International Journal of Allied Practice, Research and Review
Website: www.ijaprr.com (ISSN 2350-1294)

The Basics of MVC

Maan Singh Rathore
Research Scholar, Shri Shri Jagdishprasad Jhabarmal Tibrewala University,
Jhunjhunu, Rajasthan

Abstract - The MVC pattern or Model View Controller architecture, was founded in around in the year of 1970, is software pattern or architecture or we can say a pattern of designing an application built on the basis of keeping the layer of data separate from the business methods that interact with the data.
So in a nutshell, a well-designed Model View Controller system allows developer at front-end and a developer at back-end to work on the same application without disturbing or we could say editing files either both is working on simultaneously same application.

Keywords – *Software Pattern, Architectures, Reusability*

I. Introduction

It has three main or core parts - Model View and Controller

Here will go through each one deeply and will understand how it flows and make request with each other.

Model – Why it is given the name model, the reason is the name given to the permanent storage of the data used in the overall design and its application. It givesthe access for the data to be displayed or viewed, or insert and save to, and is the middle layerfor the view components and the controller components in the whole architecture.

View - View is going to be used for the data which we requested from the model. It is viewed their and its final output is determined. In a general, web applicationwhich built using Model View Controller, the view is the part of the application where HTML, CSS and Javascript is generated and displayed as a final output. The View also capturesactions and reactions from the end user, who then goes to interact with the controller which is the middle layer of system. The basic example of this is a button generated by HTML which could be display and clicking on that it will redirect to controller.

Controller- The last and final element of the three is the controller. Its duty is to take care of data that the user submits or inputs, and do a quick update the model accordingly. Controller has no meant without user interaction mean it must have some use action without that it has no meaning. This is the only part of the architecture the user should be interacting.

II. Design basic stuff

Let's design some basic stuff to know the MVC concept and their reusability.

```
<?php
classappModel
{
public$sample_string;
public function __construct(){
      $this->sample_string = "MVC Application";
   }
}
```

The above one we created for model which have construct function which calls automatically when this class object's initiates.

```
<?php
classappView
{
public $model;
public $controller;
public function __construct($controller,$model) {
      $this->controller = $controller;
      $this->model = $model;
   }
public function appOutput(){
return "<div>" . $this->model->sample_string . "</div>";
   }
}
```

The above one we created for the application view which has construct function which calls automatically when appView class object's initiates.

```
<?php
classappController
{
public $model;
public function __construct($model) {
      $this->model = $model;
   }
}
```

The above one we created for the application controller which has construct function for the model which calls automatically when appController class object's initiates.

Now we have created our three basic elements which fulfil our aim, let see how we could combine them and make a relationship with them.

```
<?php
$appmodel = new appModel();
$appcontroller = new appController($appmodel);
$appview = new View($appcontroller, $appmodel);
echo $appview->appOutput();
```

III. Extend the design

Now we could add more as per user interaction to achieve a complete flow like this –

```
<?php
classappModel
{
public $sample_string;
public function __construct(){
      $this->sample_string = “MVC App, click here”;
   }
}

<?php
classappView
{
public $model;
public $controller;
public function __construct($controller,$model) {
      $this->controller = $controller;
      $this->model = $model;
   }
public function appoutput() {
return '<span><a href="mvcapp.php?btnaction=btnclicked"' . $this->model->sample_string .
"</a></span>";
   }
}

<?php
classappController
{
public $model;
public function __construct($model){
      $this->model = $model;
   }
public function btnclicked() {
      $this->model->sample_string = “MVC application data updated successfully”
   }
}
Now combine all these elements
<?php
$appmodel = new appModel();
$appcontroller = new appController($appmodel);
$appview = new appView($appcontroller, $appmodel);
if (isset($_GET['btnaction']) && !empty($_GET['btnaction'])) {
   $controller->{$_GET['btnaction']}();
}
echo $appview->appoutput();
```

IV. Conclusion

So we understood the basic theory about the Model View Controller architecture and have generated a basic application. We accounted this structure as a base component for any stable and reliable coding practices. It benefits the developers, testers and end user to get easiness in their area. Developers can simply get the errors and maintain their structure. Testers get test this, entire three component and user gets the better performance as all components are separately designed. The main use of this is re use the code that ultimately increases the performance of the application and makes your application stable if you have millions of users.

V. References

1) Tutorials Point - http://www.tutorialspoint.com/

2) W3Schools - http://www.w3schools.com

3) Wikipedia - https://en.wikipedia.org/wiki/Model-view-controller