

Table of Contents

1.	<u>Configuring & Installation</u>	3
1.1	<i>Web Architecture -</i>	3
1.2	<i>Important Concepts about Web Development -</i>	4
1.3	<i>Introduction of PHP -</i>	7
1.4	<i>Installation of PHP -</i>	8
1.5	<i>Configuration Of PHP & apache Server -</i>	8
1.6	<i>Features of PHP Languages -</i>	11
1.7	<i>Version of PHP -</i>	11
2.	<u>PHP Language Basics</u>	13
2.1	<i>Structure & syntax -</i>	13
2.2	<i>PHP with html -</i>	15
2.3	<i>Constant & Variable -</i>	17
2.4	<i>Passing variable Value from one page to another page -</i>	22
2.5	<i>Operator in PHP -</i>	23
2.6	<i>Conditional Statements -</i>	30
2.7	<i>PHP Loop Types</i>	33
2.8	<i>Include</i>	38
2.9	<i>Functions</i>	39
2.10	<i>Array</i>	45
2.11	<i>Using \$_GET & \$_POST For Working with Forms -</i>	48
2.12	<i>Validating forms with PHP</i>	51
2.13	<i>Validation Emailing from data -</i>	53
2.14	<i>Email validation within a form in PHP.....</i>	54
2.15	<i>PHP Error Handling.....</i>	54
2.16	<i>PHP Files & I/O</i>	59
3.	<u>Using PHP with MySql.....</u>	64
3.1	<i>Introduction of MySql.....</i>	64
3.2	<i>Installation & Configuration of MySql.....</i>	65
3.3	<i>MySQL Data Types</i>	67
3.4	<i>DDL Operations on Database using PHP Script</i>	69
3.5	<i>Data Manipulation Operation With PHP Script -</i>	75
3.6	<i>PHP mysql_fetch_array() Function.....</i>	82
3.7	<i>Perform MySQL backup using PHP.....</i>	83
4.	<u>Object Oriented Concepts in PHP.....</u>	85
4.1	<i>Introduction.....</i>	85
4.2	<i>Advantages of Object Oriented PHP -</i>	85
4.3	<i>Class & object</i>	85
4.4	<i>Data Member</i>	87
4.5	<i>Constructor & Destructor Functions:.....</i>	90
4.6	<i>Inheritance -</i>	91
4.7	<i>Access Specifiers</i>	92
4.8	<i>Final Class & Method.....</i>	95
4.9	<i>Interfaces.....</i>	96
4.10	<i>Abstract Classes</i>	96

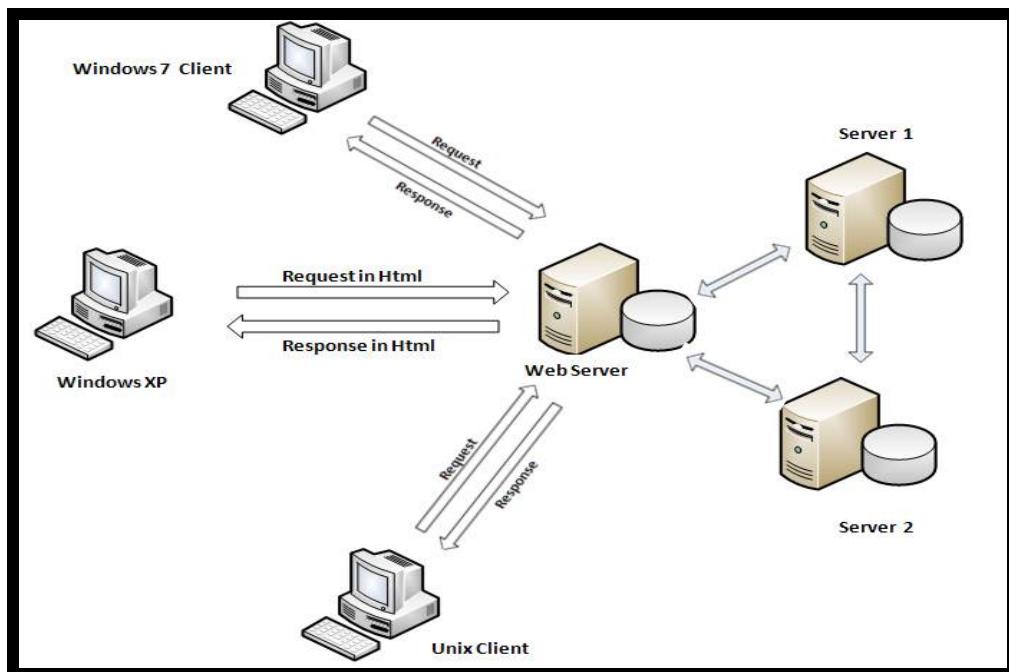
4.11	<i>Exceptions Handling</i>	97
5.	Advance PHP -	100
5.1	<i>Mailing In PHP -</i>	100
5.2	<i>Sending Free SMS to Mobile with PHP script -.....</i>	102
5.3	<i>File Uploading</i>	103
5.4	<i>Loading PHP application on web server by ftp Service -</i>	105
5.5	<i>Web Services –.....</i>	108
6.	Ajax with PHP	113
6.1	<i>Introduction Of Ajax.....</i>	113
6.2	<i>How AJAX Works.....</i>	114
6.3	<i>Steps of AJAX Operation.....</i>	115
6.4	<i>Ajax object in Different Browser –.....</i>	121
7.	CMS Technology in PHP	123
7.1	<i>Introduction of CMS.....</i>	123
7.2	<i>What is Joomla?.....</i>	123
7.3	<i>Installation of joomla-</i>	124
7.4	<i>Steps of Joomla installation & Configuration</i>	124
7.5	<i>Various Manager in Joomla.....</i>	130
7.6	<i>Installing an plug-in/extension</i>	139
Web Resources.....		1
Bibliography.....		1
INDEX		Error! Bookmark not defined.

1. Configuring & Installation

1.1 Web Architecture –

The World Wide Web is a global network that is based on the Internet. There are two important properties to its "client-server" architecture that allow it to function. The first is a set of file servers which store text, images, audio, video and other shared elements. The file servers send this data by way of the Internet to clients.

Clients are the second important feature to the Web architecture, and they exist on the user's end of the network - a.k.a. your computer. The client sends commands to the file servers to retrieve stored data, and then displays it on the user's end. The display of this data is handled through a web browser, such as Netscape Communicator or Microsoft Internet Explorer.



All of these interactions work through a hypermedia system, and hypertext, which is a term that was made up by Ted Nelson around 1965. Hypermedia is a node-link architecture, where the nodes are web pages or media (such as video, audio, and images), and the links connect all of these together through a pattern of association. Hypertext only links documents together.

The World Wide Web contains a group of protocols that enable it to function. The first of these is HTTP: Hypertext Transfer Protocol. This protocol is used by clients and servers to communicate.

This is just a brief review of how the web works. To fully understand this global hypermedia based network, it is essential to first look at the early ideas of information storage and hypertext

1.2 Important Concepts about Web Development -

Web Server –

Web server can refer to either the hardware (the computer) or the software (the computer application) that helps to deliver content that can be accessed through the Internet.

The most common use of web servers is to host web sites but there are other uses such as data storage or running enterprise applications.

The primary function of a web server is to deliver web pages on the request to clients. This means delivery of HTML documents and any additional content that may be included by a document, such as images, style sheets and scripts. While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files. Many generic web servers also support server-side scripting, e.g., Active Server Pages (ASP) and PHP. This means that the behavior of the web server can be scripted in separate files, This is referred to as dynamic and static content respectively. The former is primarily used for retrieving and/or modifying information from databases.

In simple word, “web server is software is to run the web application it handle the request from client & send the response to client.” E.g. - apache server, IIS web server. Etc.

Apache - is web server software notable for playing a key role in the initial growth of the World Wide Web. In 2009 it became the first web server software to surpass the 100 million website milestone. Apache was the first viable alternative to the Netscape Communications Corporation web server (currently named Oracle iPlanet Web Server), and since has evolved to rival other web servers in terms of functionality and performance.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The application is available for a wide variety of operating systems, including Unix, GNU, FreeBSD, Linux, Solaris, Novell NetWare, AmigaOS, Mac OS X, Microsoft Windows, OS/2, TPF, and eComStation. Released under the Apache License, Apache is open-source software.

Web Browser –

A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web. An information resource is identified by a Uniform Resource Identifier (URI) and may be a web page, image, video, or other piece of content. Hyperlinks present in resources enable users easily to navigate their browsers to related resources. A web browser can also be defined as an application software or program designed to enable users to access, retrieve and view documents and other resources on the Internet.

Today's browsers are fully-functional software suites that can interpret and display HTML Web pages, applications, JavaScript, AJAX and other content hosted on Web servers. Many browsers offer plug-ins which extend the capabilities of a browser so it can display multimedia information (including sound and video), or the browser can be used to perform tasks such as videoconferencing, to design web pages or add anti-phishing filters and other security features to the browser.

The two most popular browsers are Microsoft Internet Explorer and Firefox. Other major browsers include Google Chrome, Apple Safari and Opera. While most commonly used to access information on the web, a browser can also be used to access information hosted on web servers in private networks.

Also, there are a number of browsers that are designed to access the Web using a mobile device. These mobile browsers ("Micro browser") are optimized to display Web content on smaller mobile device screens and to also perform efficiently on these devices which have far less computing power and memory capacity as Desktop or laptop computers

Open Source –

Open source refers to a program or software in which the source code is available to the general public for use and/or modification from its original design free of charge. Open source code is typically created as a collaborative effort in which programmers improve upon the code and share the changes within the community.

The Open Source Initiative (OSI)

Open Source is a certification standard issued by the Open Source Initiative (OSI) that indicates that the source code of a computer program is made available free of charge to the general public. OSI dictates that in order to be considered "OSI. Some Open Source Software for Web Development – World Press, Joomla, Drupal, OSCommerce.

Proprietary -

Privately owned and controlled. In the computer industry, proprietary is the opposite of open. A proprietary design or technique is one that is owned by a company. It also implies

that the company has not divulged specifications that would allow other companies to duplicate the product.

Increasingly, proprietary architectures are seen as a disadvantage. Consumers prefer open and standardized architectures, which allow them to mix and match products from different manufacturers.

Some Proprietary Software for Web Development –

- IIS By Microsoft, Flex, flash by Micromedia etc.

CSS3, HTML5 and Fonts as a Service -

CSS3, HTML5, and Fonts as a Service such as Typekit that cater to web browsers that already support the @font-face rule, are giving web designers the creative freedom that they have been coveting for a long time.

Social media -

No one can deny that 2009 has been a big year for social media: Twitter, for example, has become the buzzword in many a boardroom and office. It's obvious that it will continue to a big part of the web in the future.

In many ways, the growth of platforms such as Twitter and Facebook has led the web to be much more community-oriented. Big changes could happen within social media and, no doubt, will be led by monetization of the media.

jQuery -

jQuery is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML. It was released in January 2006 at BarCamp NYC by John Resig. Used by over 49% of the 10,000 most visited websites, jQuery is the most popular JavaScript library in use today.

jQuery is free, open source software, dual-licensed under the MIT License or the GNU General Public License, Version 2. jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications.

Web Hosting –

If you want other people to view your web site, you must copy your site to a public server. Even if you can use your own PC as a web server, it is more common to let an Internet Service Provider (ISP) host your site.

Included in a Web hosting solution you can expect to find domain name registration and standard email services

A domain name is a unique name for a web site, like www.basponcollege.org. Domain names must be registered. When domain names are registered, they are added to a large domain name register. In addition, information about the web site, including the IP address, is stored on a DNS server.

DNS stands for Domain Name System. A DNS server is responsible for informing all other computers on the Internet about the domain name and the web site address.

Registering a Domain

Domains can be registered from domain name registration companies. These companies provide interfaces to search for available domain names, and they offer a variety of domain name extensions that can be registered at the same time.

Client side script –

The script which is executed on client side system is called as client side scripting. E.g.- HTML, JAVA script etc.

Server Side Script –

The script which is executed on web server user can see only o/p of the script. E.g.- JSP, asp, etc.

1.3 Introduction of PHP –

PHP is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications. PHP can be deployed on most web servers and as a standalone interpreter, on almost every operating system and platform free of charge. There is also commercial software such as RadPHP, a rapid application development framework for the PHP language. A competitor to Microsoft's Active Server Pages (ASP) server-side script engine and similar languages, PHP is installed on more than 20 million websites and 1 million web servers.

PHP was originally created by Rasmus Lerdorf in 1995. The main implementation of PHP is now produced by The PHP Group and serves as the de facto standard for PHP as there is no formal specification. PHP is free software released under the PHP License which is incompatible with the GNU General Public License (GPL) due to restrictions on the usage of the term PHP.

While PHP originally stood for "**Personal Home Page**", it is now said to stand for "**PHP: Hypertext Preprocessor**".

PHP development began in 1994 when the Danish/Greenlandic/Canadian programmer Rasmus Lerdorf initially created a set of Perl scripts he called "Personal Home Page Tools" to maintain his personal homepage. The scripts performed tasks such as displaying his résumé and recording his web-page traffic. Lerdorf initially announced the release of PHP on the comp.infosystems. www.authoring.cgi Usenet discussion group on June 8, 1995.

Zeev Suraski and Andi Gutmans, two Israeli developers at the Technion IIT, rewrote the parser in 1997 and formed the base of PHP 3, changing the language's name to the recursive initialism PHP: Hypertext Preprocessor. Afterwards, public testing of PHP 3 began, and the official launch came in June 1998. Suraski and Gutmans then started a new rewrite of PHP's core, producing the Zend Engine in 1999. They also founded Zend Technologies in Ramat Gan, Israel.

1.4 Installation of PHP -

WAMPP –

wampp is Environment to work with PHP Application. It's Providing PHP Software Along with MySQL Database, Apache server. It Stands for Windows, Apache, MySQL, PHP, & Perl. It supports windows operating system.

XAMPP –

a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.

XAMPP's name is an acronym for:

- **X** (to be read as "cross", meaning cross-platform)
- **Apache** HTTP Server
- **MySQL**
- **PHP**
- **Perl**

The program is released under the terms of the GNU General Public License and acts as a free web server capable of serving dynamic pages. XAMPP is available for Microsoft Windows, Linux, Solaris, and Mac OS X, and is mainly used for web development projects. This software is useful while you are creating dynamic web pages using programming languages like PHP, JSP, Servlets.

1.5 Configuration Of PHP & apache Server -

- **htdocs** – It is folder in XAMPP to save all php Programs.

- **temp** – This is the temporary location to store session, uploaded file etc.
- **php** – This folder Contain the all Configuration setting of PHP.
- **apache** – This folder contain the apache Configuration settings.
- **mysql** – This folder contain the MySQL software along with database settings.

Php.ini –

This file Contain the php configuration directive. It available in Xampp /php folder

- It is a configuration file and control some of the setting of the PHP interpreter.
- The configuration file (php.ini) is read when PHP starts.
- Where a configuration setting may be set OR in how many ways you can set your configuration parameters.
- Following modes determine when and where a PHP directive may or may not be set .
- Mode Meaning
- PHP_INI_USER Entry can be set in user scripts using function like ini_set().
- PHP_INI_PERDIR Entry can be set in php.ini, .htaccess or httpd.conf
- PHP_INI_SYSTEM Entry can be set in php.ini or httpd.conf PHP_INI_ALLEntry can be set anywhere
- In this presentation all the php.ini setting can be set in user's script.

Basic Important Settings in PHP.ini

- Language Options
- Resource Limit
- Data Handling
- File Uploads
- Others

Language Options

- 1) **short_open_tag** : Tell whether to use short form (<? ?>) .If you disabled then you must use the long form of the PHP tags i.e (<?php ?>).By default it is set to "1".
 - a. Description : <? echo "Hello reader" ?> is short-form PHP tags
 - b. <?php echo "Hello Reader" ?> is long form PHP tags
- 2) **asp_tags** : Enable the use of ASP-like <% %> tags in addition to <?php ?> tags.
- 3) **Description** : This tag setting is kept to make easiness for developers who started with PHP after ASP.

Resource Limit

- 1) **memory_limit** : Maximum amount of memory a script may consume. Default value is 128 MB.

- 2) **max_execution_time** : Maximum execution time of each script, in seconds. Default value is set to 30 sec.

Data Handling

- 1) **register_globals** : Whether or not to register the EGPCS variables as global variables. By default it is set to "Off".
- 2) **Description:** Consider a Form field <input name="name" type="text">. When form is submitted via POST the form value for name will be fetched as show below.

```
PHP code when register_globals = On  
$myName = $name;  
PHP code when register_globals = Off  
$myName = $_POST['name'];
```

- 3) **post_max_size** : Maximum size of POST data that PHP will accept or you can POST through form. Default values is 8M. This setting is helpful for file uploads.

File Uploads

- 1) **file_uploads** : Whether or not to allow HTTP file uploads. Default value is "On".
- 2) **upload_tmp_dir** : The temporary directory used for storing files when doing file upload. Will use system default if not specified. By default it is empty.
- 3) **upload_max_filesize** : The maximum size of an uploaded file. Default value is 2M.

Others

- 1) **allow_url_fopen** : Whether to allow the treatment of URLs (like http:// or ftp://) as files. Default value is "ON".
- 2) **Description:**

```
if allow_url_fopen =Off  
file_get_content( http://www.sitename.com/test.txt ) will not work !!
```

- 3) **session.cache_expire** : Document expires after n minutes. Default value is 180.
- 4) **session.gc_maxlifetime** : After this number of seconds, stored data will be seen as 'garbage' and cleaned up by the garbage collection process. Default value is 1440.
- 5) **session.save_handler** : Handler used to store/retrieve session data. Default value is files

Httpd.conf –

This file contain the apache directive it is available in xampp/apche/conf folder.

- To change the document root of the file , open httpd.conf file & change the value of Document root Directive. The default value is C:\Xampp\httdocs change this value with new path.
- To change the port number of apache server we have to change the value of 2 directive in httpd.conf file those are "listen" & "server name local host" than restart the server to reselect this value.

1.6 Features of PHP Languages -

1. PHP is cross platform
2. PHP is cross server
3. Zend engine execute the PHP application very fast.
4. In PHP script we can easily embedded external script like VB Script, JavaScript , ASP Script, JSP etc
5. PHP support different types of database servers, for that PHP provides inbuilt library function to interact with any types of server.
6. PHP 5.0 support object oriented Programming Concepts. Like. Inheritance, interface, abstracts classes etc.
7. PHP support different types of CMS(Contain management system) Technologies like joomla, drupal, ecommerce etc.
8. Zend organization provides completes support to the PHP technology & they Introduce no of tools, editors to work with PHP.
9. PHP is open source.
10. PHP is providing no. of security concept like encryption authentication, authorization etc.

1.7 Version of PHP –

- PHP 1.0 (Form Interpreter) – 1995 – it is used to handle the server side scripting request. It is not a server side script implementation. invented in 1995
- PHP 2.0 – Nov 1997 – Introduced as server side scripting language collection of login passwords access. Supports any types of database server, support files uploads & many function implemented by rossmoss lerdorgs .
- PHP 3.0 – June 1998 – zev surlesky, andy gosmoss this 2 israel programmer rewritten PHP & they release new version i.e. 3.0 .This programmer started organization with the name ZEND & they named "**personal home page**" as "**HyperText Preprocessor**". PHP 3.0 is completely server side scripting. It support multi platform. Support mail function. Easy Interaction with any type of database.
- PHP 4.0 – may 2000 – PHP 4.0 introduce zend engine. It is cross platform & cross server. It include advance feature like Http, session, O/P Buffering , support flash, xml, pdf etc.. it introduced many template system which is using to develop the web application very fast using templates.

- PHP 5.0(2005) – it include more object oriented concept like abstract, public, protected, private variable, abstract classes, interfaces & exception handling. Zend Engine 2.0 was introduced. MySQL library is introduced which gives more support to MySQL database. It supports exception handling. Improved the web services & xml Supports.
- PHP 6.0 – it is major upgrades to PHP language. The main focus is native Unicode support means programmers can able to name in function & cross in their own native language it support some advance oops concepts such as namespaces, late static binding it include new class like xmlwriter which add new functionality to PHP application.

2. PHP Language Basics

First Program of PHP –

- Write a following code with help of any Text Editor.

```
<?php  
    print "Hello World";  
?>
```

- Save this File with <File Name>.php Extension in htdocs folder of XAMPP folder.
- Open the browser & type [Http://localhost/<filename>.php](http://localhost/<filename>.php) & press enter.
(make shure that before running the php Program apache server is running or not)

2.1 Structure & syntax –

Different tag styles are available. This is the short style. If you have some problem running this script, it might be because short tags are not enabled in your PHP installation.

PHP Tag Styles –

There are actually four Different Style of PHP tags we can use. Each of the following fragments of code is equivalent.

- **Short Style-**

```
<?  
    Echo "Hello World";  
?>
```

- **XML Style –**

```
<?php  
    Echo "Hello World";  
?>
```

This style of tag can be used with XML documents. If you plane to serve XML on your site, you should use this style of tag.

- **Script Style –**

```
<script Language='php'>  
    Echo "Hello World";  
</script>
```

This style is the longest and will be familiar if you've used JavaScript or VB Script. It can be used if you are using an HTML Editor that gives you Problems with the Other Tag styles.

- **ASP Style –**

```
<%  
    Echo "Hello World";  
%>
```

This style of tag is the name as used in Active server Pages (ASP). It can be used if you have enabled the asp_tags Configuration setting. By default we cant not execute asp style tag because value of ASP_tag is off in PHP.ini Configuration file. We have to change this value as on & then save the file & restart the server then we can execute tag.

Points to remember.....

- PHP variables are case sensitive
- PHP functions are case insensitive.
- Every PHP application extension should be ".php".
- PHP is loosely typed language.
- Every line in PHP ends with semicolon (:).
- Every PHP core block we should include within the declaration style tag.
- Every variable name start with \$ symbols.
- If the server is not running or not available then browser display "**The page cannot be displayed**".
- If the requested file is not available then server display message "**object is not found**".

Output function in PHP –

1) Print() –

This function displays the message on web page & return Boolean value true if the printed is successfully. We can't not print multiple value by using print().

```
<?  
    $no=10;  
    Print "welcome";  
    Print $no;  
?>
```

```
<?php  
    $r_val= print "welcome";  
    Print $r_val;  
    Print "hello"."welcome"; //error
```

```
?>
```

2) Echo() -

Echo function can print multiple values but it will not return any value. That's why it is faster than print function.

```
<?
    Echo "welcome"."to","php";
?>
```

3) Printf() -

Using the specified format we can print the output.

```
<
    $name="abc";
    $house=5;

    Printf("%s has %d house",$name,$house);
?>
```

4) Print_r()-

This function display the all elements of array & properties of objects.

```
<
    $arr=array(10,20,30);
    Print_r($arr);
?>
```

2.2 PHP with html -

There are two ways to use HTML on your PHP page. The first way is to put the HTML outside of your PHP tags. You can even put it in the middle if you close and reopen the <?php - and- ?> tags. Here is an example of putting the HTML outside of the tags:

```
<html>
<title>HTML with PHP</title>
<body>
<h1>My Example</h1>

<?php
    //your php code here
?>
```

Here is some more HTML

```
<?php  
//more php code  
?>
```

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

As you can see you can use any HTML you want without doing anything special or extra in your .php file, as long as it is outside of the PHP tags.

The second way to use HTML with PHP is by using PRINT or ECHO. By using this method you can include the HTML inside of the PHP tags. This is a nice quick method if you only have a line or so to do. Here is an example:

```
<?php  
Echo "<html>";  
Echo "<title>HTML with PHP</title>";  
Echo "<b>My Example</b>";  
  
//your php code here  
  
Print "<i>Print works too!</i>";  
?>
```

Using one or both of these methods you can easily embed HTML code in your PHP pages, to give them a nicer more formatted look, and make them more user friendly.

Comments in PHP-

Comments are used to stop the execution of line or set of line.

1. Single Line comment. // , #
2. Multiline comment. /* - - - - */

2.3 Constant & Variable –

Variables

PHP is a Loosely Typed Language. In PHP, a variable does not need to be declared before adding a value to it. PHP automatically converts the variable to the correct data type, depending on its value.

In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it. In PHP, the variable is declared automatically when you use it.

Naming Rules for Variables

- A variable name must start with a letter or an underscore "_"
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and _)
- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore (\$my_string), or with capitalization (\$myString)

Data Types -

PHP supports eight primitive Data types- divided in to 3 categories

Scalar data types:

1. **boolean** – Boolean variable holds true & false value. If the value is true then it holds 1 & is value is false it holds empty value.
 - **Is_bool()** is the function used to check whether the input value is Boolean or not.
 - **(bool)<variable>** - Convert variable in to Boolean.

```
<?
    $a="false";
    Echo is_bool($a);
    $a=(bool)$a;
    Echo is_bool($a);
?>
```

2. **integer** - this variable store the integer value.
 - **Is_int(), is_integer()** – this function check input value is integer or not. Return true if value is integer.
 - **(int)<variable>** - Convert variable in to integer.

```
<?
```

```
$a=100;  
Echo is_int($a);  
Echo $a;  
  
?>
```

3. **float** - this data type represents the floating point numbers

- **Is_float()** – this function check input value is float or not. Return true if value is Float.
- **(float)<variable>** - Convert variable in to Floating point.

```
<?  
$a=10.15;  
Echo is_int($a);  
Echo $a;  
  
?>
```

Note – precision is the configuration setting specifies total number of digit displayed by floating point number.

4. **string** - It is collection of characters. We can declared a string variable . The simplest way to specify a string is to enclose it in single quotes & Double quotes.

```
<?  
$str='abc';  
Echo $str;  
  
?>
```

Compound Data types:

5. **array** – array is collection of elements.(we will see array Concept in detail in this chapter later)

```
<?php  
  
$arr=array{10,20,30};  
  
Print_r($arr);  
  
?>
```

- 6. object** – object is an instance of a class. (we will see about object & class later in chapter 4)

```
<?
Class emp
{
    Function empdata()
    {
        Echo("within th class");
    }

    $obj=new emp();
    $obj->empdata();
}
%>
```

Special Data types:

- 7. Resource** -

A resource variable refers the external resources like database collection, file pointer, ftp connection etc.

```
<
If($con=mysql_connect("localhost","root",""))
{
    Echo("Connected");
}
Echo $con;
Echo get_resource_type($con);
?>
```

- 8. NULL** -

In PHP null is a value we can consider a variable as null data type based on 3 condition.

- A variable is not assign.
 - The variable assign with null.
 - If the variable value is unset.
- In **php unset()** is function that delete the value of the variable.
 - **Is_null()** is a function to check whether the variable is null or not.

```
<
$x=100;
If(is_null($x))
```

```
Echo "x is null";
Else
    Echo "x is not null";

Unset($x);

If(is_null($x))
    Echo "x is null";
Else
    Echo "x is not null";
?>
```

Type of variable –

- **Local variable** – variable declaration within the function the scope of that variable is only for that function.
- **Global Variable** – a variable declared in global location means global variable, global location means outside all the function. We can access this global variable from any location form script, but we cannot access golable variable from function directly to access this function we have use \$GLOBALS [] or reduced the variable using global key word.

```
<?php
    error_reporting(E_ALL);
    $x=100;                                //Global Variable
    function fun1()
    {
        $x=1234;                            //Local Variable
        echo $GLOBALS['x'];                //calling Global Variable
        echo $x;                           //calling local Variable
    }
?>
```

- **Variables variable** – if we assign a variable name as the value of another variable comes under “variables” variable.

```
<?
    $a=2500;
    $b= 'a';
    Echo $$b;
?>
```

- **Static variable** – This variable use to same copy of variable all function. Static variable will not lose its value when the function exits and will still hold that value should the function be called again.

You can declare a variable to be static simply by placing the keyword STATIC in front of the variable name.

```
<?php
    Function fun()
    {
        Static $x=100;
        $x++;
        Echo $x;
    }
    Fun();
    Fun();
    Fun();
?
>
```

- **Reference variable** – a reference variable is an allies name of actual variable both reference variable are original variable same address location. Use & Symbole to assign reference of variable.

```
<?php
    $x=100;
    $y=&$x;           //creating reference variable
    Echo $y;
    $y=200;
    Echo $x;
?
>
```

- **Super global variable** – This are the special variable in PHP we can access them from the any page within the project. They are array data type. The variable are - \$_GET, \$_POST, \$_REQUEST, \$_COOKIES, \$_SESSION, \$_ENV, \$_SERVER etc.

Constant -

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

To define a constant you have to use define () function and to retrieve the value of a constant, you have to simply specifying its name. Unlike with variables, you do not need to have a constant with a \$. You can also use the function constant () to read a constant's value.

```
<?php  
define("MINSIZE", 50); //Defining The Constant  
  
echo MINSIZE; //Using the Constant  
echo constant("MINSIZE"); //Same thing as the previous  
line  
  
?>
```

Note - Only scalar data (Boolean, integer, float and string) can be contained in constants.

Differences between constants and variables are:

- There is no need to write a dollar sign (\$) before a constant, where as in Variable one has to write a dollar sign.
- Constants cannot be defined by simple assignment, they may only be defined using the define () function.
- Constants may be defined and accessed anywhere without regard to variable scoping rules.
- Once the Constants have been set, may not be redefined or undefined.

2.4 Passing variable Value from one page to another page –

We can pass the variable or value of variable from one page to another page by using super global variable provided by PHP. Like- \$_GET, \$_REQUEST etc.

Using \$_GET –

- First we will need an HTML form to collect the data we will pass.

Demo.html

```
<html>  
<body>  
<form method="get" action="test.php">  
    <input type="text" name="name0"/>  
    <input type="text" name="name1"/>
```

```
<input type="text" name="name2"/>
<input type="text" name="name3"/>
<input type="text" name="name4"/>

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

Here we create a one html page name it as demo.html. In html page we define form tag with 4 textbox & 1 submit button, & use get method in form. So when we press submit button value of all textbox will move in to super global variable array i.e. `$_GET[]` because we use get method in form declarations. So as we know super global variable are accessible to all project so we can access this array in next action page.

- **Create a another page of PHP script to display value of `$_GET`.**

Test.php

```
<?php
    echo $_GET['name0'];
    echo $_GET['name1'];
    echo $_GET['name2'];
    echo $_GET['name3'];
    echo $_GET['name4'];
?>
```

We can also use `$_post` variable, just we have use POST method in form declaration & while accessing value use `$_POST`.

2.5 Operator in PHP -

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Let's have a look on all operators one by one.

Arithmetic Operators:

Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide first operant by second operant	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Example -

```

<html>
<head><title>Arithmetical Operators</title></head>
<body>
<?php
    $a = 42;
    $b = 20;

    $c = $a + $b;
    echo "Addition Operation Result: $c <br/>";

    $c = $a - $b;
    echo "Subtraction Operation Result: $c <br/>";

    $c = $a * $b;
    echo "Multiplication Operation Result: $c <br/>";

    $c = $a / $b;
    echo "Division Operation Result: $c <br/>";

    $c = $a % $b;
    echo "Modulus Operation Result: $c <br/>";

    $c = $a++;
    echo "Increment Operation Result: $c <br/>";

    $c = $a--;
    echo "Decrement Operation Result: $c <br/>";

?>
</body>
</html>

```

This will produce following result-

```

Addition Operation Result: 62
Subtraction Operation Result: 22
Multiplication Operation Result: 840

```

```

Division Operation Result: 2.1
Modulus Operation Result: 2
Increment Operation Result: 42
Decrement Operation Result: 43

```

Comparison Operators:

There are following comparison operators supported by PHP language

Operator	Description	Example
<code>==</code>	Checks if the value of two operands are equal or not, if yes then condition becomes true.	$(A == B)$ is not true.
<code>!=</code>	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	$(A != B)$ is true.
<code>></code>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	$(A > B)$ is not true.
<code><</code>	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	$(A < B)$ is true.
<code>>=</code>	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	$(A >= B)$ is not true.
<code><=</code>	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	$(A <= B)$ is true.

Example -

```

<html>
<head><title>Comparison Operators</title></head>
<body>
<?php
    $a = 42;
    $b = 20;

    if( $a == $b ){
        echo "TEST1 : a is equal to b<br/>";
    }else{
        echo "TEST1 : a is not equal to b<br/>";
    }

    if( $a > $b ){
        echo "TEST2 : a is greater than b<br/>";
    }else{
        echo "TEST2 : a is not greater than b<br/>";
    }
}

```

```

if( $a < $b ){
    echo "TEST3 : a is less than b<br/>";
}else{
    echo "TEST3 : a is not less than b<br/>";
}

if( $a != $b ){
    echo "TEST4 : a is not equal to b<br/>";
}else{
    echo "TEST4 : a is equal to b<br/>";
}

if( $a >= $b ){
    echo "TEST5 : a is either grater than or equal to b<br/>";
}else{
    echo "TEST5 : a is greater than nor equal to b<br/>";
}

if( $a <= $b ){
    echo "TEST6 : a is either less than or equal to b<br/>";
}else{
    echo "TEST6 : a is less than nor equal to b<br/>";
}
?>
</body>
</html >

```

This will produce following result –

```

TEST1 : a is not equal to b
TEST2 : a is greater than b
TEST3 : a is not less than b
TEST4 : a is not equal to b
TEST5 : a is either grater than or equal to b
TEST6 : a is less than nor equal to b

```

Logical Operators:

There are following logical operators supported by PHP language

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.

&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
 	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Example –

```

<html>
<head><title>Logical Operators</title></head>
<body>
<?php
    $a = 42;
    $b = 0;

    if( $a && $b ){
        echo "TEST1 : Both a and b are true<br/>";
    }else{
        echo "TEST1 : Either a or b is false<br/>";
    }
    if( $a and $b ){
        echo "TEST2 : Both a and b are true<br/>";
    }else{
        echo "TEST2 : Either a or b is false<br/>";
    }
    if( $a || $b ){
        echo "TEST3 : Either a or b is true<br/>";
    }else{
        echo "TEST3 : Both a and b are false<br/>";
    }
    if( $a or $b ){
        echo "TEST4 : Either a or b is true<br/>";
    }else{
        echo "TEST4 : Both a and b are false<br/>";
    }
    $a = 10;
    $b = 20;
    if( $a ){
        echo "TEST5 : a is true <br/>";
    }else{
        echo "TEST5 : a is false<br/>";
    }
    if( $b ){
        echo "TEST6 : b is true <br/>";
    }

```

```

}else{
    echo "TEST6 : b is false<br/>";
}
if( !$a ){
    echo "TEST7 : a is true <br/>";
}else{
    echo "TEST7 : a is false<br/>";
}
if( !$b ){
    echo "TEST8 : b is true <br/>";
}else{
    echo "TEST8 : b is false<br/>";
}
?>
</body>
</html >

```

This will produce following result

```

TEST1 : Either a or b is false
TEST2 : Either a or b is false
TEST3 : Either a or b is true
TEST4 : Either a or b is true
TEST5 : a is true
TEST6 : b is true
TEST7 : a is false
TEST8 : b is false

```

Assignment Operators:

There are following assignment operators supported by PHP language:

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assigne value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left	C /= A is equivalent to

	operand with the right operand and assign the result to left operand	$C = C / A$
$\%=$	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	$C \%= A$ is equivalent to $C = C \% A$

Example –

```

<html>
<head><title>Assignment Operators</title></head>
<body>
<?php
    $a = 42;
    $b = 20;

    $c = $a + $b; /* Assignment operator */
    echo "Addition Operation Result: $c <br/>";
    $c += $a; /* c value was 42 + 20 = 62 */
    echo "Add AND Assignment Operation Result: $c <br/>";

    $c -= $a; /* c value was 42 + 20 + 42 = 104 */
    echo "Subtract AND Assignment Operation Result: $c <br/>";

    $c *= $a; /* c value was 104 - 42 = 62 */
    echo "Multiply AND Assignment Operation Result: $c <br/>";

    $c /= $a; /* c value was 62 * 42 = 2604 */
    echo "Division AND Assignment Operation Result: $c <br/>";

    $c %= $a; /* c value was 2604/42 = 62*/
    echo "Modulus AND Assignment Operation Result: $c <br/>";
?>
</body>
</html>

```

This will produce following result

```

Addition Operation Result: 62
Add AND Assignment Operation Result: 104
Subtract AND Assignment Operation Result: 62
Multiply AND Assignment Operation Result: 2604
Division AND Assignment Operation Result: 62
Modulus AND Assignment Operation Result: 20

```

Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax:

Operator	Description	Example
:	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Example -

```
<html>
<head><title>Arithmetical Operators</title></head>
<body>
<?php
    $a = 10;
    $b = 20;

    /* If condition is true then assign a to result otherwise b */
    $result = ($a > $b) ? $a : $b;
    echo "TEST1 : Value of result is $result<br/>";
    /* If condition is true then assign a to result otherwise b */
    $result = ($a < $b) ? $a : $b;
    echo "TEST2 : Value of result is $result<br/>";
?
</body>
</html>
```

This will produce following result

```
TEST1 : Value of result is 20
TEST2 : Value of result is 10
```

2.6 Conditional Statements -

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

You can use conditional statements in your code to make your decisions. PHP supports following three decision making statements:

- if...else statement - use this statement if you want to execute a set of code when a condition is true and another if the condition is not true

- elseif statement - is used with the if...else statement to execute a set of code if one of several conditions are true
- switch statement - is used if you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

If...Else Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax

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

Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
    $d=date("D");
    if ($d=="Fri")
        echo "Have a nice weekend!";
    else
        echo "Have a nice day!";
?>
</body>
</html>
```

The Elseif Statement

If you want to execute some code if one of several conditions are true use the elseif statement

Syntax

```
if (condition)
    //code to be executed if condition is true;
elseif (condition)
    //code to be executed if condition is true;
```

```
else
//code to be executed if condition is false;
```

Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<html>
<body>
<?php
    $d=date("D");
    if ($d=="Fri")
        echo "Have a nice weekend!";
    elseif ($d=="Sun")
        echo "Have a nice Sunday!";
    else
        echo "Have a nice day!";
?>
</body>
</html>
```

The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement. The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax

```
switch (expression)
{
    case Label 1:
        code to be executed if expression = Label 1;
        break;
    case Label 2:
        code to be executed if expression = Label 2;
        break;
    default:
        code to be executed
        if expression is different
        from both Label 1 and Label 2;
}
```

Example

The switch statement works in an unusual way. First it evaluates given expression then seeks a label to match the resulting value. If a matching value is found then the code associated with

the matching label will be executed or if none of the labels match then statement will will execute any specified default code.

```
<html>
<body>
<?php
    $d=date("D");
    switch ($d)
    {
        case "Mon":
            echo "Today is Monday";
            break;
        case "Tue":
            echo "Today is Tuesday";
            break;
        case "Wed":
            echo "Today is Wednesday";
            break;
        case "Thu":
            echo "Today is Thursday";
            break;
        case "Fri":
            echo "Today is Friday";
            break;
        case "Sat":
            echo "Today is Saturday";
            break;
        case "Sun":
            echo "Today is Sunday";
            break;
        default:
            echo "Wonder which day is this ?";
    }
?>
</body>
</html>
```

2.7 PHP Loop Types

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

for loop -

The for statement is used when you want initialization, Condition checking & increment in one block& execute a statement or a block of statements.

Syntax

```
for (initialization; condition; increment)
{
    //code to be executed;
}
```

The initialize is used to set the start value for the counter of the number of loop iterations.

Example

```
<html>
<body>
<?php
    $a = 0;
    $b = 0;

    for( $i=0; $i<5; $i++ )
    {
        $a += 10;
        $b += 5;
    }
    echo ("At the end of the loop a=$a and b=$b" );
?>
</body>
</html>
```

This will produce following result:

At the end of the loop a=50 and b=25

While loop -

The while statement will execute a block of code if and as long as a test expression is true.

If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

Syntax

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

Example

```
<html>
<body>
<?php
    $i = 0;
    $num = 50;

    while( $i < 10)
    {
        $num--;
        $i++;
    }
    echo ("Loop stopped at i = $i and num = $num" );
?>
</body>
</html>
```

This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

This will produce following result:

```
Loop stopped at i = 1 and num = 40
```

do...while loop

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax

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

Example

The following example will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than 10:

```
<html>
<body>
<?php
    $i = 0;
    $num = 0;
    do
    {
```

```
$i++;
}while( $i < 10 );
echo ("Loop stopped at i = $i");
?>
</body>
</html >
```

This will produce following result:

```
Loop stopped at i = 10
```

foreach loop -

The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to \$value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

```
foreach (array as value)
{
    //code to be executed;

}
```

Example

Try out following example to list out the values of an array.

```
<html >
<body>
<?php
    $array = array( 1, 2, 3, 4, 5);
    foreach( $array as $value )
    {
        echo "Value is $value <br />";
    }
?>
</body>
</html >
```

This will produce following result:

```
Value is 1
Value is 2
Value is 3
```

```
Value is 4  
Value is 5
```

Now we will discuss about continue and break keywords used to control the loops execution.

break statement

The PHP break keyword is used to terminate the execution of a loop prematurely.

The break statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.

Example

In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

```
<html>  
<body>  
    <?php  
    $i = 0;  
  
    while( $i < 10)  
    {  
        $i++;  
        if( $i == 3 )break;  
    }  
    echo ("Loop stopped at i = $i");  
?>  
</body>  
</html>
```

This will produce following result:

```
Loop stopped at i = 3
```

Continue statement

The PHP continue keyword is used to halt the current iteration of a loop but it does not terminate the loop.

Just like the break statement the continue statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering continue statement, rest of the loop code is skipped and next pass starts.

Example

In the following example loop prints the value of array but for which condition becomes true it just skip the code and next value is printed.

```
<html>
<body>
    <?php
        $array = array( 1, 2, 3, 4, 5);
        foreach( $array as $value )
        {
            if( $value == 3 )continue;
            echo "Value is $value <br />";
        }
    ?>
</body>
</html>
```

This will produce following result:

```
Value is 1
Value is 2
Value is 4
Value is 5
```

2.8 Include

By using this function we can embed 1 PHP script in another script. 4 type of include function are available –

- 1) **include** – this function include external script no of time. If external script is not available it returns a warning message & executes the rest of the script.

Page1.php

```
<?php
    Echo "from page1";
    Include "page2.php";

    Fun1();
    Echo $x;
```

```
>     Echo constant("ci ty");
?>
```

Page2.php

```
<?php
    Echo "from page2";
    Function fun1()
    {
        Echo "in the function of page2";
    }
    $x=100;
    Define("ci ty", "jal gaon");
?>
```

- 2) **include_once** - it is same as include but only one time we can embed the external file.
- 3) **Require** – it is same as include but return fatal error if the file is not found & stop the execution of rest of the script.
- 4) **Require_once** – it is same as require but it can only one time embed the external file.

Note - File name extension should be .inc or .php

2.9 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.

PHP functions are divided into 2 categories –

1. User Define Functions
2. Built-in Functions

User Define Function

User Define Function is those which are created by user while programming. There are two parts which should be clear to you:

- Creating a PHP Function
- Calling a PHP Function

Creating PHP Function:

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

Note that while creating a function its name should start with keyword **function** and all the PHP code should be put inside { and } braces as shown in the following example below:

Syntax –

```
Function <function name>
{
    // PHP Code
}
```

Example-

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

This will display following result:

You are really a nice person, Have a nice time!

PHP Functions with Parameters:

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters you like. These parameters work like variables inside your function. Following example takes two integer parameters and adds them together and then prints 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 >
```

This will display following result:

```
Sum of the two numbers is : 30
```

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.

Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Following example depicts both the cases.

```
<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 $origum<br />";
addSiz( $origum );
echo "Original Value is $origum<br />";
?
</body>
</html >
```

This will display following result:

```
Original Value is 15
Original Value is 21
```

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.

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

Example -

Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that return keyword is used to return a value from a function.

```
<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 >
```

This will display following result:

```
Returned value from the function : 30
```

Setting Default Values for Function Parameters:

You can set a parameter to have a default value if the function's caller doesn't pass it.

Example -

Following function prints NULL in case user does not pass any value to this function.

```
<html>
<head>
<title>Writing PHP Function which returns value</title>
</head>
<body>
<?php
    function printMe($param = NULL)
    {
        print $param;
    }
    printMe("This is test");
    printMe();
?
</body>
</html>
```

This will produce following result:

This is test

Built-in functions –

PHP is very rich in terms of Built-in functions. Here is the list of various important function categories. There are various other function categories which are not covered here. (refer official web site of PHP to read all built-in function).

- PHP Character Functions
- PHP Date & Time Functions
- PHP String Functions

Character Functions

The functions provided by this extension check whether a character or string falls into a certain character class according to the current locale.

Function	Description
ctype_digit()	Check for numeric character(s)
ctype_lower()	Check for lowercase character(s)
ctype_punct()	Check for any printable character which is not whitespace or an alphanumeric character
ctype_space()	Check for whitespace character(s)
ctype_upper()	Check for uppercase character(s)

Date and Time Functions

These functions allow you to get the date and time from the server where your PHP scripts are running. You can use these functions to format the date and time in many different ways.

Function	Description
date_create()	Returns new DateTime object
date_default_timezone_get()	Returns the default time zone
date_default_timezone_set()	Sets the default time zone
date_format()	Returns date formatted according to given format
date_time_set()	Sets the time
date()	Formats a local time/date
getdate()	Returns an array that contains date and time information for a Unix timestamp
localtime()	Returns an array that contains the time components of a Unix timestamp
microtime()	Returns the microseconds for the current time
time()	Returns the current time as a Unix timestamp

PHP String Functions

Function	Description
addcslashes()	Returns a string with backslashes in front of the specified characters
addslashes()	Returns a string with backslashes in front of predefined characters
chr()	Returns a character from a specified ASCII value
echo()	Outputs strings
fprintf()	Writes a formatted string to a specified output stream
ltrim()	Strips whitespace from the left side of a string
money_format()	Returns a string formatted as a currency string
rtrim()	Strips whitespace from the right side of a string
str_repeat()	Repeats a string a specified number of times
str_replace()	Replaces some characters in a string (case-sensitive)
strlen()	Returns the length of a string

strcmp()	String comparison of the first n characters (case-sensitive)
strrev()	Reverses a string
strtolower()	Converts a string to lowercase letters
strtoupper()	Converts a string to uppercase letters
strtr()	Translates certain characters in a string
substr()	Returns a part of a string
trim()	Strips whitespace from both sides of a string

2.10 Array -

A variable is a storage area holding a number or text. The problem is, a variable will hold only one value. An array is a special variable, which can store multiple values in one single variable.

An array can hold all your variable values under a single name. And you can access the values by referring to the array name. Each element in the array has its own index so that it can be easily accessed.

In PHP, there are three kind of arrays:

- Numeric array - An array with a numeric index
- Associative array - An array where each ID key is associated with a value
- Multidimensional array - An array containing one or more arrays

Numeric Arrays-

A numeric array stores each array element with a numeric index.

There are two methods to create a numeric array.

- In the following example the index are automatically assigned (the index starts at 0):

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

- In the following example we assign the index manually:

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

Example

In the following example you access the variable values by referring to the array name and index:

```
<?php
$cars[0] = "Saab";
$cars[1] = "Volvo";
$cars[2] = "BMW";
$cars[3] = "Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

The code above will output:

Saab and Volvo are Swedish cars.

Associative Arrays

An associative array, each ID key is associated with a value. When storing data about specific named values, a numerical array is not always the best way to do it. With associative arrays we can use the values as keys and assign values to them.

Example 1

In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Example 2

This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
```

The code above will output:

```
Peter is 32 years old.
```

Multidimensional Arrays

In a multidimensional 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.

Example

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array(
    "Griffin"=>array("Peter", "Lois", "Megan"),
    "Quagmire"=>array("Glenn"),
    "Brown"=>array("Cleveland", "Loretta", "Junior")
);
```

The array above would look like this if written to the output:

```
Array
(
    [Griffin] => Array
        (
            [0] => Peter
            [1] => Lois
            [2] => Megan
        )
    [Quagmire] => Array
        (
            [0] => Glenn
        )
    [Brown] => Array
        (
            [0] => Cleveland
            [1] => Loretta
            [2] => Junior
        )
)
```

PHP Array Functions

Function	Description
array()	Creates an array
array_fill()	Fills an array with values
array_keys()	Returns all the keys of an array
array_merge()	Merges one or more arrays into one array
array_pop()	Deletes the last element of an array
array_push()	Inserts one or more elements to the end of an array
array_rand()	Returns one or more random keys from an array
array_reverse()	Returns an array in the reverse order
array_search()	Searches an array for a given value and returns the key
array_sum()	Returns the sum of the values in an array
array_unique()	Removes duplicate values from an array
array_values()	Returns all the values of an array
arsort()	Sorts an array in reverse order and maintain index association
count()	Counts elements in an array, or properties in an object
current()	Returns the current element in an array
sort()	Sorts an array

2.11 Using \$_GET & \$_POST For Working with Forms –

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

Try out following example by putting the source code in -

Test.php

```
<?php
    if( $_GET["name"] || $_GET["age"] )
    {
        echo "Wel come ". $_GET[' name' ]. "<br />";
        echo "You are ". $_GET[' age' ]. " years old. ";
        exit();
    }
?>
```

Default.html

```
<html>
<body>
    <form action="test.php" 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 you can make sure that your information is secure.
- The PHP provides `$_POST` associative array to access all the sent information using POST method.

Try out following example by putting the source code in test.php script.

```
<?php
if( $_POST["name"] || $_POST["age"] )
{
    echo "Wel come ". $_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>
```

`$_PHP_SELF` is predefine variable of php which is use represent self page.

The `$_REQUEST` variable

The PHP `$_REQUEST` variable contains the contents of both `$_GET`, `$_POST`, and `$_COOKIE`. We will discuss `$_COOKIE` variable when we will explain about cookies.

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

Try out following example by putting the source code in test.php script.

```
<?php
if( $_REQUEST["name"] || $_REQUEST["age"] )
{
    echo "Wel come ". $_REQUEST[' name']. "<br />";
    echo "You are ". $_REQUEST[' age']. " years old.";
    exit();
}
?>

<html>
```

```
<body>
    <form action=<?php $_PHP_SELF ?> method="POST">

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

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

2.12 Validating forms with PHP

So, how do you validate form data? The very least you should do is pass all variables through PHP's `htmlspecialchars()` function. This function will replace HTML chars like `<` and `>` to their HTML version `<` and `>`. Let's rewrite the previous example:

```
<?php
    $yourname = htmlspecialchars($_POST['yourname']);
    $email = htmlspecialchars($_POST['email']);
    $likeit = htmlspecialchars($_POST['likeit']);
    $comments = htmlspecialchars($_POST['comments']);
?>
<html>
    <body>
        Your name is: <?php echo $yourname; ?><br />
        Your e-mail: <?php echo $email; ?><br />
        <br />
        Do you like this website? <?php echo $likeit; ?><br />
        <br />
        Comments: <br /><?php echo $comments; ?>
    </body>
</html >
```

This is much safer now and prevents possible attackers from exploiting our code by injecting HTML or Javascript code. Now if someone submitted the same code as before...

```
<script>location.href('http://www.SPAM.com')</script>
```

... this would not be executed anymore, because it would be saved as HTML escaped code rather than valid HTML code:

```
&lt;script&gt;location.href('http://www.SPAM.com')&lt;/script&gt;
```

Such code can now do no harm and is safe to be displayed on a page or inside an e-mail. Sure, it may not look nice and tell you someone has been trying to mess with your script, but the important thing is he/she had failed!

If you know exactly what kind of data to expect you can make further steps to ensure the user has entered what you want. We will cover a few samples like validating e-mail address and URLs later.

Let's do two more things:

1. strip unnecessary characters from the data.
2. if quotes are escaped with a slash \ let's remove that.

Instead of writing the same code over and over again we can create a function that will do all the checking for us. Here we will name it check_input and simply call this function whenever we need to validate simple input data:

```
<?php
    $yourname = check_input($_POST['yourname']);
    $email    = check_input($_POST['email']);
    $likeit   = check_input($_POST['likeit']);
    $comments = check_input($_POST['comments']);
?>
<html>
<body>
    Your name is: <?php echo $yourname; ?><br />
    Your e-mail: <?php echo $email; ?><br />
    <br />
    Do you like this website? <?php echo $likeit; ?><br />
    <br />
    Comments: <br /><?php echo $comments; ?>
</body>
</html>

<?php
function check_input($data)
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

?>
```

Note the check_input function at the bottom. What it does is takes the data passed to the function, strips unwanted characters (extra space, tab, newline) from the beginning and end of the data using the PHP trim() function, strips any quotes escaped with slashes and passes it through htmlspecialchars().

So now instead of typing the same code for each of our input fields we simply check each \$_POST variable with the check_input function and that's it.

2.13 Validation Emailing from data -

Using the PHP form validation script

Include formvalidator.php in your form processing script

```
require_once "formvalidator.php"
```

Create a FormValidator object and add the form validation descriptors.

```
$validator = new FormValidator();
$validator->addValidation("Name", "req", "Please fill in Name");
$validator->addValidation("Email", "email",
    "The input for Email should be a valid email value");
$validator->addValidation("Email", "req", "Please fill in Email");
```

The first argument is the name of the input field in the form. The second argument is the validation descriptor that tells the type of the validation required. The third argument is the error message to be displayed if the validation fails.

Validate the form by calling ValidateForm() function

```
if($validator->validateForm())
{
    echo "<h2>Validation Success! </h2>";
    $show_form=false;
}
else
{
```

```
echo "<B>Validation Errors: </B>";
$error_hash = $validator->GetErrors();
foreach($error_hash as $inputname => $input_err)
{
    echo "<p>$inputname : $input_err</p>\n";
}
}
```

2.14 Email validation within a form in PHP

In an input field we can verify the email address entered is in correct format or not using PHP eregi function. This is a part of php form validation we do for other entries to check proper formatted data is entered by the user or not. You can enter any email address and check it is a valid address or not.

Here is the code of this validation in PHP write this code in PHP file Along with html code.

```
if(isset($todo) and $todo=="test"){
if (!eregi ("^[_a-z0-9-]+(\. [_a-z0-9-]+)*@[a-z0-9-]+(\. [a-z0-9-]+)*(\. [a-z]{2, 3})$", $email)){
echo "<center>Invalid email </center>";
} else{
echo "<center>Valid Email </center>"; }
}
```

2.15 PHP Error Handling

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

This tutorial contains some of the most common error checking methods in PHP.

We will show different error handling methods:

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

Using the die() function

The first example shows a simple script that opens a text file:

```
<?php
```

```
$file=fopen("welcome.txt", "r");  
?>
```

If the file does not exist you might get an error like this:

Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\webfolder\test.php on line 2

To avoid that the user gets an error message like the one above, we test if the file exist before we try to access it:

```
<?php  
if(!file_exists("welcome.txt"))  
{  
    die("File not found");  
}  
else  
{  
    $file=fopen("welcome.txt", "r");  
}  
?>
```

Now if the file does not exist you get an error like this:

File not found

The code above is more efficient than the earlier code, because it uses a simple error handling mechanism to stop the script after the error.

Custom Error Handler

Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.

This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

Syntax

```
error_function(error_level, error_message,  
              error_file, error_line, error_context)
```

Parameter	Description
error_level	Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

Error Report levels

These error report levels are the different types of error the user-defined error handler can be used for:

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings, except level E_STRICT (E_STRICT will be part of E_ALL as of PHP 6.0)

Now lets create a function to handle errors:

```
function customError($errno, $errstr)
{
    echo "<b>Error: </b> [$errno] $errstr<br />";
    echo "Ending Script";
    die();
}
```

The code above is a simple error handling function. When it is triggered, it gets the error level and an error message. It then outputs the error level and message and terminates the script.

Now that we have created an error handling function we need to decide when it should be triggered.

Set Error Handler

The default error handler for PHP is the built in error handler. We are going to make the function above the default error handler for the duration of the script.

It is possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways. However, in this example we are going to use our custom error handler for all errors:

```
set_error_handler("customError");
```

Since we want our custom function to handle all errors, the `set_error_handler()` only needed one parameter, a second parameter could be added to specify an error level.

Example

Testing the error handler by trying to output variable that does not exist:

```
<?php
//error handler function
function customError($errno, $errstr)
{
    echo "<b>Error: </b> [$errno] $errstr";
}

//set error handler
set_error_handler("customError");

//trigger error
echo($test);
?>
```

The output of the code above should be something like this:

Error: [8] Undefined variable: test

Trigger an Error

In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the trigger_error() function.

Example

In this example an error occurs if the "test" variable is bigger than "1":

```
<?php
$test=2;
if ($test>1)
{
trigger_error("Value must be 1 or below");
}
?>
```

The output of the code above should be something like this:

Notice: Value must be 1 or below in C:\webfolder\test.php on line 6

An error can be triggered anywhere you wish in a script, and by adding a second parameter, you can specify what error level is triggered.

Possible error types:

- E_USER_ERROR - Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted
- E_USER_WARNING - Non-fatal user-generated run-time warning. Execution of the script is not halted
- E_USER_NOTICE - Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally

Example

In this example an E_USER_WARNING occurs if the "test" variable is bigger than "1". If an E_USER_WARNING occurs we will use our custom error handler and end the script:

```
<?php
//error handler function
function customError($errno, $errstr)
{
echo "<b>Error: </b> [$errno] $errstr<br />";
```

```
echo "Ending Script";
die();
}

//set error handler
set_error_handler("customError", E_USER_WARNING);

//trigger error
$test=2;
if ($test>1)
{
    trigger_error("Value must be 1 or below", E_USER_WARNING);
}
?>
```

The output of the code above should be something like this:

Error: [512] Value must be 1 or below Ending Script

2.16 PHP Files & I/O

Opening and Closing Files

The PHP fopen() function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate.

File modes can be specified as one of the six options in this table.

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

If an attempt to open a file fails then fopen returns a value of false otherwise it returns a file pointer which is used for further reading or writing to that file.

The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened:

```
<html>
<body>
    <?php
        $file=fopen("wel come. txt", "r");
    ?>
</body>
</html>
```

After making a changes to the opened file it is important to close it with the fclose() function. The fclose() function requires a file pointer as its argument and then returns true when the closure succeeds or false if it fails.

```
<?php
$file = fopen("test. txt", "r");

//some code to be executed

fclose($file);
?>
```

Reading a file

Once a file is open using fopen() function it can be read with a function called fread(). This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.

The file's length can be found using the filesize() function which takes the file name as its argument and returns the size of the file expressed in bytes.

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

- Open a file using fopen() function.
- Get the file's length using filesize() function.
- Read the file's content using fread() function.
- Close the file with fclose() function.

The following example assigns the content of a text file to a variable then displays those contents on the web page.

```
<html>
<head>
<title>Reading a file using PHP</title>
</head>
<body>

<?php
$filename = "/home/user/guest/tmp.txt";
$file = fopen( $filename, "r" );
if( $file == false )
{
    echo ( "Error in opening file" );
    exit();
}
filesize = filesize( $filename );
$filetext = fread( $file, $filesize );

fclose( $file );

echo ( "File size : $filesize bytes" );
echo ( "<pre>$text</pre>" );
?>

</body>
</html>
```

Writing a file

A new file can be written or text can be appended to an existing file using the PHP fwrite() function. This function requires two arguments specifying a file pointer and the string of data that is to be written. Optionally a third integer argument can be included to specify the length of the data to write. If the third argument is included, writing would stop after the specified length has been reached.

The following example creates a new text file then writes a short text heading inside it. After closing this file its existence is confirmed using file_exists() function which takes file name as an argument

```
<?php
$filename = "/home/user/guest/newfile.txt";
$file = fopen( $filename, "w" );
if( $file == false )
{
    echo ( "Error in opening new file" );
    exit();
}
fwrite( $file, "This is a simple test\n" );
```

```

fclose( $file );
?>

<html>
<head>
<title>Writing a file using PHP</title>
</head>
<body>

<?php
if( file_exists( $filename ) )
{
    $filesize = filesize( $filename );
    $msg = "File created with name $filename ";
    $msg .= "containing $filesize bytes";
    echo ($msg );
}
else
{
    echo ("File $filename does not exist" );
}
?>
</body>
</html>

```

Working with directory-

Some PHP Function for performing operation on Directory –

Function	Description
chdir	Change directory
chroot	Change the root directory
dir	Return an instance of the Directory class
closedir	Close directory handle
getcwd	Gets the current working directory
opendir	Open directory handle
readdir	Read entry from directory handle
rewinddir	Rewind directory handle
scandir	List files and directories inside the specified path

Example –

```

<?php
if($dir = opendir("c:\program Files"))
{

```

```
while ($files = @readdir($dir))
{
    if(($files != '.') && ($files != '..') && ($files[0] != '.'))
    {
        if(filtype($files) == 'dir')
        echo '<a href="'. $files. '">' . $files. '</a><br />';
        else
        echo '<a href="'. $files. '">' . $files. '</a><br />';
    }
    closedir($dir);
}
else
    echo 'Directory can\'t be opened.';
?>
```

3. Using PHP with MySql

3.1 Introduction of MySql

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

Milestones in MySQL development include:

- Original development of MySQL by Michael Widenius and David Axmark beginning in 1994
- First internal release on 23 May 1995

The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, after that MySQL acquired by SunMicro System & now currently owned by Oracle Corporation.

Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, phpBB, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google (though not for searches), Facebook, and Twitter.

MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP web application software stack—LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python".

MySQL is written in C and C++. Its SQL parser is written in yacc, and a home-brewed lexical analyzer named `sql_lex.cc`.^[17]

MySQL works on many different system platforms, including AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, Mac OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists

Many programming languages with language-specific APIs include libraries for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's Visual Studio (languages such as C# and VB are most commonly used) and the JDBC driver for Java. In

addition, an ODBC interface called MyODBC allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP

3.2 Installation & Configuration of MySql

MySQL for Windows is available in several distribution formats, MySQL provided with some distribution packages like Xampp, Wampp , etc . so when we install this packages MySQL Will be automatically Installed & Configured With PHP.

Separate MySQL Installing packages

The MySQL Installing packages comes in three flavors, you can choose any of the Installer package as per your requirements and your expertise level. These packages are:

1. The Essential package

The Essential package has a Similar filename to MySQL-essential-5.0.25-win32.msi and it contains the minimum set of files needed for installation MySQL on Windows, including the Configuration Wizard except embedded server and benchmark suite. This package is recommended most of the users as the user can install multiple instances of MySQL on a single server host and the advanced user can complete control over server configuration.

2. The Complete package –

The complete package has all the needed files that are required for windows installation including configuration wizard, embedded server and bench mark suite. This package has a similar filename to MySQL 5 . 0 . 25- win32 zip . For using it, your system must have a tool that can unpack .zip files.

3. The Noinstall Archive -

The Noinstall archive package has a similar filename to mysql-noinstall-5.0 .25 -win32 . zip. This package contain all the needed set of files for a complete installation wizard excluding all the configuring wizard. This package has a facility of installing manually not automatically. For using this package, there must be a tool for unpacking .zip files in your system.

Downloading MySQL

The latest version of MySQL can be downloaded from <http://www.mysql.com>. Understanding the MySQL Installation wizard MySQL Installation Wizard uses the new installer technologies for Microsoft Windows. And that's why it is an installer for MySQL server. This MySQL installation wizard is combined with the MySQL configuration wizard who allows user to install and configure a MySQL server, and it can be used immediately after installation. MySQL provide the standard installation for all MySQL Server distribution or higher version 4. 1. 5. In this version MySQL need to shut down and remove their existing MySQL installation and manually installation before the MySQL wizard. This installation process use of this technology to provide a smoother and flexible process.

MySQL Installing Type

There are three type of MySQL Installation:-

- **Typical**

This MySQL installing process install the MySQL server, command- line client and command line utilities. This command line and command client included in mysqldump, myisamchk and several tools to manage the MySQL Server.

- **Complete**

By this installation type we install all the component included in the installation package. And this package include the component such as support script, the embedded server library. MySQL complete installer also included the benchmark suit and documentation.

- **Custom**

MySQL is use Custom installation type command that provide you the complete control over package that you want to install and also the installation path. If you choose the Custom type then you can change the installation path or components , which is installed by Installation Wizard

We can directly access Php MyAdmin page by following URL –

<http://localhost/phpmyadmin/>

Note - Environment Provided by Mysql For working with database is called “PHPMyAdmin”.

PHPMyAdmin –

Option of PHPMyAdmin –

- Brows – display the Total records of MySql database table.
- Sql- it display the textbox to enter the Sql Statements.
- Insert – to insert the records in database table.
- Structure – by using this option we can see the table Structure to add new column, to edit the column.
- Search –to search the table records in ascending & descending format.
- Empty – To remove all records of table.
- Drop – to delete the table structure.
- Export- to export the table records as a Sql File.
- Import – to import the table records from Sql file.
- Designer – by using this we can create relationship between tables.

3.3 MySQL Data Types

In MySQL there are three main types : text, number, and Date/Time types.

Text types:

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. Note: If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let 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.

	Note: The values are sorted in the order you enter them.
	You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

Number types:

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis
FLOAT(size,d)	A small number with a floating decimal point. 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
DOUBLE(size,d)	A large number with a floating decimal point. 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
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. 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

Date types:

Data type	Description
DATE()	A date. Format: YYYY-MM-DD Note: The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MM:SS Note: The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD HH:MM:SS

	Note: The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MM:SS Note: The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. Note: Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

3.4 DDL Operations on Database using PHP Script

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);
```

Parameter	Description
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.

NOTE: You can specify server, user, password in php.ini file instead of using them again and again in your every PHP scripts. Check php.ini file configuration.

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 open database is closed.

Example:

Try out following example to open and close a database connection:

```
<?php
    $dbhost = 'localhost:3036';
    $dbuser = 'guest';
    $dbpass = 'guest123';
    $conn = mysql_connect($dbhost, $dbuser, $dbpass);
    if(! $conn )
    {
        die('Could not connect: ' . mysql_error());
    }
    echo 'Connected successfully';
    mysql_close($conn);

?>
```

Creating a Database:

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

Syntax:

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

Parameter	Description
sql	Required - SQL query to create a database
connection	Optional - if not specified then last opened connection by <code>mysql_connect</code> will be used.

Example:

Try out following example to create a database:

```
<?php
    $dbhost = 'localhost:3036';
    $dbuser = 'root';
    $dbpass = 'rootpassword';
    $conn = mysql_connect($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 you establish a connection with a database server then it is required to select a particular database where all the tables are associated. This is required because there may be multiple databases residing on a single server and you can do work with a single database at a time.

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 );
```

Parameter	Description
db_name	Required - Database name to be selected
connection	Optional - if not specified then last opened connection by mysql_connect will be used.

Example:

Here is the example showing you how to select a database.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'guest';
$dbpass = 'guest123';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if( ! $conn )
{
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_select_db('test_db');
```

```
mysql_close($conn);  
?>
```

Creating Database Tables:

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

Example:

Try out following example to create a table:

```
<?php  
$dbhost = 'localhost:3036';  
$dbuser = 'root';  
$dbpass = 'rootpassword';  
$conn = mysql_connect($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);  
?>
```

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

Consider the following content in **sql_query.txt** file

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

Create another **test.php** file

```
<?php
$dbhost = 'localhost: 3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$query_file = 'sql_query.txt';

$fp      = fopen($query_file, 'r');
$sql   = fread($fp, filesize($query_file));
fclose($fp);

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);

?>
```

Deleting a Database:

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

Example:

Try out following example to drop a database.

```
<?php
    $dbhost = 'localhost: 3036';
    $dbuser = 'root';
    $dbpass = 'rootpassword';
    $conn = mysql_connect($dbhost, $dbuser, $dbpass);
    if(! $conn )
    {
        die('Could not connect: ' . mysql_error());
    }
    $sql = 'DROP DATABASE test_db';
    $retval = mysql_query( $sql, $conn );
    if(! $retval )
    {
        die('Could not delete database db_test: ' .
            mysql_error());
    }
    echo "Database deleted successfully\n";
    mysql_close($conn);

?>
```

WARNING: it's very dangerous to delete a database and any table. So before deleting any table or database you should make sure you are doing everything intentionally.

Deleting a Table:

Its again a matter of issuing one SQL command through mysql_query function to delete any database table.

Example:

Try out following example to drop a table:

```
<?php
    $dbhost = 'localhost: 3036';
```

```
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$sql = 'DROP TABLE employee';
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not delete table employee: ' . mysql_error());
}
echo "Table deleted successfully\n";
mysql_close($conn);

?>
```

3.5 Data Manipulation Operation With PHP Script -

Data Insertion In Database

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

Example:

Try out following example to insert record into employee table.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$sql = 'INSERT INTO employee ' .
        '(emp_name, emp_address, emp_salary, join_date) ' .
        'VALUES ( "guest", "XYZ", 2000, NOW() )';

mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
```

```
        die(' Could not enter data: ' . mysql_error());
    }
echo "Entered data successfully\n";
mysql_close($conn);

?>
```

In real application, all the values will be taken using HTML form and then those values will be captured using PHP script and finally they will be inserted into MySQL tables.

While doing data insert its best practice to use function **get_magic_quotes_gpc()** to check if current configuration for magic quote is set or not. If this function returns false then use function **addslashes()** to add slashes before quotes.

Retrieving Data from Database

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

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

Below is a simple example to fetch records from employee table.

Example:

Try out following example to display all the records from employee table.

```
<?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);

?>
```

The content of the rows are assigned to the variable \$row and the values in row are then printed.

NOTE: Always remember to put curly brackets when you want to insert an array value directly into a string.

Updating Records

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

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

Example:

Try out following example to understand update operation. You need to provide an employee ID to update an employee salary.

```
<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 echo $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" value="Update">
        </td>
    </tr>
</table>
```

```
</form>
<?php
}
?>
</body>

</html >
```

Deleting Records -

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

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

Example:

Try out following example to understand delete operation. You need to provide an employee ID to delete an employee record from employee table.

```
<html >
<head>
<title>Delete a Record from MySQL Database</title>
</head>
<body>

<?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);
}
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"> </td>
            <td> </td>
        </tr>
        <tr>
            <td width="100"> </td>
            <td>
                <input name="delete" type="submit" id="delete"
                value="Delete">
            </td>
        </tr>
    </table>
</form>
<?php
?
?>
</body>

</html >
```

WHERE clause

The WHERE clause is used to extract only those records that fulfill a specified criterion.

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
```

To get PHP to execute the statement above we must use the `mysql_query()` function. This function is used to send a query or command to a MySQL connection.

Example

The following example selects all rows from the "Persons" table where "FirstName='Peter':

```
<?php
$con = mysql_connect("localhost", "peter", "abc123");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons
WHERE FirstName='Peter'");

while($row = mysql_fetch_array($result))
{
    echo $row['FirstName'] . " " . $row['LastName'];
    echo "<br />";
}
?>
```

ORDER BY clause

The ORDER BY keyword is used to sort the data in a recordset. The ORDER BY keyword sort the records in ascending order by default. If you want to sort the records in a descending order, you can use the DESC keyword.

Syntax

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

Example

The following example selects all the data stored in the "Persons" table, and sorts the result by the "Age" column:

```
<?php
$con = mysql_connect("local host", "peter", "abc123");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons ORDER BY age");

while($row = mysql_fetch_array($result))
{
    echo $row['FirstName'];
    echo " " . $row['LastName'];
    echo " " . $row['Age'];
    echo "<br />";
}

mysql_close($con);
?>
```

3.6 PHP mysql_fetch_array() Function

The `mysql_fetch_array()` function returns a row from a recordset as an associative array and/or a numeric array. This function gets a row from the `mysql_query()` function and returns an array on success, or FALSE on failure or when there are no more rows.

Syntax

```
mysql_fetch_array(data,array_type)
```

Parameter	Description
data	Required. Specifies which data pointer to use. The data pointer is the result from the <code>mysql_query()</code> function
array_type	Optional. Specifies what kind of array to return. Possible values: <ul style="list-style-type: none">• <code>MYSQL_ASSOC</code> - Associative array• <code>MYSQL_NUM</code> - Numeric array

- | | |
|--|--|
| | <ul style="list-style-type: none">• MYSQL_BOTH - Default. Both associative and numeric array |
|--|--|

Note: After the data is retrieved, this function moves to the next row in the recordset. Each subsequent call to mysql_fetch_array() returns the next row in the recordset.

Example

```
<?php
$con = mysql_connect("I ocal host", "peter", "abc123");
if (!$con)
{
    die(' Could not connect: ' . mysql_error());
}

$db_selected = mysql_select_db("test_db", $con);
$sql = "SELECT * from Person WHERE Lastname='Refsnes'";
$result = mysql_query($sql, $con);
print_r(mysql_fetch_array($result));

mysql_close($con);
?>
```

3.7 Perform MySQL backup using PHP

It is always good practice to take a regular backup of your database. There are three ways you can use to take backup of your MySQL database.

- Using SQL Command through PHP.
- Using phpMyAdmin user interface.

Using SQL Command through PHP

You can execute SQL SELECT command to take a backup of any table. To take a complete database dump you will need to write separate query for separate table. Each table will be stored into separate text file.

Example:

Try out following example of using SELECT INTO OUTFILE query for creating table backup :

```
<?php
$dbhost = ' I ocal host: 3036' ;
$dbuser = ' root' ;
```

```
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$table_name = "employee";
$backup_file = "/tmp/employee.sql";
$sql = "SELECT * INTO OUTFILE '$backup_file' FROM
$table_name";

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

Using phpMyAdmin user interface:

If you have phpMyAdmin user interface available then it's very easy for your to take backup of your database.

To backup your MySQL database using phpMyAdmin click on the "export" link on phpMyAdmin main page. Choose the database you wish to backup, check the appropriate SQL options and enter the name for the backup file.

4. Object Oriented Concepts in PHP

4.1 Introduction

Object-oriented programming (OOP) is a programming paradigm using "objects" – data structures consisting of data fields and methods together with their interactions – to design applications and computer programs. Programming techniques may include features such as data abstraction, encapsulation, messaging, modularity, polymorphism, and inheritance. Many modern programming languages now support OOP, at least as an option.

The object-oriented approach encourages the programmer to place data where it is not directly accessible by the rest of the program. Instead, the data is accessed by calling specially written functions, commonly called methods, which are either bundled in with the data or inherited from "class objects." These act as the intermediaries for retrieving or modifying the data they control.

4.2 Advantages of Object Oriented PHP -

- **Inheritance:** When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Polymorphism:** This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it make take different number of arguments and can do different task.
- **Data Abstraction:** Any representation of data in which the implementation details are hidden (abstracted).
- **Encapsulation:** refers to a concept where we encapsulate all the data and member functions together to form an object.
- **Class:** This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.
- **Object:** An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.

4.3 Class & object

A class is user defined data type that contains attributes or data members; and methods which work on the data members.

Creating a class in PHP Script -

To create a class, you need to use the keyword `class` followed by the name of the class. The name of the class should be meaningful to exist within the system. The body of the class is placed between two curly brackets within which you declare class data members/variables and class methods.

Following is a prototype of a PHP5 class structure

```
class <class-name> {  
    <class body :- Data Members & Methods>  
}
```

Definition of an Object- An object is a living instance of a class. This means that an object is created from the definition of the class and is loaded in memory.

Creating Objects in PHP5 Class

To create an object of a PHP5 class we use the keyword `new`. Below is the syntax style of how to create objects in PHP5:

```
$obj_name = new ClassName();
```

In the above syntax style, `$obj_name` is a variable in PHP. '`new`' is the keyword which is responsible for creating a new instance of `ClassName`.

Therefore, an object is a living instance of a class. Each object / living instance has its own memory space that can hold independent data values.

Example -

```
class Customer {  
    private $first_name, $last_name;  
  
    public function getData($first_name, $last_name) {  
        $this->first_name = $first_name;  
        $this->last_name = $last_name;  
    }  
  
    public function printData() {  
        echo $this->first_name . " : " . $this->last_name;  
    }  
}  
  
$c1 = new Customer();  
$c2 = new Customer();
```

4.4 Data Member

Instance Data Members and Methods

When you create a class, it's a template. In order to use the template, you have to instantiate an object based on that template. Once your object is created, the variables inside that object are accessible to you anywhere in your script. This is a deviation from a full implementation of OOP because generally, these variables are supposed to be hidden or 'private' - meaning, the only functions that can access the variables are those inside the class definition.

So, when you instantiate an object using:

```
$c1 = new Customer();
```

\$C1 now contains all the variables listed under your class definition. You can access these variables using the following syntax.

```
echo $c1->first_name;
```

As per above demonstrate we can access all data member of class like, variable, methods etc.

Static Data Members and Methods

A data member that is commonly available to all objects of a class is called a static member. Unlike regular data members, static members share the memory space between all objects of the same class.

To define a static member in PHP5 you need to prefix the class member name with the keyword 'static'. Look at the example below.

```
class Customer {  
    private $first_name; // regular member  
    static public $instance_count; //static data member  
}
```

In the above example \$instance_count is declared as a static data member

A static member data can be accessed using the name of the class along with the scope resolution operator (::) i.e. you don't need to create an instance of that class

Example -

```
class Customer {  
  
    static public $instance_count = 0; //static data member  
  
    public function __construct() {  
        Customer::$instance_count++;  
    }  
  
    public function __destruct() {  
        Customer::$instance_count--;  
    }  
  
    public function getFirstName() {  
        //body of method  
    }  
  
    static public function getInstanceCount() {  
        //body of method  
    }  
}  
  
$c1 = new Customer();  
$c2 = new Customer();  
  
echo Customer::$instance_count;
```

Output:

2

Static methods

A static method is a class method that can be called without creating an instance of a class. Such methods are useful when creating utility classes.

To define static data methods in PHP5 you need to prefix the class method name with the keyword 'static'. Look at the example below.

```
class Customer {  
  
    public function getFirstName() {  
        //body of method  
    }  
}
```

```
    static public function getInstanceCount() {
        //body of method
    }
}
```

In the above example getInstanceCount is declared as a static method

A static method can be accessed using the name of the class along with the scope resolution operator (:) i.e. you don't need to create an instance of that class. However, you can also access it with an instance variable.

Example -

```
class Customer {

    static public $Stat_count = 1; //static data member

    public function getFirstName() {
        //body of method
    }

    static public function getInstanceCount() {
        return Customer::$Stat_count;
    }
}

$c1 = new Customer();
$c2 = new Customer();

echo Customer::getInstanceCount(); //this is using the scope resolution
operator
echo $c1->getInstanceCount(); //this is using the instance variable

Output:
1
1
```

Rules to keep in mind for static methods

- A static method can only access static data members
- A static method does not have access to the \$this variable

4.5 Constructor & Destructor Functions:

Constructor

Constructor Functions are special type of functions which are called automatically whenever an object is created. So we take full advantage of this behaviour, by initializing many things through constructor functions.

PHP provide 2 way to create Constructor –

- 1) By traditional way using same class name for function.

```
class Foo {  
    function __construct(){  
        //do stuff  
    }  
}
```

- 2) By Using a special function called `__construct()` to define a constructor. You can pass as many as arguments you like into the constructor function.

Following example will create one constructor for Books class and it will initialize price and title for the book at the time of object creation.

```
function __construct( $par1, $par2 ){  
    $price = $par1;  
    $title = $par2;  
}
```

Now we don't need to call set function separately to set price and title. We can initialize these two member variables at the time of object creation only. Check following example below:

```
<?php  
class BaseClass {  
    function __construct() {  
        print "In BaseClass constructor\n";  
    }  
}  
  
$obj = new BaseClass();  
?>
```

This will produce following result:

In BaseClass constructor

Destructor:

Like a constructor function you can define a destructor function using function `__destruct()`. You can release all the resources with-in a destructor.

```
<?php
class MyDestructableClass {
    function __construct() {
        print "In constructor\n";
        $this->name = "MyDestructableClass";
    }

    function __destruct() {
        print "Destroying " . $this->name . "\n";
    }
}

$obj = new MyDestructableClass();
?>
```

4.6 Inheritance -

Inheritance is the mechanism of deriving a new class from an existing class. It allows a sub-class / child class to share/inherit the attributes and behaviors of a base-class or parent class. These inherited attributes and behaviors are usually modified by means of extension.

To inherit in PHP5, you should use the keyword 'extends' in the class definition. In PHP5 only single inheritance is allowed. Look at the example below:

Example -

```
class Person {
    private $name;
    private $address;

    public function getName() {
        return $this->name;
    }
}

class Customer extends Person {
    private $customer_id;
```

```
private $record_date;

public getCustomerId() {
    return $this->customer_id;
}

public getCustomerName() {
    return $this->getName(); // getName() is in Person
}
```

In the above example, class Customer extends from the Person class. This means that Customer class is the child class and the Person base class. The child class Customer extends the method getName() and calls it in the getCustomerName() method of the Customer class.

4.7 Access Specifiers

Access specifiers specify the level of access that the outside world (i.e. other class objects, external functions and global level code) have on the class methods and class data members. Access specifiers can either be public, private or protected.

Access specifiers are used as a key component of Encapsulation and Data Hiding. By using either of the access specifiers mentioned above i.e. public, private or protected you can hide or show the internals of your class to the outside world.

Explanation of each access specifier

1. Private
2. Protected
3. Public

Private

A private access specifier is used to hide the data member or member function to the outside world. This means that only the class that defines such data member and member functions have access them. Look at the example below:

```
class Customer {
    private $name;

    public function setName($name) {
        $this->name = $name;
    }

    public function getName() {
        return $this->name;
}
```

```
    }  
  
$c = new Customer();  
$c->setName("Dhanpal ");  
echo $c->name; //error, $name cannot be accessed from outside the class  
                // $name can only be accessed from within the class  
  
echo $c->getName(); //this works, as the methods of the class have  
access  
                    //to the private data members or methods
```

In the above example, echo \$c->name will give you an error as \$name in class Customer has been declared private and hence only be accessed by its member functions internally. Therefore, the following line echo \$c->getName() will display the name.

Public

A public access specifier provides the least protection to the internal data members and member functions. A public access specifier allows the outside world to access/modify the data members directly unlike the private access specifier. Look at the example below:

```
class Customer {  
    public $name;  
  
    public function setName($name) {  
        $this->name = $name;  
    }  
  
    public function getName() {  
        return $this->name;  
    }  
}  
  
$c = new Customer();  
$c->setName("Dhanpal ");  
echo $c->name;           // this will work as it is public.  
$c->name = "New Name"; // this does not give an error.
```

In the above example, echo \$c->name will work as it has been declared as public and hence can be accessed by class member functions and the rest of the script.

Protected

A protected access specifier is mainly used with inheritance. A data member or member function declared as protected will be accessed by its class and its base class but not from the outside world (i.e. rest of the script). We can also say that a protected data member is

public for the class that declares it and its child class; but is private for the rest of the program (outside world). Look at the example below:

```
class Customer {  
    protected $name;  
  
    public function setName($name) {  
        $this->name = $name;  
    }  
  
    public function getName() {  
        return $this->name;  
    }  
}  
  
class DiscountCustomer extends Customer {  
  
    private $discount;  
  
    public function setData($name, $discount) {  
        $this->name = $name; //this is storing $name to the Customer  
                           //class $name variable. This works  
                           //as it is a protected variable  
  
        $this->discount = $discount;  
    }  
}  
  
$dc = new DiscountCustomer();  
$dc->setData("Dhanpal ", 10);  
echo $dc->name; // this does not work as $name is protected and hence  
                  // only available in Customer and DiscountCustomer class
```

In the above example, echo \$dc->name will not work as \$name has been defined as a protected variable and hence it is only available in Customer and DiscountCustomer class.

You will learn more about inheritance later in this tutorial series. You should revisit this tutorial and read more on the protected section again when you understand inheritance better.

In PHP5, access specifiers are public by default. This means that if you don't specify an access specifier for a data member or method then the default 'public' is applicable.

4.8 Final Class & Method

A final class is a class that cannot be extended. To declare a class as final, you need to prefix the 'class' keyword with 'final'.

```
final class BaseClass {
    public function myMethod() {
        echo "BaseClass method called";
    }
}

//this will cause Compile error
class DerivedClass extends BaseClass {
    public function myMethod() {
        echo "DerivedClass method called";
    }
}

$c = new DerivedClass();
$c->myMethod();
```

In the above example, BaseClass is declared as final and hence cannot be extended (inherited). DerivedClass tries to extend from BaseClass and hence the compiler will throw a compile error.

Final Method

A final method is a method that cannot be overridden. To declare a method as final, you need to prefix the function name with the 'final' keyword. Example below:

```
class BaseClass {
    final public function myMethod() {
        echo "BaseClass method called";
    }
}

class DerivedClass extends BaseClass {
    //this will cause Compile error
    public function myMethod() {
        echo "DerivedClass method called";
    }
}

$c = new DerivedClass();
$c->myMethod();
```

In the above example, DerivedClass extends from BaseClass. BaseClass has the method myMethod() declared as final and this cannot be overridden. In this case the compiler causes a compile error.

4.9 Interfaces

Interfaces are defined to provide a common function names to the implementers. Different implementers can implement those interfaces according to their requirements. You can say, interfaces are skeletons which are implemented by developers.

As of PHP5, it is possible to define an interface, like this:

```
interface Mail {  
    public function sendMail();  
}
```

Then, if another class implemented that interface, like this:

```
class Report implements Mail {  
    // sendMail() Definition goes here  
}
```

4.10 Abstract Classes

An abstract class is one that cannot be instantiated, only inherited. You declare an abstract class with the keyword `abstract`, like this:

```
abstract class MyAbstractClass {  
    abstract function myAbstractFunction();  
}  
}
```

When inheriting from an abstract class, all methods marked abstract in the parent's class declaration must be defined by the child; additionally, these methods must be defined with the same visibility.

```
<?php  
abstract class AbstractClass  
{  
    // Force Extending class to define this method  
    abstract protected function getValue();
```

```
abstract protected function prefixValue($prefix);

// Common method
public function printOut() {
    print $this->getValue() . "\n";
}

class ConcreteClass1 extends AbstractClass
{
    protected function getValue() {
        return "ConcreteClass1";
    }

    public function prefixValue($prefix) {
        return "{$prefix}ConcreteClass1";
    }
}

$class1 = new ConcreteClass1;
$class1->printOut();
echo $class1->prefixValue('hi i i') . "\n";

?>
```

Note that function definitions inside an abstract class must also be preceded by the keyword `abstract`. It is not legal to have abstract function definitions inside a non-abstract class.

4.11 Exceptions Handling

PHP 5 has an exception model similar to that of other programming languages. Exceptions are important and provides a better control over error handling.

Lets explain three keyword related to exceptions.

- Try - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown".
- Throw - This is how you trigger an exception. Each "throw" must have at least one "catch".
- Catch - - A "catch" block retrieves an exception and creates an object containing the exception information.

When an exception is thrown, code following the statement will not be executed, and PHP will attempt to find the first matching catch block. If an exception is not caught, a PHP Fatal Error will be issued with an "Uncaught Exception" ...

- An exception can be thrown, and caught ("caught") within PHP. Code may be surrounded in a try block.
- Each try must have at least one corresponding catch block. Multiple catch blocks can be used to catch different classes of exceptions.
- Exceptions can be thrown (or re-thrown) within a catch block.

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';

?>
```

In the above example `$e->getMessage` function is used to get error message. There are following functions which can be used from Exception class.

- `getMessage()` - message of exception
- `getCode()` - code of exception
- `getFile()` - source filename
- `getLine()` - source line
- `getTrace()` - an array of the backtrace()
- `getTraceAsString()` - formated string of trace

Creating Custom Exception Handler:

You can define your own custom exception handler. Use following function to set a user-defined exception handler function.

```
string set_exception_handler (callback $exception_handler )
```

Here exception_handler is the name of the function to be called when an uncaught exception occurs. This function must be defined before calling set_exception_handler().

Example:

```
<?php
function exception_handler($exception) {
    echo "Uncaught exception: " , $exception->getMessage(), "\n";
}

set_exception_handler('exception_handler');

throw new Exception('Uncaught Exception');

echo "Not Executed\n";
?>
```

5. Advance PHP -

5.1 Mailing In PHP -

The mail() function allows you to send emails directly from a script. For the mail functions to be available, PHP requires an installed and working email system. The program to be used is defined by the configuration settings in the php.ini file.

Mail configuration options:

Name	Default	Description
SMTP	"localhost"	Windows only: The DNS name or IP address of the SMTP server
smtp_port	"25"	Windows only: The SMTP port number. Available since PHP 4.3
sendmail_from	NULL	Windows only: Specifies the "from" address to be used in email sent from PHP

PHP makes use of mail() function to send an email. This function requires three mandatory arguments that specify the recipient's email address, the subject of the message and the actual message additionally there are other two optional parameters.

`mail(to, subject, message, headers, parameters);`

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n)
parameters	Optional. Specifies an additional parameter to the send mail program

As soon as the mail function is called PHP will attempt to send the email then it will return true if successful or false if it is failed.

Note: - Multiple recipients can be specified as the first argument to the mail() function in a comma separated list.

Example:

Following example will send an HTML email message to xyz@somedomain.com. You can code this program in such a way that it should receive all content from the user and then it should send an email.<html>

```
<head>
<title>Sending email using PHP</title>
</head>
<body>
<?php
    $to = "xyz@somedomain.com";
    $subject = "This is subject";
    $message = "This is simple text message.";
    $header = "From: abc@somedomain.com \r\n";
    $retval = mail ($to,$subject,$message,$header);
    if( $retval == true )
    {
        echo "Message sent successfully...";
    }
    else
    {
        echo "Message could not be sent...";
    }
?>
</body>
</html >
```

Sending HTML Formatted email:

When you send a text message using PHP then all the content will be treated as simple text. Even if you will include HTML tags in a text message, it will be displayed as simple text and HTML tags will not be formatted according to HTML syntax. But PHP provides option to send an HTML message as actual HTML message.

While sending an email message you can specify a Mime version, content type and character set to send an HTML email.

Example:

Following example will send an HTML email message to xyz@somedomain.com copying it to agh@somedomain.com. You can code this program in such a way that it should receive all content from the user and then it should send an email.

```
<html >
<head>
<title>Sending HTML email using PHP</title>
</head>
<body>
<?php
```

```
$to = "xyz@somedomain.com";
$subject = "This is subject";
$message = "<b>This is HTML message.</b>";
$message .= "<h1>This is headline.</h1>";
$header = "From: abc@somedomain.com \r\n";
$header .= "Cc: agh@somedomain.com \r\n";
$header .= "MIME-Version: 1.0\r\n";
$header .= "Content-type: text/html \r\n";
$retval = mail ($to,$subject,$message,$header);
if( $retval == true )
{
    echo "Message sent successfully...";
}
else
{
    echo "Message could not be sent...";
}
?>
</body>
</html >
```

5.2 Sending Free SMS to Mobile with PHP script -

Sending free SMS with PHP is very easy process. We required registered SMS gateway server. & to send the SMS we have to pass only message & Mobile number of recipient to SMS Gateway. Remaining process handle by the SMS Gateway.

An **SMS gateway** is a telecommunications network facility for sending or receiving Short Message Service (SMS) transmissions to or from a telecommunications network that supports SMS. Most messages are eventually routed into the mobile phone networks. Many SMS gateways support media conversion from email and other formats.

Example -

```
<?php
If(isset($_POST['sub']))
{
    $to=$_POST['txtno'];
    $message=$_POST['txtmsg'];

    Header("Location: <URL of SMS Geatway>?username=<gateway
username>&password=<gateway password &to=$to &text=$message")
}
<form method="post" action="">
Phone no: -<input type="text" name="txtno">
<br>
Message: <textarea name="txtmsg"></textarea><br>
```

```
<input type="submit" action="" name="sub" value="send">  
</form>  
?>
```

In header function we have pass Query string provided by SMSGeatway Server along with the username, password, mobile number, & message. When we register on server they provide the QueryString & Username & password.

IMPORTANT Note – According to Telecom Regulatory Authority of India (TRAI) has issued a new regulation on the SMS industry in India on September 27, 2011. They decided to ban on Free SMSGeatway Server in India. So now we can't send free SMS by using above script. But solution is that we can purchase the SMSGeatway server which is located outside to India and then send the sms.

5.3 File Uploading

A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script.

Information in the phpinfo.php page describes the temporary directory that is used for file uploads as upload_tmp_dir and the maximum permitted size of files that can be uploaded is stated as upload_max_filesize. These parameters are set into PHP configuration file php.ini

The process of uploading a file follows these steps

- The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
- The user clicks the browse button and selects a file to upload from the local PC.
- The full path to the selected file appears in the text filed then the user clicks the submit button.
- The selected file is sent to the temporary directory on the server.
- The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
- The PHP script confirms the success to the user.

As usual when writing files it is necessary for both temporary and final locations to have permissions set that enable file writing. If either is set to be read-only then process will fail.

An uploaded file could be a text file or image file or any document.

Creating an upload form:

The following HTML code below creates an uploader form. This form is having method attribute set to post and enctype attribute is set to multipart/form-data

```
<html>
<head>
<title>File Uploading Form</title>
</head>
<body>
<h3>File Upload:</h3>
Select a file to upload: <br />
<form action="/php/file_uploader.php" method="post"
      enctype="multipart/form-data">
<input type="file" name="file" size="50" />
<br />
<input type="submit" value="Upload File" />
</form>
</body>
</html>
```

This will display following result:

File Upload:
Select a file to upload:

Creating an upload script:

There is one global PHP variable called `$_FILES`. This variable is an associate double dimension array and keeps all the information related to uploaded file. So if the value assigned to the input's name attribute in uploading form was file, then PHP would create following five variables:

- `$_FILES['file']['tmp_name']`- the uploaded file in the temporary directory on the web server.
- `$_FILES['file']['name']` - the actual name of the uploaded file.
- `$_FILES['file']['size']` - the size in bytes of the uploaded file.
- `$_FILES['file']['type']` - the MIME type of the uploaded file.
- `$_FILES['file']['error']` - the error code associated with this file upload.

The following example below attempts to copy a file uploaded by the HTML Form listed in previous section page to /var/www/html directory which is document root of your PHP server and it will display all the file's detail upon completion. Please note that if you are going to display uploaded file then don't try with binary files like images or word document.

Here is the code of uploader.php script which will take care of uploading a file.

```
<?php
if( $_FILES['file']['name'] != "" )
{
    copy( $_FILES['file']['name'], "/var/www/html" ) or
        die( "Could not copy file!" );
}
else
{
    die("No file specified!");
}
?>
<html>
<head>
<title>Uploading Complete</title>
</head>
<body>
<h2>Uploaded File Info:</h2>
<ul>
<li>Sent file: <?php echo $_FILES['file']['name']; ?>
<li>File size: <?php echo $_FILES['file']['size']; ?> bytes
<li>File type: <?php echo $_FILES['file']['type']; ?>
</ul>
</body>
</html>
```

5.4 Loading PHP application on web server by ftp Service –

The process of transferring a file from your computer to your website is often referred to as "uploading" that file or "publishing" it. For web hosts that support FTP, or the File Transfer Protocol, you need a program called an "FTP client" to transfer the file.

Download and Install FileZilla

First, go to the FileZilla website download page and obtain the appropriate version for your system. Once you have downloaded the program, you will have to install it. Go to your desktop and double click the file that you have just downloaded. Follow the instructions to install it to your hard disk.

Preliminary Steps

Before you can upload any file to your site, you will also need some information from your web host. In particular, you will need to find out the following:

- The name of the FTP server for your website. For example, your host may tell you that your FTP hostname is "ftp.example.com".
- Your user id or login username for your FTP account.
- Your password for your FTP account.
- The directory where you need to place your files so that they can be seen by a web browser visiting your site. For example, your host may tell you to place the files in a subdirectory called "www" or "public_html" or even the default directory that you see when you log into your FTP site.

When you open the program for the first time you need to enter in all of your connection information to connect to the website. This can be done in one of two ways. If you have many FTP sites, you will want to keep them organized by using the 'Site Manager'. If you're only managing one site, use the 'Quickconnect'.

Site Manager

1. Click File.
2. Select the first option; 'Site Manager'.
3. A window will open called 'Site Manager'. On the left you see a yellow folder that says 'My FTP Sites' and on the right you see 'Site Details'. Make sure 'My FTP Sites' is highlighted.
4. Down in the lower left hand corner click the button that says 'New Site'. A new FTP site connection will appear, give it a name that will help you to remember what website it is. You will now be able to enter all of your information into 'Site Details'.
5. Enter all of your information into 'Site Details'.
Host: In your 'Connecting with FTP' information this is 'Server or Host Name:'
Port: is always 21 and is usually pre-filled in most FTP programs.
Server Type: Leave as FTP or select SFTP (recommended)
Logon type: Set to 'Normal'
User: enter the information from 'User Name:'
Password: Enter your password
6. Hit Save and Exit.

Once you have created your connection you can connect to it by going to File -> Site Manager. Under 'My FTP Sites' select your website and click 'Connect' down at the bottom this will then connect you to your website.

OR

 Look to the icon bar in the upper left of your application. Click on the arrow within the 'Site Manager' icon. A list of your sites will drop down and you can choose the one you want to connect to.

Quickconnect

You can also create a quick connection:

1. At the top of the screen you will see four input boxes: Address, User, Password, and Port.
2. Enter your address in the Address Box.
3. Enter your user number in the User Box.
4. Enter your Bravenet password in the Password box.
5. Leave the port as is.
6. Click 'Quickconnect'. This will connect you to your website.

You can quickly and easily connect to your website - click on the arrow next to 'Quickconnect' for your list of quick connect sites.

After connecting; on the right hand side under 'Remote Site' you should see either one or two folders. Double click the folder, now you are in the root directory of your website. This is where you will save your files.

Now you can begin uploading your web files.

Uploading Files

On the left side you will see a pane that says 'Local Site'. There you can browse through all the files currently on your computer.

To upload a file to your site:

1. Navigate to the file(s) that you wish to upload on your computer.
2. If you want to upload the file(s) to a certain location on your website (ie. a folder called 'images') make sure you have opened that folder by doubling clicking it in your 'Remote Site' pane.
3. Select the file(s) (click to highlight them).
4. Right Click your file(s) and select Upload.

This will upload the file(s) to your website.

If you need to create a folder in your website, do so by right clicking anywhere in your 'Remote Site' pane and selecting 'Create Directory'. Enter a name for your folder and click 'OK'.

TIP: You can upload more than one file at a time! To upload more than one file at a time SHIFT+CLICK all the files (as long as they are grouped together). If the files are not grouped together CTRL+CLICK all the files you wish to upload. Once you have all of the files selected, right click and select upload. You can also upload a folder and all of its contents by selecting the folder.

REMEMBER - Local Site is your computer and Remote Site is your website!

Downloading Files

Downloading files is the exact opposite of uploading. Select your files in your 'Remote Site' pane and Right Click and select Download. You can copy entire folders, create new folders (Directory), and do everything just the same as when uploading.

Renaming uploaded Files

You can rename any file or folder by Right Clicking the file and selecting Rename.

Deleting Files

You can delete any file or folder in one of two ways. Right Click and select Delete. Or click to highlight the file and hit the 'Delete' button on your keyboard. It will ask you; 'Really Delete "FileName"' make sure you have selected to delete the correct file, and if you are SURE you wanted it deleted select 'OK'. Once you delete something it is GONE and can only be uploaded/downloaded again if you have a backup copy.

5.5 Web Services –

A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, and then waits for a corresponding XML response. Because all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications.

A complete web service is, therefore, any service that:

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

Components of Web Services

The basic Web services platform is XML + HTTP. All the standard Web Services works using following components

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)
- XML-RPC(RPC stands for Remote Procedure)

SOAP

SOAP is an XML-based protocol for exchanging information between computers. SOAP is an application of the XML specification.

- SOAP is acronym for Simple Object Access Protocol
- SOAP is a communication protocol
- SOAP is designed to communicate via Internet
- SOAP can extend HTTP for XML messaging
- SOAP provides data transport for Web services
- SOAP can exchange complete documents or call a remote procedure
- SOAP can be used for broadcasting a message
- SOAP is platform and language independent
- SOAP is the XML way of defining what information gets sent and how

A SOAP message is an ordinary XML document containing the following elements.

- **Envelope:** (Mandatory)
Defines the start and the end of the message.
- **Header:** (Optional)
Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate end point.
- **Body:** (Mandatory)
Contains the XML data comprising the message being sent.
- **Fault:** (Optional)
An optional Fault element that provides information about errors that occurred while processing the message

UDDI

UDDI is an XML-based standard for describing, publishing, and finding Web services.

- UDDI stands for Universal Description, Discovery and Integration.
- UDDI is a specification for a distributed registry of Web services.
- UDDI is platform independent, open framework.
- UDDI can communicate via SOAP, CORBA, Java RMI Protocol.
- UDDI uses WSDL to describe interfaces to web services.

- UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
- UDDI is an open industry initiative enabling businesses to discover each other and define how they interact over the Internet.

WSDL

- WSDL stands for Web Services Description Language
- WSDL is an XML based protocol for information exchange in decentralized and distributed environments.
- WSDL is the standard format for describing a web service.
- WSDL definition describes how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of UDDI, an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL was developed jointly by Microsoft and IBM.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'

WSDL Usage:

WSDL is often used in combination with SOAP and XML Schema to provide web services over the Internet. A client program connecting to a web service can read the WSDL to determine what functions are available on the server. Any special datatypes used are embedded in the WSDL file in the form of XML Schema. The client can then use SOAP to actually call one of the functions listed in the WSDL.

How WSDL Work?

You can build a Java-based Web Service on Solaris that is accessible from your Visual Basic program that runs on Windows. You can also use C# to build new Web Services on Windows that can be invoked from your Web application that is based on JavaServer Pages (JSP) and runs on Linux.

XML-RPC

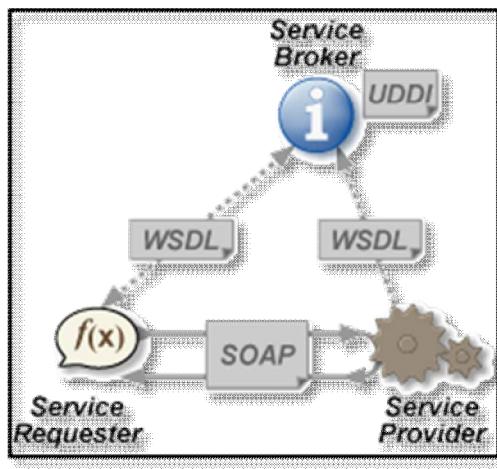
RPC stands for Remote Procedure Call. As its name indicates it is a mechanism to call procedure or function available on a remote computer. Remote Procedure Calls (RPC) are a much older technology than the Web. Effectively, RPC gives developers a mechanism for defining interfaces that can be called over a network. These interfaces can be as simple as a single function call or as complex as a large API.

- XML-RPC permits programs to make function or procedure calls across a network.
- XMLRPC uses the HTTP protocol to pass information from a client computer to a server computer.
- XML-RPC uses a small XML vocabulary to describe the nature of requests and responses.
- XML-RPC client specify a procedure name and parameters in the XML request, and the server returns either a fault or a response in the XML response.
- XML-RPC parameters are a simple list of types and content - structs and arrays are the most complex types available.
- XML-RPC has no notion of objects and no mechanism for including information that uses other XML vocabularies.
- With XML-RPC and web services, however, the Web becomes a collection of procedural connections where computers exchange information along tightly bound paths.
- XML-RPC emerged in early 1998; it was published by UserLand Software and initially implemented in their Frontier product.

Advantages of using Web Services

- Exposing the existing function on to network:
- Connecting Different Applications i.e. Interoperability:
- Standardized Protocol:
- Low Cost of communication:

Web Services Architecture



Web Service Roles

There are three major roles within the web service architecture:

- **Service provider:**

This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

- **Service requestor**

This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

- **Service registry**

This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearinghouse for companies and their services.

Based on the Web Service Architecture we will create following two components as a part of Web Services implementation

- **Service provider or publisher:**

This is the provider of the web service. The service provider implements the service and makes it available on the Internet or intranet. We will write and publish a simple web Service using .NET SDK.

- **Service requestor or consumer**

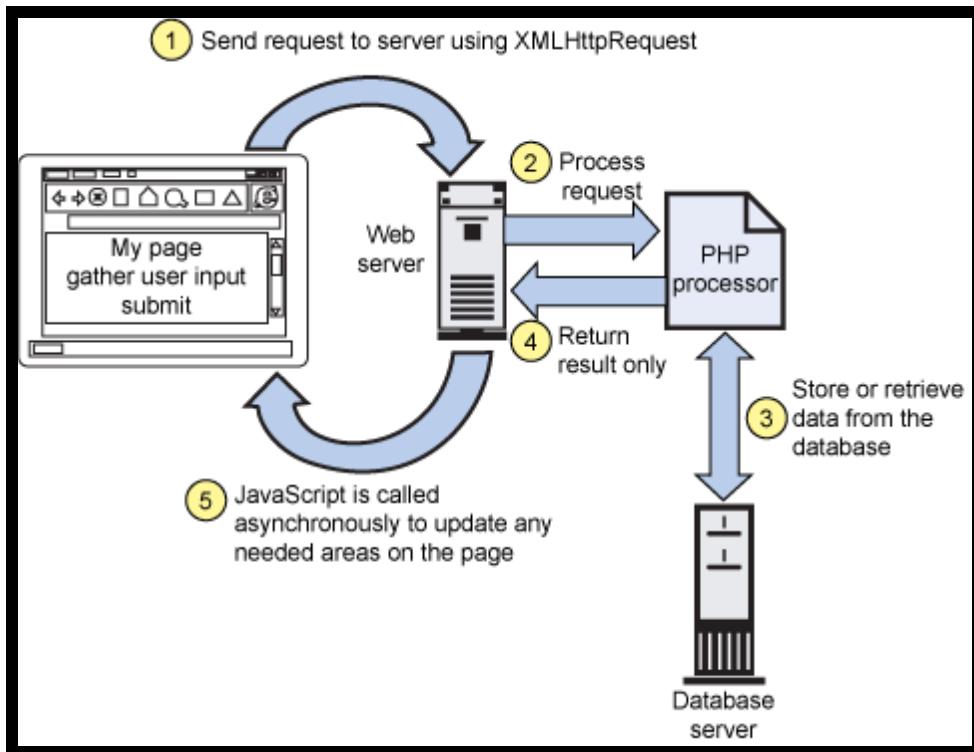
This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request. We will also write two Web Service requestors: one Web-based consumer (ASP.NET application) and another Windows application-based consumer.

6. Ajax with PHP

6.1 Introduction Of Ajax

- AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script.
- Ajax uses XHTML for content and CSS for presentation, as well as the Document Object Model and JavaScript for dynamic content display.
- Conventional web application transmit information to and from the sever using synchronous requests. This means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX when submit is pressed, JavaScript will make a request to the server, interpret the results and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background
- Intuitive and natural user interaction. No clicking required only Mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven

6.2 How AJAX Works



AJAX is based on internet standards, and uses a combination of:

- XMLHttpRequest object (to exchange data asynchronously with a server)
- JavaScript/DOM (to display/interact with the information)
- CSS (to style the data)
- XML (often used as the format for transferring data)

XMLHttpRequest

JavaScript object that performs asynchronous interaction with the server

JavaScript

- Loosely typed scripting language
- JavaScript function is called when an event in a page occurs
- Glue for the whole AJAX operation

DOM

- API for accessing and manipulating structured documents
- Represents the structure of XML and HTML documents

CSS

- Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript

Here is the list of famous web applications which are using AJAX

- **Google Maps**
A user can drag the entire map by using the mouse instead of clicking on a button or something
<http://maps.google.com/>
- **Google Suggest**
As you type, Google will offer suggestions. Use the arrow keys to navigate the results
<http://www.google.com/webhp?complete=1&hl=en>
- **Gmail**
Gmail is a new kind of webmail, built on the idea that email can be more intuitive, efficient and useful
<http://gmail.com/>
- **Yahoo Maps (new)**
Now it's even easier and more fun to get where you're going!
<http://maps.yahoo.com/>

6.3 Steps of AJAX Operation

1. Create an Ajax object based on the browser what user using. If user using IE we have to create object to "ActiveXObject" class otherwise create object to "XmlHttpRequest" class.

```
// creates XMLHttpRequest Instance
function createXmlHttpRequestObject(){

    // will store XMLHttpRequest object
    // at here
    var xmlhttp;

    // works all except IE6 and older
    try
    {

        // try to create XMLHttpRequest object
        xmlhttp = new XMLHttpRequest();
    }
    catch(e)
    {
```

```
// assume IE 6 or older
try
{
    xmlhttp = new ActiveXObject("Microsoft.XMLHttp");
}
catch(e){ }
}

// return object or display error
if (!xmlhttp)
    alert("Error creating the XMLHttpRequest Object");
else
    return xmlhttp;
}
```

2. After creating the XMLHttpRequest object, we will look what process that we can do. The key is understand object's methods and properties. We learn by practice.

```
function process()
{
    if(xmlhttp)
    {
        try
        {
            xmlhttp.open("Get", "test.txt", true);
            xmlhttp.onreadystatechange = handleRequestStateChange;
            xmlhttp.send(null);
        }
        catch(e)
        {
            alert("Can't connect to server\n" + e.toString());
        }
    }
}
```

From the code, we see 2 methods:

- `open("method", "URL"[, asyncFlag[, "userName"[, "password"]]])` initialize the request parameters
- `send(content)` Performs the HTTP Request

We see 1 property:

- `onreadystatechange` Used to set the callback function that handles request state changes

- we will see how to handle response from server. We know a property named "readyState". This property have possible value like:

0 = uninitialized
1 = loading
2 = loaded
3 = interactive
4 = complete

We will use this property to build little fun for your client. You want to show the AJAX steps at browser

```
function handleRequestStateChange()
{
    myDiv = document.getElementById("myDivElement");

    if(xmlHttp.readyState == 1)
    {
        myDiv.innerHTML += "Request status: 1 (loading) <br/>";
    }
    else if (xmlHttp.readyState == 2)
    {
        myDiv.innerHTML += "Request status: 2 (loaded) <br/>";
    }
    else if (xmlHttp.readyState == 3)
    {
        myDiv.innerHTML += "Request status: 3 (interactive) <br/>";
    }
    else if (xmlHttp.readyState == 4)
    {
        if(xmlHttp.status == 200)
        {
            try
            {
                response = xmlHttp.responseText;
                myDiv.innerHTML += "Request status: 4 (complete). Server said: <br/>";
                myDiv.innerHTML += response;
            }
            catch(e)
            {
                alert("Error reading the response: " + e.toString());
            }
        }
        else
    }
}
```

```
{  
    alert("Problem retrieving data: \n" + xmlhttp.statusText);  
}  
  
}  
  
}
```

In this code, in every steps, we will write to browser what happen.

- Now, we will combine codes. We can see how they work. First, create a file named "ajaxclient.html" Enter following code & also create 1 text file name it is as test.txt with some contents & place this both file in same Directory.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
<title>Ajax Client</title>  
<script type="text/javascript">  
  
var xmlhttp = createXmlHttpRequestObject();  
// creates XMLHttpRequest Instance  
function createXmlHttpRequestObject(){  
  
    // will store XMLHttpRequest object  
    // at here  
    var xmlhttp;  
  
    // works all except IE6 and older  
    try  
    {  
  
        // try to create XMLHttpRequest object  
        xmlhttp = new XMLHttpRequest();  
    }  
    catch(e)  
    {  
        // assume IE 6 or older  
        try  
        {  
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");  
        }  
        catch(e){ }  
    }  
}
```

```
// return object or display error
if (!xmlHttp)
    alert("Error creating the XMLHttpRequest Object");
else
    return xmlHttp;
}

function process()
{
    if(xmlHttp)
    {
        try
        {
            xmlHttp.open("Get", "test.txt", true);
            xmlHttp.onreadystatechange = handleRequestStateChange;
            xmlHttp.send(null);
        }
        catch(e)
        {
            alert("Can't connect to server\n" + e.toString());
        }
    }
}

function handleRequestStateChange()
{
    myDiv = document.getElementById("myDivElement");

    if(xmlHttp.readyState == 1)
    {
        myDiv.innerHTML += "Request status: 1 (Loading) <br/>";
    }
    else if (xmlHttp.readyState == 2)
    {
        myDiv.innerHTML += "Request status: 2 (Loaded) <br/>";
    }
    else if (xmlHttp.readyState == 3)
    {
        myDiv.innerHTML += "Request status: 3 (Interactive) <br/>";
    }
    else if (xmlHttp.readyState == 4)
    {
        if(xmlHttp.status == 200)
        {
            try
            {
                response = xmlHttp.responseText;
            }
            catch(e)
            {
                alert("Error reading response\n" + e.toString());
            }
        }
    }
}
```

```
        myDiv.innerHTML += "Request status: 4 (complete). Server  
said: <br/>";  
        myDiv.innerHTML += response;  
    }  
    catch(e)  
    {  
        alert("Error reading the response: " + e.toString());  
    }  
}  
else  
{  
    alert("Problem retrieving data: \n" + XMLHttpRequest.statusText);  
}  
}  
}  
</script>  
</head>  
  
<body onload="process()">  
<div id="myDivElement" />  
</body>  
</html>
```

XMLHttpRequest Methods

- **abort()**
Cancels the current request.
- **getAllResponseHeaders()**
Returns the complete set of HTTP headers as a string.
- **getResponseHeader(headerName)**
Returns the value of the specified HTTP header.
- **open(method, URL)**
open(method, URL, async)
open(method, URL, async, userName)
open(method, URL, async, userName, password)
Specifies the method, URL, and other optional attributes of a request.

The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods, such as "PUT" and "DELETE" (primarily used in REST applications), may be possible

The "async" parameter specifies whether the request should be handled asynchronously or not . "true" means that script processing carries on after the

- send() method, without waiting for a response, and "false" means that the script waits for a response before continuing script processing.
- send(content)
Sends the request.
- setRequestHeader(label, value)
Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Properties

- onreadystatechange
An event handler for an event that fires at every state change.
- readyState

The readyState property defines the current state of the XMLHttpRequest object.

Here are the possible values for the readyState property:

- readyState=0 after you have created the XMLHttpRequest object, but before you have called the open() method.
 - readyState=1 after you have called the open() method, but before you have called send().
 - readyState=2 after you have called send().
 - readyState=3 after the browser has established a communication with the server, but before the server has completed the response.
 - readyState=4 after the request has been completed, and the response data have been completely received from the server.
-
- responseText
Returns the response as a string.
 - responseXML
Returns the response as XML. This property returns an XML document object, which can be examined and parsed using W3C DOM node tree methods and properties.
 - status
Returns the status as a number (e.g. 404 for "Not Found" and 200 for "OK").
 - statusText
Returns the status as a string (e.g. "Not Found" or "OK").

6.4 Ajax object in Different Browser –

Ajax can run in all type of browsers but there is a conflict in older (IE 5 and IE 6), and newer browsers (Mozilla, Netscape, IE 8, Chrome, Safari, Opera).

It can be rectified by checking the older and newer browser and implementing the below code.

For New Browsers: `xmlhttp=new XMLHttpRequest();`

For older Browsers: `xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");`

Code for Checking the older and Newer Browser:

```
if (window.XMLHttpRequest)
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();
    }
else
    {// code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
```

The code above tells that it first it gets the browser windows XMLHttpRequest object if it present then it's a newer browser. If it does not get the browser XMLHttpRequest object then it's an older browser.

If its older browser then it create an ActiveXObject to xmlhttp.

7. CMS Technology in PHP

7.1 Introduction of CMS

A web site's content is a combination of different components like texts, graphics, images, scripts, embedded files such as flash animations, audio/video streams or downloadable files. All of these may be components of one document (or HTML page in case of a web content management system).

Content management systems (CMS) are computer software systems for organizing, displaying and facilitating collaborative creation of this content.

One principle of many content management systems is, to separate the content from the layout, which makes it easier to preset the same content in different layouts for different media ("cross-media publishing") like web browser and printer. Separating content and layout also enables website designers to concentrate on the presentation, while others attend the content.

This can be achieved by storing the content and the layout in different resources and dynamically merge them together to the final document.

Advantages of Content Management Systems

- CMS facilitate the collaborative creation of websites. People can concentrate on the content while others care for the template to present the content. Also many CMS provide systems to enable users to add or modify content via their web browser
- CMS make it easier to display the same content in different ways, like a normal view for web browsers and a printer friendly view
- CMS make it easier to create new documents as one can concentrate on the content and do not have to care about the layout
- CMS make it easier to modify the layout of a website as one only has to modify the template at a single source instead of having to modify each single page to reflect the change
- CMS often can automatically create additional content like menus, sitemaps etc.
- CMS often provide methods to find content, for example by providing search functionality on the content

7.2 What is Joomla?

Joomla! is software that lets you make and update web pages easily.

You can think of a Joomla! website as bringing together three elements.

- Your content, which is mainly stored in a database.

- Your template, which controls the design and presentation of your content (such as fonts, colors and layout).
- Joomla! which is the software that bring the content and the template together to produce webpages.

7.3 Installation of joomla-

1. First Download latest version of joomla software from official Web site, this software is freely available on Internet
2. Unzip this software & copy or upload this unzipped folder on web server's htdocs folder.
3. Go to browser & type The following link to access the joomla admin page.

http:\\domainname\\projectname

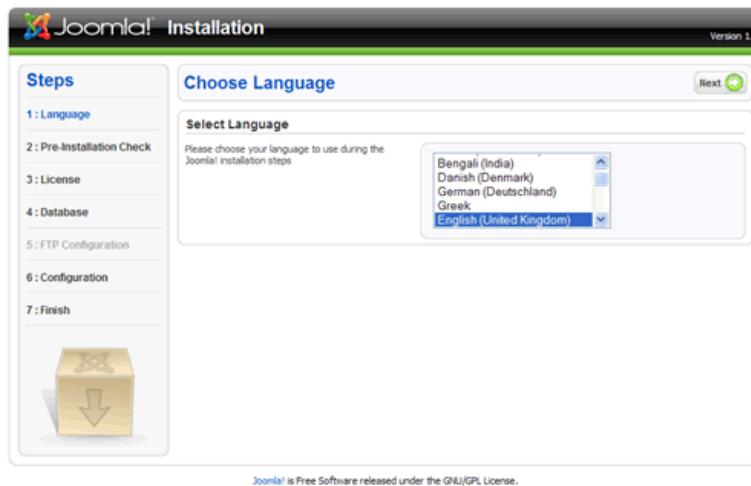
- DomainName is the name or ip of Webserver
- Projectname is the folder name which Content all joomla files.

eg.

http:\\localhost\\joomlaProj

7.4 Steps of Joomla installation & Configuration

1. Choose Language:



Here is the first look at some of the internationalization features of Joomla, you can select amongst many languages for the installation instructions.

2. Pre-Installation Check

The screenshot shows the Joomla! Installation interface at step 2: Pre-Installation Check. The title bar says "Joomla! Installation" and "Version 1.5". On the left, a vertical sidebar lists steps: 1: Language, 2: Pre-Installation Check (which is active), 3: License, 4: Database, 5: FTP Configuration, 6: Configuration, and 7: Finish. Step 6 has a yellow download icon. The main content area is titled "Pre-Installation Check" and displays a table of system requirements:

	PHP version >= 4.3.0	Yes
- zip compression support	Yes	
- XML support	Yes	
- MySQL support	Yes	
MB language is default	Yes	
MB string overload off	Yes	
configuration.php Writable	Yes	

Below this, under "Recommended settings:", there is a table comparing Directive values:

Directive	Recommended	Actual
Safe Mode:	Off	Off
Display Errors:	On	On
File Uploads:	On	On
Magic Quotes Runtime:	Off	Off
Register Globals:	Off	Off
Output Buffering:	Off	Off
Session auto start:	Off	Off

At the bottom of the page, it says "Joomla! is Free Software released under the GNU/GPL License."

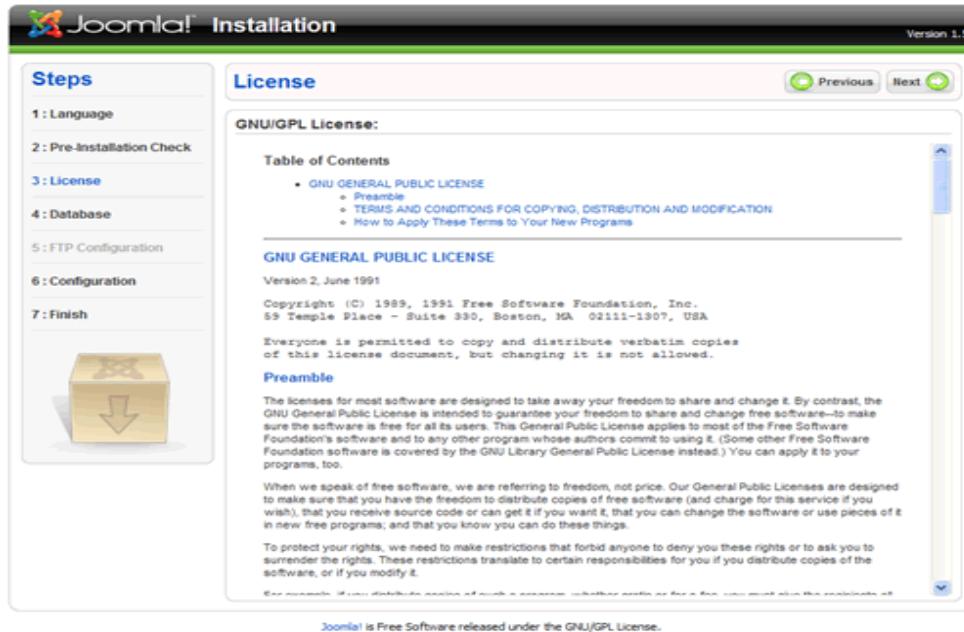
A critical part in the installation process, this checks if all the minimum system requirements are met.

The first set are required minimums, if they are red (not met) then you need to find a new environment (change hosts), talk them into changing their environment (upgrading PHP for example). Note that the last item is a permissions issue on a file that is much easier to rectify. You can usually change permissions through the cpanel provided by your host. This tool is generally an industry standard.

The second set is recommended settings. If you don't meet them you can still install Joomla but it you experience problems with functionality and security.

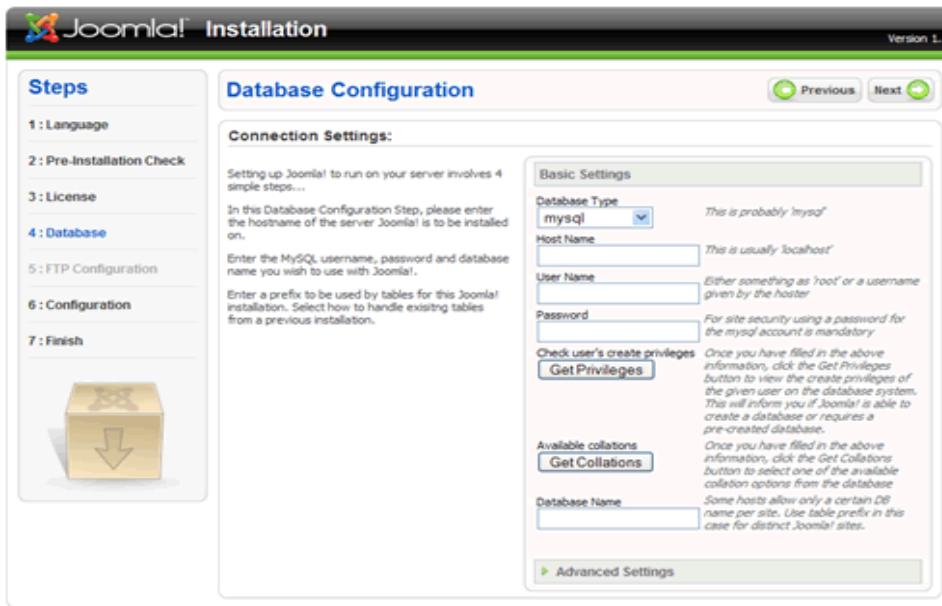
Once you are green to go, click next.

3. License



Joomla is released under a GNU/GPL license. One of the most common questions regarding this license is “can I remove the footer link that says Powered by Joomla”. It’s actually perfectly OK to do this; you just have to keep the copyright statement in the source code. However, I would recommend that you keep the link.

4. Database Configuration



This is one of the main pages of the installation process, it's where you need to enter important information about the database that your Joomla site will use.

The hostname will almost always be "localhost"

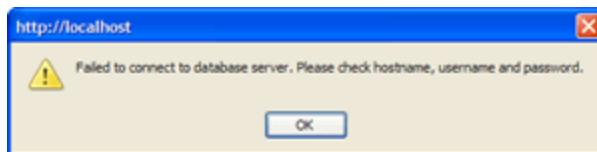
The Username and password will have been provided by your hosting company, usually in an email you got when you created the account.

If you are installing on a localhost using WAMP or XAMPP, the Username is usually "root" and the password is nothing/blank.

Once you have entered this information, click Get Privileges and the Joomla installer checks to see if that user has rights to create a database. You should see this message:

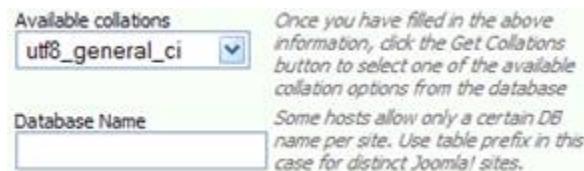


If you made an error, or the user does not have sufficient permissions, then you get this message (after a small delay):



If the user you have does not have permissions then you can ask your hoster to pre-create a SQL database for you to use.

Available collations refer to various character sets available for different languages. When you click the button you get a list of what is available:



Pick a Database name for the SQL database that Joomla will use. Use some sort of name that is not confusing. Other scripts use SQL databases and before you know it you might

have several on your server and will need to tell them apart. Don't use spaces in the name.

Note: if your user did not have database creation privileges and you were provided with a database instead you would obviously put that in as the name.

The advanced settings are concerned with what content the site starts with and also the table prefixes mentioned above.

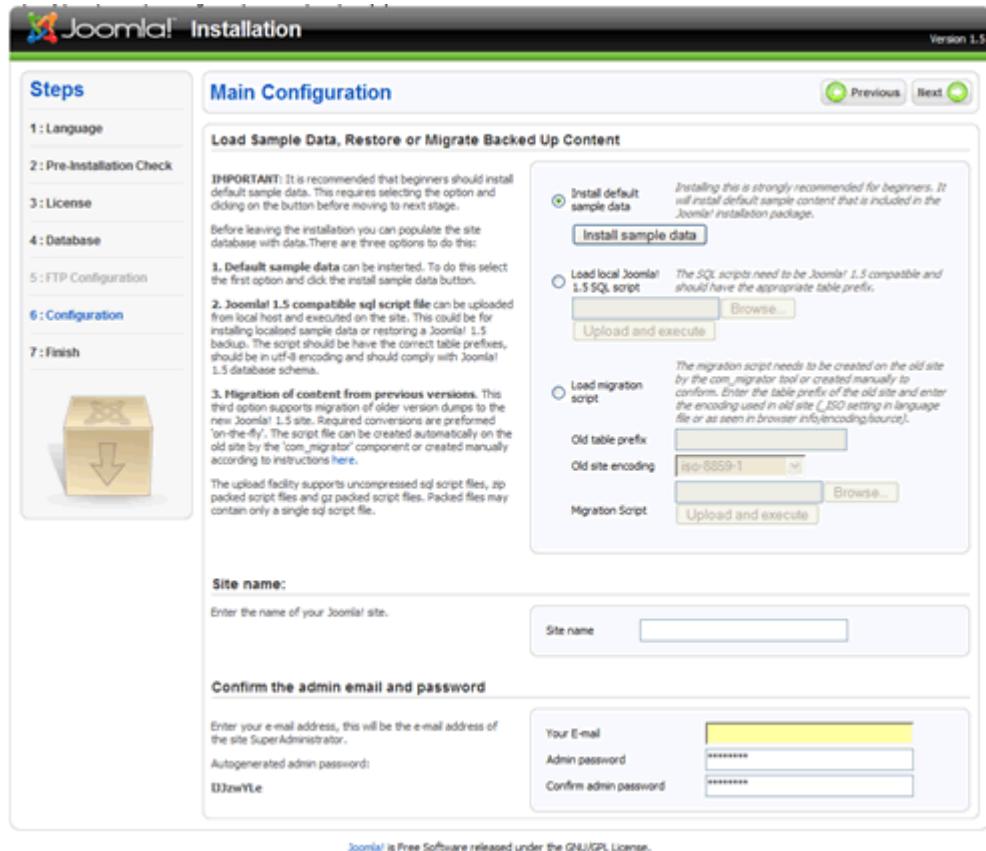


If you have an existing site and you are reinstalling over the top, you will need to select Drop Existing Tables. If you need to keep a back of them, select Backup Old Tables.

5. FTP Configuration

Enter the ftp username & password & root path, if you installing on linux platform.

6. Main Configuration



The Main Configuration page determines how you will insert content into your site. You have three choices:

- **Install Sample Data**

This installs the default Joomla content that you have probably seen all over the web with "Welcome to Joomla". Note that is also includes all the menus, navigation links and sections/categories. If you are learning how to use Joomla this is highly recommended. Its easier to adapt and revise than to start from scratch.

- **Load SQL script**

This is a SQL file that might have a customized set of content

- **Migration from previous versions**

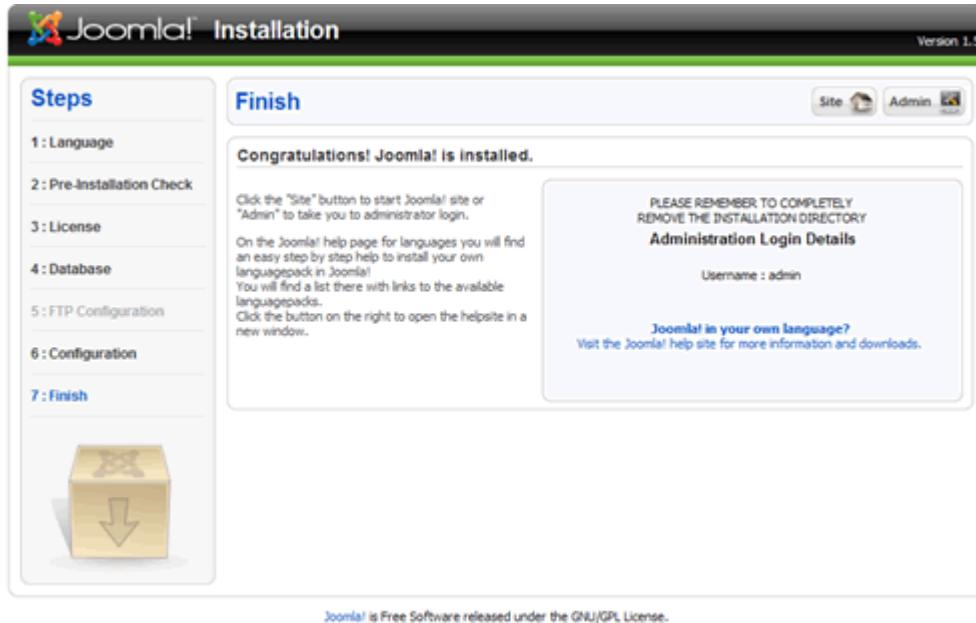
This is a special function that is part of a process to migrate a Joomla site running on 1.0 to 1.5 and requires a special component to do so. This is discussed more in the Appendix.

Give your site a name (pay attention to SEO keywords) and then enter the super administration information. This will be the first user in the site and automatically gets that status. Note that if

you don't change the password, it will use the one shown on the left. Make sure you write it down!

7. Finish

Cross your eyes, close your fingers and click next. Hopefully you will see this screen



If you do get this result, you successfully Installed joomla on your web server, only **remove the installation folder** from your joomla project Folder. & now you can access joomla Website & Administrator page.

- To open the web site the URL address is-
http://domainname/projectname/
- To open the Administrator page of web site the URL address is-
http://domainname/projectname/Administrator

7.5 Various Manager in Joomla

1. Template Manager

The Template Manager is where you assign a default Template to your Joomla! web site. You can also edit and preview Templates here.

In Joomla!, the visual layout of both the Front-end and Back-end of your site is controlled by the Template. Templates are extensions that contain layout and style information that tells Joomla! exactly how to draw each page of your site.

When you first install Joomla!, one Back-end Template and two Front-end templates are included. Other Templates can be installed from third-party developers as Extensions.

Select Extensions --> Template Manager from the drop-down menu in the back-end of your Joomla! installation.

#	Template Name	Default	Assigned	Version	Date	Author
1	beez			1.0.0	19 February 2007	Angie Radtke/Robert Deutz
2	rhuk_milkyway		★	1.0.2	11/20/06	Andy Miller

At the top right you will see the toolbar:



- **Default.** Select the Template that you want to be the default Template. Then click this button. The default star symbol will show in the Default column, indicating that this is now the default Template.
- **Edit.** Select one item and click on this button to open it in edit mode. If you have more than one item selected, the first item will be opened. You can also open an item for editing by clicking on its Title or Name. See the section below called Template - Edit for information on the edit screen.
- **Help.** Opens this Help Screen.

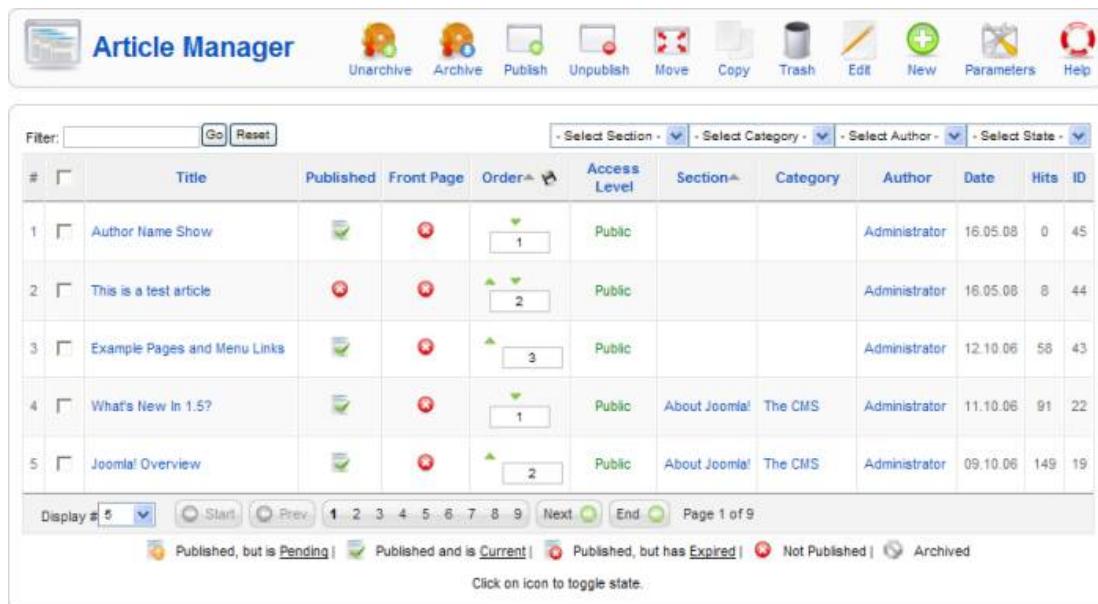
2. FrontPage Manager

Display the list of available front page articles we can publish & unpublished the articles using its manager.

3. Article Manager

The Article Manager is the place in the back-end where you can add and manage all of the articles for your web site.

Click the Article Manager icon in the Control Panel, or click 'Article Manager' in the 'Content' menu in the Back-end of your Joomla! installation.



The screenshot shows the Joomla Article Manager interface. At the top, there is a toolbar with icons for Unarchive, Archive, Publish, Unpublish, Move, Copy, Trash, Edit, New, Parameters, and Help. Below the toolbar is a search bar with 'Filter:' and 'Go'/'Reset' buttons, followed by dropdown menus for 'Select Section', 'Select Category', 'Select Author', and 'Select State'. The main area is a table with the following data:

#	Title	Published	Front Page	Order	Access Level	Section	Category	Author	Date	Hits	ID	
1	Author Name Show				1	Public		Administrator	16.05.08	0	45	
2	This is a test article				2	Public		Administrator	16.05.08	8	44	
3	Example Pages and Menu Links				3	Public		Administrator	12.10.06	56	43	
4	What's New In 1.5?				1	Public	About Joomla!	The CMS	Administrator	11.10.06	91	22
5	Joomla! Overview				2	Public	About Joomla!	The CMS	Administrator	09.10.06	149	19

Below the table, there are buttons for 'Display #' (set to 5), 'Start', 'Prev', 'Next', 'End', and 'Page 1 of 9'. A legend at the bottom explains the icons: Published, but is Pending | Published and is Current | Published, but has Expired | Not Published | Archived. A note says 'Click on icon to toggle state.'

Column Headers

- **#.** An indexing number automatically assigned by Joomla! for ease of reference.
- **Checkbox.** Check this box to select one or more items. To select all items, check the box in the column heading. After one or more boxes are checked, click a toolbar button to take an action on the selected item or items. Many toolbar actions, such as Publish and Unpublish, can work with multiple items. Others, such as Edit, only work on one item at a time. If multiple items are checked and you press Edit, the first item will be opened for editing.

- **Title.** The name of the item. For a Menu Item, the Title will display in the Menu. For an Article, Section, or Category, the Title may optionally be displayed on the web page. This entry is required. You can open the item for editing by clicking on the Title.
- **Published.** The Article's Published status. The possible values are shown under the main window, as shown below:



You can toggle the Published state on and off by clicking on the icon in this column. A status of Pending means that an Article's Start Publishing Date is in the future. You can hover on the icon to see an Article's Start and Stop Publishing dates.

- **Front Page.** Whether or not the Article will show on the Front Page. You can change an Article's published state by clicking on the icon in the column.
- **Order.** The order to display items. If the list is sorted by this column, you can change the order by clicking the arrows or by entering the sequential order and clicking 'Save Order'. Note that the display order on a page is set in the Parameters - Advanced section for each Menu Item. If that order is set to use something other than 'Order' (for example, 'Title - Alphabetical'), then the order value in this screen will be ignored. If the Menu Item Order parameter is set to use 'Order', then the items will display on the page based on the order in this screen.
- **Access Level.** Who has access to this item. Current options are:
 - Public: Everyone has access
 - Registered: Only registered users have access
 - Special: Only users with author status or higher have access

You can change an item's Access Level by clicking on the icon in the column.

- **Section.** The Section this item belongs to. Clicking on the Section Title opens the Section for editing. See Section Manager - Edit.
- **Category.** The Category this item belongs to. Clicking on the Category title opens the Category for editing. See Category Manager - Edit.
- **Author.** Name of the Joomla! User who created this item. Clicking on the Author opens the User for editing. See User Manager - Edit.
- **Date.** The date this Article was created. This date is added automatically by Joomla!, but you may change it in the Parameters - Article section of the Article Manager - New/Edit.

- **Hits.** The number of hits for an Article. A hit is the number of times a page has been viewed. Hits can be reset to 0 in the Article Manager - New/Edit screen.
- **ID.** The ID number. This is a unique identification number for this item assigned automatically by Joomla!. It is used to identify the item internally, for example in internal links. You can not change this number.
- **Display #.** The number of items to display on one page. If there are more items than this number, you can use the page navigation buttons (Start, Prev, Next, End, and page numbers) to navigate between pages. Note that if you have a large number of items, it may be helpful to use the Filter options, located above the column headings, to limit which items display.

Toolbar

At the top right you will see the following toolbar:



The functions are:

- **Unarchive.** To remove one or more Articles from Archived status, select them and press this button. Archived Articles are retained on the site and are available for viewing on Menu Items with a Type of Archive Layout. Note that when an Article is unarchived, its status is changed to 'Not Published'. Also, if an Article does not have a status of 'Archived', this button has no effect.
- **Archive.** To change one or more Articles to Archived status, select them and press this button. Archived Articles are retained on the site and are available for viewing on Menu Items with a Type of Archive Layout. They can not be published to other pages while set to Archived status.
- **Publish.** To publish one or more items, select them and click on this button.
- **Unpublish.** To unpublish one or more items, select them and click on this button.
- **Move.** Select one or more items and click on this button to move them. A new screen will display showing the possible "Move to" locations on the left and the list of item(s) being moved on the right. To complete the move, select the desired "Move to" location and press the Move button. To cancel the operation, press Cancel.

- **Copy.** Select one or more items and click this button to copy them. A new screen will display showing the possible "Copy to" locations on the left and the list of item(s) being copied on the right. To complete the copy, select the desired "Copy to" location and press the Copy button. To cancel the operation, press Cancel.
- **Trash.** Select one or more Articles and click on this button to move them to the Trash Manager. Note that Articles can be restored from the Trash Manager as long as they are not permanently deleted. See Trash Manager for more information.
- **Edit.** Select one item and click on this button to open it in edit mode. If you have more than one item selected, the first item will be opened. You can also open an item for editing by clicking on its Title or Name.
- **New.** Click on this button to create a new item. You will enter the New page for this item.
- **Parameters.** Click this button to open the Global Configuration window. This window allows you to set default parameters for Articles. This default parameter will take effect if the corresponding Menu Item parameter and Article parameter are both set to 'Use Global'. See Global Configuration below.

4. Menu Manager

The Menu Manager controls the framework, or container, of the Menu(s) in Joomla! The actual menu functionality and appearance of a Menu is created automatically by Joomla! by triggering a new instance of mod_mainmenu when a Menu is first created. It is this, together with the Menu Item(s) selected, that creates the finished Menu.

Toolbar:



- **Copy:** Select the radio button of the Menu that is to be the copied. Click the Copy icon. The Copy Menu screen will appear.
- **Delete:** To delete a Menu, select the radio button next to the name of the Menu to be deleted and click the Delete icon. A pop up dialogue box will appear requesting a confirmation of the required delete operation. Click the OK button to delete the Menu, and the Menu file will be permanently deleted from the server. Select the Cancel button to abort the delete operation. Upon completing the deletion - or cancelling the action - the administrator will be returned to the Menu Manager screen. It is possible to delete more than one Menu at a time.

- Edit: Click the name of the Menu to be edited, or select the radio button next to the name and click the Edit icon. This opens the selected Menu file in the Menu Details (edit) screen.
- New: Click the New icon to create a new Menu. The Menu Details (new) screen will be displayed.
- Help: Click the Help icon at any time to view this Help Screen.

5. Media Manager

The Joomla! Media Manager is the main control screen for all the various images that can be used in Content Items and various Menus, Category, and Section displays. From here it is possible to upload new images, edit existing image details, and create new directories.

NOTE: It is not possible to move or copy images between directories in the standard Media Manager. This task must either be carried out directly in the joomla_root/images directory of the Joomla! web site, or an Extension needs to be added to the installation to allow this functionality.

NOTE: The Media Manager is not the only location where images are stored in a Joomla! installation. Individual extensions and templates may very well have their own image directories.

Toolbar:



Upload: Click the Upload icon to transfer an image file from a local computer to the joomla_root/images directory. Ensure that the action is activated whilst in the desired directory where the new image is to be stored. This function is used in conjunction with the File Upload field below

Create: Click the Create icon to create a new directory within the joomla_root/images directory tree. Ensure that the action is activated whilst in the desired directory where the new directory is to be stored. This function is used in conjunction with the Create Directory field below.

Cancel: Click the Cancel icon to exit the Media Manager and return to the Control Panel screen. Any details entered but not actioned will be lost.

6. Language Manager

The Language Manager displays a list of the Languages available for the Front-end of the Joomla! web site. These Languages apply to the Joomla! core only and may not affect any Components, Modules, or Mambots that may be in use. A Language is only available to Users if it is published.

Toolbar:



Publish: To publish a Language, select the radio button next to the name of the Language and click the Publish icon.

New: Click the New icon to upload and install a new Language to the Language Manager. The Install new Language - Site screen will be displayed.

Edit: Click the name of the Language, or select the radio button next to the name, and click the Edit icon. This opens the selected Language file in the Language Editor where it is possible to edit the phrases and the wording of particular screen labels, references, and messages of the selected Language.

Delete: To delete a Language, select the radio button next to the name of the Language to be deleted and click the Delete icon. A pop up dialogue box will appear requesting a confirmation of the required delete operation. Click the OK button to delete the Language, and the Language file will be deleted from the server, or select the Cancel to abort the delete operation.

7. User Manager

This page shows a list of the Users registered on the web site. Users can be created, edited, blocked, and deleted from the functions of this screen.

Toolbar:



Logout: Select the User by checking the check box next to their name, and click the Logout icon. This will forcibly disconnect them from their current session, and return them to a Guest status.

Delete: Select the User, by checking the check box next to their name, and click the Delete icon. This will delete them from the database. A pop up will appear seeking a confirmation as to whether to delete the selected Item (i.e. the User!). Clicking OK will complete the deletion. Clicking Cancel will terminate the process.

Edit: Select the User to Edit either by checking the check box next to their name and clicking the Edit icon, or click the Username. Both of these actions will open the User : Edit screen. Here it is possible to configure all the details, User Group, and Parameters associated with the User.

New: Click the New icon to create a new User. The User : Add screen will be displayed. Here it is possible to configure all the details, User Group, and Parameters associated with the new User. The system will e-mail the User to confirm their registration on the web site.

8. Module Manager

The Module Manager: Site is the main control panel for all Site Modules (Modules that can be published to the Front-end of Joomla!), both published and unpublished, installed on the web site. It provides access to the main creation and editing functions.

Toolbar:



Publish: Check the check box next to the Module that is to be Published (displayed). Multiple Modules may be selected. Then click the Publish icon.

Unpublish: Check the check box next to the Module that is to be Unpublished (not displayed). Multiple Modules may be selected. Then click the Unpublish icon. Unpublishing a Module does not delete it.

Copy: Check the check box next to the Module to be copied. Click the Copy icon. A copy of the selected Module will appear in the Module Manager. The name of the Module will be prefixed by Copy of to distinguish it from the original.

Delete: Check the check box next to the Module to be deleted. Click the Delete icon to delete the Module selected. A cautionary pop up dialogue will appear asking for confirmation of the proposed deletion. Multiple Modules can be deleted at the same time. Click the OK button to delete the Module. Select the Cancel button to abort the delete operation.

Edit: Check the check box next to the Module that is to be edited. Click the Edit icon to edit the details and parameters of an existing Module. The Module: Edit [module_name] screen will open - see the list of Modules in the Related Information section below. The **Module:** Edit screen can also be accessed by clicking on the Module Name.

New: Click the New icon to create a new Module. The Site Module: New screen will open.

7.6 Installing an plug-in/extension

- Download the extension to your local machine as a zip file package.
- From the backend of your Joomla site (administration) select Extensions -> Install/Uninstall.
- Click the Browse button and select the extension package on your local machine.
- Click the Upload File & Install button.
- Some extensions may provide further instructions on installation.
- Note that modules and plugins must be enabled before they will work.

There are some situations in which this procedure will not work.

Sometimes you need to unzip the file locally prior to installing. If you get an error saying that the file is not in the correct format, the need to unzip is a common cause of this. After unzipping try installing the individual items. Note that the files you upload using the installer still need to be zipped.

Information about Some Plug-ins-

Sourcer –

by default we cant execute PHP,HTML & javascript within the articals to execute the scriptwe have to install Sourcerer Plug-in.

Sourcerer enables you to place PHP and any kind of HTML style code (including CSS and JavaScript) right in to your content! Not only in your articles, but also in sections, categories, modules, components, META tags, etc.

You can now just place your original codes right into your WYSIWYG editor. The only thing you have to do is surround the code with the Sourcerer tags.

```
{source}
    //Code
{/source}
```

Akindic plug-in – this plug-in support different types of Indian languages like, hindi, telgu etc.

We can download plug in from internet. Some plug in are free & some plug in are premium. Plug in are available with documentation for how to use.

Web Resources

This appendix lists some of the many resources available on the Web, which can be used to find tutorials, articles, news, and sample PHP code. These are just some of the many out there. Obviously there are far more than we could possibly list in one appendix. And many more that are popping up daily as the usage of and familiarity with PHP and MySQL continues to increase among Web developers. Some of these resources will be in different languages like German or French or something other than your native language. We suggest using a translator like <http://www.Systransoft.com> to browse the Web resource in your native language.

PHP Resources

PHP.Net—<http://www.php.net> - The original site for PHP. Go here to download all the sources of PHP and for a copy of the manual.

ZEND.Com—<http://www zend com>— The source for the ZEND engine that powers PHP 4.0. A portal site that contains forums, and a database of sample classes and code that you can use. A must see.

PHPWizard.net—<http://www.phpwizard.net> - The source of many cool PHP applications like phpMyAdmin: an excellent front end GUI for Managing MySQL Servers. You can also find tutorials on PHP at this site.

PHPBuilder.com—<http://www.phpbuilder.com> - The portal for PUP tutorials. At this site you will find tutorials on just about anything you can think of. Site also has a forum and message board for people to post questions.

DevShed.com—<http://www.devshed.com>—Portal type site offers excellent tutorials on PHP, MySQL, Perl, and other development languages. A must see for new learner.

PX-PHP Code Exchange - <http://px.sklar.com> - A great place to start. Here you will find many sample scripts and useful functions. The site is organized for finding things easily.

The PHP4 Resource—<http://www.php-resource.de>—A very nice source for tutorials, articles and scripts. The only "problem" is that the site is in German. We recommend using a translator service site to view it.

WeberDev.com—<http://www.WeberDev.com>—Formerly known as Berber's PHP sample page. this site grew significantly from nothing to a place for tutorials and sample codes. The site targets PHP and MySQL users, and covers security and general databases, as well as NT. The only issue is that it requires you to subscribe. But it's well worth it, considering the information it provides.

MySql

MySQL and SQL Specific Resources

The MySQL site—<http://mysql.com>—The official MySQL Web site. Provides excellent documentation, support, and information. A must-see if you are using MySQL. Provides sources for the MySQL DB server.

SQL Tutorial— <http://W3.one.net/~jhoffman/sqltut.htm>— Comprehensive SQL tutorial with example and exercises.

The SQL Course—<http://sqlcourse.com>— Provides an introductory SQL tutorial with easy-to-understand instructions. Allows you to practice what you learn on an online SQL interpreter. Advance version is provided at <http://www.sqlcourse2.com>.

DatabaseCentral.com— <http://databascentral.com>- Nice portal with lots of useful information on DB. Provides excellent tutorials, tips, white papers. FAQs. reviews, and so on. A must-see!

The SQL Pro—<http://www.inquiry.com/techtips/thesqlpro> -Have programming questions? Ask the pros! Search database of questions and answers about database development and SQL.

Web Development

Philip and Alex's Guide to Web Publishing—<http://www.arsdigita.com/books/panda/> —A witty, irreverent guide to software engineering as it applies to the Web. One of the few books on the topic co-authored by a Samoyed.

Bibliography

Beginning PHP5 [Book] / auth. Dave Mercer Allan Kent, Steven Nowicki, David Mercer, Dan. - [s.l.] : Wiley Publishing(Wrox).

PHP and MySQL [Book] / auth. Thomson Luke Willing.

PHP for Beginners [Book] / auth. Ivan Bayross, Sharanam Shah, THE X Team. - [s.l.] : SPD.

www.basicphpprogramming.com [Online].

www.tutorialPoints.com [Online].

www.w3school.org [Online]

