



Xây dựng ứng dụng Laravel
Theo dạng PACKAGES

Nguyễn Minh Sang - Creator of Botble



Mục lục

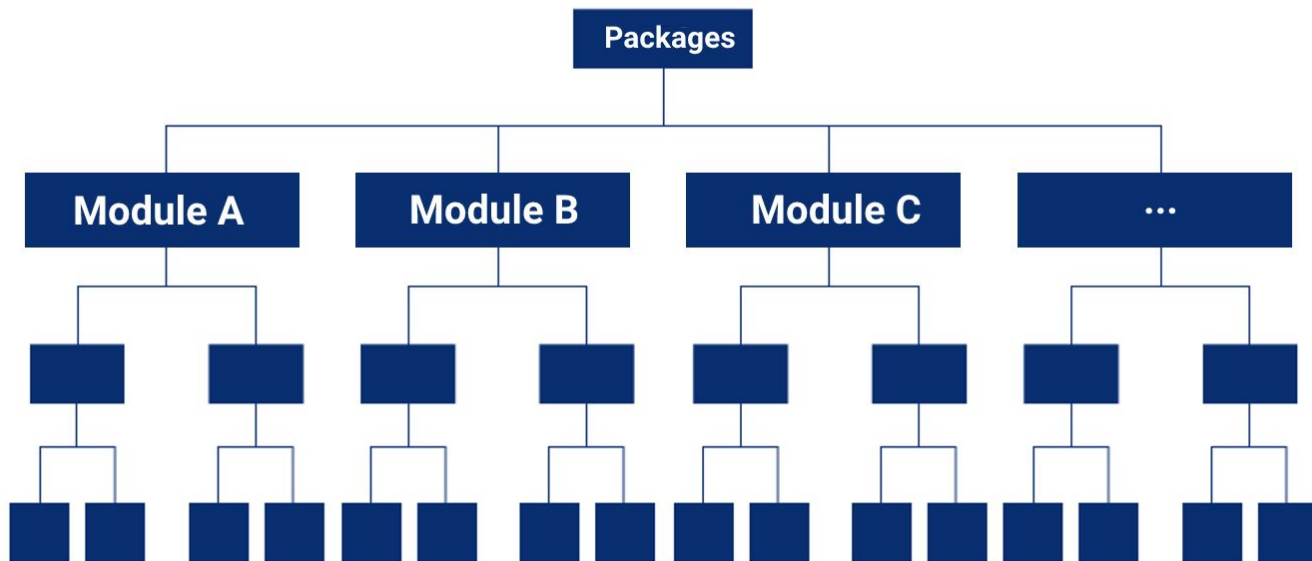
1. Tổng quan
2. Cách thức triển khai
3. Ưu & nhược điểm
4. Demo thực tế
5. Hỏi đáp



1. Tổng quan



MÔ HÌNH TỔ CHỨC THEO CHỨC NĂNG





Cấu trúc mặc định của Laravel

```
.
├── app/
│   ├── Console/
│   │   ├── Commands
│   │   └── Kernel.php
│   ├── Exceptions/
│   │   └── Handler.php
│   ├── Http/
│   │   ├── Controllers
│   │   ├── Requests
│   │   ├── Middleware
│   │   └── Kernel.php
│   ├── Models/
│   │   └── User.php
│   └── Providers/
│       ├── AppServiceProvider.php
│       ├── AuthServiceProvider.php
│       ├── BroadcastServiceProvider.php
│       ├── EventServiceProvider.php
│       └── RouteServiceProvider.php
```

```
├── routes/
│   ├── api.php
│   ├── channels.php
│   ├── console.php
│   └── web.php
├── lang/
│   └── en/
│       ├── auth.php
│       ├── pagination.php
│       ├── passwords.php
│       └── validation.php
└── ...
```



Khó quản lý khi ứng dụng lớn dần

```
.  
└─ app/  
    └─ Models/  
        ├── User.php  
        ├── Post.php  
        ├── Page.php  
        ├── Newsleter.php  
        ├── Address.php  
        ├── Brand.php  
        ├── BrandTranslation.php  
        ├── Currency.php  
        ├── Customer.php  
        ├── Discount.php  
        ├── DiscountCustomer.php  
        ├── DiscountProduct.php  
        ├── DiscountProductCollection.php  
        ├── FlashSale.php  
        ├── FlashSaleTranslation.php  
        ├── GroupedProduct.php  
        └── Order.php
```

```
└─ OrderAddress.php  
└─ OrderHistory.php  
└─ OrderProduct.php  
└─ OrderReferral.php  
└─ Product.php  
└─ ProductAttribute.php  
└─ ProductAttributeSet.php  
└─ ProductAttributeSetTranslation.php  
└─ ProductAttributeTranslation.php  
└─ ProductCategory.php  
└─ ProductCategoryTranslation.php  
└─ ProductCollection.php  
└─ ProductCollectionTranslation.php  
└─ ProductLabel.php  
└─ ProductLabelTranslation.php  
└─ ProductTag.php  
└─ ProductTagTranslation.php  
└─ ProductTranslation.php  
└─ ...
```



Một số dạng cấu trúc

github.com/nWidart/laravel-modules

☰ README.md

Autoloading

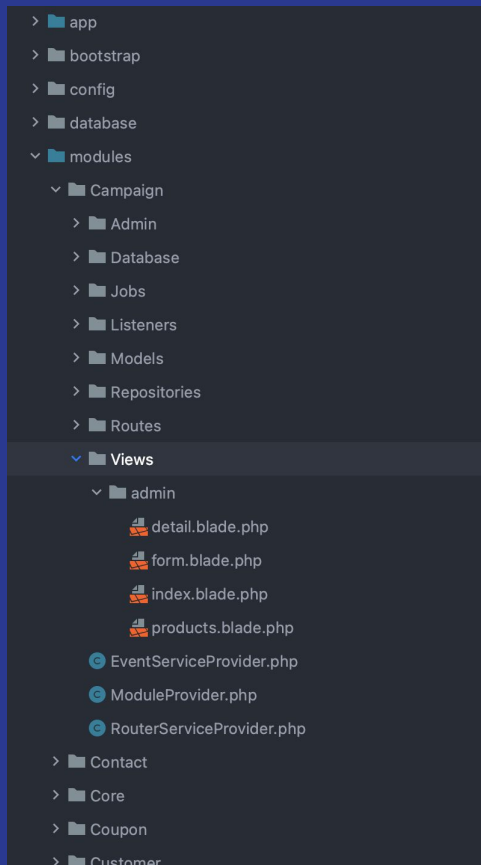
By default, the module classes are not loaded automatically. You can autoload your modules using `psr-4`. For example:

```
{
  "autoload": {
    "psr-4": {
      "App\\": "app/",
      "Modules\\": "Modules/",
      "Database\\Factories\\": "database/factories/",
      "Database\\Seeders\\": "database/seeders/"
    }
  }
}
```

Tip: don't forget to run `composer dump-autoload` afterwards.

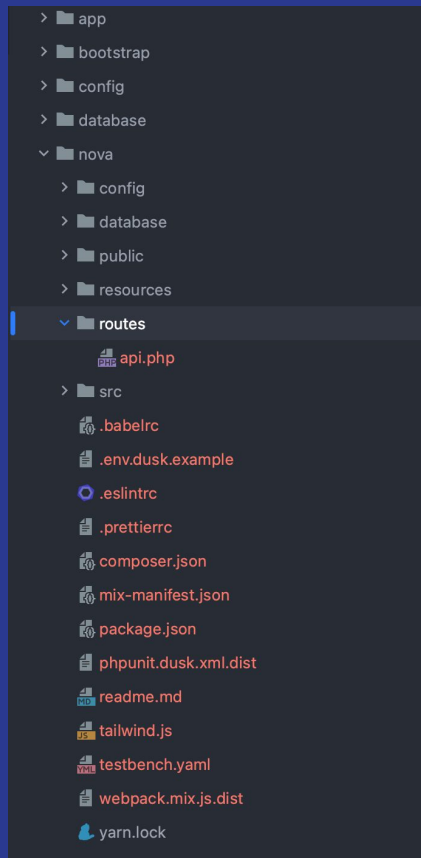


Một số dạng cấu trúc





Một số dạng cấu trúc





Cấu trúc dạng packages

```
.
├── app
├── packages/
│   ├── core/
│   │   ├── acl/
│   │   │   ├── lang
│   │   │   ├── routes
│   │   │   ├── resources
│   │   │   ├── src/
│   │   │   │   ├── Console
│   │   │   │   ├── Http/
│   │   │   │   │   ├── Controllers
│   │   │   │   │   ├── Requests
│   │   │   │   │   └── Middleware
│   │   │   │   ├── Events
│   │   │   │   ├── Models
│   │   │   │   ├── Providers
│   │   │   │   └── ...
│   │   │   ├── tests
│   │   │   ├── webpack.mix.js
│   │   │   └── composer.json
│   │   ├── media
│   │   ├── setting
│   │   └── ...
│   └── addons/
│       ├── blog
│       ├── newsletter
│       ├── ecommerce
│       ├── contact-form
│       ├── paypal
│       ├── stripe
│       ├── momo
│       ├── ghtk
│       └── ...
├── routes
├── lang
├── resources
└── ...
```



Composer.json

```
Project
├── .github
├── app
├── bootstrap
├── config
├── database
├── lang
├── node_modules library root
├── packages
├── public
├── resources
├── routes
├── storage
├── tests
├── vendor
├── .editorconfig
├── .env
├── .env.example
├── .gitattributes
├── .gitignore
├── .phpunit.result.cache
├── .styleci.yml
├── _ide_helper.php
├── artisan
├── CHANGELOG.md
├── composer.json
├── composer.lock
├── package.json
├── phpunit.xml
├── README.md
└── webpack.mix.js
```

```
1  {
2      "type": "project",
3      "description": "The Laravel Framework.",
4      "keywords": ["framework", "laravel"],
5      "license": "MIT",
6      "require": {
7          "php": "^8.0.2",
8          "guzzlehttp/guzzle": "^7.2", 7.4.5
9          "laravel/framework": "^9.19", v9.20.0
10         "laravel/sanctum": "^2.14.1", v2.15.1
11         "laravel/tinker": "^2.7", v2.7.2
12         "org/core": "*@dev" dev-master
13     },
14     "require-dev": {"fakerphp/faker": "^1.9.1"...},
15     "autoload": {...},
16     "autoload-dev": {...},
17     "scripts": {...},
18     "extra": {...},
19     "config": {"optimize-autoloader": true...},
20     "minimum-stability": "dev",
21     "prefer-stable": true,
22     "repositories": [
23         {
24             "type": "path",
25             "url": "./packages/*"
26         }
27     ]
28 }
```



Composer.json

```
Project ▾
├── lang
├── node_modules library root
├── packages
│   └── core
│       ├── config
│       ├── resources
│       ├── routes
│       ├── src
│       └── tests
│           ├── composer.json
│           └── webpack.mix.js
├── public
├── resources
├── routes
├── storage
├── tests
├── vendor
├── .editorconfig
├── .env
├── .env.example
├── .gitattributes
├── .gitignore
├── .phpunit.result.cache
├── .styleci.yml
├── _ide_helper.php
├── artisan
├── CHANGELOG.md
├── composer.json
├── composer.lock
└── package.json
```

```
1 {
2     "name": "org/core",
3     "description": "The core module",
4     "type": "package",
5     "autoload": {
6         "psr-4": {
7             "Org\\Core\\": "src",
8             "Org\\Core\\Tests\\": "tests"
9         }
10    },
11    "require": {
12        "barryvdh/laravel-dumpdf": "^2.0",
13        "mews/purifier": "^3.3"
14    },
15    "extra": {
16        "laravel": {
17            "providers": [
18                "Org\\Core\\Providers\\CoreServiceProvider"
19            ],
20            "aliases": {
21                "Core": "Org\\Core\\Facades\\CoreFacade"
22            }
23        }
24    }
25 }
```



Provider

packages

core

config

resources

routes

src

Commands

Events

Exceptions

Facades

Http

Listeners

Models

Providers

CoreServiceProvider.php

EventServiceProvidi

Core.php

tests

DemoTest.php

composer.json

public

resources

routes

storage

tests

vendor

.editorconfig

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

```
class CoreServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->singleton( abstract: ExceptionHandler::class, concrete: CustomHandlerException::class);
    }

    public function boot()
    {
        // NOTE: Load routes, you can load many route files as you wish.
        $this->loadRoutesFrom( path: __DIR__ . '/../routes/web.php');

        // NOTE: Load views, the same as load routes
        $this->loadViewsFrom( path: __DIR__ . '/../resources/views', namespace: 'org/core');

        // NOTE: Load configs
        $this->mergeConfigFrom( path: __DIR__ . '/../config/core.php', key: 'core');

        $this->app->bind( abstract: 'core', function () {
            return new Core();
        });

        // NOTE: This facade can be loaded directly via PHP code, not using composer
        AliasLoader::getInstance()->alias( alias: 'CoreFacadeLoadedDirectly', class: CoreFacadeLoadedDirectlyFacade::class);
    }
}
```



Provider

packages

core

config

resources

routes

src

Commands

Events

Exceptions

Facades

Http

Listeners

Models

Providers

CoreServiceProvider

EventServiceProvider

Core.php

tests

DemoTest.php

composer.json

public

resources

routes

storage

tests

vendor

editorconfig

env

env.example

gitattributes

gitignore

phpunit.result.cache

styleci.yml

ide_helper.php

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

```
/**
 * @var Router $router
 */
$router = $this->app['router'];

// NOTE: Option 1: this middleware will be added into group web so every requests from web will trigger this middleware
$router->pushMiddlewareToGroup( group: 'web', middleware: DemoMiddleware::class);

// NOTE: Option 2: named for middleware, then we can use it in routes.
$router->aliasMiddleware( name: 'demoMiddleware', class: DemoMiddleware::class);

// NOTE: Option 3: define a group, easy to use multiple middleware in routes, the same as "web".
$router->middlewareGroup( name: 'demo', [DemoMiddleware::class, ValidateSignature::class]);

// NOTE: Override existing middleware in app/Http/Middleware
$this->app->instance( abstract: VerifyCsrfToken::class, new CustomVerifyCsrfTokenMiddleware());

// NOTE: Register a new command
$this->commands([
    DemoCommand::class,
]);

// NOTE: Used to register events & its listeners
$this->app->register( provider: EventServiceProvider::class);
}

public function provides()
{
    return ['core'];
}
```



Facades

Cách thông thường

```
// config/app.php

'aliases' => Facade::defaultAliases()->merge([
    // 'ExampleClass' => App\Example\ExampleClass::class,
    'Core' => \Org\Core\Facades\CoreFacade::class
])->toArray(),
```

Hoặc

```
"extra": {
    "laravel": {
        "dont-discover": [],
        "aliases": {
            "Core": "Org\\Core\\Facades\\CoreFacade"
        }
    }
},
```



Facades

packages/core/composer.json

```
"extra": {  
    "laravel": {  
        "providers": [  
            "Org\\Core\\Providers\\CoreServiceProvider"  
        ],  
        "aliases": {  
            "Core": "Org\\Core\\Facades\\CoreFacade"  
        }  
    }  
}
```




Facades

Hoặc dùng AliasLoader

```
// packages/core/src/Providers/CoreServiceProvider.php  
// NOTE: This facade can be loaded directly via PHP code, not using composer  
AliasLoader::getInstance()->alias( alias: 'CoreFacadeLoadedDirectly', class: CoreFacadeLoadedDirectlyFacade::class);
```



Commands

Cách thông thường

The screenshot shows an IDE interface with a project explorer on the left and a code editor on the right. The project explorer shows a directory structure with 'app' as the root, containing 'Console', 'Commands', 'Exceptions', 'Http', 'Models', 'Providers', and 'bootstrap'. The 'Commands' directory is expanded, showing 'DemoCommand.php' and 'Kernel.php'. The code editor displays the contents of 'DemoCommand.php', which is a new file. The code defines a namespace, uses the 'Illuminate\Console\Command' class, and defines a 'DemoCommand' class that extends 'Command'. A docblock is also present, describing the command's name and signature.

```
1 <?php
2
3 namespace App\Console\Commands;
4
5 use Illuminate\Console\Command;
6
7 class DemoCommand extends Command
8 {
9     /**
10      * The name and signature of the console command.
11      *
12      * @var string
```



Commands

Theo dạng packages

The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure is as follows:

- Project
 - packages
 - core
 - config
 - resources
 - routes
 - src
 - Commands
 - DemoCommand.php
 - Events
 - Exceptions
 - Facades
 - Http
 - Listeners
 - Models

The code editor displays the content of `DemoCommand.php`:

```
1 <?php
2
3 namespace Org\Core\Commands;
4
5 use Core;
6 use Illuminate\Console\Command;
7
8 class DemoCommand extends Command
9 {
10     /**
11      * The name and signature of the console command.
12      *
13      * @var string
14      */
15     protected $signature = 'demo';
16
17     /**
```



Commands

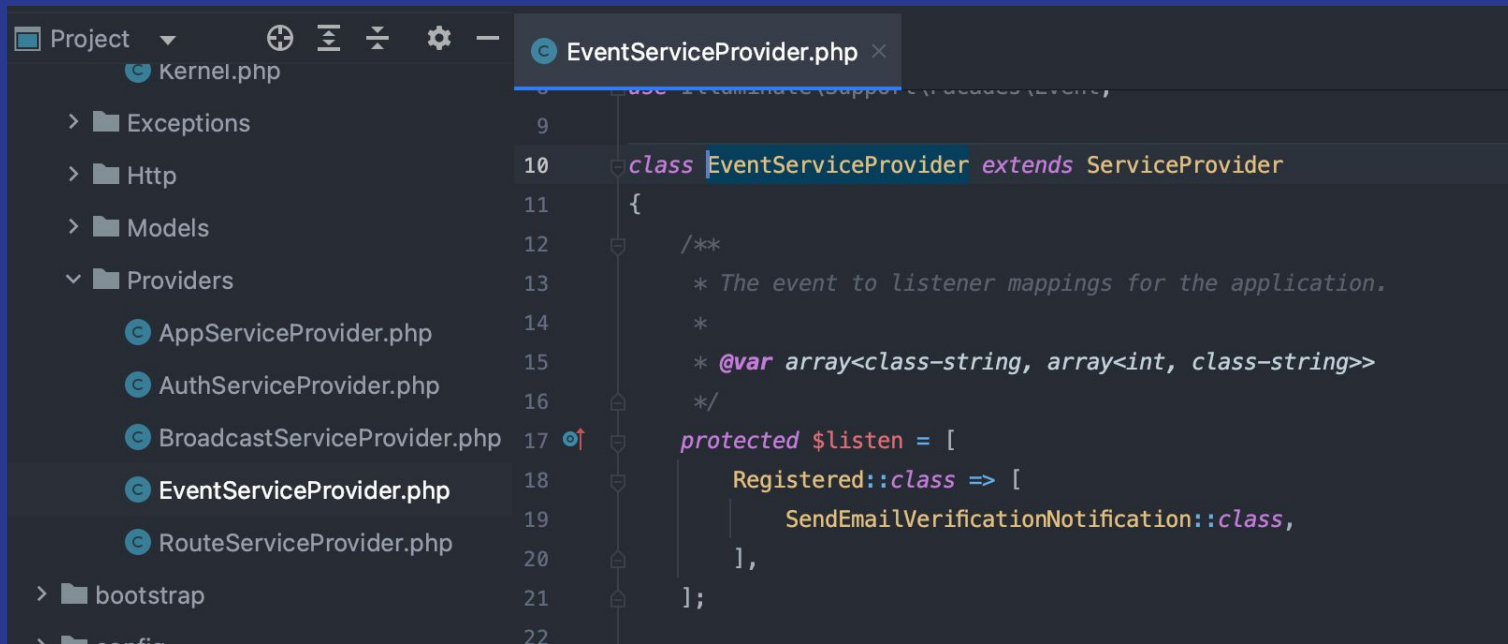
Khai báo command

```
// packages/core/src/Providers/CoreServiceProvider.php  
// NOTE: Register a new command  
$this->commands([  
    DemoCommand::class,  
]);
```



Events

Cách thông thường



```
Project ▾
  Kernel.php
  > Exceptions
  > Http
  > Models
  ▾ Providers
    AppServiceProvider.php
    AuthServiceProvider.php
    BroadcastServiceProvider.php
    EventServiceProvider.php
    RouteServiceProvider.php
  > bootstrap
  > config
```

```
EventServiceProvider.php
1 use Illuminate\Support\Facades\Event;
2
3 class EventServiceProvider extends ServiceProvider
4 {
5     /**
6      * The event to listener mappings for the application.
7      *
8      * @var array<class-string, array<int, class-string>>
9      */
10    protected $listen = [
11        Registered::class => [
12            SendEmailVerificationNotification::class,
13        ],
14    ];
15 }
```



Events

Theo dạng packages

```
<?php
namespace Org\Core\Providers;

use Illuminate\Foundation\Support\Providers\EventServiceProvider as ServiceProvider;
use Org\Core\Events\DemoEvent;
use Org\Core\Listeners\DemoListener;

class EventServiceProvider extends ServiceProvider
{
    protected $listen = [
        DemoEvent::class => [
            DemoListener::class,
        ],
    ];
}
```



Events

Đăng ký EventServiceProvider

```
// packages/core/src/Providers/CoreServiceProvider.php  
// NOTE: Used to register events & its listeners  
$this->app->register( provider: EventServiceProvider::class);
```

Hoặc trong composer.json

```
"extra": {  
    "laravel": {  
        "providers": [  
            "Org\\Core\\Providers\\CoreServiceProvider",  
            "Org\\Core\\Providers\\EventServiceProvider"  
        ],  
        "aliases": {
```



Migrations

Load migrations

```
25 }
26
27 // packages/core/src/Providers/CoreServiceProvider.php
28 public function boot()
29 {
30     // NOTE: Load migrations from path
31     $this->loadMigrationsFrom( paths: __DIR__ . '/../../database/migrations');
32 }
```

File explorer on the left shows the directory structure:

- core
 - config
 - database
 - migrations
 - 2022_08_05_152815_dei
 - resources



Routes

Load routes

26	
27	<code>// packages/core/src/Providers/CoreServiceProvider.php</code>
28	<code>public function boot()</code>
29	<code>{</code>
30	<code> <i>// NOTE: Load routes, you can load many route files as you wish.</i></code>
31	<code> \$this->loadRoutesFrom(path: __DIR__ . '/../../routes/web.php');</code>
32	

Khai báo routes

1	<code><?php</code>
2	
3	<code>use Illuminate\Support\Facades\Route;</code>
4	<code>use Org\Core\Http\Controllers\DemoController;</code>
5	
6	<code>Route::group(['middleware' => ['web']], function () {</code>
7	<code> Route::get(uri: 'demo', [DemoController::class, 'demo']);</code>
8	<code> Route::get(uri: 'demo2', [DemoController::class, 'demo2']);</code>
9	<code> Route::get(uri: 'demo3', [DemoController::class, 'demo3']);</code>
10	<code>});</code>
11	
12	



Views

Load views

26	
27	<code>// packages/core/src/Providers/CoreServiceProvider.php</code>
28	<code>public function boot()</code>
29	<code>{</code>
30	<code> // NOTE: Load views, the same as load routes</code>
31	<code> \$this->loadViewsFrom(path: __DIR__ . '/../resources/views', namespace: 'org/core');</code>
32	<code>}</code>

views

demo2.blade.php

demo3.blade.php

routes

src



Config

Load config

```
25  }
26
27  // packages/core/src/Providers/CoreServiceProvider.php
28  public function boot()
29  {
30      // NOTE: Load configs
31      $this->mergeConfigFrom( path: __DIR__ . '/../../config/core.php', key: 'core');
32  }
```

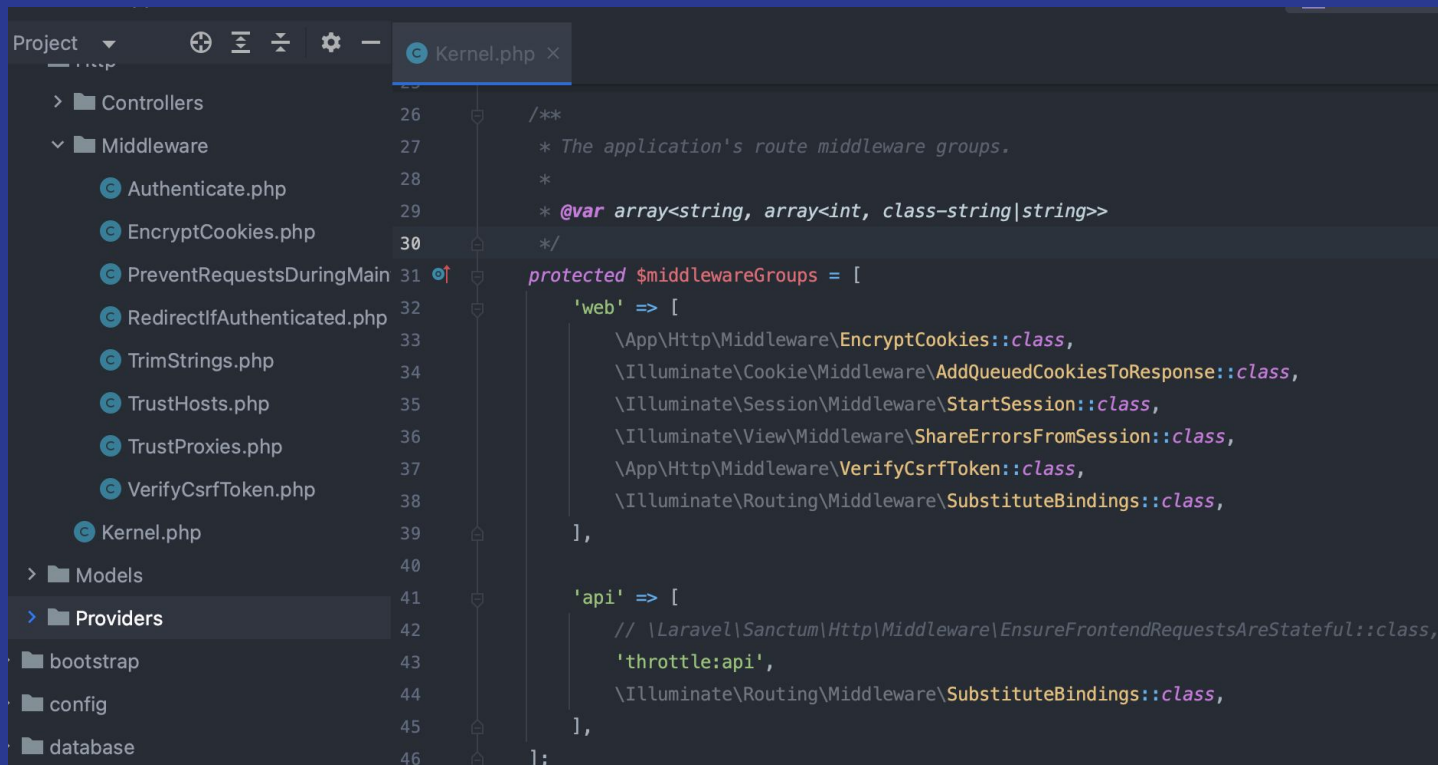
File explorer on the left shows the directory structure:

- packages
 - core
 - config
 - core.php
 - resources
 - routes



Middleware

Cách thông thường



```
Project ▾
  ▾ ...
    ▸ Controllers
    ▾ Middleware
      Authenticate.php
      EncryptCookies.php
      PreventRequestsDuringMain
      RedirectIfAuthenticated.php
      TrimStrings.php
      TrustHosts.php
      TrustProxies.php
      VerifyCsrfToken.php
      Kernel.php
    ▸ Models
    ▸ Providers
  bootstrap
  config
  database

Kernel.php x
26  /**
27   * The application's route middleware groups.
28   *
29   * @var array<string, array<int, class-string|string>>
30   */
31  protected $middlewareGroups = [
32      'web' => [
33          \App\Http\Middleware\EncryptCookies::class,
34          \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
35          \Illuminate\Session\Middleware\StartSession::class,
36          \Illuminate\View\Middleware\ShareErrorsFromSession::class,
37          \App\Http\Middleware\VerifyCsrfToken::class,
38          \Illuminate\Routing\Middleware\SubstituteBindings::class,
39      ],
40
41      'api' => [
42          // \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
43          'throttle:api',
44          \Illuminate\Routing\Middleware\SubstituteBindings::class,
45      ],
46  ];
```



Middleware

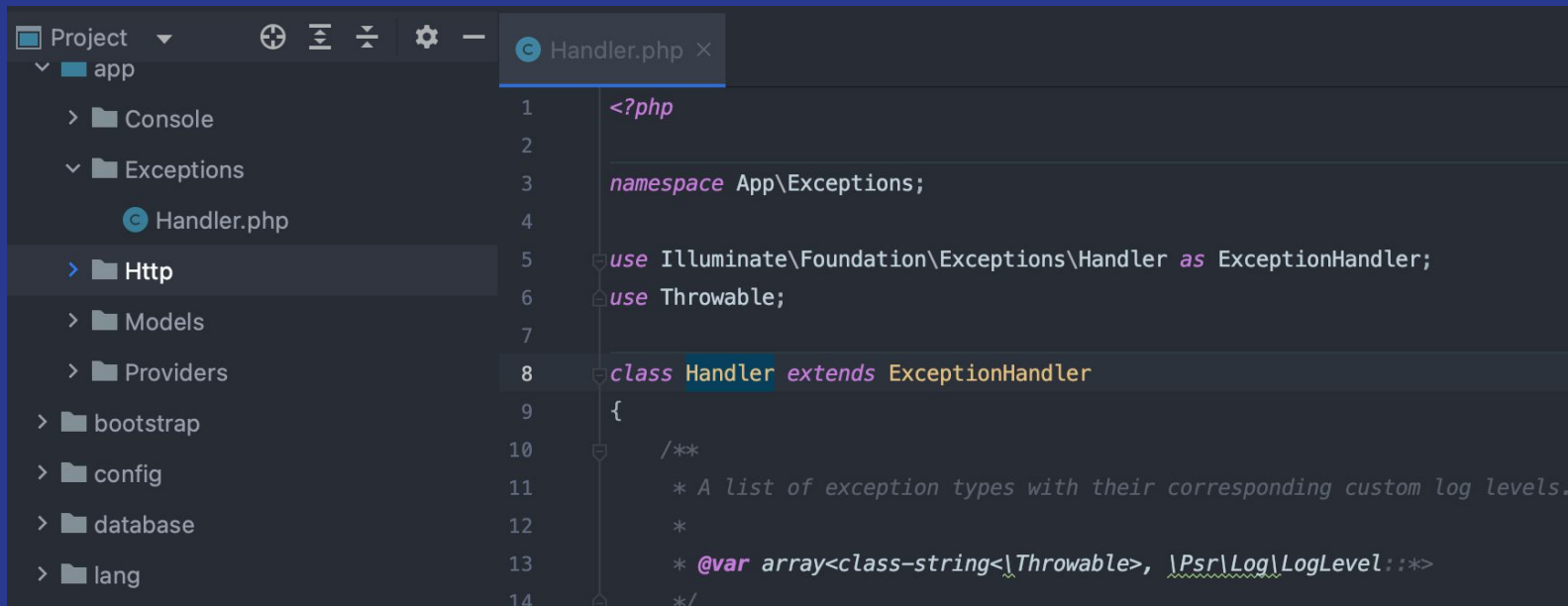
Dạng packages

```
Event::listen( events: RouteMatched::class, function () {  
    /**  
     * @var Router $router  
     */  
    $router = $this->app['router'];  
  
    // NOTE: Option 1: this middleware will be added into group web so every requests from web will trigger this middleware  
    $router->pushMiddlewareToGroup( group: 'web', middleware: DemoMiddleware::class);  
  
    // NOTE: Option 2: named for middleware, then we can use it in routes.  
    $router->aliasMiddleware( name: 'demoMiddleware', class: DemoMiddleware::class);  
  
    // NOTE: Option 3: define a group, easy to use multiple middleware in routes, the same as "web".  
    $router->middlewareGroup( name: 'demo', [DemoMiddleware::class, ValidateSignature::class]);  
  
    // NOTE: Override existing middleware in app/Http/Middleware  
    $this->app->instance( abstract: VerifyCsrfToken::class, new CustomVerifyCsrfTokenMiddleware());  
});
```



Handler.php

Dạng thông thường



The screenshot shows an IDE with a project structure on the left and the content of `Handler.php` in the main editor. The project structure includes folders for `Console`, `Exceptions`, `Http`, `Models`, `Providers`, `bootstrap`, `config`, `database`, and `lang`. The `Handler.php` file is located in the `Exceptions` folder. The code in the editor is as follows:

```
1  <?php
2
3  namespace App\Exceptions;
4
5  use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
6  use Throwable;
7
8  class Handler extends ExceptionHandler
9  {
10     /**
11      * A list of exception types with their corresponding custom log levels.
12      *
13      * @var array<class-string<Throwable>, \Psr\Log\LogLevel::*>
14      */
```



Handler.php

Dạng packages

The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes folders for routes, src (with subfolders Commands, Events, Exceptions, Facades, Http, Listeners, Models, Providers), tests, and files like composer.json, webpack.mix.js, and Core.php. The file CustomHandlerException.php is selected in the Exceptions folder. The code editor shows the following PHP code:

```
1 <?php
2
3 namespace Org\Core\Exceptions;
4
5 use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
6 use Symfony\Component\HttpKernel\Exception\HttpExceptionInterface;
7 use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
8
9 class CustomHandlerException extends ExceptionHandler
10 {
11     protected function renderHttpException(HttpExceptionInterface $e)
12     {
13         if ($e instanceof NotFoundHttpException) {
14             dd( ...vars: 'You are lost.' );
15         }
16
17         return parent::renderHttpException($e);
18     }
19 }
```



Handler.php

Khai báo

```
class CoreServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->singleton( abstract: ExceptionHandler::class, concrete: CustomHandlerException::class);
    }
}
```




Webpack

webpack.mix.js x

```
1  let glob = require('glob');
2
3  // Run all webpack.mix.js in app
4  glob.sync( pattern: './packages/**/webpack.mix.js').forEach(item => require(item));
5
```



Webpack

A screenshot of a code editor with a dark theme. On the left is a file explorer showing a project structure with folders like bootstrap, config, database, lang, node_modules, packages, public, resources, routes, storage, and tests. The 'webpack.mix.js' file is selected under the 'packages' folder. The main editor area shows the content of 'webpack.mix.js' with line numbers 1 through 12. The code defines a 'mix' object with methods for including JavaScript and Sass files, using 'source' and 'dist' variables to specify file paths.

```
1 let mix = require('laravel-mix');  
2  
3 const path = require('path');  
4 let directory = path.basename(path.resolve(__dirname));  
5  
6 const source = 'packages/' + directory;  
7 const dist = 'public/vendor/' + directory;  
8  
9 mix  
10   .js( src: source + '/resources/assets/js/script.js', output: dist + '/js')  
11   .sass( src: source + '/resources/assets/sass/style.scss', output: dist + '/js');  
12
```



Tests

A screenshot of an IDE interface showing a project structure on the left and the contents of a composer.json file in the center. The project structure includes a 'packages' directory with a 'core' subdirectory. The 'core' directory contains 'config', 'resources', 'routes', 'src', and 'tests' subdirectories. The 'src' directory contains 'Commands', 'Events', 'Exceptions', 'Facades', 'Http', 'Listeners', 'Models', 'Providers', and 'Core.php'. The 'tests' directory contains 'DemoTest.php'. The composer.json file is open in the center, showing the 'autoload' section with a red box highlighting the 'psr-4' configuration. The 'psr-4' configuration maps '0rg\\Core\\' to 'src' and '0rg\\Core\\Tests\\' to 'tests'. The 'require' section lists 'barryvdh/laravel-dompdf' as a dependency. The 'extra' section is empty. The 'laravel' section lists 'providers' as '0rg\\Core\\Providers\\CoreServiceProvider' and 'aliases' as 'Core' pointing to '0rg\\Core\\Facades\\CoreFacade'.

```
1 {
2     "name": "org/core",
3     "description": "The core module",
4     "type": "package",
5     "autoload": {
6         "psr-4": {
7             "0rg\\Core\\": "src",
8             "0rg\\Core\\Tests\\": "tests"
9         }
10    },
11    "require": {"barryvdh/laravel-dompdf": "^2.0"...},
12    "extra": {
13        "laravel": {
14            "providers": [
15                "0rg\\Core\\Providers\\CoreServiceProvider"
16            ],
17            "aliases": {
18                "Core": "0rg\\Core\\Facades\\CoreFacade"
19            }
20        }
21    }
22 }
```

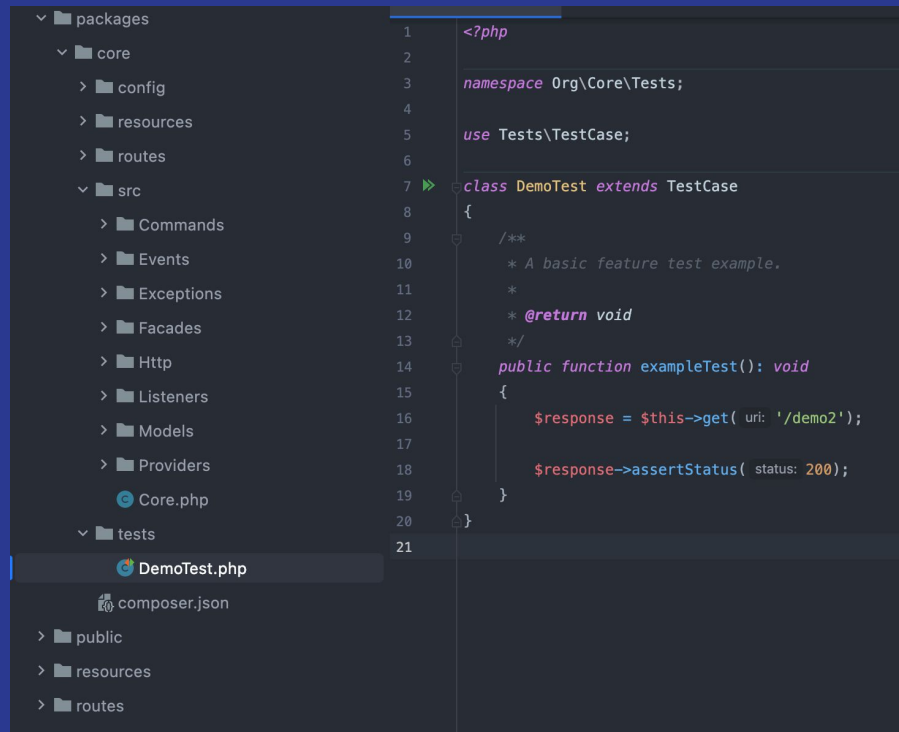


Tests

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="./vendor/phpunit/phpunit/phpunit.xsd"
  bootstrap="vendor/autoload.php"
  colors="true"
>
  <testsuites>
    <testsuite name="Unit">
      <directory suffix="Test.php">./tests/Unit</directory>
    </testsuite>
    <testsuite name="Feature">
      <directory suffix="Test.php">./tests/Feature</directory>
    </testsuite>
    <testsuite name="Core">
      <directory suffix="Test.php">./packages/**/tests</directory>
    </testsuite>
  </testsuites>
  <coverage processUncoveredFiles="true">
    <include>
      <directory suffix=".php">./app</directory>
    </include>
  </coverage>
  <php>
    <env name="APP_ENV" value="testing"/>
    <env name="BCRYPT_ROUNDS" value="4"/>
    <env name="CACHE_DRIVER" value="array"/>
    <!-- <env name="DB_CONNECTION" value="sqlite"/> -->
    <!-- <env name="DB_DATABASE" value="memory:"/> -->
    <env name="MAIL_MAILER" value="array"/>
    <env name="QUEUE_CONNECTION" value="sync"/>
    <env name="SESSION_DRIVER" value="array"/>
    <env name="TELESCOPE_ENABLED" value="false"/>
  </php>
</phpunit>
```



Tests



```
1 <?php
2
3 namespace Org\Core\Tests;
4
5 use Tests\TestCase;
6
7 class DemoTest extends TestCase
8 {
9     /**
10      * A basic feature test example.
11      *
12      * @return void
13      */
14     public function exampleTest(): void
15     {
16         $response = $this->get( uri: '/demo2');
17
18         $response->assertStatus( status: 200);
19     }
20 }
21
```



3. Ưu và nhược điểm



Ưu điểm

1. Tái sử dụng
2. Khả năng mở rộng
3. Dễ quản lý code

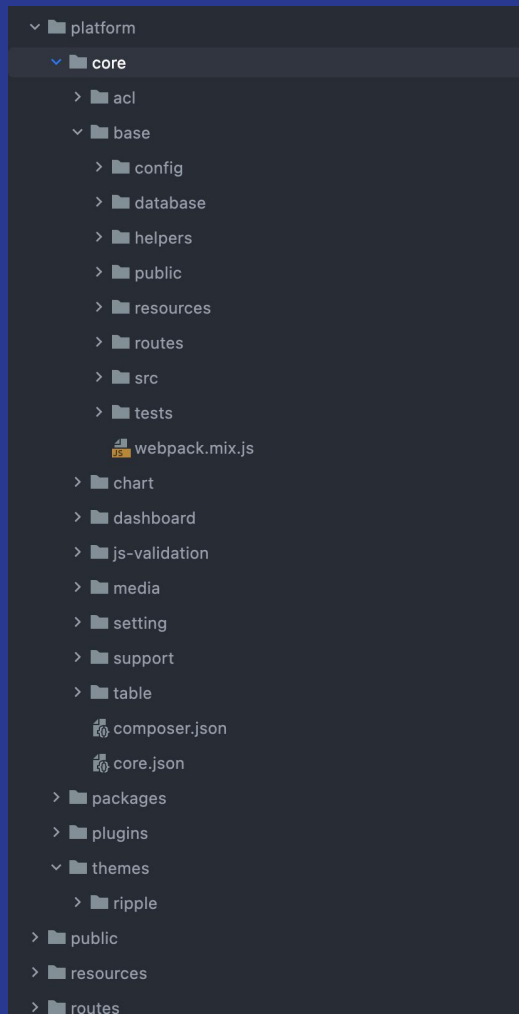


Nhược điểm

1. Khó tiếp cận cho người mới bắt đầu
2. Cần kinh nghiệm để phân chia tính năng một cách hợp lý



4. Demo thực tế





- ▼ platform
 - > core
 - ▼ packages
 - > api
 - > dev-tool
 - > menu
 - > optimize
 - > page
 - > plugin-generator
 - > plugin-management
 - > revision
 - > seo-helper
 - > shortcode
 - > sitemap
 - > slug
 - > theme
 - > theme-generator
 - > widget
 - > widget-generator
 - > plugins
 - > **themes**
 - > public
 - > resources
 - > routes
 - > storage
 - > tests



- ▼ platform
 - > core
 - > packages
 - ▼ **plugins**
 - > analytics
 - > audit-log
 - > backup
 - > block
 - > blog
 - > captcha
 - > contact
 - > cookie-consent
 - > custom-field
 - > faq
 - > gallery
 - > language
 - > language-advanced
 - > location
 - > member
 - > newsletter
 - > note
 - > request-log
 - > rss-feed
 - > simple-slider
 - > social-login
 - > testimonial
 - > translation
 - > themes
- > public
- > resources



- ▼ platform
 - > core
 - > packages
 - > plugins
 - ▼ themes
 - ▼ ripple
 - > assets
 - > functions
 - > lang
 - > layouts
 - > partials
 - > public
 - > routes
 - > src
 - > views
 - > widgets
 - config.php
 - screenshot.png
 - theme.json
 - webpack.mix.js
 - > public
 - > resources
 - > routes
 - > storage



5. Hỏi đáp



<https://github.com/sangnguyenplus/laravel-demo>