COLLEGE OF COMPUTING.

DEPARTEMENT OF COMUTER SCIENISE.

SUBJECT- SELECTED TOPICS IN CS.

PREPARED BY- CHILENEWU HAILEMARIAM.

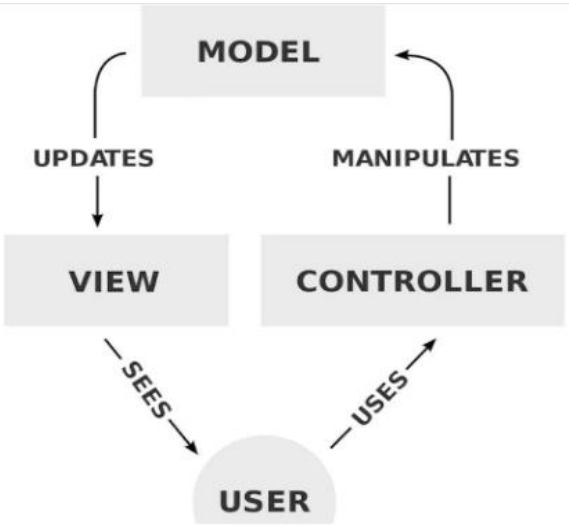
ID DBUE/774/11

SUBMITTED T0 GIRMACHEW G

SUMBISSION DATE 28/01/2023

Contents

1.What Is MVC



MVC :-stands for Model View Controller structure. This is a very useful development structure and also very popular in the market. Laravel has these features and this is why Laravel is one of the most used frameworks now.

What is Model in Laravel

Model is where we perform the database-related operations. Like, insert, delete, edit, update query, etc. The model folder exists in the App folder in Laravel 8. Here you have to create models corresponding to the database tables.

Like if your table name is users, where you want to do operations through a model then the model name will be User.

What is View

The view is where we store our front-end templates. These are called blade templates.

What is Controller

The controller is the central part of the MVC model. The controller controls the model and views. Also, this is responsible for the Application logic.

The controller takes the user input values from the view and passes the values to the model. Then also return the data from the model to view

2.Laravel Routing

Routing is one of the essential concepts in Laravel. Routing in Laravel allows you to route all your application requests to their appropriate controller. The main and primary routes in Laravel acknowledge and accept a URI (Uniform Resource Identifier) along with a closure, given that it should have to be a simple and expressive way of routing.

Creates Routes in Laravel:

All the routes in Laravel are defined within the route files you can find in the routes sub-directory. These route files get loaded and generated automatically by the Laravel framework. The application's route file gets defined in the app/Http/routes.php file. The general routing in Laravel for each of the possible requests looks something like this:

http://localhost/

```
Route:: get ('/', function () {
   return 'Welcome to index';
});
```

http://localhost/user/dashboard

```
Route:: post('user/dashboard', function () {
   return 'Welcome to dashboard';
});
```

http://localhost/user/add

```
Route:: put('user/add', function () {
//
});
```

http://localhost/post/example

```
Route:: delete('post/example', function () {
//
});
```

Routing Mechanism in laravel

The routing mechanism takes place in three different steps:

1. First of all, you have to create and run the root URL of your project.
2. The URL you run needs to be matched exactly with your method defined in the root.php file, and it will execute all related functions.
3. The function invokes the template files. It then calls the view() function with the file name located in resources/views/, and eliminates the file extension blade.php at the time of calling.

Example:

app/Http/routes.php

```
<?php
Route:: get ('/', function () {
   return view('laravel');
});
```

resources/view/laravel.blade.php

```
<!DOCTYPE html>
<html>

    <head>
        <title>Laravel5 Tutorial</title>
    </head>

    <body>
        <h2>Laravel5 Tutorial</h2>
        <p>Welcome to Laravel5 tutorial.</p>
    </body>

</html>
```

Route Parameters

In many cases, a situation arises in your application when you have to capture the parameters sent through the URL. To use these passed parameters effectively in Laravel, you must change the routes.php code.

Laravel provides two ways of capturing the passed parameter:

- Required parameter
- Optional Parameter

Required Parameter

You sometimes had to work with a segment(s) of your project's URL (Uniform Resource Locator). Route parameters are encapsulated within {} (curly-braces) with alphabets inside. Let us take an example where you have to capture the customer's ID or employee from the generated URL.

Example:

```
Route :: get ('emp/{id}', function ($id) {
    echo 'Emp '.$id;
});
```

Optional Parameter

Many parameters do not remain present within the URL, but the developers had to use them. So such parameters get indicated by a "?" (question mark sign) following the parameter's name.

Example:

```
Route :: get ('emp/{desig?}', function ($desig = null) {
    echo $desig;
});
```

```
Route :: get ('emp/{name?}', function ($name = 'Guest') {
    echo $name;
});
```

3.Larave Migration & Relationships

What is Laravel Migration?

Laravel Migration is an essential feature in Laravel that allows you to create a table in your database. It allows you to modify and share the application's database schema. You can modify the table by adding a new column or deleting an existing column.

Why do we need Laravel Migration?

Suppose we are working in a team, and some idea strikes that require the alteration in a table. In such a case, the SQL file needs to be passed around, and some team member has to import that file, but team member forgot to import the SQL file. In this case, the application will not work properly, to avoid such situation, Laravel Migration comes into existence.

Laravel Migration allows you to add a new column or delete the records in your database without deleting the records that are already present.

What is Laravel Relationship

An **Eloquent relationship** is a very important feature in Laravel that allows you to relate the tables in a very easy format.

Eloquent relationships are defined as functions on your eloquent model classes. Since, like Eloquent models themselves, relationships also serve as powerful query builders, defining relationships as functions provides powerful method chaining and querying capabilities.

Relationship Types

- ✓ **One to one relationship**

One to one relationship provides the one-to-one relationship between the columns of different tables. For example, every user is associated with a single post or maybe multiple posts, but in this relationship, we will retrieve the single post of a user. To define a relationship, we need first to define the **post()** method in User model. In the post() method, we need to implement the **hasOne()** method that returns the result.

- ✓ **Polymorphic relationship**

    - **One-to-many (Polymorphic)**

The Polymorphic relationship is similar to the one-to-many relationship. When a single model belongs to more than one type of model on a single association is known as one-to-one polymorphic relationship. For example, if we have three tables, posts, users, and photo table, where photo table represents the polymorphic relation with the users and posts table.

    - **Many-to-many polymorphic relationship**

In a many-to-many polymorphic relationship, a target model consists of unique records that are shared among the various models. For example, a tag table shares the polymorphic relation between the videos and the posts table. A tag table consists of the unique list of tags that are shared by both the tables, videos, and the posts table.

- ✓ **Inverse Relation**

Inverse relation means the inverse of the one-to-one relationship. In the above, we have retrieved the post belonging to a particular user. Now, we retrieve the user information based on the post. Let's understand this through an example.

4.Blade template engine

Blade Template

The Blade is a powerful templating engine in a Laravel framework. The blade allows to use the templating engine easily, and it makes the syntax writing very simple. The blade templating engine provides its own structure such as conditional statements and loops. To create a blade template, you just need to create a view file and save it with a .blade.php extension instead of .php extension. The blade templates are stored in the /resources/view directory. The main advantage of using the blade template is that we can create the master template, which can be extended by other files.

Why Blade template?

Blade template is used because of the following reasons:

- **Displaying data**
  If you want to print the value of a variable, then you can do so by simply enclosing the variable within the curly brackets.
  Syntax

  {{$variable}};

In blade template, we do not need to write the code between **<?php echo $variable; ?>**. The above syntax is equivalent to **<?= $variable ?>**.

- **Ternary operator**
  In blade template, the syntax of ternary operator can be written as:

{{ $variable **or** 'default value'}}

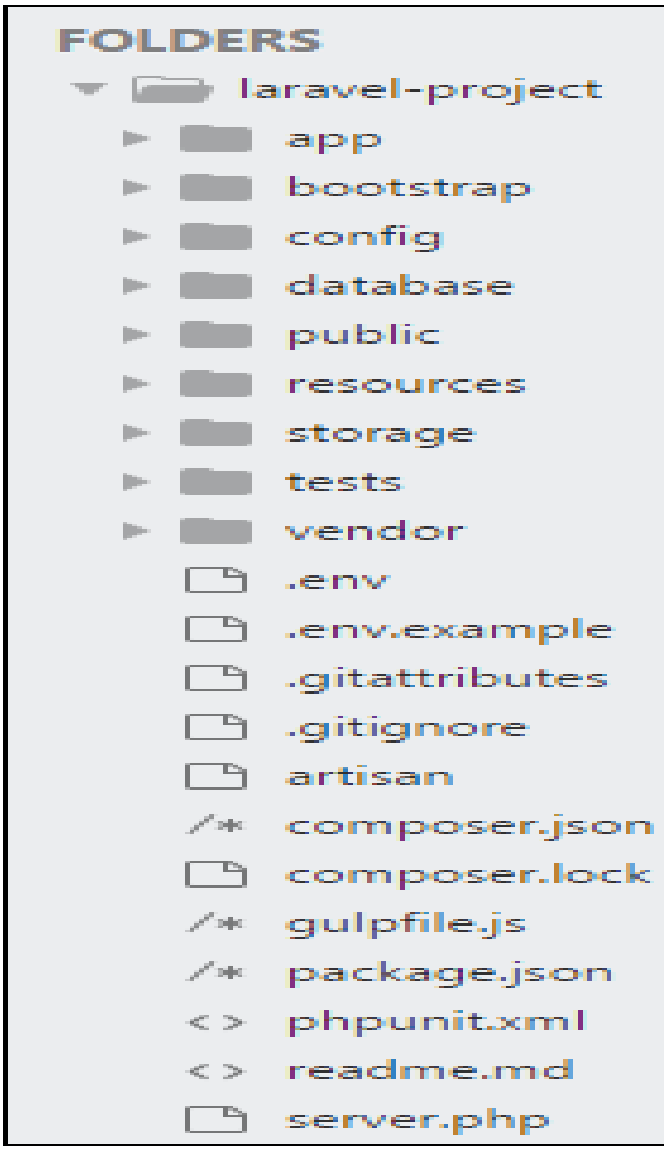
The above syntax is equivalent to **<?= isset($variable) ? $variable : ?default value? ?>**

- ✓ Blade template engine also provides the control statements in laravel as well as shortcuts for the control statements.
- ✓ Blade template provides **@unless** directive as a conditional statement. The above code is equivalent to the following code:
- ✓ The blade templating engine also provides the @hasSection directive that determines whether the specified section has any content or not.
- ✓ The blade templating engine provides loops such as @for, @endfor, @foreach, @endforeach, @while, and @endwhile directives. These directives are used to create the php loop equivalent statements.

5.Directives

The application structure in Laravel is basically the structure of folders, sub-folders and files included in a project. Once we create a project in Laravel, we get an overview of the application structure as shown in the image here.

The snapshot shown here refers to the root folder of Laravel namely **laravel-project**. It includes various sub-folders and files. The analysis of folders and files, along with their functional aspects is given below –



Sometimes in **laravel views**, you have to write certain logic: checking for an admin, or a regular user, checking their rights. However, thanks to the built-in directive creation capabilities, this process can be made more convenient by taking all the validation logic into directives.

If you are new to learning Laravel, luckily enough, there are already quite a **few built-in blade templating directives** out there. The creators of the framework took care of solving the most demanded tasks and programmed them into the core of the framework in the form of directives.

References

1 What is Model View Controller (MVC) In Laravel 9(https://shoyeblaravel.hashnode.dev/what-is-model-view-controller-mvc-in-laravel-8-explained

2 Laravel Routing: A Comprehensive Guide to Defining and Managing Routes in Your Laravel App (https://www.w3schools.in/laravel /routing)

3 Laravel Relationship - Javatpoint(https://www.javatpoint.com/laravel-relationship)

4 Useful Laravel Blade Directives - LaravelProject(https://laravelproject.com/useful-laravel-blade-directives)