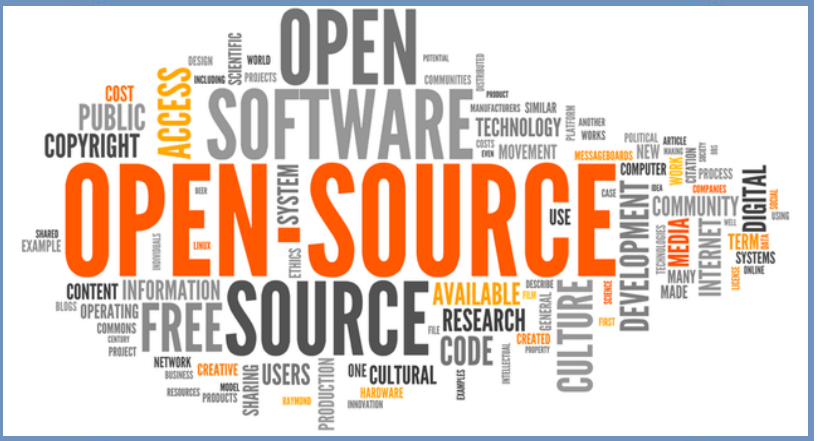
Open Source Software

Van Tham Nguyen

Thuyloi Univerity
175 Tay Son, Dong Da, Ha Noi

LARAVEL



1. INSTALL COMPOSER
2. Create Project From Composer
3. Blade

…..

1 INSTALL COMPOSER

What is Composer ?

Composer is a dependency management tool for PHP. It allows you to manage the libraries and software packages used in your PHP projects. Composer solves the problem of managing dependencies by automatically downloading and installing software packages from online repositories.

With Composer, you can define your project's dependencies and configuration in a file called composer.json. This file describes the software packages that your project needs and specific versions of them. When you run Composer, it looks at the composer.json file and downloads the corresponding software packages from repositories such as Packagist (Composer's default repository) and installs them into your project.

Composer provides a convenient way to manage the dependencies of your PHP project, making it easy to install, update, and remove software packages. It has become a popular tool in the PHP community and is widely used in the development of modern PHP applications.

Download Composer

https://getcomposer.org/download/

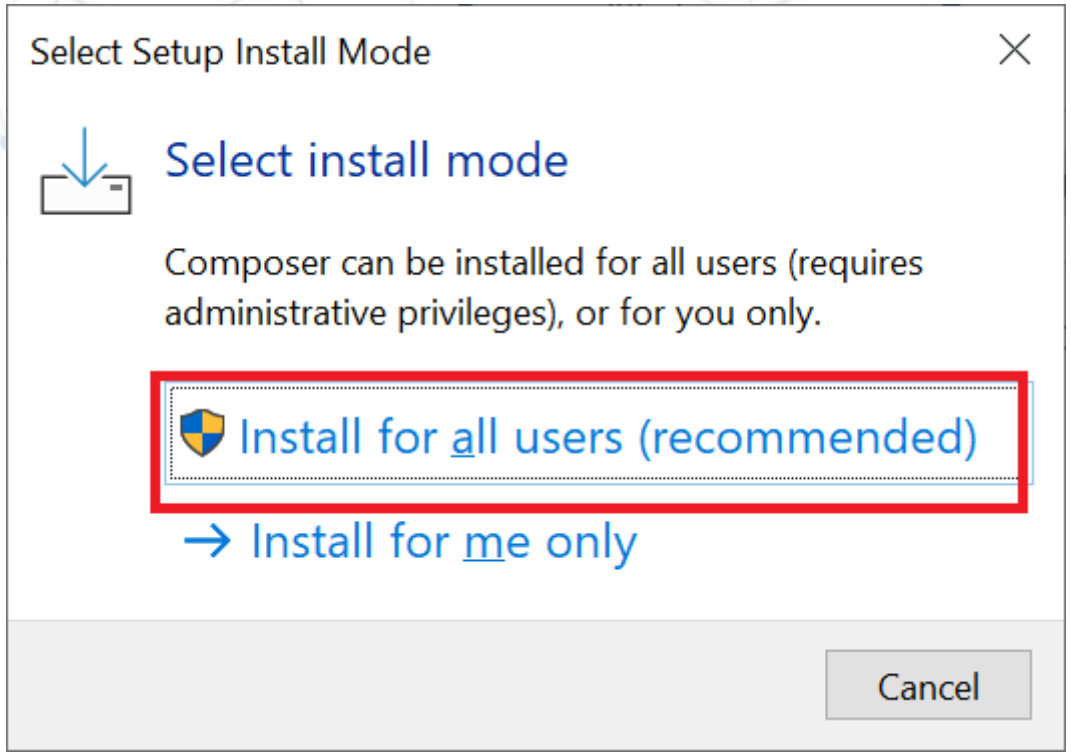
Download Composer Latest: v2.7.6

Windows Installer

The installer - which requires that you have PHP already installed - will download Composer for you and set up your PATH environment variable so you can simply call `composer` from any directory.

Download and run Composer-Setup.exe - it will install the latest composer version whenever it is executed.

Install Composer



Check Composer

Chạy lệnh composer trên Terminal của VC

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

```
PS D:\Code_TeachingSubject\Code_OSSD\OSSD_Example> composer
   ______
  / ____/___  ____ ___  ____  ____  ________  _____
 / /   / __ \/ __ `__ \/ __ \/ __ \/ ___/ _ \/ ___/
/ /___/ /_/ / / / / / / /_/ / /_/ (__  )  __/ /
\____/\____/_/ /_/ /_/ .___/\____/____/\___/_/
                    /_/
Composer version 2.7.6 2024-05-04 23:03:15

Usage:
  command [options] [arguments]

Options:
  -h, --help                     Display help for the given command. When no command is given display help for the list comma
nd
  -q, --quiet                    Do not output any message
  -V, --version                  Display this application version
      --ansi|--no-ansi           Force (or disable --no-ansi) ANSI output
  -n, --no-interaction           Do not ask any interactive question
      --profile                  Display timing and memory usage information
      --no-plugins               Whether to disable plugins.
```

2 Create Project From Composer

Create Laravel Project

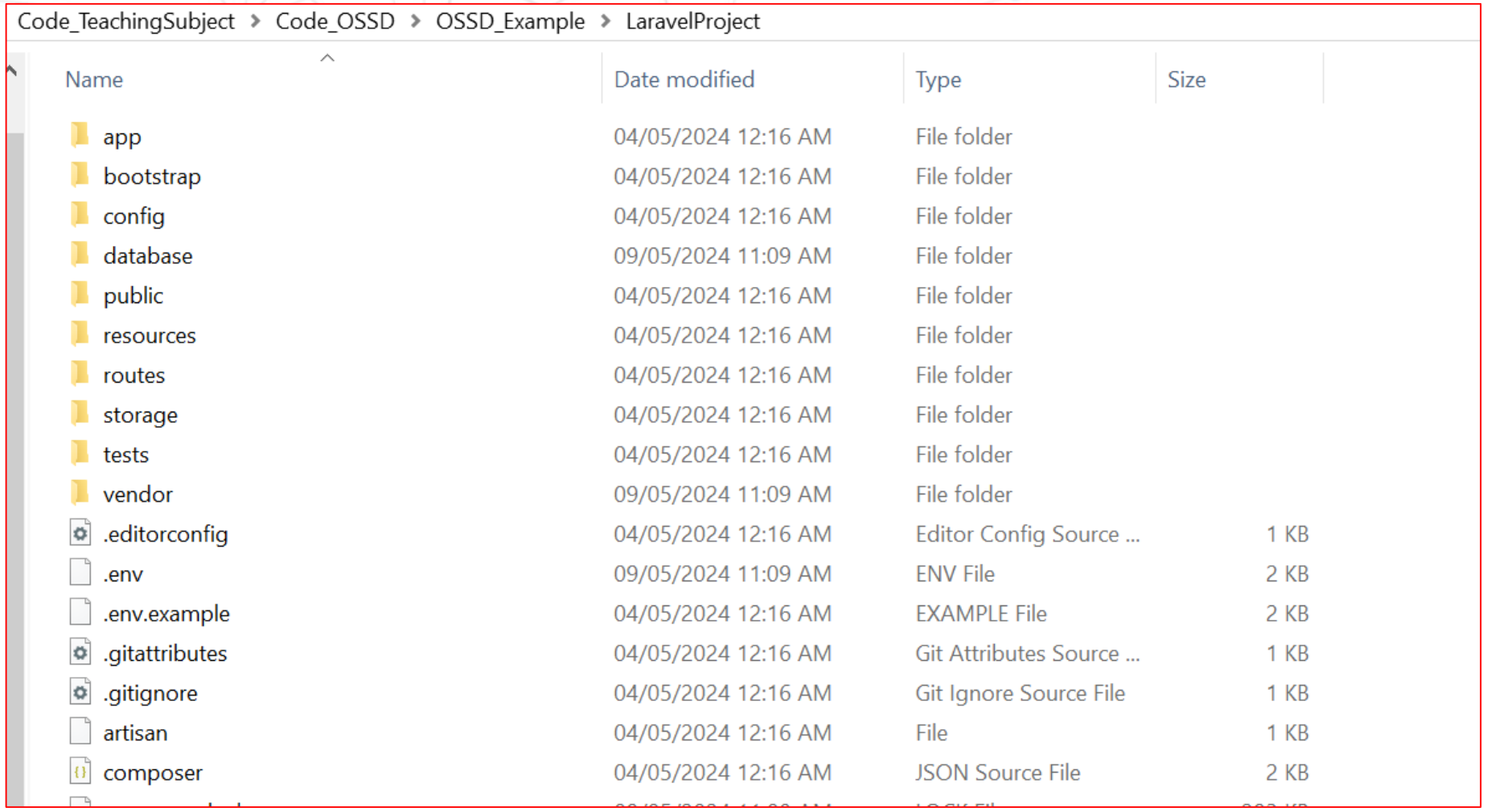
Step 1: Create Laravel Project

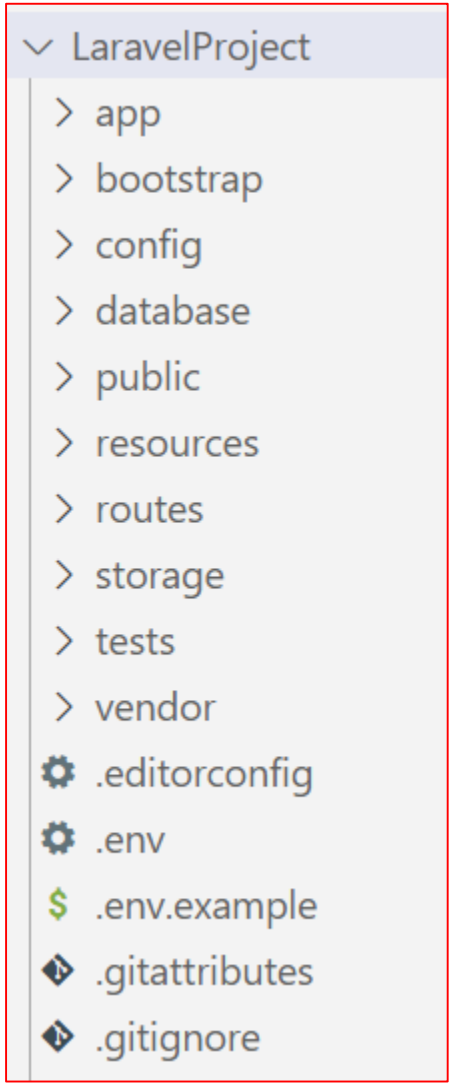
composer create-project laravel/laravel myproject

Step 2: Check Laravel Project

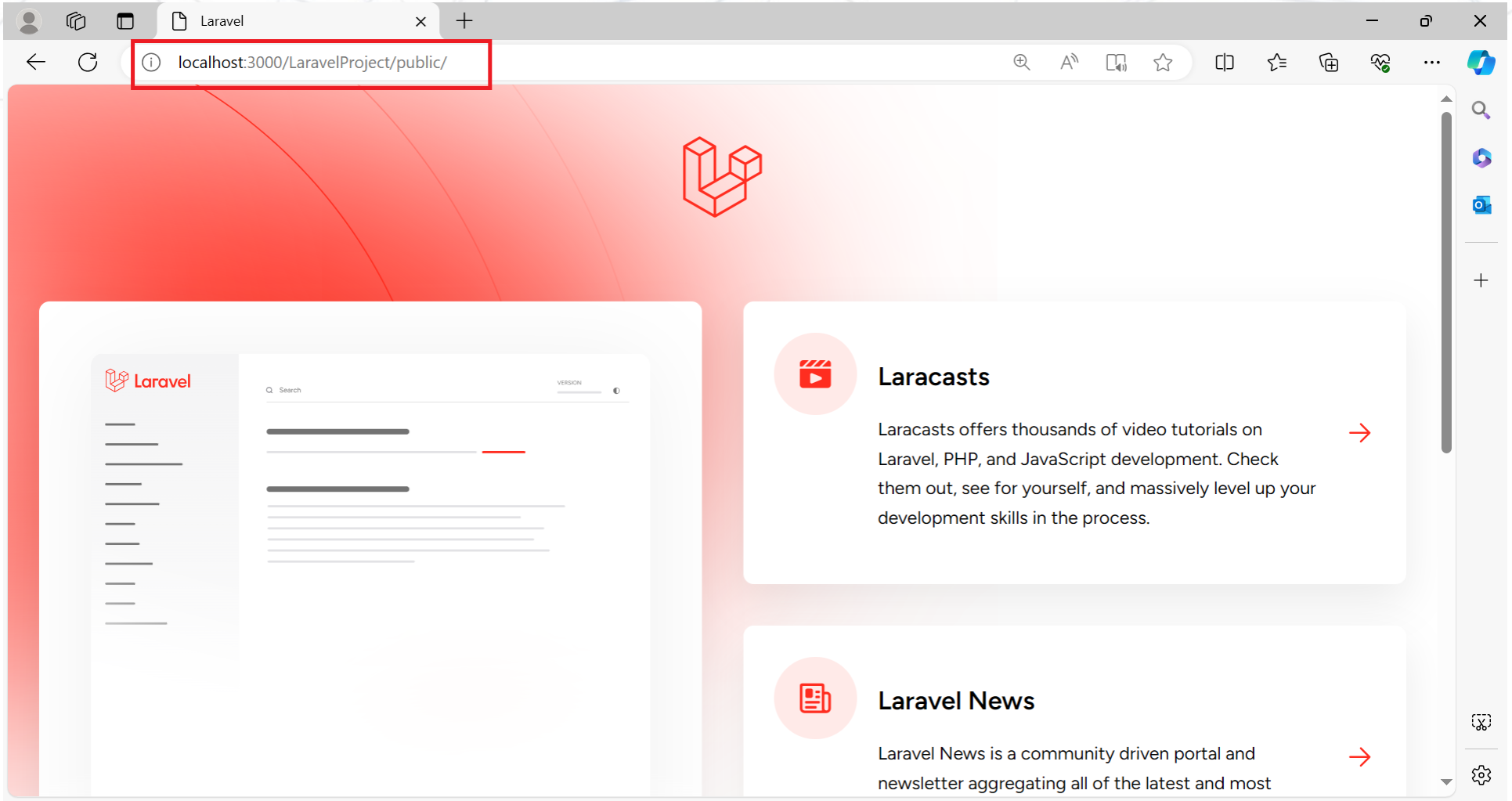
Create Laravel Project

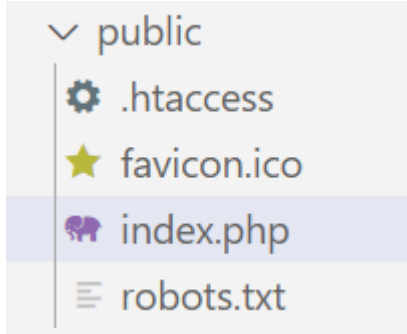
Step 2: Check Laravel Project





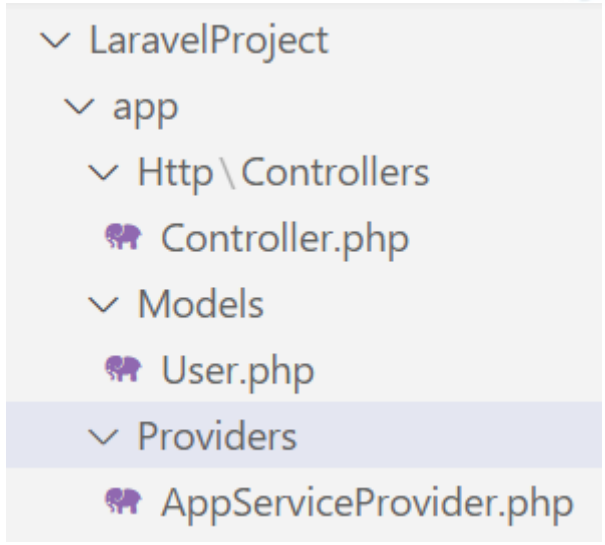
Run Laravel Project





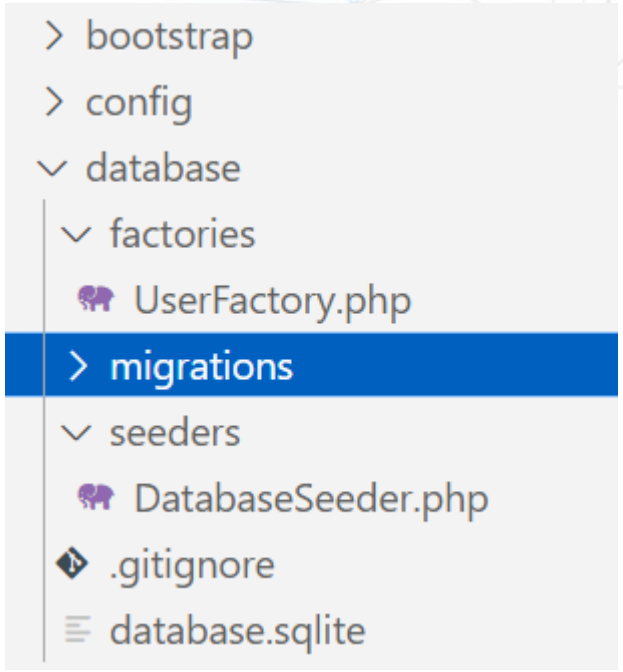
Chạy trang index.php trong thư mục public

Laravel directory structure



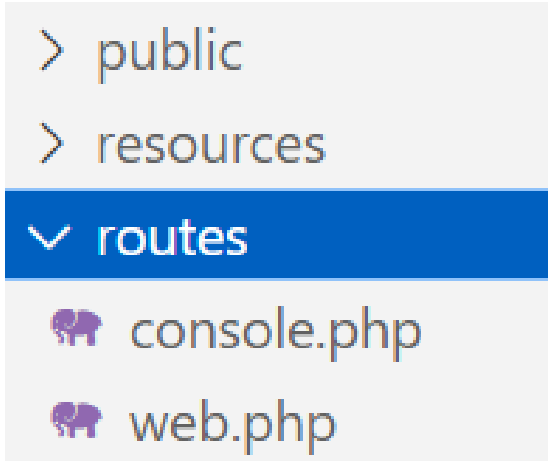
Folder / File	Mô tả
app	Thư mục app, chứa tất cả các project được tạo, hầu hết các class trong project được tạo đều ở trong đây. Không giống các framwork khác, các file model không được chứa trong một thư mục riêng biệt, mà được chứa ngay tại thư mục app này.
app/Http/Controllers	Thư mục Controllers, chứa các controller của project.
app/Providers	Thư mục Providers, chứa các file thực hiện việc khai báo service và bind vào trong Service Container.
app/Models	Thư mục Controllers, chứa các model của project.

Laravel directory structure



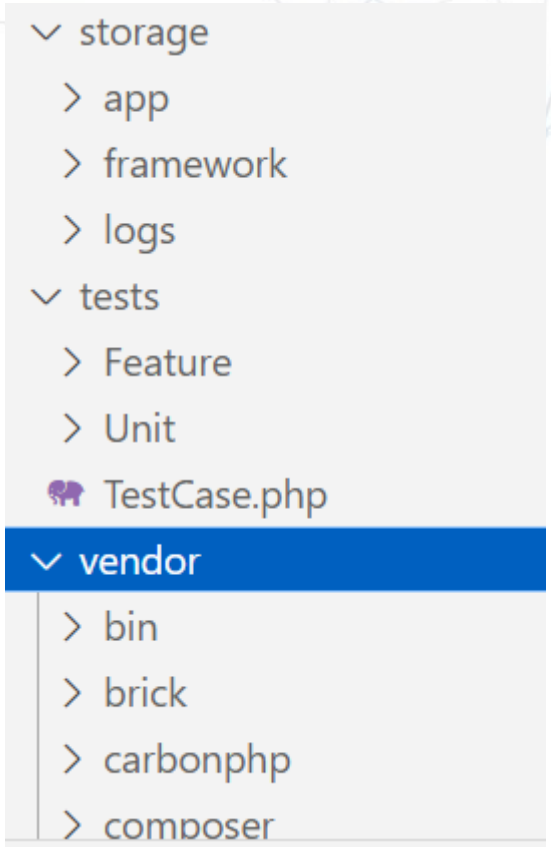
Folder / File	Mô tả
bootstrap	Thư mục bootstrap, chứa những file khởi động của framework và những file cấu hình auto loading, route, và file cache.
config	Thư mục config, chứa tất cả những file cấu hình.
database	Thư mục database, chứa 2 thư mục migration (tạo và thao tác database) và seeds (tạo dữ liệu mẫu), tiện lợi để lưu trữ dữ liệu sau này.
database/factories	Thư mục factories, chứa các file định nghĩa các cột bảng dữ liệu để tạo ra các dữ liệu mẫu.
database/migrations	Thư mục migrations, chứa các file tạo và chỉnh sửa dữ liệu.
database/seeds	Thư mục seeds, chứa các file tạo dữ liệu thêm vào CSDL.

Laravel directory structure



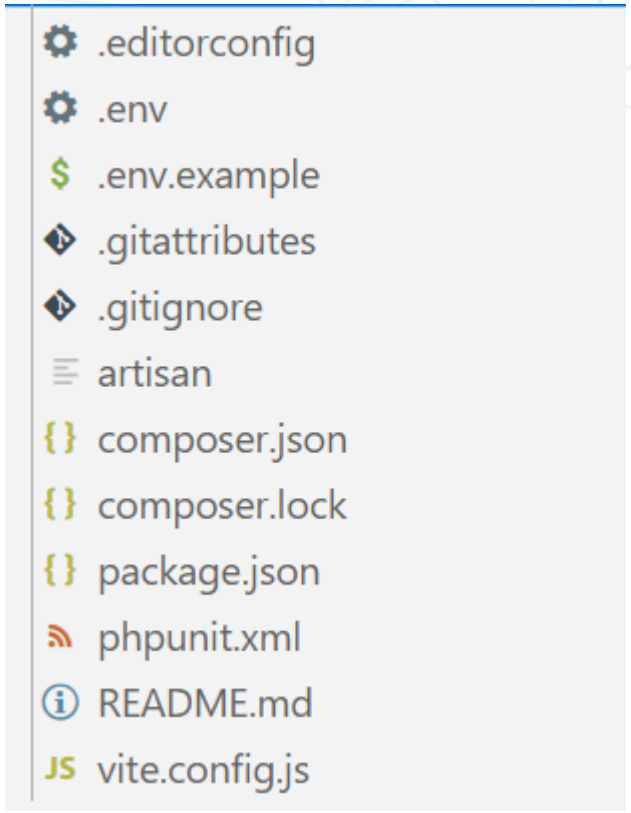
Folder / File	Mô tả
public	Thư mục public, chứa file index.php giống như cổng cho tất cả các request vào project, bên trong thư mục còn chứa file JavaScript, và CSS.
resources	Thư mục resources, chứa những file view và raw, các file biên soạn như LESS, SASS, hoặc JavaScript. Ngoài ra còn chứa tất cả các file lang trong project.
resources/views	Thư mục views, chứa các file view xuất giao diện người dùng.
routes	Thư mục routes, chứa tất cả các điều khiển route (đường dẫn) trong project. Chứa các file route sẵn có: web.php, channels.php, api.php, và console.php.
routes/console.php	file console.php, điều khiển các route của ứng dụng, như route của ứng dụng User (đăng ký, đăng nhập, ...).
routes/web.php	file web.php, điều khiển các route của view, như route của trang top, sản phẩm, ...

Laravel directory structure



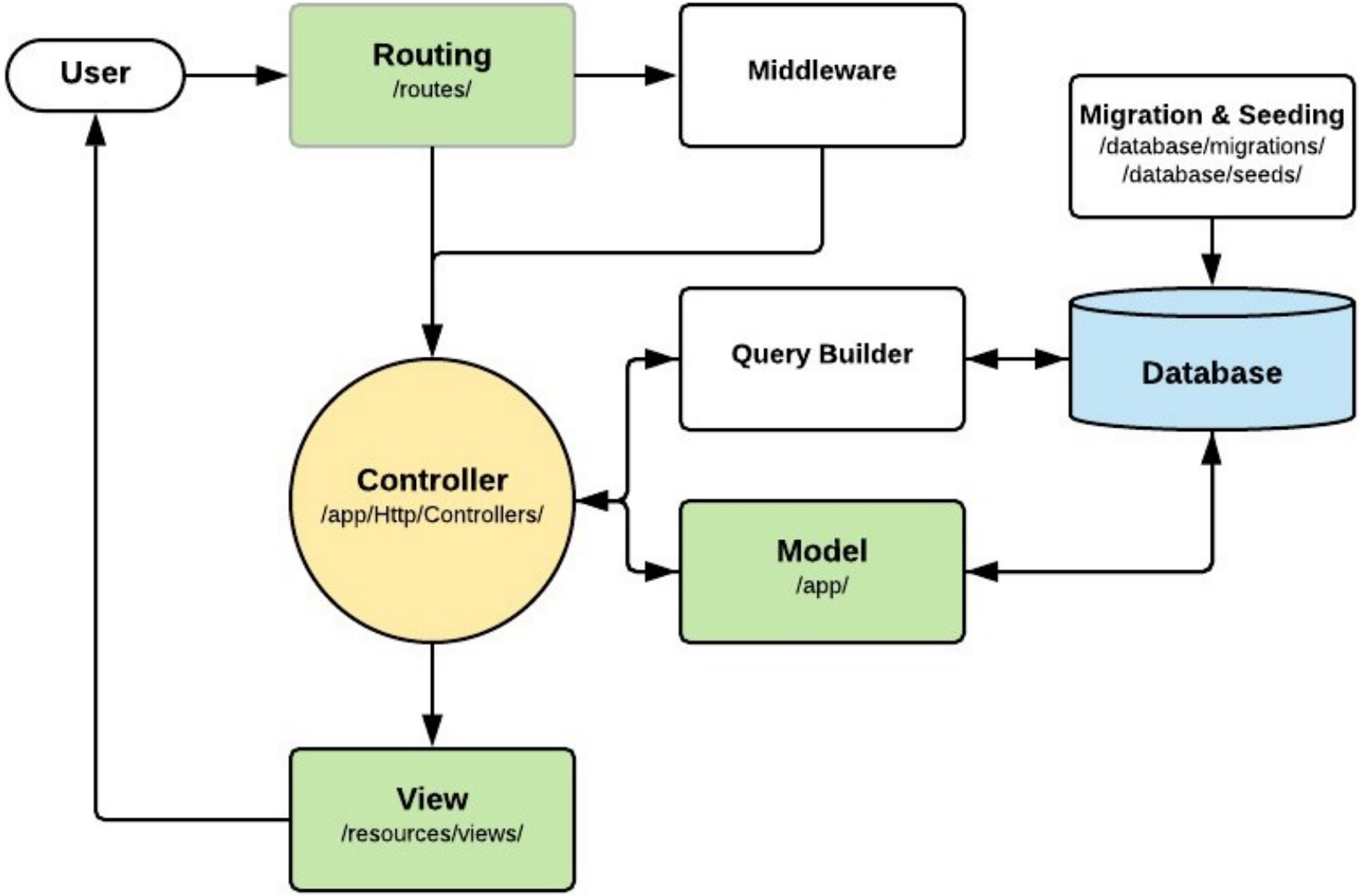
Folder / File	Mô tả
storage	Thư mục storage, chứa các file biên soạn blade templates của bạn, file based sessions, file caches, và những file sinh ra từ project. •Thư mục app, dùng để chứa những file sinh ra từ project. •Thư mục framework, chứa những file sinh ra từ framework và caches. •Thư mục logs, chứa những file logs. •Thư mục /storage/app/public, lưu những file người dùng tạo ra như hình ảnh.
tests	Thư mục tests, chứa những file tests, như PHPUnit test.
vendor	Thư mục vendor, chứa các thư viện của Composer.

Laravel directory structure



Folder / File	Mô tả
.env	file .env, chứa các config chính của Laravel.
artisan	file thực hiện lệnh của Laravel.
.gitattributes .gitignore	File dành cho xử lý git.
composer.json composer.lock composer-setup.php	File của Composer.
package.js	file package.js, chứa các package cần dùng cho projects.
phpunit.xml	file phpunit.xml, xml của phpunit dùng để testing project.
webpack.mix.js	file webpack.mix.js, file dùng để build các webpack.

Laravel's processing flow



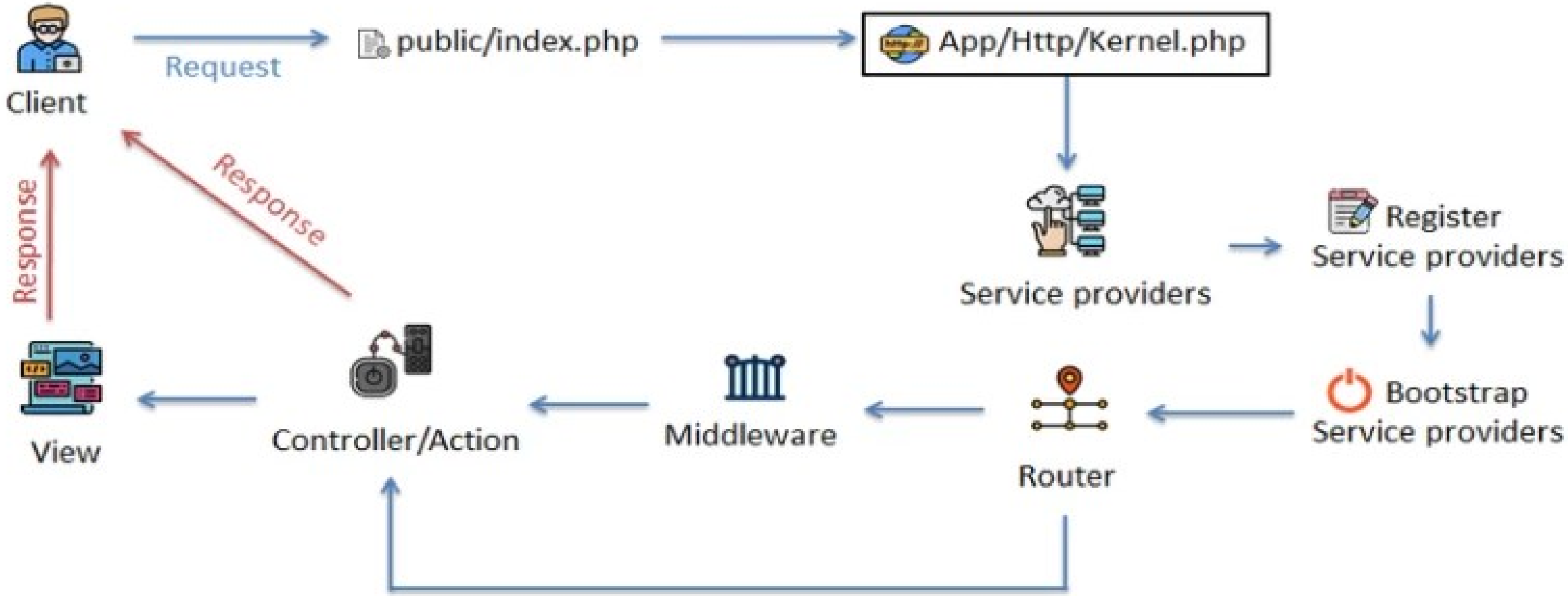
Laravel's processing flow

Folder / File	Mô tả
User	User gửi yêu cầu.
Routing	Yêu cầu từ User sẽ được Routing điều hướng: •Tới Controller để xử lý yêu cầu. •Tới Middleware để kiểm tra điều hướng cần thiết. •Tới thẳng View nếu không cần xử lý.
Middleware	Middleware có thể xem như bộ lọc HTTP từ request, VD như dùng để kiểm tra, xác thực người dùng đăng nhập vào hệ thống. Sau khi kiểm tra, xác thực xong sẽ trả về Controller để tiếp tục 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. •Controller kết nối với Database thông qua điều khiển từ Model. Hoặc có thể thao tác trực tiếp tới Database thông qua các Query Builder •Kết quả xử lý sẽ được trả về view.

Laravel's processing flow

Folder / File	Mô tả
Model	Khi có yêu cầu từ Controller, Model sẽ tương tác với Database và trả kết quả về Controller, một số trường hợp cần thiết thì Model cũng có thể trả thẳng kết quả về View.
Database	Lưu trữ và chứa dữ liệu.
Migration & Seeding	•Migration, được dùng để tạo Database. •Seeding, được dùng để tạo dữ liệu ảo cho Database.
Query Builder	Câu truy vấn Database.
View	View nhận dữ liệu xử lý từ Controller (hoặc Model, Routing), hiển thị kết quả cho người dùng.

Request LifeCycle Laravel



3 Blade

What is Blade?

Blade là dạng cú pháp đơn giản được cung cấp bởi Laravel.

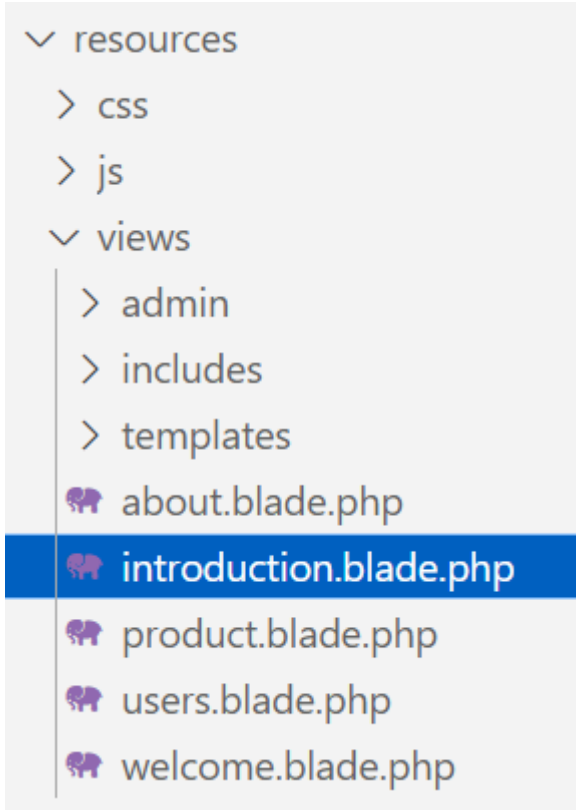
Sử dụng Blade:

- Để tạo template header, footer, sidebar hay bất kỳ thành phần nào, include vào Views.
- Sử dụng những câu lệnh cần thiết điều khiển các thành phần trong Views như: If else, for, foreach, ...
- Còn nhiều tính năng khác, bạn có thể tham khảo thêm tại trang chính của Laravel: Blade

Cơ bản về Blade

Blade được thể hiện dưới dạng file có tên: name.blade.php

Blade được chứa bên trong thư mục /resources/views/



Cấu trúc tương đồng giữa PHP và Blade

PHP	Blade
`<?php ?>`	`{{ }}`
`<?php echo $name ?>`	`{{ $name }}`
`<?php include 'footer.php' ?>`	`{{ @include('includes.footer') }}`

`if(điều_kiện):`
`  câu_lệnh`
`elseif(điều_kiện):`
`  câu_lệnh;`
`else:`
`  câu_lệnh;`
`endif;`
	`@if(điều_kiện)`
	`  câu_lệnh;`
	`@elseif(điều_kiện)`
	`  câu_lệnh;`
	`@else`
	`  câu_lệnh;`
	`@endif;`

`for ($i = 0; $i < 10; $i++) {`
`  câu_lệnh;`
`}`
	`@for ($i = 0; $i < 10; $i++)`
	`  câu_lệnh;`
	`@endfor`

Cấu trúc tương đồng giữa PHP và Blade

PHP	Blade

while (true) {
    câu_lệnh;
}
	@while (true)
	    câu_lệnh;
	@endwhile

foreach ($users as $user) {
    câu_lệnh;
}
	@foreach ($users as $user)
	    câu_lệnh;
	@endforeach

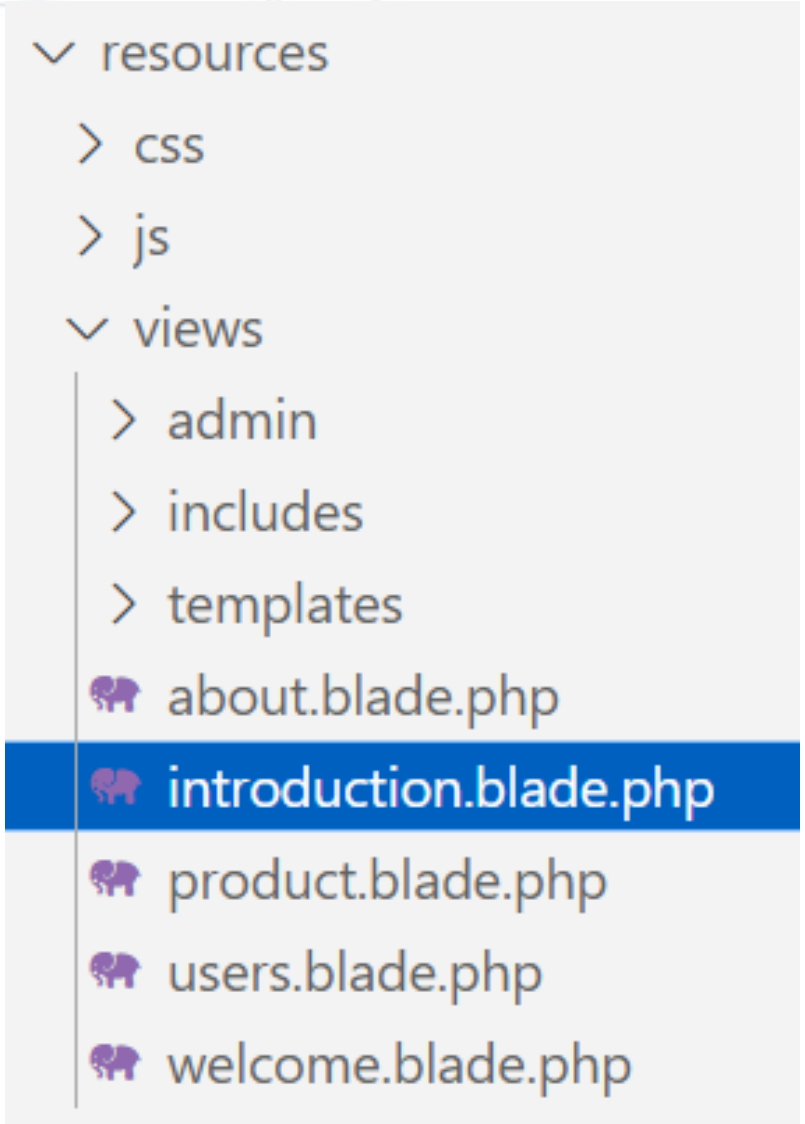
Cần câu lệnh if và foreach	@forelse ($users as $user)
	    {{ $user->name }}
	@empty
	    Không có user nào!
	@endforelse

Cấu trúc folder mẫu của Blade

/resources/views/ là thư mục làm việc chính của Blade dành cho việc layout (không bao gồm chứa các files: css, js, image, ...).

Tổ chức folder bên trong /resources/views/ sao cho phù hợp với yêu cầu project cần,

Ví dụ có thể tạo cấu trúc thư mục như sau:



Cấu trúc folder mẫu của Blade

Folder	Mô tả
admin	Chứa giao diện của trang Admin
includes	Chứa giao diện của những phần include như: header, footer, aside, banner, ..., tùy vào mình mà có thể sắp xếp file cho phù hợp.
template	Chứa giao diện template (mẫu) cho từng trang có cấu trúc giống nhau.

Blade template

Tạo 1 template, dùng chung cho các trang news và các trang product.

- Các file include dùng chung cho các trang: header, footer.

- File template dành cho trang news và trang product.

- 2 trang hiển thị (Views) cho trang news và product.

Tất cả sẽ được chứa trong /resources/views/, các file được tổ chức như sau:

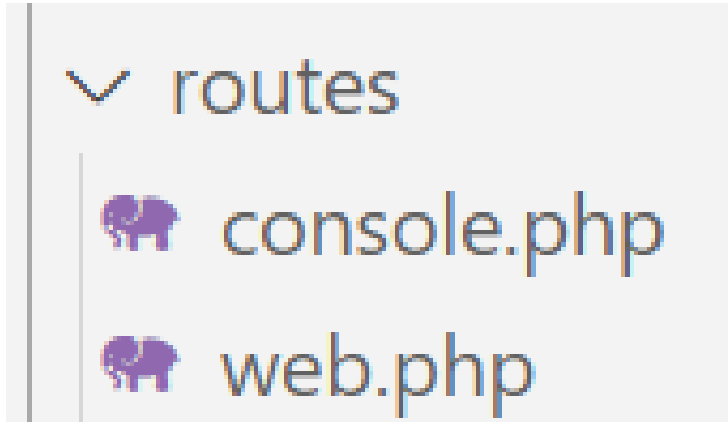
4 Route

What is Route?

Laravel không cho phép chạy file PHP bất kỳ nào một cách tùy tiện bằng việc gõ tên file lên thanh địa chỉ trình duyệt, mà tất cả phải được điều khiển thông qua Routing.

Routing được chứa trong thư mục /routes/

- /routes/api.php: điều khiển các route của ứng dụng, như route của ứng dụng User (đăng ký, đăng nhập, ...).
- /routes/web.php: điều khiển các route của view, như route của trang top, sản phẩm, ...



Cấu trúc Route

```
Route::METHOD('URL', ACTION);
```

METHOD

Route::get($uri, $callback): Nhận resquest với phương thức GET.

Route::post($uri, $callback): Nhận resquest với phương thức POST.

Route::put($uri, $callback): Nhận resquest với phương thức PUT.

Route::patch($uri, $callback): Nhận resquest với phương thức PATCH.

Route::delete($uri, $callback): Nhận resquest với phương thức DELETE.

METHOD

Route::options($uri, $callback): Nhận resquest với phương thức OPTIONS.

Route::match(['METHOD1', 'METHOD2',...], $uri, $callback): Nhận resquest với các phương thức kết hợp.

Route::any($uri, $callback): Nhận resquest với tất cả các phương thức.

Route::group('prefix'=>$uri, $callback): Nhóm các route.

METHOD

```php
Route::get(
    '/home',
    function () {
        $html="<h1>Welcome to Hanoi</h1>";
        return  $html;
    }
);
```

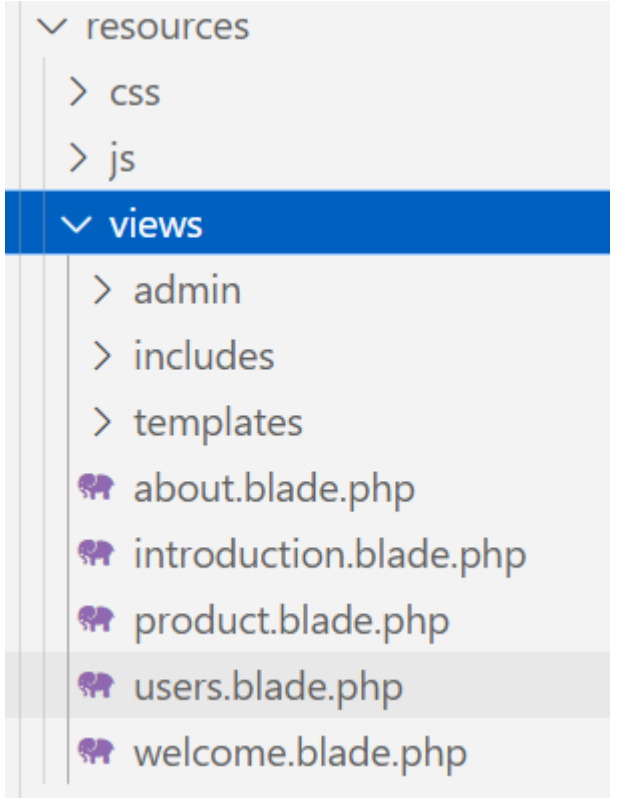
5 View

What is View?

Views là nơi giúp hiển thị những gì người dùng nhìn thấy trên trình duyệt, phần lớn là cấu trúc HTML.

Views chứa trong thư mục /resources/views/ đối với Laravel 8.x

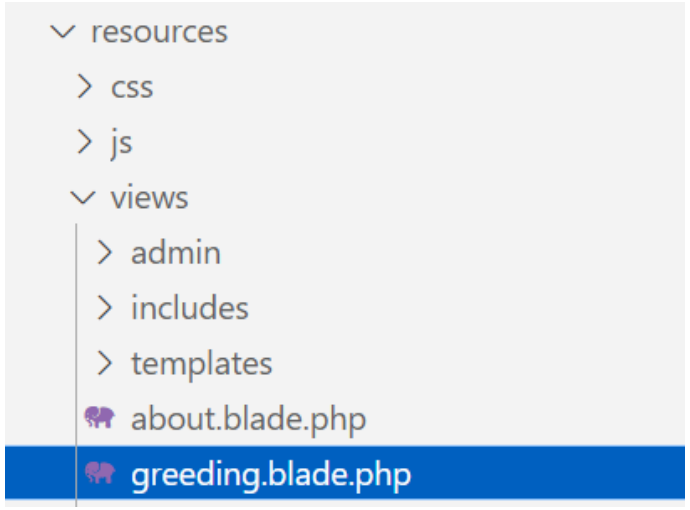
Views sử dụng cấu trúc Blade Templates, nên tên file của views sẽ là: name.blade.php, trong đó name là tên views bất kỳ.



What is View?

Views contain the HTML served by your application and separate your controller / application logic from your presentation logic.

Views are stored in the resources/views directory. A simple view might look something like this:

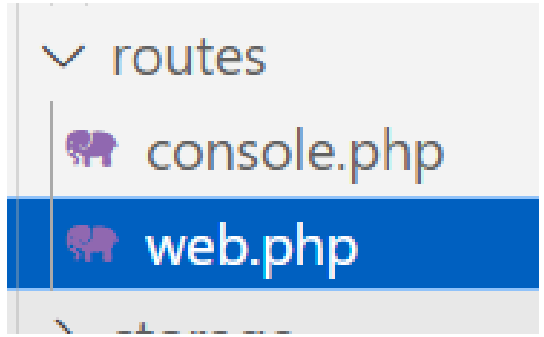


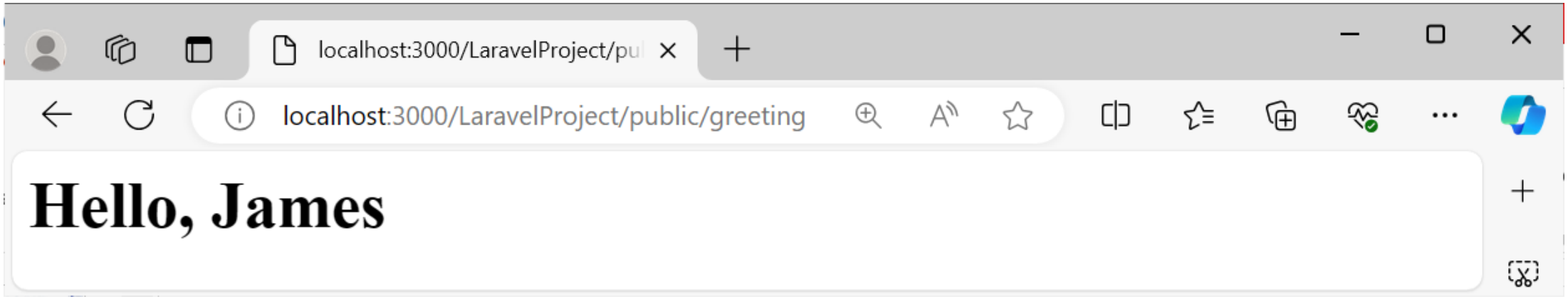

```
1  <!-- View stored in resources/views/greeting.blade.php -->
2
3  <html>
4      <body>
5          <h1>Hello, {{ $name }}</h1>
6      </body>
7  </html>
```

Create a view

Since this view is stored at resources/views/greeting.blade.php, we may return it using the global view helper like so (in /routes/web.php):

```php
Route::get('/greeting', function () {
    return view('greeting', ['name' => 'James']);
});
```





Create a view

As you can see, the first argument passed to the view helper corresponds to the name of the view file in the resources/views directory.

The second argument is an array of data that should be made available to the view.

In this case, we are passing the name variable, which is displayed in the view using Blade syntax.

```
Route::get('/greeting', function () {
    return view('greeting', ['name' => 'James']);
});
```

Create a view

Views may also be nested within sub-directories of the resources/views directory. "Dot" notation may be used to reference nested views.

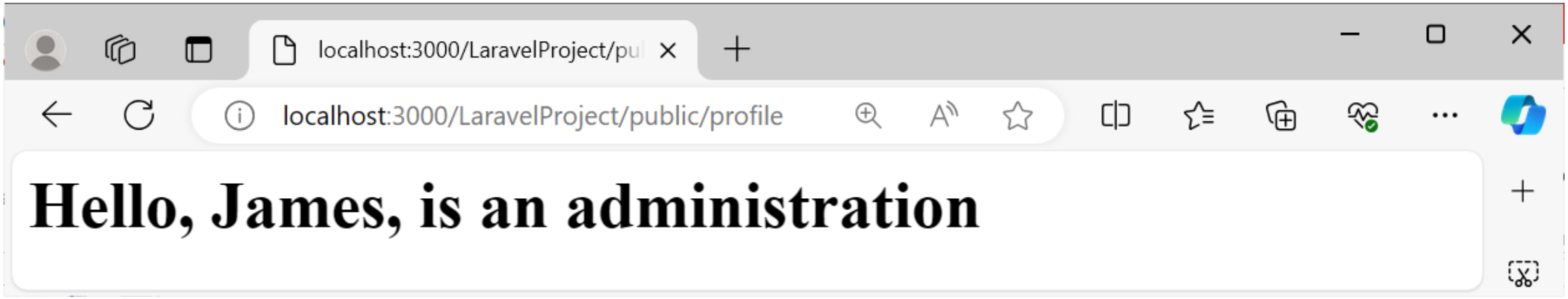
For example, if your view is stored at resources/views/admin/profile.blade.php, you may reference it like so:

```
<!-- View stored in resources/views/admin/profile.blade.php -->

<html>
    <body>
        <h1>Hello, {{ $name }}, is an administration</h1>
    </body>
</html>
```

Create a view

```php
Route::get('/profile', function () {
    return view('admin.profile', ['name' => 'James']);
});
```



6 Controllers

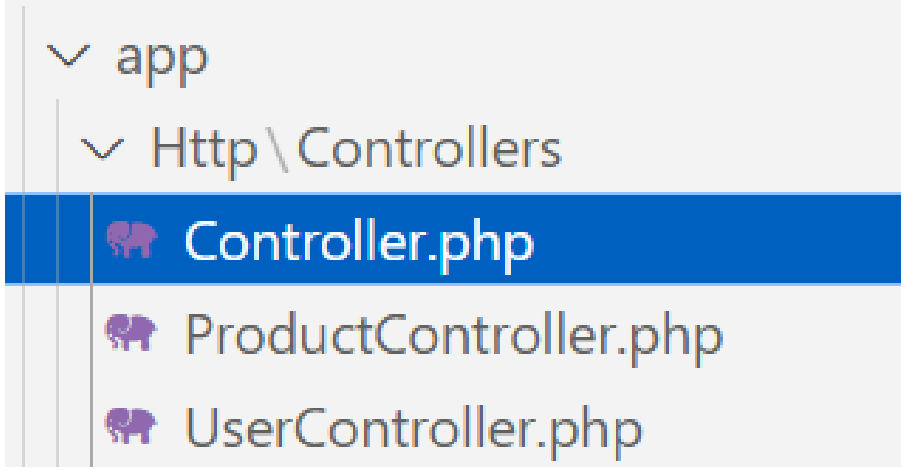
What is controller ?

Để xử lý logic (tính toán) thì 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 là trung tâm xử lý logic:

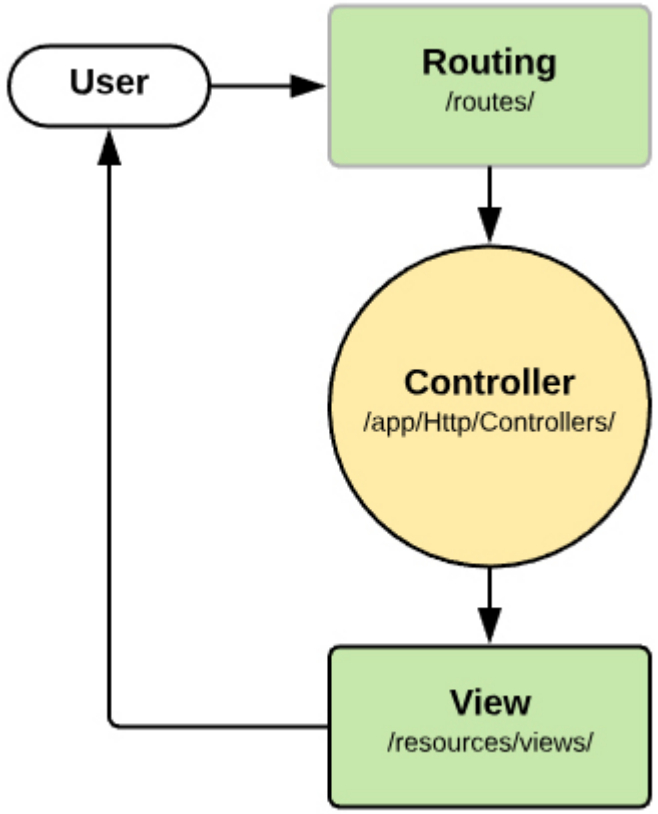
- Controller được đặt bên trong thư mục

/app/Http/Controllers/



What is 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.



Create a Controller

```
php artisan make:controller <Controller Name>
```

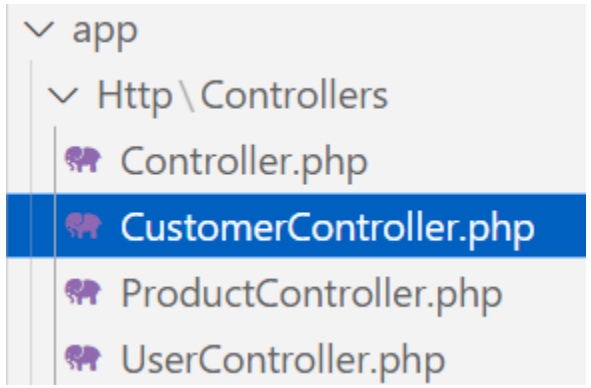
Where:

- php artisan: Công cụ hỗ trợ viết command line tích hợp sẵn trong Laravel
- make:controller: Lệnh tạo Controller.
- Controller Name: Tên Controller

Create a Controller

```
PS D:\Code_TeachingSubject\Code_OSSD\OSSD_Example\LaravelProject> php artisan make:controller CustomerController
```

```
  INFO  Controller [D:\Code_TeachingSubject\Code_OSSD\OSSD_Example\LaravelProject\app\Http\Controllers\CustomerController.php] created successfully.
```




```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class CustomerController extends Controller
{
    //
}
```

Create a Controller

- namespace App\Http\Controllers; là tên dùng để phân biệt những Controller có tên giống nhau, tuy nhiên thay vì dùng tên đại diện riêng, thì lại dùng đường dẫn thư mục tới file Controller App\Http\Controllers, do đó với namespace ta cần ghi đầy đủ cấu trúc thư mục chứa file Controller là được.
- use Illuminate\Http\Request; khi khai báo để sử dụng được các Request (như Get, Post, ...), hoặc dùng để gọi, quản lý các Class.
- CustomerController là tên class, tên này thường trùng với tên file Controller vừa tạo. Nội dung Controller được viết bên trong dấu ngoặc móc {}

Run a Controller

Step 1: Viết nội dung file Controller

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class CustomerController extends Controller
{
    public function index()
    {
        return view('customer'); // trả về trang customer.blade.php
    }
}
```

Run a Controller

Step 2: Tạo thêm một file View customer.blade.php đặt bên trong thư mục /resources/views/,

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>Customer</title>
</head>
<body>
    Đây là trang Customer !
</body>
</html>
```

Run a Controller

Step 3: Viết một routing trong /routes/web.php hiển thị trang view /resources/views/customer.blade.php thông qua một controller:

```
Route::get('/customer',
'App\Http\Controllers\CustomerController@index');
```

Trong đó:

- '/customer': là đường dẫn trên trình duyệt.

- 'App\Http\Controllers\CustomerController@index'

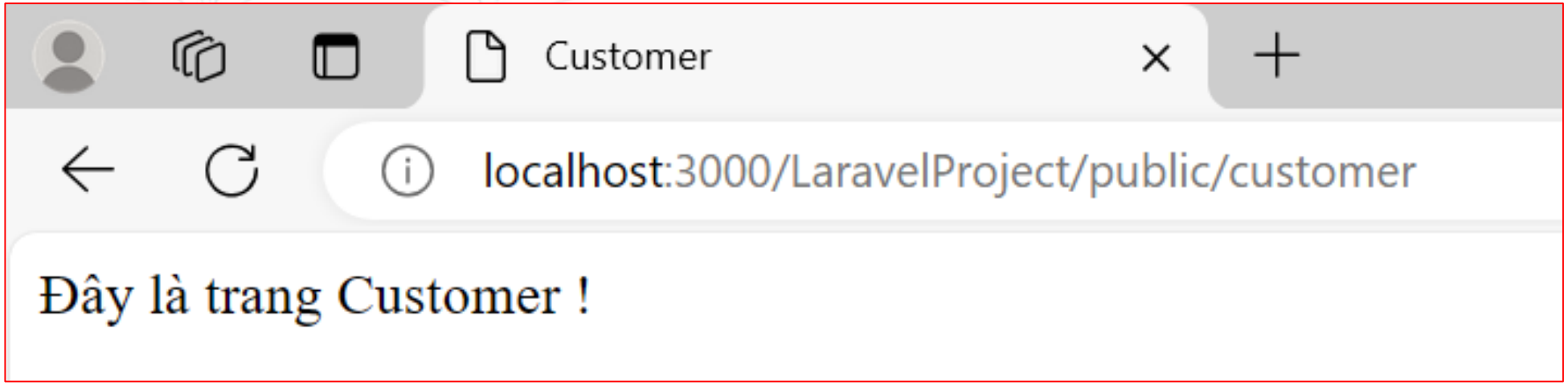
+ App\Http\Controllers:

+ CustomerController: là tên Controller.

+ index: là tên function của Controller.

Run a Controller

Step 4: Hiển thị trình duyệt



Handling a variable with Controllers

Step 1: In class CustomerController, create method login

```php
public function login()
{
    $name = "Nguyễn Minh Anh!";
    return view('login')->with('name', $name);

}
```

Biến $name được tạo và gửi biến này tới trang Views login thông qua phương thức with.

Handling a variable with Controllers

Step 2: In views, create file login.blade.php

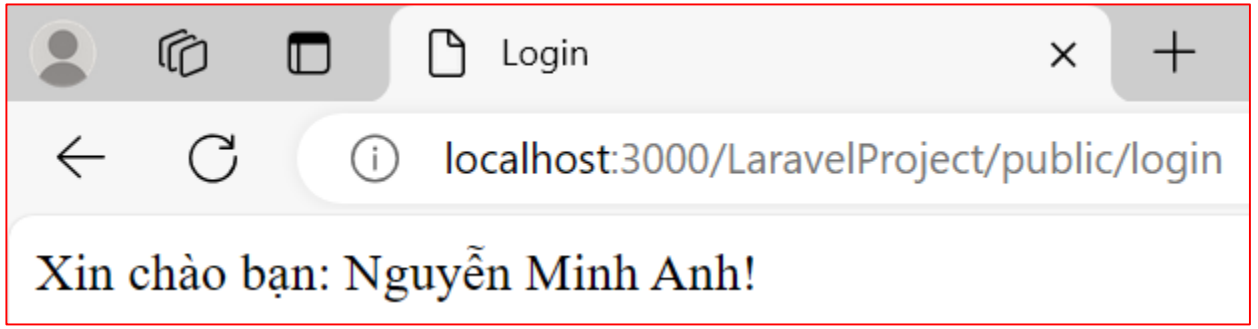
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Login</title>
</head>
<body>
Xin chào bạn: {{ $name }}
</body>
</html>
```

Handling a variable with Controllers

Step 3: Viết một routing trong /routes/web.php hiển thị trang view /resources/views/login.blade.php thông qua một controller:

```
Route::get('/login',
'App\Http\Controllers\CustomerController@login');
```

Step 4: Hiển thị trình duyệt



Handling variables with Controllers

Step 1: In class CustomerController, create method register

```
public function register()
{
    $name = "Nguyễn Minh Tùng";
    $gender = "Anh";
    return view('register')->with(['name' => $name, 'gender' => $gender]);
}
```

Handling variables with Controllers

Step 2: In views, create file register.blade.php

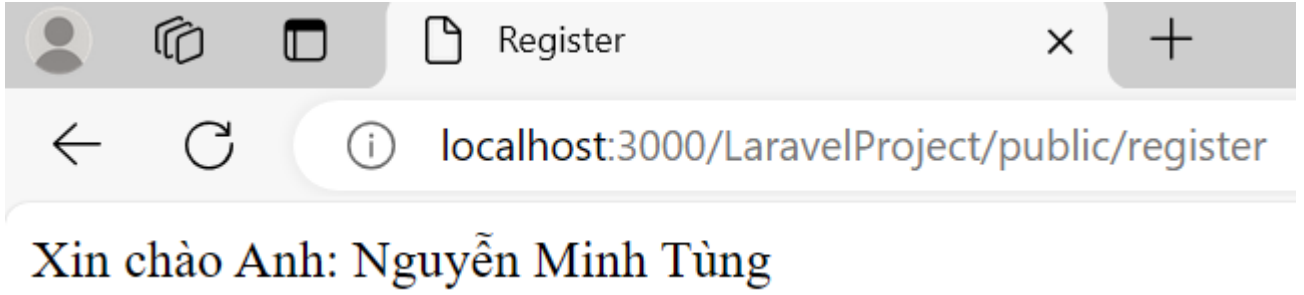
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Register</title>
</head>
<body>
Xin chào {{$gender}}: {{ $name }}
</body>
</html>
```

Handling variables with Controllers

Step 3: Viết một routing trong /routes/web.php hiển thị trang view /resources/views/register.blade.php thông qua một controller:

```
Route::get('/register',
'App\Http\Controllers\CustomerController@register');
```

Step 4: Hiển thị trình duyệt



7 Model

Create a Model

```
php artisan make:model <Model Name>
```

Where:

- php artisan: Công cụ hỗ trợ viết command line tích hợp sẵn trong Laravel
- make:model: Lệnh tạo Model.
- Model Name: Tên Model

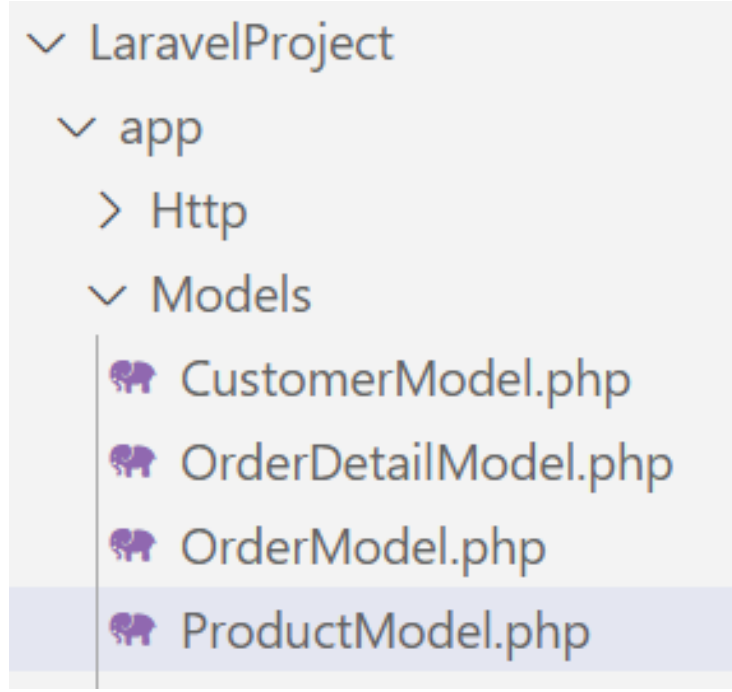
Create a Model

```
PS D:\Code_TeachingSubject\Code_OSSD\OSSD_Example> cd LaravelProject
PS D:\Code_TeachingSubject\Code_OSSD\OSSD_Example\LaravelProject> php artisan make:model ProductModel

  INFO  Model [D:\Code_TeachingSubject\Code_OSSD\OSSD_Example\LaravelProject\app\Models\ProductModel.php] created successful
ly.
```

```
php artisan make:model OrderDetailModel
```

```
php artisan make:model OrderModel
```



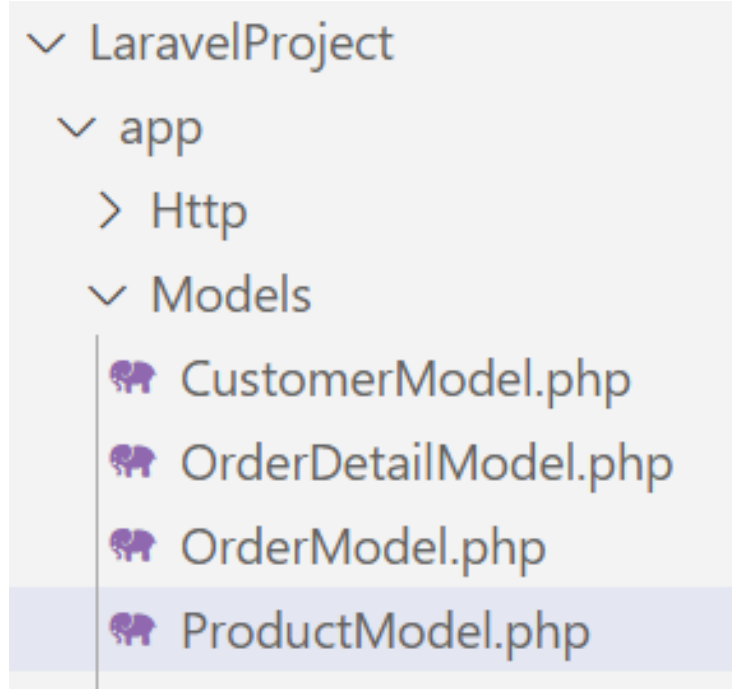
Create a Model

```
PS D:\Code_TeachingSubject\Code_OSSD\OSSD_Example> cd LaravelProject
PS D:\Code_TeachingSubject\Code_OSSD\OSSD_Example\LaravelProject> php artisan make:model ProductModel

  INFO  Model [D:\Code_TeachingSubject\Code_OSSD\OSSD_Example\LaravelProject\app\Models\ProductModel.php] created successful
ly.
```

```
php artisan make:model OrderDetailModel
```

```
php artisan make:model OrderModel
```



Create a Model

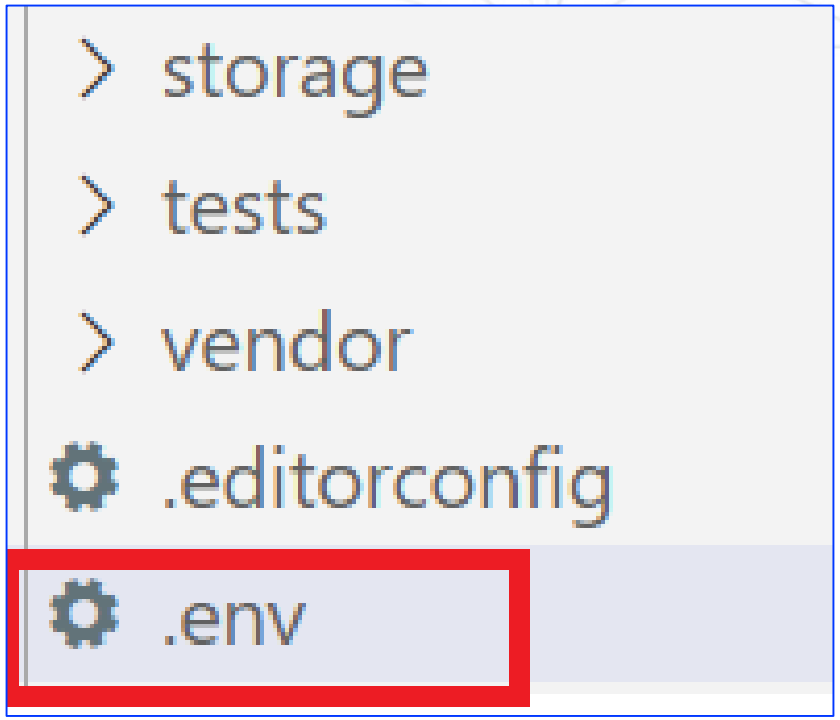
```php
class ProductModel extends Model
{
    use HasFactory;
    public $timestamps = false;

    protected $table = "tblProductA";

    public function hasOrderDetail()
    {
        return $this->hasMany(OrderDetailModel::class,'pid','odid');
    }
}
```

8 Database

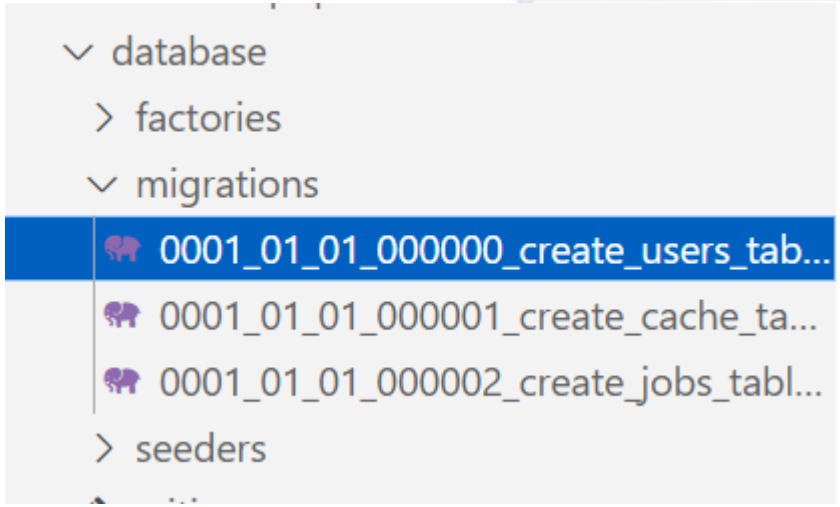
Config Database




```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=bmdb
DB_USERNAME=root
DB_PASSWORD=
```

Nếu thay đổi tên cơ sở dữ liệu thì cần chạy php artisan migrate

Run migration



php artisan migrate

```
PS D:\Code_TeachingSubject\Code_OSSD\OSSD_Example\LaravelProject> php artisan migrate
```

9 Route Group

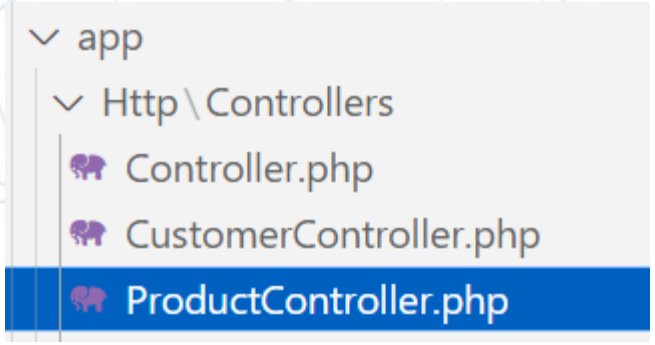
```php
Route::group(['prefix' => 'admin'], function () {
    Route::group(['prefix' => 'product'], function () {

    });

    Route::group(['prefix' => 'customer'], function () {
    });
});
```

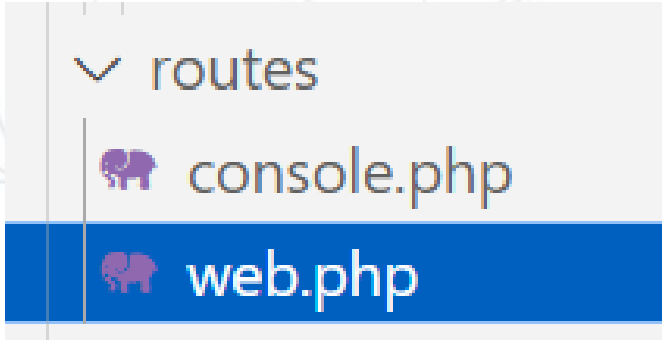
10 Find

ProductController



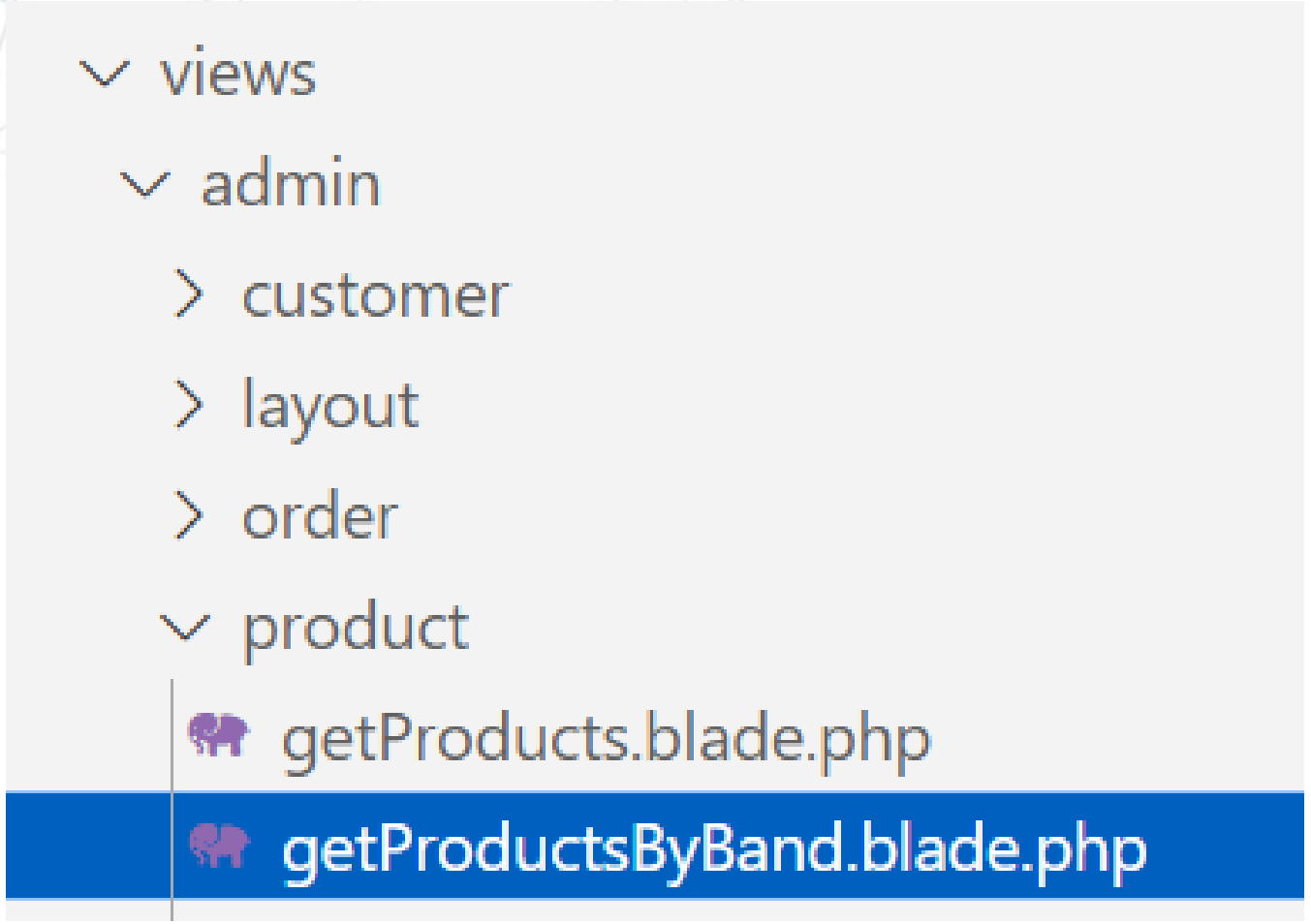

```php
function getProductsbyBand(Request $request)
{
  $band = $request->input('selectBand');
  $products = ProductModel::where('band', $band)->get();
  return view('admin.product.getProductsByBand', ['products' => $products])
}
```

routes/web.php




```
Route::get(
    'getProductsbyBand',
    'App\Http\Controllers\ProductController@getProductsbyBand'
)
    ->name('admin.product.getProducstByBand');
```

getProductsByBand.blade.php



getProductsByBand.blade.php

```
@extends('layout.master');
@section('Content')
//code here...
@endsection
```

getProductsByBand.blade.php

```
<form method="get" action="{{ route('admin.product.getProducstByBand') }}">
<input type="hidden" name="_token" value="{{csrf_token()}}" />
    <div class="mb-3">
        <label for="" class="form-label">Select Band</label>
        <select class="form-select form-select-lg" name="selectBand" id="">
            <option value="Minocin">Minocin</option>
            <option value="AeroGreen Antibac">AeroGreen Antibac</option>
            <option value="Bodycology">Bodycology</option>
            <option value="ibuprofen">ibuprofen</option>
        </select>
    </div>
    <div class="mb-3">
        <button type="submit" class="btn btn-primary" name="btSearch">
            Submit
        </button>
    </div>
</form>
```

getProductsByBand.blade.php

```
<table>
    <tbody>
        <thead>
            <tr>
                <th class=\"text-center\">PID</th>
                <th class=\"text-center\">Name</th>
                <th class=\"text-center\">Company</th>
                <th class=\"text-center\">Year</th>
                <th class=\"text-center\">Band</th>
                <th class=\"text-center\">Image</th>
                <th class=\"text-center\">Edit</th>
                <th class=\"text-center\">Delete</th>
            </tr>
        </thead>
```

getProductsByBand.blade.php

```
        @foreach($products as $product)
        <tr>
            <td class=\"text-left\"> {{$product->pid}} </td>
            <td class=\"text-left\"> {{$product->pname}} </td>
            <td class=\"text-left\"> {{$product->company}} </td>
            <td class=\"text-left\"> {{$product->year}} </td>
            <td class=\"text-left\">{{$product->band}} </td>
            <td class=\"text-left\"><img src="{{$product->pimage}} " alt="Image"></td>
            <td class="center"><i class="fa fa-trash-o  fa-fw"></i>
            <a href="deleteProduct/{{$product->pid}}"> Delete</a></td>
            <td class="center"><i class="fa fa-pencil fa-fw"></i>
            <a href="updateProduct/{{$product->pid}}">Edit</a></td>
        </tr>
        @endforeach
    </tbody>
</table>
```

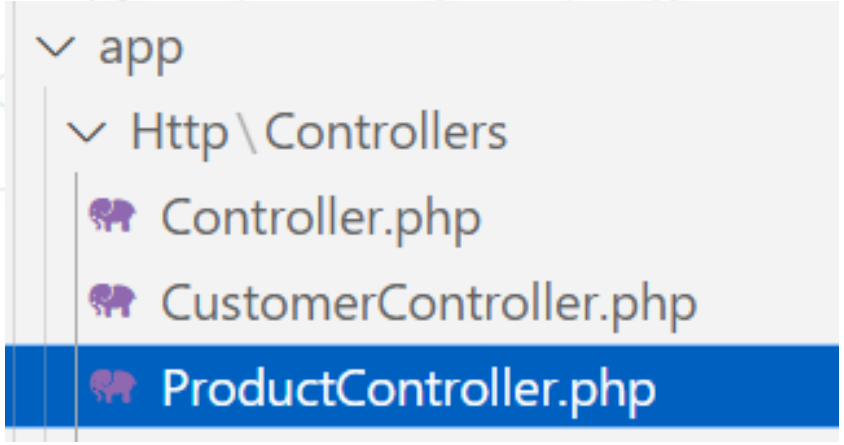
ProductModel

```
class ProductModel extends Model
{
    use HasFactory;
    public $timestamps = false;

    protected $table = "tblProductA";
    protected $primaryKey = 'pid';
    public $incrementing = false;
    protected $keyType = 'string';
```

11 Insert

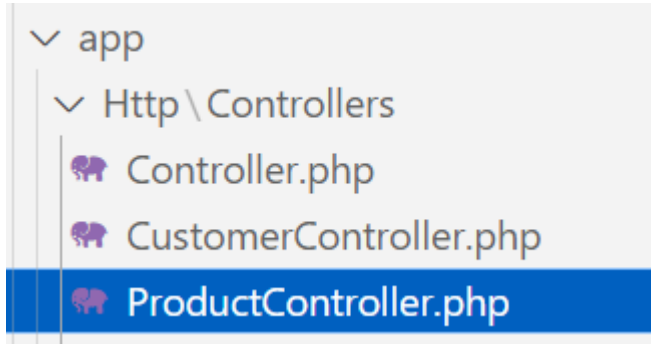
ProductController



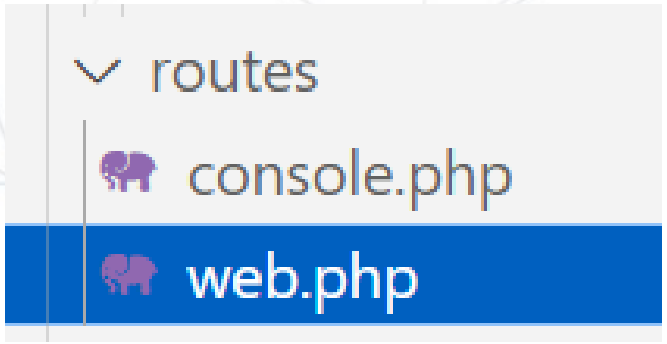

```php
function forminsertProduct()
{
    return view('admin.product.insertProduct');
}
```

ProductController

```
function insertProduct(Request $request)
{
  $product = new ProductModel;
  $product->pid = $request->id;
  $product->pname =  $request->pname;
  $product->company =  $request->company;
  $product->band = $request->selectBand;
  $product->year = $request->selectYear;
  if (isset($_FILES['imageFile']) && $_FILES['imageFile']['error'] === UPLOAD_ERR_OK) {
    $pimage = ''
      . base64_encode(file_get_contents($_FILES['imageFile']['tmp_name']));
    $product->pimage =   $pimage;
  }
  $product->save();
  return redirect('admin/product/insertProduct')
  ->with("Note", "Thêm thành công!");
}
```



routes/web.php




```
Route::get('insertProduct',
'App\Http\Controllers\ProductController@forminsertProduct');
Route::post('insertProduct',
'App\Http\Controllers\ProductController@insertProduct');
```

insertProduct.blade.php

```
@extends('layout.master');
@section('Content')
//code here...
@endsection
```

insertProduct.blade.php

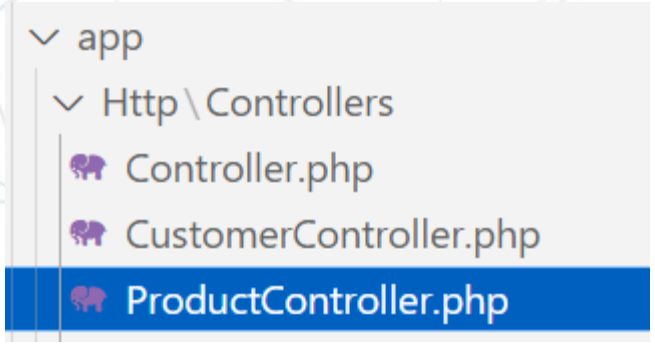
```
@if(session('Note'))
<div class="alert alert-success">
    {{session('Note')}}
</div>
@endif
```

insertProduct.blade.php

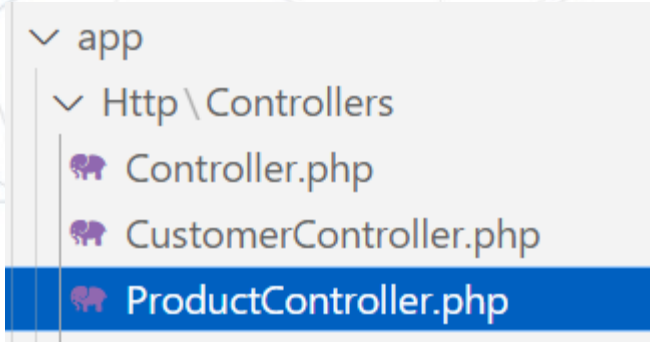
```
<form action="insertProduct" method="post" enctype="multipart/form-data">
    <div class="mb-3">
        <input type="hidden" name="_token" value="{{csrf_token()}}" />
    </div>
```

12 Update

ProductController

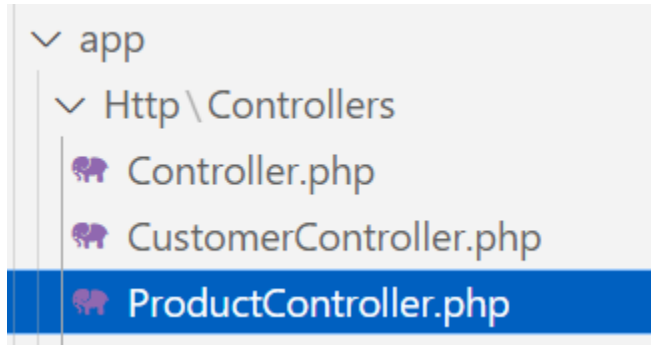


ProductController



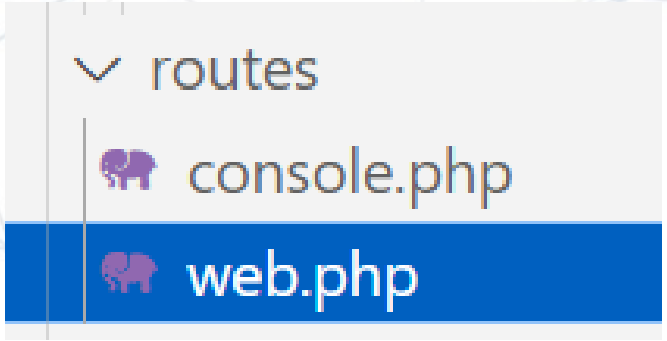

```php
function editProduct($pid)
{
    $product = ProductModel::where('pid', $pid)->first();
    return view('admin/product/updateProduct', ['product' => $product]
}
```

ProductController




```php
function updateProduct(Request $request, $pid)
{
  $product = ProductModel::where('pid', $pid)->first();
  $product->pid = $request->pid;
  $product->pname =  $request->pname;
  $product->company =  $request->company;
  $product->band = $request->selectBand;
  $product->year = $request->selectYear;
  if (isset($_FILES['imageFile']) && $_FILES['imageFile']['error'] === UPLOAD_ERR_OK) {
    // Get the image data
    $pimage = ''
      . base64_encode(file_get_contents($_FILES['imageFile']['tmp_name']));
    $product->pimage =   $pimage;
  }
  $product->save();
  return redirect('admin/product/updateProduct/' . $pid)
  ->with("Note", "Sửa thành công!");
}
```

routes/web.php




```php
//Route update
Route::get(
    'updateProduct/{pid}',
    'App\Http\Controllers\ProductController@editProduct'
);
Route::post(
    'updateProduct/{pid}',
    'App\Http\Controllers\ProductController@updateProduct'
);
```

updateProduct.blade.php

```
@extends('layout.master');
@section('Content')
//code here...
@endsection
```

updateProduct.blade.php

```
@if(session('Note'))
<div class="alert alert-success">
    {{session('Note')}}
</div>
@endif
```

updateProduct.blade.php

```
<form action="{{$product->pid}}" method="post" enctype="multipart/form-data">
    <div class="mb-3">
        <input type="hidden" name="_token" value="{{csrf_token()}}" />
    </div>
```

updateProduct.blade.php

```
<div class="mb-3">
    <label for="" class="form-label">ID Product </label>
    <input type="text" class="form-control" name="pid" id=""  value="{{$product->pid}}" />
</div>
<div class="mb-3">
    <label for="" class="form-label">Product Name</label>
    <input type="text" class="form-control" name="pname" value="{{$product->pname}}" id=""/>
</div>

<div class="mb-3">
    <label for="" class="form-label">Company</label>
    <input type="text" class="form-control" name="company" value="{{$product->company}}" id=""/>
```

updateProduct.blade.php

```
<div class="mb-3">
    <label for="" class="form-label">Select Band</label>
    <select class="form-select form-select-lg" name="selectBand" id="">
        <option value="Minocin">Minocin</option>
        <option value="AeroGreen Antibac">AeroGreen Antibac</option>
        <option value="Bodycology">Bodycology</option>
        <option value="ibuprofen">ibuprofen</option>

    </select>
</div>
```

updateProduct.blade.php

```
<div class="mb-3">
    <label for="" class="form-label">Select Year</label>
    <select class="form-select form-select-lg" name="selectYear" id="">
        <?php
        for ($year = 2015; $year <= 2025; $year++) {
            echo '<option value="' . $year . '">' . $year . '</option>';
        }
        ?>
    </select>

</div>
```

updateProduct.blade.php

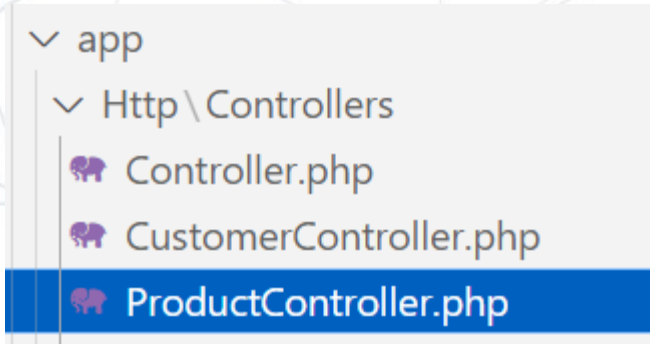
```
    <div class="mb-3">
        <label for="" class="form-label">Choose image</label>
        <img src="{{$product->pimage}} " alt="Image">
        <input type="file" class="form-control" name="imageFile">
    </div>

    <button type="submit" name="btUpdate" class="btn btn-primary">
        Update
    </button>

</form>
```

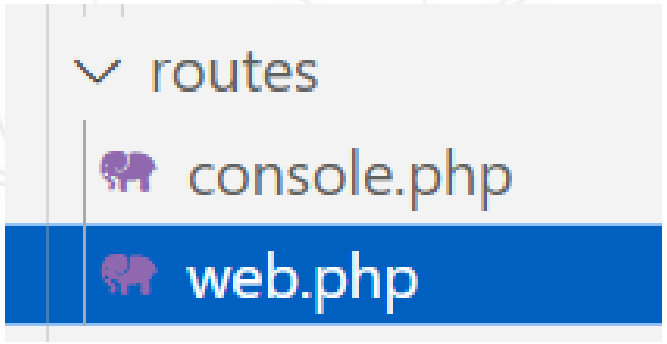
13 Delete

ProductController




```php
function deleteProduct($pid)
{
    $product = ProductModel::where('pid', $pid)->first();
    $product->delete();
    return redirect('admin/product/getProducts')
    ->with("Note", "Xoá thành công!");;
}
```

routes/web.php




```php
//Route delete
Route::get(
    'deleteProduct/{pid}',
    'App\Http\Controllers\ProductController@deleteProduct'
);
```

getProducts.blade.php

```
@if(session('Note'))
<div class="alert alert-success">
    {{session('Note')}}
</div>
@endif
```

Any

Question