



PHP-Handbook with CI

Version, Jan-2016

Contents

Module-1 [Fundamentals]	5
Basic OOPS Concepts	5
OOPS features are	5
Software Engineering.....	8
SDLC (Software Development Life-cycle).....	8
SQL Queries.....	9
Database	10
Normalization.....	10
Web Programming	10
What is Website?	11
Website – basic parts	11
Web browsers	11
Web servers	11
CLIENTS AND SERVERS	11
Module – 2 [Learning the Language]	12
Introduction to PHP	12
How PHP Works	12
Sort Summary of PHP.....	12
What's the difference between PHP and HTML?	12
When to use PHP?	12
What makes PHP a choice among the other scripting languages?	13
PHP Syntax	13
Declaring PHP	13
Example.....	13
Variables in PHP:-	13
Expressions.....	14
PHP is a Loosely Typed Language:-	14
PHP Operators:-	14
Arithmetic Operators.....	14
Assignment Operators.....	15
Comparison Operators.....	15
• If& Else Statement.....	16
while	17
break.....	18
Continue	19
Switch	19
Functions and Arrays using PHP.....	19
PHP Date() functions.....	21
Arrays in PHP	23
PHP String.....	31
PHP String Functions.....	32
PHP Include File.....	37
require().....	37

require_once()	37
include_once()	37
PHP Header	38
PHP \$_GET variable	39
PHP \$_POST variable	39
PHP \$_REQUEST variable	40
Module – 3[Database Connectivity]	40
DBMS & RDBMS	40
MySQL	41
MySQL DataTypes	41
Normalization	42
Connection With Mysql and Database	44
Triggers	52
Indexing In Mysql	53
What is a SQL Injection Attack?	53
Module- 4 [HTML]	54
Introduction to HTML	54
My First Heading	54
HTML Element and Tags	55
HTML Images	55
Module – 5[CSS]	57
Introduction	57
CSS Selector	57
Module – 6[Applicability To Industry Standards]	60
OOPS Concepts	60
Key OOP Concepts	60
Overview of MVC Architecture	69
Sessions and Cookies in PHP	75
Cookies in PHP	76
Files and Directory Access	77
Handling e-mail	80
JAVA Script	82
What is JavaScript?	82
What can a JavaScript do?	83
Form Validation	84
XML	87
AJAX	89
Module – 8[CodeIgniter]	93
What is CodeIgniter?	93
Application Flow Chart	94
Creating Simple Application in CodeIgniter	94
URI Routing(application\config\routes.php)	95
Libraries and Helpers	95
Auto-loading Resources	96

Config Class	96
Database Class	96
Database Configuration	96
Database Functions.....	97
Insert in CodeIgniter	97
Fetch data in CodeIgniter.....	99
URI Class.....	100
\$this->uri->segment(<i>n</i>).....	100
Here	100
➤ \$this->uri->segment(1) = mycontroller(Class Name).....	100
➤ \$this->uri->segment(2) = del(Function name).....	100
➤ \$this->uri->segment(3) =3	100
Delete in CodeIgniter	100
Session Class	101
Initializing a Session.....	101
Adding Custom Session Data.....	101
Retrieving Session Data	101
Removing Session Data	102
Login with Session	102
Form Validation Class.....	104
File Uploading Class	104

Module-1 [Fundamentals]

Basic OOPS Concepts

- Class – group of data members & member functions
- Like person can be class having data members height and weight and member functions as get_details() and put_details() to manipulate on details
- Class is nothing until you create it's object
- Object – instantiates class allocates memory
- Access to data members & member functions can be done using object only (if they are not static!)

Example:

class abc

```
{
    Private:
        Int a;
    Public:
        Void getdata()
        {
            Cout<<"Enter any value";
            Cin>>a;
        }
        Void display()
        {
            Cout<<"Your value is: "<<a;
        }
};
Void main()
{
    abc a1;
    a1.getdata();
    a1.display();
}
```

- In above Example we create one class "abc". In that we create data member "a" and member function getdata() and display(). In getdata() we get value from user and in display we display value of a.
- Now we have to create object "a1" of that class for calling member function. So, we create object in main part and using object we call both function.

OOPS features are

- Encapsulation
- Data hiding
- Data reusability
- Overloading (polymorphism)
- Overriding

Encapsulation

- Encapsulation – making one group of data members & member functions
- Can be done through class
- Then group of data members & Member functions will be available just by creating object.

Example:

class number

```
{
    Private:
        Int n;
    Public:
        Void getdata()
        {
            Cout<<"Enter Any number";
            Cin>>n;
        }
        Void putdata()
        {
            Cout<<"Value is : "<<n;
        }
};
Void main()
{
    number n1;
    n1.getdata();
    n1.putdata();
}
```

Data Hiding

- Data Hiding – can be done through access modifiers
- Access modifiers are private, public, protected and internal
- Private members or member function won't be available outside class
- Public – available all over in program outside class also
- Protected – members that are available in class as well as in its child class

Example:

class visible

```
{
    Private:
        Int n = 5;
    Protected:
        Int a = 10;
    Public:
        Void putdata()
        {
            Cout<<"Value is : "<<n;
            Cout<<"Value is : "<<a;
        }
};
Void main()
{
```

```
visible v1;  
v1.getdata();  
cout<<v1.n;//error  
cout<<v1.a;//error
```

```
}
```

- In above example we create two variable “n” and “a”, private and protected respectively. And create one method public.
- Now we access private and protected variable in same class then it will be print that value but if we access them in main method then it will display error.

Overriding

- Overloading – taking different output of one method or operator based on parameters and return types
- Like add() method performs addition and add(int a,int b) performs addition of ‘a’ and ‘b’ passed when calling

Example:

Class abc

```
{  
    Public:  
        Void add(int a,int b)  
        {  
            Cout<<"Addition is "<<a+b;  
        }  
  
        Void add(int a,int b, int c)  
        {  
            Cout<<"Addtion is: "<<a+b+c;  
        }  
};  
  
Void main()  
{  
    abc obj;  
    obj.add(5,4);  
    obj.add(5,6,7);  
}
```

- In Above Example we create two methods with same name “add” but in that argument list is different.
- So, in main method, at the time of function calling we pass two arguments then respective function is work and if we pass three arguments then its respective function is call.

Data Reusability

- Data Reusability – helps in saving developers time
- You can use already created class to create new one
- Already existing class is base class and new created is derived class
- Base class members can be available in derived class and to access them create object of derived class

Example:

```
class abc  
{
```

```
Public:
    Void display1()
    {
        Cout<<"This is TOPS Technologies"";
    }
};
Class tops:public abc
{
    Public:
        Void display2()
        {
            Cout<<"This is TOPS"";
        }
}
Void main()
{
    tops a1;
    a1.display1();
    a1.display2();
}
```

- In above example we create one class "abc" with one method display1.
- Then we create second class "tops" with method display2().
- Now, If we use display1() method by object of "tops" class then we have to inherit "abc" class in "tops" class.

Software Engineering

The term *software engineering* first appeared in the 1968 NATO Software Engineering Conference and was meant to provoke thought regarding the current "software crisis" at the time.

Definition:-

- "State of the art of developing quality software on time and within budget"
- Trade-off between perfection and physical constraints
- SE has to deal with real-world issues
- State of the art!
- Community decides on "best practice" + life-long education

SDLC (Software Development Life-cycle)

- For project development rules & regulation need to be followed for best quality output at defined time limit.
- It's part of software engineering
- **Six rules to be followed...**
 - Requirement Gathering
 - Analysis & SRS
 - Designing
 - Implementation (Coding)
 - Testing
 - Maintenance
- **Requirement**
 - Phase of collecting requirements from client
 - Will be done by business analyst of company
 - He will create questioner, in which put answers from client

- **Analysis & SRS (Software Requirement Specification)**
 - Collected requirements will be analyzed for time limit, budget and market trade
 - Will be filtered and SRS will be created as result
 - Will be discussed with client
- **Designing**
 - The Design document should reference what you are going to build to meet the requirements, and not how it can include pseudo code but shouldn't contain actual code functionality.
 - Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary.
- **Implementation (Coding)**
 - To launch the coding phase, develop a shell program that is then put under some form of version control.
 - This phase includes the set up of a development environment, and use of an enhanced editor for syntax checking.
- **Testing**
 - Each developer insures that their code runs without warnings or errors and produces the expected results.
 - Types of testing: Defect testing, Path testing, Data set testing, Unit testing, System testing, Integration testing, Black box testing, White box testing, Regression testing, Automation testing, User acceptance testing, Performance testing, etc.
- **Maintenance**
 - User's guides and training are developed to reflect any new functionality and changes which need to be identified to the production staff.
 - Any changes needed to operations and/or maintenance needs to be addressed.

SQL Queries

DDL – data definition Language

- Commands are : create, alter, truncate, drop
- Syntax :
`Create table table_name(col_name datatype(size)...);`
- Example :
`Create table Person_Master (name nvarchar(50));`

DML – data manipulation language

- Like insert, update, delete
- Syntax:
`insert into table_name(col1,col2,..coln) values(val1,val2,..valn);`
- Example:
`insert into Person_Master(name) values('name1');`

Update Syntax:

- `update table_name set col1=val1, col2=val2 where col = val;`
- Ex :
`update Person_Master set name = 'name1' where ID=1;`
- It will set name to 'name1' for which ID is 1

Delete syntax:

`delete from table_name where condition;`

Example :

delete from Person_Master where ID=1;

- It will delete whole row for which ID is 1
- It is conditional delete operation
- To delete all rows simply omit the condition

Select syntax :

select * from table_name;

- It will select all rows from specified table
- To select particular row
select * from table_name where condition;
- To select particular column
select col1,col2 from table_name;

Database

- RDBMS stores data in form of tables and relationship between those tables in form of tables

Queries – sentences executed on database for data manipulation

- Will be handled by database engines and will perform action on data that are stored on database server
- Ex are create, alter, insert, update, delete etc
- SQL – structured query language is used for this

Constraints – terms that needs to be satisfied on data

- For ex all students must have unique roll number
- Can be defined as primary key, foreign key, unique key etc.
- Primary key – column of table whose value can be used to uniquely identify records
- Foreign key – column inside table that is primary key of another table
- Unique key – like primary key can be used to uniquely identify a record
- Difference between primary key and unique key is primary key will never allow null where as unique key will allow it for once

Normalization

- The process of structuring data to minimize duplication and inconsistencies.
- The process usually involves breaking down the single table into two or more tables and defining relationships between those tables.

1NF

- Eliminate duplicative columns from the same table.
- Create separate tables for each group related data and identify each row with unique columns.
- In this, one table will be divided in two tables with relevant contents

2NF

- Once 1NF has been done then and only then 2NF can be done
- In this, provide key constraints to the columns of tables based on uniqueness
- Like assign primary key to one table column and refer this as foreign key in another table

3NF

- Only after 2NF, third normalization can be done
- In this, further more analyze tables and divide them for data uniqueness.

Web Programming

Scripting language, script language or extension language is a programming language that allows control of one or more software applications. "Scripts" are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-

user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code. Scripting languages are nearly always embedded in the applications they control.

What is Website?

- Website – Everyday you visit on internet
- Follows some rules & regulations i.e. client-server architecture standard
- Websites – providing information from anywhere in world

Website – basic parts

- Websites are consist of three parts
- GUI – web pages that you visit
- Coding – logic that provides functionality or makes website dynamic
- Database – manages data provided by end user
- For GUI building HTML is used from long time

Web browsers

Main article: Client-side scripting

Web browsers are applications for displaying web pages. A host of special-purpose languages has developed to control their operation. These include JavaScript, a scripting language superficially resembling Java; VBScript by Microsoft, which only works in Internet Explorer; XUL by the Mozilla project, which only works in Firefox; and XSLT, a presentation language that transforms XML content into a new form.

Web servers

Main article: Server-side scripting

On the server side of the HTTP link, application servers and other dynamic content servers such as Web content management systems provide content through a large variety of techniques and technologies typified by the scripting approach.

CLIENTS AND SERVERS

- Web programming languages are usually classified as server-side or client-side. Some languages, such as JavaScript, can be used as both client-side and server-side languages, but most Web programming languages are server-side languages.
- The client is the Web browser, so client-side scripting uses a Web browser to run scripts. The same script may produce different effects when different browsers run it.
- A Web server is a combination of software and hardware that outputs Web pages after receiving a request from a client. Server-side scripting takes place on the server. If you look at the source of a page created from a server-side script, you'll see only the HTML code the script has generated. The source code for the script is on the server and doesn't need to be downloaded with the page that's sent back to the client.
- **Example of Web Programming Language**
 - PHP
 - ASP
 - Perl
 - JAVA

Module – 2 [Learning the Language]

Introduction to PHP

PHP stands for "PHP: HyperText Preprocessor". PHP is a server side scripting language for making logic driven websites. Ever wonder how they made that "contact us" form on their site, which sends out emails? Well, they used PHP. Or, how they made that image upload tool? Well, they used PHP. PHP written scripts can use databases to keep track of your customer's and visitors activities on your site, send out periodical newsletters to your subscribers, upload files or images and drive the content on your site dynamically. The possibilities are endless. Most of the social networking websites you visit are writing in PHP. Yep! PHP is that powerful. Learning The Basics of PHP will help you tremendously in your Webpage development.

PHP sits between your browser and the web server. When you type in the URL of a PHP website in your browser, your browser sends out a request to the web server. The web server then calls the PHP script on that page. The PHP module executes the script, which then sends out the result in the form of HTML back to your browser, which you see on the screen. Here is a basic php diagram which illustrates the process.

How PHP Works

The PHP software works with the web server, which is the software that delivers web pages to the world. When you type a URL into your web browser's address bar, you're sending a message to the web server at that URL, asking it to send you an HTML file. The web server responds by sending the requested file. Your browser reads the HTML file and displays the web page. You also request a file from the web server when you click a link in a web page. In addition, the web server processes a file when you click a web page button that submits a form. This process is essentially the same when PHP is installed. You request a file, the web server happens to be running PHP, and it sends HTML back to the browser, thanks to the programming in PHP.

Sort Summary of PHP

- PHP stands for Hypertext Preprocessor.
- PHP scripts run inside Apache server or Microsoft IIS.
- PHP and Apache server are free.
- PHP code is very easy.
- PHP is the most used server side scripting language.
- PHP files contain PHP scripts and HTML.
- PHP files have the extension "php", "php3", "php4", or "phtml".

What's the difference between PHP and HTML?

- PHP is a scripting language while HTML is a markup language.
- The output of PHP is usually in HTML code which the browser can then interpret
- HTML codes are static and they are always the same every time they are opened while PHP files are dynamic and the output might not always be the same
- HTML is very easy and forgiving of mistakes while PHP isn't
- HTML files are requested by browser, and returned by server.
- PHP files are requested by browser, and executed by the server to output a plain HTML that is returned to the browser.

When to use PHP?

- Creating Web pages that contain dynamic contents.

- responding to HTML forms.
- Accessing databases.
- Securing data.
- You're responsible for a Website
- Others in your organization like using PHP
- You want something "lighter" and easier to learn than Perl or Python
- You favor free software
- You need access to data locked in databases
- You want to exploit Java
- You already use PHP in one way, and have been wondering what else it can do for you

What makes PHP a choice among the other scripting languages?

- PHP is easy to learn.
- PHP is free.
- PHP can run on Windows and Unix servers.
- PHP is very fast.

PHP Syntax

Declaring PHP

PHP scripts are always enclosed in between two PHP tags like `<?php ?>`. This tells your server to parse the information between them as PHP. The three different forms are as follows:

Example

Code:-

```
<?php
    echo "Welcome to PHP World";
    // echo statemnet is print welcome
to php world
?>
```

Output:-

Welcome to PHP World

Variables in PHP:-

A variable is a holder for a type of data. So, based on its type, a variable can hold numbers, strings, Booleans, objects, resources or it can be NULL. In PHP all the variables begin with a dollar sign "\$" and the value can be assigns using the "=" operator. The dollar sign is not technically part of the variable name, but it is required as the first character for the PHP parser to recognize the variable as such. Another important thing in PHP is that all the statements must end with a semicolon ";". In PHP we needn't have to specify the variable type, as it takes the data type of the assigned value. The contents of a variable can be changed at any time, and so can its type. To declare a variable, you must include it in your script. You can declare a variable and assign it a value in the same statement.

Code:-

```
<?php
$txt="HelloWorld!";
// assign a stirng value in "txt" valivable
$x=16;
// assign a INT value in "x" valivable
echo $txt; //print variable value
?>
```

Output:-

HelloWorld!

Rules of Variable

- Names (also called identifiers)
- Must always begin with a dollar-sign (\$)
- generally start with a letter and can contain letters, numbers, and underscore characters “_”
- Names are case sensitive
- Values can be numbers, strings, Boolean, etc
- change as the program executes

Expressions

Expressions are the most important building stones of PHP. In PHP, almost anything you write is an expression. Expression means we have to define any variable or assign value to the variable or do some mathematical operations.

Example

Code:-

```
<?php
    $a = 5; // Define variable and assign
value
    $a++; // Increment 1 in variable
    echo $a; // print value of a
?>
```

Output:-

6

Here, this is simple example of expression. In this we have define variable “a” and then we increase its value by using “++”.

PHP is a Loosely Typed Language:-

- 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.

PHP Operators:-

Operator is something that you feed with one or more values (or expressions, in programming jargon) which yields another value (so that the construction itself becomes an expression).

There are five types of operators.

Arithmetic Operator

- Comparison Operator
- Logical Operator
- Increment and Decrement Operator
- Concatenation Operator

Arithmetic Operators

- Arithmetic operator is used for doing arithmetic operation between two or more variables.
- Arithmetic operator is +, -, /, *, %.

the value of the assignment expression on the right.

```
<?php
$a = ($b = 4) + 5; // $a is equal to 9 now, and $b has been set to 4.
?>
```

Example:

Code:-

```
<?php
$a = 5;
$b = 10;
$add = $a + $b; // Addition
$sub = $a - $b; // Subtraction
$mul = $a * $b; // Multiplication
$div = $a / $b; // Division
echo "Addtion is :" . $add . "<br>";
echo "Subtraction is :" . $sub . "<br>";
echo "Multiplication is :" . $mul . "<br>";
echo "Division is :" . $add . "<br>";
?>
```

Output:-

```
Addtion is :15
Subtraction is :-5
Multiplication is :50
Division is :15
```

Comparison Operators

Comparison operators, as their name implies, allow you to compare two values. You may also be interested in viewing the type comparison tables, as they show examples of various type related comparisons.

Operator	Name	Example
==	Equal	<code>\$x == \$y</code>
===	Identical	<code>\$x === \$y</code>
!=	Not Equal	<code>\$x != \$y</code>
<>	Not Equal	<code>\$x <> \$y</code>
!==	Not Identical	<code>\$x !== \$y</code>
>	Graeter Than	<code>\$x > \$y</code>
<	Less Than	<code>\$x > \$y</code>
>=	Greater Than or Equal	<code>\$x >= \$y</code>
<=	Less Than or Equal	<code>\$x <= \$y</code>

Example:-

Code:-

```
<?php
$a = 5;
$b = 10;
var_dump($a == $b); //Return False
echo "<br>";
var_dump($a != $b); //Return True
echo "<br>";
var_dump($a > $b); //Return False
echo "<br>";
var_dump($a <= $b); //Return True ?>
```

Output:-

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

Conditional Test, Events and Flows in PHP

PHP script is built out of a series of statements. A statement can be an assignment, a function call, a loop, a conditional statement or even a statement that does nothing (an empty statement). Statements usually end with a semicolon. In addition, statements can be grouped into a statement-group by encapsulating a group of statements with curly braces. A statement-group is a statement by itself as well.

The various statement types are described as bellow:-

- **If& Else Statement**

The if construct is one of the most important features of many languages, PHP included. It allows for conditional execution of code fragments. PHP features and if structure that is similar to that of C:

Syntax:

```
if (expression){
    Statement
}else{
    Statement
}
```

Example:

Code:-

```
<?php
$a = 5;$b = 10;
if($a > $b) { // Check condition
    echo "a is big";
}
else{
    echo "b is big": }?>
```

Output:-

b is big

elseif/else if

If...elseif...else statement it will first see if the If statement is true. If that tests comes out false it will then check the first elseif statement. If that is false it will either check the next elseif statement, or if there are no more elseif statements, it will evaluate the else segment,

Syntax:

```
if (expression)
    Statement
else if(expression)
    Statement
else if(expression)
    Statement
else
    statement
```

Example

Code:- <?php
\$a = 5;
\$b = 10;
if(\$a > \$b)

```
{  
    echo "a is big";  
}  
else if($a < $b){  
    echo "b is big"; }  
else{  
    echo "Both are equal"; } ?>
```

Output:-

b is big

In this example, First condition is false than it will check second condition. If the second condition is false than it will check else part. But here second condition is true than it will execute it's statement.

while

- While loop is used for executing some statement multiple times.
- In while loop First check condition than execute it's statement until the condition become false.
- That's way while loop is known as Entry Control Loop.

Syntax:

```
while (expression)  
{  
    Statement  
}
```

Example:

Code:-

```
<?php  
$a = 1;  
while($a <= 5)  
{echo "value of a is: ". $a . "<br>";  
  $a++; }  
?>
```

Output:-

value of a is: 1
value of a is: 2
value of a is: 3
value of a is: 4
value of a is: 5

In This Example we have print 1 to 10 using while Loop. When condition become false than it will stop looping.

do-while

- Do-While loop is used for executing some statement multiple times.
- In do-while loop First execute it's statement then check condition.
- That's way do-while loop is known as Exit Control Loop.

Syntax:

```
do{  
    Statement  
} while (expression);
```

Code:-

```
<?php
$a = 10;
do{
    echo "value of a is: ". $a . "<br>";
    $a++;
}while($a <= 5);?>
```

Output:-

value of a is: 10

In this Example, the value of variable is 10. Then it will enter in loop and print value of a is 10. Then value of "a" is increment by 1 and check condition. It is false and control goes out of loop

. for

for loops are the most complex loops in PHP. In *for* loop there is three partitions. First initialization, second condition checking and third is increment or decrement.

Syntax:

for (Initialization; Condition; Increment/decrement){statement}

Example:

Code:-

```
<?php
for($i=1;$i<=5;$i++)
    echo "Value of i is: " . $i.
    "<br>";
?>
```

Outp Value of i is: 1
Value of i is: 2
Value of i is: 3
Value of i is: 4
Value of i is: 5

foreach

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

Syntax:

foreach (array expression as \$value)
Statement

Example:

Code:-

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

Output:

red
green
blue
yellow

In above example, We define one array and create four elements in that. After that we write *foreach* loop and assign "value" variable to "colors" variable. And then print value of color.

break

- *break* ends execution of the current *for*, *foreach*, *while*, *do-while* or *switch* structure. *break* accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.

Code:-

```
<?php
for($i=1;$i<=5;$i++)
{ if($i==3)
    {break;
    } echo "Value of i is: " . $i. "<br>";
}
?>
```

Output:

```
Value of i is: 1
Value of i is: 2
```

In above example, we put the condition when the value of "i" become 3 than it execute break statement.

Continue

Continue is used within looping structures to skip the rest of the current loop iteration and continue execution at the

Code:-

```
<?php
for($i=1;$i<=5;$i++) {
    if($i==3){
        continue; }
    echo "Value of i is: " . $i."<br>";
}??>
```

Output:

```
Value of i is: 1
Value of i is: 2
Value of i is: 4
Value of i is: 5
```

In above example, We put condition when value of "i" become 3. So, in output value of 3 is skip because when the condition is true than "continue" skip the current statement and start further execution.

Switch

The *switch* statement is similar to a series of IF statements on the same expression. In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to

```
<?php
switch ($i) {
    case 0:
        echo "i equals 0";
        break;
    case 1:
        echo "i equals 1";
        break;
    case 2:
        echo "i equals 2";
        break;
}
?>
```

Functions and Arrays using PHP

Function

- A function is a block of code that can be executed whenever we need it.
- All functions start with the word "function()"

- Name the function -It should be possible to understand what the function does by its name. The name can start with a letter or underscore (not a number)
- Add a "{"-The function code starts after the opening curly brace
- Insert the function code
- Add a "}"-The function is finished by a closing curly brace

Code:-

```
<?php
Function test();// test is a function name
{
    echo "TOPS Technology";
}
test();// here call function
?>
```

Output:

TOPS Technology

Here, We create function test. And then we call by using only function name.

PHP Functions -Adding parameters

Function adding parameter means we have pass some argument in function.

Syntax:

```
function function_name(argument list)
{
    Statement:
}
```

Example:

Code:-

```
<?php
function test($fname)//here pass
$fname parameters in test function{
    Echo $fname. "Technology.<br/>";
}
echo "My company is " .
test("TOPS");//call test function with
parameters
?>
```

Output:

My company is TOPS Technology

PHP Functions -Return values

- Functions can also be used to return values.

Example:

Code:-

```
<?php
function add($a,$b)
{
    $add = $b + $b;
    return $add; // here return value
}echo "Addition is : " . add(5,4);
?>
```

Output:

Addition is : 9

This Example, We have to create one simple function “add”. And then we make addition of two variable and return to the function. And after that we have to call function and pass value in the function.

Passing function parameters by reference

- Pass by reference means we have pass reference variable at the time of function calling not value.
- References allow two variables to refer to the same content.
- In other words, a variable points to its content (rather than becoming that content). Passing by reference allows two variables to point to the same content under different names.
- The ampersand (&) is placed before the variable to be referenced.

Code:-

```
<?php
function display(&$a)
{
    $a=$a. 'Good';
}
$b = 'Hello.., ';
display($b);
echo $b;
?>
```

Output:

Hello.., Good

In this example we create one function display and pass reference variable with ampersand (&). Then we create function body. Now we have to create another variable and pass in the function calling.

Use default parameters in Function

- Default parameter means we have create parameter list with default value.
- And At the time of function call, if we do not pass any value in the function then it will take default value otherwise it take value which we have to pass in function.
- And if we pass NULL in the it will return blank value.

Example:

```
<?php
function test($a="Hello")
{echo $a;}
test(); // This function take default value
test("TOPS"); // This function take TOPS
test(NULL); // This function return NULL value
?>
```

PHP Date() functions

- Php date() function is used display current date with different format.
- The PHP date() function formats a timestamp to a more readable date and time.
- A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

Syntax:

date(format,timestamp)

Example:

Code:-

```
<?php
    $d = date('d/m/y');
    echo $d;
?>
```

Output:

23/11/13

Timestamp:

- Timestamp is the number of second from 1st January 1970. It is also known as Unix timestamp.

Example

Code:-

```
<?php
    $t = time();
    echo $t;
?>
```

Output:

1385188379

Above Example Display total number of seconds from 1st January 1970. If you refresh your browser than second will be increase.

Important Functions

- time() Returns the Current Time as a Unix Timestamp
- date() Formats a Local Time/Date
- strtotime() Parses an English Textual Date or Time Into a Unix Timestamp

mktime() Function

- Now if you want to find out future date than you have to use mktime() function

Syntax:

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

Example:

Code:-

```
<?php
    $t = mktime(0,0,0,date('m'),date('d')+5,date('y'));
    $n = date('d/m/y',$t);
    echo $n;
?>
```

Output:

28/11/13

In this example we have to find out date of after 5 days from current date.

- So, We have to pass hour,minute,second value is 0 because we have to find future date from current time and then we have to pass month, day and year.
- And we add "+5" in day because we find date after 5 days from current date.

PHP Date - Reference

- Now that you know the basics of using PHP's *date* function, you can easily plug in any of the following letters to format your timestamp to meet your needs.

Important Full Date and Time:

- **r**: Displays the full date, time and timezone offset. It is equivalent to manually entering date("D, d M Y H:i:s O")

Time:

- **a**: am or pm depending on the time
- **A**: AM or PM depending on the time
- **g**: Hour without leading zeroes. Values are 1 through 12.
- **G**: Hour in 24-hour format without leading zeroes. Values are 0 through 23.
- **h**: Hour with leading zeroes. Values 01 through 12.
- **H**: Hour in 24-hour format with leading zeroes. Values 00 through 23.
- **i**: Minute with leading zeroes. Values 00 through 59.
- **s**: Seconds with leading zeroes. Values 00 through 59.

Day:

- **d**: Day of the month with leading zeroes. Values are 01 through 31.
- **j**: Day of the month without leading zeroes. Values 1 through 31
- **D**: Day of the week abbreviations. Sun through Sat
- **l**: Day of the week. Values Sunday through Saturday
- **w**: Day of the week without leading zeroes. Values 0 through 6.
- **z**: Day of the year without leading zeroes. Values 0 through 365.

Month:

- **m**: Month number with leading zeroes. Values 01 through 12
- **n**: Month number without leading zeroes. Values 1 through 12
- **M**: Abbreviation for the month. Values Jan through Dec
- **F**: Normal month representation. Values January through December.
- **t**: The number of days in the month. Values 28 through 31.

Year:

- **L**: 1 if it's a leap year and 0 if it isn't.
- **Y**: A four digit year format
- **y**: A two digit year format. Values 00 through 99.

Other Formatting:

- **U**: The number of seconds since the Unix Epoch (January 1, 1970)
- **O**: This represents the Timezone offset, which is the difference from Greenwich Meridian Time (GMT).
100 = 1 hour, -600 = -6 hours
- We suggest that you take a few minutes to create several timestamps using PHP's *mktime* function and just try out all these different letters to get your feet wet with PHP's *date* function.

Arrays in PHP

An array is assigned to a single variable, but it can hold dozens of individual pieces of information. Each individual bit of information, or row, is referred to as an **array element**. Within every array element there are two parts: the **value**, which contains the actual information you want to store, and a unique **key** which identifies the value. You'll learn later how keys play an important role in the process of accessing and manipulate array elements.

Keys can either be non-negative integers, or strings. Arrays with integers as keys are the most common types and known as Scalar Arrays. Those with strings as keys are called Associative Arrays.

Indexed Versus Associative Arrays

There are two kinds of arrays in PHP: indexed and associative. The keys of an *indexed* array are integers, beginning at 0. Indexed arrays are used when you identify things by their position. *Associative* arrays have strings as keys and behave more like two-column tables. The first column is the key, which is used to access the value.

Types of Array

There are three types of array in php

- 1) Numeric Array
- 2) Associative Array
- 3) Multidimensional Array

Numeric Array

- In numeric array have to create only elements, id will be automatically assign to the element.

Example:

Code:-

```
<?php
    $a = array("a","b","c");
    print_r($a);
?>
```

Output:

```
Array ( [0] => a [1] => b [2] => c )
```

- In above Example, create only elements in array. When we run than it will be display with it's id. And for array we have to use print_r() to display value of array.

Associative Array

- In associative array we have create element with id.

Example:

Code:-

```
<?php
    $a = array("1"=>"a","2"=>"b","3"=>"c");
    print_r($a);
?>
```

Output:

```
Array ( [1] => a [2] => b [3] => c )
```

- In above Example, create elements with its id in array.

Multidimensional Array

- Multidimensional array means we have to create array within array.

Example:

Code:-

```
<?php
    $a = array("a","b"=>array("p","q"),"c");
    print_r($a);
?>
```

Output:

```
Array ( [0] => a [b] => Array ( [0] => p
[1] => q ) [1] => c )
```

other Array Function:

array_combine();

Use: This Function is Creates an array by using the elements from one "keys" array and One "values" array

Example:

Code:-

```
<?php
$animal1=array("a"=>"Tiger","b"=>"Lion");
$fruit=array("Apple","mango");
print_r(array_combine($animal1,$fruit));
?>
```

Output:

```
Array ( [Tiger] => Apple [Lion] => mango )
```

array_count_values()

Use: This Function is Counts all the values of an array

Example:

Code:-

```
<?php
$no=array(0=>"5",1=>"10",2=>15,3=>20);
print_r(array_count_values($no));
?>
```

Output:

```
Array ( [5] => 1 [10] => 1 [15] => 1 [20] => 1 )
```

array_diff()

Use: This Function is Compare arrays, and returns the differences (compare values only)

Example:

Code:-

```
<?php
$animal1=array("a"=>"Tiger","b"=>"Lion");
$animal2=array("c"=>"Dog","b"=>"Lion");
print_r(array_diff($animal1,$animal2)); ?>
```

Output:

```
Array ( [a] => Tiger )
```

array_merge()

Use: This Function Merges one or more arrays into one array

Example:

Code:-

```
<?php
$animal1=array("a"=>"Tiger","b"=>"Lion");
$animal2=array("c"=>"Dog","b"=>"Lion");
print_r(array_merge($animal1,$animal2));
?>
```

Output:

```
Array ( [a] => Tiger [b] => Lion [c] => Dog )
```

array_merge_recursive()

Use: This Function Merges one or more arrays into one array but in recursive

Example:

Code:-

```
<?php
$animal1=array("a"=>"Tiger","b"=>"Lion");
$animal3=array("c"=>"Dog","b"=>"Cat");
print_r(array_merge_recursive($animal1,$animal3));
?>
```

Output:

```
Array ( [a] => Tiger [b] => Array ( [0] => Lion [1] => Cat ) [c] => Dog )
```

array_reverse()

Use: This Function Returns an array in the reverse order

Example:

Code:-

```
<?php
$animal1=array("a"=>"Tiger","b"=>"Lion");
print_r(array_reverse($animal1));
?>
```

Output:

```
Array ( [b] => Lion [a] => Tiger )
```

array_search()

Use: This Function Returns Searches an array for a given value and returns the key

Example:

Code:-

```
<?php
$animal1=array("a"=>"Tiger","b"=>"Lion");
print_r(array_search("Tiger",$animal1));
?>
```

Output:

```
a
```

array_sum()

Use: This Function Returns the sum of the values in an array

Example:

Code:-

```
<?php
$no=array(0=>"5",1=>"10",2=>15,3=>20);
print_r(array_sum($no));
?>
```

Output:

```
50
```

Example :

Code:-

```
<?php
$fruits =
array("d"=>"lemon","a"=>"orange","b"=>"
banana","c"=>"apple");
arsort($fruits);
foreach ($fruits as $key => $val)
{
    echo "$key = $val". "<br>";
}
?>
```

Output:

```
a = orange
d = lemon
b = banana
c = apple
```

arsort()

Use: This Function Sorts an associative array in ascending order, according to the value

Example :

Code:-

```
<?php
$fruits =
array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"a
pple");
arsort($fruits);
foreach ($fruits as $key => $val)
{
    echo "$key = $val". "<br>";
}
?>
```

Output:

```
c = apple
b = banana
d = lemon
a = orange
```

krsort()

Use: This Function Sorts an associative array in descending order, according to the key

Example:

Code:-

```
<?php
$fruits =
array("d"=>"lemon","a"=>"orange","b"=>"banana","c"=>"apple");
arsort($fruits);
foreach ($fruits as $key => $val) {
    echo "$key = $val". "<br>";} ?>
```

Output:

```
d = lemon
c = apple
b = banana
a = orange
```

ksort()

Use: This Function Sorts an associative array in ascending order, according to the key

Example:

Code:-

```
<?php
$fruits =
array("d"=>"lemon","a"=>"orange","b"=>"banana","c"
=>"apple");
ksort($fruits);
foreach ($fruits as $key => $val)
{
    echo "$key = $val". "<br>";
}
?>
```

Output:

```
a = orange
b = banana
c = apple
d = lemon
```

sizeof()

Use: This Function Find out the number of element in array

Example:

Code:-

```
<?php
$fruit=array("Apple","mango");
print_r(sizeof($fruit));
?>
```

Output:

```
2
```

array_shift()

Use: Removes the first element from an array, and returns the value of the removed element

Example:**Code:-**

```
<?php
$animal1=array("a"=>"Tiger","b"=>"Lion");
echo array_shift($animal1);
echo "<br>";
print_r ($animal1);
?>
```

Output:

```
Tiger
Array ( [b] => Lion )
```

array_slice()

Use: Returns selected parts of an array

Example:**Code:-**

```
<?php
$no=array(0=>"5",1=>"10",2=>15,3=>20);
print_r(array_slice($no,1,2));
?>
```

Output:

```
Array ( [0] => 10 [1] => 15 )
```

array_unique()

Use: This function Removes duplicate values from an array

Example:**Code:-**

```
<?php
$no=array(0=>"5",1=>"10",2=>15,3=>20);
print_r(array_unique($no));
?>
```

Output:

```
Array ( [0] => 5 [1] => 10 [2] => 15 [3] => 20 )
```

array_unshift()

Use: This function Adds one or more elements to the beginning of an array

Example:

Code:-

```
<?php
$animal1=array("a"=>"Tiger","b"=>"Lion");
array_unshift($animal1,"Horse");
print_r($animal1);
?>
```

Output:

```
Array ( [0] => 5 [1] => 10 [2] => 15 [3] => 20 )
```

array_values()

Use: This function Returns all the values of an array

Example:

Code:-

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

Output:

```
Array ( [0] => XL [1] => gold )
```

array_walk_recursive()

Use: This function Applies a user function recursively to every member of an array

Example:

Code:-

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

Output:

```
Array ( [0] => XL [1] => gold )
```

list ()

Use: This function Assigns variables as if they were an array

Example:

Code:-

```
<?php
$info = array('coffee','brown','caffeine');
list($a[0],$a[1],$a[2])=$info;
var_dump($a);
?>
```

Output:

```
array(3) { [2]=> string(8)
"caffeine" [1]=> string(5)
"brown" [0]=> string(6)
"coffee" }
```

array_intersect()

Use: This function Compare arrays, and returns the matches (compare values only)

Example:

Code:-

```
<?php
$array1=array("a"=>"green","red","blue");
$array2=array("b"=>"green","yellow","red");
$result=array_intersect($array1,$array2);
print_r($result); ?>
```

Output:

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

array_key_exists()

Use: This function Checks if the specified key exists in the array

Example:

Code:-

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

Output:

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

array_keys()

Use: Returns all the keys of an array

Example:

Code:-

```
<?php
$array = array(0 => 100, "color" => "red");
print_r(array_keys($array));
?>
```

Output:

```
Array ( [0] => 0 [1] => color )
```

array_multisort()

Use: This function Sorts multiple or multi-dimensional arrays

Example:

Code:-

```
<?php
$ar1 = array(10, 100, 100, 0);
$ar2 = array(1, 3, 2, 4);
array_multisort($ar1, $ar2);
var_dump($ar1);
var_dump($ar2);
?>
```

Output:

```
array(4) { [0]=> int(0) [1]=>
int(10) [2]=> int(100) [3]=>
int(100) } array(4) { [0]=> int(4)
[1]=> int(1) [2]=> int(2) [3]=>
int(3) }
```

array_pad()

Use: This function Inserts a specified number of items, with a specified value, to an array

Example:

```
<?php
$input=array(12,10,9);
$result=array_pad($input,5,0);
print_r($result);
?>
```

```
Array ( [0] => 12 [1] => 10 [2] => 9 [3]
=> 0 [4] => 0 )
```

array_pop ()

Use: This function Deletes the last element of an array

Example:

Code:-

```
<?php
$stack=array("orange","banana","apple","raspberry");
$fruit=array_pop($stack);
print_r($stack);
?>
```

Output:

```
Array ( [0] => orange [1] =>
banana [2] => apple )
```

array_push()

Use: This function Inserts one or more elements to the end of an array

Example:

Code:-

```
<?php
$stack=array("orange","banana");
array_push($stack,"apple","raspberry");
print_r($stack);
?>
```

Output:

```
Array ( [0] => orange [1] => banana [2]
=> apple [3] => raspberry )
```

array_rand()

Use: This function Returns one or more random keys from an array

Example:

Code:-

```
<?php
$input=array("a","b","c","d","e");
$rand_keys = array_rand($input, 2);
echo $input[$rand_keys[0]] . "\n";
echo $input[$rand_keys[1]] . "\n";
?>
```

Output:

```
a c
```

PHP String

- A string variable is used to store and manipulate a piece of text.
- String variables are used for values that contain character strings.
- A string can be used directly in a function or it can be stored in a variable.

Example:

Code:-

```
<?php
$s = "Good Morning";
echo $s;
?>
```

Output:

```
Good Morning
```

PHP String Functions

addslashes()

Use: This function Returns a string with backslashes in front of predefined characters

Example:

Code:-

```
<?php
$a="pratik's";
echo addslashes($a)." hy....";
?>
```

Output:

```
pratik\'s hy....
```

explode()

Use: This function Breaks a string into an array

Example:

Code:-

```
<?php
$name="Pratik Sheth";
print_r(explode(" ", $name));
?>
```

Output:

```
Array ( [0] => Pratik [1] => Sheth )
```

implode()

Use: This function Returns a string from the elements of an array

Example:

Code:-

```
<?php
$im=array("Pratik","Sheth");
echo (implode(" ", $im));
?>
```

Output:

```
Pratik Sheth
```

join()

Use: This function is Alias of implode()

Example:

Code:-

```
<?php
$im=array("Pratik","Sheth");
echo (join(" ", $im));
?>
```

Output:

```
Pratik Sheth
```


md5()

Use: This function Calculates the MD5 hash of a string

Example:

Code:-

```
<?php
$md="Pratik";
echo md5($md);
?>
```

Output:

```
7bccf0bdc2a136dd7bbeaf1d93822d93
```

nl2br()

Use: This function Inserts HTML line breaks in front of each newline in a string

Example:

Code:-

```
<?php
$nl="Pratik<br>Sheth";
echo nl2br($nl);
?>
```

Output:

```
Pratik
Sheth
```

str_split()

Use: This function Splits a string into an array

Example:

Code:-

```
<?php
$name="Pratik Sheth";
print_r(str_split(" ", $name));
?>
```

Output:

```
Array ( [0] => Pratik [1] => Sheth )
```

strcmp()

Use: This function Compares two strings (case-sensitive)

Example:

Code:-

```
<?php
echo strcmp("Pratik", "Pratik");
?>
```

Output:

```
0
```

Use: This function Converts a string to lowercase letters

Example:

Code:-

```
<?php
echo strtolower("PRATIK");
?>
```

Output:

```
pratik
```

strtoupper()

Use: This function Converts a string to uppercase letters

Example:

Code:-

```
<?php
echo strtoupper("pratik");
?>
```

Output:

PRATIK

trim()

Use: This function Removes whitespace or other characters from both sides of a string

Example:

Code:-

```
<?php
$t=" pratik sheth ";
echo trim($t);
?>
```

Output:

pratik sheth

number_format()

Use: This function Formats a number with grouped thousands

Example:

Code:-

```
<?php
echo number_format("10000",2);
?>
```

Output:

10,000.00

rtrim()

Use: This function Removes whitespace or other characters from the right side of a string

Example:

Code:-

```
<?php
$r="pratik sheth ";
echo rtrim($r);
?>
```

Output:

pratik sheth

str_ireplace()

Use: This function Replaces some characters in a string (case-insensitive)

Example:

Code:-

```
<?php
echo str_ireplace("Pratik","Petrix","Pratik sheth"); ?>
```

Output:

Petrix sheth

str_repeat()

Use: This function Repeats a string a specified number of times

Example:

Code:-

```
<?php
echo str_repeat("Pratik",3);
?>
```

Output:

```
PratikPratikPratik
```

str_replace()

Use: This function Replaces some characters in a string (case-sensitive)

Example:

Code:-

```
<?php
$r = array("Hello","Hy","Pratik","Hello");
print_r(str_replace("Hello","How",$r,$i));
echo "Replacements: $i";
?>
```

Output:

```
Array ( [0] => How [1] => Hy [2] => Pratik
[3] => How ) Replacements: 2
```

str_shuffle()

Use: This function Randomly shuffles all characters in a string

Example:

Code:-

```
<?php
echo str_shuffle("Pratik");
?>
```

Output:

```
iaPrkt
```

str_word_count

Use: This function Count the number of words in a string

Example:

Code:-

```
<?php
echo str_word_count("Pratik sheth");
?>
```

Output:

```
2
```

strcasecmp()

Use: This function Compares two strings (case-insensitive)

Example:

Code:-

```
<?php
echo strcasecmp("pratik","Pratik");
?>
```

Output:

```
0
```

strchr()

Use: This function Finds the first occurrence of a string inside another string (alias of strstr())

Example:

Code:-

```
<?php
echo strchr("Pratik sheth","sheth");
?>
```

Output:

```
sheth
```

strstr()

Use: This function Finds the first occurrence of a string inside another string (case-sensitive)

Example:

Code:-

```
<?php
echo strstr("Pratik sheth","sheth");
?>
```

Output:

```
sheth
```

stristr()

Use: This function Finds the first occurrence of a string inside another string (case-insensitive)

Example:

Code:-

```
<?php
echo stristr("Pratik sheth","Sheth");
?>
```

Output:

```
sheth
```

strcspn()

Use: This function Returns the number of characters found in a string before any part of some specified characters are found

Example:

Code:-

```
<?php
echo strcspn("pratik","i");
?>
```

Output:

```
4
```

stripos()

Use: This function Returns the position of the first occurrence of a string inside another string (case-insensitive)

Example:

Code:-

```
<?php
echo stripos("Pratik","a");
?>
```

Output:

```
2
```

strspn()

Use: This function Returns the number of characters found in a string that contains only characters from a specified charlist

Example:

```
<?php
echo strspn("pratik","zik");
?>
```

Output:

```
0
```

PHP Include File

Server Side Includes (SSI) is used to create functions, headers, footers, or elements that will be reused on multiple pages.

include()

The **include()** statement includes and evaluates the specified file.

require()

require() works same as **include()** but the difference is that if we use **include()** and file does not exist then it will give (**E_WARNING**) which allows the script to continue and if we use **require()** and file does not exist then it will give fatal **E_ERROR** level error and stop the execution of script.

require_once()

The **require_once()** statement is identical to **require()** except PHP will check if the file has already been included, and if so, not include (require) it again.

include_once()

The **include_once()** statement includes and evaluates the specified file during the execution of the script. This is a behavior similar to the **include()** statement, with the only difference being that if the code from a file has already been included, it will not be included again. As the name suggests, it will be included just once.

Example of include:

Code:-

```
(file1.php)
<?php
echo "Good";
?>

(file2.php)
<?php
include('file1.php'); //access the code of file 1.php
echo "Morning";
?>
```

Output:

GoodMorning

Example of require:

Code:-

```
(file1.php)
<?php
echo "Good";
?>

(file2.php)
<?php
require('file1.php'); //access the code of file 1.php
echo "Morning";
?>
```

Output:

GoodMorning

PHP Header

Php header function is used to redirect from one page to another page.

Description

- Remember that **header()** must be called before any actual output is sent, either by normal HTML tags, blank lines in a file, or from PHP.
- It is a very common error to read code with `include()`, or `require()`, functions, or another file access function, and have spaces or empty lines that are output before **header()** is called. The same problem exists when using a single PHP/HTML file.

Parameters

There are four types of parameter we have used in PHP

- 1) Location
- 2) Refresh (with delay second)
- 3) Content-type (With MIME (Multipurpose internet Mail Extentaion))
- 4) Content-disposition (With attachment File)

Location:

Location parameter is used to redirect from one page to another.

Example:

```
<?php
    header("location:www.google.co.in");
?>
```

In this example, When you load this page then page will be redirect to google homepage

Refresh:

Refresh parameter is used to redirect from one page to another after some delay time.

Example:

```
<?php
    header("refresh:5;www.google.co.in");
?>
```

In this example, When you load this page then page will be redirect to google homepage after 5 second.

Content Type & content-disposition:

Content type is used for defining the type of file which we have to download. And content-disposition is used for attaching that file which we have to download.

Example

```
<?php
// We'll be outputting a PDF
header('Content-type: application/pdf');

// It will be called downloaded.pdf
header('Content-Disposition: attachment; filename="downloaded.pdf"');

// The PDF source is in original.pdf
readfile('original.pdf');
?>
```

PHP \$_GET variable

- The \$_GET variable is used to collect values from a form with method = 'get'.
- If we use get method in form than all the data shown in url. So the get method is not secure.
- And If we access the value of form in another than we have to use \$_GET variable.
- By default method of form id "get"

Example:

- In below example, We create one file "get1.php". In that we create Form with method get and give the action of seocnd page where we get the value of textbox.
- In second file get2.php we have access the value of file 1 than we have to use \$_GET variable.
- So, We have to pass value in get1.php file and click on button than it will redirect get2.php file give the output as shown in figure.

Code:-

```
(get1.php)
<form method="get" action="get2.php">
Name: <input type="text" name="name" /><br />
<input type="submit" name="Submit" value="Click" />
</form>
```

```
(get2.php)
<?php
echo "My name is: " . $_GET['name'];
?>
```

Output:

My name is: pratik

PHP \$_POST variable

- The \$_POST variable is used to collect values from a form with method = 'post'.
- If we use post method in form than all the data is not shown in url. So the post method is secure.
- And If we access the value of form in another than we have to use \$_POST variable.

Example:

Code:-

```
(get1.php)
<form method="post" action="get2.php">
Name: <input type="text" name="name" /><br />
<input type="submit" name="Submit" value="Click" /></form>
```

```
(get2.php)
<?php
echo "My name is: " . $_POST['name'];
?>
```

Output:

My name is: pratik

- In Above example, We create one file "get1.php". In that we create Form with method post and give the action of seocnd page where we get the value of textbox.
- In second file get2.php we have access the value of file 1 than we have to use \$_POST variable.

- So, We have to pass value in get1.php file and click on button than it will redirect get2.php file give the output as shown in figure.

PHP \$_REQUEST variable

- The \$_REQUEST variable is used to collect values from a form whatever we define method get or post.

Example:

Code:-

```
(get1.php)
<form method="post" action="get2.php">
Name: <input type="text" name="name" /><br />
<input type="submit" name="Submit" value="Click" />
</form>

(get2.php)
<?php
echo "My name is: " . $_REQUEST['name'];
?>
```

Output:

My name is: pratik

- In Above example, We create one file “get1.php”. In that we create Form with method POST and give the action of second page where we get the value of textbox.
- In second file get2.php we have access the value of file 1 than we have to use \$_REQUEST variable.
- So, We have to pass value in get1.php file and click on button than it will redirect get2.php file give the output as shown in figure.

Module – 3[Database Connectivity]

DBMS & RDBMS

What is DBMS?

A Database Management System (DBMS) is a set of programs that controls the creation, maintenance, and the use of the database in a computer platform or of an organization and its end users. It allows organizations to place control of organization-wide database development in the hands of database administrators (DBAs) and other specialists.

A DBMS is a system software package that helps the use of integrated collection of data records and files known as databases. It allows different user application programs to easily access the same database. DBMSs may use any of a variety of database models, such as the network model or relational model. In large systems, a DBMS allows users and other software to store and retrieve data in a structured way. Instead of having to write computer programs to extract information, user can ask simple questions in a query language.

What is RDBMS?

RDBMS stands for Relational Database Management System. RDBMS data is structured in database tables, fields and records. Each RDBMS table consists of database table rows. Each database table row consists of one or more database table fields.

RDBMS store the data into collection of tables, which might be related by common fields (database table columns). RDBMS also provide relational operators to manipulate the data stored into the database tables. Most RDBMS use SQL as database query language.

Some Important Terminologies

- **Entity:** An entity is a person, place, thing or concept about which data can be collected.
- **Attribute:** In a database management system (DBMS), an attribute may describe a component of the database, such as a table or a field, or may be used itself as another term for a field
- **Relationship:** A relationship, in the context of databases, is a situation that exists between two relational database tables when one table has a foreign key that references the primary key of the other table.
- **Primary Key:** It means Uniquely identify each row in table.
- **Foreign Key:** A FOREIGN KEY in one table points to a PRIMARY KEY in another table.

Difference between DBMS and RDBMS

DBMS:	RDBMS:
1)In dbms no relationship concept	1)It is used to establish the relationship concept between two database objects, i.e, tables
2)It supports Single User only	2)It supports multiple users
3)It treats Data as Files internally	3)It treats data as Tables internally
4)It supports 3 rules of E.F.CODD out of 12 rules	4)It supports minimum 6 rules of E.F.CODD
5)It requires low Software and Hardware Requirements.	5)It requires High software and hardware requirements.
6)FoxPro, IMS are Examples	6)SQL-Server, MySql server, Oracle are examples

MySQL

MySQL is a relational database management system (RDBMS)[1] which has more than 6 million installations. MySQL stands for "My Structured Query Language". The program runs as a server providing multi-user access to a number of databases.

MySQL is often used in free software projects which require a full-featured database management system, such as WordPress, phpBB and other software built on the LAMP software stack. It is also used in very high-scale World Wide Web products including Wikipedia, Google and Facebook

MySQL DataTypes

Below are the different MySQL data types that can be used.

TEXT TYPES

CHAR()	A fixed section from 0 to 255 characters long.
VARCHAR()	A variable section from 0 to 255 characters long.
TINYTEXT	A string with a maximum length of 255 characters.
TEXT	A string with a maximum length of 65535 characters.
BLOB	A string with a maximum length of 65535 characters.
MEDIUMTEXT	A string with a maximum length of 16777215 characters.
MEDIUMBLOB	A string with a maximum length of 16777215 characters.
LONGTEXT	A string with a maximum length of 4294967295 characters.

LONGBLOB	A string with a maximum length of 4294967295 characters.
----------	--

NUMBER TYPES

TINYINT()	-128 to 127 normal 0 to 255 UNSIGNED.
SMALLINT()	-32768 to 32767 normal 0 to 65535 UNSIGNED.
MEDIUMINT()	-8388608 to 8388607 normal 0 to 16777215 UNSIGNED.
INT()	-2147483648 to 2147483647 normal 0 to 4294967295 UNSIGNED.
BIGINT()	-9223372036854775808 to 9223372036854775807 normal 0 to 18446744073709551615 UNSIGNED.
FLOAT	A small number with a floating decimal point.
DOUBLE(,)	A large number with a floating decimal point.
DECIMAL(,)	A DOUBLE stored as a string , allowing for a fixed decimal point.

DATE TYPES

DATE	YYYY-MM-DD.
DATETIME	YYYY-MM-DD HH:MM:SS.
TIMESTAMP	YYMMDDHHMMSS.
TIME	HH:MM:SS.

MISC TYPES

ENUM ()	Short for ENUMERATION which means that each column may have one of a specified possible values.
SET	Similar to ENUM except each column may have more than one of the specified possible values.

- ENUM is short for ENUMERATED list. This column can only store one of the values that are declared in the specified list contained in the () brackets.
Example: ENUM('y','n')
- SET is similar to ENUM except SET may contain up to 64 list items and can store more than one choice

Normalization

- The process of structuring data to minimize duplication and inconsistencies.
- The process usually involves breaking down the single table into two or more tables and defining relationships between those tables.

1NF

- Eliminate duplicative columns from the same table.
- Create separate tables for each group related data and identify each row with unique columns.
- In this, one table will be divided in two tables with relevant contents

Example:

We have one table Customer with following field.

Table Name: Customer

Customer ID	First Name	Surname	Telephone Number
123	Robert	Ingram	555-861-2025
456	Jane	Wright	555-403-1659 555-776-4100
789	Maria	Fernandez	555-808-9633

In above table, We have divide **Customer** table into two table **Customer Name** and **Customer Telephone Number** and give relationship between them with common field **Customer Id**

Customer Name		
Customer ID	First Name	Surname
123	Robert	Ingram
456	Jane	Wright
789	Maria	Fernandez

Customer Telephone Number	
Customer ID	Telephone Number
123	555-861-2025
456	555-403-1659 555-776-4100
789	555-808-9633

2NF

- Once 1NF has been done then and only then 2NF can be done
- In this, provide key constraints to the columns of tables based on uniqueness
- Like assign primary key to one table column and refer this as foreign key in another table

Example:

We have one table Customer with following field.

Table Name: Customer

Cust_id	Cust_Name	Order_id	Order_Name	Sale_Detail
101	Pratik	10	Order1	Sale1
101	Pratik	11	Order2	Sale2
102	Nikunj	12	Order3	Sale3
103	Indra	13	Order4	Sale4

Now in 2NF we have to breaking down above table in to three tables and give the relationship between them

Table Name: Customer

Cust_id	Cust_Name
101	Pratik
102	Nikunj
103	Indra

Table Name: OrderDetail

Order_id	Order_Name
10	Order1
11	Order2

12	Order3
13	Order4

Table Name: SalesDetail

Cust_id	Order_id	Sale_detail
101	10	Sale1
101	11	Sale2
102	12	Sale3
103	13	Sale4

3NF

- Only after 2NF, third normalization can be done
- In this, further more analyze tables and divide them for data uniqueness.

Example:

We have student detail table with following field.

Table Name: StudentDetail

Student_id	Student_Name	DOB	Street	city	state	zip
------------	--------------	-----	--------	------	-------	-----

In above table **Student_id** is a primary key, but **street,city** and **state** depend on **zip**. So, the dependency between **zip** and other field is called transitive dependency. So in 3NF we move the **street, city** and **state** to new table with **zip** as a primary key

Table Name: New StudentDetail

Student_id	Student_Name	DOB	zip
------------	--------------	-----	-----

Table Name: Address

zip	street	city	state
-----	--------	------	-------

The Advantage of removing transitive dependency is,

- Amount of data duplication is reduced.
- Data integrity achieved

Connection With Mysql and Database

- For database connection, First of all we have to create one database in Mysql.
- Suppose We create database "User" in phpmyadmin.
- Then we have to create table Country with following structure.

Field Name	Datatype(size)	Constraint
cid	Int(11)	Primarykey with AutoIncrement
cname	Varchar(50)	

- Now create another table reg with following structure.

Field Name	Datatype(size)	Constraint
uid	Int(11)	Primarykey with AutoIncrement
uname	Varchar(50)	
pass	Varchar(50)	
gender	Varchar(50)	

hobby	Varchar(50)	
country	Int(11)	Foreign Key

- Now we have to create php file for Registration Form (reg.php).
- We have to take all type of field.

```
<form method="post">
    <table align="center" border="1">
        <tr>
            <td>User Name</td>
            <td><input type="text" name="uname" /></td>
        </tr>
        <tr>
            <td>Password</td>
            <td><input type="password" name="pass" /></td>
        </tr>
        <tr>
            <td>Gender</td>
            <td>
                <input type="radio" name="gender" value="male"/>Male
                <input type="radio" name="gender" value="female"/>Female
            </td>
        </tr>
        <tr>
            <td>Hobby</td>
            <td>
                <input type="checkbox" name="chk[]" value="read"/>Read
                <input type="checkbox" name="chk[]" value="play"/>Play
            </td>
        </tr>
        <tr>
            <td>Country</td>
            <td>
                <select name="country">
                    <option value="Select">Select</option>
                </select>
            </td>
        </tr>
        <tr>
            <td align="center" colspan="2">
                <input type="submit" name="submit" value="Insert" />
            </td>
        </tr>
    </table>
</form>
```

- Now we have to create connection file. (conn.php)

```
<?php
$con = mysqli('localhost','root','','user');
?>
```

- Now first of all in reg.php page we have fetch country in dropdown list.
- For that we have to include conn.php file into reg.php file and than we have fetch country.

reg.php

```
<?php
include('conn.php');
?>
```

- And do following code at dropdwon place.

```
<tr>
    <td>Country</td>
    <td>
        <?php
            $sel = "select * from country";
            $ex = $con->query($sel);

            ?>
            <select name="country">

                <option value="Select">Select</option>

                <?php
                    while($r = $ex->fetch_object())
                    {
                        ?>
                        <option value="<?php echo $r->cid; ?>"><?php echo $r->cname; ?></option>
                    }
                ?>
            </select>
        </td>
    </tr>
```

- Now, craete folowing code of insert data into database in reg.php

```
//insert
if(isset($_REQUEST['submit']))
{
    $u = $_REQUEST['uname'];
    $p = $_REQUEST['pass'];
    $g = $_REQUEST['gender'];
    $h = implode(",", $_REQUEST['chk']);
    $c = $_REQUEST['country'];

    $ins = "insert into reg (uname,pass,gender,hobby,country) values ('$u','$p','$g','$h','$c')";
    $ex = $con->query($ins);
}
```

- After insert record in database we have to view the record in another page.
- So we have to create view record page (view.php) with delete(link), edit(link), status(link), join , order by and multidelete checkbox with following code.

```
<?php
include('conn.php');
$sel = "select reg.*,country.cname from reg join country on reg.country = country.cid order by
uid"; //select query with join and order by
$ex = $con->query($sel);
?>
<form method="post">
<table align="center" border="1">
    <tr>
        <th>#</th>
        <th>Uid</th>
        <th>UserName</th>
        <th>Password</th>
        <th>Gender</th>
        <th>Hobby</th>
        <th>Country</th>
        <th>Status</th>
        <th align="center" colspan="2">Action</th>
    </tr>
    <?php
    while($r = mysql_fetch_array($ex))
    {
        ?>
        <tr>
            <td><input type="checkbox" name="chk[]" value="<?php echo $r['uid'];?>"
            /></td>
            <td><?php echo $r->uid;?></td>
            <td><?php echo $r->uname;?></td>
            <td><?php echo $r->pass;?></td>
            <td><?php echo $r->gender;?></td>
            <td><?php echo $r->hobby;?></td>
            <td><?php echo $r->cname;?></td>
            <td><a href="view.php?st_id=<?php echo $r->uid;?>&&sts=<?php echo $r->
            status;?>"><?php echo $r->status;?></a></td>
            <td><a href="view.php?del_id=<?php echo $r->uid;?>">Delete</a></td>
            <td><a href="reg.php?edit_id=<?php echo $r->uid;?>">Edit</a></td>
        </tr>
        <?php
        }
        ?>
        <tr>
            <td><input type="submit" name="del" value="Delete" /></td>
        </tr>
    </table>
```

</form>

- Now we have create code for delete particular record. So, we already create link for delete in view.php with query string.
- Now if we get the request of “del_id” variable from query string than particular record will be deleted. For that we have to add following code in view.php.

```
//delete
if(isset($_REQUEST['del_id']))
{
    $d = $_REQUEST['del_id'];
    $del = "delete from reg where uid = '$d'";
    $ex = $con->query($del);
    header('location:view.php');
}
```

- Now we have create code for edit particular record. So, we already create link for edit in view.php with query string.
- Now if we get the request of “edit_id” variable from query string than particular record will be edit and page will be redirect on reg.php.
- Then we have to create code for fetch record of particular user in reg.php.

```
//fetch
if(isset($_REQUEST['edit_id']))
{
    $ed = $_REQUEST['edit_id'];
    $sel = "select * from reg where uid = '$ed'";
    $ex = $con->query($sel);
    $rw = $ex->fetch_Object();
}
```

- Now we have display record in form of particular user with use of **\$rw** variable.
- For that we have to edit following code in reg.php.

```
<form method="post">
<table align="center" border="1">
<tr>
<td>User Name</td>
<td><input type="text" name="uname" value="<?php echo $rw->uname;?>" /></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" name="pass" value="<?php echo $rw->pass;?>" /></td>
</tr>
<tr>
<td>Gender</td>
<td>
```



```
<input type="radio" name="gender" value="male"
<?php
    if(isset($_REQUEST['edit_id']))
    {
        if($rw->gender == 'male')
        {
            echo "checked = 'checked'";
        }
    }
?>/>Male
<input type="radio" name="gender" value="female"
<?php
if(isset($_REQUEST['edit_id']))
{
    if($rw->gender == 'female')
    {
        echo "checked = 'checked'";
    }
}
?>/>Female
</td>
</tr>
<tr>
    <td>Hobby</td>
    <td>
        <?php
        if(isset($_REQUEST['edit_id']))
        {
            $t = $r->hobby;
            $h = explode(",",$t);
        }
        ?>
        <input type="checkbox" name="chk[]" value="read"
        <?php
        if(isset($_REQUEST['edit_id']))
        {
            if($h[0] == 'read')
            {
                echo "checked = 'checked'";
            }
        }
        ?>/>Read
        <input type="checkbox" name="chk[]" value="play"
        <?php
        if(isset($_REQUEST['edit_id']))
        {
            if($h[0] == 'play' || $h[1] == 'play')
            {
```

```
        echo "checked = 'checked'";
    }
}
?> />Play
</td>
</tr>
<tr>
    <td>Country</td>
    <td>
        <?php
            $sel = "select * from country";
            $ex = $con->query ($sel);

            ?>
            <select name="country">
            <option value="Select">Select</option>
            <?php
                while($r = $ex->fetch_object())
                {
                    if($r->id == $r->id)
                    {
                        ?>
                        <option value="<?php echo $r->id; ?>" selected="selected"><?php echo $r->name;
?></option>
                        <?php
                            }
                            else
                            {
                                ?>
                                <option value="<?php echo $r->id; ?>"><?php echo $r->name; ?></option>
                                <?php }
                                } ?>
                            </select>
                        </td>
                    </tr>

                    <tr>
                        <td align="center" colspan="2">
                            <?php
                                if(isset($_REQUEST['edit_id']))
                                {
                                    ?>
                                    <input type="submit" name="update" value="Update" />
                                    <?php
                                        }
                                        else
                                        {
                                            ?>
                                            <input type="submit" name="submit" value="Insert" />

```

```
<?php
}
?>
</td>
</tr>
</table>
</form>
```

- After we have to create following code we get the value of particular user in reg form.
- Now we have create following code for update that recored in reg.php.

```
//update
if(isset($_REQUEST['edit_id']) && ($_REQUEST['update']))
{
    $e = $_REQUEST['edit_id'];
    $u = $_REQUEST['uname'];
    $p = $_REQUEST['pass'];
    $g = $_REQUEST['gender'];
    $h = implode(",", $_REQUEST['chk']);
    $c = $_REQUEST['country'];

    $up = "update reg set uname = '$u',pass = '$p',gender = '$g',hobby = '$h',country = '$c'
where uid = '$e'";
    $ex = $con->query ($up);
    header('location:view.php'); }
```

Now change the status of user “enable/disable” we have to add new filed in our “reg” table.

FieldName: status

Data type: ENUM

Value : ‘enable’,‘disable’

Default: as define > disable

- we have already fetch status in view.php page.
- Now we have to change status of user disable to enable or enable to disable, we have to do following cde.

```
//status
if(isset($_REQUEST['st_id']))
{
    $stid = $_REQUEST['st_id'];
    $sts = $_REQUEST['sts'];

    if($sts == 'disable')
    {
        $up = "update reg set status = 'enable' where uid = '$stid'";
        $ex = $con->query ($up);
        header('location:view.php');
    }
}
```

```
else
{
    $up = "update reg set status = 'disable' where uid = '$stid'";
    $ex = $con->query ($up);
    header('location:view.php');
}
}
```

- Now for multidelete, we already create checkbox and delete button in view.php.
- So, For delete multiple record we have to following code.

```
if(isset($_REQUEST['del']))
{
    $ch = $_REQUEST['chk'];
    $c = count($ch);
    for($i = 0; $i < $c; $i++)
    {
        $del = "delete from reg where uid = '$ch[$i]'";
        $ex = $con->query ($del);
        header('location:view.php');
    }
}
```

Triggers

Types of Triggers

A trigger defines an action the database should take when some database-related event occurs. The execution of triggers is transparent to users. It means actions performed by a trigger are executed automatically. Triggers are executed by the database when specific types of data manipulation commands are performed on specific tables. Such commands may include *inserts*, *updates*, *deletes*. Updates of specific columns may also be used as triggering events. The type of a trigger is defined by the type of triggering transaction and by the level at which the trigger is executed.

Row-Level Triggers

Row-level triggers execute once for each row in a transaction. Row-level triggers are created using the **for each row** clause in the **create trigger** command. For instance if we insert in a single transaction 20 rows to the table EMPLOYEE, the trigger is executed 20 times.

Statement-Level Triggers

Statement-level triggers execute once for each transaction. When we insert in one transaction 20 rows to EMPLOYEE table, then statement-level trigger is executed only once. Statement-level triggers are default type of triggers created via the **create trigger** command.

BEFORE and AFTER Triggers

Since triggers occur because of events, they may be set to occur immediately before or after those events. Within the trigger, you will be able to reference the old and new values involved in transaction. The access required for the old and new data may determine which type of trigger you need. "Old" refers to the data as it existed prior to the transaction, like *update* or *delete*. "New" values are the data values that the transaction creates, like *insert* record.

INSTEAD OF Triggers

The INSTEAD OF triggers are used to tell ORACLE (from version 8) what to do instead of performing triggering action. For example, you could use an INSTEAD OF trigger to redirect table insert into a different table. The code in the INSTEAD OF triggers is executed in place of the *insert*, *update* or *delete* commands you enter.

Syntax:

DELIMITER&&

create trigger <trigger_name><action><Event_name> ON <table_name>

For each row

Begin

New.<field_name> = value

End&&

DELIMITER;

Example:

DELIMITER&&

create trigger t1 before insert ON country FOR EACH ROW

BEGIN

NEW.cname = UCASE(new.cname);

END&&

DELIMITER;

In above Example, if we insert any countryname in country table by default it take in uppercase.

Indexing In Mysql

An index is a physical structure containing pointers to the data. Indices are created in an existing table to locate rows more quickly and efficiently. It is possible to create an index on one or more columns of a table, and each index is given a name. The users cannot see the indexes, they are just used to speed up queries. Effective indexes are one of the best ways to improve performance in a database application. A table scan happens when there is no index available to help a query. In a table scan SQL Server examines every row in the table to satisfy the query results. Table scans are sometimes unavoidable, but on large tables, scans have a terrific impact on performance.

Clustered indexes define the physical sorting of a database table's rows in the storage media. For this reason, each database table may have only one clustered index.

Non-clustered indexes are created outside of the database table and contain a sorted list of references to the table itself.

Syntax:

create index <index_name> on <table_name><field_name>

Example:

create index userindex on user (uname);

This example create index on username field in user table.

What is a SQL Injection Attack?

Many web applications take user input from a form

- Often this user input is used literally in the construction of a SQL query submitted to a database.

For example:

- SELECT productdata FROM table WHERE productname = 'user input product name';
- A SQL injection attack involves placing SQL statements in the user input

An Example SQL Injection Attack

Product Search: OR 'x' = 'x'

- This input is put directly into the SQL statement within the Web application:
 - \$query = "SELECT prodinfo FROM proddtable WHERE prodname = "" .
 - \$_POST['prod_search'] . """;
- Creates the following SQL:
 - SELECT prodinfo FROM proddtable WHERE prodname = 'blah' OR 'x' = 'x'
 - Attacker has now successfully caused the entire database to be

mysql_real_escape_string()

This function escapes special characters in a string for use in SQL statements.

- \$esc_input = mysql_real_escape_string(\$user_input);
- Specifically, it prepends backslashes to the following characters:
 - \x00, \n, \r, \, ', " and \x1a.
 - By escaping quotes, makes it much more difficult to perform SQL injection attacks
- This function assumes a MySQL connection is active
 - It considers the current character set used by the connection

Module- 4 [HTML]

Introduction to HTML

- HTML, which stands for Hyper Text Markup Language, is the predominant markup language for web pages.
- It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists etc as well as for links, quotes, and other items.
- It allows images and objects to be embedded and can be used to create interactive forms.
- It is written in the form of HTML elements consisting of "tags" surrounded by angle brackets within the web page content.
- It can include or can load scripts in languages such as JavaScript which affect the behavior of HTML processors like Web browsers; and Cascading Style Sheets (CSS) to define the appearance and layout of text and other material.
- The W3C, maintainer of both HTML and CSS standards, encourages the use of CSS over explicit presentational markup
- Hyper Text Markup Language (HTML) is the encoding scheme used to create and format a web document.

Example:

<pre><html> <body> <h1>My First Heading</h1> <p>My first paragraph.</p> </body> </html></pre>	<p>Output</p> <p>My First Heading</p> <p>My first paragraph.</p>
---	---

HTML Element and Tags

HTML tags are used for designing your web page. There are so many tags available in HTML.

For Ex: <p>: Used for creating paragraph.

: Used for creating unordered list.

: Used for creating item list.

: Used for creating ordered list.

: Used for making font in bold.

<i>: Used for making font in italic.

: Used for inserting image in web form.

<a>: Used for creating hyperlink.

and so on.

HTML Anchor Tag

- HTML Anchor tag is used for creating hyperlink.

Example:

Code:

```
<body>
<a href="reg.php">Registration</a>
</body>
```

Output

[Registration](#)

- In above example, if we have to create one link Registration.
- In <a> tag we use href. In href we have to write the name of page when we have to redirect.
- So, if user clicks on this link then he is redirected on reg.php page.

HTML Images

- In HTML Images are used for inserting image in our web page.

For inserting image we have to use Tag

Example:

Code:

```
<body>

</body>
```

- In above example, if we have to insert image in our webpage
- In tag we use src. In src we have to write or select the path of image where image is stored in our computer.
- We also set the height and width of image using height and width property of tag.

HTML List

- In HTML and tags are used for creating list in our webpage.
- tag is used for creating ordered list

 tag is used for create unordered list.

Example:

Code:-

```
<body>
<b>Tops courses</b>
<ul>
    <li>PHP</li>
    <li>JAVA</li>
    <li>.Net</li>
    <li>iPhone</li>
</ul>
</body>
```

Output:

Tops courses

- PHP
- JAVA
- .Net
- iPhone

HTML Table

- Table is collection of Row and Column we design content in table format -

Example:

Code:-

```
<body>
<table>
<tr><td align="center" colspan="2">Application Form</td></tr>
<tr><td>Username</td><td><input type="text" /></td></tr>
<tr><td>Password</td><td><input type="password" /></td></tr>
<tr><td align="center" colspan="2"><input type="submit"
value="save"/></td></tr>
</table>
</body>
```

Intro to HTML Forms

- Types of input we can collect via forms:
- Text via text boxes and text areas
- Selections via radio buttons, check boxes, pull down menus, and select boxes
- Form actions do all of the work
- Usually through button clicks

Form Actions

```
<form name="input" action="html_form_action.php" method="get">
```

- Action specifies what file the form data will be sent to (on the server)
- Method specifies by which HTTP protocol
- get
- post
- Method is important on the receiving end

Module – 5[CSS]

Introduction

- Cascading Style Sheets (CSS).
- With CSS you will be able to:
- Add new looks to your old HTML
- Completely restyle a web site with only a few changes to your CSS code
- Use the "style" you create on any web page you wish!

CSS Selector

- `SELECTOR { PROPERTY: VALUE }`
- `HTML tag{"CSS Property": "Value" ; }`
- The selector name creates a direct relationship with the HTML tag you want to edit. If you wanted to change the way a paragraph tag behaved, the CSS code would look like:
- `p { PROPERTY: VALUE }`
- The above example is a template that you can use whenever you are manipulating the paragraph HTML element

Three Method of CSS

- 1.Internal
- 2.External
- 3.Inline

Three Types of CSS

- Using Selector
- Using Class
- Using ID


Example of Internal CSS Code

Here we create sample code for external css in head section by using selector.

Code:-

```
<head>
<style type="text/css">
  p {color: white; }
  body {background-color: black; }
</style>
</head>
<body>
  <p>White text on a black background!</p>
</body>
```

Output:



White text on a black background!

- We chose the HTML element we wanted to manipulate. `-p{ ; ; }`

- Then we chose the CSS attribute color. -p { **color;** }
- Next we choose the font color to be white. -p { color:**white;** }
- Now all text within a paragraph tag will show up as white! Now an explanation of the CSS code that altered the <body>'s background:
- We choose the HTML element Body -**body { : ; }**
- Then we chose the CSS attribute. -body { **background-color;** }
- Next we chose the background color to be black. -body { background-color:**black;** }

Example of Internal CSS Code

- When using CSS it is preferable to keep the CSS
- separate from your HTML. Placing CSS in a separate file
- allows the web designer to completely differentiate
- between content (HTML) and design (CSS). External
- CSS is a file that contains only CSS code and is saved
- with a ".css" file extension. This CSS file is then
- referenced in your HTML using the <link>**instead** of <style>.
- Now First of all we have to create one css file with extension .css like style.css
- In that we have to create different css code for different code.

Code:-

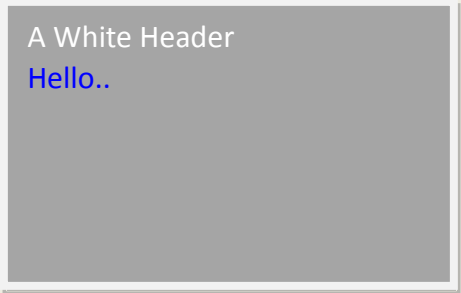
```
body{ background-color: gray } p { color: blue; }  
h3{ color: white; }
```

- Save this file as a style.css.
- Now we have to create html file and include this style.css file with <link> tag

Code:-

```
<html>  
<head>  
<link href="style.css" rel="stylesheet" type="text/css" />  
</head>  
<body>  
    <h3> A White Header </h3>  
    <p> Hello.. </p>  
</body>  
</html>
```

Output:



A White Header
Hello..

Example of Inline css

- It is possible to place CSS right in the thick of your HTML code, and this method of CSS usage is referred to as inline css.

- Inline CSS has the highest priority out of external, internal, and inline CSS. This means that you can override styles that are defined in external or internal by using inline CSS

Example:

Code:-

```
<p style="background: blue; color: white;">  
A new background and font color with inline CSS  
</p>
```

Output:

A new background and font
color with inline CSS

Margins and Padding

- **Margin** and **padding** are the two most commonly used properties for spacing-out elements. A margin is the space **outside** of the element, whereas padding is the space **inside** the element
- The four sides of an element can also be set individually. **margin-top**, **margin-right**, **margin-bottom**, **margin-left**, **padding-top**, **padding-right**, **padding-bottom** and **padding-left** are the self-explanatory properties you can use.

CSS Pseudo-classes

Anchor/Link States

- You may not know it, but a link has four different states that it can be in. CSS allows you to customize each state. Please refer to the following keywords that each correspond to one specific state:
- link - this is a link that has not been used, nor is a mouse pointer hovering over it
- visited - this is a link that has been used before, but has no mouse on it
- hover - this is a link currently has a mouse pointer hovering over it/on it
- active - this is a link that is in the process of being clicked
- Using CSS you can make a different look for each one of these states, but at the end of this lesson we will suggest a good practice for CSS Links.

Example:

- Specify the color of links:
- ```
a:link {color:#FF0000} /* unvisited link */
a:visited {color:#00FF00} /* visited link */
a:hover {color:#FF00FF} /* mouse over link */
a:active {color:#0000FF} /* selected link */
```

## Module – 6[Applicability To Industry Standards]

### OOPS Concepts

#### The Benefits of OOP

The birth of object-oriented programming represented a major paradigm shift in development strategy, refocusing attention on an application's data rather than its logic. To put it another way, OOP shifts the focus from a program's procedural events toward the real-life entities it ultimately models. The result is an application that closely resembles the world around us.

#### Key OOPS Concepts

##### Class:

- Group of data member and member method is called class.
- Every class definition begins with the keyword class, followed by a class name, followed by a pair of curly braces which enclose the definitions of the class's properties and methods.
- The class name can be any valid label which is not a PHP reserved word.
- A valid class name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.
- A class may contain its own constants, variables (called "properties"), and functions (called "methods").
- The pseudo-variable *\$this* is available when a method is called from within an object context. ]
- *\$this* is a reference to the calling object (usually the object to which the method belongs, but possibly another object, if the method is called statically from the context of a secondary object).

##### Syntax:

```
class <class_name>
{
 Datamember list;
 Member function list
}
```

##### Example:

###### Code:-

```
<?php
class simpleclass
{
 public $var = "Hello Calss...";

 public function display()
 {
 echo $this->var;
 }
}
$s= new simpleclass();
$s->display();
?>
```

###### Output:

Hello Calss...

one object "\$s". And using "\$s" we access the function display() for print value.

## Extends

- A class can inherit the methods and properties of another class by using the keyword extends in the class declaration.
- It is not possible to extend multiple classes; a class can only inherit from one base class.
- The inherited methods and properties can be overridden by redeclaring them with the same name defined in the parent class.
- However, if the parent class has defined a method as final, that method may not be overridden. It is possible to access the overridden methods or static properties by referencing them with parent::

## Syntax:

```
class <class_name1>{
 Datamember list;
 Member function list }
class <class_name2> extends <class_name1>{
 Datamember list;
 Member function list }
```

## Example:

### Code:-

```
<?php
class a
{
 public function adisplay()
 {
 echo "Good";
 }
}
class b extends a
{
 public function bdisplay()
 {
 echo "Morning";
 }
}
$b = new b();
$b->bdisplay();
$b->adisplay();
?>
```

### Output:

MorningGood

- In above example, we create "class a" and create function "adisplay" in it.
- Now we create "class b" and inherit "class a" by using extends keyword, and create function "bdisplay" in it.
- Now we create object of child "class b" and access the function of base "class a" and as well as "class b".

## Visibility

- The visibility of a property or method can be defined by prefixing the declaration with the keywords: public, protected or private.
- Public Declared items can be accessed everywhere.
- Protected limits access to inherited and parent classes (and to the class that defines the item).
- Private limits visibility only to the class that defines the item.

### Members Visibility

- Class members must be defined with public, private, or protected

## Example:

### Code:-

```
<?php
class myclass
{
 public $public = "Public";
 private $private = "private";
 protected $protected = "protected";
 function display()
 {
 echo $this->public;
 echo $this->private;
 echo $this->protected;
 }
}
$my = new myclass();
//echo $my->public;
//echo $my->protected;
//echo $my->private;
$my->display();
?>
```

### Output:

Publicprivateprotected

## Constructor

- We can define constructor using “\_\_construct” or same name as class name.
- PHP 5 allows developers to declare constructor methods for classes.
- Classes which have a constructor method call this method on each newly-created object, so it is suitable for any initialization that the object may need before it is used.
- **Note:** Parent constructors are not called implicitly if the child class defines a constructor. In order to run a parent constructor, a call to
- **parent::\_\_construct()**
- within the child constructor is required.

### Syntax:

```
class <class_name1>
{public function __construct() or public function <same name as class name>{
 statement}
}
```

## Example:

### Code:-

```
<?php
class a
{
 public function __construct()
 {
 echo "In base class constructor....";
 }
}
class b extends a
{
 public function b()
 {
 parent::__construct();
 echo "in subclass construct";
 }
}
$b= new b();
?>
```

### Output:

In base class constructor....in  
subclass construct

- In above example we create class a and in that create constructor “\_\_construct”.
- And then create second “class b” and create another constructor “b” (same name as class name).
- And after that we create object of child class so constructor will automatically call.

## Destructor

- We create destructor by using “\_\_destruct” function.
- PHP 5 introduces a destructor concept similar to that of other object-oriented languages, such as C++.
- The destructor method will be called as soon as all references to a particular object are removed or when the object is explicitly destroyed or in any order in shutdown sequence.

### Syntax:

```
class <class_name1>
{
 public function __destruct()
 {
 Statement
 }
}
```

## Example:

### Code:-

```
<?php
class a
{public function __destruct()
 {
 echo "destructor call";
 }
}
```

### Output:

destructor call

- In above example, we create “class a” and in that we create one object of “class a” so, destructor automatically called.

## Scope Resolution Operator (::)

- The Scope Resolution Operator (also called Paamayim Nekudotayim) or in simpler terms, the double colon, is a token that allows access to static, constant, and overridden properties or methods of a class.
- When referencing these items from outside the class definition, use the name of the class.
- As of PHP 5.3.0, it's possible to reference the class using a variable. The variable's value cannot be a keyword (e.g. *self*, *parent* and *static*).

**Example:**

**Code:-**

```
<?php
class A
{const va = "Constant value";}
A::va;
class B extends A
{public Static $st = "static value";
 function display(){
 echo parent::va;
 echo self::$st;
 }
}
$obj = new B();
$obj->display();
?>
```

**Output:**

Constant value static value

- In above, example we create “class A” and define one constant variable.
- Then We create class B and define static variable and access the constant and static variable in display() function by using scope resolution operator.

When an extending class overrides the parent's definition of a method, PHP will not call the parent's method. It's up to the extended class on whether or not the parent's method is called. This also applies to Constructors and Destructors, Overloading, and Magic method definitions.

## Class Abstraction

- PHP 5 introduces abstract classes and methods. It is not allowed to create an instance of a class that has been defined as abstract.
- Any class that contains at least one abstract method must also be abstract.
- Methods defined as abstract simply declare the method's signature they cannot define the implementation.



- 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 (or a less restricted) visibility

## Example

- In below example, We create one abstract class and define two abstract methods.
- And in another class we implement both of the methods but using an object we can only access one method.
- In abstraction, we must implement all abstract methods in another class otherwise it shows an error.

### Code:-

```
<?php
abstract class a
{
 abstract public function dis1();
 abstract public function dis2();
}
class b extends a
{
 public function dis1()
 {
 echo "Method 1";
 }
 public function dis2()
 {
 echo "Method 2";
 }
}
$obj = new b();
$obj->dis1();
?>
```

### Output:

Method 1

## Object Interfaces

- Object interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are handled.
- Interfaces are defined using the interface keyword, in the same way as a standard class, but without any of the methods having their contents defined.
- All methods declared in an interface must be public, this is the nature of an interface.
- If we create more than one interface then we can inherit it by using “implements” operator in another class.
- Classes may implement more than one interface if desired by separating each interface with a comma.
- **Note:** A class cannot implement two interfaces that share function names, since it would cause ambiguity.

## Example:

## Code:-

```
<?php
interface i1
{
 public function dis1();
}
interface i2
{
 public function dis2();
}
class b implements i1,i2
{
 public function dis1()
 {
 echo "Method 1";
 }
 public function dis2()
 {
 echo "Method 2";
 }
}
$obj = new b();
$obj->dis1();
$obj->dis2();
```

## Output:

Method 1Method 2

- In above Example, we create two interface i1,i2 and define two method respectively.
- And after that we create one class and implement two interface in that by using “implements” operator.

## Overloading

- Overloading in PHP provides means to dynamically "create" properties and methods.
- These dynamic entities are processed via magic methods one can establish in a class for various action types.
- The overloading methods are invoked when interacting with properties or methods that have not been declared or are not visible in the current scope.
- The rest of this section will use the terms "inaccessible properties" and "inaccessible methods" to refer to this combination of declaration and visibility.
- All overloading methods must be defined as public.
- Note: None of the arguments of these magic methods can be passed by reference. PHP's interpretation of "overloading" is different than most object oriented languages.
- Overloading traditionally provides the ability to have multiple methods with the same name but different quantities and types of arguments.

## Property overloading

- Void **\_\_set**(string \$name, mixed \$value)
- Mixed **\_\_get**(string \$name)
- **\_\_set()** is run when writing data to inaccessible properties.

- **\_\_get()** is utilized for reading data from inaccessible properties.
- being interacted with. The
- **\_\_set()** method's *\$value* argument specifies the value the *\$name* property should be set to. Property overloading only works in object context. These magic methods will not be triggered in static context. Therefore these methods can not be declared static.
- **Note:** The return value of **\_\_set()** is ignored because of the way PHP processes the assignment operator. Similarly, **\_\_get()**
- is never called when chaining assignments together like this:

## Example:

### Code:-

```
<?php
class myclass
{
 function __set($data,$value)
 {
 echo "__set is called
";
 echo "Variable= ".$data."
";
 echo "Value= ".$value;
 }
}
$myclassobj = new myclass();
$myclassobj->x=30;
?>
```

### Output:

```
__set is called
Variable= x
Value= 30
```

- In above example, we create one method “\_\_set” and pass two parameters \$data and \$value.
- In that method we print the value of \$data and \$value.
- And using object we have to direct values not call the “\_\_set” function as par normal method because \_\_set method get the value automatically.

## Method overloading

mixed **\_\_call**( string \$name, array \$arguments)

- **\_\_call()** is triggered when invoking inaccessible methods in an object context.
- The *\$name* argument is the name of the method being called. The *\$arguments* argument is an enumerated array containing the parameters passed to the *\$name*'ed method.

## Example:

### Code:-

```
<?php
class abc
{private $x = array(1,2,3);
 public function __call ($m,$a){
 print "Method $m called:\n";
 var_dump($a);
 return $this->x;}}

$obj = new abc();
$a = $obj->test(1,"2",3.4,true);
var_dump($a);
?>
```

### Output:

```
Method test called: array(4) { [0]=>
int(1) [1]=> string(1) "2" [2]=> float(3.4)
[3]=> bool(true) } array(3) { [0]=> int(1)
[1]=> int(2) [2]=> int(3) }
```

# TOPS Technologies

## Type Hinting

PHP 5 introduces Type Hinting. Functions are now able to force parameters to be objects (by specifying the name of the class in the function prototype) or arrays (since PHP 5.1). However, if NULL is used as the default parameter value, it will be allowed as an argument for any later call.

### Example:

#### Code:-

```
<?php
class a
{
 public $var = 'Hello World';
}
function display(a $v)
{ echo $v->var;}
$myclass = new a;
display($myclass);
?>
```

#### Output:

```
Hello World
```

## Object Iteration

PHP 5 provides a way for objects to be defined so it is possible to iterate through a list of items, with, for example a foreach statement. By default, all visible properties will be used for the iteration.

### Example:

#### Code:-

```
<?php
class MyClass
{
 public $var1 = 'value 1';
 public $var2 = 'value 2';
 public $var3 = 'value 3';

 protected $protected = 'protected var';
 private $private = 'private var';

 function iterateVisible()
 {
 echo "MyClass::iterateVisible:
";
 foreach($this as $key => $value)
 {
 print "$key => $value
";
 }
 }
}
$class = new MyClass();
foreach($class as $key => $value)
{
 print "$key => $value
";
}
echo "
";
$class->iterateVisible();
?>
```

#### Output:

```
var1 => value 1
var2 => value 2
var3 => value 3

MyClass::iterateVisible:
var1 => value 1
var2 => value 2
var3 => value 3
protected => protected var
private => private var
```

## Final Keyword

PHP 5 introduces the final keyword, which prevents child classes from overriding a method by prefixing the definition with final. If the class itself is being defined final then it cannot be extended.

### Example:

#### Code:-

```
<?php
class a
{final function test(){
 echo "This is final method";}}
class b extends a{
 function test() {
 echo "This is final method of B class";
 }
}
?>
```

#### Output:

**Fatal error:** Cannot override final method a::test() in **C:\xampp\htdocs\c\temp.php** on line 15

## Overview of MVC Architecture

MVC implements the Model-View-Controller (MVC) design pattern, and encourages application design based on the Model 2 paradigm. This design model allows the Web page or other contents (View) to be mostly separated from the internal application code (Controller/Model), making it easier for designers and programmers to focus on their respective areas of expertise.

- The Model contains the business logic for the application.
- The Controller then forwards the request to the appropriate View component, which is usually implemented using a combination of HTML with PHP tags in the form of templates.
- The resulting contents are returned to the client browser, or via another protocol such as SMTP.
- For example, we have to create one application using MVC.
- So, First of we have to create one database in phpmyadmin.
- And create two tables **country** and **reg** and give the relationship between them.
- Then we have to create registration page (reg.php) and save in one folder.

Here following code is for registration page(reg.php)

```
<?php
include 'control.php';
?>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<title></title>
</head>
<body>
<form method="post">
<table align="center" border="1">
<tr>
<td>Username</td>
<td><input type="text" name="uname"/></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" name="pass" /></td>
</tr>
<tr>
<td>Country</td>
<td>
<select name="country">
<option value="select">Select</option>
<?php
 foreach($con as $cn)
 {
 ?>
<option value="<?php echo $cn->cid;?>"><?php echo $cn->cname;?></option>

<?php
 }
 ?>
</select>
</td>
</tr>
<tr>
<td align="center" colspan="2">
<input type="submit" value="Submit" name="submit"/>
</td>
</tr>
</table>
```

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

Create Connection file

Class Connect

```
{
 public function model()
 {$con = new mysqli('localhost','root','','sample');
 Return $con;
 }
}
```

Now create one model file (model.php)

```
<?php
```

```
include('conn.php');
```

```
$obj = new conn();
```

```
$c = $obj->connect();
```

```
class model {
```

```
 public function selectall($c, $table) {
 $sel = "select * from $table";
 $ex = $c->query($sel);
 while ($r = $ex->fetch_object()) {
 $res[] = $r;
 }
 return $res;
 }
}
```

```
 public function add_data($c, $table, $data) {
```

```
 $k = array_keys($data);
 $field = implode(",", $k);
 $v = array_values($data);
 $value = implode("'", $v);
 $ins = "insert into $table ($field) values ('$value')";
 $c->query($ins);
 }
```

```
 public function del_data($c, $table, $id) {
```

```
 $k = array_keys($id);
 $field = implode(",", $k);
 $v = array_values($id);
 $value = implode("'", $v);
```

```
$del = "delete from $table where $field = '$value'";
$c->query($del);
}

public function selectwhere($c, $table, $where) {
 $key = array_keys($where);
 $value = array_values($where);
 $i = 0;
 $sql = "select * from $table where 1 = 1";

 foreach ($where as $v) {
 $sql.=" and " . $key[$i] . " = '" . $value[$i] . "' ";
 $i++;
 }

 $v = $c->query($sql);

 return $v;
}

public function update_data($c, $table, $data, $where) {

 $dkey = array_keys($data);
 $dvalue = array_values($data);

 $wkey = array_keys($where);
 $wvalue = array_values($where);
 $i = 0;
 $k = 0;
 $sql = "update $table set";

 $size = sizeof($data);

 foreach ($data as $v) {
 if ($size == $i + 1) {
 $sql.=" " . $dkey[$i] . " = '" . $dvalue[$i] . "' ";
 } else {
 $sql.=" " . $dkey[$i] . " = '" . $dvalue[$i] . "', ";
 }
 $i++;
 }
 $sql.=" where 1 = 1";

 foreach ($where as $val) {
 $sql.=" and " . $wkey[$k] . " = '" . $wvalue[$k] . "' ";
 $k++;
 }
}
```



```
$query = $c->query($sql);
return $query;
}

public function sel_join($c,$tbl1,$tbl2,$where)
{
 $sel="select * from $tbl1 join $tbl2 on $where";

 $sex = $c->query($sel);
 while ($r = $sex->fetch_object())
 {
 $res[] = $r;
 }
 return $res; }?>
```

- Now create create controller file (control.php) and save in seprate folder “**control**”.
- In that we have to include model.php filr from model folder.
- Than we have to create object of model class which we create in model.php and then we have to take request of form element and call function for insert data from model file.

```
<?php
session_start();
include 'model.php';

$obj = new model();

$con = $obj->selectall($c, 'country');

//insert

if (isset($_POST['submit'])) {
 $u = $_POST['uname'];
 $p = $_POST['pass'];
 $country = $_POST['country'];

 $data = array('uname' => $u, 'pass' => $p, 'country' => $country);
 $obj->add_data($c, 'user', $data);
 header('location:login.php');
}

//view record

//$view = $obj->selectall($c, 'user');

//view record with join

$view = $obj->sel_join($c,'user','country','user.country = country.cid');

//delete record
```

```
if (isset($_GET['del_id'])) {
 $d = $_GET['del_id'];
 $id = array('uid' => $d);
 $obj->del_data($c, 'user', $id);
 header('location:show.php');
}

//edit data

if (isset($_REQUEST['edit_id'])) {
 $ed = $_GET['edit_id'];
 $id = array('uid' => $ed);
 $res = $obj->selectwhere($c, 'user', $id);
 $rw = $res->fetch_object();

 if (isset($_POST['update'])) {
 $u = $_POST['uname'];
 $p = $_POST['pass'];
 $country = $_POST['country'];

 $data = array('uname' => $u, 'pass' => $p, 'country' => $country);
 $obj->update_data($c, 'user', $data, $id);
 header('location:show.php');
 }
}

//login
if (isset($_REQUEST['login'])) {
 $u = $_POST['uname'];
 $p = $_POST['pass'];
 $where=array("uname"=>$u,"pass"=>$p);
 $res = $obj->selectwhere($c, 'user', $where);
 $rw = $res->num_rows;
 if($rw==1)
 {
 $fet=$res->fetch_object();
 $_SESSION["uid"]=$fet->uid;
 $_SESSION["uname"]=$fet->uname;
 header("location:show.php");
 }
}
?>
```

- After this code, include control.php file into top of reg.php file. Thn run your application.

## Sessions and Cookies in PHP

- A **session** refers to all the connections that a single client might make to a server in the course of viewing any pages associated with a given application.
- Sessions are specific to both the individual user and the application.
- As a result, every user of an application has a separate session and has access to a separate set of session variables.
- The default time-out for session variables is 20 minutes. You can change the default time-out on the Memory Variables page of the ColdFusion MX Administrator Server tab.
- Before you can begin storing user information in your PHP session, you must first start the session.
- When you start a session, it must be at the very beginning of your code, before any HTML or text is sent.
- Below is a simple script that you should place at the beginning of your PHP code to start up a PHP session.

### Example:

First of all we create on file using following code and save as (file1.php)

#### Code:-

```
<form method="post">
 <table align="center" border="1">
 <tr>
 <td>User Name</td>
 <td><input type="text" name="uname" /></td>
 </tr>
 <tr>
 <td align="center" colspan="2">
 <input type="submit" name="submit" value="Insert" />
 </td>
 </tr>
 </table>
</form>
```

In this file we create session using php code as par following.

#### Code:-

```
<?php
session_start();

if(isset($_REQUEST['submit']))
{
 $u = $_REQUEST['uname'];

 $_SESSION['name'] = $u;
 header("location:file2.php");
}
?>
```

- In above code, First of all we start session using “session\_start()” function.
- Than we get request of button and username textbox.
- And create session variable using “\$\_SESSION” and send information of textbox to another file (file2.php).
- Now in file2.php we have to get value of textbox which we create in file1. So, following code is done.

Code:-

```
<?php
session_start();
$n = $_SESSION['name'];
echo "Welcome: ".$n;
?>
```

Output

Welcome: pratik

- For example. In (file2.php) file we create one link “logout” and if we click on this link session will be destroy and page will be redirect to (file1.php).
- For this we have to add following code in file2.php.

Code:-

```
Logout
```

Now we have to create “signout.php” file and do following code for session destroy.

Code:-

```
<?php
session_start();
unset($_SESSION['name']);
session_destroy();
header("location:file1.php");
?>
```

## Cookies in PHP

A message given to a Web browser by a Web server. The browser stores the message in a text file. The message is then sent back to the server each time the browser requests a page from the server. A cookie is often used to identify a user

### What is a Cookie?

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

### How to Create a Cookie?

- The setcookie() function is used to set a cookie.

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

### Syntax

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

### Example:

- First of all we have to create on php file (c1.php). In that we write following code.

```
<form method="post">
 <table align="center" border="1">
 <tr>
 <td>User Name</td>
 <td><input type="text" name="uname" /></td>
 </tr>
 <tr>
 <td align="center" colspan="2">
 <input type="submit" name="submit" value="Insert" />
 </td>
 </tr>
 </table>
</form>
```

- Now add following code at top of c1.php

```
Code:- <?php
if(isset($_REQUEST['submit']))
{
 $u = $_REQUEST['uname'];
 setcookie("name",$u,time()+10);
 header("location:c2.php");
}
?>
```

- In above code we get request of username and stored its value in cookie.
- So, We create cookie by using setcookie function. And redirect to c2.php file and get the value of uname. After 20 second cookie will destroy.

```
Code:- <?php
$n = $_COOKIE['name'];
echo "Welcome: " . $n;
?>
```

## Output

Welcome: pratik

## Files and Directory Access

In file and directory access first we create code for upload any file or image in one folder with storing path in database.

- First of all we create database and in that we have to create one table with two field imageid and imagepath

**Database Name: db\_image**

**Table Name: tbl\_image (image\_id (pk,auto\_increment), image\_path (varchar(100)))**

- Now create following code file uploading. First we create form (upload.php.)

```
<form method="post" enctype="multipart/form-data">
 <table align="center" border="1">
 <tr>
 <td>Seelct File</td>
 <td><input type="file" name="file" /></td>
 </tr>
 <tr>
 <td align="center" colspan="2">
 <input type="submit" name="submit" value="Insert" />
 </td>
 </tr>
 </table>
</form>
```

- Now create one folder “upload” where you save your upload.php file.
- Now create following code for database connection and file uploading at top of upload.php file.

```
<?php
mysql_connect("localhost","root","");
mysql_select_db("db_image");

if(isset($_REQUEST['submit']))
{
 if(((($_FILES['file']['type'] == "image/gif") || ($_FILES['file']['type'] == "image/jpeg") ||
($_FILES['file']['type'] == "image/pjpeg")) && ($_FILES['file']['size'] < 2000000))
 {
 if($_FILES['file']['error'] > 0)
 {
 echo "Return Code:" . $_FILES['file']['error'] . "
";
 }
 else
 {
 echo "Upload " . $_FILES['file']['name'] . "
";
 echo "Type " . $_FILES['file']['type'] . "
";
 echo "Size " . ($_FILES['file']['size']/1024) . "KB
";
 echo "TempFile " . $_FILES['file']['tmp_name'] . "
";

 if(file_exists("upload/" . $_FILES['file']['name']))
 {
 echo $_FILES['file']['name'] . "
Already Exists...
";
 }
 else
 {
 move_uploaded_file($_FILES['file']['tmp_name'], "upload/" . $_FILES['file']['name']);
 $i = "upload/" . $_FILES['file']['name'];
 $ins = "insert into tbl_image (image_path) values ('$i')";
 }
 }
 }
}
```

```
 }
 }
 else
 {
 echo "invalid file.....";
 }
}
?>
```

- In above code we upload image only “gif,jpeg,pjpeg” type with less than 200kb size.
- And also check, if the file already exists then show error message “file already exists”.

## File Handling

File handling is a very important feature in PHP. Using these concepts you can read, write, append file using PHP code. For file handling we have to use `fopen()`, `fclose()`, `fwrite()`, `fgets()` etc. Here we define some mode which use in `fopen()` function.

- r Read only. Starts at the beginning of the file
- r+ Read/Write. Starts at the beginning of the file
- w Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
- w+ Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
- a Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
- a+ Read/Append. Preserves file content by writing to the end of the file
- x Write only. Creates a new file. Returns FALSE and an error if file already exists
- x+ Read/Write. Creates a new file. Returns FALSE and an error if file already exists

## Example of create text file using PHP code.

### Code:-

```
<?php
$myfile = "test.txt";
$f = fopen($myfile,'w');
$str = "Pratik";
fwrite($f,$str);
fclose($f);

echo "file created...";
?>
```

- In above example, we create one text file `test.txt` and write some text in that. If we run this file we get `test.txt` file in our folder with text.

## Example of read data from text file using PHP code.

### Code:-

```
<?php
$file = fopen("welcome.txt","r") or exit("unable to open file");
while(!feof($file))
{

 echo fgets($file) . "
";
}
fclose($file);
?>
```

- In above example, we read the data of welcome.txt file using fread function.
- If we run this code we get the data of welcome.txt file.

## Handling e-mail

PHP allows you to send e-mails directly from a script.

The PHP mail() Function

The PHP mail() function is used to send emails from inside a script.

### Syntax

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. Note: 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 sendmail program

### Example:

First all we create form using following for send e-mail and save as mail.php.

```
<form method="post">
<table align="center" border="1">
<tr>
<td>Email</td>
<td><input type="text" name="email"></td>
</tr>
<tr>
<td>Subject</td>
<td><input type="text" name="subject"></td>
</tr>
<tr>
<td>Message</td>
<td><textarea cols="20" rows="10" name="message"></textarea></td>
```



```
</tr>
<tr>
<td><input type="submit" name="submit" value="submit"></td>
</tr>
</table></form>
```

Now, we have to add following code at the top of mail.php file

```
<?php
if(isset($_REQUEST['email']))
{
 $email = $_REQUEST['email'];
 $subject = $_REQUEST['subject'];
 $message = $_REQUEST['message'];

 mail("prati@gmail.com", "Subject:$subject", $message, "From:$email");
 echo "Thank you.....";
}
?>
```

- In above example, we get rebuset of different input types and use that in mail() function.
- If you run this file you get the message "Thank you". But you can see one warning.
- Because from localhost we cannot send mail. For that we have to change some setting in php.ini file.

## PHP Secure E-mails

- The problem with the code above is that unauthorized users can insert data into the mail headers via the input form.
- What happens if the user adds the following text to the email input field in the form?  
someone@example.com%0ACc:person2@example.com  
%0ABcc:person3@example.com,person3@example.com,  
anotherperson4@example.com,person5@example.com  
%0ABTo:person6@example.com
- The mail() function puts the text above into the mail headers as usual, and now the header has an extra Cc:, Bcc:, and To: field. When the user clicks the submit button, the e-mail will be sent to all of the addresses above!

## PHP Stopping E-mail Injections

- The best way to stop e-mail injections is to validate the input.
- The code below is the same as in the previous chapter, but now we have added an input validator that checks the email field in the form:  
<form action="" method="post">  
<?php  
function spamcheck(\$fld)  
{

```
$field=filter_var($fld,FILTER_SANITIZE_EMAIL);

if(filter_var($field,FILTER_VALIDATE_EMAIL))
{
 return true;
}
else
{
 return false;
}
}

if(isset($_REQUEST['email']))
{
 $mailcheck=spamcheck($_REQUEST['email']);

 if($mailcheck==false)
 {
 echo "Invalid email address";
 }
 else
 {
 $email=$_REQUEST['email'];
 $sub=$_REQUEST['sub'];
 $msg=$_REQUEST['msg'];
 mail("taher@gmail.com",$sub,$msg,"From: $email")or die("ERROR");
 echo "Thank you for using services";
 }
}
else
{
 ?>
 Your email add: <input type="text" name="email" />

 Subject : <input type="text" name="sub" />

 Message : <input type="text" name="msg" />

 <input type="submit" value="send" />
 <?php
}
?>
</form>
```

## JAVA Script

### What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages

- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

## What can a JavaScript do?

- **JavaScript gives HTML designers a programming tool** - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- **JavaScript can put dynamic text into an HTML page** - A JavaScript statement like this: `document.write("<h1>" + name + "</h1>")` can write a variable text into an HTML page
- **JavaScript can react to events** - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements** - A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data** - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer.

## How to implement JavaScript to an HTML page

- You can link to outside files (with the file extension .js), or write blocks of code right into your HTML documents with the `<script>` tag.
- So, the `<script type="text/javascript">` and `</script>` tells where the JavaScript starts and ends.
- For example, we have to create one form using following code and create script display alert box when we click on button.

### Example:

Code:-

```
<html>
<head>
<link rel="shortcut icon" href="">
<script type="text/javascript">
 function showmsg(){ alert("Javascript!");
 }
</script>
</head>
<body>
```

```
<form name="frm">
<input type="button" value="Click me" onClick="showmsg()">
</form>
</body>
</html>
```

## Form Validation

- Any interactive web site has form input -a place where the users input different kind of information. This data is passed to script, or some other technology and if the data contains an error, there will be a delay before the information travels over the Internet to the server, is examined on the server, and then returns to the user along with an error message.
- If you run a validation of the user's form input before the form is submitted, there will be no wait time and redundant load on the server. "Bad data" are already filtered out when input is passed to the server-based program.
- Client side form validation usually done with JavaScript. For the majority of your users, JavaScript form validation will save a lot of time up front, but double-checking the data on the server remains necessary, in case the user has turned JavaScript off.

## Events

- By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.
- Every element on a web page has certain events which can trigger JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

## List of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke
- onLoad and onUnload
- onFocus, onBlur and onChange

## Example of mouse hover and mouse out event

**Code:-**

```
<html>
<head>
<link rel="shortcut icon" href="">
</head>
<body>
<a href="#" onMouseOver="style.color='red'"
onMouseOut="style.color='blue'">Hello
</body>
</html>
```

## Output

[Hello](#)

In above example, if you take your mouse pointer on link tahn color will be chage to “red” and if remove mouse pointer from link, color will be change red to blue.

## Example of onchange event

### Code:-

```
<head>
<script type="text/javascript">
 function change()
 {
 var x = document.getElementById("name");
 x.value = x.value.toUpperCase();
 }
</script>
</head>
<body>
Enter Your Name: <input type="text" id="name" onChange="change()">
</body>
```

In above example, If enetr any text in textbox and enter “tab” button, name will be automatically convert in upparcase.

## Example of onfocus event

### Code:-

```
<html>
<head>
<script type="text/javascript">
 function set(x)
 {
 document.getElementById(x).style.background="red";
 }
</script>
</head>
<body>
Enter Your Name: <input type="text" id="name" onFocus="set(this.id)">
</body>
</html>
```

In above example, if we focus on the textbox the color will be change

## Example of form validation with regular expression.

- First of all we create one registration form using following code.

```
<form method="post" name="frm" onSubmit="return valid()">
 Enate Name:<input type="text" name="name" onBlur="return allLetter()">

 Enate LastName:<input type="text" name="lname">

 Enate Email:<input type="text" name="email" onBlur="return ValidateEmail()">

 <input type="submit" name="submit" value="submit">
</form>
```

- Now we create different function for name, lname and Email.
- We have to create function for valid email address, name can access only character.
- Use Following code is for access only latter.
- For that we have to use “/^ [A-Za-z]+\$/” expression and match().
- Add following code in script tag and call on onblur event of Name.

Code:-

```
function allLetter(name)
{
 var letters = /^ [A-Za-z]+$/;
 if(document.frm.name.value.match(letters))
 {
 return true;
 }
 else
 {
 alert('Username must have alphabet characters only');
 name.focus();
 return false;
 }
}
```

- Add following code for check Email address is valid or not and call this function on blur event of Email text box.

Code:-

```
function ValidateEmail(email)
{
 var mailformat = /^ \w+ ([\. -] ? \w+) * @ \w+ ([\. -] ? \w+) * (\. \w { 2, 3 }) + $ /;
 if(document.frm.email.value.match(mailformat))
 {
 return true;
 }
 else
 {
 alert("You have entered an invalid email address!");
 email.focus();
 return false;
 }
}
```

Now add following code for checking all validation and call on onsubmit event of form tag.

```
function valid()
{
 var letters = /^ [A-Za-z]+$/;
 var mailformat = /^ \w+ ([\. -] ? \w+) * @ \w+ ([\. -] ? \w+) * (\. \w { 2, 3 }) + $ /;

 if(document.frm.name.value == "")
 {
```

```
document.frm.name.style.background = 'yellow';
alert('Please fil First Name');
return false;
}
if((document.frm.name.value.length < 3) && (document.frm.name.value.length >
15))
{
 alert('Invalid Length');
 return false;
}
//valid email
if(document.frm.email.value.match(mailformat))
{
 return true;
}
else
{
 alert("You have entered an invalid email address!");
 email.focus();
 return false;
}
//valid name (only character)
if(document.frm.name.value.match(letters))
{
 return true;
}
else
{
 alert('Username must have alphabet characters only');
 name.focus();
 return false;
}
}
```

## XML

XML stands for EXtensible Markup Language, which became a W3C Recommendation on 10. February 1998.

XML is a markup language which is like HTML. XML and HTML both use tags.

### Difference between html and XML

- HTML was designed for how to display data. And XML was designed for how to store data.
- HTML tags are predefined. But XML tags are not predefined.
- You must define your own tags Before you learn XML, you should have a basic understanding of HTML.

## Syntax

```
<?xml version="1.0"?>
<note>
 <to>Tove</to>
 <from>Jani</from>
 <heading>Reminder</heading>
 <body>Don't forget me this weekend!</body>
</note>
```

### Above example

- The first line is the XML declaration. It defines the XML version (1.0) and the encoding used (ISO-8859-1 = Latin-1/West European character set).
- The next line describes the **root element** of the document (like saying: "this document is a note"):
- The next 4 lines describe 4 **child elements** of the root (to, from, heading, and body):
- And finally the last line defines the end of the root element:

### XML Naming Rules

- Names can contain letters, numbers, and other characters
- Names cannot start with a number or punctuation character
- Names cannot start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces
- 

```
<?php
mysql_connect("localhost","root","");
mysql_select_db("regdb");
$myFile = "country.xml";
$fhh = fopen($myFile, 'w') or die("can't open file");
$rss_txt = '<?xml version="1.0" encoding="utf-8"?>';
$rss_txt .= "<rss version='2.0'>";
$rss_txt .= '<channel>';
 $query = mysql_query("select * from country");
 while($values_query = mysql_fetch_assoc($query))
 {
 $rss_txt .= '<country>';
 $rss_txt .= '<cid>' . $values_query['cid'] . '</cid>';
 $rss_txt .= '<cname>' . $values_query['cname'] . '</cname>';
 $rss_txt .= '</country>';
 }
$rss_txt .= '</channel>';
$rss_txt .= '</rss>';
fwrite($fhh, $rss_txt);
fclose($fhh);
?>
```

**Note:-** before use this code first create a database and create a one country table (file name cid and cname)



## Read XML file in PHP

```
<!DOCTYPE html>
<html>
<body>
<?php
$xml=simplexml_load_file("country.xml");
print_r($xml);
?>
</body>
</html>
```

## AJAX

- **Ajax** (shorthand for asynchronous JavaScript + XML) is a group of interrelated web development techniques used on the client-side to create interactive web applications.
- With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page.
- The use of Ajax techniques has led to an increase in interactive or dynamic interfaces on web pages and better quality of Web services due to the asynchronous mode.
- Data is usually retrieved using the *XMLHttpRequest* object.
- Despite the name, the use of XML is not actually required, nor do the requests need to be asynchronous.
- **XMLHttpRequest (XHR)** is a DOM API that can be used inside a web browser scripting language, such as JavaScript, to send an HTTP or an HTTPS request directly to a web server and load the server response data directly back into the scripting language.
- Once the data is within the scripting language, it is available as both an XML document, if the response was valid XML markup, and as plain text.
- The XML data can be used to manipulate the currently active document in the browser window without the need of the client loading a new web page document.
- Plain text data can be evaluated within the scripting language to manipulate the document, too; in the example of JavaScript, the plain text may be formatted as JSON by the web server and evaluated within JavaScript to create an object of data for use on the current DOM.
- XMLHttpRequest has an important role in the AJAX web development technique. It is currently used by many websites to implement responsive and dynamic web applications. Examples of these web applications include Gmail, Google Maps, Bing Maps, the MapQuest dynamic map interface, Facebook, and others.

### AJAX Uses XML And HTTP Requests

- A traditional web application will submit input (using an HTML form) to a web server. After the web server has processed the data, it will return a completely new web page to the user.
- Because the server returns a new web page each time the user submits input, traditional web applications often run slowly and tend to be less user friendly.

- With AJAX, web applications can send and retrieve data without reloading the whole web page. This is done by sending HTTP requests to the server (behind the scenes), and by modifying only parts of the web page using JavaScript when the server returns data.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- The XMLHttpRequest object makes AJAX possible.

## The XMLHttpRequest

- The XMLHttpRequest object is the key to AJAX.
- It has been available ever since Internet Explorer 5.5 was released in July 2000, but not fully discovered before people started to talk about AJAX and Web 2.0 in 2005.

## Creating An XMLHttpRequest

- Different browsers use different methods to create an XMLHttpRequest object.
- Internet Explorer uses an **ActiveXObject**.
- Other browsers use a built in JavaScript object called **XMLHttpRequest**.
- Here is the simplest code you can use to overcome this problem:

## Ready State Property of AJAX

State	Description
0	request not initialized
1	server connection established
2	request received
3	processing request
4	request finished and response is ready

- Now we create an example of AJAX.
- First of all create one database, in that create two tables country and state. Then insert some data in that tables.
- Now give the relationship between them.
- **DatabaseName: ajax**
- **Table: country (cid int primary key auto\_increment, cname varchar(50))**
- **Table: state (sid int primary key auto\_increment, cid int index, sname varchar(50))**
- Now create a file (ajax.php) using following code. And add database connection code on top of this code. And fetch country from country table.

## ajax.php

```
<?php
$con = new mysqli("localhost","root","","jquery");

?>
<html>
<head>
```

```
<TITLE>jQuery DropDown List - Countries and States</TITLE>
<head>
<style>
body{width:610px;}
.frmDronpDown {border: 1px solid #F0F0F0;background-color:#C8EEFD;margin: 2px 0px;padding:40px;}
.demoInputBox {padding: 10px;border: #F0F0F0 1px solid;border-radius: 4px;background-color:
#FFF;width: 50%;}
.row{padding-bottom:15px;}
</style>
<script src="jquery.js" type="text/javascript"></script>
<script>
function getState(val) {
 $.ajax({
 type: "POST",
 url: "get_state.php",
 data:'country_id='+val,
 success: function(data){
 $("#state-list").html(data);
 }
 });
}

</script>
</head>
<body>
<div class="frmDronpDown">
<div class="row">
<label>Country:</label>

<select name="country" id="country-list" class="demoInputBox" onChange="getState(this.value);">
<option value="">Select Country</option>
<?php
 $sql ="select * from country";
 $q =$con->query($sql);
 while($f=$q->fetch_object())
 {
 ?>
 <option value="<?php echo $f->cid; ?>"><?php echo $f->cname; ?></option>
 <?php
 }
 ?>
</select>
</div>
<div class="row">
<label>State:</label>

<select name="state" id="state-list" class="demoInputBox">
<option value="">Select State</option>
</select>
```

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

## getdata.php

```
<?php
$con=new mysqli("localhost","root","","jquery");
if(isset($_POST["country_id"]))
{
 $query ="select * from state where cid = " . $_POST["country_id"] . " ";
 $q = $con->query($query);

 ?>
 <option value="">Select State</option>
 <?php
 while($res=$q->fetch_object())
 {
 ?>
 <option value="<?php echo $res->sid; ?>"><?php echo $res->sname; ?></option>
 <?php
 }
 }
 ?>
```

## Module – 8[CodeIgniter]

### What is CodeIgniter?

- Open Source PHP web Apps Framework for use in building dynamic web sites based on the popular Model – View –Controller development pattern with PHP.
- Free for use and Develop
- CodeIgniter is faster and lighter compared to other PHP frameworks
- CodeIgniter is the foundation of the new version of ExpressionEngine CMS developed by EllisLab.

### History

- In 2001, **Rick Ellis** started work on a blogging engine for his clients.
- The first installation of pMachine was for Nancy Sinatra (Frank Sinatra's daughter).
- In 2002, pMachine was released.
- Rick Ellis starts developing software full-time.
- ExpressionEngine succeeds pMachine and is a more fully feature complete Content Management System.
- ExpressionEngine sees great success within the designer market, on their personal blogs
- EllisLab pulls the core code out of ExpressionEngine and refactors into a framework.
- **Goal** : "Its goal is to enable [developers] to develop projects much faster than...writing code from scratch, by providing a rich set of libraries for commonly needed tasks, as well as a simple interface and logical structure to access these libraries."
- CodeIgniter's source code is maintained at **GitHub Inc.**

### Key Feature of CodeIgniter

	Run 1	Run 2	Run 3	Run 4	Average
<b>CakePHP</b>	36.0	36.1	36.1	36.2	<b>36.1</b>
<b>CodeIgniter</b>	383.3	377.9	371.8	385.2	<b>379.5</b>
<b>Zend Framework</b>	129.2	128.5	129.0	128.9	<b>128.9</b>

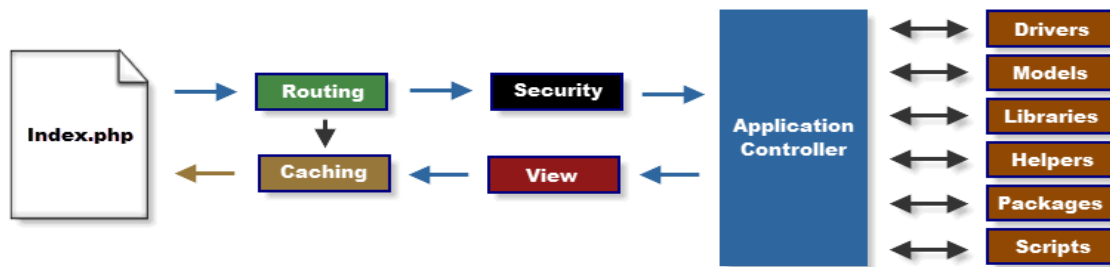
- 10.5x faster than CakePHP, 2.9x faster than Zend Framework (similar configurations / similar functionality)
- **SEO friendly URLs**
- CodeIgniter automatically generates SEO friendly URLs

- This is easily overridden: regular expression pattern matching to point to controller/method and pass variables
- Infinitely extensible  

```
$this->load->library('upload', $config);
```

## Application Flow Chart

The following graphic illustrates how data flows throughout the system:



1. The index.php serves as the front controller, initializing the base resources needed to run CodeIgniter.
2. The Router examines the HTTP request to determine what should be done with it.
3. If a cache file exists, it is sent directly to the browser, bypassing the normal system execution.
4. Security. Before the application controller is loaded, the HTTP request and any user submitted data is filtered for security.
5. The Controller loads the model, core libraries, helpers, and any other resources needed to process the specific request.
6. The finalized View is rendered then sent to the web browser to be seen. If caching is enabled, the view is cached first so that on subsequent requests it can be served.

## Creating Simple Application in CodeIgniter

### Creating View

- A View is a simple php file that is named in a way that can be associated with your page.
- First of all go to **application/views/** folder and create home.php

```
<html>
 <head>
 <title>Tops Technologies</title>
 </head>
 <body>
 Welcome To TOPS Technologies
 </body>
</html>
```

## Creating Controller

- A Controller is a class file that is named in a way that can be associated with your page.
- First of all go to application/controllers/ folder and create mycontroller.php
- Then go to in your controller's method and load the view like :

```
<?php
class Mycontroller extends CI_Controller
{
 public function index()
 {
 $this->load->view('home');
 }
}
?>
```

- CI\_Controller class is the inbuilt class which handles the Controller's functionalities
- Then go to browser and load the URL like:  
**localhost/index.php/mycontroller /index**

## URI Routing(application\config\routes.php)

- This file lets you re-map URI requests to specific controller functions.
- Typically there is a one-to-one relationship between a URL string and its corresponding Controller class/method. The segments in a URL normally follow this pattern:  
example.com/class/method/id/
- If you want to set your controller as default controller then do the change in file like:  
\$route['default\_controller'] = "your\_controller\_name";

**Ex : \$route['default\_controller'] = "mycontroller";**

## Libraries and Helpers

### Libraries

- Library can be considered as a collection of tools that can be used to assist the process. CodeIgniter has provided a lot of libraries that can be used mainly directly.
- Library is essentially a class that is placed in the folder system / libraries or application / libraries
- Library located in the system folder is a default library of CodeIgniter that by default in the given prefix CI\_. For homemade library should be placed in the folder application / libraries

### Helpers

- Helpers like the name suggests will help you build an application with a specific task.
- Unlike a library, not a helper Object Oriented but procedural form.
- Each helper contains one or more functions, each focusing on a specific task that no dependence with other functions
- Helper is a collection of functions placed in the folder system / helpers or applications / helpers.

### Loader Class

- In the CodeIgniter, there is a CI\_Loader class that function to load all the source code is needed. It's maybe library, models, databases, helpers, config, etc.
- You can see the code of CI\_Loader in systems/core/Loader.php.
- it will load a class and instance it, the object of library/helper will be create as properties in \$this object, not returned as new object.

## CI Libraries

By default CodeIgniter has provided a library that can be used directly. As for the library that are available

## Auto-loading Resources

- CodeIgniter comes with an "Auto-load" feature that permits libraries, helpers, and models to be initialized automatically every time the system runs. If you need certain resources globally throughout your application you should consider auto-loading them for convenience.
- To autoload resources, open the **application/config/autoload.php** file and add the item you want loaded to the autoload array.

Ex :

```
$autoload['libraries'] = array('form_validation','database','session');

$autoload['helper'] = array('url','form','cookie','html');
```

## Config Class

- This library functions to retrieve the data in the configuration file. This library is loaded automatically by CodeIgniter.
- By default, CodeIgniter has one primary config file, located at application/config/config.php there is config items are stored in an array called *\$config*.
- We can set our index page name, Languages, Query String Enable/Disable, Base url etc... from config.php file

Like:

```
$config['index_page'] = 'index.php';
```

## Database Class

Library databases are used to manipulate and obtain data from a database system. By default database that is supported by CodeIgniter is MySQL, MSSQL, Oracle, Postgres.

## Database Configuration

- First of all we set the configuration of the database like Host Name, User name, Password and Database Name in **application/config/database.php**

Like :



```
$db['default']['hostname'] = "localhost";
$db['default']['username'] = "root";
$db['default']['password'] = "";
$db['default']['database'] = "Your Database";
```

## Database Functions

The most amazing thing about CodeIgniter is that you do not need to write **"Database Query"**. As it gives so many functions for this

### **\$this->db->query('Query');**

- It will return Database Result set when READ type query and return TRUE/FALSE when WRITE type query

```
$this->db-> query('Query');
```

- It will return only TRUE/FALSE on success or failure not return a database result set

### **\$this->db->insert();**

- This function used to insert data into table and returns true/false.
- Syntax:

```
$this->db->insert('Table Name', Associative Array);
```

### **\$this->db->get();**

- This function used to select records from the table and returns result set.
- Syntax:

```
$query = $this->db->get('Table Name');
```

### **\$this->db->get\_where();**

- This function used to fetch particular row from the table. It means we can use "where" clause in the second parameter.
- Syntax:

```
$query = $this->db->get_where(Table,WHERE);
```

### **\$this->db->update();**

- Used to update records in database table

Syntax:

```
$this->db->update('table', array/object, where);
```

### **\$this->db->delete();**

- Used to delete records from database table

Syntax:

```
$this->db-> delete('table', where);
```

## Insert in CodeIgniter

### **Model(mymodel.php)**

```
<?php
class Mymodel extends CI_Model
{
 public function ins($table,$data)
 {
 $this->db->insert($table,$data); } ?>
```

## View(reg.php)

```
<tr>
 <td>Username</td>
 <td><input type="text" name="un" /></td>
</tr>

<tr>
 <td>Password</td>
 <td><input type="password" name="ps" /></td>
</tr>

<tr>
 <td><input type="submit" name="reg" value="REG" /></td>
</tr>
```

## Controller(mycontroller.php)

```
<?php
class Mycontroller extends CI_Controller
{
 public function index()
 {
 $this->load->model('mymodel');
 $this->load->view('reg');
 if($this->input->post('reg')) //Click on reg button
 {
 $un = $this->input->post('un');
 $ps = $this->input->post('ps');
 $data = array("uname"=>$un,"password"=>$ps);

 $this->mymodel->ins('user_mst',$data); //Method from model to insert data
 }
 }
}
```

## Fetch data in CodeIgniter

### Model(mymodel.php)

```
<?php
class Mymodel extends CI_Model
{
 public function shw($table)
 {
 $sel = $this->db->get($table);//select * from table;
 return $sel->result_array();
 }
}
}??>
```

### Controller(home.php)

```
public function home()
{
 $this->load->model('mymodel');
 $view['data'] = $this->mymodel->shw('user_mst');
 $this->load->view('home',$view);
}
```

- Here **\$view** will save data fetch from database so when we load view page we have to load same variable.

### View(home.php)

```
<table align="center" border="1">
<tr>
 <th>Username</th>
 <th>Password</th>
</tr>

<?php
foreach($data as $dt)
{
 ?>
 <tr>
 <td><?php echo $dt['uname']?></td>
 <td><?php echo $dt['password']?></td>
 </tr>
 <?php
 }
 ?>
</table>
```

## URI Class

- The URI Class provides functions that help you retrieve information from your URI strings. If you use URI routing, you can also retrieve information about the re-routed segments.
- This class is initialized automatically by the system so there is no need to do it manually.

### **`$this->uri->segment(n)`**

- Permits you to retrieve a specific segment. Where *n* is the segment number you wish to retrieve. Segments are numbered from left to right

Ex :

**`http://localhost/Total_CI/Step4(Delete)/index.php/mycontroller/del/3`**

Here

- `$this->uri->segment(1) = mycontroller(Class Name)`
- `$this->uri->segment(2) = del(Function name)`
- `$this->uri->segment(3) = 3`

## Delete in CodeIgniter

### **Model(mymodel.php)**

```
public function del($table,$where)
{
 $this->db->delete($table,$where);
}
```

### **View(home.php)**

- Here code of delete link is given, consider \$dt from previous section for fetch data

```
<td><?php echo anchor("mycontroller/del/".$dt['uid'], "Delete")?></td>
```

- Anchor is used for <a> tag

### **Controller(mycontroller.php)**

- The first thing we need to consider while deleting data is URI Segment from link , we need ID(**unique identification for delete data**) so in home page where we have created delete link, we have defined that on uri segment 3
- And second segment will be the function name of delete in controller page
- So for delete controller code will be like : -

```
public function del()
{
 $this->load->model('mymodel');
 $id = $this->uri->segment(3);
 $where = array("uid"=>$id);
 $this->mymodel->del('user_mst',$where);
 redirect("mycontroller/home");
}
```

## Session Class

- The Session class permits you maintain a user's "state" and track their activity while they browse your site.
- The Session class stores session information for each user as serialized (and optionally encrypted) data in a cookie.
- It can also store the session data in a database table for added security, as this permits the session ID in the user's cookie to be matched against the stored session ID.
- The Session class does **not** utilize native PHP sessions. It generates its own session data, offering more flexibility for developers.
- Even if you are not using encrypted sessions, you must set an encryption key in your config file which is used to aid in preventing session data manipulation.

## Initializing a Session

- To initialize the Session class manually in your controller constructor, use the `$this->load->library` function:  
**`$this->load->library('session');`**
- Alternately you can initialize that in autoload.php also so you will do not need to initialize that all the time.

## Adding Custom Session Data

- To add your data to the session array involves passing an array containing your new data to this function:  
**`$this->session->set_userdata($array);`**

## Retrieving Session Data

- Any piece of information from the session array is available using the following function:  
**`$this->session->userdata('item');`**

## Removing Session Data

- Just as `set_userdata()` can be used to add information into a session, `unset_userdata()` can be used to remove it, by passing the session key  
**`$this->session->unset_userdata($array);`**

## Login with Session

### Model(mymodel.php)

```
public function shw_whr($table,$where)
{
 $sel = $this->db->get_where($table,$where);
 return $sel->result_array();
}
```

### View(login.php)

```
<tr>
 <th align="left">Name</th>
 <td><input type="text" name="un" /></td>
</tr>

<tr>
 <th align="left">Password</th>
 <td><input type="password" name="ps" /></td>
</tr>

<tr>
 <td colspan="2" align="center">
 <input type="submit" name="lg" value="LOGIN" />
 </td>
</tr>
```

### Controller(mycontroller.php)

- First check if Username & Password is correct or not using following code
- Then if it is possible then store data using **`$this->session->set_userdata(session_array)`** and redirect it to new function as here it is mentioned as **`signin()`**
- Else load login page again

```
public function index()
{
 $this->load->model('mymodel');
 if($this->input->post('lg'))
 {
 $ln = $this->input->post('un');
 $lp = $this->input->post('ps');

 $where = array("uname"=>$ln,"password"=>$lp);
 $view = $this->mymodel->shw_whr('user_mst',$where);
 if(count($view)==1)
 {
 $ses = array("uid"=>$view[0]['uid'], "user_name"=>$ln, "log_in"=>TRUE);
 $this->session->set_userdata($ses);
 redirect("mycontroller/signin");
 }
 else
 {
 ?>
 <script type="text/javascript">
 alert("Wrong user name or Password");
 </script>
 <?php
 $this->load->view('login');
 }
 }

 else
 {
 $this->load->view('login');
 }
 }

 public function signin(){
 $this->load->model('mymodel');
 if($this->session->userdata('log_in')){
 $where = array("uid"=>$this->session->userdata('uid'));
 $data['ldata'] = $this->mymodel->shw_whr('user_mst',$where);
 $this->load->view('home',$data);
 }
 else
 {
 redirect('mycontroller','refresh');
 }
 }
}
```

## View(home.php)

- You can retrieve data of session on home page using \$ldata as we have load that with home page in sign in function
- You can do that like  

```
<p align="right"><?php echo $ldata[0]['uname'];?> || <?php echo
anchor("mycontroller/logout","Logout");?></p>
```
- For Destroying Session you need to create one more function in controller like given below

## Controller(mycontroller.php)

```
public function logout()
{
 $this->session->unset_userdata('uid');
 $this->session->unset_userdata('user_name');

 $this->session->sess_destroy();
 redirect('mycontroller','refresh');
}
```

## Form Validation Class

- CodeIgniter provides a comprehensive form validation class that helps minimize the amount of code
- Library Validation form is used to check the validity of the forms that is submitted by the user
- To set validation rules will use the *set\_rules()* function:

```
$this->form_validation->set_rules();
```

The above function takes three parameters as input:

- 1) The field name that you've given the form field.
- 2) Your Error Message.
- 3) The validation rules for this form field.

Like:

```
$this->form_validation->set_rules('username', 'Username', 'required');
```

## File Uploading Class

Uploading Library is used to upload files. This class is equipped with checking file type, and file size.

### Setting Configuration and Load Library

```
$config['upload_path'] = './uploads/';
$config['allowed_types'] = 'gif|jpg|png';
$config['max_size'] = '100';
$config['max_width'] = '1024';
$config['max_height'] = '768';
$this->load->library('upload', $config);
```



## Interview Question

1. What's PHP ?
2. Explain the ternary conditional operator in PHP?
3. What are the different functions in sorting an array?
4. How can we know the count/number of elements of an array?
5. How many ways we can pass the variable through the navigation between the pages?
6. How To Read the Entire File into a Single String?
7. What Is a Session?
8. What Is a Persistent Cookie?
9. How to set cookies?
10. How to reset/destroy a cookie
11. How To Turn On the Session Support?
12. What is meant by PEAR in php?
13. How can we know the number of days between two given dates using PHP?
14. What is the difference between \$message and \$\$message?
15. What does a special set of tags do in PHP?
16. How do you define a constant?
17. How To Write the FORM Tag Correctly for Uploading Files?
18. What are the differences between require and include, include\_once?
19. What is meant by urlencode and urldecode?
20. How To Get the Uploaded File Information in the Receiving Script?
21. What is the difference between mysql\_fetch\_object and mysql\_fetch\_array?
22. How can I execute a PHP script using command line?
23. I am trying to assign a variable the value of 0123, but it keeps coming up with a different number, what's the problem?
24. Would I use print "\$a dollars" or "{ \$a } dollars" to print out the amount of dollars in this example?
25. What are the different tables present in MySQL? Which type of table is generated when we are creating a table in the following syntax: create table employee(eno int(2),ename varchar(10))?
26. How To Create a Table?
27. How can we encrypt the username and password using PHP?
28. How do you pass a variable by value?

29. What is the functionality of the functions STRSTR() and STRISTR() ?
30. When are you supposed to use endif to end the conditional statement?
31. How can we send mail using JavaScript?
32. What is the functionality of the function strstr and stristr?
33. What is the difference between ereg\_replace() and eregi\_replace()?
34. How do I find out the number of parameters passed into function? ?
35. What is the purpose of the following files having extensions: frm, myd, and myi? What these files contain?
36. If the variable \$a is equal to 5 and variable \$b is equal to character a, what's the value of \$\$b?
37. Are objects passed by value or by reference?
38. What are the differences between DROP a table and TRUNCATE a table?
39. What are the differences between GET and POST methods in form submitting, give the case where we can use GET and we can use POST methods?
40. How do you call a constructor for a parent class?
41. What are the different types of errors in php ?
42. What's the special meaning of \_\_sleep and \_\_wakeup?
43. How can we submit a form without a submit button?
44. Would you initialize your strings with single quotes or double quotes?
45. How can we extract string 'abc.com' from a string http://info@abc.com using regular expression of php?
46. What is the difference between the functions unlink and unset?
47. How come the code works, but doesn't for two-dimensional array of mine?
48. How can we register the variables into a session?
49. What is the difference between characters \023 and \x23?
50. How can we submit form without a submit button?
51. How can we create a database using PHP and mysql?
52. How many ways we can retrieve the date in result set of mysql using php?
53. Can we use include ("abc.php") two times in a php page "makeit.php"?
54. For printing out strings, there are echo, print and printf. Explain the differences.

55. I am writing an application in PHP that outputs a printable version of driving directions. It contains some long sentences, and I am a neat freak, and would like to make sure that no line exceeds 50 characters. How do I accomplish that with PHP?
56. What's the output of the ucwords function in this example?
57. What's the difference between htmlentities() and htmlspecialchars()?
58. So if md5() generates the most secure hash, why would you ever use the less secure crc32() and sha1()?
59. How can we destroy the session, how can we unset the variable of a session?
60. How many values can the SET function of MySQL take?
61. How can we find the number of rows in a table using MySQL?
65. Use this for MySQL
66. What's the difference between md5(), crc32() and sha1() crypto on PHP?
67. How can we find the number of rows in a result set using PHP?
68. What is the difference between GROUP BY and ORDER BY in SQL?
69. What are the features and advantages of OBJECT ORIENTED PROGRAMMING?
70. What is the use of friend function?
71. How can we get second of the current time using date function?
72. What are the difference between abstract class and interface?