ISSN 2454-8596
www.vidhyayanaejournal.org

An International Multidisciplinary Research E-Journal

Model View Controller (MVC) : A Latest Mobile & Web Application Development Approaches

Dharmendra Ambani[1], Dr.Ashwin Rathod[2]

[1]Lecturer at Department of Computer Science, Harivandana College, Rajkot (Guj-India)

[2]Principal at Harivandana College, Rajkot (Guj-India)

ABSTRACT

This paper is an effort to present the understanding of Model, view and controller(MVC). MVC is known as three layer development architecture used for both mobile and web application developments. Not necessarily only for web, MVC depends on how you use it. It's just framework that should work on both mobile and web. iOS development also work done on MVC style This paper present the details of MVC layers, it uses, advantages, disadvantages, detail work and functionalities. Main focus of this paper is explain the best way practices concern MVC based mobile web and mobile application developments.

KEYWORDS: IDE, MVC architecture, Literature review, MVC example, layers, functionalities, MVC.

INTRODUCTION

Model view controller (MVC) is an architectural pattern usually used in web application development [1] and mobile application development. MVC is complete framework. MVC work with three layers:

Model: Model layer used to work with logic of data. It will work with CRUD (Create, Read,, Update and Delete) operation with the associated databases with our application.

View: View are used to manage and prepare the user interface (UI) of our application. By using that UI user interact with our application.

Controller: Controller layer used to respond to the user request. It will work with user requested actions. These layer work with the model classes and select view and kit will display as per user requested.

MVC pattern architecture separates the characteristics of an application. First layer (Model) is related to the user logic, second layer (View) related to the business logic, and third layer (controller) is used to implement UI logic. MVC is used to specify the proper location of each logic in application[8]. It will provide parallel development. In simple way to a separation of concern, the view display data, the controller accepts the input and model is responsible for internal logic and data.

If using MVC pattern then three developer will work on the single app simultaneously [3].

Popular programming languages like JavaScript, Python, Ruby, PHP, Java, C# and swift work based on MVC framework that are used for web or mobile application development. (Online Wikipedia).

When to use MVC pattern

MVC pattern give idea of separation among the model, view and controller classes within the app. MVC help to implement a test driven development means implement automated test case before write the code or logic.

If require to develop application with the enough stimulating for client side for any particular language then no need to use MVC. The following characteristics that will help us whether to use MVC architecture in our app or not:

1) Our app needs communication on the backend
2) Our app has features which result is not to reload.
3) Data manipulate mostly on the client side compare to server side
4) Some of the data represent is being delivered in different ways on single page.

Advantages of MVC architecture

Faster development process: MVC support the parallel development. If MVC model is used to web application or as mobile application then it is possible that one programmer can work on view, another programmer can work on controller and so on. We can say MVC model can be completed two time faster compare to other development patterns[4].

Multiple view ability: In our era demand is increased day by day of MVC because you can create multiple view for a model. And code duplication is low because it display separate data and business logic from the display.

Support for asynchronous technique: MVC support the asynchronous technique which helps developers to develop an application that loads very fast.

Modification does not affect the entire model: This feature explain that you make frequent changes in your application like color, font, layout and adding new device support for mobile phones or tablet.

Model return without formatting: The same component can be used and called for use with any UI.

SEO friendly: MVC platform supports the development of SEO friendly web pages for particular web development. This features is not allow to mobile app development.

Tools and technologies used with MVC

There are many tools and technologies which can be used to developed web or mobile application with the use of MVC architecture [7]. Depends upon requirement and interest of developer they can use any of tools to develop an application. Here is present tools and technologies as below:

Tools:

1) MySQL server – relational database management server to maintain the DB [10]
2) Net-Beans – Integrated Development Environment provide to develop different types of application.[5]
3) Glassfish Server: Java EE application server [6].
4) Microsoft Visual Studio: Includes almost everything needed to develop, test and debug web applications.
5) XCode: iOS MVC pattern is commonplace in iOS development.[9]

Technologies:

1) HTML, CSS, AJAX and Jquery
2) Servlet and JSP
3) EJB
4) JSTL
5) JDBC
6) ASP.NET
7) Knockout
8) Angularjs
9) Underscore.js
10) Django
11) Rails
12) Laravel
13) Spring
14) React
15) Vue
16) Ember
17) Backbone

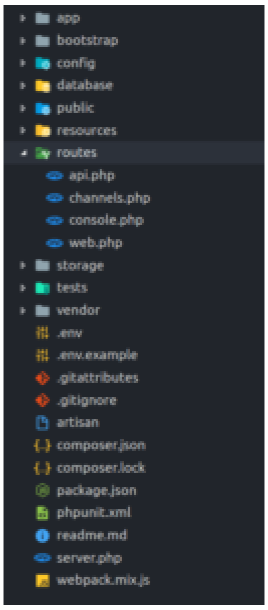
Implementation using Laravel

Laravel is a PHP based web framework. Laravel was created to make it easier for developers to get started on PHP programming[11].

How Laravel requests work in framework

Before driving into how laravel programming implements MVC let us see the work how requests are handled in Laravel.

Whenever you create a new project in laravel, (you can create one by running the command laravel new project name), the project has the belwo structure:



```
app
bootstrap
config
database
public
resources
routes
    api.php
    channels.php
    console.php
    web.php
storage
tests
vendor
.env
.env.example
.gitattributes
.gitignore
artisan
composer.json
composer.lock
package.json
phpunit.xml
readme.md
server.php
webpack.mix.js
```

There file in the routes directory called web.php file. This file is where you handle the requests when users visit your application. The file looks like this:

```
<?php
  /*
           | Web Routes
  */
  Route::get('/', function () {
    return view('welcome');
  });
```

In above file, you can route URLs to controllers in your application, for example, what happens when a user goes to 'yourwebsite.com'.

To create a new project run the command below in terminal:

```
 $ laravel new product-store
```

To see the starter app at work, you need to update the APP_URL in your .env file to http://localhost:8000 then run the below command in your terminal:

```
$ php artisan serve
```

All artisan commands for a project of laravel have to run inside the root of the Laravel project so you have to cd there first before running the command.

This will run your application and you can visit it on 127.0.0.1:8000/ default path. You should see a welcome view in your browser. To stop the server press ctrl + c on keyboard.

In our sample database, a product will have the following properties:

- Name (name) – Name of product.
- Short Description (description) – Short description of a product.

- Count (count) – The number of products available.
- Unit Price (price) – How much a single product costs.

To create a model, run the command in your terminal:

```
$ php artisan make:model Product
```

When you run above command, Laravel will create a Product.php file authomatically in the app directory. This will be a PHP class with the name Product and it will be the model for our products table in the database.

In the Product model, add the $fillable property as shown below:

```
<?php
    namespace App;
    use Illuminate\Database\Eloquent\Model;
    class Product extends Model {
        protected $fillable = [
            'name',
            'count',
            'price',
            'description',
        ];
    }
```

Migrations can be used to make managing databases easy and predictable. To create a migration, run the following in your terminal:

```
$ php artisan make:migration create_products_table
```

When the command is executed, we should see a new *_create_products_table.php file in the database/migrations directory and we can edit it to have our products table schema like this:

```
  <?php
    [...]
    class CreateProductsTable extends Migration
    {
        public function up()
        {
            Schema::create('products', function (Blueprint $table) {
                $table->increments('id');
                $table->string('name');
                $table->text('description');
                $table->integer('count');
                $table->integer('price');
                $table->softDeletes();
                $table->timestamps();
            });
        }
        public function down()
        {
            Schema::dropIfExists('products');
        }
    }
```

Next copy the .env.example file in the root of your project to .env and then in the copied file, change the following lines:

```
DB_CONNECTION=mysql
DB_DATABASE=homestead
 DB_USERNAME=username
 DB_PASSWORD=password
```

With

```
DB_CONNECTION=sqlite
DB_DATABASE=/full/path/to/database.sqlite
```

That is all for our database setup. Now to run the migrations using below command.

```
$ php artisan migrate
```

Before running the command, you need to have set up your database and set the connection details in your .env file in the root of the project.

Now work with controller

Laravel resource routing assigns the CRUD routes to a controller with a single line of code.

```
$ php artisan make:controller ProductController -r
```

The -r flag makes it a resource controller and thus creates all the methods required for CRUD operation.

When the command is execute then, Laravel will create a new file in the app/Http/Controllers directory called ProductController.php.

Before we start adding logic to the controller, go to the routes/web.php file and add the below route:

```
 Route::resource('/products', 'ProductController');
```

This tells Laravel to create all the routes necessary for a resource controller and map them to the ProductController class. Now let's go to the controller file and update the methods in them with the following logic:

The create method:

```
 public function create()

   {

       return view('createproduct');

   }
```

The store method:

```
public function store(Request $request) {

        \App\Product::create([

          'name' => $request->get('name'),

          'description' => $request->get('description'),

          'price' => $request->get('price'),

          'count' => $request->get('count'),

        ]);

        return redirect('/products');

   }
```

The store method is called when a user sends a POST HTTP request to the /products endpoint. This logic above gets the data from the sent request and stores it in the DB using the Product model.

The index method:

```
 public function index()

   {

       $products = \App\Product::all();

       return view('viewproducts', ['allProducts' => $products]);

   }
```

The index method is called when the products route is loaded with a GET HTTP method.

To keep the article short, we will limit the controller logic to these three methods. Now create the views that are loaded by the controller methods above.

Now work with views

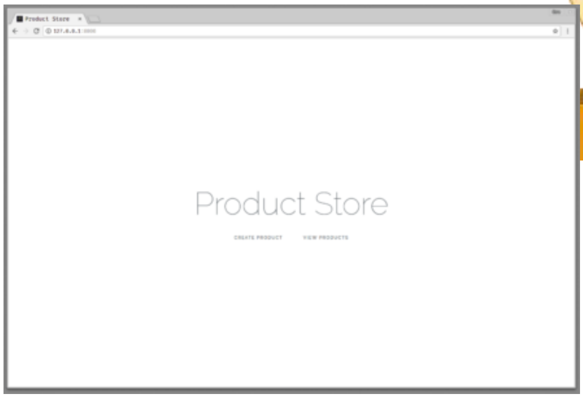
In Laravel project, all the views are stored in the resources/views default directory. Your views usually store the HTML of your page and are the presentation layer of the MVC architecture.

Now create the home page view. Update the welcome.blade.php file in the resources/views directory to include the below code inside the body tag of the existing HTML:

```
  [...]

    <div class="flex-center position-ref full-height">

      <div class="content">

        <div class="title m-b-md">Product Store</div>

        <div class="links">

          <a href="{{ config('app.url')}}/products/create">Create Product</a>

          <a href="{{ config('app.url')}}/products">View Products</a>

        </div>

      </div>

    </div>

    [...]
```

Laravel uses Blade as it's templating engine. Blade is pretty much HTML but with some injectable PHP-like syntax.

If you go back to the routes/web.php you see it stated that the welcome view should be rendered to the user when the / is visited. If you visit the webpage URL http://127.0.0.1:8000/ you will see this page:



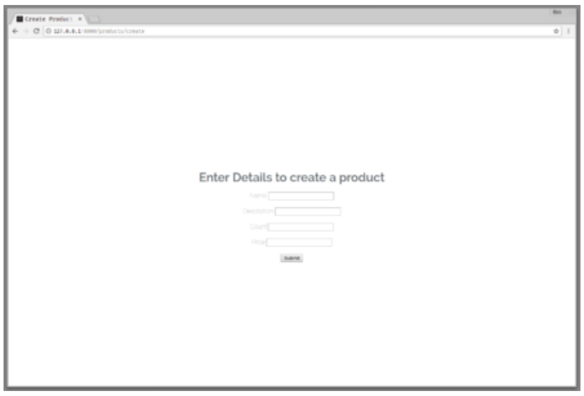
Next, let's make the 'Create Product' view. Create a createproduct.blade.php file in the resources/views directory of our project. In there add the following:

```
<!doctype html>

    <html lang="{{ app()->getLocale() }}">

    <head>

      <title>Create Product | Product Store</title>

      <!-- styling etc. -->
```

```
</head>

<body>

  <div class="flex-center position-ref full-height">

    <div class="content">

      <form method="POST" action="{{ config('app.url')}}/products">

        <h1> Enter Details to create a product</h1>

        <div class="form-input">

          <label>Name</label> <input type="text" name="name">

        </div>

        <div class="form-input">

          <label>Description</label> <input type="text" name="description">

        </div>

        <div class="form-input">

          <label>Count</label> <input type="number" name="count">

        </div>

        <div class="form-input">

          <label>Price</label> <input type="number" name="price">

        </div>

        <button type="submit">Submit</button>

      </form>

    </div>

  </div>

</body>

</html>
```

This view above is a simple form that collects and submits requests to create products. When the form is submitted, a POST request is made to the /products route of the application which is handled by the store method in our ProductController.

Here is how the /products/create route will look after adding the view and visiting the route:



Enter Details to create a product

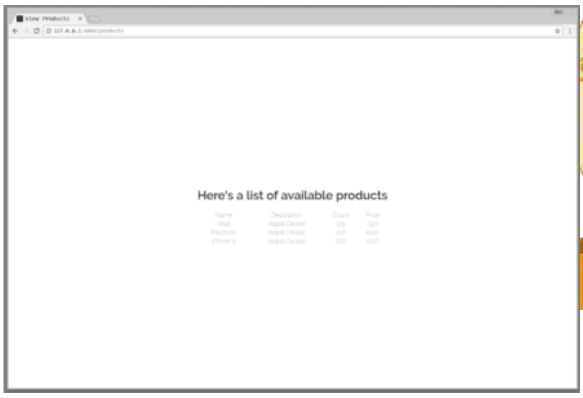
The next view we want to add is the viewproducts.blade.php view. Create that file in the resources/views directory and add the following code to the file:

```
<!doctype html>
  <html lang="{{ app()->getLocale() }}">
  <head>
    <title>View Products | Product Store</title>
    <!-- Styles etc. -->
  </head>
  <body>
    <div class="flex-center position-ref full-height">
      <div class="content">
        <h1>Here's a list of available products</h1>
        <table>
          <thead>
            <td>Name</td>
            <td>Description</td>
            <td>Count</td>
            <td>Price</td>
          </thead>
          <tbody>
            @foreach ($allProducts as $product)
              <tr>
                <td>{{ $product->name }}</td>
```

```
                        <td class="inner-table">{{ $product->description }}</td>

                        <td class="inner-table">{{ $product->count }}</td>

                        <td class="inner-table">{{ $product->price }}</td>

                    </tr>

                @endforeach

            </tbody>

        </table>

    </div>

  </div>

</body>

</html>
```

In the view above, the data that was sent from the controller, $allProducts, is iterated on and displayed to the user.

Now, if you visit the /products route you should see something like this:


Here's a list of available products

Name	Description	Count	Price
iPad	Apple Device	[illegible]	320
Macbook	Apple Device	100	1200
iPhone X	Apple Device	200	1000

In this sample example of MVC pattern work using Laravel, we considered how MVC works and how Laravel implements it. We considered why you should use MVC and how to implement it in a real-world Laravel application.

Disadvantages of MVC architecture

- It is hard to understand of the MVC architecture
- Must have strict rules on methods
- The complexity is high to develop the applications using this pattern.
- Not right suitable for small applications which has adverse effect in the application’s performance and design.
- In terms of servlet and JSP, both often contains business logic and presentation tier.
- The isolated development process by UI authors, business logic authors and controller authors may lead to delay in their respective modules development.

Limitation of Survey

In this paper we have discuss about MVC pattern architecture and also discuss about the three main components elements Model, View and Controller. In this paper provide not deep details of all elements.

Conclusion

Portable and scalable web or mobile application developed in MVC pattern architecture. MVC architecture is parallel development. So many developer work with simultaneously. The cost of setup MVC architecture is very high compare to non-MVC architecture. When very small project may not be used MVC patterns while very large project or any MNC project or long term project generally use MVC pattern architecture because MVC architecture will proceed and development speed is faster compare to non-MVC architecture.

REFERENCES

[1] Gupta P, Govil MC (2010) MVC Design pattern for the multi framework distributed applications using XML, spring and struts framework. International Journal on Computer Science and Engineering 2(4): 1047- 1051.

[2] Web Development in .Net, [ONLINE] Available at https://www.dotnetcurry.com/aspnet-core/1501/web-development-in-dotnet [Accessed on 22nd April,2020].

[3] Selfa DM, Carrillo M, Boone MDR (2006) A database and web application based on MVC architecture. 16th International Conference on Electronics, Communications and Computers (CONIELECOMP'06), p. 48.

[4] Benefits of MVC model for effective web application development, [ONLINE] Available at: https://www.brainvire.com/six-benefits-of-using-mvc-model-for-effective-web-application-development/ [Accessed on 23rd April, 2020].

[5] Benson C, Prove MM, Mzourek J (2004) Professional usability in open source projects. CHI '04 Extended Abstracts on Human Factors in Computing Systems, pp. 1083-1084.

[6] Moulton B, Chaczko Z, Karatovic M (2009) International journal of digital content technology & its Applications GYEONGJU. The International Association for Information, Culture, Human and Industry Technology.

[7] Tools and Technologies for web application development, [ONLINE] Available at: http://www.lucemorker.com/blog/best-tools-technologies-for-net-based-web-application-development [Accessed on 22nd April, 2020].

[8] Shu-qiang H, Huan-ming Z (2008) Research on improved MVC design pattern based on struts and XSL. 2008 International Symposium on Information Science and Engineering, pp. 451-455.

[9] Model View Controller in iOS Modern Approach, [ONLINE] Available at : https://www.raywenderlich.com/1000705-model-view-controller-mvc-in-ios-a-modern-approach [Accessed on 23rd April, 2020].

[10] Ahmed M, Uddin MM, Azad MS, Haseeb S (2010) MySQL performance analysis on a limited resource server. SpringSim 10 Proceedings of the 2010 Spring Simulation Multiconference, p. 1.

[11] How laravel implement MVC, [ONLINE] Available at :https://blog.pusher.com/laravel-mvc-use/ [Accessed on 29th April, 2020].