

# Beginner's Guide to PHP Development with MVC Architecture

*Version 1.0 Published 8 May 2010*

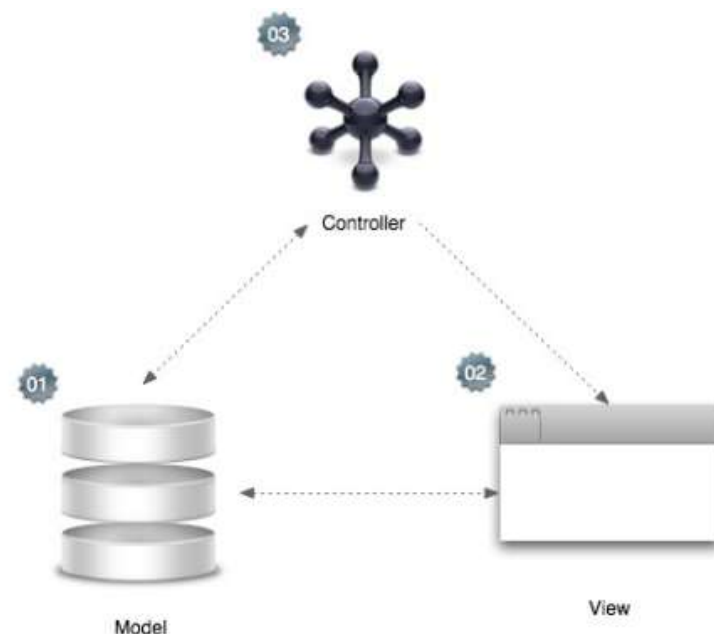
## Table of Contents

Beginner's Guide for PHP Development with MVC Architecture .....	1
MVC Architecture Part 1: Introduction to the Architecture .....	2
MVC Architecture Part 2: Understanding the Interiors.....	4
Example of MVC .....	4
Model Directory .....	5
Files in the model directory .....	5
Template Directory.....	5
Controller Directory.....	6
.htaccess file .....	6
Index.php .....	7
MVC Architecture Part 3: Creating a New Page in MVC Architecture.....	7
Login.tpl.php.....	8

## MVC Architecture Part 1: Introduction to the Architecture

Web development in PHP introduces a powerful architecture for PHP frameworks like Zend, CodeIgniter, and CakePHP – Model-View-Controller (MVC). MVC is more than it meets the eye. It is not just any other web development framework architecture for building elegant and systematic websites, but it has a full capability to support rapid web application development and dynamic interactivity with the database as well. You may find numerous articles on this subject over the Internet and some of them will tell a tale of comprehensive exclusivity as well. This is not a topic to be discussed in one article. A wide prospect such as MVC architecture should be dedicated more articles to elaborate on the complexity of the architecture. Here we try to cover maximum ground as much possible in three parts – the first post shall introduce and talk a wee bit on the MVC architecture, the following will delve deeper and explain the interiors of MVC, and the last article is a tutorial type on creating a website using MVC architecture.

MVC methodology typically splits the architecture of the website into 3 distinct parts which are kept operationally separate but interact with each other to deliver all aspects of the website and the administration system. The main aim of the MVC architecture is to separate the business logic and application data from the presentation data to the user. Adhering to MVC architecture benefits you in attaining a perfect design for an enterprise web application.



## Beginner's Guide to PHP Development with MVC Architecture

There is a common control flow in all forms of MVC built web applications. The Controller lies at the core of the architecture and it interacts with the user through the web browser. Behind the scene it communicates with the Model and the View components of the architecture.

The Model actually envelops the database so that the Controller can notify and direct the database of the user actions on the browser. It is here that the database is modified in accordance with the requests submitted by the user. This improves user interactivity and hence the overall seamless experience.

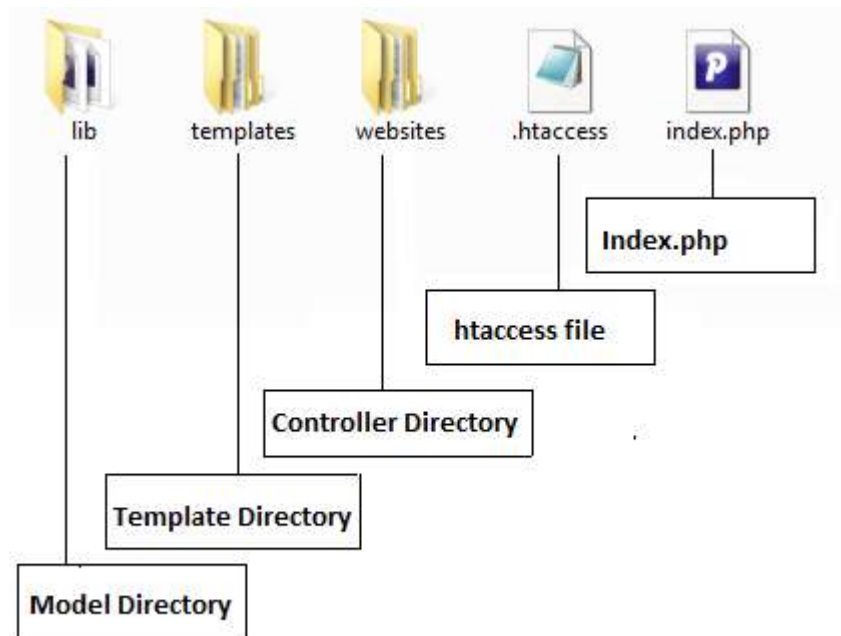
On receiving the query request from the user the controller handles the Model and in return Model notifies View to update the user interface (UI) on the screen in accordance to the database change.

When the user enters another query, the cycle is restarted.

## MVC Architecture Part 2: Understanding the Interiors

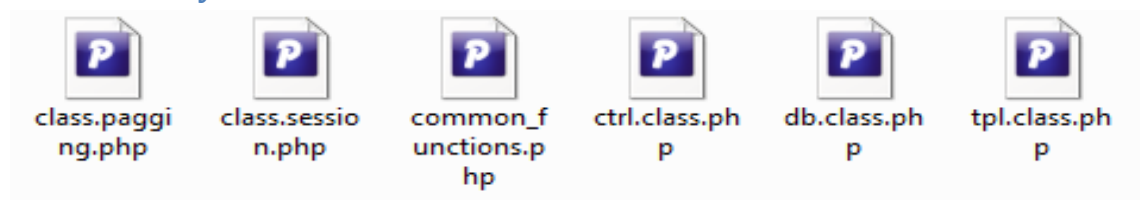
In the previous post we learnt what MVC actually is and how does it work. In this post we shall elaborate our discussion on the interiors of the MVC architecture. To understand the architecture we first suppose an example of a typical website built on the MVC architecture. The following picture gives an idea of the directory structure of the website and so to understand the architecture, we shall understand the structure of mentioned here.

### Example of MVC



Now, any web developer will know what the *index.php* and the *.htaccess* files are for, though we shall mention them in our explanation a little later. But the point here is what are the other folders viz. lib, templates, and websites for? Well, these are the MVC folders, and they are labeled according to their roles in the architecture. We shall start from the left most label and continue on to their right ones.

### Model Directory



The Model is where business logic is stored. Business logic is loosely defined as database connections or connections to data sources, and provides the data to the controller. The Model object knows about all the data that need to be displayed. It is Model who is aware about all the operations that can be applied to transform that object. It only represents the data of an application. The Model represents enterprise data and the business rules that govern access to and updates of this data. Model is not aware about the presentation data and how the data will be display to the browser.

### Files in the model directory:

**class.pagging.php** - Model for managing pagging of the whole site.

**class.session.php** - Model that is used for managing session of whole site.

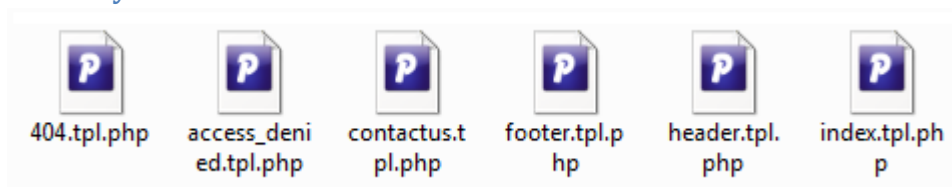
**common.functions.php** - Model that is used for managing common function of the whole site.

**ctrl.class.php** - Controller that is used for communication between the template and the Model file

**db.class.php** - Model that is used for Database connection, insert, update, and delete functionality.

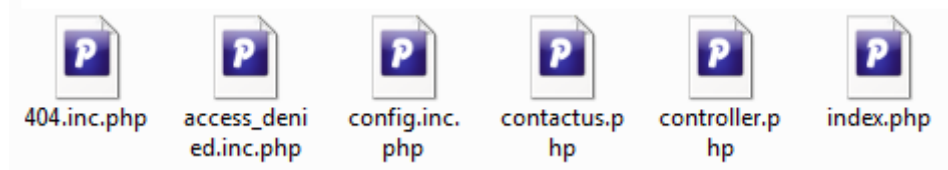
**tpl.class.php** - Model that is used to display template file.

### Template Directory



The View contains code that relates to presentation and presentation logic such as templating and caching. The View represents the presentation of the application. The View object refers to the model. It uses the query methods of the Model to obtain the contents and renders it. The View is not dependent on the application logic. It remains same if there is any modification in the business logic. In other words, we can say that it is the responsibility of the View's to maintain the consistency in its presentation when the Model changes.

### Controller Directory



Controller is often referred to as the application layer of the website. The Controller component is basically the code that processes data, writes out pages, gets data, logs, creates events and so on. Essentially this is the active part of the site & system which interface between the database, assets, templates et al, generating a result which the end user can see. Whenever the user sends a request for something then it always go through the Controller. The Controller is responsible for intercepting the requests from View and passes it to the model for the appropriate action. After the action has been taken on the data, the Controller is responsible for directing the appropriate View to the user

### .htaccess file

```
1 #website name
2 SetEnv website admin
3
4 #default Theme
5 SetEnv default_theme admin new
6
7 # Restrict apache from automatically including path related file
8 Options -MultiViews
9
10 # Enable Mod Rewrite
11 RewriteEngine on
12 RewriteCond %{REQUEST_FILENAME} !-f
13 RewriteCond %{REQUEST_FILENAME} !-d
14 RewriteRule ^(.*)$ index.php?q=$1 [L,QSA]
```

This is how a typical *.htaccess* file is written. The generic names such as “REQUEST\_FILENAME” should be replaced by the respective names for your website.

## Index.php

```
1 <?php
2
3 # @todo : implement all the use cases for config file's physical existence
4
5 # Get Physical Path of config file ( set in .htaccess )
6 define('_PATH', $SERVER['path']);
7
8 global $_user_msg;
9
10 include _PATH."config.inc.php";
11 include _PATH."websites/"._WEBSITE_NAME."/config.inc.php";
12 #unset autologin session so not occur any problem
13 # for update lang (for reflection) no access check, no authentication needed.
14 # else go for authentication & access check.
15 $url = _authenticate_url($url);
16 $url = _do_access_check($url);
17
18 define('_REQUEST_PAGE', $url);
19
20 # define lang_id constant for default language or any other requested
21 define('_REQUEST_LANG_ID', dc_find_current_language());
22 #storing currentlystorwed lagnauge in session
23 $_SESSION['lid'] = _REQUEST_LANG_ID;
24
25 # define page_id constant for default language or any other requested
26 define('_REQUEST_PAGE_ID', dc_find_current_page(_REQUEST_PAGE));
27
28 $ctrl->load();
29
30 $tpl->load();
31
32 ?>
```

In the index.php file first we set the path variable. Include config.inc.php file. The first config.inc.php file is global configuration file and second one is site configuration file if we have multiple site then we require to use this two configuration files lest we only include first configuration file.

On line no 15 we check the authentication for particular user.

On line no 16 we check the page access for particular user and anonymous user.

Define “REQUEST\_PAGE” and “\_REQUEST\_LANG\_ID” (Use for Multilanguage)

Load Controller and Tpl file using load function.

## MVC Architecture Part 3: Creating a New Page in MVC Architecture

## Beginner's Guide to PHP Development with MVC Architecture

To create a new page in MVC architecture we need to create two new files - one file in template directory and other file in controller directory.

Please check the below example

We first create one .tpl file in template directory named login.tpl.php and put the below code in that file.

### Login.tpl.php

```
1 <table>
2 <form name="login_frm" id="login_frm" method="post">
3 <tr>
4 <td colspan='2'><h1>Login</h1></td>
5 </tr>
6 <tr>
7 <td>UserName</td>
8 <td><input name="username" type="text" /></td>
9 </tr>
10 <tr>
11 <td>Password</td>
12 <td><input name="password" type="password" style="width:286px; height:19px;" value="" /></td>
13 </tr>
14 <tr>
15 <td><input class="inputbutton" type="submit" border="0" alt="" name="login_btn" value="Login" /></td>
16 </tr>
17 </form>
18 </table>
19
```

## Beginner's Guide to PHP Development with MVC Architecture

Secondly, we put one .php file in the Controller file and put the below code into that file.

```
1 <?php
2 $error = '';
3 $greetings = '';
4 $validation_result = true;
5 $_t=array();
6 $_t['username']['value'] = 'admin@admin.com';
7 $_t['password']['value'] = 'admin';
8
9 $error = isset($_SESSION['wanted_url']) ? 'Authentication Required. Please login below' : '';
10
11 if(isset($_POST['login_btn'])){
12     $validation_result=true;
13     $db = db::d();
14     $query =sprintf("SELECT au.au_id, au.au_first_name, au.au_last_name, ur.ur_role_id
15 FROM admin_users AS au WHERE au.au_email = '%s'
16 AND au.au_password = '%s'",es($_POST['username']),md5(es($_POST['password'])));
17
18     $res = $db->query($query);
19     $no_of_rows = mysqli_num_rows($res);
20     if($no_of_rows == 0){
21         $error .= "Please enter correct username and password";
22     }else{
23         $data = $db->format_data($res);
24         $_SESSION['user_id'] = $data[0]['au_id'];
25
26         $redirect_url = 'home/.homepage_by_userrole($data[0]['ur_role_id']);
27         header("Location: ". $_U.$redirect_url);
28         exit;
29     }
30 }
31 }
32 }
33 ?>
```

And thus you get your first MVC architecture website. So now you can use your new website by typing the URL of your website and accessing it from a web browser.