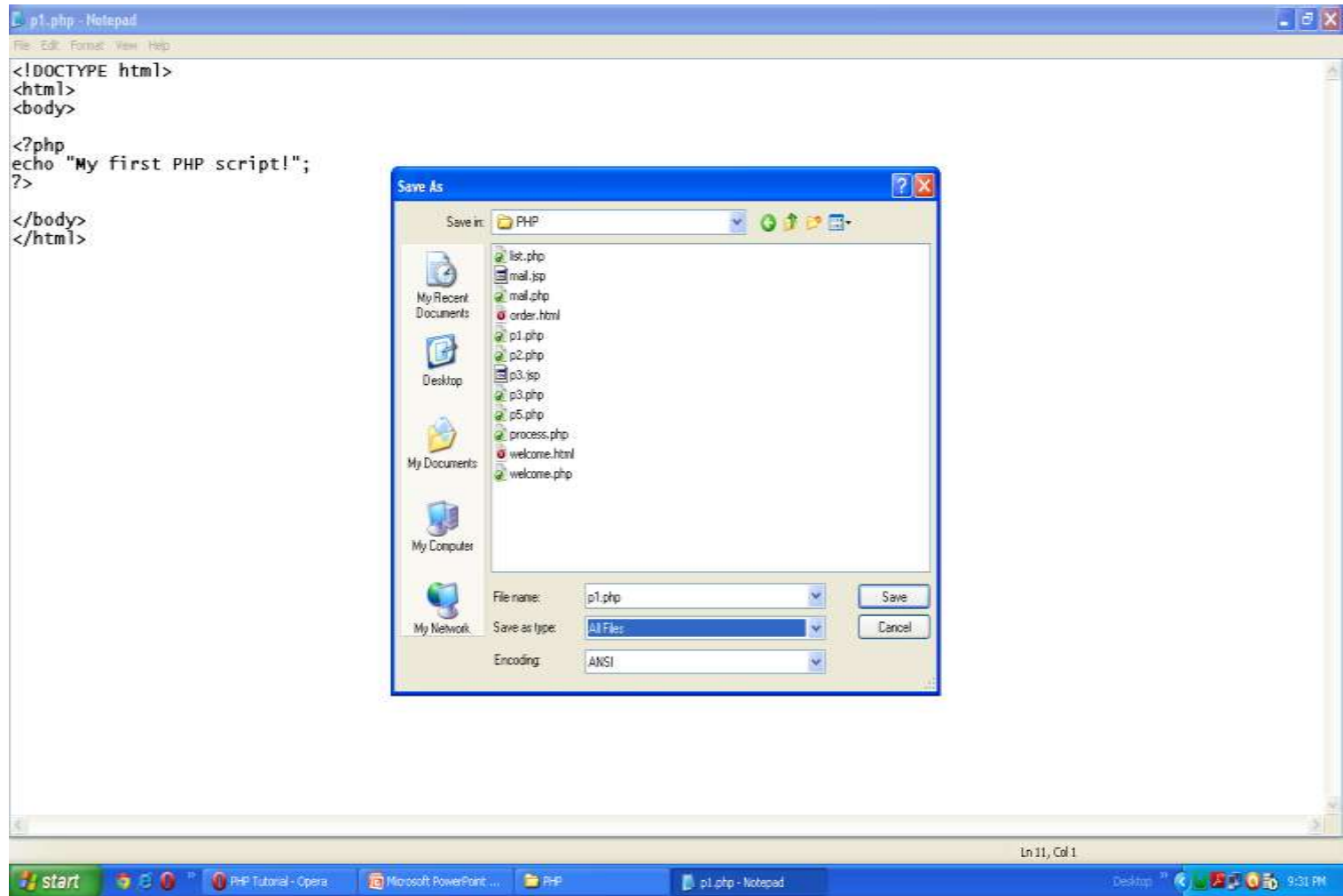
Tools

- phpinfo()
- phpmyadmin

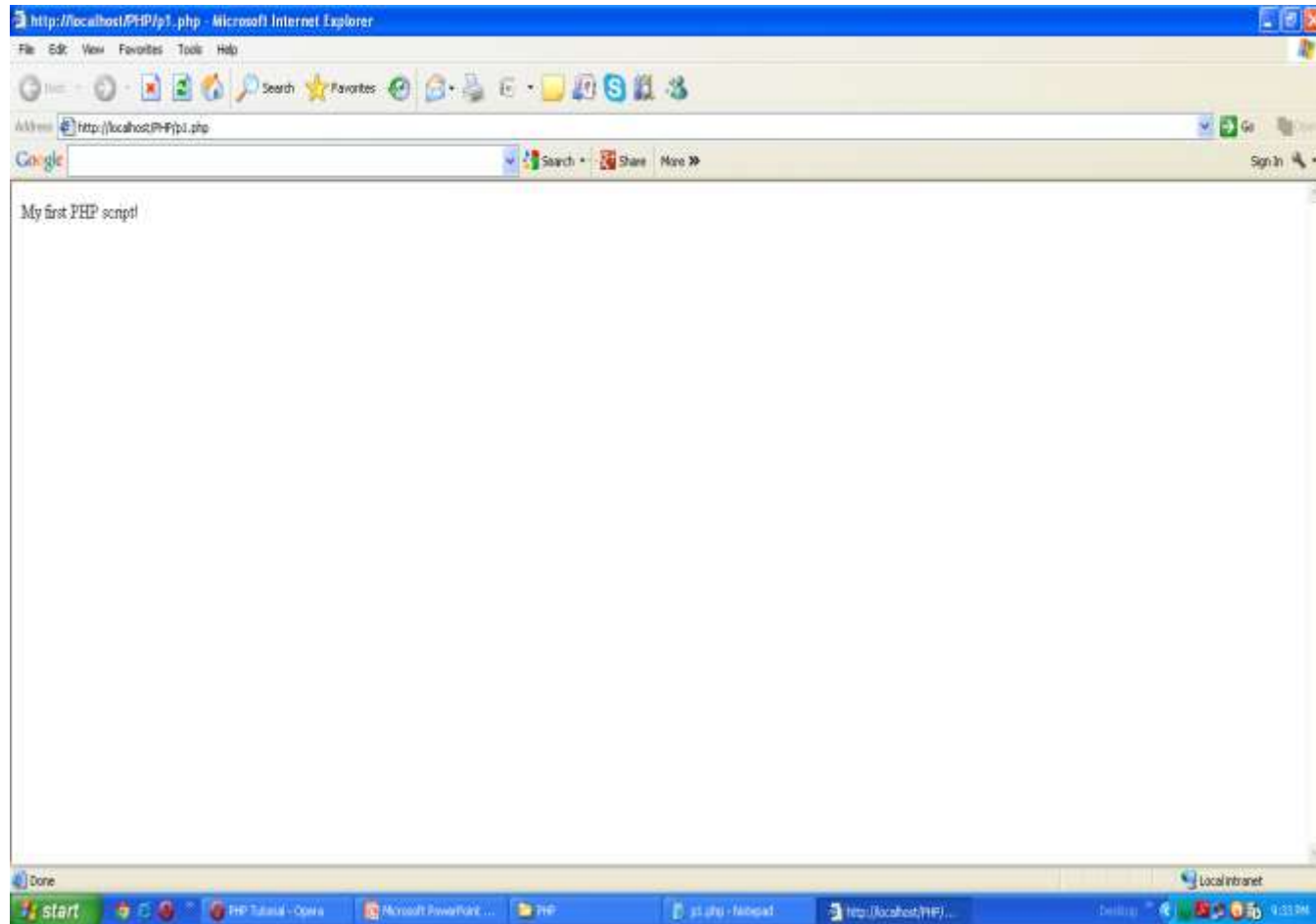
Your Projects

- PHP
- PHP1
- ignou
- oes
- OBS

P1.php



Open IE



Variable

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

→ Every variable has a name, which is preceded by a dollar (\$) symbol.

→ Variable names are case sensitive and they must begin with a letter or underscore character.

We can replace the PHP code above with:

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

This will produce the same result as before.

→ To assign a value to a variable, use the equality (=) symbol (ex: \$answer = 'A: Chameleon';).

→ To use a variable value in our script, call the variable by its name. PHP will substitute its value when the code is executed (ex: Echo "<h2><i>\$answer</i></h2>");).

Data Types

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

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

Operators

There are over 15 operators in PHP

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

Conditional Statement

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

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

Example

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

Switch Statement

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

Example

```
<?php
$favcolor="red";
switch ($favcolor)
{
case "red":
    echo "Your favorite color is red!";
    break;
case "blue":
    echo "Your favorite color is blue!";
    break;
case "green":
    echo "Your favorite color is green!";
    break;
default:
    echo "Your favorite color is neither red, blue, or green!";
}
?>
```

Looping Statements-while

```
<?php
// define number and limits for multiplication tables
$num = 11;
$upperLimit = 10;
$lowerLimit = 1;

// loop and multiply to create table
while ($lowerLimit <= $upperLimit)
{
echo "$num x $lowerLimit =" . ($num*$lowerLimit);
$lowerLimit++;
}
?>
```

Looping Statements-do while

```
<?php
// define number and limits for multiplication tables
$num = 11;
$upperLimit = 10;
$lowerLimit = 12;

// loop and multiply to create table
do
{
    echo "$num x $lowerLimit =" . ($num*$lowerLimit);
    $lowerLimit++;
} while ($lowerLimit <= $upperLimit)
?>
```

Looping Statements-for

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



Arrays

An array is a data structure that stores one or more similar type of values in a single value. For example if we want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

There are three different kind of arrays and each array value is accessed using an ID which is called array index.

Numeric array - An array with a numeric index. Values are stored and accessed in linear fashion

Associative array - An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.

Multidimensional array - An array containing one or more arrays and values are accessed using multiple indices

Numeric Array

- These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

```
<html>
<body>
<?php
/* First method to create array. */
$numbers = array( 1, 2, 3, 4, 5);
foreach( $numbers as $value )
{
    echo "Value is $value <br />";
}
/* Second method to create array. */
$numbers[0] = "one";
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";

foreach( $numbers as $value )
{
    echo "Value is $value <br />";
}
?>
</body>
</html>
```

The foreach loop is used to loop through arrays

Associative Array

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that we can establish a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

Example

```
<html>
<body>
<?php
/* First method to associate create array. */
$salaries = array(
    "mohammad" => 2000,
    "qadir" => 1000,
    "zara" => 500
);

echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
echo "Salary of qadir is ". $salaries['qadir'] . "<br />";
echo "Salary of zara is ". $salaries['zara'] . "<br />";

/* Second method to create array. */
$salaries['mohammad'] = "high";
$salaries['qadir'] = "medium";
$salaries['zara'] = "low";

echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
echo "Salary of qadir is ". $salaries['qadir'] . "<br />";
echo "Salary of zara is ". $salaries['zara'] . "<br />";
?>
</body>
</html>
```

Multidimensional Array

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

```
<html>
<body>
<?php
    $marks = array(
        "mohammad" => array
            (
                "physics" => 35,
                "maths" => 30,
                "chemistry" => 39
            ),
        "qadir" => array
            (
                "physics" => 30,
                "maths" => 32,
                "chemistry" => 29
            ),
    );
    /* Accessing multi-dimensional array values */
    echo "Marks for mohammad in physics : " ;
    echo $marks["mohammad"]["physics"] . "<br />";
    echo "Marks for qadir in maths : ";
    echo $marks["qadir"]["maths"] . "<br />";
?>
</body>
</html>
```

Strings

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

```
$string_1 = "This is a string in double quotes";  
$string_2 = "This is a somewhat longer, singly quoted string";  
$string_39 = "This string has thirty-nine characters";  
$string_0 = ""; // a string with zero characters  
<?  
$variable = "name";  
$literally = 'My $variable will not print!\n';  
print($literally);  
$literally = "My $variable will print!\n";  
print($literally);  
?>
```

This will produce following result:

```
My $variable will not print!\n  
My name will print
```

Functions: strlen, strtolower, strtoupper, strncmp, substr,

Web Concepts

Identifying Browser & Platform

PHP creates some useful environment variables that can be seen in the `phpinfo.php` page that was used to setup the PHP environment.

One of the environment variables set by PHP is `HTTP_USER_AGENT` which identifies the user's browser and operating system.

PHP provides a function `getenv()` to access the value of all the environment variables. The information contained in the `HTTP_USER_AGENT` environment variable can be used to create dynamic content appropriate to the browser.

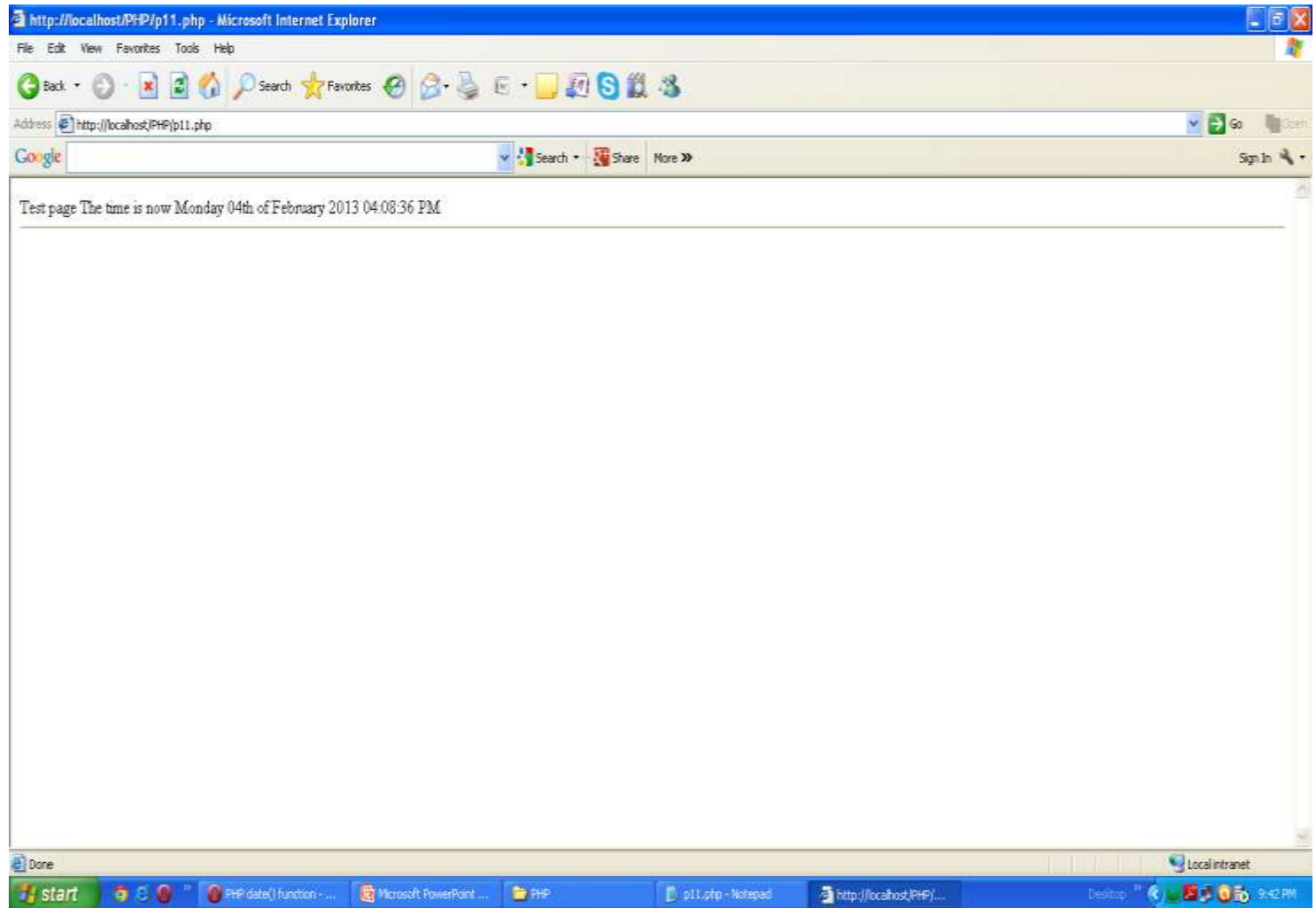
Example

```
<html>
<body>
<?php
    $viewer = getenv( "HTTP_USER_AGENT" );
    $browser = "An unidentified browser";
    if( preg_match( "/MSIE/i", "$viewer" ) )
    {
        $browser = "Internet Explorer";
    }
    else if( preg_match( "/Netscape/i", "$viewer" ) )
    {
        $browser = "Netscape";
    }
    else if( preg_match( "/Mozilla/i", "$viewer" ) )
    {
        $browser = "Mozilla";
    }
    $platform = "An unidentified OS!";
    if( preg_match( "/Windows/i", "$viewer" ) )
    {
        $platform = "Windows!";
    }
    else if ( preg_match( "/Linux/i", "$viewer" ) )
    {
        $platform = "Linux!";
    }
    echo("You are using $browser on $platform");
?>
</body>
</html>
```

Another example

```
<html>
  <head>Test page</head>
  <body>
    The time is now
    <?php
      echo(date("l dS \of F Y h:i:s A") . "<br />");
    ?>
    <hr>
  </body>
</html>
```

Output:



Using HTML Forms

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to our PHP scripts.

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

The PHP default variable `$_PHP_SELF` is used for the PHP script name and when you click "submit" button then same PHP script will be called

Processing HTML Forms- welcome.html

```
<html>
```

```
<body>
```

```
<form action="welcome.php" method="post">
```

```
Name: <input type="text" name="name">
```

```
Age: <input type="text" name="age">
```

```
<input type="submit">
```

```
</form>
```

```
</body>
```

```
</html>
```

Welcome.php

```
<html>
```

```
<body>
```

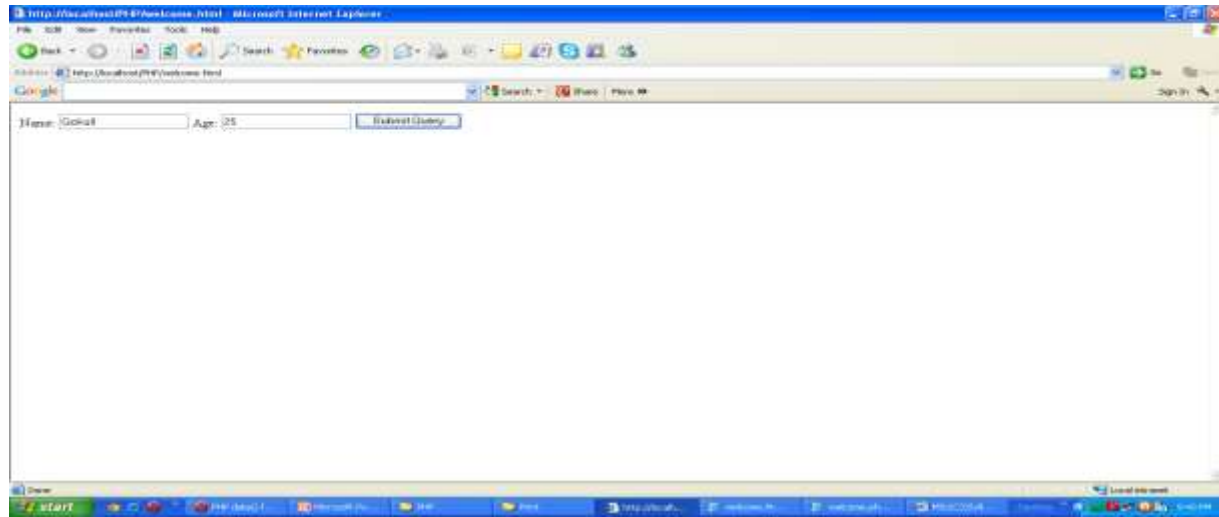
```
Welcome <?php echo $_POST["name"]; ?>.<br>
```

```
You are <?php echo $_POST["age"]; ?> years old.
```

```
</body>
```

```
</html>
```

Output



Get & Post Method

There are two ways the browser client can send information to the web server.

The GET Method

The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

`name1=value1&name2=value2&name3=value3`

Spaces are removed and replaced with the + character and any other nonalphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

Get Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? character.
`http://www.test.com/index.htm?name1=value1&name2=value2`

The GET method produces a long string that appears in your server logs, in the browser's Location: box.

The GET method is restricted to send upto 1024 characters only.

Never use GET method if we have password or other sensitive information to be sent to the server.

GET can't be used to send binary data, like images or word documents, to the server.

The data sent by GET method can be accessed using QUERY_STRING environment variable.

The PHP provides `$_GET` associative array to access all the sent information using GET method.

Example

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

Post Method

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

The POST method does not have any restriction on data size to be sent.

The POST method can be used to send ASCII as well as binary data.

The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP we can make sure that your information is secure.

The PHP provides `$_POST` associative array to access all the sent information using GET method.

Example

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

    Name: <input type="text" name="name" />
    Age: <input type="text" name="age" />

    <input type="submit" />
    </form>
</body>
</html>
```

\$_REQUEST variable

The PHP \$_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

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

    Name: <input type="text" name="name" />
    Age: <input type="text" name="age" />

    <input type="submit" />
    </form>
</body>
</html>
```

Functions

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

There are two part.

Creating a PHP Function

Calling a PHP Function

Creating PHP Function

Its very easy to create our own PHP function. Suppose we want to create a PHP function which will simply write a simple message on our browser when we will call it. Following example creates a function called writeMessage() and then calls it just after creating it.

```
<html>
<head>
<title>Writing PHP Function</title>
</head>
<body>

<?php
/* Defining a PHP Function */
function writeMessage()
{
    echo "You are really a nice person, Have a nice time!";
}
/* Calling a PHP Function */
writeMessage();
?>
</body></html>
```

Function with parameters

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside our function.

Following example takes two integer parameters and add them together and then print them.

```
<html>
<head>
<title>Writing PHP Function with Parameters</title>
</head>
<body>

<?php
function addFunction($num1, $num2)
{
    $sum = $num1 + $num2;
    echo "Sum of the two numbers is : $sum";
}
addFunction(10, 20);
?>
</body>
</html>
```

Passing Arguments by Reference

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.

```
<html><head>
<title>Passing Argument by Reference</title>
</head><body>
<?php
function addFive($num)
{
    $num += 5;
}
function addSix(&$num)
{
    $num += 6;
}
$orignum = 10;
addFive( &$orignum );
echo "Original Value is $orignum<br />";
addSix( $orignum );
echo "Original Value is $orignum<br />";
?></body></html>
```

PHP Functions returning value:

A function can return a value using the return statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.

We can return more than one value from a function using return array(1,2,3,4).

Example

```
<html>
<head>
<title>Writing PHP Function which returns value</title>
</head>
<body>

<?php
function addFunction($num1, $num2)
{
    $sum = $num1 + $num2;
    return $sum;
}
$return_value = addFunction(10, 20);
echo "Returned value from the function : $return_value"
?>
</body>
</html>
```

Error Handling

- Die()
- Custom Error Handling Function
`error_function(error_level,error_message,
error_file,error_line,error_context);`

Example

```
<?php
if(!file_exists("/tmp/test.txt"))
{
die("File not found");
}
else
{
$file=fopen("/tmp/test.txt","r");
print "Opend file sucessfully";
}

// Test of the code here. ?>
```

Exception Handling

- Try
 - Catch
 - throw
-

Example

```
<?php
try {
$error = 'Always throw this error';
throw new Exception($error);
// Code following an exception is not executed.
echo 'Never executed';
}
catch (Exception $e)
{
echo 'Caught exception: ', $e->getMessage(), "\n"; }
// Continue execution
echo 'Hello World';
?>
```

Cookies

Cookies are text files stored on the client computer and they are kept of use tracking purpose. PHP transparently supports HTTP cookies.

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.

Setting Cookies with PHP:

PHP provided `setcookie()` function to set a cookie. This function requires upto six arguments and should be called before `<html>` tag. For each cookie this function has to be called separately.

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

Here is the detail of all the arguments:

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("name", "John Watkin", time()+3600, "/", "", 0);
    setcookie("age", "36", time()+3600, "/", "", 0);
?>
<html>
<head>
<title>Setting Cookies with PHP</title>
</head>
<body>
<?php echo "Set Cookies"?>
</body>
</html>
```

Accessing Cookies with PHP

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

```
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?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 />";
?>
</body>
</html>
```

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

```
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
    if( isset($_COOKIE["name"]))
        echo "Welcome " . $_COOKIE["name"] . "<br />";
    else
        echo "Sorry... Not recognized" . "<br />";
?>
</body>
</html>
```

Deleting Cookie with PHP

Officially, to delete a cookie you should call `setcookie()` with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired:

```
<?php
    setcookie( "name", "", time()- 60, "/", "", 0);
    setcookie( "age", "", time()- 60, "/", "", 0);
?>
<html>
<head>
<title>Deleting Cookies with PHP</title>
</head>
<body>
<?php echo "Deleted Cookies" ?>
</body>
</html>
```

Session

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

Starting a PHP Session:

A PHP session is easily started by making a call to the `session_start()` function. This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to `session_start()` at the beginning of the page.

Session variables are stored in associative array called `$_SESSION[]`. These variables can be accessed during lifetime of a session.

Example

The following example starts a session then register a variable called counter that is incremented each time the page is visited during the session. Make use of `isset()` function to check if session variable is already set or not

```
<?php
    session_start();
    if( isset( $_SESSION['counter'] ) )
    {
        $_SESSION['counter'] += 1;
    }
    else
    {
        $_SESSION['counter'] = 1;
    }
    $msg = "You have visited this page ". $_SESSION['counter'];
    $msg .= "in this session.";
?>
<html>
<head>
<title>Setting up a PHP session</title>
</head>
<body>
<?php echo ( $msg ); ?>
</body>
</html>
```

Destroying a PHP Session:

A PHP session can be destroyed by `session_destroy()` function. This function does not need any argument and a single call can destroy all the session variables. If we want to destroy a single session variable then we can use `unset()` function to unset a session variable.

Here is the example to unset a single variable:

```
<?php
    unset($_SESSION['counter']);
?>
```

Here is the call which will destroy all the session variables

```
:<?php
    session_destroy();
?>
```

Object Oriented Programming in PHP

The general form for defining a new class in PHP is as follows:

```
<?php
class phpClass{
    var $var1;
    var $var2 = "constant string";
    function myfunc ($arg1, $arg2) {
        [..]
    }
    [..]
}
?>
```

Here is the description of each line:

The special form class, followed by the name of the class that you want to define.

A set of braces enclosing any number of variable declarations and function definitions.

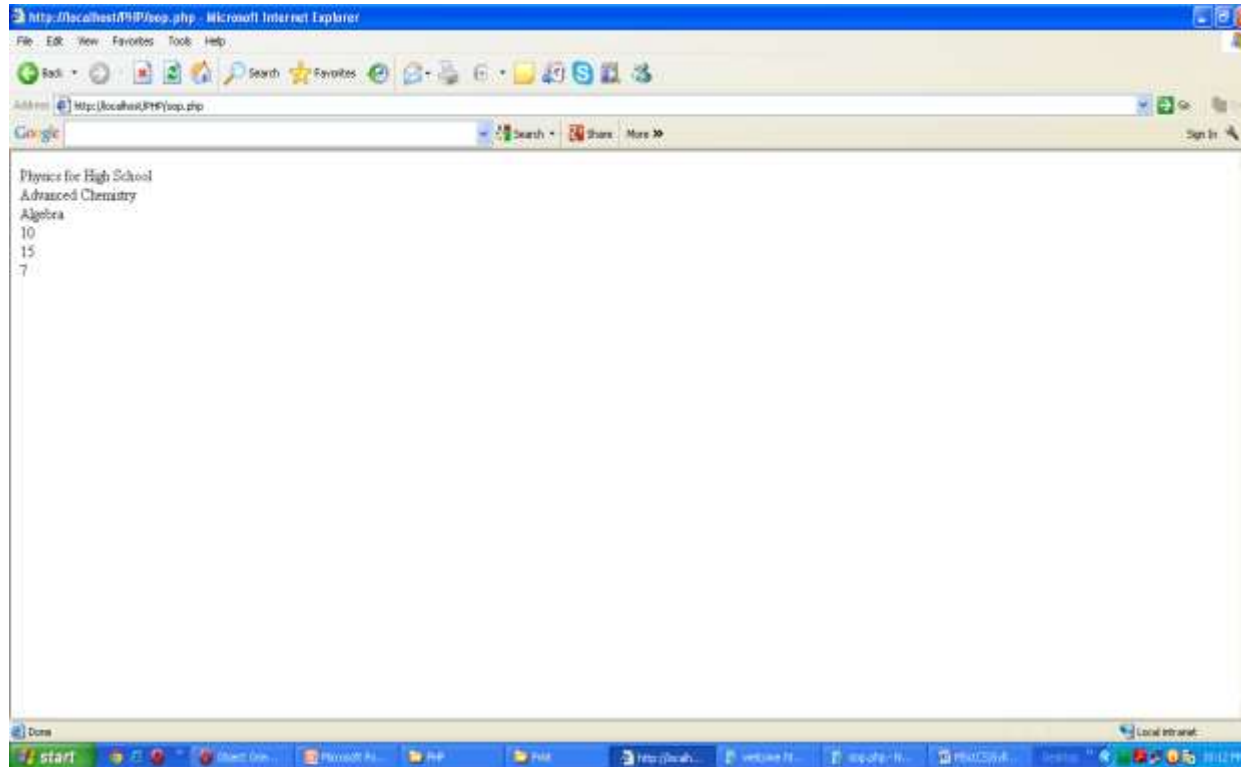
Variable declarations start with the special form var, which is followed by a conventional \$ variable name; they may also have an initial assignment to a constant value.

Function definitions look much like standalone PHP functions but are local to the class and will be used to set and access object data.

Class

```
<?php
class Books{
    /* Member variables */  var $price;  var $title;
    /* Member functions */
    function setPrice($par){
        $this->price = $par;  }
    function getPrice(){
        echo $this->price ."<br/>";  }
    function setTitle($par){
        $this->title = $par;  }
    function getTitle(){
        echo $this->title ." <br/>";  }  }
$physics = new Books;  $maths = new Books;  $chemistry = new Books;
$physics->setTitle( "Physics for High School" );  $chemistry->setTitle( "Advanced Chemistry" );
    $maths->setTitle( "Algebra" );
    $physics->setPrice( 10 );  $chemistry->setPrice( 15 );  $maths->setPrice( 7 );
$physics->getTitle();  $chemistry->getTitle();  $maths->getTitle();
    $physics->getPrice();  $chemistry->getPrice();  $maths->getPrice();
?>
```

Output:



PHP-MySQL

PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database

Opening Database Connection:

PHP provides `mysql_connect` function to open a database connection. This function takes five parameters and returns a MySQL link identifier on success, or `FALSE` on failure.

Syntax: `connection mysql_connect(server,user,passwd,new_link,client_flag);`

`server` Optional - The host name running database server. If not specified then default value is `localhost:3036`.

`user` Optional - The username accessing the database. If not specified then default is the name of the user that owns the server process.

`passwd` Optional - The password of the user accessing the database. If not specified then default is an empty password.

`new_link` Optional - If a second call is made to `mysql_connect()` with the same arguments, no new connection will be established; instead, the identifier of the already opened connection will be returned.

`client_flags` Optional - A combination of the following constants:

`MYSQL_CLIENT_SSL` - Use SSL encryption

`MYSQL_CLIENT_COMPRESS` - Use compression protocol

`MYSQL_CLIENT_IGNORE_SPACE` - Allow space after function names

`MYSQL_CLIENT_INTERACTIVE` - Allow interactive timeout seconds of inactivity before closing the connection

Closing Database Connection:

Its simplest function `mysql_close` PHP provides to close a database connection. This function takes connection resource returned by `mysql_connect` function. It returns `TRUE` on success or `FALSE` on failure.

Syntax:bool

```
mysql_close ( resource $link_identifier );
```

If a resource is not specified then last opened database is closed.

Example

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = '' ;
$conn = mysqli_connect($dbhost, $dbuser, $dbpass, 'db1');
if(! $conn )
{
    die('Could not connect: ' . mysqli_error());
}
echo 'Connected successfully';
mysqli_close($conn);
?>
```

Creating a Database:

To create and delete a database we should have admin privilege. Its very easy to create a new MySQL database. PHP uses `mysql_query` function to create a MySQL database. This function takes two parameters and returns `TRUE` on success or `FALSE` on failure.

Syntax:`bool mysql_query(sql, connection);`

`sql` Required - SQL query to create a database

`connection` Optional - if not specified then last opened connection by `mysql_connect` will be used.

Example

```
<?php
$dbhost = 'localhost:3306';
$dbuser = 'root';
$dbpass = '';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
$sql = 'CREATE Database test_db';
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not create database: ' . mysql_error());
}
echo "Database test_db created successfully\n";
mysql_close($conn);
?>
```

Selecting a Database:

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

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

PHP provides function `mysql_select_db` to select a database. It returns `TRUE` on success or `FALSE` on failure.

Syntax: `bool mysql_select_db(db_name, connection);`

`db_name` Required - Database name to be selected

`connection` Optional - if not specified then last opened connection by `mysql_connect` will be used.

Example

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = '' ;
$conn = mysqli_connect($dbhost, $dbuser, $dbpass, 'db1');
if(! $conn )
{
    die('Could not connect: ' . mysqli_error());
}
echo 'Connected successfully';
mysqli_close($conn);
?>
```

Creating Database Tables:

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = "";
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
$sql = 'CREATE TABLE employee( ' .
    'emp_id INT NOT NULL AUTO_INCREMENT, ' .
    'emp_name VARCHAR(20) NOT NULL, ' .
    'emp_address VARCHAR(20) NOT NULL, ' .
    'emp_salary INT NOT NULL, ' .
    'join_date timestamp(14) NOT NULL, ' .
    'primary key ( emp_id ))';

mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not create table: ' . mysql_error());
}
echo "Table employee created successfully\n";
mysql_close($conn);
?>
```

Insert data into MySQL

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = " ";
$conn = mysqli_connect($dbhost, $dbuser, $dbpass, 'db1');
if(! $conn )
{
    die('Could not connect: ' . mysqli_error());
}
echo 'Connected successfully';

$sql = 'INSERT INTO Login ' .
    '( UserID, Pwd ) ' .
    'VALUES ( "admin", "india" )';

$retval = mysqli_query( $conn , $sql );
if(! $retval )
{
die('Could not enter data: ' . mysqli_error());
}
echo 'Inserted successfully';
```

Getting Data From MySQL Database

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';

mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "EMP ID :{$row['emp_id']} <br> ".
        "EMP NAME : {$row['emp_name']} <br> ".
        "EMP SALARY : {$row['emp_salary']} <br> ".
        "-----<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>
```

Updating Data into MySQL Database

```
<html>
<head>
<title>Update a Record in MySQL Database</title>
</head>
<body>

<?php
if(isset($_POST['update']))
{
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}

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

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

mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not update data: ' . mysql_error());
}

echo "Updated data successfully\n";
mysql_close($conn);
}
```

```
else
{
?>
<form method="post" action="<?php $_PHP_SELF ?>">
<table width="400" border="0" cellspacing="1" cellpadding="2">
<tr>
<td width="100">Employee ID</td>
<td><input name="emp_id" type="text" id="emp_id"></td>
</tr>
<tr>
<td width="100">Employee Salary</td>
<td><input name="emp_salary" type="text" id="emp_salary"></td>
</tr>
<tr>
<td width="100"> </td>
<td> </td>
</tr>
<tr>
<td width="100"> </td>
<td>
<input name="update" type="submit" id="update" value="Update">
</td>
</tr>
</table>
</form>
<?php
}
?>
</body>
</html>
```

Deleting Data from MySQL Database

```
<?php
if(isset($_POST['delete']))
{
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}

$emp_id = $_POST['emp_id'];

$sql = "DELETE employee ".
    "WHERE emp_id = $emp_id" ;

mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not delete data: ' . mysql_error());
}
echo "Deleted data successfully\n";
mysql_close($conn);
```