MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

DEPARTMENT OF ICT

Lab Report No : 05

Course Code : ICT-3210

Course Title : Software Engineering Lab

Lab Report name : Model View Controller

Submitted by
Md. Faruk Hosen
ID : IT-17035
Session : 2016-2017
Year : 3rd Semester : 2nd

Submitted to
Mr. Tanvir Rahman
Lecturer,
Department of ICT,MBSTU
Santosh,Tangail-1902

Date of Submission : 31 October 2020

Model View Controller

1.What is MVC (Model View Controller) ?

Ans: **MVC :** MVC is a software architecture pattern and it stands for Model View Controller. MVC is a software architecture that separates domain / application/business…logic from the rest of the user interface. It does this by separating the application into three parts: the model, the view, and the controller.

The **model** manages fundamental behaviors and data of the application. It can respond to requests for information, respond to instructions to change the state of its information, and even notify observers in event-driven systems when information changes. This could be a database or any number of data structures or storage systems. In short, it is the data and data-management of the application. Models to interact with your database and retrieve your objects' information.

The **view** effectively provides the user interface element of the application. It'll render data from the model into a form that is suitable for the user interface. Views to render pages

The **controller** receives user input and makes calls to model objects and the view to perform appropriate actions. Controllers to handle user requests and retrieve data. All in all, these three components work together to create the three basic components of MVC.

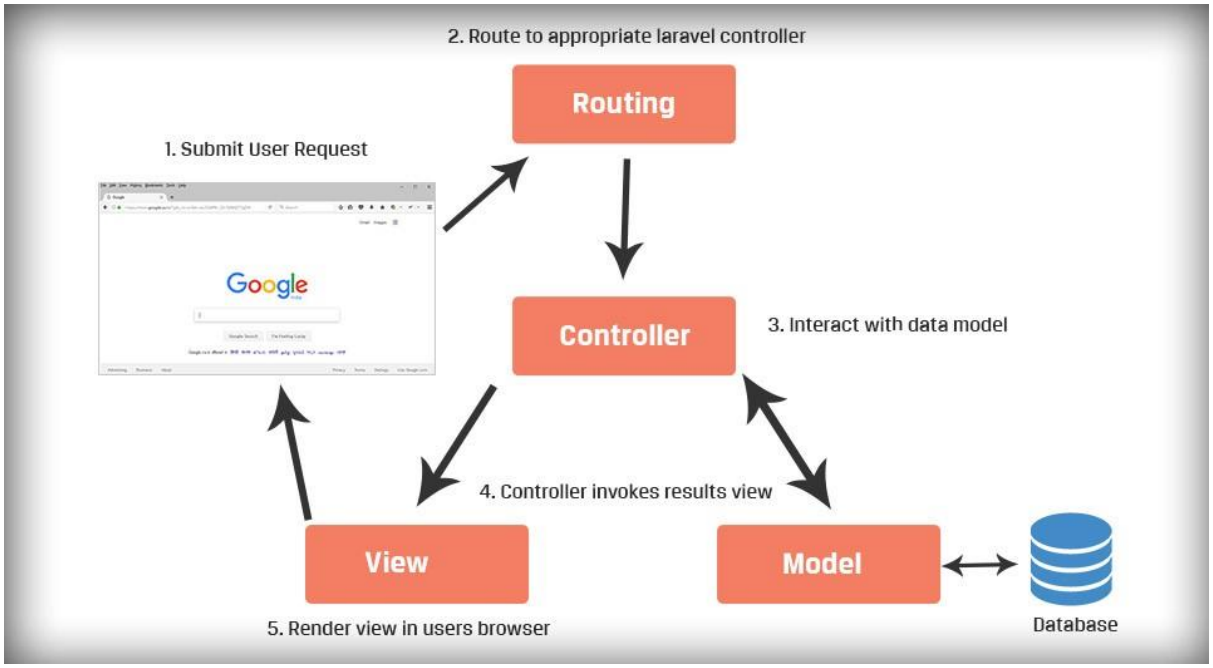
Let's look at a simple scenario.

If you go to an e-commerce website, the different pages you see are provided by the View layer. When you click on a particular product to view more, the Controller layer processes the user's action. This may involve getting data from a data source using the Model layer. The data is then bundled up together and arranged in a View layer and displayed to the user. Rinse and repeat.

2. Explain the MVC structure using figure ?

Ans : We have a structure that looks like this:

Here**, routes** are used to map URLs to designated controller actions.



3.Why use MVC ?

Ans : When we have to work with codebases that have no architecture, it will become extremely grueling, especially if the project is big and we have to deal with unstructured code laying everywhere. Using MVC can give your code some structure and make it easier to work with.

On a more technical note, when we build using the MVC architecture, we have the following strategic advantages:

i. Splitting roles in your project are easier.
ii. Structurally 'a-okay'.
iii. Responsibility isolation.
iv. Full control of application URLs.

4. How MVC is implemented in Laravel applications ?

Ans : Let's take a look at how Laravel uses MVC during development. To do this, let's create a sample Laravel project that dealing with cars.

The Model :

run the following command to generate a new Car model:

```
PS C:\Users\Md Faruk Hosen\Desktop\App\CreditFeePaymentSystem> php artisan make:model Car
```

Models are all stored in the main app directory, so that command will generate an app/Car.php model file with the following code:

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Car extends Model
{
    //
}
```

Due to Laravel's built-in model functionality, just by creating an empty model class, Laravel will assume that this model is associated with a database table named cars.

And, actually, by supplying that --migration option when generating the model, Laravel also generated a database migration file for creating that cars database table. The migration file is located at [timestamp]_create_cars_table.php.

All you have to do now is use Laravel's Schema builder documentation to complete the migration file. So you could define a few additional columns to, say, store the cars' make, model, and production date:

```php
Schema::create('cars', function (Blueprint $table) {
    $table->increments('id');
    $table->string('make');
    $table->string('model');
    $table->date('produced_on');
    $table->timestamps();
});
```

And then you can run the migration to create the cars table using the following Artisan command:

```
PS C:\Users\Md Faruk Hosen\Desktop\App\CreditFeePaymentSystem> php artisan migrate
```

The Controller :

run the following command to create controller :

```
PS C:\Users\Md Faruk Hosen\Desktop\App\CreditFeePaymentSystem> php artisan make:controller CarController
```

That will generate an app/Http/Controllers/CarController.php controller file with the following code:

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Http\Requests;
use App\Http\Controllers\Controller;
class CarController extends Controller
{
    public function index()
    {
    }
    public function create()
    {
    }
    public function store()
    {
    }
    public function show($id)
    {

    }
    public function edit($id)
    {
    }
    public function update($id)
    {
    }
    public function destroy($id)
    {
    }
}
```

The View :

Laravel view files are all stored in the resources/views folder. And they can be organized into subfolders within that directory.

In the previous step, we passed the view function the view name cars.show. That tells Laravel to look for a view file located in a subfolder named cars (within the main resources/views directory) named show.blade.php.

Laravel view files utilize the **Blade templating engine**, and hence are named .blade.php.

So, to finish the implementation of this Show Car page, we can create the resources/views/cars/show.blade.php view file with the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Car {{ $car->id }}</title>
  </head>
  <body>
    <h1>Car {{ $car->id }}</h1>
    <ul>
      <li>Make: {{ $car->make }}</li>
      <li>Model: {{ $car->model }}</li>
      <li>Produced on: {{ $car->produced_on }}</li>
    </ul>
  </body>
</html>
```

Since we passed the Car object to the view — back in the show action of the controller — with the array key car, we can access it in the view via a variable of the same name, $car.

Objects retrieved via a model are instances of that model class.

And, as you can see, the Car object's values can be accessed using the same names as its associated cars database table's column names.

That is how, we can build a MVC web application using Laravel.

Conclusion :

In this lab, we learned how MVC works and how Laravel implements it. We learned why we should use MVC and how to implement it in a real-world Laravel application.