

***DEBRE BIRHAN UNIVERSITY***

**COLLEGY OF COMPUTING**

**DEPARTMENT OF COMPUTER SCIENCE**

**Selected topic computer science**

**INDIVIDUAL ASSIGNMENT**

**EXTENSION PROGRAM**

NAME : GEBEYEHU SHEWAFERA

ID NO:DBUE/781/11

2015 E.C

1 Explain MVC of laravel

MVC is an acronym for ‘Model View Controller’. It represents architecture developers adopt when building applications. With the MVC architecture, we look at the application structure with regards to how the data flow of our application works

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.

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.  
The controller receives user input and makes calls to model objects and the view to perform appropriate actions.

All in all, these three components work together to create the three basic components of MVC.  
– [**Bob, Stack Overflow**](https://softwareengineering.stackexchange.com/questions/127624/what-is-mvc-really)

2 Explain Routing

outing is one of the essential concepts in Laravel. Routing in Laravel allows you to route all your application requests to its 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. **In this article we will discuss a brief concept of Laravel routing.**

**What is Laravel routing?**

Using Routing you can create a request URL for your application. you can design set of HTTP request like POST Request, GET Request, PUT Request and DELETE Request using routing in Laravel 8.

In the Laravel routes folder contain two routes’ files, one is api.php, which is used to API routes and another is web.php, which is used for routes for the web application.

Routing in Laravel includes various methods. **We explore each one with an example code step by step.**

• Basic Routing

• Route Parameters

• Named Routes

• Route Groups

• Route Model Binding

**Basic routing**

Most of the routes for your application will be defined in the app/routes.php file. The simplest Laravel routes consist of a URI and a Closure callback.

3 Explain Migration and Relationship

A Laravel Migrations are the version control method for the database. It is done to allow the team to modify the database as well as share it across domains and platforms. Migrations are an integral part of the Laravel procedure and are usually done in conjecture with the Laravel schema builder. This helps to build easily the database schema. Laravel Migrations are an essential part of the Laravel architecture. One of the features which helps in the smooth migration of the database is the Laravel Schema Façade. This allows the database to be modified and manipulated across all the supported database systems of Laravel.

### How to Generate Migration?

One of the most pertinent question that hence arises is how to generate the migration?

The easiest way is to use the following query:

php artisan make:migration create\_users\_table

This allows any new migrations to be placed in the said database or migration directory. It is also important to know that each migration bears a timestamp. This information allows the determination of the order in which the migration has taken place, by Laravel.

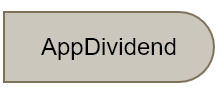
One may also use the “table” and “create” options.

This helps to locate the identity of the table and whether the migration will be building a table a new or not.

The query hence will look like this:

***B Laravel Relationships Example****is today’s main topic. In this tutorial, I will define two tables and establish the relationships* between the tables. We will explore **hasMany**() relationship in this example from scratch. Eloquent Relationships are very compelling and save you lots of time. So let us get started this tutorial of **Laravel Relationships Example.**

5.9M



Defining relationships in Laravel is something that every Laravel developer probably has done more than once.

Advertisement

But there are a lot of issues that you can run into when trying to implement a relationship.

Since there are different types of relationships, how do you know which one to pick? And how can you use the full potential of relationships when it comes to down to querying the models.

## ****Laravel Relationships Example****

Database tables are often related to one another. For example, a blog post may have many comments, or an order could be related to the user who placed it. Eloquent makes managing and working with these relationships efficiently and supports several different types of relationships.

* One To One
* One To Many
* Many To Many
* Has One Through
* Has Many Through
* One To One (Polymorphic)
* One To Many (Polymorphic)
* Many To Many (Polymorphic)

## ****#Defining Relationships in Laravel****

Eloquent relationships are defined as methods on your Eloquent model classes.

Since, like Eloquent models themselves, relationships also serve as robust query builders, defining relationships as methods provide powerful method chaining and querying capabilities.

For example, we may chain additional constraints on this posts relationship.

Now, let’s dive into the example

4 Explain Blade template engine

Laravel Blade Template Engine Explained — Part-I



Blade

Laravel is one of the best PHP framework which is highly acknowledged for its **inbuilt lightweight templates** that help you **create amazing layouts using dynamic content seeding**.

Pre-requisites: I assume you are familiar with installing and running Laravel application and are familiar with creating Routes.

The templates of the Laravel framework are innovatively designed to create a simple layout with distinctive sections.

When compared to other templating engines, Blade is unique in the following ways −

* We can use plain PHP code in views. The blade views thus designed, are compiled and cached until they are modified.

Remember: you can clear the compiled and cached view files by running this command in the terminal.

php artisan view:clear

Blade view files use the .blade.php file extension and are typically stored in the resources/views directory.

# [Inheritance](https://laravel.com/docs/5.8/blade#template-inheritance)

Two of the primary benefits of using Blade are template inheritance and sections. We can define a blade page as a combination of layout and sections.

Since most of the general web applications will have the same layout across the web pages. In this example, we will examine the “master layout” with sections like “Top nav”, “Sidenav” and the “body content”.

I'm using MaterializeCss in this example as it has some useful prebuilt components like sidenav.

Lets start:

Blade view files use the .blade.php file extension and are typically stored in the resources/views directory.

Let us create the following blade files in your project’s resource/views directory.

1. master.blade.php
2. topnav.blade.php
3. sidenav.blade.php
4. home.blade.php

5 Directives

**Laravel Blade Directives** are syntactic sugar functions that hide the underlying complex and ugly code. This make code more readable and clear.

Blade includes lots of built-in **directives**. But, this tutorial will help you with laravel blade directives that you’ll often reach out during your time in laravel.

## How to Create Custom Blade Directive in Laravel?

Follow the following steps to create custom blade directive in laravel apps:

* Step 1: Create Custom Blade Directive
* Step 2: Create Route
* Step 3: Create Blade File
* Step 4: Run Development Server

### Step 1: Create Custom Blade Directive

In this step, you need to declare custom blade directive in app service provide file.

So, Navigate to **app/Providers/**directory and open**AppServiceProvider.php**file. Then add the following code into your **AppServiceProvider.php** file:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | <?php    namespace App\Providers;    use Illuminate\Support\ServiceProvider;  use Illuminate\Pagination\Paginator;  use Blade;    class AppServiceProvider extends ServiceProvider  {      /\*\*       \* Register any application services.       \*       \* @return void       \*/      public function register()      {        }      /\*\*       \* Bootstrap any application services.       \*       \* @return void       \*/      public function boot()      {          Paginator::useBootstrap();            Blade::directive('nl2br', function ($string) {              return "<?php echo nl2br($string); ?>";          });      }  } |

**Recommended:-** [Laravel Livewire Load More OnScroll Tutorial Example](https://www.tutsmake.com/laravel-8-livewire-load-more-onscroll-tutorial-example/)

### Step 2: Create Route

In this step, Navigate to **routes** folder and open **web.php**. Then add the following routes into your **web.php** file:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | Route::get('directive', function () {        $body = '';        if(request()->filled('body')){          $body = request()->body;      }        return view('directive', compact('body'));  }); |

### Step 3: Create Blade File

In this step, navigate to **resources/views/livewire** folder and create one blade view files that name **directive.blade.php** file. Then add the following code into your **directive.blade.php** file:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <!DOCTYPE html>  <html>  <head>      <title></title>      <link href="//netdna.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css" rel="stylesheet">  </head>  <body>    <div class="container">      <h1>Laravel - How to create directive - Tutsmake.com</h1>        <form>          <strong>Enter Something:</strong>          <textarea name="body" class="form-control" style="height: 200px"></textarea>            <button type="submit" class="btn btn-success">Submit</button>      </form>        <p>Body:</p>      <p>@nl2br($body)</p>  </div>    </body>  </html> |

**Recommended:-** [Laravel Livewire CRUD with Jetstream Tutorial](https://www.tutsmake.com/laravel-8-livewire-crud-with-jetstream-tutorial/)

### Step 4: Run Development Server

Finally, you need to run the following **PHP artisan serve** command to start your laravel livewire upload file app:

php artisan serve  
  
If you want to run the project diffrent port so use this below command   
  
php artisan serve --port=8080

Reference

1 <https://blog.pusher.com/laravel-mvc-use/>

2 <https://www.bing.com/search?q=directive>

3 https://medium.com/@ajtech.mubasheer/laravel-blade.