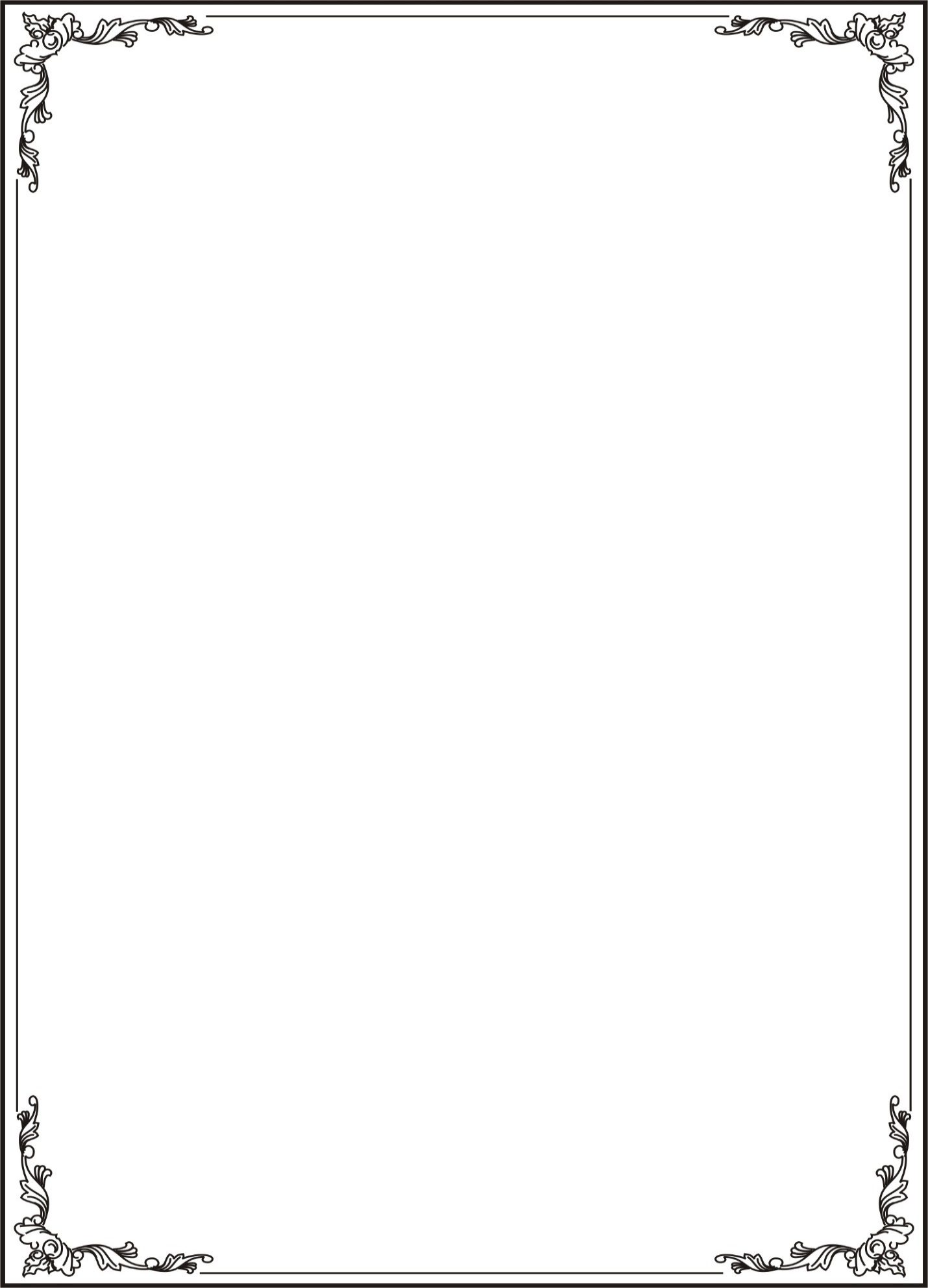
**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**---🙠🙠🙠 🕮 🙢🙢🙢---**



**BÁO CÁO MÔN HỌC: KỸ THUẬT PHÁT TRIỂN HỆ THỐNG WEB**

**CHỦ ĐỀ BÁO CÁO: Tìm hiểu về Framework Laravel**

Giảng viên hướng dẫn: **Ths. Võ Ngọc Tân**

|  |  |
| --- | --- |
| Nhóm sinh viên thực hiện: | 17520334 – Nguyễn Quang Đạt |
|  | 17520765 – Trần Thị Ngọc Minh |
|  | 17520687 – Nguyễn Ngọc Anh Linh |
|  | 17520979 – Trương Hữu Sang |
|  | 17520448 – Nguyễn Mạnh Hào |
|  | 17520357 – Lê Huỳnh Đức |
|  | 17520673 – Lê Thị Ngọc Lan |
|  | 17520704 – Trần Nguyên Lợi |
|  | 17520914 – Nguyễn Lê Thanh Phụng |
|  | 17520795 – Nguyễn Hữu Nghị |

Mục lục

[**I.** **GIỚI THIỆU LARAVEL VÀ HƯỚNG DẪN CÀI ĐẶT** 2](#_Toc38268818)

[**1.** **Giới thiệu** 3](#_Toc38268819)

[**2.** **Cài đặt** 3](#_Toc38268820)

[**II.** **Sử dụng Laravel căn bản:** 10](#_Toc38268821)

[**1.** **Route** 10](#_Toc38268822)

[a) Cấu trúc thư mục 10](#_Toc38268823)

[b) Tham số tùy chọn 11](#_Toc38268824)

[c) Tên Route 11](#_Toc38268825)

[d) Nhóm Route 11](#_Toc38268826)

[**2.** **Controller** 12](#_Toc38268827)

[a) Cấu trúc controller 12](#_Toc38268828)

[b) Tạo controller: 13](#_Toc38268829)

[c) Gọi controller 18](#_Toc38268830)

[**3.** **View** 19](#_Toc38268831)

[a) Tạo view trong Laravel 20](#_Toc38268832)

[b) Gọi view trong Laravel 20](#_Toc38268833)

[c) Chia sẻ dữ liệu trong View 24](#_Toc38268834)

[**4.** **Model** 24](#_Toc38268835)

[a) Cách tạo Model 24](#_Toc38268836)

[b) Khai báo các thông số tùy chỉnh 26](#_Toc38268837)

[**5.** **Database** 26](#_Toc38268838)

[a) Một vài câu lệnh kết nối Database 26](#_Toc38268839)

[b) Các truy vấn (Select, Insert, Update, Delete, Statement, Listen) 27](#_Toc38268840)

[c) Database Transactions 28](#_Toc38268841)

[d) Nhật ký truy vấn 28](#_Toc38268842)

[e) Trình tạo truy vấn 29](#_Toc38268843)

[**6.** **Middleware trong Laravel** 34](#_Toc38268844)

[a) Giới thiệu 34](#_Toc38268845)

[b) Tạo Middleware 35](#_Toc38268846)

[c) Before và After Middle 36](#_Toc38268847)

[d) Đăng kí Middleware 37](#_Toc38268848)

[e) Tạo nhóm middleware 38](#_Toc38268849)

[f) Sắp xếp Middleware 40](#_Toc38268850)

[g) Middleware Parameters 40](#_Toc38268851)

[h) Terminable Middleware 41](#_Toc38268852)

[**7.** **Auth – Authentication** 42](#_Toc38268853)

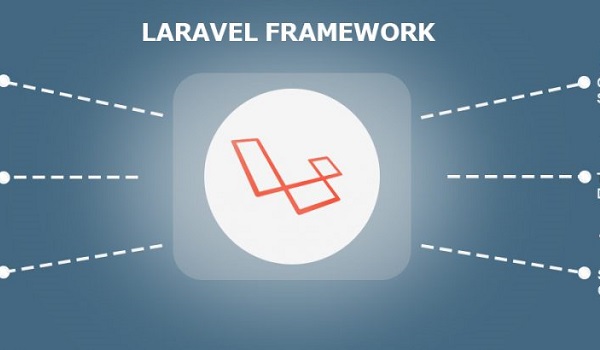
[a) Giới thiệu 42](#_Toc38268854)

[b) Tạo route và view cho Auth 43](#_Toc38268855)

[c) Một số hàm của class Auth 43](#_Toc38268856)

1. **GIỚI THIỆU LARAVEL VÀ HƯỚNG DẪN CÀI ĐẶT**
   1. **Giới thiệu**

* **Laravel là một PHP framework, có mã nguồn mở và miễn phí, được xây dựng nhằm hỗ trợ phát triển các phần mềm, ứng dụng, theo kiến trúc MVC.**Hiện nay, Laravel đang là PHP framework phổ biến nhất và tốt nhất.



Hình 1 - Giới thiệu Laravel

* Những lý do khiến Laravel trở nên rộng rãi:
* Cú pháp dễ hiểu – rõ ràng
* Hệ thống đóng gói modular và quản lý gói phụ thuộc
* Nhiều cách khác nhau để truy cập vào các cơ sở dữ liệu quan hệ
* Nhiều tiện ích khác nhau hỗ trợ việc triển khai vào bảo trì ứng dụng.
  1. **Cài đặt**
* Yêu cầu hệ thống để cài đặt Laravel:
* Để có thể cài Laravel, bạn phải đáp ứng được các yêu cầu bắt buộc sau:
* PHP >= 5.5.9
* OpenSSL PHP Extension
* PDO PHP Extension
* Mbstring PHP Extension
* Tokenizer PHP Extension
* Đối với Windows: sử dụng phần mềm tạo Webserver trên Windows như Openserver, Wamp, Xampp, Ampps…
* Thông qua Composer

Vào trang <https://getcomposer.org/download/> và tải về

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Hình 2 - Tải file Composer về máy

Tiến hành cài đặt Composer

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Hình 3 - Cài đặt Composer – Install for all users

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Hình 4 - Cài đặt Composer bước tiếp theo – Composer Setup

Vào Webserver của bạn, ở đây mình dùng Xampp, nhấn giữ phím Shift + chuột trái chọn mục Open PowerShell Window here

Ảnh có chứa ảnh chụp màn hình

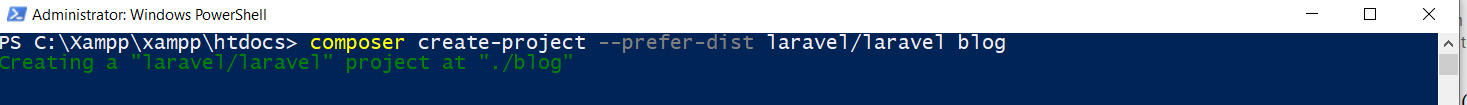
Mô tả được tạo tự động

Hình 5 - Tiến hành mở Power shell

Chúng ta sẽ di chuyển thẳng vào thư mục htdocs của XAMPP, tại đây các bạn mở cửa sổ lệnh (như trên) và gõ dòng lệnh sau:



Trong đó blog chính là tên thư mục laravel project của bạn.



Cài đặt hoàn tất

Ảnh có chứa máy tính

Mô tả được tạo tự động

Hình 6 - Hoàn tất cài đặt

Thư mục blog xuất hiện trong mục htdocs

Ảnh có chứa ảnh chụp màn hình, máy tính

Mô tả được tạo tự động

Mở localhost chọn vào thư mục blog vừa tạo, chọn thứ mục public

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Cài đặt thành công

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Hình 7 - Cài đặt thành công

1. **Sử dụng Laravel căn bản:**
   1. **Route**

* Khái niệm: Route thực hiện chức năng định tuyến cho những đường dẫn(URL) tới 1 function() hành động giúp server đưa ra những thông tin mà người dùng cần
  + 1. **Cấu trúc thư mục**
* Khi muốn tạo các Route trong Laravel thì chúng ta sẽ viết trong routes/web.php – định nghĩa các route cho web.
* Còn routes/api.php – để định nghĩa route cho api.
* Cấu trúc route
* ***Route::get($url, $callback);***
* Trong đó $url: đường dẫn route và $callback : là một hành động nào đó sẽ được thực hiện để trả về.

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

return view('welcome');

});

* Các phương thức laravel hỗ trợ

oute::get($uri, $callback);

Route::post($uri, $callback);

Route::put($uri, $callback);

Route::patch($uri, $callback);

Route::delete($uri, $callback);

Route::options($uri, $callback);

* + 1. **Tham số tùy chọn**
* Nhiều khi chúng ta sẽ cần truyền tham số trên đường dẫn, chúng ta cũng có thể truyền trong route. Ví dụ,chúng ta cần lấy ID của user từ URL, chúng ta có thể viết route như sau:

Route::get('user/{id}', function($id) {

echo "ID của user là : " . $id;

});

Ngoài ra chúng ta cũng có thể truyền nhiều tham số trong URL

Route::get('user/{id}/{name}/{comment}', function($id, $name, $comment) {

echo "ID của user là : " . $id;

echo "<br>Tên của user là : " . $name;

echo "<br> Comment của user: " . $comment;

});

* + 1. **Tên Route**
* Tên routes cho phép chúng ta thuận tiện hơn khi chuyển hướng các route cụ thể. Chúng ta có thể đặt tên route bằng cách thêm name khi chúng ta định nghĩa route.

Route::get('posts', function () {

//code

})->name('nameroute');

Route::get('posts', [‘as’=>’nameroute’, function () {

//code

}]);

* + 1. **Nhóm Route**

Nhiều khi những route có chung hành vi thì chúng ta sẽ nhóm route vào.

Route::group(['namespace' => 'Admin'], function () {

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

//code

});

Route::get('posts', function () {

//code

});

});

Namespace trong Laravel giông như PHP namespace được chỉ định với một nhóm controller.

Xét ví dụ

Route::get('admin/posts', 'PostController@index');

Route::get('admin/posts/create', 'PostController@create');

Route::post('admin/posts/store', 'PostController@store');

Route::get('admin/posts/{id}/edit', 'PostController@edit');

Route::patch('admin/posts/{id}', 'PostController@update);

Route::patch('admin/posts/{id}', 'PostController@destroy');

Các bạn có thấy đặc điểm chung của các route trên không, nó đều bắt đầu bằng admin vì thể để viết gọn lại các URL thì chúng ta sẽ sử dụng prefix để URL khi định nghĩa route ngắn gọn dễ nhìn hơn.

Route::group(['prefix' => 'admin'], function () {

Route::get('posts', 'PostController@index');

Route::get('posts/create', 'PostController@create');

Route::post('posts/store', 'PostController@store');

Route::get('posts/{id}/edit', 'PostController@edit');

Route::patch('posts/{id}', 'PostController@update);

Route::patch('posts/{id}', 'PostController@destroy');

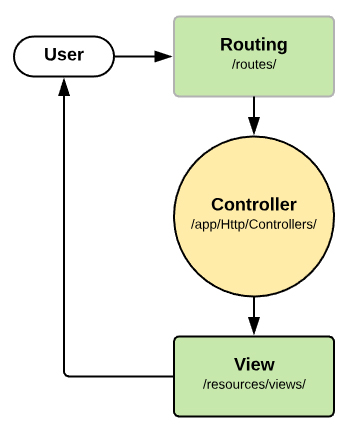
});

* 1. **Controller**



Hình 8 - Tổng quan Controller

* 1. **Cấu trúc controller**
* Để xử lý logic (tính toán) thì chúng ta có thể viết trong nhiều chỗ khác nhau, nhưng tốt nhất là gom chung viết bên trong Controller, đặc biệt là các tính toán dài, phức tạp.
* Controller sinh ra là để trở thành trung tâm xử lý logic, nên cần phải tận dụng, thay vì viết tính toán ở nơi khác.
* Controller được đặt bên trong thư mục */app/Http/Controllers/*.
* Để hiểu một cách cơ bản nhất cách hoạt động của một Controller ta xem cấu trúc đơn giản sau:



Hình 9 - Cấu trúc Controller

|  |  |
| --- | --- |
| Folder / file | Mô tả |
| User | User gửi yêu cầu. |
| Routing | Yêu cầu gửi từ User sẽ được Routing điều hướng:   * Tới Controller để xử lý yêu cầu. * Tới thẳng View nếu không cần xử lý. |
| Controller | Controller được xem như trung tâm điều khiển của hệ thống, tất cả thao tác xử lý nên được thực hiện ở đây.   * Kết quả xử lý sẽ được trả về view. |
| View | View nhận dữ liệu xử lý từ Controller (hoặc Routing), hiển thị kết quả cho người dùng. |

* 1. **Tạo controller:**
* Để tạo controller trong Laravel cũng phải có một số ràn buộc như sau:
* Controller phải ở trong đường dẫn App\http\controllers.
* Tên của controller phải giống với tên class trong file controller đó.
* Class controller (do bạn tạo) phải extends (kế thừa) từ controller (chú ý chữ C phải viết hoa).
* **Cách tạo file**: Ở trong Laravel để tạo controller thì chúng ta có hai cách:
* **Cách 1:**Tạo bằng tay.
* Chúng ta có thể tạo một file controller bằng tay hoặc bằng cmd thuần trong đường dẫn App\Http\Controllers. Và để sử dụng được controller đó thì bắt buộc các bạn phải tuân thủ 3 điều kiện phía trên nhé.
* Ví dụ: mình tạo một Homecontroller.php trong App\Http\Controllers thì trong file sẽ có nội dung như sau:

|  |  |
| --- | --- |
|  | <?php  namespace App\Http\Controllers;  use Illuminate\Http\Request;    class MyController extends Controller{      //code php  } |

* Cách 2:
* Tạo bằng lệnh laravel cung cấp sẵn.
* Để tạo một controller bằng lệnh trong Laravel thì mở commanline lên và trỏ vào thư mục chứa project Laravel và gõ lệnh theo cú pháp.

php artisan make:controller Controllername

Hoặc

php artisan make:controller ControllerName --resource

Để tạo resource controllers

* Chú thích: controllername là tên controller mà mình muốn tạo.
* Khi đó Laravel sẽ tự sinh cho chúng ta 1 file **ControllerName**và có sẵn code như sau:

<?php

namespace App\Http\Controllers;

use App\User;

use App\Http\Controllers\Controller;

class ControllerName extends Controller

{

//

}

* Và với resource controllers:

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Http\Requests;

class ControllerName extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

//

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

//

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

//

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function edit($id)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

//

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function destroy($id)

{

//

}

}

* 1. **Gọi controller**
* Để gọi Controller trong Route chúng ta viết trong file Route (**route/web.php**) theo cấu trúc như sau:

Route::get('URI','controller@action');

* **URI**: Liên kết khi gõ trên trình duyệt
* **Controller**: Tên Controller chúng ta đã tạo
* **Action**: Các phương thức trong controller

Ví dụ: Chúng ta có controller home với nội dung như sau:

**<?php**

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller

{

public function index(){

echo "Đây là trang Home";

}

}

Để gọi tên Controller home chúng ta vào method index chúng ta viết trong Route như sau:

Route::get('/home','HomeController@index');

Như vậy khi chúng ta gõ trên trình duyệt: **http://example.com/home** thì trên màn hình sẽ hiển thị:

“*Đây là trang Home”*

* 1. **Nhận dữ liệu từ Route**
* Truyền tham số id từ Route Nhận dữ liệu bên Controller

|  |  |
| --- | --- |
| **Truyền tham số id từ Route** | **Nhận dữ liệu bên Controller** |
| Route::get('goiController/{id}', ‘MyController@GetData’); | class MyController extends Controller  {  public function GetData($id)  {  echo “Đã nhận ”. $id;  }  } |

## **View**

* View là một thành phần trong mô hình MVC, thì V chính là view. Thực chất, view là các file đuôi . php chứa mã nguồn html, nhiệm vụ của nó là lấy dữ liệu từ controller in ra html hiển thị cho người dùng xem. View được lưu tại resource/views.

Route

Controller (function)

View (<html>)

Computer

Hình 10 - Tổng quan về View

### Tạo view trong Laravel

* Mặc định tất cả các file view sẽ được lưu trong app/views với đuôi file .php hoặc blade.php (sử dụng blade template).
* Nội dung file view có thể chứa các mã nguồn html, css, javascript, php, jquery, . . .
* Nếu bạn muốn tạo sub-view thì chỉ cần tạo thư mục con và tạo ngay file view trong file đó.

Ví dụ:

<!-- View stored in resources/views/greeting.blade.php -->

<html>

<body>

<h1>Hello, {{ $name }}</h1>

</body>

</html>

### Gọi view trong Laravel

* Cú pháp:

View::make($view, $data = array(), $mergeData = array());

* Trong đó:
* $view: tên của view đã tạo (bắt buộc).
* $data: là mảng dữ liệu truyền vào, thao tác hiển thị.
* $mergeData: là mảng dữ liệu truyền vào sau khi đã merge với $data bằng hàm Array().
* Cấu hình Route:

Đường dẫn file route: resources/routes.php

Ví dụ: Đường dẫn: resources/views/greeting.blade.php

// Path: resource/routes.php

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

return view('greeting', ['name' => 'James']); // cú pháp gọi view

});

*Chú ý: Nếu một trang web phức tạp, ta có thể viết hàm gọi view ở controllers và trong routes ta gọi ra.*

Ví dụ: // path: resource/routes.php

Route::controller('/views','ViewController');

* Ta tạo file ViewController.php trong thư mục resources/controllers

// chứa các hàm gọi view

<?php

    class ViewController extends Controller

    {

        public function getReturn()

        {

            return View::make('return');

        }

        public function getVariable()

        {

            $var= View::make('return');

            return $var;

        }

    }

* Ta tạo file view return.blade.php ở thư mục app/views

<h1>Garona Store</h1>

Kết quả in ra: <http://tentrangweb/views/return>

Garona Store

* Ta cũng có thể sử dụng ký tự “.” để gọi sub view.

Ví dụ: Ta gọi profile trong thư mục admin.

Đường dẫn: resources/views/admin/profile.blade.php

return view('admin.profile', $data);

* Để xác định xem View có tồn tại hay không ta kiểm tra như sau:

Nếu kết quả trả về của phương thức exists là true => view Customer chứa trong thư mục emails tồn tại

use Illuminate\Support\Facades\View;

if (View::exists('emails.customer')) {

// }

**Truyền dữ liệu qua View**

* Có 3 cách truyền biến qua View:
* Cách 1: Truyền biến ngay khi gọi View

// Path: resource/routes.php

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

return view('greeting', ['name' => 'James']); // cú pháp gọi view

});

* Như ví dụ trên, biến ở đây là array có biến là ‘name’ và value là ‘James’.
* Ta tạo view greeting tại đường dẫn: resources/views/greeting.php

<p><strong>Username:</strong> {{$name}}</p>

* Kết quả tại đường dẫn: <http://tentrangweb/>

**“**Username**”:** James

* Cách 2: Sử dụng with
* Cú pháp: View::make($view)->with($key,$value);
* Trong đó:

$view (string): là tên view sẽ được gọi

$key (string hoặc array): nếu là chuỗi thì $key sẽ đóng vai trò là biến dùng để sử dụng trong view nếu là mảng thì mỗi khóa của mảng sẽ là 1 biến trong view

$value (string): nếu $key là chuỗi thì $value chính là giá trị của $key trong view, ngược lại thì $value không có giá trị trong view.

// Path: resource/routes.php - string

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

return view('greeting') -> with(‘name’, ‘James’); // cú pháp gọi view

});

// Path: resource/routes.php - array

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

return view('greeting') -> with([‘name’ => ‘James’]); // cú pháp gọi view

});

* Ta tạo view greeting tại đường dẫn: resources/views/greeting.php

<p><strong>Username:</strong> {{$name}}</p>

* Kết quả tại đường dẫn: <http://tentrangweb/>

“Username**:”** James

* Cách 3: Sử dụng magic method
* Cú pháp: View::make($view)->withName($value);
* Trong đó:

$view (string): là tên view sẽ được gọi.

withName:

with là bắt buộc có và viết thường.

Name ở đây là tên biến sẽ gọi trong view và chữ cái đầu tiên viết in hoa và các chữ cái còn lại viết thường.

$value: giá trị của biến của $name

* Ví dụ:

// Path: resource/routes.php

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

return view('greeting') -> withName(‘James’); // cú pháp gọi view

});

* Ta tạo view greeting tại đường dẫn: resources/views/greeting.php

<p><strong>Username:</strong> {{$name}}</p>

* Kết quả tại đường dẫn: <http://tentrangweb/>

**“**Username**:”** James

* + 1. **Chia sẻ dữ liệu trong View**
* Ta sử dụng phương thức share:
* Cú pháp: View::share('key', 'value');
* Ví dụ:

// Path: resource/routes.php

View::share(‘name’,‘James’);

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

return view('greeting') // cú pháp gọi view

});

* Ta tạo view greeting tại đường dẫn: resources/views/greeting.php

<p><strong>Username:</strong> <?php echo $name ?>

* Kết quả tại đường dẫn: <http://tentrangweb/>

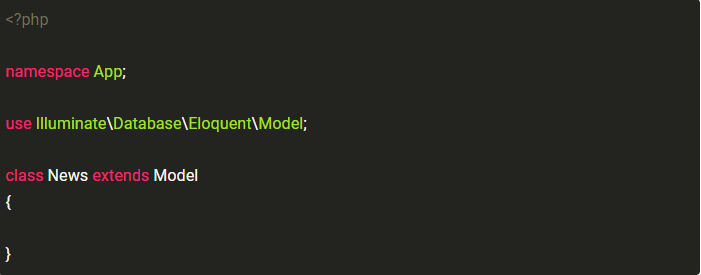
**“**Username**:”** James

## **Model**

### Cách tạo Model

* Có 2 cách tạo Model trong Laravel:
* Tạo thủ công:
* Để tạo một model có hiệu lực thì bắt buộc các bạn phải:
* Đặt nó ở trong thư mục **App/**
* Tên class bên trong file phải cùng với tên file. VD: file news.php thì phải có class là **news.**
* Class vừa tạo phải kế thừa **Model**gốc của Laravel. VD: **class news extends Model{...}**
* Để kế thừa thành công Model trong Laravel thì bắt buộc bạn phải khai báo sử dụng **namespace**Của Model gốc của Laravel và đồng thời khai báo **namespace** mới cho model vừa tạo.

VD: Tôi tạo một model **News.php** trong App/ của Laravel.



* Tạo bằng dòng lệnh có sẵn
* Tạo model bằng cách này thì khá là đơn giản. Các bạn chỉ cần mở cmd(commandline) lên và trỏ đến thư mục chứa project Laravel của các bạn và gõ lệnh.

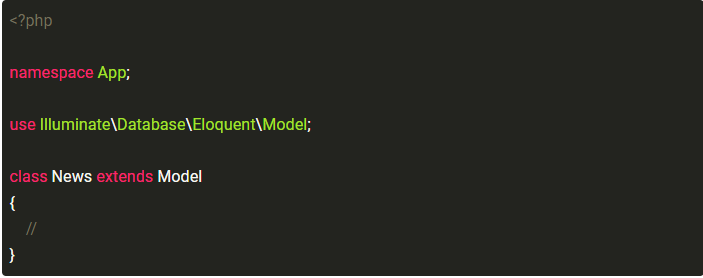


Hoặc



Trong đó:

* News là tên model, các bạn có thể chỉnh thành tên model các bạn muốn tạo.
* Ở dòng thứ hai thì chúng ta cũng tạo được model nhưng đồng thời chúng ta sẽ được tạo luôn một file migration( nói ở bài sau).
* -Sau đó enter và vào trong thư mục **App/** bạn sẽ thấy có một file News.php trong đó và bên trong sẵn có đoạn code.



### Khai báo các thông số tùy chỉnh

* Khai báo table cần sử dụng trong Model.
* Ở trong Laravel mỗi một model ứng với một bảng(table) dữ liệu trong CSDL và để khai báo model sử dụng bảng dữ liệu nào trong database thì mọi người khai báo dòng sau trong class model.



* Trong đó : tableName là tên table các bạn muốn sử dụng.
* Lọc cột dữ liệu trong model.
* Trong laravel chẳng hạn như bảng user tôi chỉ muốn truy vấn cột username,useremail mà không muốn sử dụng cột passwork thì sau. Chính vì điều đó Laravel cũng đã cung cấp cho chúng ta một thông số là **fillable** để điều chỉnh các cột cần sử dụng.



* Khai báo timestamps.
* Laravel cũng cung cấp cho chúng ta tùy biến có sử dụng time stamps hay không. Nếu để **true** là có và ngược lại **false** là không.



* 1. **Database**
     1. **Một vài câu lệnh kết nối Database**
* Laravel làm cho việc kết nối với database và chạy các truy vẫn cực kỳ đơn giản. Tệp cấu hình database là app/config/database.php. Trong tệp này có thể xác định tất cả database connections, cũng như chỉ định kết nối nào sẽ được default.
* Read / Write Connections
* Cấu hình Read / write, ví dụ:



* Sử dụng nhiều Database Connections
* Khi sử dụng nhiều kết nối, bạn có thể truy cập từng kết nối thông qua phương thức connection trên mặt tiền DB. Truyền name cho phương thức connection phải tương ứng với một trong các kết nối được liệt kê trong tệp cấu hình: config/database.php



* Kết nối với database
* Có thể truy cập vào thể hiện PHP Data Objects (PDO) cơ bản bằng cách sử dụng phương thức getPdo trên một kết nối:



* Kết nối lại với database



* Ngắt kết nối khỏi database di vượt quá max\_connections giới hạn Pdo



* + 1. **Các truy vấn (Select, Insert, Update, Delete, Statement, Listen)**
* Select sẽ luôn trả về kết quả là array.



* Insert



* Update



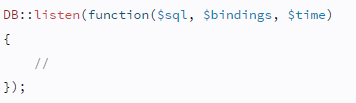
* Delete



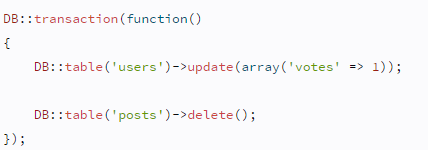
* Statement



* Listen có thể lắng nghe các sự kiện truy vấn



* + 1. **Database Transactions**
* Để chạy một tập hợp các hoạt động trong một database transaction, sử dụng phương thức transaction.



* Bắt đầu một transaction



* Phục hồi transaction



* Cam kết transaction



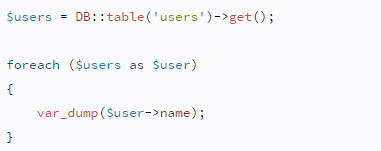
* + 1. **Nhật ký truy vấn**
* Theo mặc định, Laravel giữ một bản ghi trong bộ nhớ của tất cả các truy vấn đã được chạy cho yêu cầu hiện tại. Tuy nhiên một số trường hợp, chẳng hạn như khi chèn một số lượng lớn hàng, điều này có thể khiến ứng dụng sử dụng bộ nhớ dư thừa.
* Vô hiệu hóa nhật ký



* Để có được một mảng các truy vấn được thực hiện

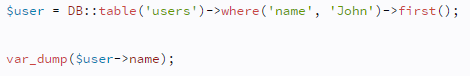


* + 1. **Trình tạo truy vấn**
* Select
* Lấy tất cả các hàng từ một bảng



*//Select \* from users;*

* Lấy một hàng đơn từ một bảng



*//Select first(\*) from users where name = ‘John’;*

* Lấy một cột từ một hàng

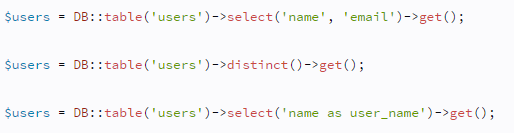


*//Select name from users where name = ‘John’;*

* Lấy một danh sách các giá trị cột



* Chỉ định khoản

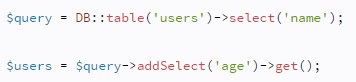


*//Select name, email from users;*

*//Select distinct \* from users;*

*//Select name as user\_name form users;*

* Thêm một khoản vào một truy vấn hiện có



*//Select name from users;*

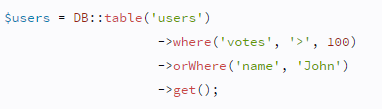
*//Select name, age from users;*

* Operators



*//Select \* from users where votes > 100;*

* Statements

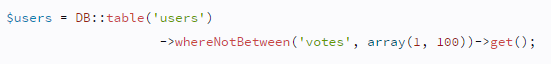


*//Select \* from users where votes > 100 or name = ‘John’;*

* Between

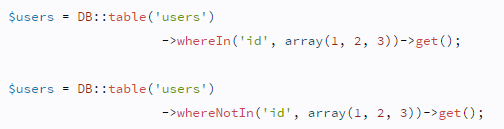


*//Select \* from users where votes between 1 and 100;*



*//Select \* from users where votes not between 1 and 100;*

* Where in



*//Select \* from users where id in (1,2,3);*

*//Select \* from users where id not in (1,2,3);*

* Where Null



*//Select \* from users where updated\_at is NULL*

* Order By, Group By, Having



*//Select count(\*) from users order by name desc group by count having count(\*) > 100;*

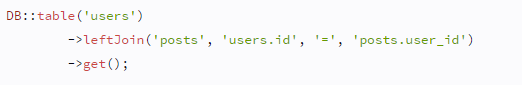
* Offset & Limit



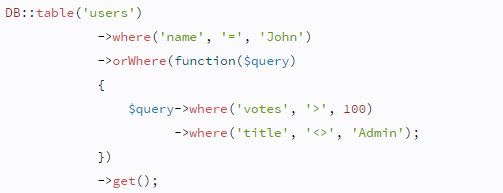
* Joins
* Lệnh nối cơ bản



* Lệnh nối trái

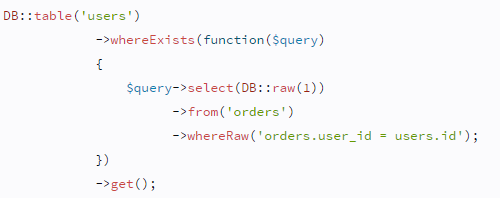


* Where
* Phân nhóm



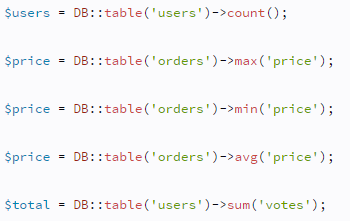
*// Select \* from users where name = 'John' or (votes > 100 and title <> 'Admin');*

* Exists Statements

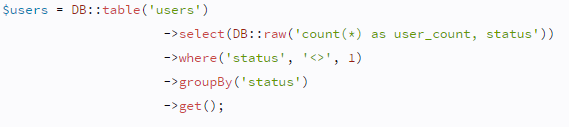


*// Select \* from users where exists (select 1 from orders where orders.user\_id = users.id);*

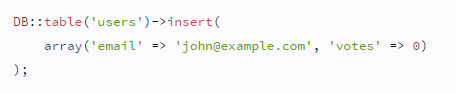
* Aggregates



* Raw



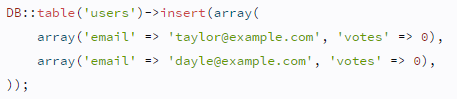
* Insert
* Chèn vào bảng



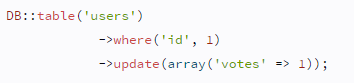
* Chèn vào bảng có ID auto



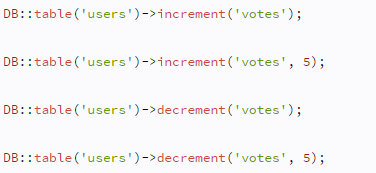
* Chèn nhiều vào bảng



* Update
* Cập nhật trong bảng



Tăng hoặc giảm giá trị của một cột



* Delete
* Xóa trong bảng



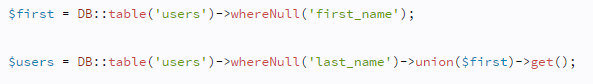
Xóa tất cả từ một bảng



* Cắt bớt bảng



* Union



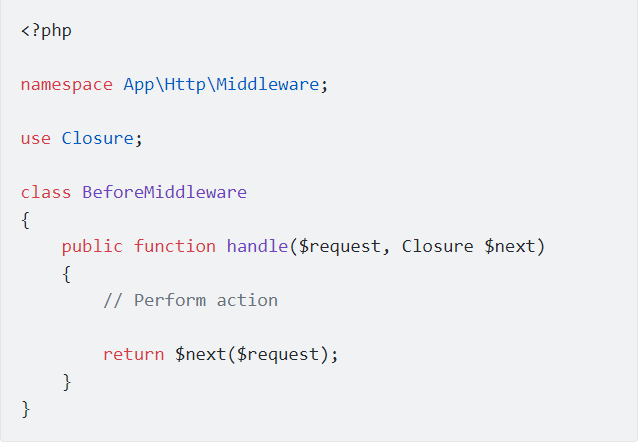
* 1. **Middleware trong Laravel**
     1. **Giới thiệu**
* Middleware cung cấp một cơ chế thuận tiện để lọc các HTTP Request vào ứng dụng của bạn. Ví dụ: Laravel bao gồm một Middleware xác minh người dùng truy cập vào ứng dụng của bạn. Nếu người dùng không được xác thực, Middleware sẽ chuyển hướng người dùng đến màn hình đăng nhập. Ngược lại, nếu người dùng được xác thực, Middleware sẽ cho phép Request tiếp tục vào ứng dụng.
* Middleware có thể được viết để thực hiện thêm các nhiệm vụ khác bên cạnh việc xác thực. Middleware CORS chịu trách nhiệm thêm các tiêu đề thích hợp cho tất cả các response rời khỏi ứng dụng của bạn. Middleware log có thể thực hiện ghi log cho tất cả các request tới ứng dụng.
* Vài middleware đã có sẵn trong Laravel framework, bao gồm middleware cho bảo trì, xác thực, phòng chống CSRF và còn nữa. Tất cả những middleware này nằm trong thư mục app/Http/Middlware.
  + 1. **Tạo Middleware**
* Để tạo một middleware mới, các bạn hãy sử dụng câu lệnh Artisan: make:middleware php artisan make:middleware CheckAge
* Lệnh này sẽ tạo một class CheckAge trong thư mục app/Http/Middleware. . Trong middleware này, chúng ta sẽ chỉ chp phép truy cập vào route nếu giá trị age lớn hơn 200. Nếu không, chúng ta sẽ chuyển hướng người dùng quay lại 'home' URL.



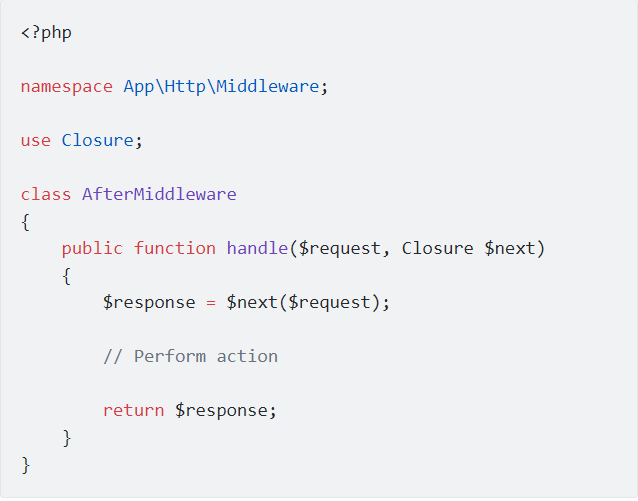
Như các bạn đã thấy, nếu giá trị age đã cho nhỏ hơn hoặc bằng 200, middleware sẽ trả lại một HTTP chuyển hướng tới client; ngược lại, request sẽ được gửi tiếp để xử lý. Để truyền request vào sâu hơn trong ứng dụng (cho phép middleware "vượt qua"), đơn giản chỉ cần gọi callback $next với $request. Tốt nhất là hãy hình dung middleware là một chuỗi các "layers" trên HTTP request cần phải đi qua trước khi đi vào trong chương trình. Mỗi layer sẽ thực hiện kiểm tra request và thậm chí có thể huỷ từ chối request hoàn toàn.

* + 1. **Before và After Middle**

Việc một middleware chạy trước hay sau một request phụ thuộc vào chính middleware nó. Ví dụ, middleware dưới đây sẽ thực hiện vài tác vụ **trước khi** request được chương trình xử lý:



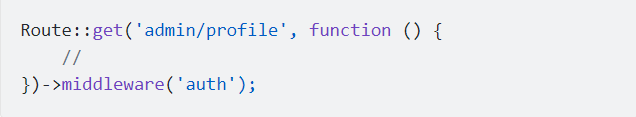
Tuy nhiên, middleware này sẽ thực hiện nhiệm vụ của mình sau khi request được ứng dụng xử lý:



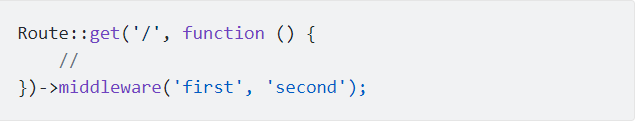
* + 1. **Đăng kí Middleware**
* Global Middleware:
* Nếu bạn muốn một middleware chạy trong mỗi HTTP Request đến ứng dụng của bạn, đơn giản chỉ cần thêm tên class của middleware đó vào trong thuộc tính $middleware của class app/Http/Kernel.php.
* Thiết lập middleware cho route
* Nếu bạn muốn gán middleware cho các route cụ thể, trước tiên bạn cần phải thêm middleware vào trong biến $routeMiddleware trong file app/Http/Kernel.php và đặt cho nó một key:

|  |
| --- |
| // Within App\Http\Kernel Class...  protected $routeMiddleware = [  'auth' => \App\Http\Middleware\Authenticate::class,  'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,  'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,  'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,  'can' => \Illuminate\Auth\Middleware\Authorize::class,  'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,  'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,  'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,  'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,  ]; |

Sau khi đã được khai báo trong HTTP Kernel, bạn có thể sử dụng khoá middleware để thiết lập thông số cài vào trong route:



Bạn cũng có thể chỉ định nhiều middleware cho route



Khi gán middleware, bạn cũng có thể sử dụng tên class đầy đủ của middleware muốn :



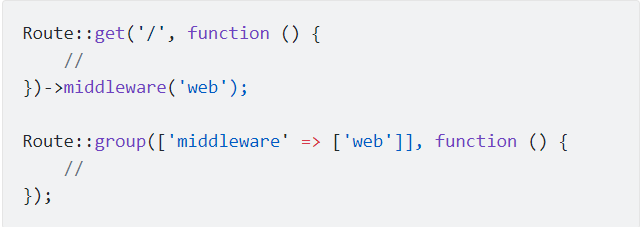
* + 1. **Tạo nhóm middleware**

Đôi khi bạn muốn nhóm một số middleware bằng một khóa duy nhất để có thể gán vào route dễ dàng hơn. Bạn có thể làm điều này bằng cách sử dụng $middlewareGroups của HTTP kernel.

Về cơ bản, Laravel cung cấp sẵn hai nhóm middleware thường sử dụng mà bạn có thể muốn áp dụng cho web UI hay API:

|  |
| --- |
| /\*\*  \* The application's route middleware groups.  \*  \* @var array  \*/  protected $middlewareGroups = [  'web' => [  \App\Http\Middleware\EncryptCookies::class,  \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,  \Illuminate\Session\Middleware\StartSession::class,  \Illuminate\View\Middleware\ShareErrorsFromSession::class,  \App\Http\Middleware\VerifyCsrfToken::class,  \Illuminate\Routing\Middleware\SubstituteBindings::class,  ],  'api' => [  'throttle:60,1',  'auth:api',  ],  ]; |

Các nhóm middleware có thể gán cho các route và controller action sử dụng cú pháp tương tự như với từng middleware riêng. Việc sử dụng nhóm middleware sẽ làm cho việc gán các middleware vào trong một route trở nên tiện hơn:



\*Tip: nhóm middleware web được tự động áp dụng vào trong file routes.php qua RouteServiceProvider

* + 1. **Sắp xếp Middleware**

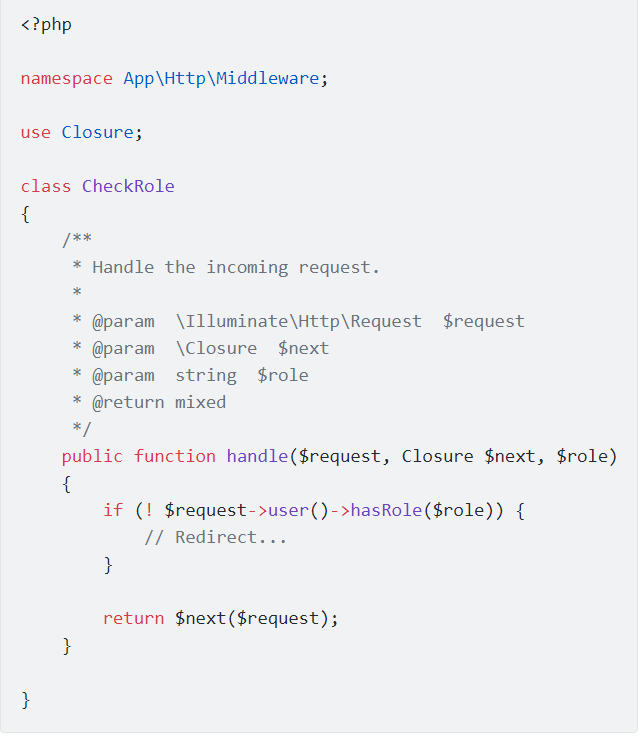
Đôi khi, bạn cần middleware của mình để thực thi theo một thứ tự cụ thể nhưng không có quyền kiểm soát thứ tự của chúng khi chúng được gán cho route. Trong trường hợp này, bạn có thể chỉ định mức độ ưu tiên của middleware bằng cách sử dụng $middlewarePriority ở trong: app/Http/Kernel.php:

|  |
| --- |
| /\*\*  \* The priority-sorted list of middleware.  \*  \* This forces non-global middleware to always be in the given order.  \*  \* @var array  \*/  protected $middlewarePriority = [  \Illuminate\Session\Middleware\StartSession::class,  \Illuminate\View\Middleware\ShareErrorsFromSession::class,  \App\Http\Middleware\Authenticate::class,  \Illuminate\Session\Middleware\AuthenticateSession::class,  \Illuminate\Routing\Middleware\SubstituteBindings::class,  \Illuminate\Auth\Middleware\Authorize::class,  ]; |

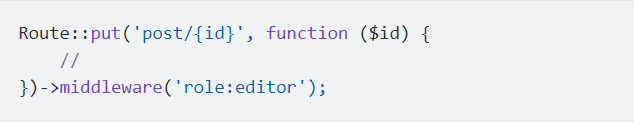
* + 1. **Middleware Parameters**

Middleware cũng có thể nhận các tham số truyền vào. Ví dụ, nếu chương trình cần xác nhận user đã được xác thực có "role" cụ thể trước khi thực hiện một thao tác nào đó, bạn có thể tạo ra CheckRole để nhận tên của role như một tham số.

Các tham số của middleware sẽ được truyền vào thành tham số của hàm handle ngay sau tham số $next:

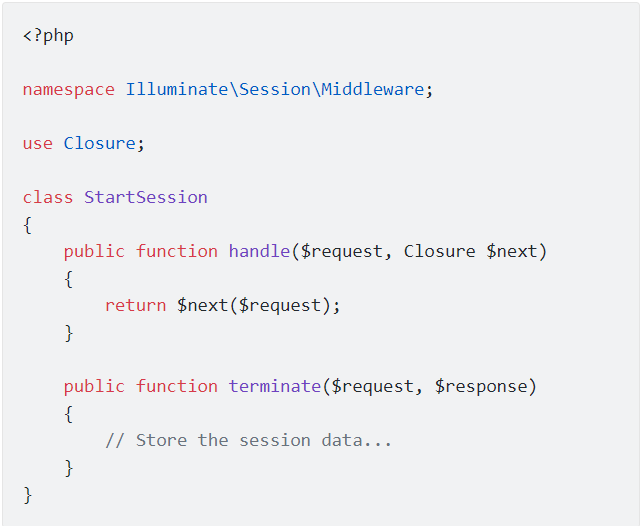


Tham số cho middleware cũng có thể được khai báo trên route bằng cách phân cách tên middleware và tham số bởi dấu :. Nhiều tham số phân cách nhau bởi dấu phẩy , :



* + 1. **Terminable Middleware**

Đôi khi một middleware cần thực hiện chỉ sau khi HTTP response đã được gửi xong cho trình duyệt. Ví dụ, "session" middleware đi kèm với Laravel cung cấp session data cho storage sau khi response được gửi tới trình duyệt. Để làm được việc này, cần phải tạo một middleware kiểu "kết thúc" bằng cách thêm vào hàm terminate vào trong middleware:



Hàm terminate sẽ nhận cả request và response. Khi mà bạn khai báo một terminable middleware, bạn phải thêm nó vào trong danh sách global middleware trong app/Http/Kernel.php.

Khi gọi hàm terminate trong middleware, Laravel sẽ thực hiện resolve một instance mới cho middleware từ service container. Nếu bạn muốn sử dụng cùng một instance khi mà handle và terminate được gọi, đăng kí middleware vào trong container sử dụng hàm singleton.

## **Auth – Authentication**

### Giới thiệu

Laravel giúp cho việc thực hiện việc xác thực vô cùng đơn giản. Trong thực tế, hầu hết mọi thứ đã được cấu hình cho bạn mà bạn đéo thể tưởng tượng nổi (out of the box). Các file cấu hình xác thực được đặt tại config/auth.php, bao gồm một số hướng dẫn tùy biến rõ ràng cho việc tinh chỉnh cách xử lí của các dịch vụ authentication.

Tại phần lõi của nó, các cơ sở của Laravel's authentication được tạo bởi các "guards" và "providers". Guards định nghĩa cái cách mà các user được xác thực cho mỗi request. Ví dụ, Laravel mang theo một session guard cái mà duy trì trạng thái bằng cách sử dụng session storage và cookies và một token guard, cái mà xác thực user bằng cách sử dụng một "API token" cái mà được truyền cùng mỗi request.

Providers định nghĩa cách mà user được truy xuất từ lưu trữ không đổi (persistent storage) của bạn. Laravel hỗ trợ cho việc truy xuất các user sử dụng Eloquent và Query Builder.

### Tạo route và view cho Auth

Laravel cung cấp một cách nhanh chóng để sinh ra toàn bộ các route và view cần thiết cho authentication chỉ với 1 command:

php artisan make:auth

Command này nên được dùng trên các ứng dụng mới và sẽ cài đặt các view đăng kí và đăng nhập cũng như các route cho toàn bộ việc xác thực đầu cuối. Một HomeController cũng sẽ được sinh ra, phục vụ các request post-login tới ứng dụng. Tuy nhiên, bạn có thể tự do tùy chỉnh hoặc xóa controller này dựa trên sự cần thiết trong ứng dụng của bạn.

Command này cũng sẽ tạo ra các view hỗ trợ cho các chức năng login của người dung. Các view được tạo ra trong folder resources/views/auth, và cũng tạo một thư mục resources/views/layouts bao gồm các layout cơ bản cho ứng dụng. Toàn bộ những view này sử dụng framework Bootstrap CSS, nhưng bạn tự do tùy chỉnh nếu bạn thích.

### Một số hàm của class Auth

* **Attempt()**
* Ngoài việc xác thực bằng các controller có sẳn, ta hoàn toàn có thể xác thực người dung thủ công bằng cách sử dụng các class có sẳn trong Auth bằng cách import Illuminate\Support\Facades\Auth

Ví dụ, ta có thể xác thực như sau:

if (Auth::attempt(['email' => $email, 'password' => $password])) {

// xác thực thành công

}

Trong đó, hàm attempt() sẽ nhận mạng dữ liệu dạng key/values. Đầu tiên sẽ xác nhận xem email có tồn tại trong bảng users hay không, nếu có tiếp tục trường password của users sẽ được lấy ra so sánh với hash của password. Nếu đúng trả về true, ngược lại false.

* **Check()**

Hàm check() được sử dụng để xác thực xem người dùng đã đăng nhập vào hệ thống chưa. Để sử dụng check() ta cần import như trên.

Ví dụ

if (Auth::check()) {

echo "Nguoi dung da dang nhap he thong";

}

Ngoài ra, khi người dung đã đăng nhập vào hệ thống ta có thể truy suất dữ liệu của người dùng đã xác thực ở mọi nơi trước khi phiên đăng nhập kết thúc.

Để truy cập thông tin người dung, Auth hỗ trợ một số hàm sau:

$id = Auth::user()->id; hoặc Auth::id() //Lấy về ID người .

$user = Auth::user() // Lấy về tất cả các thông tin của người dùng.

$email = Auth::user()->email // Lây về email người dùng.

* **Logout()**

Hàm logout() trong Auth để đăng xuất người dùng ra khỏi hệ thống, ta có thể truy suất hàm này theo 2 cách : route('logout') và Auth::logout()

* **Middleware**

Trong Laravel , authentication có middleware auth , hệ thống muốn chắc chắn rằng người dùng phải đăng nhập xác thực trước khi được làm một số các thao tác đối với hệ thống. Middleware nằm trong ở app/Http/Middleware/RedirectIfAuthenticated.php. Các bạn có thể sử dụng ngay ở routes hoặc trong hàm \_\_contruct() ở mỗi Controller.

Route::get('profile', function () {

// Only authenticated users may enter...

})->middleware('auth');

public function \_\_construct()

{

$this->middleware('auth');

}