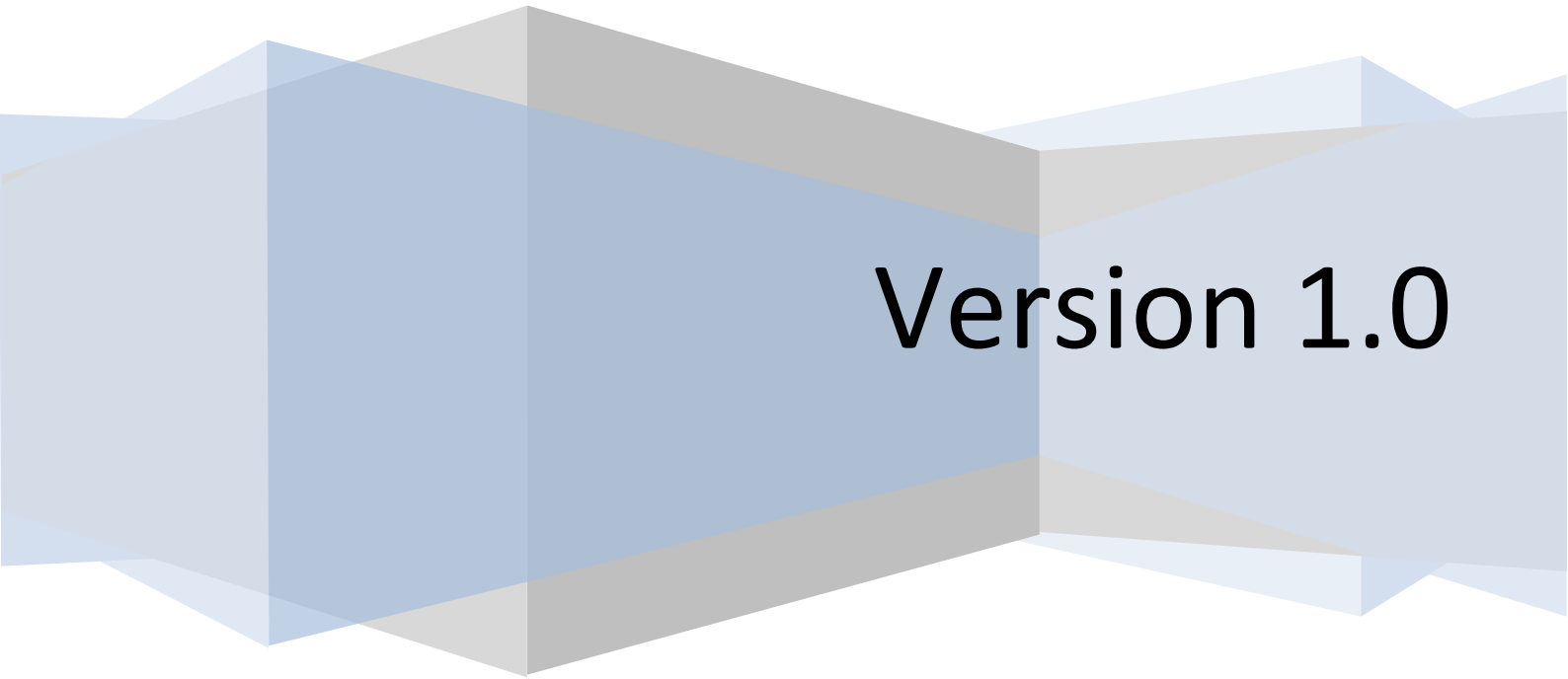
PHP Mini Framework

User Guide

Developed By : Bharat Parmar



Version 1.0

Introduction

PHP Mini Framework is developed for the PHP beginners, Students from the IT Institutes and all the Developers who want to develop the project in very short code and short time. PHP Mini Framework is very developed with a view to be useful to all. The Framework does not require any other Framework knowledge or High Level of Programming Experience. The Framework has been developed using the OOP concepts in PHP. The Framework includes HTML, CSS, Javascript and PHP scripting languages.

PHP Mini Framework includes the Demo Application. Developer can use the same demo application for their projects or can modify as per requirements. There is not any special standard required to use this Framework. Developer can use their own code and method to use the framework.

PHP Mini Framework Version 1.0 is in the beta version which will be extended with more modules and libraries very soon.

This framework is freeware. Anyone can use and modify as per the requirement. After modification to any file or database structure, the framework will not be responsible for any type of error occurred to the application.

In case of any error or confusion, you can contact me :

Bharat Parmar

Email : bharatparmar383@gmail.com

Mobile : +91 9687766553

Features

1) Easy installation

2) Light weight, Simple and Short Code

3) Core PHP with OOP based coding Structure.

4) Readymade CMS Pages, Contact Us Page, User Login and Register and Admin Panel with Advance Function.

5) MySQL Database Import, Export and Download from Admin Panel.

6) Easy to modify and re-usable. All the class files are extends the Main.class.php which includes all the main functions of the framework for the database access and many more.

7) Independent Code : You can use the Main.class.php file for any other live project instead of the whole framework. It requires minimum settings from the config.php file.

8) Bootstrap template based Frontend and Backend Themes. Easy to modify the Design/Layout as per the requirement.

9) Freeware : Anyone can use this framework and modify.

10) Free Support : If any error or confusion found about this framework, free support is available to you. Just send one mail to bharatparmar383@gmail.com

Requirement

1) PHP 3.0 or above

2) MySQL

Installation

Extract the zip file in your server directory.

Installation Process :

You can install this framework directly from the web browser. You need not to modify any file.

Step 1 : Create MySQL Database and User. Privilege the rights to the Created MySQL User for the Database.

Step 2 : Go to Installation Page. URL : YOUR_DIRECTORY_PATH/install

Step 3 : Follow the instruction given on the Installation Page.

Step 4 : After installation successfully done, you must need to remove/delete the install directory.

You will get all the necessary details about the Site URL, Admin URL and Admin Login Details after successfully installation.

NOTE : DO NOT CHANGE SECRET KEY ONCE ANY USER HAS BEEN CREATED.

Demo Application

PHP Mini Framework comes with Demo Application to understand it's structure as well as saves time for the project setup.

FILES & DIRECTORIES STRUCTURE :

A) Public Access : Login Does Not Require

B) User Access : User Login Require.

C) Admin Access : Admin Login Require.

A) Public Access :

 # Home Directory :

1) index.php : Index or Home page for the site.

2) aboutus.php : About Us Page. CMS Page.

3) contactus.php : Contact Us Page. Visitor of your site can send message to Admin.

4) login.php : Login page for the Registered User.

5) register.php : Registration for the new User.

6) config.php : Configuration file for the script.

 # "class" Directory :

1) Main.class.php : This is the mail class file for the script. All other classes will extends this class.

2) Frontend.class.php : This class will be used for the Frontend only. For the frontend module, You can extends this class. This class extends the class/Main.class.php Class. This class does not require User Login

3) User.class.php : User class is used for the User Registration, Login, Forgot Password and Profile Update. This class extends the class/Frontend.class.php Class.

If you want to add any new module to your site, You can create new class file for your module and you must need to extend class/Frontend.class.php.

You can create unlimited class and modify any class file on your own risk.

 # "css" Directory :

1) bootstrap.min.css : Bootstrap stylesheet.

 # "includes" Directory :

1) phpinfo.php : To check the PHP Configuration on your server.

2) header.php : Header Template file for the Frontend Theme.

3) footer.php : Footer Template file for the Frontend Theme.

 # "js" Directory :

1) customjquery.js : Custom JQuery functions for the frontend.

Other Necessary Javascript file for the frontend.

B) User Access :

 # "user" Directory :

1) profile.php : User Profile Page. This file can be access after User Login.

C) Admin Access :

1) login.php : Admin Login Page. (class/Admin.class.php)

2) index.php : Admin Dashboard Page. (class/Dashboard.class.php)

3) admin.php : Admin Profile Page. (class/Admin.class.php)

4) cmshomepage.php : CMS Pages Management. (class/CMS.class.php)

5) contactmessages.php : Contact Us page messages. (class/Contact.class.php)

6) database.php : Database Management. Admin Can Import/Export/Download the MySQL Database. (class/Database.class.php)

7) emailtemplate.php : Email Template Management. (class/CMS.class.php)

8) manageuser.php : User Management. (class/User.class.php)

9) settings.php : Site Settings Page (class/Admin.class.php)

 # "class" Directory :

1) Admin.class.php : Admin Login, Forgot Password, Admin Profile Site Settings etc methods included. This class extends Home_directory/class/Main.class.php

2) CMS.class.php : CMS Pages and Email Template Management related methods included. This class extends class/Admin.class.php

3) Contact.class.php : Contact Us Page Messages Management related methods included. This class extends class/Admin.class.php

4) Dashboard.class.php : Admin Dashboard for the statistics related methods included. This class extends class/Admin.class.php

5) Database.class.php : Database Import, Export and Download related methods included. This class extends class/Admin.class.php

6) User.class.php : User Management related methods included. This class extends class/Admin.class.php

 # "database" Directory :

All the exported MySQL Database from the Admin Panel will be stored here. It can be downloaded and imported from the Admin Panel.

 # "images" Directory :

All the Admin Panel Theme related Images stored here.

 # "includes" Directory :

1) header.php : Admin Panel Theme Header Template.

2) footer.php : Admin Panel Theme footer Template.

 # "lib" Directory :

Javascript and Jquery plugins which are used in Admin Panel has been stored here.
"admin.customjquery.js" is developed for the Admin Panel related custom functions.

 # "stylesheets" Directory :

Stylesheets (CSS) which are used in the Admin Panel theme and other plugins has been stored here.

Basic Knowledge

Before using the PHP Mini Framework, look at once for the Basic Knowledge of the framework.

Configuration :

The framework works on the below configuration which are set in the config.php file.

1) DATABASE DETAILS :

$hostname = DATABASE HOST NAME. eg : localhost

$username = DATABASE MySQL user name. eg : root

$password = DATABASE MySQL password. eg : 12345

$database = DATABASE MySQL Database name. eg : bharatcode

2) Pagination :

Record per Page : $_SESSION['pagerecords_limit'] = 20;

If you want to change by default record per page, you can change the value.

3) URL MANAGEMENT :

The framework comes with three different panel.

A) Public , B) User Panel, C) Admin Panel.

If you want to add any new directory or panel for your script, you must need to add that directory name in the "$panel_array" array.

$panel_array = array("install","admin","user");

Site URL will be automatically generate from the file accessing in the browser.

You can use the below URL in any file as below :

SITE_URL : Your site url (eg. http://localhost/bharatcode)

ADMIN_URL : Admin Panel URL. (eg. http://localhost/bharatcode/admin)

USER_URL : User Panel URL. (eg. http://localhost/bharatcode/user)

All these are set in the config.php file which are CONNSTANT.

4) Table Prefix :

If you have set the Table Prefix during the Installation, it will be set as TABLE_PREFIX. This is constant value and never changed. If you do no want to set the Table Prefix, leave it blank.

Note : If Table Prefix has been set, then all the tables must have that prefix. In the Database Record Methods, you never need to add the Table Prefix. But if you want to make your custom query, you need to use the full table name.

Example :

Table Prefix : bh_

Table 1 Name : bh_user_master

Table 2 Name : bh_contact_messages

if you want to retrive records from bh_user_master table using the class file, you can use as below :

Example :

$Main->GetRecord("user_master",$info_array);

Table Prefix "bh_" will be automatically added in the GetRecord method.

if you want to use your custom code or query, you can use as below :

mysql_query("select * from bh_user_master");

5) SECRET_KEY :

This key is used for the Password Encryption. Once the users created, it should never change. You can use alpha-numeric-symbols here.

Code Structure

Jquery Validation :

1) User Input

1.1 Alpahabets : Add "alpha" class to input. The input element will only accept alphabets only.

Example :

<input type="text" class="form-control alpha">

1.2 Numeric : Add "numeric" class to input. The input element will only accept numeric value only.

Example :

<input type="text" class="form-control numeric">

1.3 Alpha-Numeric : Add "alphanumeric" class to input. The input element will only accept alphabets and numeric only.

Example :

<input type="text" class="form-control alphanumeric">

1.4 Email : Add "email" class to input. The input element will only accept email address only.

Example :

<input type="text" class="form-control email">

You will find the code in below file :

File Path : js/customjquery.js

(INPUT VALIDATION ON KEYUP)

2) Blank Validation

To check the input value is blank or not, its very simple here. You just need to add new "title" attribute to the element for which you want to use the Blank Validation. The Validation will validates on form submit. If the value is blank of that element, it will be focused with RED border color and the "title" attribute's value will be added as Placeholder which display the error message.

Syntax : title="Please Enter First Name. It should not be blank."

Example :

<input type="text" class="form-control" id="firstname" name="firstname" title="Enter First Name" value="">

You will find the code in below file :

File Path : js/customjquery.js

(INPUT VALIDATION ON KEYUP)

You can check the login.php and register.php file for the both above validation code reference.

PHP Validation :

All required PHP validation has been added to the class/Main.class.php file which can be used in all pages.

validate() method will provide the validation like blank value, alpha, numeric, alpha-numeric, email and url.

This function require two parameters. 1) value of the input 2) validation type.

Syntax :

$Main->validate($parameter1, $parameter2);

$parameter1 = User Input

$parameter2 = require, numeric, alpha, alphanumeric, email and url.

Example :

$Main->validate("bharat383","numeric")

Return :

This method will return TRUE or FALSE. If value is correct as per the validation, it will return TRUE else FALSE.

1) Blank Validation :

$Main->validate("","require"); => FALSE.

$Main->validate("Bharat383","require"); => TRUE.

2) Numeric Validation :

$Main->validate("Bharat383","numeric"); => FALSE.

$Main->validate("123456","numeric"); => TRUE.

3) Alphabets only Validation :

$Main->validate("Bharat383","alpha"); => FALSE.

$Main->validate("Bharat","alpha"); => TRUE.

4) Alpha Numeric only Validation :

$Main->validate("Bharat383!@#$$%%","alphanumeric"); => FALSE.

$Main->validate("Bharat383","alphanumeric"); => TRUE.

$Main->validate("Bharat","alphanumeric"); => TRUE.

$Main->validate("383","alphanumeric"); => TRUE.

5) Email Validation :

$Main->validate("Bharat383","email"); => FALSE.

$Main->validate("bharat@","email"); => FALSE.

$Main->validate("bharat@domain","email"); => FALSE.

$Main->validate("bharat@domain.com","email"); => TRUE.

6) URL Validation :

$Main->validate("Bharat383","url"); => FALSE.

$Main->validate("bharat@","url"); => FALSE.

$Main->validate("www.domain","url"); => FALSE.

$Main->validate("www.domain.com","url"); => TRUE.

$Main->validate("http://www.domain.com","url"); => TRUE.

Class File

A) Main.class.php :

The class file is the main class file for the framework which has been extended by all other classes.

The Class having following global variables.

1) pagefilename : Current Page File Name.

2) sitedata : All the site settings values.

The Class includes following methods :

Basic Functions of the Framework :

1) RedirectPage($parameter1)

Description : This method will redired to the given URL or page. It will stop the PHP Code execution once redirect.

Parameters :

$parameter1 = URL or file name where you want to redirect page.

Return : No Return Method.

Syntax :

```
$Main->RedirectPage("aboutus.php");
```

Example :

```
$url = "user/profile.php";
$Main->RedirectPage($url);
```

2) AddLink()

Description : This method will provide Add Link for the record.

Parameters : Parameter not required.

Return : This method will return the URL for Add New Record for the Particular page/file.

Syntax :

$Main->AddLink();

Example :

<a href="<?php $Main->AddLink();">Add New

Output :

Add New

3) ViewLink($parameter1)

Description : This method will provide View Action Link for the record.

Parameters :

$parameter1 : Record Unique ID.

Return : This method will return the URL for View Action Record for the Particular Record.

Syntax :

$Main->ViewLink($id);

Example :

<a href="<?php $Main->ViewLink(1);">View Details

Output :

View Details

4) EditLink($parameter1)

Description : This method will return Edit Action Link for the record.

Parameters :

$parameter1 : Record Unique ID.

Return : This method will return the URL for Edit Action Record for the Particular Record.

Syntax :

$Main->EditLink($id);

Example :

<a href="<?php $Main->EditLink(1);">Edit

Output :

Edit

5) StatusLink($parameter1,$parameter2)

Description : This method will return Status Change Action Link for the record.

Parameters :

$parameter1 : Record Unique ID.

$parameter2 : Status that you want to set.

Return : This method will return the URL for Status Change Action Record for the Particular Record.

Syntax :

$Main->StatusLink($id,$status);

Example :

<a href="<?php $Main->StatusLink(1,'0');">Set Inactive

<a href="<?php $Main->StatusLink(1,'1');">Set Active

Output :

Inactive

Active

6) DeleteLink($parameter1)

Description : This method will return Delete Action Link for the record.

Parameters :

$parameter1 : Record Unique ID.

Return : This method will return the URL for Delete Action Record for the Particular Record.

Syntax :

$Main->DeleteLink($id);

Example :

<a href="<?php $Main->DeleteLink(1);">Delete This Record

Output :

Delete This Record

7) Validate($parameter1,$parameter2)

Syntax :

$Main->validate($parameter1, $parameter2);

$parameter1 = User Input

$parameter2 = require, numeric, alpha, alphanumeric, email and url.

Example :

$Main->validate("bharat383","numeric")

Return :

This method will return TRUE or FALSE. If value is correct as per the validation, it will return TRUE else FALSE.

1) Blank Validation :

$Main->validate("","require"); => FALSE.

$Main->validate("Bharat383","require"); => TRUE.

2) Numeric Validation :

$Main->validate("Bharat383","numeric"); => FALSE.

$Main->validate("123456","numeric"); => TRUE.

3) Alphabets only Validation :

$Main->validate("Bharat383","alpha"); => FALSE.

$Main->validate("Bharat","alpha"); => TRUE.

4) Alpha Numeric only Validation :

$Main->validate("Bharat383!@#$$%%","alphanumeric"); => FALSE.

$Main->validate("Bharat383","alphanumeric"); => TRUE.

$Main->validate("Bharat","alphanumeric"); => TRUE.

$Main->validate("383","alphanumeric"); => TRUE.

5) Email Validation :

$Main->validate("Bharat383","email"); => FALSE.

$Main->validate("bharat@","email"); => FALSE.

$Main->validate("bharat@domain","email"); => FALSE.

$Main->validate("bharat@domain.com","email"); => TRUE.

6) URL Validation :

$Main->validate("Bharat383","url"); => FALSE.

$Main->validate("bharat@","url"); => FALSE.

$Main->validate("www.domain","url"); => FALSE.

$Main->validate("www.domain.com","url"); => TRUE.

$Main->validate("http://www.domain.com","url"); => TRUE.

8) SendMail($mailto,$subject,$message)

Description : This method will sent mail using PHP Mail() function. Mail will be sent with Headers.

From Email and From Name will be set as your Site Email and Site Title. You can set the Site Email and Site Title from the Site Settings page in the Admin Panel. You can send HTML Mail Template by this.

Parameters :

$mailto : Receiver Email Address. You can send the same mail to multiple email addresses, by provide comma (,) separated email addresses.

$subject : Subject for the Mail.

$message : Message Contain for the Mail. You can send plain text or HTMl Mail Content.

Return : No Return Method

Syntax :

```
$Main->SendMail("bharatparmar383@gmail.com","Test Subject","Test Message");
```

Example :

```
$Main->SendMail("bharatparmar383@gmail.com","Test Subject","Test Message");
```

9) GetRandomString($length,$stringtype)

Description : This method will be used to get the Random String for the Verificatin Code, Random Password etc. The method will return Only Alphabets String, Only Numeric String and Alpha-Numeric String.

Parameters :

$length : Number of Characters String. eg : 5

$stringtype :

0 => Alpha-Numeric Value (Default)

1 => Numeric Value

2 => Alphabets Value

Return : An3Y08eiS9m

Syntax :

$Main->GetRandomString(5,0);

Example :

$Main->GetRandomString(5); => H93vK

$Main->GetRandomString(5,0); => 9mTk1

$Main->GetRandomString(5,1); => 93861

$Main->GetRandomString(5,2); => bJpwI

10) DateDifference($date1,$date2)

Description : This method will return the ARRAY of the date difference from the 2 parameters. It will return total hours of difference, total days and extra hours if hours remain more than days.

Parameters :

$date1 = Date 1 any date format

$date2 = Date 2 any date format.

Return :

Array :

hours : 54 => Total Hours

days : 2 => Total Days

extra_hours : 6 => Extra hours after 2 days.

Syntax :

$Main->DateDifference("25-10-2015 12:00:00","22-11-2015 06:30:00");

Example :

$Main->DateDifference("10-11-2015 06:00:00","12-11-2015 14:00:00");

11) MakePassword($parameter1);

Description : This method will return the Encrypted format string which will be generated with the combination of the SECRET KEY and the string which you will provide here. You can modify the password encryption method as per your requirement. Here Password has been encrypted with sha1.

Parameters :

$parameter1 : String which you want to encrypt.

Return : Encrypted String

Syntax :

$Main->MakePassword("12345");

Example :

$Main->MakePassword("12345");

Output : 91454792315318ed74d3925f08ff833bae214594

12) UploadFile($files,array $array)

Description : This method will upload the file. It has maximum file size limitation, file type validation and maximum allowed multiple files. You can upload the single file or multiple files.

Parameters :

$files : This should be the file array of the file upload elements. $_FILES

$array : This array should have basic requirements for the file upload as below :

a) uploadpath* : this should be the relative path where the file will be uploaded. This parameter must be set properly. Required Parameter. Parameter value should be string.

b) limit : if you want to make limit to allow limited files to upload when using the multiple file select. Optional Parameter. Parameter value should be numeric format. By default it will upload the selected file(s).

c) filetype* : This parameter should have array of the allowed file types. eg. If you want to upload image file, then you can set array here with jpg, gif, png etc. This parameter is required.

d) maxsize* : This parameter will check the file size limitation. It will accept the numeric value which should be KB. (Kilo Bytes : 1024 KB for 1MB)

Return : This function will return the array of the uploaded files. By default file name will be changed as per the time and random numeric string which will be unique.

Example :

```
<?php


$filesetting = array(
        "filetype"=>array("jpg","png","gif","jpeg","bmp"),
        "uploadpath"=>"../images/",
        "maxsize"=>1000
        );
$filename_array= $Main->UploadFile($_FILES['logo_image'],$filesetting);


?>
```

For details, You can check the admin/settings.php and admin/class/admin.class.php file for the Logo Image Upload Action.

13) DeleteFile(array $array)

Description : This method will delete single or multiple file(s).

Parameter :

Array :

a) files : file name array

b) uploadpath : File Location Path from where to all the files array file need to remove.

Return : No Return Method

Syntax :

$Main->DeleteFile($array);

Example :

1) Single File :

$array = array("bharat383.jpg","images/profile/");

$Main->DeleteFile(array $array)

2) Multiple Files :

$file_list = array ("1.jpg","2.jpg","3.jpg","4.jpg");

$array = array($file_list,"images/profile/");

$Main->DeleteFile(array $array)

Database Operation with MySQL

All the Database Operation Related functions are added to class/Main.class.php which has been extended by all other classes. Currently the framework support only MySQL Database. If you want to use any other database type or MySQLi, you just need to modify little bit in all these functions.

1) InsertRecord($tablename, array $values)

Description :

This method will be used to Insert New Record in the Particular Table.

Parameters :

a) $tablename (String) : The name of table. Please note that if you have set the Table Prefix for you database table, you need not to provide full name of the table. Just provide the table name after table prefix. Table Prefix has been added automatically from the Method.

b) $value (Array) : The array should have associated array with the table field name and their values which you want to set.

Return :

The method will return the Last Insert Id. On error, it will throgh the error.

Syntax :

$Main->InserRecord("user_master",$info_array);

Example :

Table Name : bh_user_master

Fields : user_id, firstname, lastname, email, password, register_date, register_ipaddress, user_type, active_status, verify_code, verify_status.

```
<?php

$info_array = array(
        "firstname"=>"Bharat",
        "lastname"=>"Parmar",
        "email"=>"bharatparmar383@gmail.com",
        "password"=>$Main->MakePassword("12345"),
        "register_date"=>date("Y-m-d H:i:s"),
        "register_ipaddress"=>$_SERVER['REMOTE_ADDR']
);

echo $Main->InserRecord("user_master",$info_array);

?>
```

Output : 1 (Last User ID)

2) InsertMultipleRecord($tablename,$fieldarray,$valuearray)

Description :

This method will be used to Insert New Multiple Records in the Particular Table at single time.

Parameters :

a) $tablename (String) : The name of table. Please note that if you have set the Table Prefix for you database table, you need not to provide full name of the table. Just provide the table name after table prefix. Table Prefix has been added automatically from the Method.

b) $fieldarray (Array) : The array of the fields name which you want to set the values in the table.

c) $valuearray (Array) : The Multi-Dimension Array of the values which you want to save.

Return :

The method will return the number of records inserted . On error, it will throgh the error.

Syntax :

$Main->InsertMultipleRecord($tablename,$fieldarray,$valuearray);

Example :

Table Name : bh_user_master

Fields : user_id, firstname, lastname, email, password, register_date, register_ipaddress, user_type, active_status, verify_code, verify_status.

```
<?php

$fieldarray =
array("firstname","lastname","email","password","register_date","register_ipaddress");

$info_array = array(

        array("Bharat","Parmar","bharatparmar383@gmail.com",$Main-
>MakePassword($Main->GetRandomString(7)),date("Y-m-d H:i:s"),$_SERVER['REMOTE_ADDR']),

        array("Jack","Thomas","jackthomas1122@gmail.com",$Main-
>MakePassword($Main->GetRandomString(7)),date("Y-m-d H:i:s"),$_SERVER['REMOTE_ADDR']),

        array("Lucy","Grill","lucygrill1122@gmail.com",$Main->MakePassword($Main-
>GetRandomString(7)),date("Y-m-d H:i:s"),$_SERVER['REMOTE_ADDR']),

        array("Jemmy","Christ","jemmychrist1122@gmail.com",$Main-
>MakePassword($Main->GetRandomString(7)),date("Y-m-d H:i:s"),$_SERVER['REMOTE_ADDR']),

);

echo $Main->InsertMultipleRecord("user_master",$fieldarray,$info_array);

?>
```

Output : 3 (Total number of records inserted.)

3) GetSingleRecord($tablename,array $array)

Description : This method is used to retrive single record from the particular table.

Parameters :

a) $tablename (String) : Table name without Table Prefix.

b) $array (Array) : Array for the query setup as below .

b.1) fields => Fields name which you want to retrive from the table. By Default it will retrive all fields. You can set "*" for all fields. If you want to retrive some particular fields value, you can set fields value as comma separated field name of the table. You can also Use the MySQL Math and other functions as field here like count, max, min etc.

b.2) where => Where clause for the query related to your data retrivle. By Default it will be blank.

Return : This method will return the array of the fields and its value. It will be in associated value.

Syntax :

$Main->GetSingleRecord($tablename,array $array);

Example :

Table Name : bh_user_master

```
<?php
	$info_array = array(
		"where"=>"user_id='1'"
	);

	$result  = $Main->GetSingleRecord("user_master",$info_array);
?>
```

Output :

```
Array (
	"firstname"=>"Bharat",
	"lastname"=>"Parmar",
	"email"=>"bharatparmar383@gmail.com",
	"password"=>"12313131313123",
	"register_date"=>"22-11-2015 06:00:00",
	"register_ipaddress"=>"127.0.0.0",
	"active_status"=>"1",
	"verify_code"=>"12345",
	"verify_status"=>"1"
)
```

```
<?php
	$info_array = array(
		"fields"=>"user_id,firsname,lastname,email"
		"where"=>"user_id='1'"
	);

	$result  = $Main->GetSingleRecord("user_master",$info_array);
?>
```

Output :

```
Array (
	"user_id"=>"1",
	"firstname"=>"Bharat",
	"lastname"=>"Parmar",
	"email"=>"bharatparmar383@gmail.com"
)
```

4) GetRecord($tablename,array $array)

Description : This method will be used to get Multiple records from the table. You can user where caluse, order, group, limit etc here.

Parameters :

a) $tablename* (String) : Table name without Table Prefix.

b) $array* (Array) : Array for the query setup as below .

b.1) fields (String) => Fields name which you want to retrive from the table. By Default it will retrive all fields. You can set "*" for all fields. If you want to retrive some particular fields value, you can set fields value as comma separated field name of the table. You can also Use the MySQL Math and other functions as field here like count, max, min etc.

b.2) where (String) => Where clause for the query related to your data retrivle. By Default it will be blank.

b.3) orderby (String) => You can retrive the record with sorting order using orderby here. You can use single or multiple fields here. By Default it will sort by first field in desc order.

b.4) ordertype (String) => You can retrive the record by order ascending or descending order. By Default it will be set as desc order.

b.5) limit (Integer) => If you want to add the limit for the data retrival or want to retrive limited number of records, you can set the number of records here. By default it will retrive all the records.

b.6) startfrom (Integer) => This parameter is used to get records while retriving pagination based records.

b.7) groupby (String) => You can retrive records by using "groupby". You can also user multiple fields by using comma separated string value. By default it will be blank.

Return : This method will return the array of the all records. It will be in associated value. The array will be in multi-dimensional array format.

Syntax :

```
$Main->GetRecord($tablename,array $array);
```

Example :

Table Name : bh_user_master

```
<?php
        $info_array = array();
        $result  = $Main->GetSingleRecord("user_master",$info_array);
?>
```

Output :

```
Array (
            Array[0] (
                    "firstname"=>"Bharat",
                    "lastname"=>"Parmar",
                    "email"=>"bharatparmar383@gmail.com",
                    "password"=>"12313131313123",
                    "register_date"=>"22-11-2015 06:00:00",
                    "register_ipaddress"=>"127.0.0.0",
                    "active_status"=>"1",
                    "verify_code"=>"12345",
                    "verify_status"=>"1"
            ),
```

```
Array[1] (
        "firstname"=>"Jack",
        "lastname"=>"Thomas",
        "email"=>"jackthomas1122@gmail.com",
        "password"=>"12313131313123",
        "register_date"=>"22-11-2015 06:00:00",
        "register_ipaddress"=>"127.0.0.0",
        "active_status"=>"1",
        "verify_code"=>"12345",
        "verify_status"=>"1"
),

Array[2] (
        "firstname"=>"Lucy",
        "lastname"=>"Grill",
        "email"=>"lucygrill1122@gmail.com",
        "password"=>"12313131313123",
        "register_date"=>"22-11-2015 06:00:00",
        "register_ipaddress"=>"127.0.0.0",
        "active_status"=>"0",
        "verify_code"=>"12345",
        "verify_status"=>"1"
),
```

```
                Array[3] (
                        "firstname"=>"Jemmy",
                        "lastname"=>"Christ",
                        "email"=>"jemmychrist1122@gmail.com",
                        "password"=>"12313131313123",
                        "register_date"=>"22-11-2015 06:00:00",
                        "register_ipaddress"=>"127.0.0.0",
                        "active_status"=>"0",
                        "verify_code"=>"12345",
                        "verify_status"=>"0"
                )

)
```

```
<?php
        $info_array = array(
                "fields"=>"user_id,firsname,lastname,email"
                "where"=>"active_status='1' and verify_status='1'"
        );

        $result  = $Main->GetSingleRecord("user_master",$info_array);
?>
```

Output :

```
Array (
        Array[0] (
                "firstname"=>"Bharat",
                "lastname"=>"Parmar",
                "email"=>"bharatparmar383@gmail.com"
        ),

        Array[1] (
                "firstname"=>"Jack",
                "lastname"=>"Thomas",
                "email"=>"jackthomas1122@gmail.com"
        )
)
```

5) UpdateRecord($tablename,array $values,$where)

Description : This method will be used to update the current records.

Parameters :

a) $tablename* (String) : Table name without Table Prefix.

b) $array* (Array) : Array of the Fields and Value as Associated Array Format.

c) where (String) : Where clause for the record update query.

Return : This method will return TRUE or FALSE. If the record has been updated, it will return TRUE else FALSE.

Syntax :

```
$Main->UpdateRecord($tablename,$info_array,$where);
```

Example :

Table Name : bh_user_master

```
<?php

$info_array = array(
        "firstname"=>"Bharat11",
        "lastname"=>"Parmar22",
        "email"=>"bharatparmar1122@gmail.com"

);

$where = "user_id='1' and active_status='1' and verify_status='1'";
echo $Main->InserRecord("user_master",$info_array,$where);

?>
```

Output : TRUE

6) DeleteRecord($tablename, $where, $limit=0)

Description : This method will be used to delete the current record(s).

Parameters :

a) $tablename* (String) : Table name without Table Prefix.

b) $where (String) : Where clause for the record update query.

c) $limit (Integer) : Maximum Number of records you want to delete. By default it will be unlimited. Record will be delete as per the where caluse.

Return : This method will return TRUE or FALSE. If the record has been delete, it will return TRUE else FALSE.

Syntax :

```
$Main->DeleteRecord($tablename,$where,$limit);
```

Example :

Table Name : bh_user_master

```
<?php
$where = "user_id='1' and active_status='1' and verify_status='1'";
echo $Main->InserRecord("user_master",$where);
?>
```

Output : TRUE

```
<?php
$where = "active_status='1' order by user_id asc";
$limit = 2;
echo $Main->DeleteRecord("user_master",$where,$limit); //THIS WILL DELETE MAXIMUM 2 RECORDS as per the where caluse.
?>
```

Output : TRUE

7) GetCustom($query_string)

Description : This method will be used to action any custom query. You can use this method for the join, union or any other query here. Please note that here, you must have to use the full name of the table. TABLE PREFIX will not be added here.

Parameters :

a) $query_string* (String) : Query string.

Return : This method will return the records in associated Array Format.

Syntax :

```
$Main->GetCustom($query_string);
```

Example :

Table Name : bh_user_master

```
<?php

$query_string = "select u.user_id, u.firstname, u.lastname, u.email,c.subject, c.message from bh_contact_messages as c, bh_user_master as u where c.email = u.email limit 10 order by c.created_date desc group by c.subject";

$result = $Main->InserRecord($query_string);

?>
```

Output :

```
Array (
        Array[0] (
                "user_id"=>1,
                "firstname"=>"Bharat",
                "lastname"=>"Parmar",
                "email"=>"bharatparmar383@gmail.com",
                "subject"=>"Test Subject",
                "message"=>"Test Message"
        ),
        Array[1] (
                "user_id"=>1,
                "firstname"=>"Bharat",
                "lastname"=>"Parmar",
                "email"=>"bharatparmar383@gmail.com",
                "subject"=>"Test Subject2",
                "message"=>"Test Message2"
        ),
        Array[2] (
                "user_id"=>1,
                "firstname"=>"Bharat",
                "lastname"=>"Parmar",
                "email"=>"bharatparmar383@gmail.com",
                "subject"=>"Test Subject3",
                "message"=>"Test Message3"
        )
)
```

Class : Frontend Class

FILE NAME : Frontend.class.php

LOCAITON : class/Frontend.class.php

FILE DETAILS : Frontend Class. Includes CMS Pages, Contact Us

Extends : Main.class.php => class/Main.class.php

Description :

Frontend.class.php file includes following Global Properties and Methods. It would be better you will use this class and extends it for other class file. All the other class for the frontend panel, should follw this. As the class contains all basic methods for the frontend.

Global Properties :

1) pagefilename (String) = Page File Name.

2) webpage_data (Array) = The array contains entire webpage data from the cms_webpages table depends on the current URL.

These methods are used for the Frontend Basic functionalities.

The class included following methods :

1) GetWebPageContent($parameter1)

Description : This method will retrive the date of the given web page id or web page file name. This method will retrive data from cms_webpages table.

Syntax :

$Frontend->GetWebPageContent($parameter1);

Parameters :

a) $parameter1 : This should be the webpage id or the webpage file name like index.php, login.php.

Example :

```
<?php
	//by web page id
	$webpage_data = $Frontend->GetWebPageContent(1);
	//by web page file name
	$webpage_data = $Frontend->GetWebPageContent("index.php");
?>
```

Output :

```
Array (
	"webpage_id"=>"1",
	"login_require"=>"1",
	"webpage_url"=>"index.php",
	"webpage_title"=>"Home Page",
	"webpage_keywords"=>"homepage",
	"webpage_descriptions"=>"homepage",
	"webpage_content"=>"This is home page content.",
	"active_status"=>"1"
)
```

2) ContactUs()

Description :

This method is used for the contact us form for the frontend. It will send the contact us reply mail to sender of the message as well as store the message with IP Address and time in the contactus_messages table which can be managed from the Admin Panel.

3) DisplayMessage()

Description :

This method is used to display the messages on the frontend for any successfull action, errors, warning or information. This method will display the message with SESSION. After displaying the message, it will be unset automatically.

The SESSION variable which is used to save the message is "$_SESSION['response']". This should be the array of the message type and the message content. You can set multiple messages with the same variable to make it multi-dimensional arry.

Syntax :

$Frontend->DisplayMessage();

Parameters : Not Required.

Return : No Return.

Example :

```
<?php
//Syntax : array("message_type","message string here")
   //FOR SINGLE MESSAGE ONLY
  $_SESSION['response'] = array("error","Error message here");

//FOR MULTIPLE MESSAGES
$_SESSION['response'][] = array("info","You can upload only jpg, gif or png image only.");
$_SESSION['response'][] = array("warning","Only jpg, gif and png image allowed.");
```

$_SESSION['response'][] = array("error","Your account has been deactivated by the Admin. You cannot Login now.");

$_SESSION['response'][] = array("success","Your profile details has been updated successfully.");

$Frontend->DisplayMessage();

?>

Output :

a) Error Message Here

b.1) You can upload only jpg, gif or png image only.

b.2) Only jpg, gif and png image allowed.

b.3) Your account has been deactivated by the Admin You cannot Login now.

b.4) Your profile details has been updated successfully.

All the above message will be displayed with the bootstrap css for the alert box as below :

a) info : alert alert-info,

b) warning : alert alert-warning

c) error : alert alert-danger

d) success : alert alert-success

You can modify the html and css in this method.

Class : User Class

FILE NAME : User.class.php

LOCAITON : class/User.class.php

FILE DETAILS : User Class. Includes User Register, Login, Forget Password, Profile, Logout

Extends : Frontend.class.php => class/Frontend.class.php

Description :

This class is used for the User Panel. The class file contains User Sign up, Login, Forgot Password, Logout, Profile Editing, Change Password and login check for the login required pages.

Methods:

1) UserRegister()

Description :

This method is used for the User Registration. The User Details will be stored in the user_master table. **register.php** file has the layout of the User Registration Form.

The function will validate the User Input with Jquery Validation and PHP Validation.

After successfully registration, the script will send the mail to user. The Registration Email template will be managed from the Admin Panel. If you want to disable the Mail you can inactive for the same.

For more details check :

i) register.php

ii) class/User.class.php

2) UserForgetPassword()

Description :

This method will change the User Password and send the mail with new Password. Password will be auto generated and random string.

For more details check :

i) login.php

ii) class/User.class.php

3) UserLogin()

Description :

This method will check user email and password. After successfull login, user will be redirect to the user profile page in the User Panel.

For more details check :

i) login.php

ii) class/User.class.php

4) CheckUserLogin()

Description :

This method will check the user is logged in or not. The method will check the current page required the login from the cms_webpages table. If the user is not logged in the system and try to access the login required page from the user panel, it will redirect the user to the login page.

For more details check :

i) class/User.class.php

5) UserProfile()

This method is used for the profile edit action from the user panel . The method will follow the Jquery and PHP validation on user input.

For more details check :

i) user/profile.php

ii) class/User.class.php

6) ChangePassword()

This method is used for the profile edit action from the user panel .

For more details check :

i) user/profile.php

ii) class/User.class.php

7) Logout()

This method will destroy the current active session and redirect user on the login page.

If you want to add more modules for the user, you can extends this class for the new class file.

Admin Panel

URL : Yourdomain/admin

Short URL : **ADMIN_URL**

Description :

Admin panel has been developed for the framework overview, you can use it for your project and modify as per your requirement or you can learn the basic structure of the framework.

During the Installation , you can set the Admin User Login Details.

File and Directory Structure :

Includes Files :

1) header.php : header view for the Admin Panel Template

2) footer.php : footer view for the Admin Panel Template

Layout/View Files :

1) admin.php : Admin Profile View File.

2) cmswebpages.php : CMS Web Pages Manage View File.

3) contactmessages.php : Contact Us Messages Manage View File.

4) database.php : Database Manage View File.

5) emailtemplate.php : Email Template View File.

6) index.php : Admin Dashboard View File.

7) login.php : Admin Login View File.

8) manageuser.php : User Management View File.

9) settings.php : Site Settings View File.

Class Files : (Class Directory)

1) Admin.class.php : Class file which contains Admin Login, Profile, Site Settings Methods.

2) CMS.class.php : Class file which contains CMS Webpages and Email Template Methods.

3) Contact.class.php : Class file which contains Contact Us Message Methods.

4) Dashboard.class.php : Class file which contains Dashboard Methods.

5) Database.class.php : Class file which contains Database Export, Import and Download Datbase Export File Methods.

6) User.class.php : Class file which contains User Manage (Add/Edit/View/Delete) Methods.

Database Directory :

This directory contains download.php which will be used to download the MySQL Exported file. Admin can export the Database file from the Admin Panel and the Exported SQL file will be store here.

Images Directory :

Admin Panel template images.

Includes Directory :

header.php and footer.php file for the Admin Template File.

Lib Directory :

Javascript plugins for the Admin Panel.

Stylesheets :

CSS files for the Admin Panel Template.

CREATE NEW CLASS FILE CRUD METHODS

Description :

It would be better if you will create separate class for your every modules like user, payment, advertisement etc.

Separate class will be helpful to you understand your long code.

To create new class, you must need to extend the following class.

1) Frontend Class (class/Frontend.class.php) for public

2) User Class (class/User.class.php) for user panel or anything which required User Athentication.

3) Admin Class (admin/class/Admin.class.php) for Admin Panel.

You can use all the methods of the Main Class (class/Main.class.php).

Example :

Let's create News Module for the Admin Panel. Here we will create two different classes for the News Module, one for Admin Panel and another for the public.

Admin Panel : News Class (admin/class/News.class.php)

Table Structure :

```
CREATE TABLE IF NOT EXISTS `bh_news` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `news_title` varchar(255) NOT NULL,
  `news_content` longtext NOT NULL,
  `active_status` int(1) NOT NULL COMMENT '0:inactive, 1:Active',
  `created_timee` datetime NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

```
<?php
@include("Admin.class.php");
class News extends Admin
{
}
?>
```

Please note that you must have to add the constructor to your new class.

```
<?php
@include("Admin.class.php");
class News extends Admin
{
	public function __construct()
	{
		parent::__construct();
	}
}
?>
```

News Add Action :

a) HTML FORM TO ADD/EDIT NEWS ITEM :

```
<form method="post">
        <div class="form-group">
          <label>News Title:</label>
          <input type="text" class="form-control" name="news_title" title="Enter News Title." value="<?php echo @$Admin->data['news_title'];?>">
        </div>
        <div class="form-group">
          <label>News Content:</label>
          <textarea class="form-control" name="news_content" title="Enter News Content."><?php echo @$Admin->data['news_content'];?>"</textarea>
        </div>
        <div class="form-group">
          <label>Status :</label>
          <select name="active_status">
              <option value="0" <?php if(@$Admin->data['news_content']=="0") echo "selected";?>>Inactive</option>
              <option value="1" <?php if(@$Admin->data['news_content']=="1") echo "selected";?>>Active</option>
          </select>
        </div>
        <input type="submit" name="submit_news" value="Save" class="btn btn-primary">
</form>
```

=> Note that title has been added for the jQuery validation on Form Submit.

B) Display News items in tabular format

```
<table>
	<tr>
		<th>#ID</th>
		<th>News Title</th>
		<th>Created Date</th>
		<th>Status</th>
		<th colspan="2">Action</th>
	</tr>
	<?php
		if(count($News->data)>0)
		{
			foreach($News->data as $newsitem) {?>
			<tr>
				<td><?php echo $newsitem['id'];?></td>
				<td><?php echo stripslashes($newsitem['news_title']);?></td>
				<td><?php echo date("d-m-Y H:i:s",strtotime($newsitem['created_time']));?></td>
				<td>
					<?php if($newsitem['active_status']=="0") {?>
						<a href="<?php $News->StatusLink($newsitem['id'],"1");?>">Inactive</a>
					<?php } else { ?>
						<a href="<?php $News->StatusLink($newsitem['id'],"0");?>">Active</a>
					<?php } ?>
				</td>
```

```
                                    <td><a href="<?php $News->EditLink($newsitem['id']);?>">Edit</a></td>

                                    <td><a href="<?php $News->DeleteLink($newsitem['id']);?>">Delete</a></td>

                                </tr>


                                <?php } ?>

                        }

                        else

                        {

                                echo "<tr><td align='center' colspan='5'>Records not available.</td></tr>"

                        }

                ?>

</table>
```

c) Class File

```
<?php

/*
FILE NAME : News.class.php
LOCAITON : admin/class/News.class.php
FILE DETAILS : News Class which contains News Add, Edit, Delete, View and Status Change.
Extends : Admin.class.php => admin/class/Admin.class.php
*/

@include("Admin.class.php");
class News extends Admin
{
	public function __construct()
	{
		parent::__construct();

		if(isset($_POST['submit_news']) && !isset($_GET['id']))
		{
			$this->AddNews();
		}
		else if(isset($_POST['submit_news']) && isset($_GET['id']) &&
is_numeric($_GET['id']))
		{
			$this->EditNews();
		}
```

```
		else if(!isset($_POST['submit_news']) && isset($_GET['id']) && is_numeric($_GET['id']))
		{
			$this->SetEditData($_GET['id']);
		}
		else if(isset($_GET['action']) && $_GET['action']=="delete" && isset($_GET['id']) && is_numeric($_GET['id']))
		{
			$this->DeleteNews($_GET['id']);
		}
		else if(isset($_GET['action']) && $_GET['action']=="status" && isset($_GET['id']) && is_numeric($_GET['id']) && isset($_GET['status']) && is_numeric($_GET['status']))
		{
			$this->ChangeStatus($_GET['id'],$_GET['status']);
		}

		$this->DisplayNewsData();

	}

	private function AddNews()
	{
		if(!$this->validate($_POST['news_title'],"required"))
		{
			$_SESSION['response'][] = array("warning","Please Enter News Title.");
		}
```

```
if(!$this->validate($_POST['news_content'],"required"))
{
	$_SESSION['response'][] = array("warning","Please Enter News Content.");
}

if(empty($_SESSION['response']))
{
	$info_array = array(
			"fields"=>"id",
			"where"=>"news_title like '".$_POST['news_title']."'"
		);

	$duplicate_record = $this->GetSingleRecord("news",$info_array);

	if(count($duplicate_record)>0)
	{
		$_SESSION['response'][] = array("error","Duplicate News Title. ");
	}
	else
	{
		array_slice($_POST,-1);
		$_POST['created_time']=date("Y-m-d H:i:s");
		$news_id = $this->InsertRecord("news",$_POST);
		$_SESSION['response'][] = array("success","News Item has been
added successfully.");
		$this->RedirectPage($this->pagefilename);
	}
```

```
		}

		$this->data=$_POST;
	}

	private function SetEditData($id)
	{
		$info_array = array(
					"where"=>"id = '".$id."'"
				);

		$this->data = $this->GetSingleRecord("news",$info_array);

		if(empty($this->data))
		{
			$_SESSION['response'][] = array("error","Record not found. ");
			$this->RedirectPage($this->pagefilename);
		}
	}

	private function EditNews($id)
	{
		if(!$this->validate($_POST['news_title'],"required"))
		{
			$_SESSION['response'][] = array("warning","Please Enter News Title.");
		}
```

```
		if(!$this->validate($_POST['news_content'],"required"))
		{
			$_SESSION['response'][] = array("warning","Please Enter News Content.");
		}

		if(empty($_SESSION['response']))
		{
			$info_array = array(
					"fields"=>"id",
					"where"=>"news_title like '".$_POST['news_title']."' and id
!='".$id."'
				);

			$duplicate_record = $this->GetSingleRecord("news",$info_array);

			if(count($duplicate_record)>0)
			{
				$_SESSION['response'][] = array("error","Duplicate News Title. ");
			}
			else
			{
				array_slice($_POST,-1);
				$where = id !='".$id."';
				$updated = $this->UpdateRecord("news",$_POST,$where);
				$_SESSION['response'][] = array("success","News Item has been
updated successfully.");
				$this->RedirectPage($this->pagefilename);
			}
```

```
		}


		$this->data=$_POST;
	}


	private function DeleteNews($id)
	{
		$where = "id='".$id."'";
		$deleted = $this->DeleteRecord("news",$where);


		if($deleted>0)
		{
			$_SESSION['response'][] = array("success","Record has been deleted
successfully. ");
		}
		$this->RedirectPage($this->pagefilename);
	}


	private function ChangeStatus($id,$status)
	{
		$where = id !='".$id."';
		$info_array = array("active_status"=>$status);
		$updated = $this->UpdateRecord("news",$info_array,$where);


		if($updated>0)
		{
			$_SESSION['response'][] = array("success","Active status has been changed
successfully.");
```

```
                }

                $this->RedirectPage($this->pagefilename);

        }


        private function DisplayNewsData()

        {

                $info_array = array();

                $this->data = $this->GetRecord("news",$info_array);

        }


}



?>
```

```
<?php

/*
FILE NAME : News.class.php
LOCAITON : class/News.class.php
FILE DETAILS : News Class which contains News List View and Details View.
Extends : Frontend.class.php => class/Frontend.class.php
*/

@include("Frontend.class.php");
class News extends Frontend
{
	public function __construct()
	{
		parent::__construct();


		if(isset($_GET['id']) && $_GET['id']!="" && is_numeric($_GET['id']))
		{
			$this->GetNewsData($id);
		}
		else
		{
			$this->GetNewsData();
		}
	}
```

```
	private function GetNewsData($id="")
	{
		if($id!="")
		{
			$info_array = array(
				"where"=>"active_status='1' and id='".$id."'"
			);
			$this->data = $this->GetSingleRecord("news",$info_array);
		}
		else
		{
			$info_array = array(
				"where"=>"active_status='1'"
			);
			$this->data = $this->GetRecord("news",$info_array);
		}
	}
}


?>
```

Short Codes

Description :

You can use the short code which will be heplful to you to develop your application in very short time.

1) HTML Form :

Description :

Develop HTML Form with the required table fields with the same name as in your table. eg. If your table contains firstname, lastname, email, birthdate fields and you want to add/update the records for the same, you can set the form input element name same.

<input type="text" name="firstname">

<input type="text" name="lastname">

<input type="email" name="email">

<input type="date" name="birthdate">

2) Jquery User Input Validation :

Description :

You can set the user input validation on the form element just adding a class as per your requirement.

e.g

a) Allow Alphabets : <input type="text" name="firstname" class="alpha">

b) Allow Numbers : <input type="text" name="price" class="numeric">

i) alpha : Allow Alphabets values to the input element

ii) numeric : Allow numeric values to the input element

iii) alphanumeric : Allow Alphabets and numeric values to input element

iv) email : Allow characters which are used for the email address.

3) Jquery Blank Validation on form submit :

Description :

You can set the blank validation on form submit just adding a title attribute to the input element of the form.

Example :

<input type="text" name="firstname" title="Please Enter First Name. It should not be blank.">

When the form will be submitted by the user and if the input element has blank value, the form will not be submitted and stop the execution. It will add the red border to the element and display the error message as the placeholder of the particular element. The Placeholder message will be the value of the title attribute.

Please note that, if any input has not the title attribute, validation will skip that element to validate.

4) Insert record to Database with PHP

Description :

If you have set the input element name as same as the database table name, it will save lots of time to make the code to insert the record in the database.

If you have set the submit name to the form, you must need to remove that element from the $_POST array as that element value will not be stored in the database table.

You can remove that element with two different methods as below :

i) <?php unset($_POST['submit_save']);?> : This will unset the last element from the $_POST array.

ii) <?php array_slice($_POST,-1);?> : This will remove the Last element from the $_POST array.

You can use any other method for the same.

Insert Action Code as below :

```
<?php

array_slice($_POST,-1);

$last_insert_id = $ClassObject->InsertRecord("tablename",$_POST);

?>
```

Update Action code as below :

```
<?php

array_slice($_POST,-1);

$updated = $ClassObject->UpdateRecord("tablename",$_POST);

?>
```

Contact Me

If there any query or confusion regarding the code structure or framework or if you find any issue in the code, do not hesitate to contact me. I will try my best to provide instant support.

Suggestions will be always accepted.

Bharat Parmar

Email : bharatparmar383@gmail.com

Mobile : +91 9687766553