Next, install Laravel’s command line installer:

**composer global require "laravel/installer"**

Once that finishes you can create the project by running:

**laravel new links**

Now scaffold out the authentication system by running:

**php artisan make:auth**

Creating a migration will be the first step, and the Laravel Artisan command line tool can help us create that.

**php artisan make:migration create\_links\_table --create=links**

Now, open the file this command created. It will be located at **database/migrations/{{datetime}}\_create\_links\_table.php**

**php artisan make:migrate**

**php artisan make:model Link -- to createw model**

**php artisan make:controller LocalizationController to create controller**

Create file **database/factories/LinkFactory.php and add below code**

$factory->define(App\Link::class, function (Faker\Generator $faker) {

return [

'title' => $faker->name,

'url' => $faker->url,

'description' => $faker->paragraph,

];

});

Next, create the link seeder, so we can easily add demo data to the table:

php artisan make:seeder LinksTableSeeder

**database/seeds/LinksTableSeeder.php**

**Open the LinksTableSeeder.php file that was just created, and in the run method we will utilize the links model factory we created above:**

**public function run()**

**{ factory(App\Link::class, 10)->create(); }**

Open the **database/seeds/DatabaseSeeder.php** and add this to the run method**:**

**$this->call(LinksTableSeeder::class);**

You can now run the migrations and seeds to add data to the table automatically:

php artisan migrate --seed

* **New @method and @csrf Blade Directives**
* **<?php dd($link); die;?> is used to print array in view blades or**

@php

var\_dump($arr);

@endphp

* For foreach

**@foreach ($watchFolder as $key => $value)**

**Key: {{ $key }}**

**Value: {{ $value }}**

**@endforeach**

* **{!! $link->title !!}** is used for raw **{!! Form::open() !!}** or **{{$link->title }}** is used to display content like echo

In Laravel 4 {{{ $var }}} is used to render data by escaping HTML entities that uses PHP htmlentities function to prevent XSS attacks.

{{ $var }} is used to render data without escaping HTML entities.

In Laravel 5 it slightly different you can use either {{ $var }} or {{{ $var }}} to escape data and to render unescaped data you can use {!! $var !!}.

Laravel comes with Blade templating engine which generates HTML based sleek designs and templates. All Laravel views built using Blade are located in resources/views.

1. create layout **in resources/views/layouts/app.blade.php**

use **@yield('content')** to add content in layout.

2. Now in view **file resources/views/submit.blade.php**

add **@extends('layouts.app')** to add layout in view blade file.

**@section('content')**

//write code inside content block

**@endsection**

1. **Middleware acts as a bridge between a request and a response.**

php artisan make:middleware <middleware-name>

Path : **app/Http/Middleware**

**Registering Middleware**

We need to register each and every middleware before using it. There are two types of Middleware in Laravel.

* Global Middleware
* Route Middleware

Middleware parameters may be specified when defining the route by separating the middleware name and parameters with a :. Multiple parameters should be delimited by commas:

Route::put('post/{id}', function ($id) {

//})->middleware('role:editor');