**Laravel Framework**

10/02/2024

**LET PANEL**

<https://www.mykoenig.com/>

Email : Registered Email id

Password : 183539

Short break : 11.00 am - 15 mints

Lunch break : 1.30pm - 45 mints

Short break : 4.00 pm - 15mints

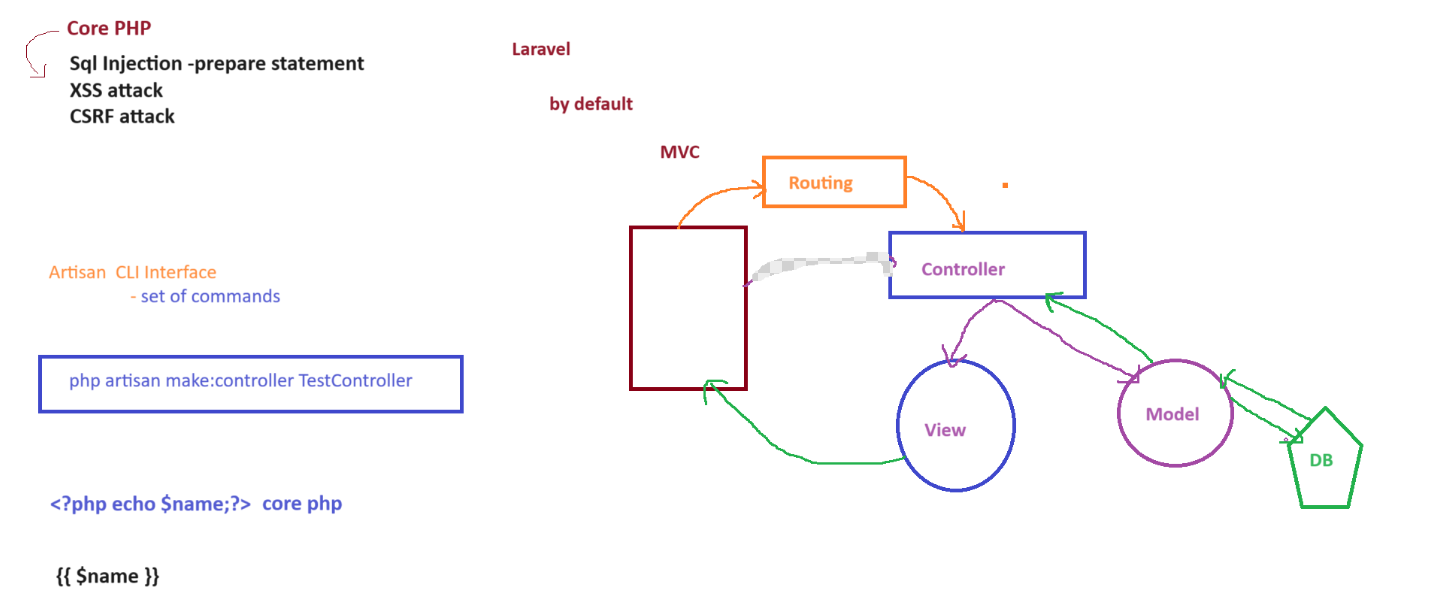
**DAY 1**

Attendance url

[**https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732318**](https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732318)

**Please share your feedback using below link for your experience so far**

[**https://rms.koenig-solutions.com/InterimFeedback.aspx?id=±¹µ¸·¾&frm=5**](https://rms.koenig-solutions.com/InterimFeedback.aspx?id=%C2%B1%C2%B9%C2%B5%C2%B8%C2%B7%C2%BE&frm=5)

****

**Official Docs**

[**https://laravel.com/docs/10.x/readme**](https://laravel.com/docs/10.x/readme)

**SETUP**

1. **XAMPP**

[**https://www.apachefriends.org/**](https://www.apachefriends.org/)

1. **COMPOSER**

[**https://getcomposer.org/**](https://getcomposer.org/)

1. **Install Laravel Application**

**Composer create-project laravel/laravel projectname**

1. **Run the Application**

**cd projectname**

**Open project in visual studio code**

**Open Terminal in vscode**

**php artisan serve**

1. **Open browser : localhost:8000**

**Route and Route Parameter**

****

**# localhost:8000 => Default Url**

**Route::get('/', function () {**

**return 'Welcome TO Laravel';**

**});**

**#localhost:8000/login**

**Route::get('/login',function(){**

**return 'Welcome To Login Panel';**

**});**

**// Example**

**// Route::get('/books',function(){**

**// return 'Book Index Page';**

**// });**

**// Route::get('/books/action',function(){**

**// return 'Book Action Page';**

**// });**

**// Route::get('/books/drama',function(){**

**// return 'Book drama Page';**

**// });**

**// Route::get('/books/comedy',function(){**

**// return 'Book comedy Page';**

**// });**

**// Route::get('/magzine',function(){**

**// return 'magzine Index Page';**

**// });**

**// Route::get('/magzine/technology',function(){**

**// return 'magzine technology Page';**

**// });**

**// Route::get('/magzine/celebrity',function(){**

**// return 'magzine celebrity Page';**

**// });**

**# Problem with above code : repetation**

**# solution : Route Parameter**

**# Required Parameter**

**# optional Parameter**

**# Required Parameter**

**// Route::get('/books',function(){**

**// return 'Book Index Page';**

**// });**

**// Route::get('/books/{bookType}',function($bookType){**

**// return "Book $bookType Page";**

**// });**

**// Route::get('/magzine',function(){**

**// return 'magzine Index Page';**

**// });**

**// Route::get('/magzine/{magzineType}',function($magzineType){**

**// return "magzine $magzineType Page";**

**// });**

**#Optional Parameter**

**// Route::get('/books/{bookType?}',function($bookType=null){**

**// if($bookType=== null){**

**// return 'Book Index Page';**

**// }**

**// return "Book $bookType Page";**

**// });**

**// Route::get('/magzine/{magzineType?}',function($magzineType=null){**

**// if($magzineType===null){**

**// return "magzine Index Page";**

**// }**

**// return "magzine $magzineType Page";**

**// });**

**// -----------------Route Parameter Constraint---------------------**

**// Route::get('/login/{username}/{age}', function($username, $age){**

**// return "Given Username = $username and age = $age";**

**// })->where(['age'=>'[0-9]+']);**

**# Set Constraint GLobally in RouteServiceProvider file**

**Route::get('/login/{username}/{age}', function($username, $age){**

**return "Given Username = $username and age = $age";**

**});**

**RouteServiceProvider.php**

**public function boot(): void**

**{**

**# Set Constraint Globally**

**Route::pattern('age','[0-9]+');**

**}**

**Response**

**View() and redirect()**

**Web.php**

**# localhost:8000 => Default Url**

**Route::get('/', function () {**

**return view('sample');**

**});**

**#pass data to view**

**// Route::get('/sample', function(){**

**// return view('sample',['user'=>'Peter']);**

**// # key will use as php variable in view file**

**// });**

**#another way of passing data**

**// Route::get('/sample', function(){**

**// return view('sample')->with('user','Rose');**

**// });**

**# Pass Route Parameter to View**

**// Route::get('/login/{user}/{pass}',function($user,$pass){**

**// return view('sample',['username'=>$user,'pass'=>$pass]);**

**// });**

**// -------------redirect()--------------------------**

**// Route::get('/login/{username}/{password}', function($username,$password){**

**// if($username === 'admin' && $password === 'admin'){**

**// return redirect('/success');**

**// }**

**// else{**

**// return redirect('/failure');**

**// }**

**// });**

**// Route::get('/success', function(){**

**// return 'Login Successfully';**

**// });**

**// Route::get('/failure',function(){**

**// return 'Login Failed...';**

**// });**

**#------------------Passing data with redirect()---------------------**

**Route::get('/login/{username}/{password}', function($username,$password){**

**if($username === 'admin' && $password === 'admin'){**

**return redirect('/dashboard')->with('status','Login Successfully..');**

**}**

**else{**

**return redirect('/dashboard')->with('status','Login Failed..');**

**}**

**});**

**Route::get('/dashboard', function(){**

**return view('sample');**

**});**

**view/Sample.php**

**<h1>Welcome TO Sample page</h1>**

**<h2>This is Day 1</h2>**

**<?php if(isset($user)):?>**

**<h4>**

**Username = <?php echo $user;?>**

**</h4>**

**<?php endif;?>**

**<?php if(isset($username)):?>**

**<h3>**

**User = <?php echo $username;?>**

**</h3>**

**<h3>**

**Password = <?php echo $pass;?>**

**</h3>**

**<?php endif;?>**

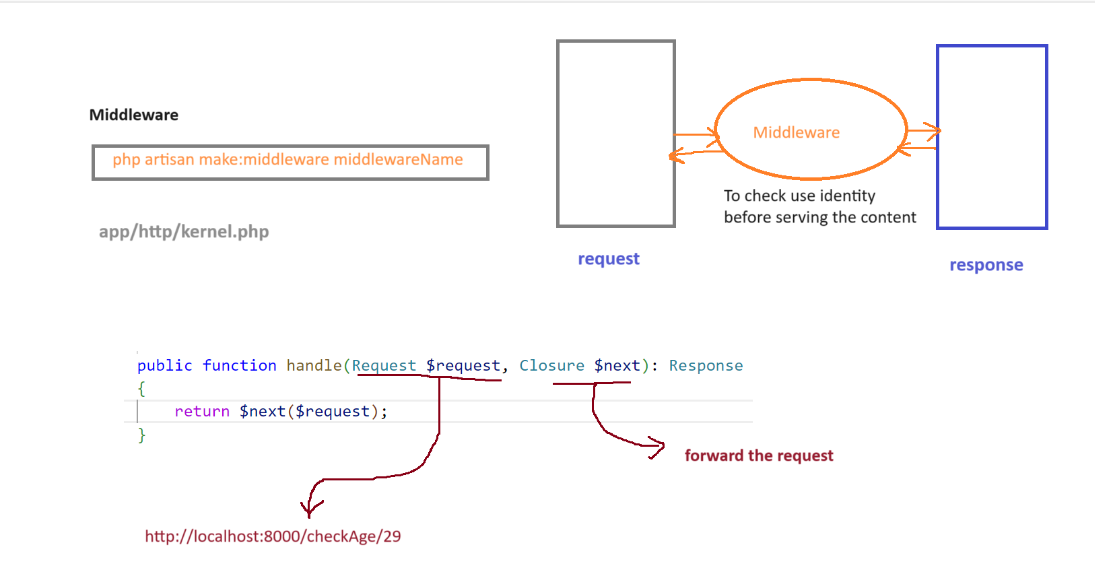
**<!-- display data coming through the redirect() -->**

**<?php if(session('status')):?>**

**<h3><?php echo session('status');?></h3>**

**<?php endif;?>**

**Middleware**

****

**Create middleware using below cmd**

php artisan make:middleware AgeMiddleware

**AgeMiddleware.php**

**public function handle(Request $request, Closure $next): Response**

**{**

**if($request->age < 18){**

**return redirect('/invalidAge');**

**}**

**return $next($request);**

**}**

**Web.php**

**#Middleware**

**#localhost:8000/checkAge/23**

**#localhost:8000/checkAge/11**

**// Route::get('/checkAge/{age}', function($age){**

**// return "Given Age is $age and can Vote";**

**// })->middleware(AgeMiddleware::class);**

**Route::get('/checkAge/{age}', function($age){**

**return "Given Age is $age and can Vote";**

**})->middleware('age');**

**Route::get('/invalidAge',function(){**

**return "can not Vote";**

**});**

**app/http/kernal.php**

**protected $middlewareAliases = [**

**'age'=> \App\Http\Middleware\AgeMiddleware::class,**

**];**

**Task**

After failure from middleware , display below output

Ex: - localhost:8000/checkAge/11

Output : **11 = can not vote**

**Solution:**

**public function handle(Request $request, Closure $next): Response**

**{**

**if($request->age < 18){**

**return redirect('/invalidAge')->with('age',$request->age);**

**}**

**return $next($request);**

**}**

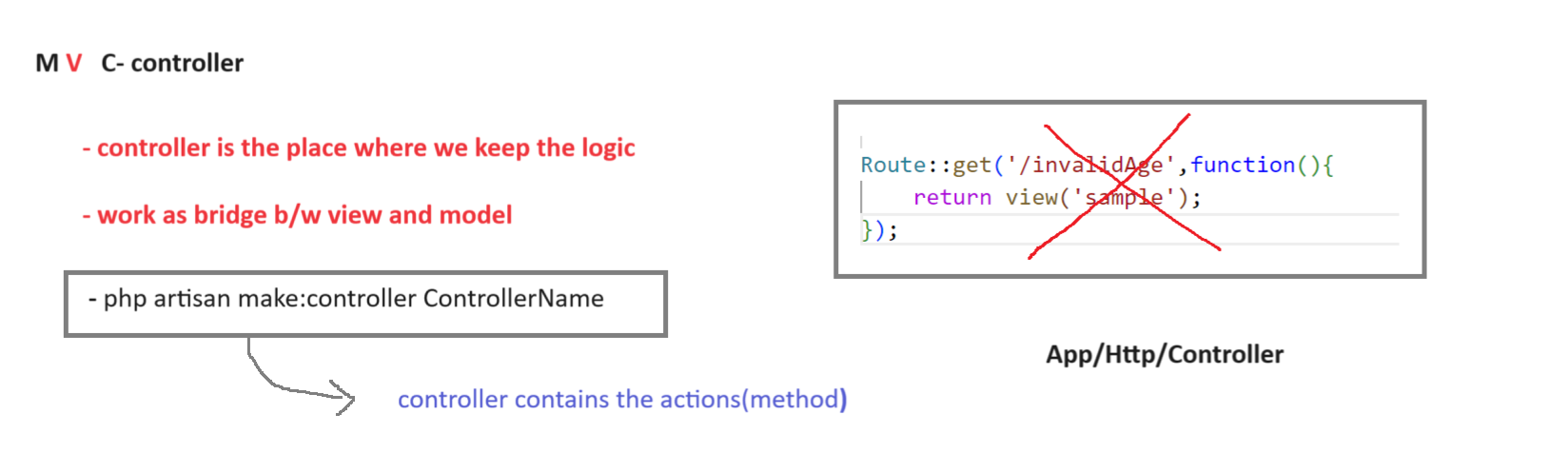
**Sample.php**

**<?php if(session('age')):?>**

**<h3><?php echo "Given Age is". session('age') ."and can not Vote";?></h3>**

**<?php endif;?>**

**Controller**

****

**php artisan make:controller TestController**

**TestController.php**

**class TestController extends Controller**

**{**

**//**

**public function showTestInfo(){**

**return view('sample');**

**}**

**public function PrintTest($testName){**

**return view('sample')->with('testName',$testName);**

**}**

**}**

**Web.php**

**Route::get('/test',[TestController::class,'showTestInfo']);**

**Route::get('/test/{testName}',[TestController::class,'PrintTest']);**

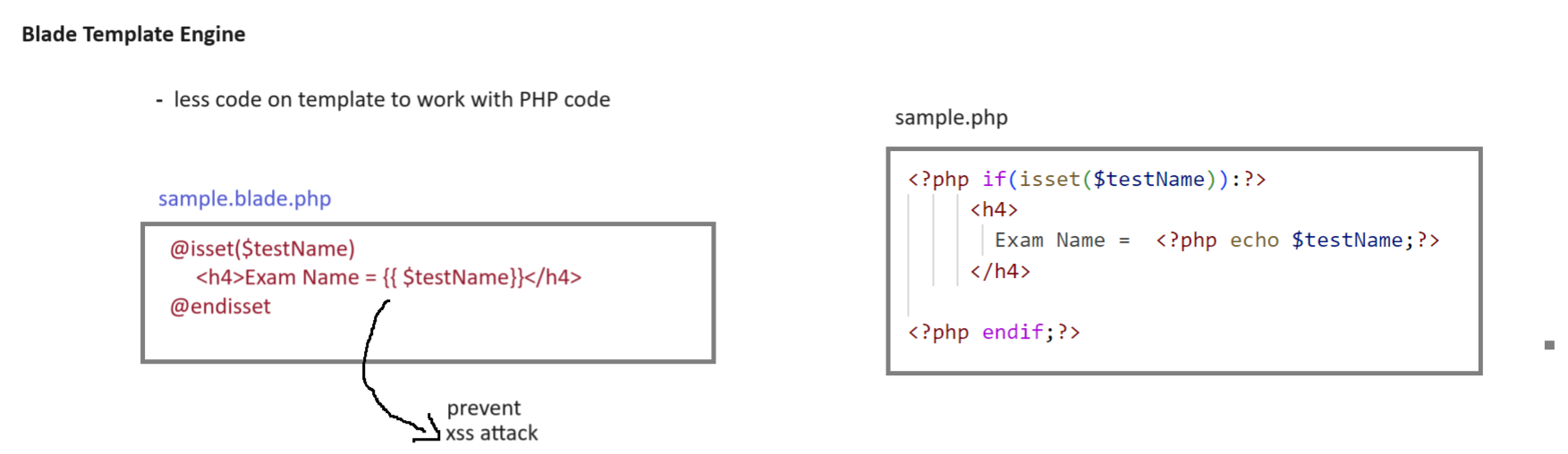
**Resource Controller**

php artisan make:controller PhotoController –resource

**Web.php**

Route::resource('/photo',PhotoController::class);

**Blade Template Engine**

****

php artisan make:controller BladeController

**Web.php**

**# Blade Controller**

**Route::get('/login/{username}/{password}',[BladeController::class,'showLoginIngo']);**

**Route::get('/user/{role}',[BladeController::class,'showUserRole']);**

**Route::get('/user/score/{score}',[BladeController::class, 'showGrade']);**

**Route::get('/employee', [BladeController::class, 'showEmployeeList']);**

**BladeController.php**

**class BladeController extends Controller**

**{**

**//**

**public function showLoginIngo($username,$password){**

**return view('bladeDemo')**

**->with(['user'=>$username,'pass'=>$password]);**

**}**

**public function showUserRole($role){**

**return view('bladeDemo')->with('role',$role);**

**}**

**public function showGrade($score){**

**return view('bladeDemo')->with('score',$score);**

**}**

**public function showEmployeeList(){**

**$users = ['Mike','Peter','Rose','Kerry'];**

**return view('bladeDemo')->with('empRecords',$users);**

**}**

**}**

**bladeDemo.blade.php**

**@isset($user)**

**<h3>Username = {{$user}}</h3>**

**<h4>Password = {{$pass}}</h4>**

**@endisset**

**@isset($role)**

**@if($role==='admin')**

**<h2>Admin Dashboard</h2>**

**@elseif($role==='guest')**

**<h2>Guest Dashboard</h2>**

**@endif**

**@endisset**

**@isset($score)**

**@switch($score)**

**@case($score >=90)**

**<strong>Grade A</strong>**

**@break**

**@case($score>=80)**

**<strong>Grade B</strong>**

**@break**

**@default**

**<strong>Fail</strong>**

**@endswitch**

**@endisset**

**@isset($empRecords)**

**<ul>**

**@foreach($empRecords as $r)**

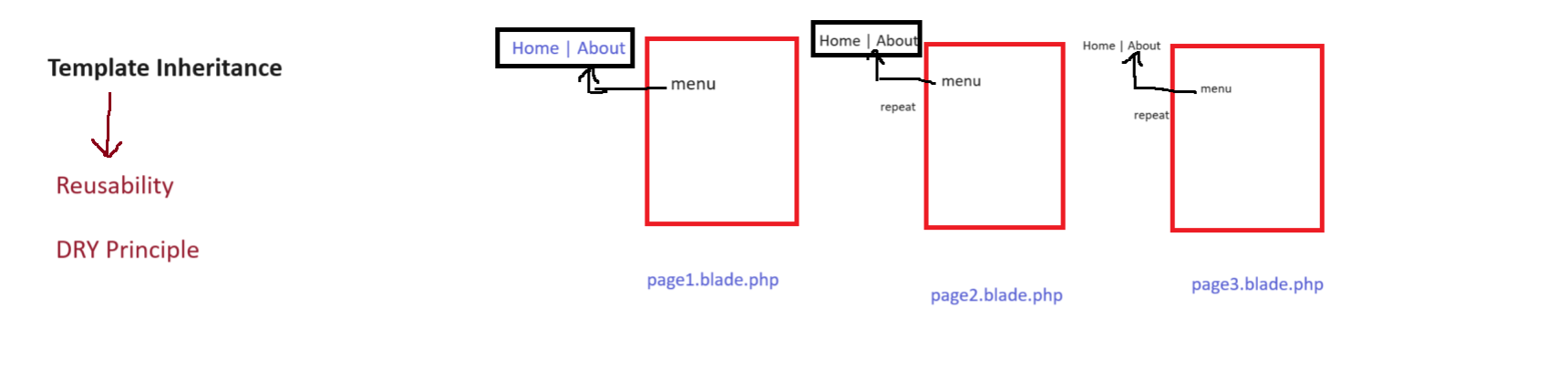
**<li>{{$r}}</li>**

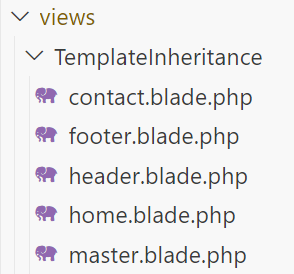
**@endforeach**

**</ul>**

**@endisset**

**Template Inheritance**

****

****

**# Template Inheritance**

**web.php**

**Route::view('/dashboard','TemplateInheritance.master');**

**// Route::get('/home',[BladeController::class,'showHomePage']);**

**// Route::get('/contact',[BladeController::class,'showContactPage']);**

**Master.blade.php**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>@yield('title')</title>**

**</head>**

**<body>**

**@include('TemplateInheritance.header')**

**@yield('content')**

**@include('TemplateInheritance.footer')**

**</body>**

**</html>**

**Header.blade.php**

**<a href="{{route('home')}}">Home</a>**

**<!-- {{route('home')}} == http://localhost:8000//home/myhome/firstpage/application/dashboard' -->**

**<a href="http://localhost:8000/contact">Contact</a>**

**<a href="{{url('login/user')}}">Login</a>**

**Footer.blade.php**

**<p>copyright@2024</p>**

**Home.blade.php**

**@extends('TemplateInheritance.master')**

**@section('title','Home Page')**

**@section('content')**

**<p>You are at <strong>HomePage</strong></p>**

**<img width="100" height="100" src="{{asset('images/camera.jpg')}}" alt="">**

**@endsection**

**Contact.blade.php**

**@extends('TemplateInheritance.master')**

**@section('title','Contact Page')**

**@section('content')**

**<p>You are at <strong>ContactPage</strong></p>**

**@endsection**

**Advance Routing**

**# Advance Routing**

**# Named Route**

**Route::get('/home/myhome/firstpage/application/dashboard/w/e/r/t/t/w/r',[BladeController::class,'showHomePage'])->name('home');**

**Route::get('/contact',[BladeController::class,'showContactPage']);**

**# Route Group with Middleware**

**Route::middleware(AgeMiddleware::class)->group(function(){**

**Route::get('/api/route1',[BladeController::class, 'showEmployeeList']);**

**Route::get('/api/route2',[BladeController::class, 'showEmployeeList']);**

**Route::get('/api/route3',[BladeController::class, 'showEmployeeList']);**

**Route::get('/api/route4',[BladeController::class, 'showEmployeeList']);**

**});**

**# Route Group with Controller**

**Route::controller(BladeController::class)->middleware(AgeMiddleware::class)->group(function(){**

**Route::get('/api/route1','showEmployeeList');**

**Route::get('/api/route2','showEmployeeList');**

**Route::get('/api/route3','showEmployeeList');**

**Route::get('/api/route4','showEmployeeList');**

**});**

**# Route Group with prefix**

**Route::prefix('/api')->controller(BladeController::class)->middleware(AgeMiddleware::class)->group(function(){**

**Route::get('/route1','showEmployeeList');**

**Route::get('/route2','showEmployeeList');**

**Route::get('/route3','showEmployeeList');**

**Route::get('/route4','showEmployeeList');**

**});**

**URL GENERATION**

**# URL GENERATION**

**Route::view('/login/user', 'TemplateInheritance.login');**

**<a href="{{route('home')}}">Home</a>**

**<!-- {{route('home')}} == http://localhost:8000//home/myhome/firstpage/application/dashboard' -->**

**<a href="http://localhost:8000/contact">Contact</a>**

**<a href="{{url('login/user')}}">Login</a>**

**Create a folder called images inside public folder**

**Put an image inside images folder**

**Home.blade.php**

**@extends('TemplateInheritance.master')**

**@section('title','Home Page')**

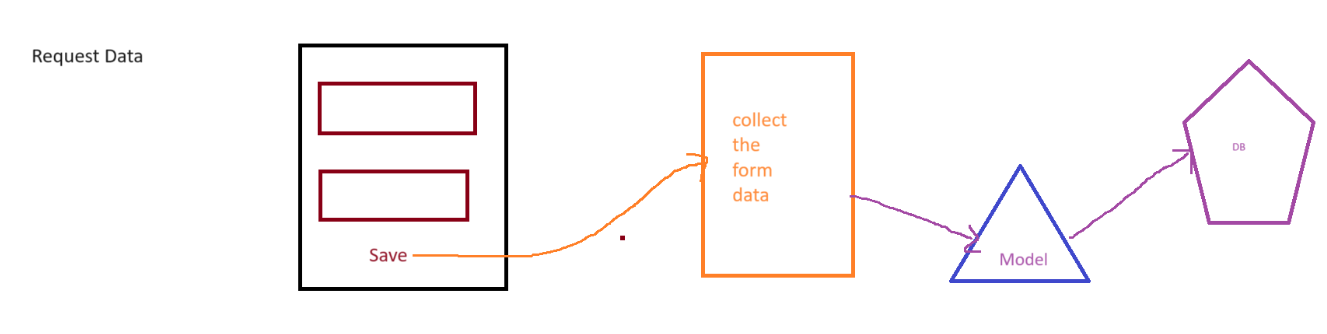
**@section('content')**

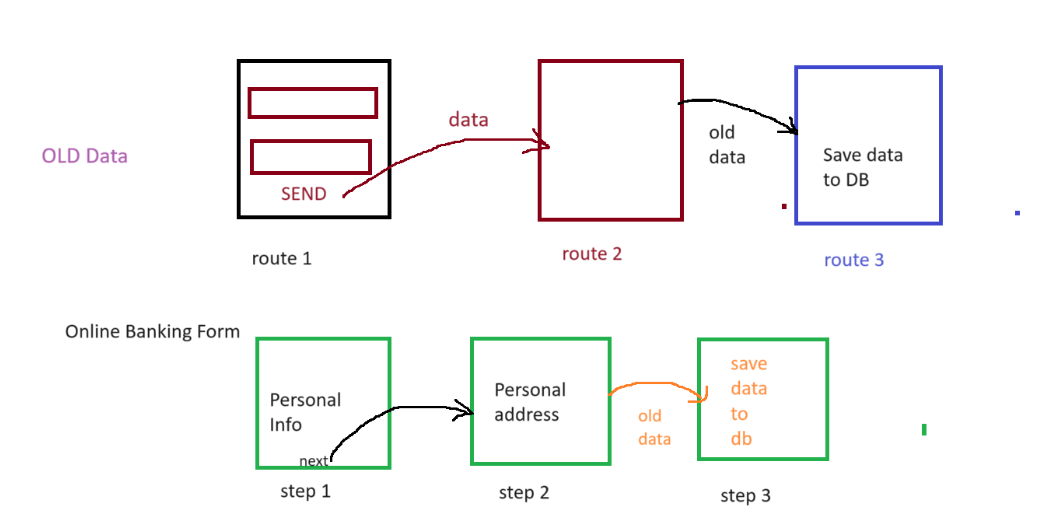
**<p>You are at <strong>HomePage</strong></p>**

**<img width="100" height="100" src="{{asset('images/camera.jpg')}}" alt="">**

**@endsection**

**REQUEST DATA**

****

****

**Web.php**

**Route::view('/form', 'Form.form');**

**Route::post('/handleSubmit',[FormController::class,'processFormData']);**

**Route::get('/oldData',[FormController::class,'workWithOldData']);**

**Default.blade.php**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>Document</title>**

**</head>**

**<body>**

**@yield('content')**

**</body>**

**</html>**

**form.blade.php**

**@extends('Form.default')**

**@section('content')**

**<form**

**action="{{url('handleSubmit')}}"**

**method="POST"**

**enctype="multipart/form-data"**

**>**

**@csrf**

**<p>**

**<label for="user">Username</label>**

**<input type="text" name="username" id="user">**

**</p>**

**<p>**

**<label for="pass">Password</label>**

**<input type="password" name="password" id="pass">**

**</p>**

**<p>**

**<input type="file" name="book">**

**</p>**

**<p>**

**<input type="submit" value="Login">**

**</p>**

**</form>**

**@endsection**

**FormController.php**

**class FormController extends Controller**

**{**

**//**

**public function processFormData(Request $request){**

**# Collect all Input Data**

**// $userInput = $request->all();**

**// echo '<pre>';**

**// var\_dump($userInput);**

**# Collect Individual Form Field**

**// $userInput = $request->input('username');**

**// echo '<pre>';**

**// var\_dump($userInput);**

**# Collect only few fields**

**// $userInput = $request->only(['username','password']);**

**// echo '<pre>';**

**// var\_dump($userInput);**

**// $userInput = $request->except(['\_token']);**

**// echo '<pre>';**

**// var\_dump($userInput);**

**# Check the existence of Field**

**// if($request->has('username')){**

**// $userInput = $request->all();**

**// echo '<pre>';**

**// var\_dump($userInput);**

**// }**

**# Working with Old Data**

**// $request->flash();**

**// // $request->flashExcept('\_token');**

**// // $request->flashOnly('username','password');**

**// return redirect('/oldData');**

**#Working with File Upload**

**$userInput = $request->file('book');**

**echo '<pre>';**

**var\_dump($userInput->path());**

**var\_dump($userInput->extension());**

**}**

**public function workWithOldData(Request $request){**

**# Collect old Data**

**// $userInput = $request->old();**

**// echo '<pre>';**

**// var\_dump($userInput);**

**}**

**}**

**Form Builder**

**Install below package**

**Composer require laravelcollective/html**

**form.blade.php**

**@extends('Form.default')**

**@section('content')**

**{{Form::open(['url'=>'handleSubmit','files'=>true])}}**

**{{Form::label('user','Username')}}**

**{{Form::text('username',null,['placeholder'=>'Enter Username'])}}**

**<br><br>**

**{{Form::label('password','Password')}}**

**{{Form::password('password',null,['placeholder'=>'Enter Password'])}}**

**<br><br>**

**<!-- Radio Button -->**

**{{Form::label('gender','Gender')}}**

**{{Form::radio('gender','male')}} Male**

**{{Form::radio('gender','female')}} Female**

**<br><br>**

**<!-- Checkbox -->**

**{{Form::label('lang','Language')}}**

**{{Form::checkbox('lang[]','English')}} English**

**{{Form::checkbox('lang[]','Hindi')}} Hindi**

**{{Form::checkbox('lang[]','Spanish')}} Spanish**

**<br><br>**

**<!-- TextArea -->**

**{{Form::label('Comment','Comment')}}**

**{{Form::textarea('comment','Please share your experience')}}**

**<br><br>**

**<!-- Dropdown list -->**

**{{Form::label('Location','Location')}}**

**{{Form::select('location',[**

**'USA'=>'USA',**

**'INDIA'=>'INDIA',**

**'UK' => 'UK'**

**])}}**

**<br><br>**

**<!-- Dropdown list -->**

**{{Form::label('Document','Upload Docs')}}**

**{{Form::file('document')}}**

**<br><br>**

**{{Form::submit('Send')}}**

**{{Form::close()}}**

**@endsection**

**Validation Message**

**Web.php**

**Route::post('/handleFormBuilderSubmit',[FormController::class,'workWithFormBuilder']);**

**FormController.php**

**public function workWithFormBuilder(Request $request){**

**#----Add Validation-----------**

**$request->validate([**

**'username'=>'required | min:4',**

**'password'=>'required | min:6'**

**]);**

**$userInput = $request->all();**

**echo '<pre>';**

**var\_dump($userInput);**

**}**

**form.blade.php**

**@extends('Form.default')**

**@section('content')**

**{{Form::open(['url'=>'handleFormBuilderSubmit','files'=>true])}}**

**{{Form::label('user','Username')}}**

**{{Form::text('username',null,['placeholder'=>'Enter Username'])}}**

**<!-- Another way of printing validation message -->**

**@error('username')**

**<span style="color:red">{{$message}}</span>**

**@enderror**

**<br><br>**

**{{Form::label('password','Password')}}**

**{{Form::password('password',['placeholder'=>'\*\*\*\*\*\*\*\*\*'])}}**

**<!-- Another way of printing validation message -->**

**@error('password')**

**<span style="color:red">{{$message}}</span>**

**@enderror**

**<br><br>**

**{{Form::submit('Send')}}**

**{{Form::close()}}**

**@endsection**

**<!-- Display Validation Error Message -->**

**<!-- @if($errors->any())**

**<ul>**

**@foreach($errors->all() as $err)**

**<li style="color:red">{{$err}}</li>**

**@endforeach**

**</ul>**

**@endif -->**

**Day 2**

Attendance Url

[**https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732319**](https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732319)

**Custom Validation Message/ Rule**

php artisan make:request CustomFormRequest

**CustomFormRequest.php**

**class CustomFormRequest extends FormRequest**

**{**

**/\*\***

**\* Determine if the user is authorized to make this request.**

**\*/**

**public function authorize(): bool**

**{**

**return true;**

**}**

**/\*\***

**\* Get the validation rules that apply to the request.**

**\***

**\* @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>**

**\*/**

**public function rules(): array**

**{**

**return [**

**//**

**// 'username'=>'required | min:4 | Uppercase',**

**'username'=>['required','min:4', new Uppercase],**

**'password'=>'required | min:6'**

**];**

**}**

**public function messages()**

**{**

**// return[**

**// 'username.required'=>'username can not be left blank',**

**// 'password.required'=>'password can not be left blank',**

**// 'username.min'=>'minimum 4 characters',**

**// 'password.min'=>'minimum 6 characters'**

**// ];**

**return[**

**'required'=>':attribute can not be left blank',**

**'username.min'=>'minimum 4 characters',**

**'password.min'=>'minimum 6 characters'**

**];**

**}**

**}**

php artisan make:rule Uppercase

**Uppercase.php**

**public function validate(string $attribute, mixed $value, Closure $fail): void**

**{**

**// user is given : JOHN**

**if(strtoupper($value) !== $value){**

**$fail('The :attribute must be uppercase');**

**}**

**}**

**FormController.php**

**public function workWithFormBuilder(CustomFormRequest $request){**

**$userInput = $request->all();**

**echo '<pre>';**

**var\_dump($userInput);**

**}**

**Practice: Create a custom validation Rule**

**Scenario:**

I want to validate the inputted birth year and only accept from 1990 to the current year.

If the user provides less than 1990 or more than the current year it will throw a validation error.

**DATABASE**

**Use Mysql Service from below cloud**

[**https://console.clever-cloud.com/**](https://console.clever-cloud.com/)

**.env**

**DB\_CONNECTION=mysql**

**DB\_HOST=bgl7re7uxl5b2366ybpk-mysql.services.clever-cloud.com**

**DB\_PORT=3306**

**DB\_DATABASE=bgl7re7uxl5b2366ybpk**

**DB\_USERNAME=unrggbhay4hn6ozv**

**DB\_PASSWORD=kz5McpZG3Zu5lu7PKW02**

**Multiple connection**

**# open database.php and create two different database**

**'database1' => [**

**'driver' => 'mysql',**

**'host' => env('DB\_HOST', '127.0.0.1'),**

**'port' => env('DB\_PORT', '3306'),**

**'database' => env('DB\_DATABASE', 'forge'),**

**'username' => env('DB\_USERNAME', 'forge'),**

**'password' => env('DB\_PASSWORD', ''),**

**'unix\_socket' => env('DB\_SOCKET', ''),**

**'charset' => 'utf8mb4',**

**'collation' => 'utf8mb4\_unicode\_ci',**

**'prefix' => '',**

**'strict' => true,**

**'engine' => null,**

**],**

**'database2' => [**

**'driver' => 'mysql',**

**'host' => env('DB\_HOST\_SECOND', '127.0.0.1'),**

**'port' => env('DB\_PORT\_SECOND', '3306'),**

**'database' => env('DB\_DATABASE\_SECOND', 'forge'),**

**'username' => env('DB\_USERNAME\_SECOND', 'forge'),**

**'password' => env('DB\_PASSWORD\_SECOND', ''),**

**'unix\_socket' => env('DB\_SOCKET\_SECOND', ''),**

**'charset' => 'utf8mb4',**

**'collation' => 'utf8mb4\_unicode\_ci',**

**'prefix' => '',**

**'strict' => true,**

**'engine' => null,**

**],**

**# Open .env file and do the setting**

**DB\_CONNECTION=mysql**

**DB\_HOST=localhost**

**DB\_PORT=3306**

**DB\_DATABASE=db1**

**DB\_USERNAME=root**

**DB\_PASSWORD=**

**DB\_CONNECTION\_SECOND=mysql**

**DB\_HOST\_SECOND=127.0.0.1**

**DB\_PORT\_SECOND=3306**

**DB\_DATABASE\_SECOND=db2**

**DB\_USERNAME\_SECOND=root**

**DB\_PASSWORD\_SECOND=**

**# use** php artisan migrate **to create base table and session table**

**# web.php**

**Route::get('/', function () {**

**// $productData = DB::select('SELECT \* FROM countries');**

**// dd($productData);**

**$result1= DB::connection('database1')->select('SELECT \* FROM countries');**

**$result = DB::connection('database2')->select('SELECT \* FROM products');**

**echo '<pre>';**

**print\_r($result1);**

**print\_r($result);**

**});**

**#Done Multiple Connection**

**config/database.php , just add below code**

**'mysql2' => [**

**'driver' => 'mysql',**

**'url' => env('DATABASE\_URL'),**

**'host' => env('DB\_HOST\_SECOND', '127.0.0.1'),**

**'port' => env('DB\_PORT\_SECOND', '3306'),**

**'database' => env('DB\_DATABASE\_SECOND', 'forge'),**

**'username' => env('DB\_USERNAME\_SECOND', 'forge'),**

**'password' => env('DB\_PASSWORD\_SECOND', ''),**

**'unix\_socket' => env('DB\_SOCKET', ''),**

**'charset' => 'utf8mb4',**

**'collation' => 'utf8mb4\_unicode\_ci',**

**'prefix' => '',**

**'prefix\_indexes' => true,**

**'strict' => true,**

**'engine' => null,**

**'options' => extension\_loaded('pdo\_mysql') ? array\_filter([**

**PDO::MYSQL\_ATTR\_SSL\_CA => env('MYSQL\_ATTR\_SSL\_CA'),**

**]) : [],**

**],**

php artisan make:controller DBController

**DBController.php**

**class DBController extends Controller**

**{**

**//**

**public function insert(){**

**$result = DB::insert('INSERT INTO products(name,image,price) VALUES(:name,:image,:price)',[**

**'image'=>'https://via.placeholder.com/160',**

**'name'=>'Product 2',**

**'price'=>94**

**]);**

**var\_dump($result);**

**}**

**public function read(){**

**// $productData = DB::select('SELECT \* FROM products');**

**// return view('database.read',['products'=>$productData]);**

**$productData = DB::select('SELECT \* FROM products WHERE id=:id',['id'=>2]);**

**echo '<pre>';**

**var\_dump($productData);**

**}**

**public function update(){**

**$result = DB::update('UPDATE products SET price=100 WHERE id = :id',['id'=>2]);**

**echo '<pre>';**

**print\_r($result);**

**}**

**public function delete(){**

**$result = DB::delete('DELETE FROM products WHERE id=:id',['id'=>2]);**

**echo '<pre>';**

**print\_r($result);**

**}**

**public function execute\_multiple\_query(){**

**DB::transaction(function(){**

**DB::update('UPDATE products SET price =33 WHERE id=:id',['id'=>1]);**

**DB::insert('INSERT INTO products(name,image,price) VALUES(:name,:image,:price)',[**

**'image'=>'https://via.placeholder.com/160',**

**'name'=>'Product 3',**

**'price'=>44**

**]);**

**});**

**}**

**public function muliple\_connection\_database(){**

**$result1= DB::connection('mysql')->select('SELECT \* FROM products');**

**$result = DB::connection('mysql2')->select('SELECT \* FROM courses');**

**echo '<pre>';**

**print\_r($result);**

**print\_r($result1);**

**}**

**}**

**Web.php**

**Route::get('/insert',[DBController::class,'insert']);**

**Route::get('/read',[DBController::class,'read']);**

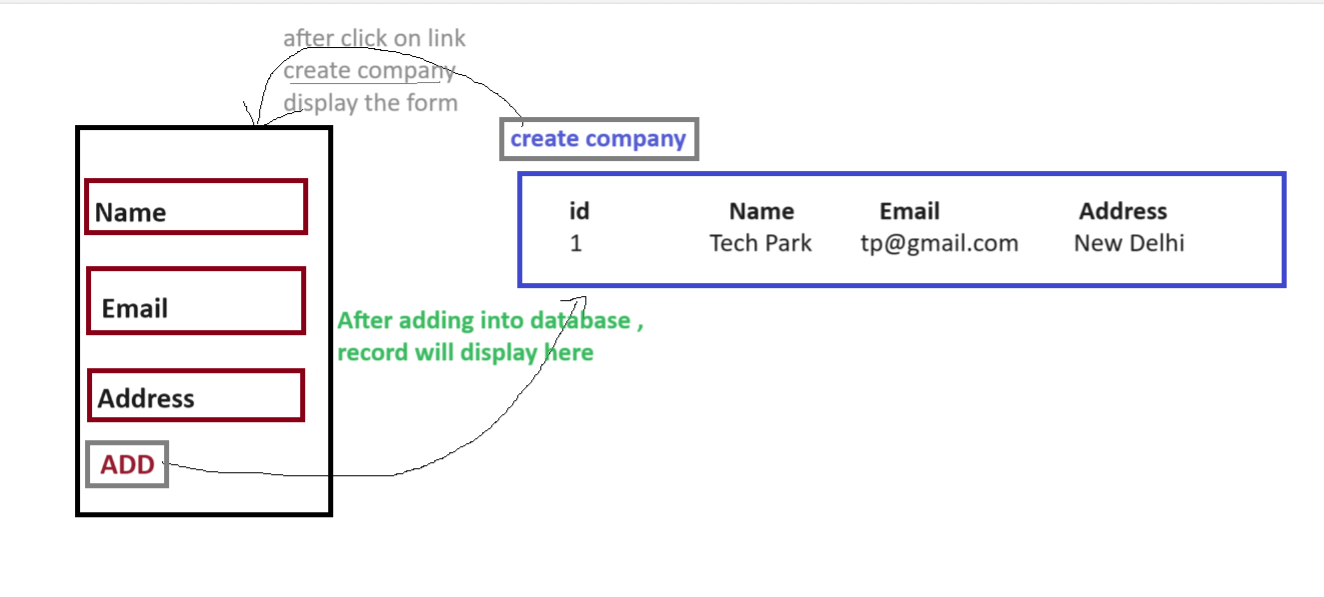
**Route::get('/update',[DBController::class,'update']);**

**Route::get('/delete',[DBController::class,'delete']);**

**Route::get('/execute\_multiple\_query',[DBController::class,'execute\_multiple\_query']);**

**Route::get('/muliple\_connection',[DBController::class,'muliple\_connection\_database']);**

**Practice**

****

**Solution:**

**Web.php**

**Route::get('/dashboard', [DBController::class,'showDashboard']);**

**Route::view('/create','Form.addNew');**

**Route::post('/handleSubmit',[DBController::class,'saveRecord']);**

**DBController.php**

**public function showDashboard(){**

**$productData = DB::select('SELECT \* FROM products');**

**return view('database.read',['products'=>$productData]);**

**}**

**public function saveRecord(Request $request){**

**DB::insert('INSERT INTO products(name,image,price) VALUES(:name,:image,:price)',[**

**'image'=>$request->image,**

**'name'=>$request->name,**

**'price'=>$request->price**

**]);**

**return redirect('/dashboard')->with('status','data added successfully');**

**}**

**addNew.blade.php**

**{{Form::open(['url'=>'handleSubmit'])}}**

**<br><br>**

**{{Form::label('name','Name')}}**

**{{Form::text('name')}}**

**<br><br>**

**{{Form::label('image','Image')}}**

**{{Form::text('image')}}**

**<br><br>**

**{{Form::label('price','Price')}}**

**{{Form::text('price')}}**

**<br><br>**

**{{Form::submit('Save')}}**

**{{Form::close()}}**

**read.blade.php**

**<a href="{{url('/create')}}">Create New Record</a>**

**<br><br>**

**@if(session('status'))**

**<span style="color:green"><strong>{{session('status')}}</strong></span>**

**@endif**

**<br>**

**<table border="1">**

**<thead>**

**<tr>**

**<th>ID</th>**

**<th>NAME</th>**

**<th>IMAGE</th>**

**<th>PRICE</th>**

**</tr>**

**</thead>**

**<tbody>**

**@foreach($products as $p)**

**<tr>**

**<td>{{$p->id}}</td>**

**<td>{{$p->name}}</td>**

**<td>{{$p->image}}</td>**

**<td>{{$p->price}}</td>**

**</tr>**

**@endforeach**

**</tbody>**

**</table>**

**Query Builder**

**web.php**

**// ----Query Builder---------------**

**// Route::get('/queryBuilder', [DBController::class, 'workWithQueryBuilder']);**

**// Route::get('/queryBuilder', [DBController::class, 'workWithQueryBuilderReading']);**

**// Route::get('/queryBuilder', [DBController::class, 'workWithQueryBuilderUpdate']);**

**Route::get('/queryBuilder', [DBController::class, 'workWithQueryBuilderDelete']);**

**DBController.php**

**public function workWithQueryBuilder(){**

**# Insertion**

**DB::table('products')->insert(**

**[**

**[**

**'image'=>'https://via.placeholder.com/160',**

**'name'=>'Product 11',**

**'price'=>98**

**],**

**[**

**'image'=>'https://via.placeholder.com/160',**

**'name'=>'Product 12',**

**'price'=>23**

**]**

**]**

**);**

**echo '<h4>Record added successfully';**

**}**

**public function workWithQueryBuilderReading(){**

**# Retrieve all Records**

**// $products = DB::table('products')->get();**

**// echo '<pre>';**

**// print\_r($products);**

**# Retrieve the first record**

**// $products = DB::table('products')->first();**

**// echo '<pre>';**

**// print\_r($products);**

**# Retrieve the 3rd record**

**// $products = DB::table('products')->find(3);**

**// echo '<pre>';**

**// print\_r($products);**

**# Retrieve the image of product based on the name**

**// $products = DB::table('products')->where('name','Product 5')->value('image');**

**// echo '<pre>';**

**// print\_r($products);**

**}**

**public function workWithQueryBuilderUpdate(){**

**DB::table('products')->where('id','=',1)->update(['name'=>'shoe']);**

**$products = DB::table('products')->get();**

**echo '<pre>';**

**print\_r($products);**

**}**

**public function workWithQueryBuilderDelete(){**

**DB::table('products')->where('id','=',1)->delete();**

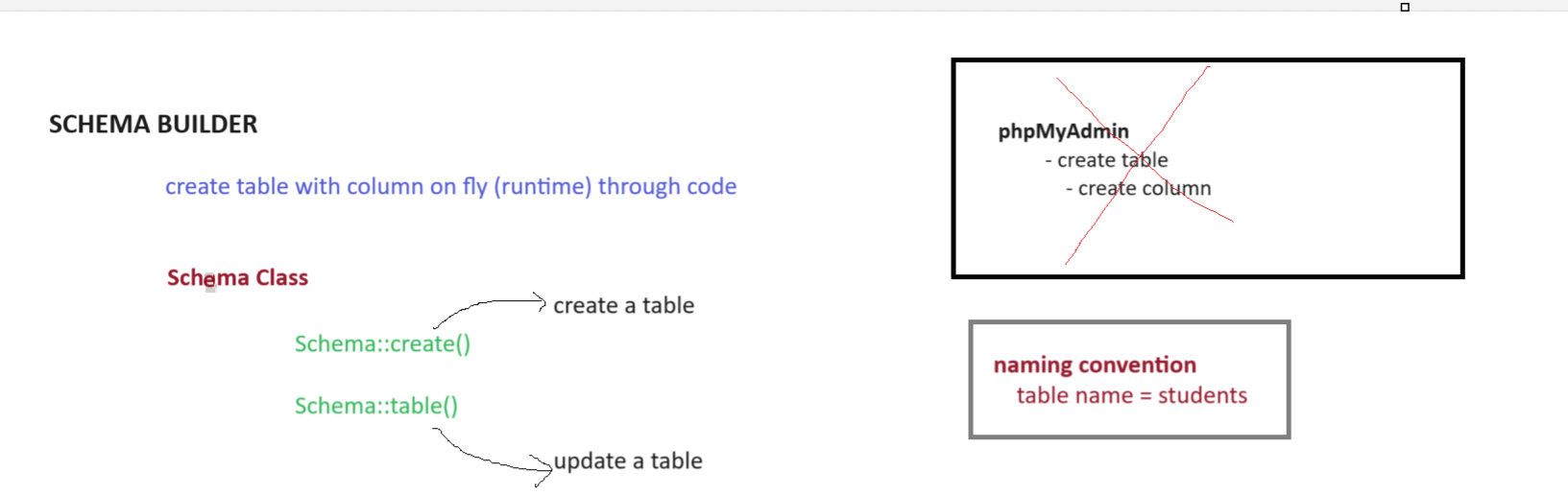
**$products = DB::table('products')->get();**

**echo '<pre>';**

**print\_r($products);**

**}**

**SCHEMA BUILDER**

****

**DBController.php**

**public function workWithSchemaBuilder(){**

**# Creating Table with Column**

**// Schema::create('students',function(Blueprint $table){**

**// $table->increments('id');**

**// $table->string('Name',60);**

**// $table->string('Email');**

**// $table->string('Password');**

**// });**

**# Updating Table**

**// Schema::table('students', function(Blueprint $table){**

**// $table->text('comment');**

**// $table->boolean('isAdmin');**

**// $table->date('dob');**

**// });**

**# Rename Table**

**// Schema::rename('students','persons');**

**# Drop Table column**

**// Schema::table('persons', function(Blueprint $table){**

**// $table->dropColumn('isAdmin');**

**// });**

**# Drop Table**

**// Schema::drop('persons');**

**# Create or Modify table with the context of Database connection**

**Schema::connection('mysql')->create('students', function(Blueprint $table){**

**});**

**Schema::connection('mysql2')->table('students', function(Blueprint $table){**

**});**

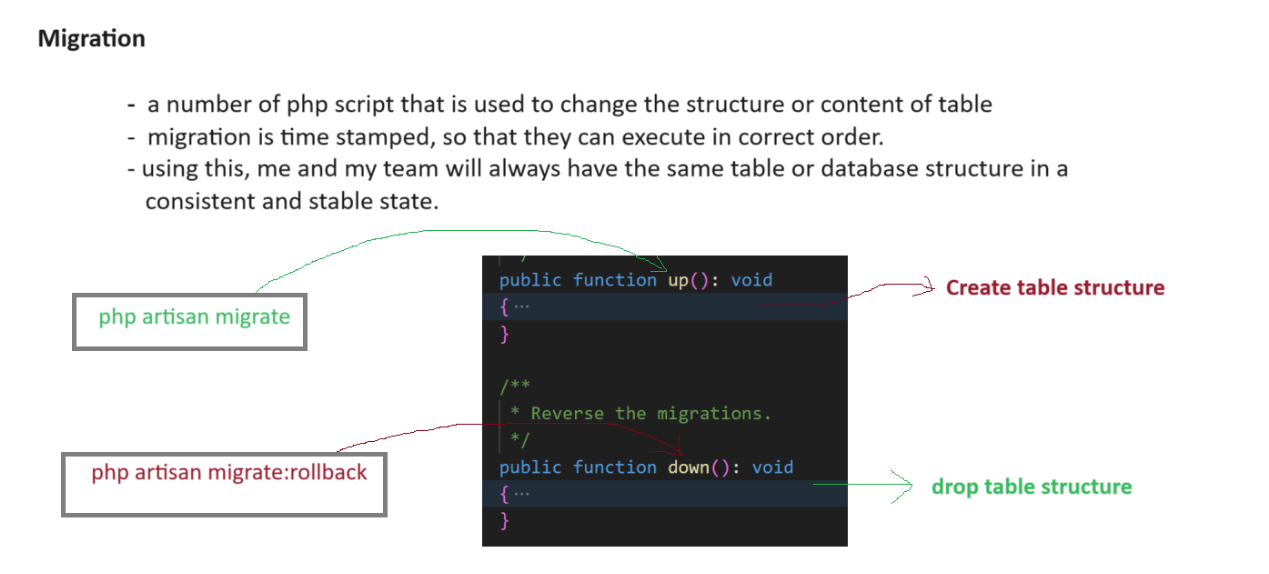
**}**

**Web.php**

**// ----------------SCHEMA BUILDER------------------------**

**Route::get('/schema', [DBController::class, 'workWithSchemaBuilder']);**

**Migration**

****

**Command**

**# Run Migration File**

php artisan migrate

**# Rollback Migration file**

Php artisan migrate:rollback

**# Rollback specific migration file**

php artisan migrate:rollback -- step=1

php artisan migrate:rollback -- path= migration file path name

**Create New Migration file**

php artisan make:migration create\_contacts\_table

public function up(): void

{

Schema::create('contacts', function (Blueprint $table) {

$table->id();

$table->string('Mobile\_Number');

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*/

public function down(): void

{

Schema::dropIfExists('contacts');

}

**Add new Column in Existing table**

php artisan make:migration add\_email\_column\_in\_contacts\_table

public function up(): void

{

Schema::table('contacts', function (Blueprint $table) {

//

$table->after('Mobile\_Number', function($table){

$table->string('email')->unique();

});

});

}

/\*\*

\* Reverse the migrations.

\*/

public function down(): void

{

Schema::table('contacts', function (Blueprint $table) {

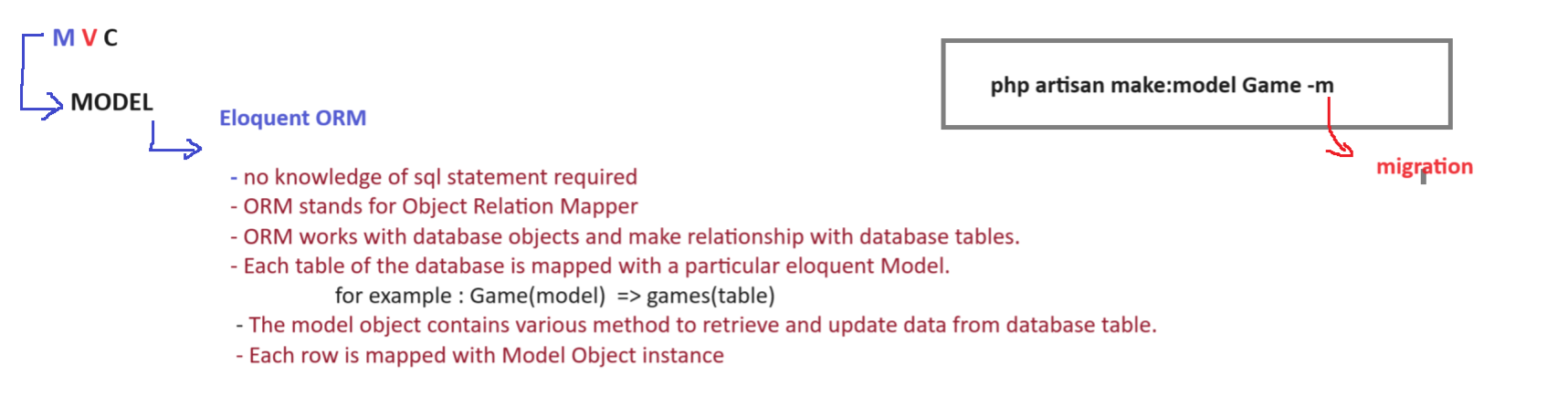
//

$table->dropColumn('email');

});

}

Eloquent ORM



MODEL

php artisan make:model Game -m

Game migration file

public function up(): void

{

Schema::create('games', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->text('Description');

$table->timestamps();

});

}

php artisan migrate

php artisan make:controller GameController

class GameController extends Controller

{

//

public function insert(){

$game = new Game();

$game->name = 'Baseball';

$game->description = 'Baseball is the best game';

$game->save();

}

public function read(){

$game = Game::all();

return view('Eloquent.display',['allGames'=>$game]);

}

public function update(){

$foundRecord = Game::find(2);

$foundRecord->name='F1 Race';

$foundRecord->Description = 'F1 Race is one of the favorite game';

$foundRecord->save();

}

public function delete(){

$foundRecord = Game::find(3);

$foundRecord->delete();

$game = Game::all();

return view('Eloquent.display',['allGames'=>$game]);

}

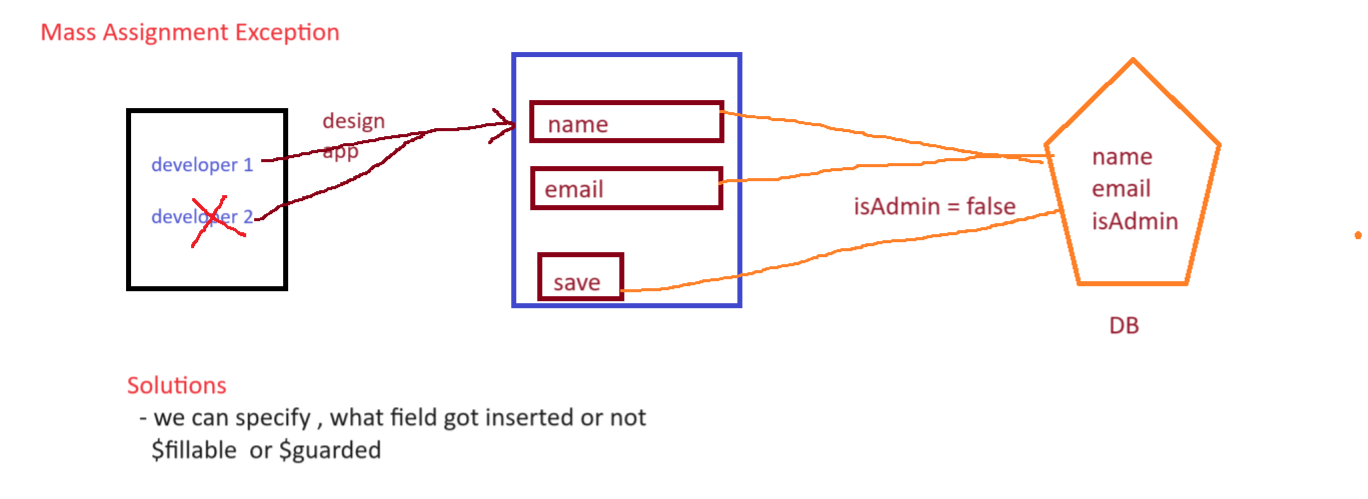
# Another way of inserting Record

public function saveGameData(Request $request){

Game::create($request->all());

}

}



Game.php

class Game extends Model

{

use HasFactory;

// protected $fillable=['name','Description'];

protected $guarded = ['\_token'];

}

Web.php

# Eloquent ORM

Route::view('/showGameForm','Eloquent.game');

// Route::get('/modelInsert', [GameController::class, 'insert']);

Route::post('/handleGameFormData', [GameController::class, 'saveGameData']);

Route::get('/modelRead', [GameController::class, 'read']);

Route::get('/modelUpdate', [GameController::class, 'update']);

Route::get('/modelDelete', [GameController::class, 'delete']);

game.blade.php

{{Form::open(['url'=>'handleGameFormData'])}}

<br><br>

{{Form::label('name','Name')}}

{{Form::text('name')}}

<br><br>

{{Form::label('description','Description')}}

{{Form::text('Description')}}

<br><br>

{{Form::submit('Save')}}

{{Form::close()}}

display.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h3>List of All Games</h3>

<table border="1">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Description</th>

</tr>

</thead>

<tbody>

@foreach($allGames as $game)

<tr>

<td>{{$game->id}}</td>

<td>{{$game->name}}</td>

<td>{{$game->Description}}</td>

</tr>

@endforeach

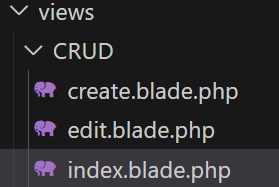
</tbody>

</table>

</body>

</html>

CRUD PROJECT with Resource Controller



php artisan make:model Company -m

Company migration file

public function up(): void

{

Schema::create('companies', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('email');

$table->string('address');

$table->timestamps();

});

}

php artisan migrate

Php artisan make:controller CompanyController –-resource

class CompanyController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*/

public function index()

{

//

$companies = Company::all();

return view('CRUD.index',['companies'=>$companies]);

}

/\*\*

\* Show the form for creating a new resource.

\*/

public function create()

{

//

return view('CRUD.create');

}

/\*\*

\* Store a newly created resource in storage.

\*/

public function store(Request $request)

{

//

$request->validate([

'name'=>'required',

'email'=>'required',

'address'=>'required',

]);

Company::create($request->all());

return redirect()->route('company.index')->with('success','company has created successfully');

}

/\*\*

\* Display the specified resource.

\*/

public function show(string $id)

{

//

}

/\*\*

\* Show the form for editing the specified resource.

\*/

public function edit(string $id)

{

//

$companyRecord = Company::find($id);

return view('CRUD.edit',['companyRecord'=>$companyRecord]);

}

/\*\*

\* Update the specified resource in storage.

\*/

public function update(Request $request, string $id)

{

//

$request->validate([

'name'=>'required',

'email'=>'required',

'address'=>'required',

]);

$companyRecord = Company::find($id);

$companyRecord->fill($request->all())->save();

return redirect()->route('company.index')->with('success','company has updated successfully');

}

/\*\*

\* Remove the specified resource from storage.

\*/

public function destroy(string $id)

{

//

$foundRecord = Company::find($id);

$foundRecord->delete();

return redirect()->route('company.index')->with('success','company has deleted successfully');

}

}

web.php

# CRUD Application------------

Route::resource('/company',CompanyController::class);

Index.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<br>

<a href="{{route('company.create')}}">Create Company</a>

<br>

@if(session('success'))

<p style="color:green">{{session('success')}}</p>

@endif

<br>

<table border="1">

<thead>

<tr>

<th>Sr.No</th>

<th>Company Name</th>

<th>Company Email</th>

<th>Company Address</th>

<th>Action(s)</th>

</tr>

</thead>

<tbody>

@foreach($companies as $c)

<tr>

<td>{{$c->id}}</td>

<td>{{$c->name}}</td>

<td>{{$c->email}}</td>

<td>{{$c->address}}</td>

<td>

{{Form::open(['route'=>['company.destroy',$c->id],'method'=>'DELETE'])}}

<a href="{{route('company.edit',$c->id)}}">Edit</a>

&nbsp;&nbsp;

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

{{Form::close()}}

</td>

</tr>

@endforeach

@if(!count($companies))

<tr>

<td colspan="4">No Records Available</td>

</tr>

@endif

</tbody>

</table>

</body>

</html>

create.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h2> Enter the Company Details below :- </h2>

{{Form::open(['route'=>'company.store'])}}

<br>

{{Form::label('name','Company Name')}}

{{Form::text('name')}}

<br><br>

{{Form::label('email','Company Email')}}

{{Form::text('email')}}

<br><br>

{{Form::label('address','Company Address')}}

{{Form::text('address')}}

<br><br>

{{Form::submit('Save')}}

{{Form::close()}}

</body>

</html>

edit.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h2> Update the Company Details below :- </h2>

{{Form::open(['route'=>['company.update',$companyRecord->id],'method'=>'PUT'])}}

<br>

{{Form::label('name','Company Name')}}

{{Form::text('name',$companyRecord->name)}}

<br><br>

{{Form::label('email','Company Email')}}

{{Form::text('email',$companyRecord->email)}}

<br><br>

{{Form::label('address','Company Address')}}

{{Form::text('address',$companyRecord->address)}}

<br><br>

{{Form::submit('Update')}}

{{Form::close()}}

</body>

</html>

**Day 3**

Attendance url

<https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732320>

**Eloquent Queries**

php artisan make:model Album -m

Album migration file

public function up(): void

{

Schema::create('albums', function (Blueprint $table) {

$table->id();

$table->string('title');

$table->string('artist');

$table->string('genre');

$table->integer('year');

$table->timestamps();

});

}

php artisan migrate

Album.php

protected $guarded = ['\_token'];

php artisan make:controller AlbumController

AlbumController.php

class AlbumController extends Controller

{

//

public function insert(){

Album::create([

'title'=>'Leaving through the window',

'artist'=>'Matt Nathanson',

'genre' => 'Piano Rock',

'year'=>2002

]);

}

public function showQueries(){

# Fetch method

// return Album::find(1);

// return Album::find([1,2]);

// return Album::all();

// return Album::first();

// Album::where('artist','=','Matt Nathanson')->update(['artist'=>'John Doe']);

// return Album::all();

// Album::where('artist','=','John Doe')->delete();

// return Album::all();

// return Album::where('title','=','North')->get();

// return Album::where('artist','=','Something corporate')->get(['id','title']);

// return Album::where('artist','=','Something corporate')->toSql();

// return Album::where('artist','=','Something corporate')->where('year','=',2001)->get();

// return Album::where('artist','=','Something corporate')->orWhere('year','=',2001)->get();

// return Album::whereRaw('artist=? AND year=?',['Something corporate',2001])->get();

// return Album::whereBetween('year',[2000,2010])->get();

// return Album::where('artist','=','Something corporate')->orderBy('year')->get();

// return Album::where('artist','=','Something corporate')->orderBy('year','desc')->get();

}

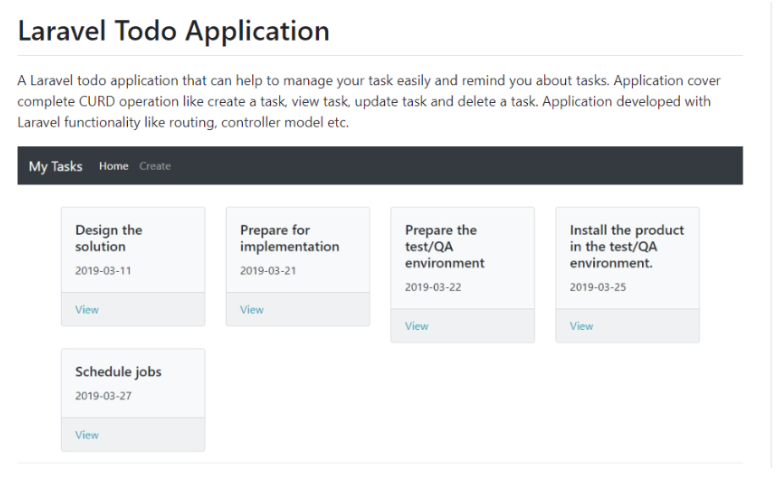
}

web.php

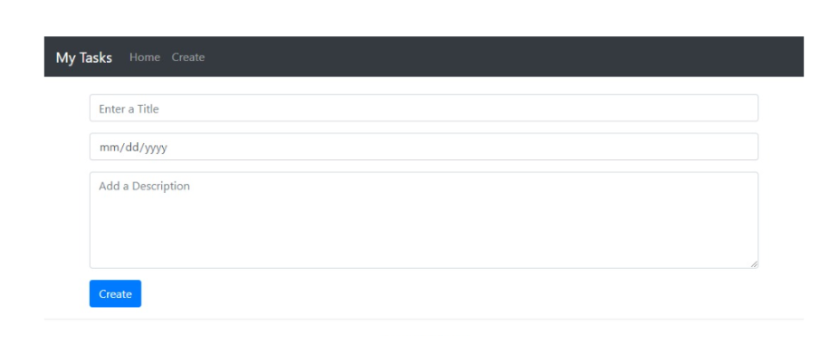
Route::get('/insert',[AlbumController::class,'insert']);

Route::get('/query',[AlbumController::class,'showQueries']);

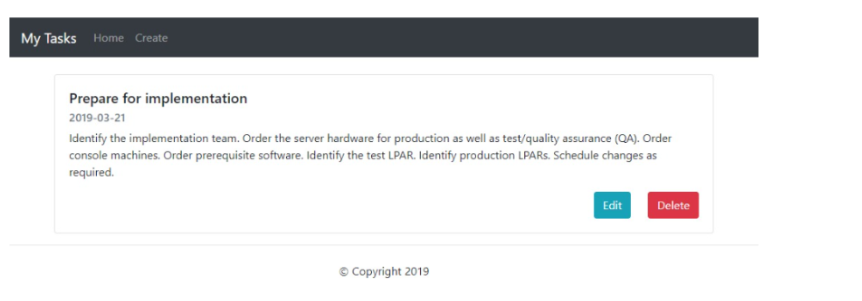
Practice on Resource Controller



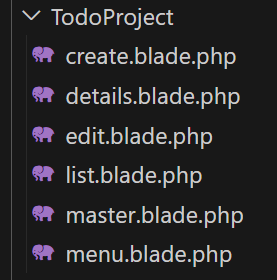
After click on Create link



After click on view link



Solution



php artisan make:controller TodoController --resource

php artisan make:model Todo -m

Todo migration file

public function up(): void

{

Schema::create('todos', function (Blueprint $table) {

$table->id();

$table->string('title');

$table->date('published');

$table->text('description');

$table->timestamps();

});

}

Todo.php

class Todo extends Model

{

use HasFactory;

protected $guarded = ['\_token'];

}

TodoController.php

class TodoController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*/

public function index()

{

//

$todos = Todo::all();

return view('TodoProject.list',['todos'=>$todos]);

}

/\*\*

\* Show the form for creating a new resource.

\*/

public function create()

{

//

return view('TodoProject.create');

}

/\*\*

\* Store a newly created resource in storage.

\*/

public function store(Request $request)

{

//

Todo::create($request->all());

return redirect()->route('todo.index')->with('status','Todo has created successfully');

}

/\*\*

\* Display the specified resource.

\*/

public function show(string $id)

{

//

$todo = Todo::find($id);

return view('TodoProject.details',['todo'=>$todo]);

}

/\*\*

\* Show the form for editing the specified resource.

\*/

public function edit(string $id)

{

//

$todo = Todo::find($id);

return view('TodoProject.edit',['todo'=>$todo]);

}

/\*\*

\* Update the specified resource in storage.

\*/

public function update(Request $request, string $id)

{

//

$todo = Todo::find($id);

$todo->fill($request->all())->save();

return redirect()->route('todo.index');

}

/\*\*

\* Remove the specified resource from storage.

\*/

public function destroy(string $id)

{

//

$todo = Todo::find($id);

$todo->delete();

return redirect()->route('todo.index');

}

}

master.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">

</head>

<body>

@include('TodoProject.menu')

@yield('content')

</body>

</html>

Menu.blade.php

<nav class="navbar navbar-expand-lg bg-info">

<div class="container-fluid">

<a class="navbar-brand" href="#">My Tasks</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarNavAltMarkup">

<div class="navbar-nav">

<a class="nav-link active" aria-current="page" href="#">Home</a>

<a class="nav-link" href="{{route('todo.create')}}">Create</a>

</div>

</div>

</div>

</nav>

List.blade.php

@extends('TodoProject.master')

@section('content')

<div class="container">

<div class="row mt-3">

@foreach($todos as $todo)

<div class="card" style="width:15rem;margin-right:8px">

<div class="card-body">

<h5>{{$todo->title}}</h5>

<p class="card-text">

{{$todo->published}}

</p>

<a href="{{route('todo.show',$todo->id)}}" class="card-link mt-4 btn btn-warning">view</a>

</div>

</div>

@endforeach

</div>

</div>

@endsection

Edit.blade.php

@extends('TodoProject.master')

@section('content')

<div class="container">

{{Form::open(['route'=>['todo.update', $todo->id],'method'=>'PUT'])}}

<div class="row mb-4 mt-4 ">

<div class="col-md-12">

{{Form::text('title',$todo->title,['placeholder'=>'Enter Title','class'=>'form-control'])}}

</div>

</div>

<div class="row mb-4">

<div class="col-md-12">

{{Form::date('published',$todo->published,['placeholder'=>'mm/dd/yyyy','class'=>'form-control'])}}

</div>

</div>

<div class="row mb-4">

<div class="col-md-12">

{{Form::textarea('description',$todo->description,['placeholder'=>'Add a Description','class'=>'form-control'])}}

</div>

</div>

<div class="row mb-4">

<div class="col-md-12">

{{Form::submit('update',['class'=>'btn btn-sm btn-success'])}}

</div>

</div>

{{Form::close()}}

</div>

@endsection

Details.blade.php

@extends('TodoProject.master')

@section('content')

<div class="container">

<div class="row mt-4 border">

<div class="col-md-12">

<h5 class="mt-2">{{$todo->title}}</h5>

<small>{{$todo->published}}</small>

<p class="mt-4">

{{$todo->description}}

</p>

{{Form::open(['route'=>['todo.destroy',$todo->id],'method'=>'DELETE'])}}

<a class="btn btn-info btn-sm" href="{{route('todo.edit',$todo->id)}}">Edit</a>

&nbsp;&nbsp;

<button type="submit" class="btn btn-danger btn-sm">Delete</button>

{{Form::close()}}

<br>

</div>

</div>

</div>

@endsection

Create.blade.php

@extends('TodoProject.master')

@section('content')

<div class="container">

{{Form::open(['route'=>'todo.store'])}}

<div class="row mb-4 mt-4 ">

<div class="col-md-12">

{{Form::text('title',null,['placeholder'=>'Enter Title','class'=>'form-control'])}}

</div>

</div>

<div class="row mb-4">

<div class="col-md-12">

{{Form::date('published',null,['placeholder'=>'mm/dd/yyyy','class'=>'form-control'])}}

</div>

</div>

<div class="row mb-4">

<div class="col-md-12">

{{Form::textarea('description',null,['placeholder'=>'Add a Description','class'=>'form-control'])}}

</div>

</div>

<div class="row mb-4">

<div class="col-md-12">

{{Form::submit('create',['class'=>'btn btn-sm btn-primary'])}}

</div>

</div>

{{Form::close()}}

</div>

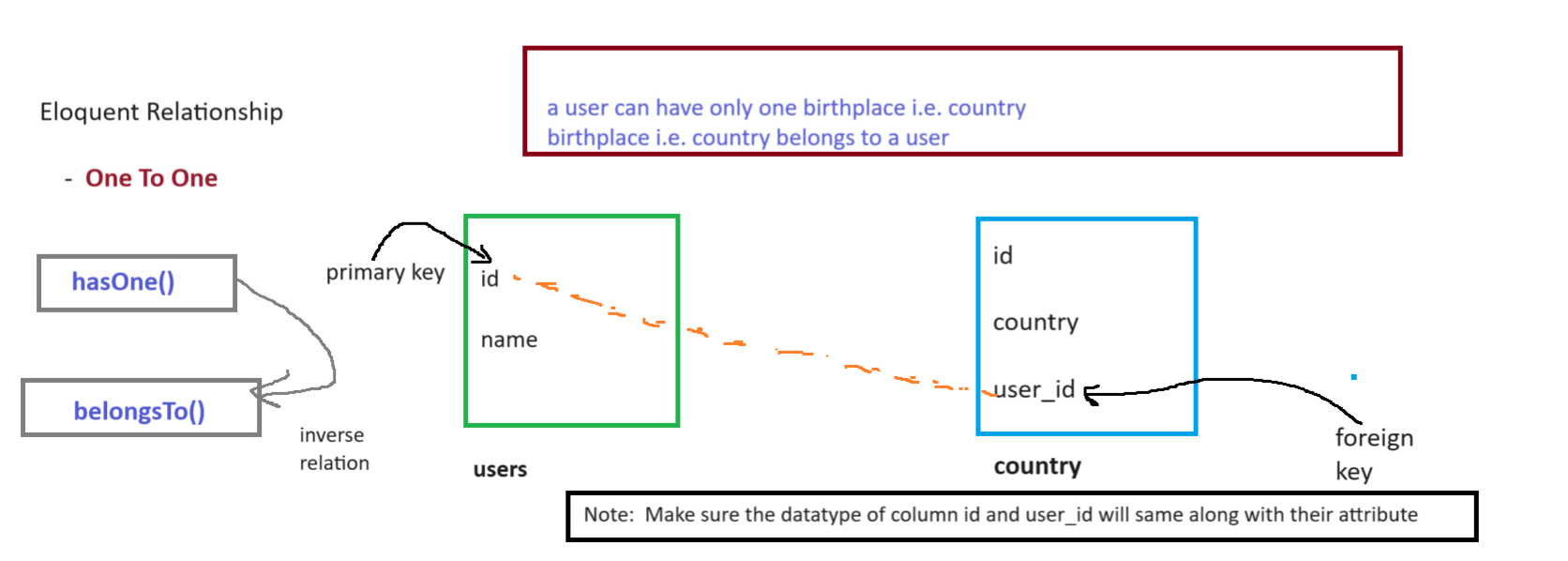
@endsection

Web.php

# CRUD Application------------

Route::resource('/todo',TodoController::class);

**Eloquent Relationship**



php artisan make:model Country -m

Country migration file

public function up(): void

{

Schema::create('countries', function (Blueprint $table) {

$table->id();

$table->string('country\_name');

#creating foreign key

$table->foreignId('user\_id')->references('id')->on('users');

$table->timestamps();

});

}

php artisan migrate

**Add Dummy Data using Factory and seeder**

php artisan make:factory CountryFactor –-model = Country

CountryFactory.php

public function definition(): array

{

return [

//

'user\_id'=>\App\Models\User::factory()->create()->id,

'country\_name'=>fake()->country()

];

}

DatabaseSeeder.php

public function run(): void

{

\App\Models\Country::factory(3)->create();

// \App\Models\User::factory()->create([

// 'name' => 'Test User',

// 'email' => 'test@example.com',

// ]);

}

**Add function to connect with Country Table in User Model**

User.php

public function country(){

return $this->hasOne(Country::class);

}

php artisan make:controller RelationalController

RelationalController.php

public function showOneToOneRelation(){

$users = User::all();

foreach($users as $u){

echo "<p>

<strong>".$u->name."</strong>

<span>has birthplace country name</span>

<strong>".$u->country->country\_name."</strong>

</p>";

}

}

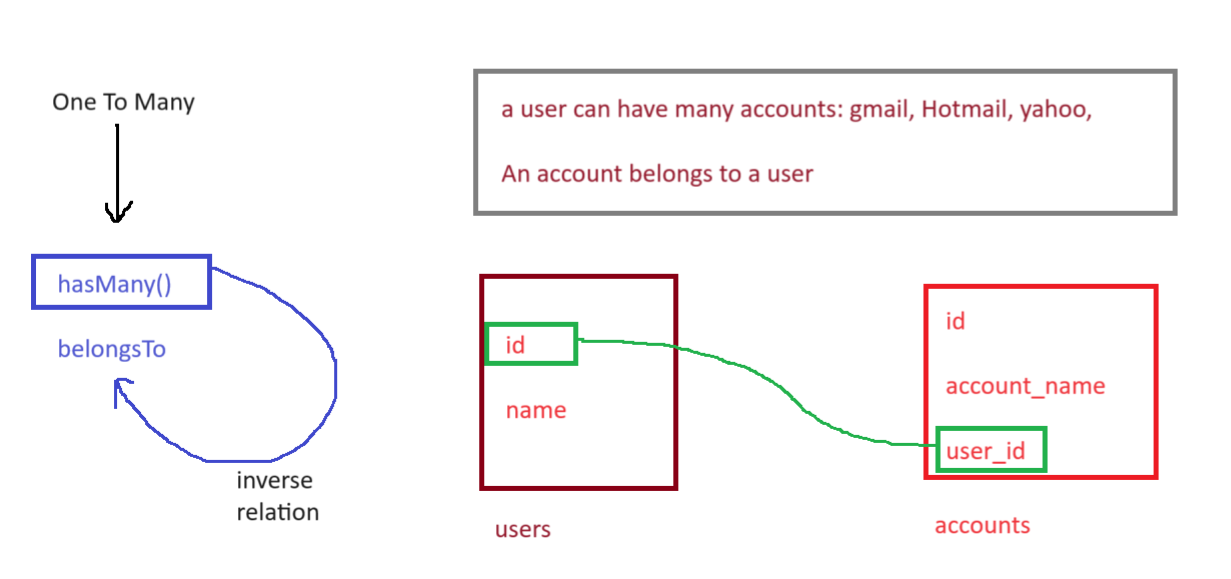
web.php

Route::get('/oneToOneRelation',[RelationalController::class,'showOneToOneRelation']);

**Day 4**

Attendance url

<https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732321>



php artisan make:model Account -m

Account Migration file

public function up(): void

{

Schema::create('accounts', function (Blueprint $table) {

$table->id();

$table->string('account\_name');

$table->foreignId('user\_id')->references('id')->on('users');

$table->timestamps();

});

}

php artisan migrate

**Add Dummy Data using Factory and seeder**

php artisan make:factory AccountFactory --model=Account

AccountFactory.php

public function definition(): array

{

return [

//

'user\_id'=>\App\Models\User::factory()->create()->id,

'account\_name'=>fake()->domainName()

];

}

DatabaseSeeder.php

public function run(): void

{

// \App\Models\Country::factory(3)->create();

\App\Models\Account::factory(6)->create();

// \App\Models\User::factory()->create([

// 'name' => 'Test User',

// 'email' => 'test@example.com',

// ]);

}

php artisan db:seed

**Add function to connect with AccountTable in User Model**

User.php

**public function account(){**

**return $this->hasMany(Account::class);**

**}**

**RelationalController.php**

**public function showOneToMany(){**

**$users = User::all();**

**foreach($users as $u){**

**echo '<p><strong>'. $u->name.'</strong> is using following email accounts</p>';**

**foreach($u->account as $acc){**

**echo '<li>'.$acc->account\_name.'</li>';**

**}**

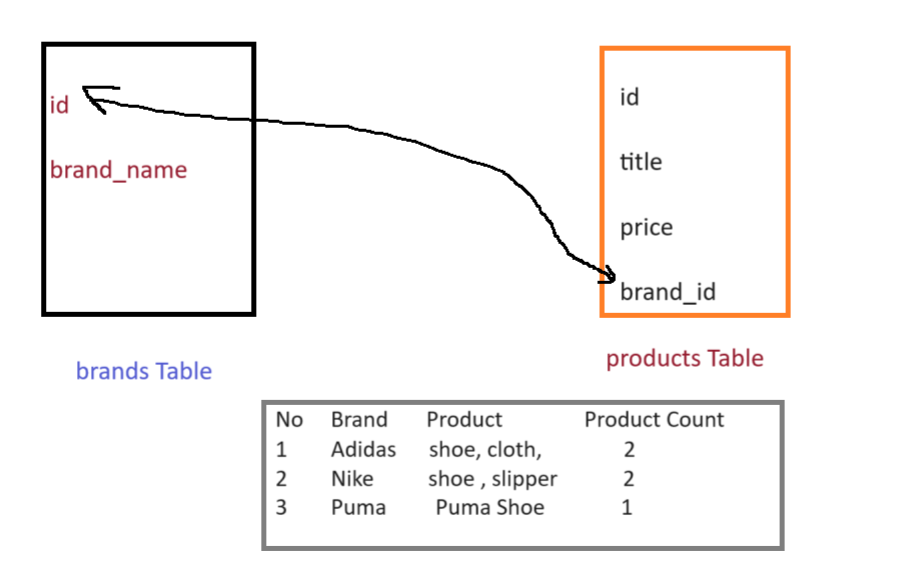
**}**

**}**

**web.php**

**Route::get('/oneToManyRelation',[RelationalController::class,'showOneToMany']);**

**Practice on One To Many**

****

**Solution:**

**php artisan make:model Brand -m**

**php artisan make:model product -m**

**Brand Migration file**

**public function up(): void**

**{**

**Schema::create('brands', function (Blueprint $table) {**

**$table->id();**

**$table->string('brand\_name');**

**$table->timestamps();**

**});**

**}**

**Product migration file**

**public function up(): void**

**{**

**Schema::create('products', function (Blueprint $table) {**

**$table->id();**

**$table->string('title');**

**$table->string('price');**

**$table->foreignId('brand\_id')->references('id')->on('brands');**

**$table->timestamps();**

**});**

**}**

**php artisan migrate**

**DatabaseSeeder.php**

**public function run(): void**

**{**

**// \App\Models\Country::factory(3)->create();**

**// \App\Models\Account::factory(6)->create();**

**// \App\Models\User::factory()->create([**

**// 'name' => 'Test User',**

**// 'email' => 'test@example.com',**

**// ]);**

**Brand::create([**

**'brand\_name'=>'Adidas'**

**]);**

**Brand::create([**

**'brand\_name'=>'Nike'**

**]);**

**Brand::create([**

**'brand\_name'=>'Specs'**

**]);**

**Brand::create([**

**'brand\_name'=>'Puma'**

**]);**

**Product::create([**

**'brand\_id'=>1,**

**'title'=>'Adidas shoes',**

**'price'=>'90'**

**]);**

**Product::create([**

**'brand\_id'=>1,**

**'title'=>'Adidas shoes 2',**

**'price'=>'94'**

**]);**

**Product::create([**

**'brand\_id'=>2,**

**'title'=>'Nike shoes',**

**'price'=>'90'**

**]);**

**Product::create([**

**'brand\_id'=>2,**

**'title'=>'Nike shoes 2',**

**'price'=>'90'**

**]);**

**Product::create([**

**'brand\_id'=>3,**

**'title'=>'Specs shoes',**

**'price'=>'100'**

**]);**

**Product::create([**

**'brand\_id'=>4,**

**'title'=>'Puma shoes',**

**'price'=>'200'**

**]);**

**}**

**}**

**php artisan db:seed**

**Brand.php**

**protected $guarded = [];**

**public function products(){**

**return $this->hasMany(Product::class);**

**}**

**Product.php**

**protected $guarded = [];**

**RelationalController.php**

**public function showOneToManyPractice(){**

**$brands = Brand::all();**

**echo '<table border="1">**

**<thead>**

**<tr>**

**<th>No</th>**

**<th>Brand</th>**

**<th>Product</th>**

**<th>Product Count</th>**

**</tr>**

**</thead>**

**';**

**$no = 1;**

**foreach($brands as $brand){**

**echo '<tr>**

**<td>'.$no++.'</td>**

**<td>'.$brand->brand\_name.'</td>**

**<td>**

**';**

**foreach($brand->products as $p){**

**echo $p->title;**

**}**

**echo '</td>**

**<td>'.$brand->products->count().'</td>**

**</tr>';**

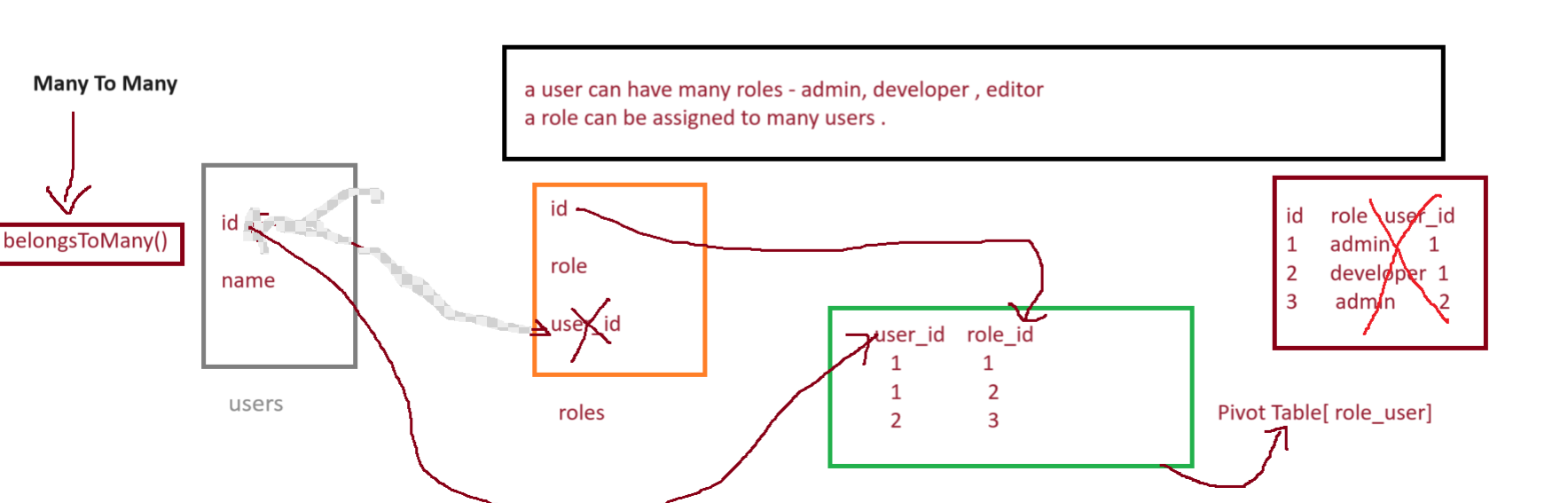
**}**

**echo '</table>';**

**}**

**Web.php**

**Route::get('/oneToManyRelation',[RelationalController::class,'showOneToManyPractice']);**

****

Note : - Pivot table name will be in alphabetical order

Example : user and role => role\_user

**php artisan make:model Role -m**

**Role migration file**

**public function up(): void**

**{**

**Schema::create('roles', function (Blueprint $table) {**

**$table->id();**

**$table->string('role');**

**$table->timestamps();**

**});**

**}**

**php artisan make:migration create\_role\_user**

**public function up(): void**

**{**

**Schema::create('role\_user', function (Blueprint $table) {**

**$table->id();**

**$table->foreignId('user\_id')->references('id')->on('users');**

**$table->foreignId('role\_id')->references('id')->on('roles');**

**$table->timestamps();**

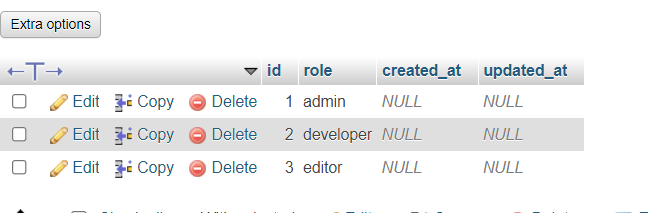
**});**

**}**

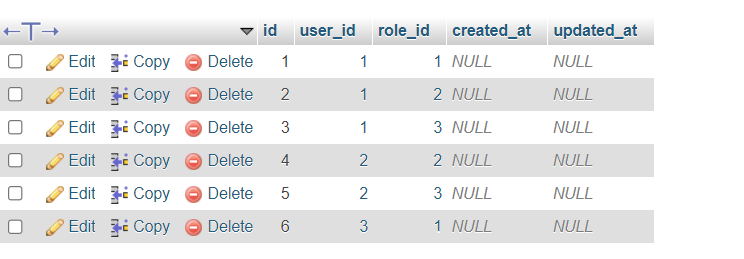
**php artisan migrate**

**Add the data**

roles table

****

**role\_user table**

****

**Model**

**User.php**

**public function role(){**

**return $this->belongsToMany(Role::class);**

**}**

**Role.php**

**public function user(){**

**return $this->belongsToMany(User::class);**

**}**

**RelationalController.php**

**public function ManyToMany(){**

**// $users = User::all();**

**// foreach($users as $u){**

**// echo '<p><strong>'.$u->name.'</strong> has following roles: -</p>';**

**// foreach($u->role as $r){**

**// echo '<li>'.$r->role.'</li>';**

**// }**

**// }**

**echo '<hr>';**

**$roles = Role::all();**

**foreach($roles as $r){**

**echo '<p><strong>'.$r->role.'</strong> is assigned to';**

**foreach($r->user as $u){**

**echo '<li>'.$u->name.'</li>';**

**}**

**echo '</p>';**

**}**

**}**

**web.php**

**Route::get('/manyToManyRelation',[RelationalController::class,'ManyToMany']);**

**Practice:- Many To Many**

Movie and Actor

Any movie could have many actors and vice versa.

**Laravel API Authentication with JWT**

**Install Package**

composer require php-open-source-saver/jwt-auth

**Publish the config file**

php artisan vendor:publish --provider="PHPOpenSourceSaver\JWTAuth\Providers\LaravelServiceProvider"

**Generate Secret Key**

php artisan jwt:secret

config/auth.php

'defaults' => [

'guard' => 'api',

'passwords' => 'users',

],

'guards' => [

'web' => [

'driver' => 'session',

'provider' => 'users',

],

'api' =>[

'driver' =>'jwt',

'provider' => 'users'

]

],

**Add Blacklist Exception**

**JWT\_SHOW\_BLACKLIST\_EXCEPTION = true**

**Prepare USER Model with JWT**

**use PHPOpenSourceSaver\JWTAuth\Contracts\JWTSubject;**

**class User extends Authenticatable implements JWTSubject**

**public function getJWTIdentifier(){**

**return $this->getKey();**

**}**

**public function getJWTCustomClaims(){**

**return [];**

**}**

**Preparing Controller**

php artisan make:controller Api/AuthController

**class AuthController extends Controller**

**{**

**//**

**public function register(Request $request){**

**$request->validate([**

**'name'=>'required',**

**'email'=>'required',**

**'password'=>'required'**

**]);**

**$user = User::create([**

**'name'=> $request['name'],**

**'email'=>$request['email'],**

**'password'=>bcrypt($request['password'])**

**]);**

**# login the user immediately and generate the token**

**$token = Auth::login($user);**

**# return response as json**

**return response()->json([**

**'meta' =>[**

**'code'=>200,**

**'status'=>'success',**

**'message'=> 'User created successfully'**

**],**

**'data'=>[**

**'user'=>$user,**

**'authorization'=>[**

**'token'=>$token,**

**'type'=> 'Bearer'**

**]**

**]**

**]);**

**}**

**public function login(Request $request){**

**$request->validate([**

**'email'=>'required',**

**'password'=>'required'**

**]);**

**# attempt a login**

**$token= Auth::attempt([**

**'email'=>$request->email,**

**'password'=>$request->password**

**]);**

**if($token){**

**return response()->json([**

**'meta' =>[**

**'code'=>200,**

**'status'=>'success',**

**'message'=> 'Quote fetched successfully'**

**],**

**'data'=>[**

**'user'=>Auth::user(),**

**'authorization'=>[**

**'token'=>$token,**

**'type'=> 'Bearer'**

**]**

**]**

**]);**

**}**

**}**

**public function logout(){**

**Auth::logout();**

**return response()->json([**

**'status'=>'success',**

**'message'=>'Successfully logged out'**

**]);**

**}**

**}**

**Create UserController**

php artisan make:controller Api/UserController

**class UserController extends Controller**

**{**

**//**

**public function userDetails(){**

**return response()->json([**

**'meta' =>[**

**'code'=>200,**

**'status'=>'success',**

**'message'=> 'User fetched successfully'**

**],**

**'data'=>[**

**'user'=>Auth::user(),**

**]**

**]);**

**}**

**}**

**Configure route**

routes/api.php

**Route::post('/register',[AuthController::class,'register']); #api/register**

**Route::post('/login',[AuthController::class,'login']); #api/login**

**Route::middleware('auth:api')->group(function(){**

**Route::get('/userInfo',[UserController::class,'userDetails']);**

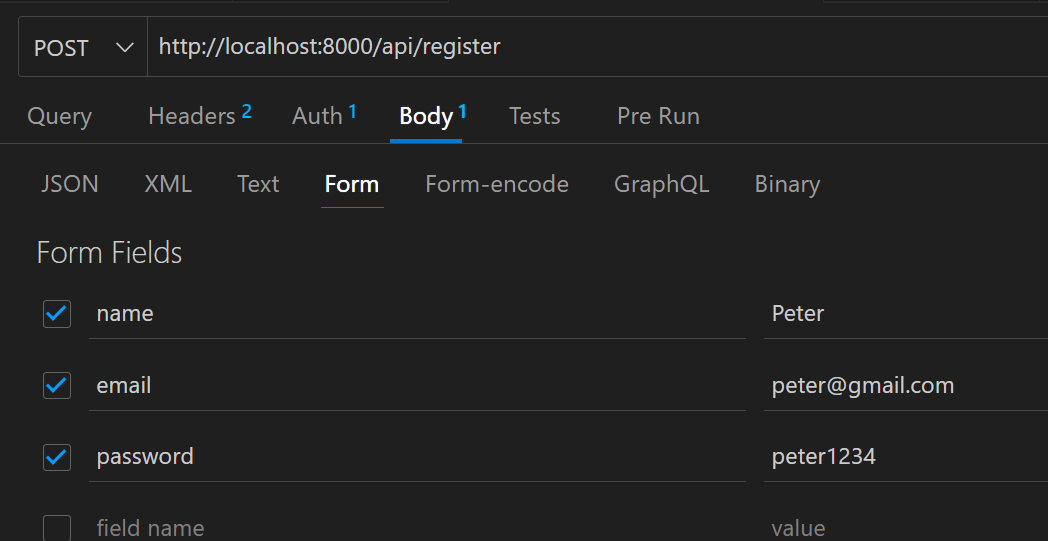
**Route::post('/logout',[AuthController::class,'logout']);**

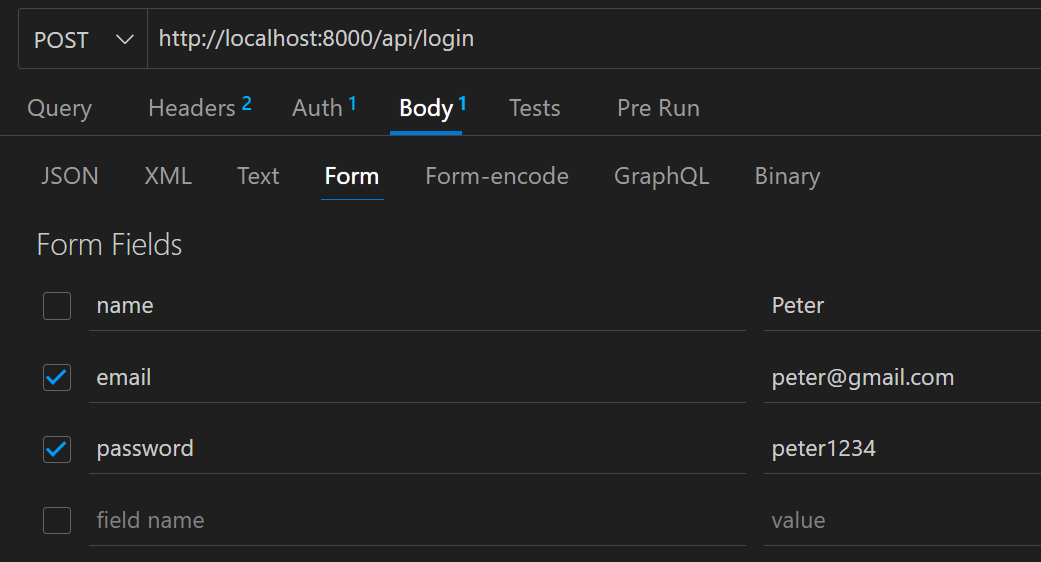
**});**

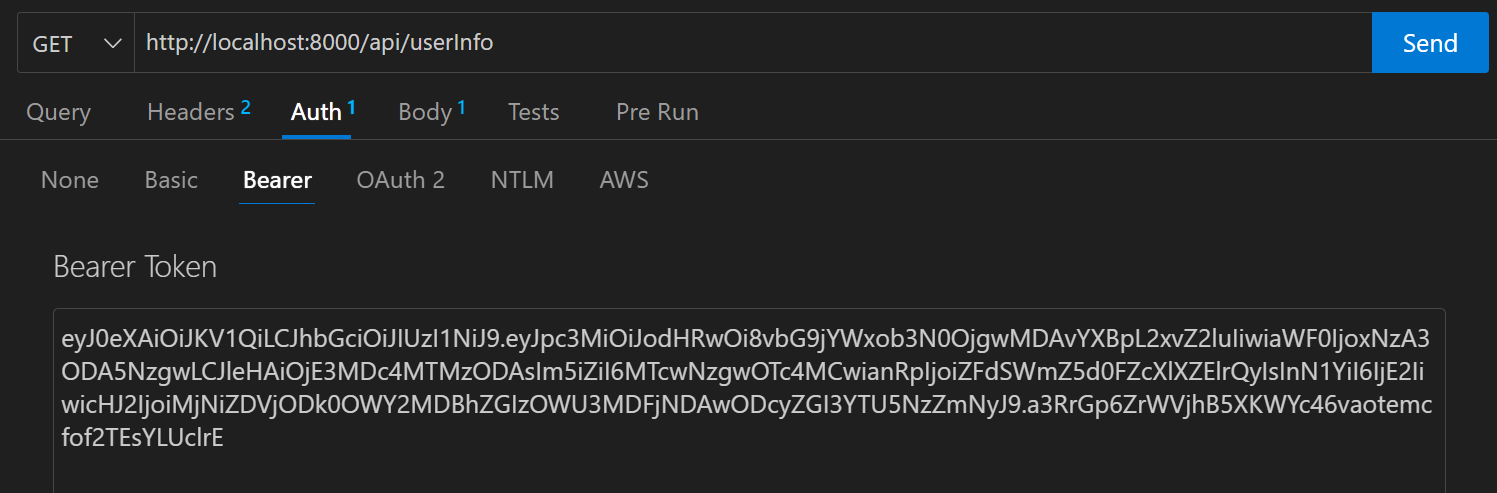
**php artisan serve**

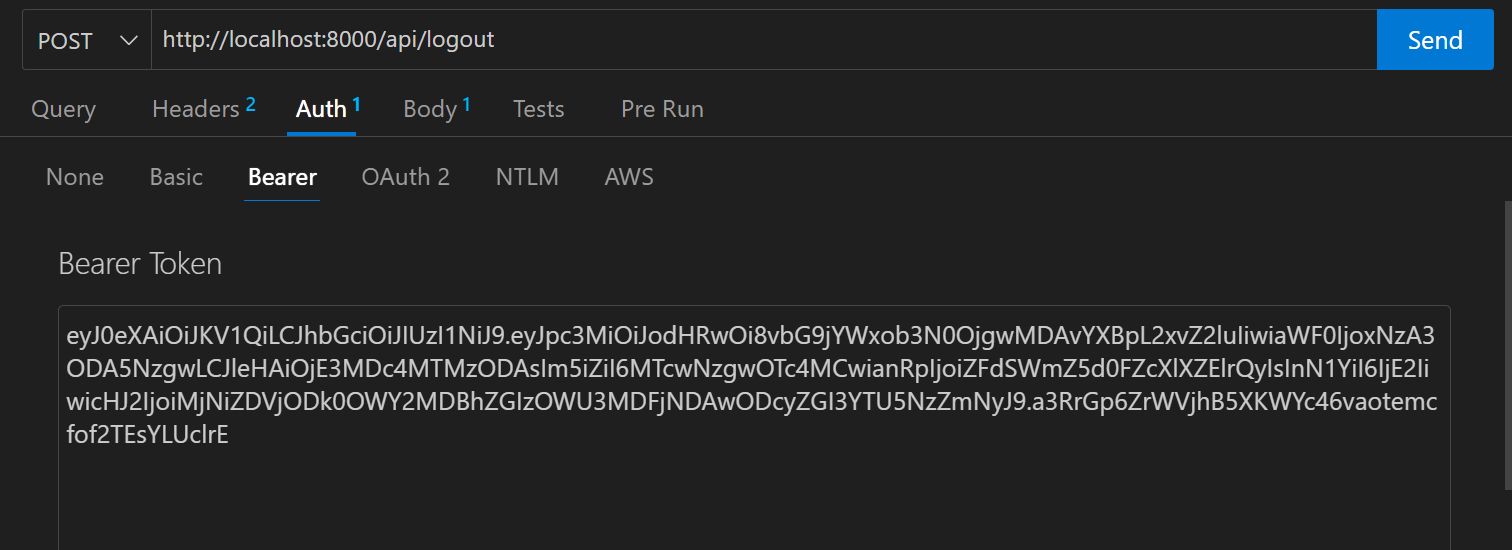
In VsCode, Add extension **Thunder client** for test api

**Click on New Request**

****

****

****

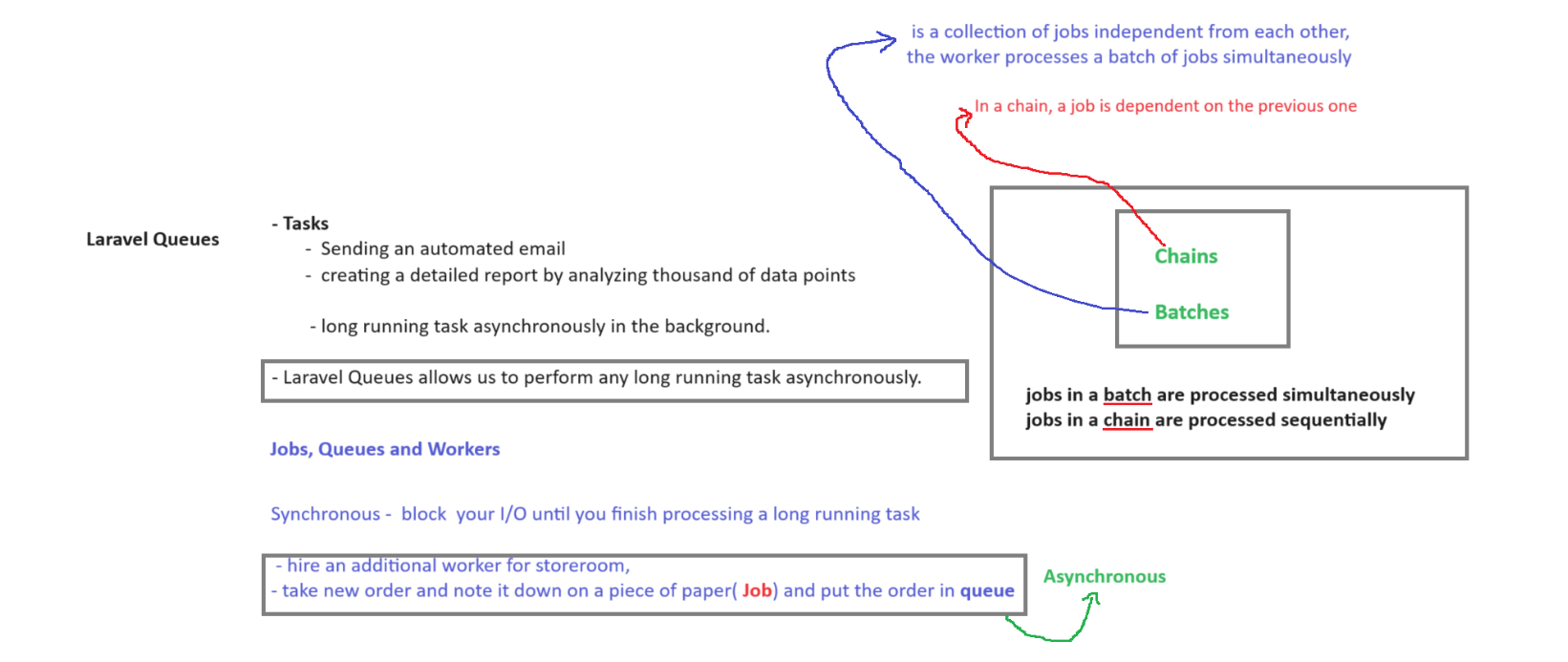
****

**Day 5**

**Attendance url**

[**https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732322**](https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732322)

**Queues**

****

**Example**

**Web.php**

**Route::get('/',function(){**

**// # dispatch() helper function that sends a new job to a queue**

**// dispatch(function(){**

**// sleep(5);**

**// logger('job done..!');**

**// });**

**// return view('welcome');**

php artisan serve

Check on browser - delay will be observed

**.env**

**QUEUE\_CONNECTION=database**

**# generate the migration script for creating jobs table**

php artisan queue:table

php artisan migrate

# **storage/logs/laravel.log** - check the logs

# check you jobs table to see the jobs

# **process the jobs start worker**

php artisan queue:work

# **Work with Job Classes**

php artisan make:job SendVerificationEmail

public function handle(): void

{

sleep(5);

logger('email sent');

}

#Web.php , put below code inside Route

#------Using Job Class-----------

# php artisan make:job SendVerifcationEmail

//dispatch(new SendVerificationEmail());

// SendVerificationEmail::dispatch();

// return view('welcome');

#**work with multiple connections**

web.php

#--------Work with multiple connection

// SendVerificationEmail::dispatch()->onConnection('redis');

// return view('welcome');

# **Working with multiple Queues**

Web.php

#-------Work with Multiple Queues--------------

#- you have jobs that must be processed before any of the others, regardless of which one comes first

// SendVerificationEmail::dispatch();

// SendVerificationEmail::dispatch()->onQueue('high-priority');

// return view('welcome');

# **Jobs Failure and Retries**

SendVerificationEmail.php

public function handle(): void

{

//

# intentionally making job fail

throw new Exception();

sleep(5);

logger('email sent');

}

php artisan queue:work

# **check failed jobs table**

php artisan queue:failed

# **retry : release failed job back to jobs table**

php artisan queue:retry jobid

# **Automatic Retries**

SendVerificationEmail.php

public $tries = 5;

php artisan queue:work

**CHAINS and BATCHES**

* Description is given in the picture above

# **Chains**

php artisan make:job EncodeVideoClip

php artisan make:job GenerateSubtitles

php artisan make:job PublishedVideoClip

Web.php

#---------Working with Chain-------------

// Bus::chain([

// new EncodeVideoClip,

// new GenerateSubtitles,

// new PublishVideoClip

// ])->dispatch();

// return view('welcome');

php artisan queue:work

# **Add code to failed jobs in chains**

GenerateSubtitles.php

\*/

public function handle(): void

{

//

throw new Exception();

}

php artisan queue:work

# **Batches**

Web.php

#-------Working with Batches---------

Bus::batch([

new EncodeVideoClip,

new GenerateSubtitles,

new PublishVideoClip,

])->dispatch();

# need to create jobs\_batches table

return view('welcome');

# **below table stores information about the job batches**

php artisan queue:batches-table

php artisan migrate

**Implement Batchable trait to each job class**

EncodingVideoClip.php

class EncodeVideoClip implements ShouldQueue

{

use Batchable, Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

Same step for another Job class

Run Application through browser

**php artisan queue:work**

**Example : Create jobs and Sending mail via queue**

.env

MAIL\_MAILER=smtp

MAIL\_HOST=smtp.gmail.com

MAIL\_PORT=465

MAIL\_USERNAME=YOUR EMAIL ID

MAIL\_PASSWORD=YOUR APP PASSWORD

MAIL\_ENCRYPTION=null

MAIL\_FROM\_ADDRESS="YOUR EMAIL ID"

MAIL\_FROM\_NAME="${APP\_NAME}"

# To generate **App Password** use below link

<https://myaccount.google.com/>

Security -> 2 step-verifications-> App Passwords -> Generate App Password

**Create Mail**

php artisan make:mail MyTestMail

use Illuminate\Mail\Mailables\Address;

class MyTestmail extends Mailable

{

use Queueable, SerializesModels;

public $data;

/\*\*

\* Create a new message instance.

\*/

public function \_\_construct($data)

{

//

$this->data = $data;

}

/\*\*

\* Get the message envelope.

\*/

public function envelope(): Envelope

{

return new Envelope(

from: new Address('gtrainerforttt@gmail.com','Nishant'),

subject: 'My First Mail Thorugh laravel',

);

}

/\*\*

\* Get the message content definition.

\*/

public function content(): Content

{

return new Content(

view: 'email.test',

with:[

'data'=>$this->data

]

);

}

/\*\*

\* Get the attachments for the message.

\*

\* @return array<int, \Illuminate\Mail\Mailables\Attachment>

\*/

public function attachments(): array

{

return [];

}

}

test.blade.php

<h2>Message notification</h2>

<div>

Message : {{$data['text']}}

</div>

Configure Queue

**.env**

QUEUE\_CONNECTION = database

**php artisan queue:table**

**php artisan migrate**

**Create Job**

php artisan make:job MyQueueMailJob

class MyQueueMailJob implements ShouldQueue

{

use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

public $data;

/\*\*

\* Create a new job instance.

\*/

public function \_\_construct($data)

{

//

$this->data = $data;

}

/\*\*

\* Execute the job.

\*/

public function handle(): void

{

//

Mail::to($this->data['email'])->send(new MyTestmail($this->data));

}

}

**web.php**

Route::get('/laravel\_test\_mail\_with\_queue',function(){

$data['text'] = 'We are sending email via queue';

$data['email'] = 'nishantharshwardhan@gmail.com';

dispatch(new MyQueueMailJob($data));

dd('Mail Send Successfully');

});

# put job into jobs table

<http://localhost:8000/laravel_test_mail_with_queue>

# start worker to process job

php artisan queue:work

Check you Email

**Example : import Large CSV File using Queue**

php artisan make:model Item -m

Item\_migration\_file

public function up(): void

{

Schema::create('items', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('amount');

$table->text('description');

$table->timestamps();

});

}

php artisan migrate

**Item.php (Model)**

**protected $guarded = ['\_token'];**

**Queue Configuration**

**.env**

QUEUE\_CONNECTION = database

**php artisan queue:table**

**php artisan queue:batches-table**

**php artisan migrate**

**Create Queue Job**

php artisan make:job ItemCSVData

https://laravel.com/docs/10.x/queues#supervisor-configuration

class ItemCSVData implements ShouldQueue

{

use Batchable, Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

public $header;

public $data;

/\*\*

\* Create a new job instance.

\*/

public function \_\_construct($data,$header)

{

//

$this->data = $data;

$this->header = $header;

}

/\*\*

\* Execute the job.

\*/

public function handle(): void

{

//

foreach($this->data as $item){

$itemInput = array\_combine($this->header, $item);

Item::create($itemInput);

}

}

}

Create Controller

php artisan make:controller ItemImportController

use Illuminate\Contracts\View\View;

class ItemImportController extends Controller

{

//

public function index(): View{

return view('itemimport');

}

public function store(Request $request){

if($request->has('csv')){

$csv = file($request->csv);

$chunks = array\_chunk($csv,1000);

$header = [];

$batch = Bus::batch([])->dispatch();

foreach($chunks as $key =>$chunk){

$data = array\_map('str\_getcsv',$chunk);

if($key == 0){

$header = $data[0];

unset($data[0]);

}

$batch->add(new ItemCSVData($data,$header));

}

return redirect('/items\_import')->with('status','CSV import added on queue');

}

}

}

Web.php

Route::get('/items\_import',[ItemImportController::class,'index']);

Route::post('/items\_import',[ItemImportController::class,'store']);

Itemimport.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h1>Import Large CSV File using Queue</h1>

@if(session('status'))

<strong>{{session('status')}}</strong>

@endif

{{Form::open(['url'=>'items\_import', 'files'=>true])}}

{{Form::file('csv')}}

{{Form::submit('Import')}}

{{Form::close()}}

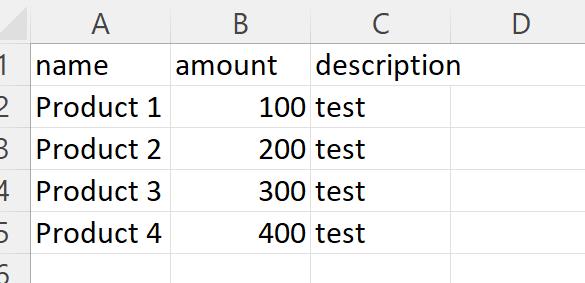
</body>

</html>

Check

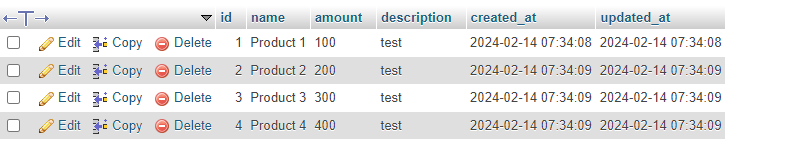
<http://localhost:8000/items_import>

Create csv file



**php artisan queue:work**

check table items

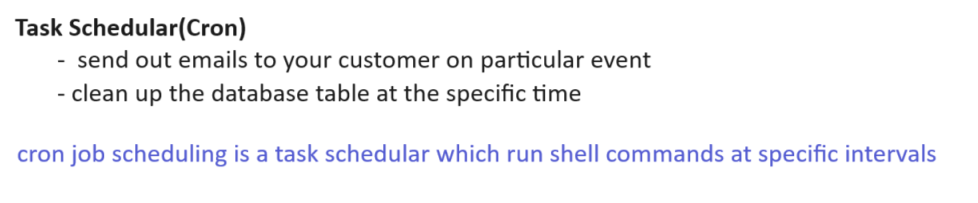


**Day 6**

Attendance url

<https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732328>

**Task Scheduling**



**# Create Laravel Artisan Command**

php artisan make:command DailyQuote

**# DailyQuote.php**

class DailyQuote extends Command

{

/\*\*

\* The name and signature of the console command.

\*

\* @var string

\*/

protected $signature = 'quote:daily';

/\*\*

\* The console command description.

\*

\* @var string

\*/

protected $description = 'Respectively send an exclusive quote to everyone daily via email';

/\*\*

\* Execute the console command.

\*/

public function handle()

{

//

$quotes = [

'Mother Teresa'=> "Spread love everywhere you go. Let no one ever come to you without leaving happier.",

'Franklin D. Roosevelt'=>'The only thing we have to fear is fear itself',

'Martin Luther King Jr.'=>'Darkness cannot drive out darkness: only light can do that. Hate cannot drive out hate: only love can do that',

'Do one thing every day that scares you.'=>'Eleanor Roosevelt'

];

# setting up a random word

$key = array\_rand($quotes);

$data = $quotes[$key];

$users = User::all();

foreach($users as $user){

Mail::raw("{$key} -> {$data}", function($mail) use ($user){

$mail->from('gtrainerforttt@gmail.com');

$mail->to($user->email)->subject('Daily New Quote');

});

};

$this->info('Successfully sent daily quote to everyone');

}

}

**#Register Task Scheduler Command**

app/console/kernal.php

class Kernel extends ConsoleKernel

{

protected $commands = [

Commands\DailyQuote::class

];

/\*\*

\* Define the application's command schedule.

\*/

protected function schedule(Schedule $schedule): void

{

// $schedule->command('inspire')->hourly();

$schedule->command('quote:daily')->everyMinute();

}

/\*\*

\* Register the commands for the application.

\*/

protected function commands(): void

{

$this->load(\_\_DIR\_\_.'/Commands');

require base\_path('routes/console.php');

}

}

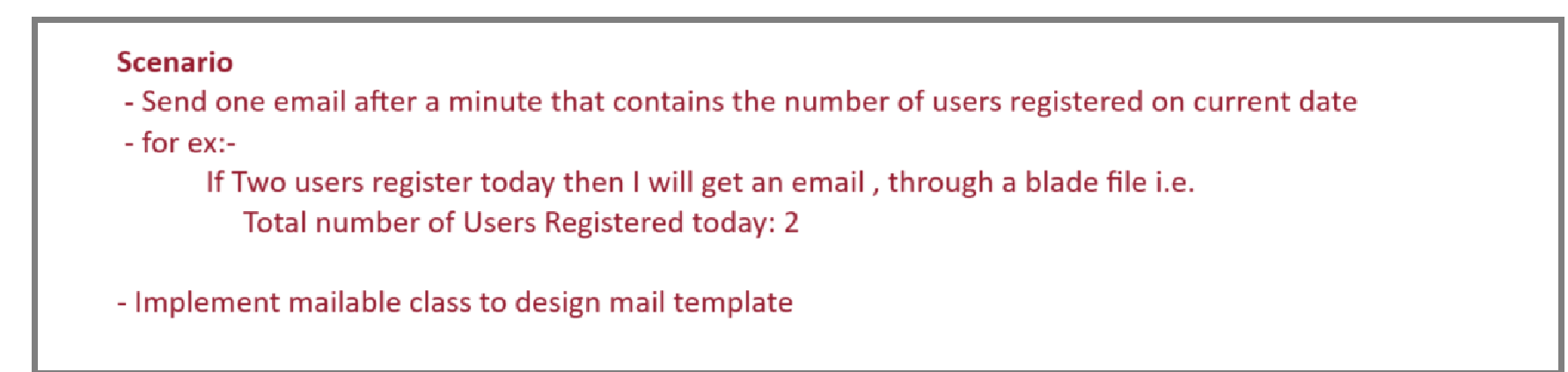
**# To Check the custom artisan command in list**

php artisan list

**# Schedule task**

php artisan quote:daily

php artisan schedule:work



Solutions:

php artisan make:command RegisteredUsers

class RegisteredUsers extends Command

{

/\*\*

\* The name and signature of the console command.

\*

\* @var string

\*/

protected $signature = 'app:registered-users';

/\*\*

\* The console command description.

\*

\* @var string

\*/

protected $description = 'Send an email of registered users';

/\*\*

\* Execute the console command.

\*/

public function handle()

{

//

$totalUsers = DB::table('users')->whereRaw('Date(created\_at)= CURDATE()')->count();

Mail::to('nishantharshwardhan@gmail.com')->send(new SendMailable($totalUsers));

}

}

php artisan make:mail SendMailable

class SendMailable extends Mailable

{

use Queueable, SerializesModels;

public $count;

/\*\*

\* Create a new message instance.

\*/

public function \_\_construct($count)

{

//

$this->count = $count;

}

/\*\*

\* Get the message envelope.

\*/

public function envelope(): Envelope

{

return new Envelope(

subject: 'User Registration Report',

);

}

/\*\*

\* Get the message content definition.

\*/

public function content(): Content

{

return new Content(

view: 'email.registercount',

);

}

/\*\*

\* Get the attachments for the message.

\*

\* @return array<int, \Illuminate\Mail\Mailables\Attachment>

\*/

public function attachments(): array

{

return [];

}

}

email/registercount.blade.php

<h3>Total number of User Registered today : {{$count}} </h3>

app/console/kernal.php

class Kernel extends ConsoleKernel

{

protected $commands = [

// Commands\DailyQuote::class

Commands\RegisteredUsers::class

];

/\*\*

\* Define the application's command schedule.

\*/

protected function schedule(Schedule $schedule): void

{

// $schedule->command('inspire')->hourly();

// $schedule->command('quote:daily')->everyMinute();

$schedule->command('app:registered-users')->everyMinute();

}

/\*\*

\* Register the commands for the application.

\*/

protected function commands(): void

{

$this->load(\_\_DIR\_\_.'/Commands');

require base\_path('routes/console.php');

}

}

DatabaseSeeder.php

public function run(): void

{

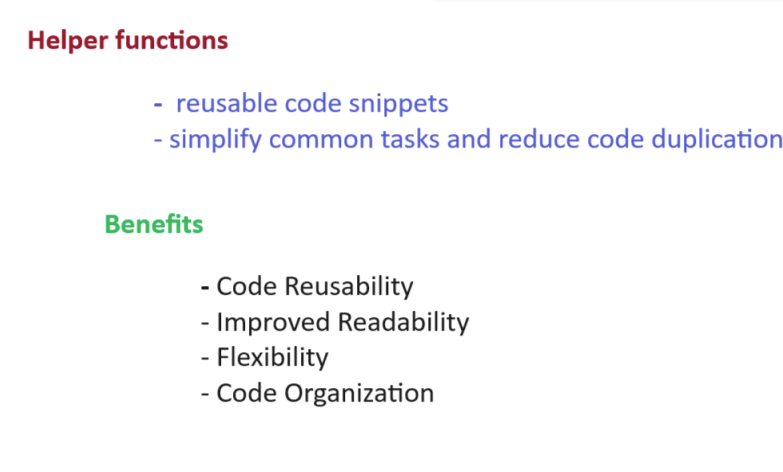
\App\Models\User::factory(3)->create();

**Run**

php artisan app:registered-users

php artisan schedule:work

**Helper Functions**

****

**# Creating Custom helper Function**

**# Create folder : App\Helpers\helpers.php**

**<?php**

**use Illuminate\Support\Str;**

**use Illuminate\Http\UploadedFile;**

**if(!function\_exists('UploadFiled')){**

**function UploadMyFile(UploadedFile $file, $folder=null, $filename=null,$disk='s3'){**

**$name= is\_null($filename)? $filename: Str::random(10);**

**return $file->storeAs(**

**$folder,**

**$name.".".$file->getClientOriginalExtension(),**

**$disk**

**);**

**}**

**}**

**?>**

**# Create Service Provider**

php artisan make:provider HelperServiceProvider

class HelperServiceProvider extends ServiceProvider

{

/\*\*

\* Register services.

\*/

public function register(): void

{

//

$files = glob(app\_path('Helpers')."/\*.php");

foreach($files as $key =>$file){

require\_once $file;

}

}

/\*\*

\* Bootstrap services.

\*/

public function boot(): void

{

//

}

}

# Register Service Provider

config/app.php

'providers' => ServiceProvider::defaultProviders()->merge([

/\*

\* Package Service Providers...

\*/

/\*

\* Application Service Providers...

\*/

App\Providers\HelperServiceProvider::class,

# Using the function

TestController.php

public function upload(Request $request){

$file\_path = UploadMyFile($request->file('csv'),'Products');

dd($file\_path);

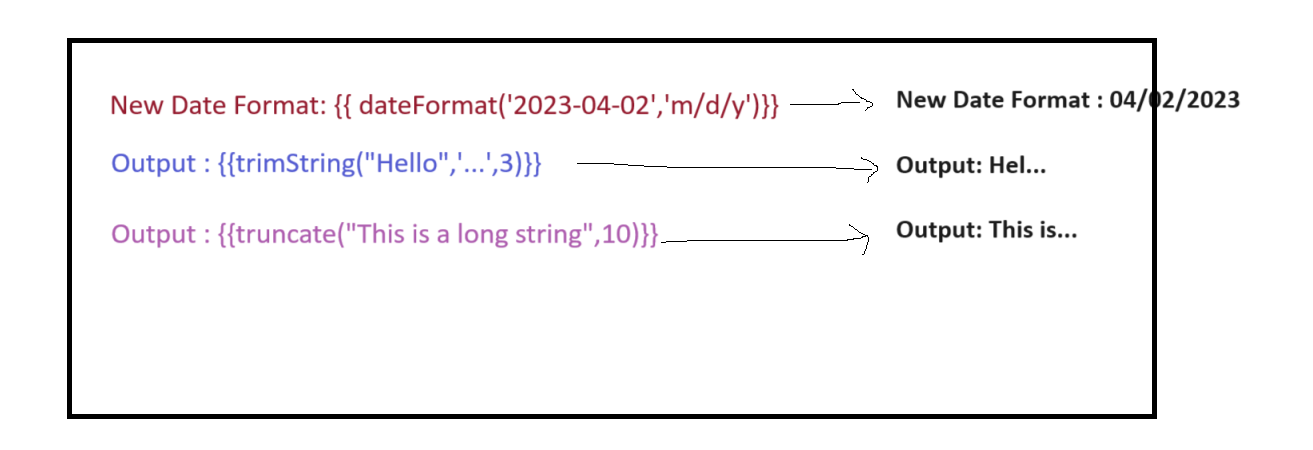
}

Web.php

Route::get('/items\_import',[ItemImportController::class,'index']);

Route::post('/items\_import',[TestController::class,'upload']);

Practice



Solution

helpers.php

function dateFormat($date,$format){

return \Carbon\Carbon::createFromFormat('Y-m-d',$date)->format($format);

}

function trimString($string,$repl,$limit){

if(strlen($string)> $limit){

return substr($string,0,$limit).$repl;

}

else{

return $string;

}

}

function truncate($string,$length){

if(strlen($string)> $length){

return substr($string,0,$length-3).'...';

}

return $string;

}

Blade file

<h4>New Date Format: {{dateFormat('2023-04-02','m/d/Y')}}</h4>

<p>Output : {{trimString("Hello","...",3)}}</p>

<p>Output : {{truncate("This is a long string",10)}}</p>

Error Handling

web.php

Route::get('/search',[TestController::class, 'index']);

Route::get('/search/result',[TestController::class, 'result']);

TestController.php

public function index(){

return view('search');

}

public function result(Request $request){

# give us an error

// $user = User::find($request->user\_id);

// return view('result',compact('user'));

# handle Error

try{

$user = User::findOrFail($request->user\_id);

return view('result',compact('user'));

}

catch(Exception $e){

$message = $e->getMessage();

if($e instanceof ModelNotFoundException){

$message = 'User with Id: '.$request->user\_id.' not found';

}

}

return back()->withError($message)->withInput();

}

.env

APP\_DEBUG=false

Search.blade.php

{{Form::open(['url'=>'/search/result','method'=>'GET'])}}

<h2>Get Users By ID</h2>

@if(session('error'))

{{session('error')}}

@endif

<br><br>

{{Form::text('user\_id',null,['placeholder'=>'Enter user Id'])}}

{{Form::submit('Search')}}

{{Form::close()}}

Result.blade.php

<h1>Result Found</h1>

<ul>

<li>{{$user->name}}</li>

<li>{{$user->email}}</li>

</ul>

**Day 7**

Attendance url

<https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732329>

Install NodeJS

<https://nodejs.org/en>

**#Authentication**

composer require laravel/breeze —dev

php artisan breeze:install blade

**#Change guard value in auth.php, only for current project**

'defaults' => [

'guard' => 'web',

'passwords' => 'users',

],

**# Authorization**

php artisan make:migration add\_role\_column\_to\_users\_table

public function up(): void

{

Schema::table('users', function (Blueprint $table) {

//

$table->string('role')->default('user');

});

}

php artisan migrate



AuthServiceProvider.php

public function boot(): void

{

//

Gate::define('is\_admin', function(User $user){

return $user->role=='admin';

});

}

dashboard.blade.php

<div class="py-12">

<div class="max-w-7xl mx-auto sm:px-6 lg:px-8">

<div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">

<div class="p-6 text-gray-900">

{{ \_\_("You're logged in!") }}

</div>

<div class="p-6 text-gray-900">

@can('is\_admin')

<div class="btn btn-sucess btn-lg">

Hello, Admin

</div>

@else

<div class="btn btn-sucess btn-lg">

Hello, User

</div>

@endcan

</div>

</div>

</div>

</div>

**Check on browser with both role**

**# Gate in Routes**

web.php

#Gates in Routes

Route::get('/user/delete',[TestController::class,'delete'])->middleware('can:is\_admin');

Route::get('/user/show',[TestController::class,'show'])->middleware('can:is\_user');

TestController.php

public function delete(){

dd('inside delete method');

}

public function show(){

dd('inside show method');

}

**#Gates in Controller**

web.php

#Gates in Controller

Route::get('/user/delete',[TestController::class,'delete']);

Route::get('/user/show',[TestController::class,'show']);

TestController.php

public function delete(){

if(!Gate::allows('is\_admin')){

abort(403);

}

dd('inside delete method');

}

public function show(){

if(Gate::denies('is\_user')){

abort(403);

}

dd('inside show method');

}

**#Policies**

**php artisan make:model Post**

**php artisan make:policy PostPolicy --model=Post**

**# Register Policies**

AuthServiceprovider.php

class AuthServiceProvider extends ServiceProvider

{

/\*\*

\* The model to policy mappings for the application.

\*

\* @var array<class-string, class-string>

\*/

protected $policies = [

//

Post::class => PostPolicy::class

];

/\*\*

\* Register any authentication / authorization services.

\*/

public function boot(): void

{

//

$this->registerPolicies();

Gate::define('is\_admin', function(User $user){

return $user->role=='admin';

});

}

}

**#Create migration file**

php artisan make:migration create\_post\_table

public function up(): void

{

Schema::create('posts', function (Blueprint $table) {

$table->id();

$table->string('title');

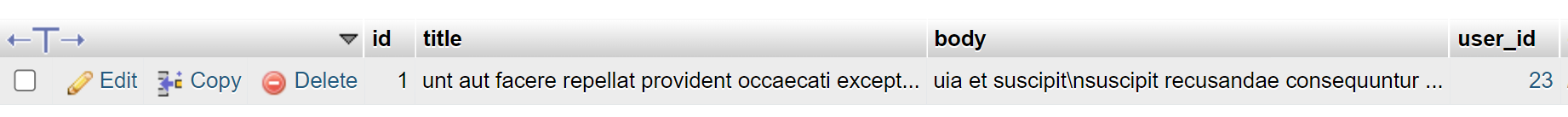
$table->string('body');

$table->foreignId('user\_id')->references('id')->on('users');

$table->timestamps();

});

}



PostPolicy.php

public function update(User $user, Post $post)

{

//

// return $user->role==='admin'|| $user->id === $post->user\_id;

return $user->id === $post->user\_id;

}

Apply admin role for all method

public function before(User $user, $ability){

if($user->role === 'admin'){

return true;

}

}

web.php

# Policies route

// Route::get('/posts/edit/{post}',[TestController::class,'edit']);

// Route::get('/posts/edit/{post}',[TestController::class,'edit'])->can('update','post');

Route::get('/posts/edit/{post}',[TestController::class,'edit'])->middleware('can:update,post');

TestController.php

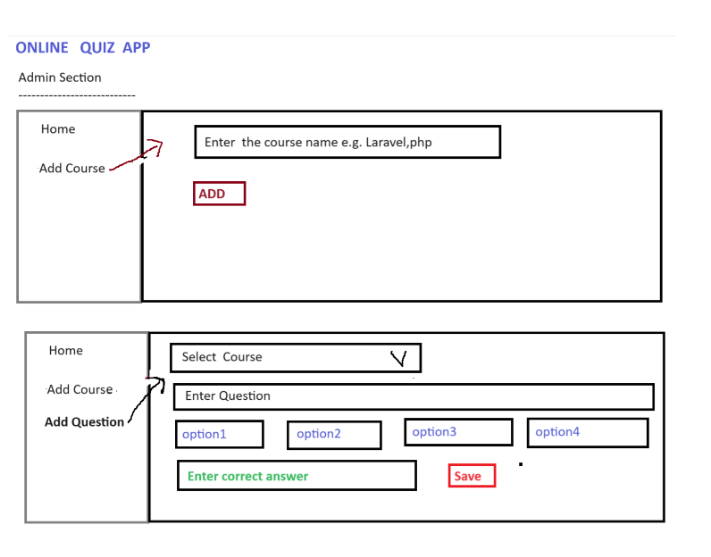
public function edit(Post $post){

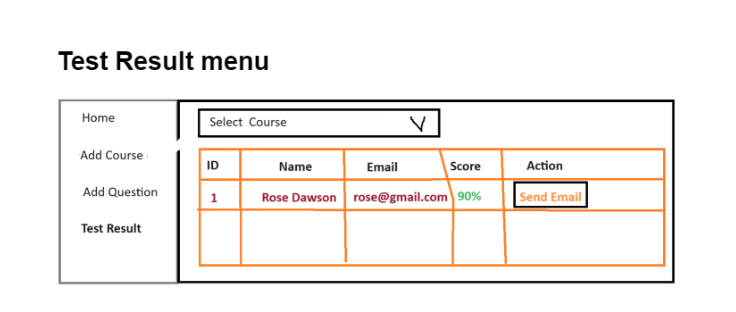
// $this->authorize('update',$post);

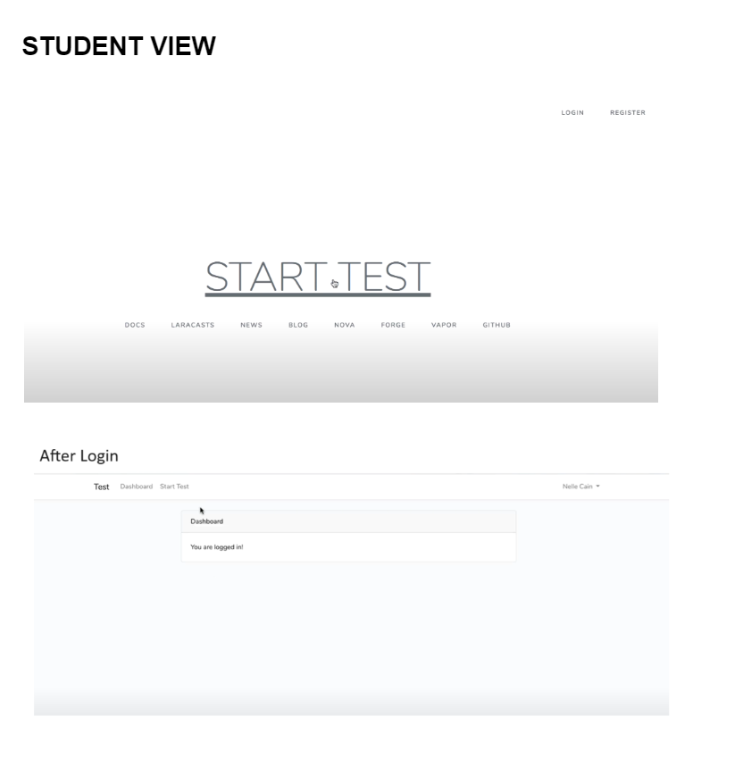
return 'You can edit this post';

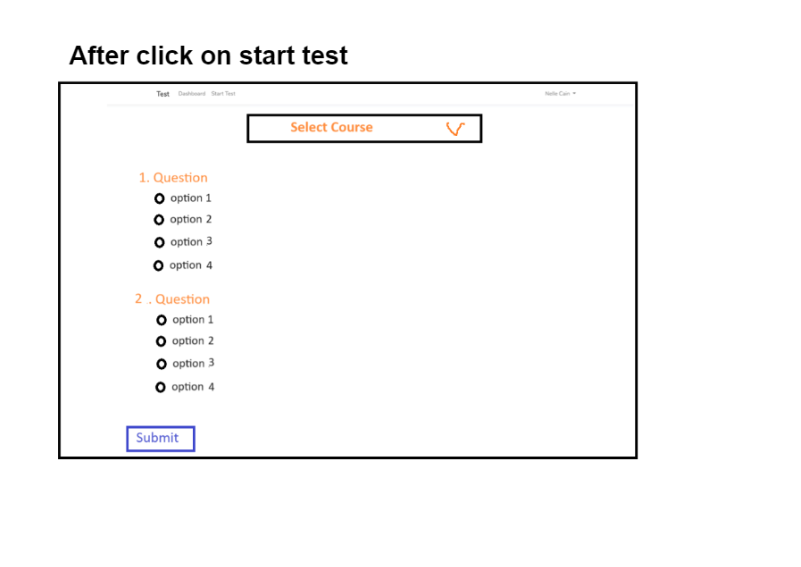
}

Practice: Authentication & Authorization









**Day 8**

Attendance url

<https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732330>

<https://github.com/barryvdh/laravel-debugbar>

1. composer require barryvdh/laravel-debugbar --dev
2. APP\_DEBUG is true.
3. App.php

'providers' => ServiceProvider::defaultProviders()->merge([

…..

Barryvdh\Debugbar\ServiceProvider::class,

])->toArray(),

'aliases' => Facade::defaultAliases()->merge([

// 'Example' => App\Facades\Example::class,

'Debugbar' => Barryvdh\Debugbar\Facades\Debugbar::class,

])->toArray(),

1. DEBUGBAR\_ENABLED=true in your .env
2. php artisan vendor:publish --provider="Barryvdh\Debugbar\ServiceProvider"

Code

Web.php

Route::get('/debug',[TestController::class,'debug']);

TestController.php

public function debug(){

// $data = User::all();

// $data = User::all()->toArray();

// $data = User::toSql();

// $data = User::Where('name','peter')->toSql();

// $data = User::Where('name','peter')->get();

// $data = User::Where('name','peter')->dd();

// $data = User::Where('name','peter')->orWhere('email','peter@gmail.com')->dd();

// dd($data);

// ------Another way-----------

// DB::enableQueryLog();

// $data = User::Where('name','peter')->orWhere('email','peter@gmail.com')->get();

// dd(DB::getQueryLog());

// ------Using Debugger library----------

$data = User::all();

return view('test',['data'=>$data]);

}

Test.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

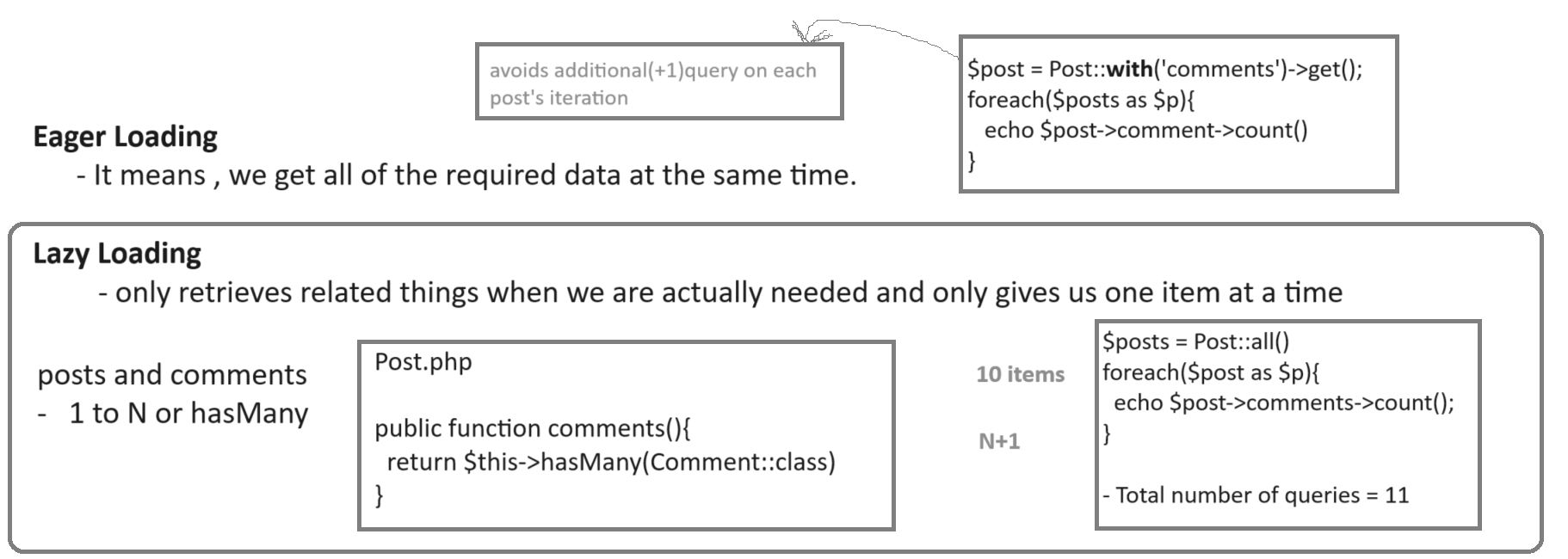
@foreach($data as $d)

<li>{{$d->name}}</li>

@endforeach

</body>

</html>



Work with Labs

* Quiz App

**Adding css and js in vite**

@import url('../theme/css/style.css');

import '../theme/js/script.js'

**Day 9**

**Attendance url**

<https://rms.koenig-solutions.com/MarkAssignmentAttendance.aspx?AId=183539&Opid=732331>

<https://laravel.com/docs/10.x/telescope>

composer require laravel/telescope --dev

php artisan telescope:install

php artisan migrate

**Open browser and check telescope panel through below url**

<http://localhost:8000/telescope>