Xây dựng ứng dụng Laravel

Theo dạng PACKAGES

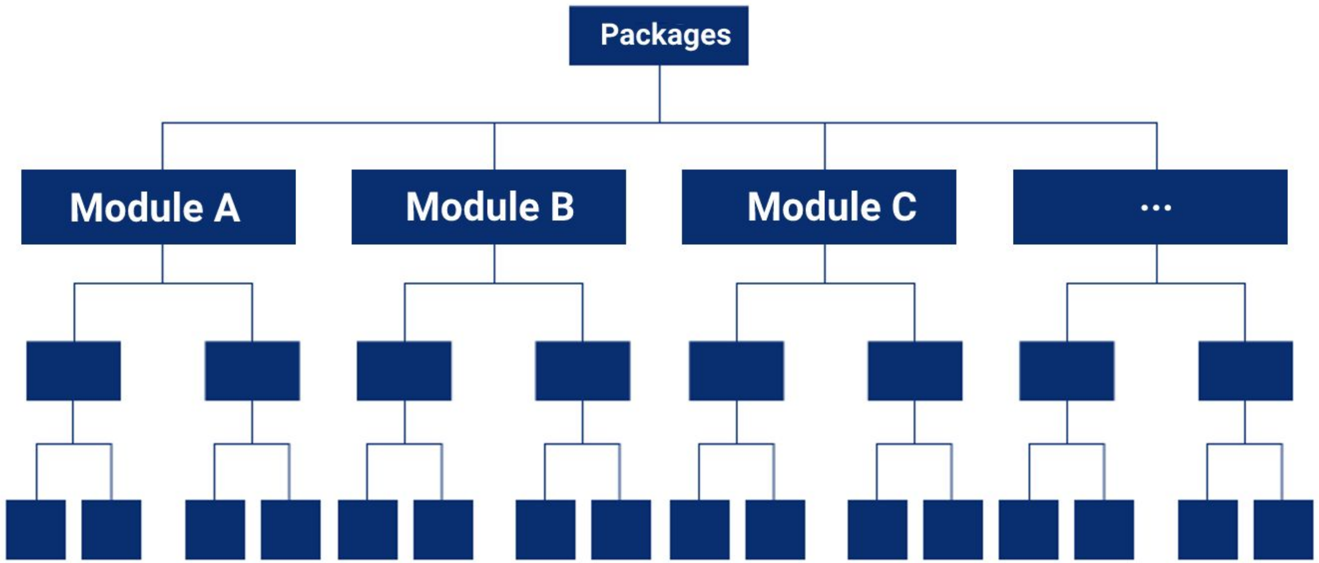
Nguyễn Minh Sang - Creator of Botble

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



Packages

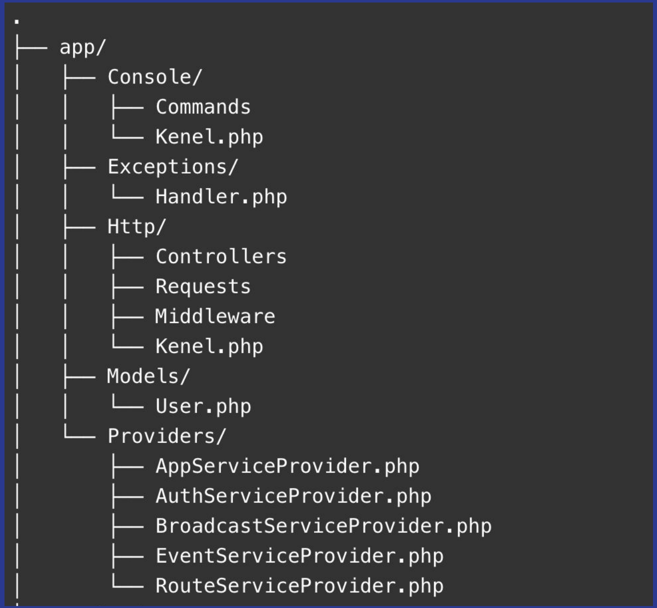
Module A

Module B

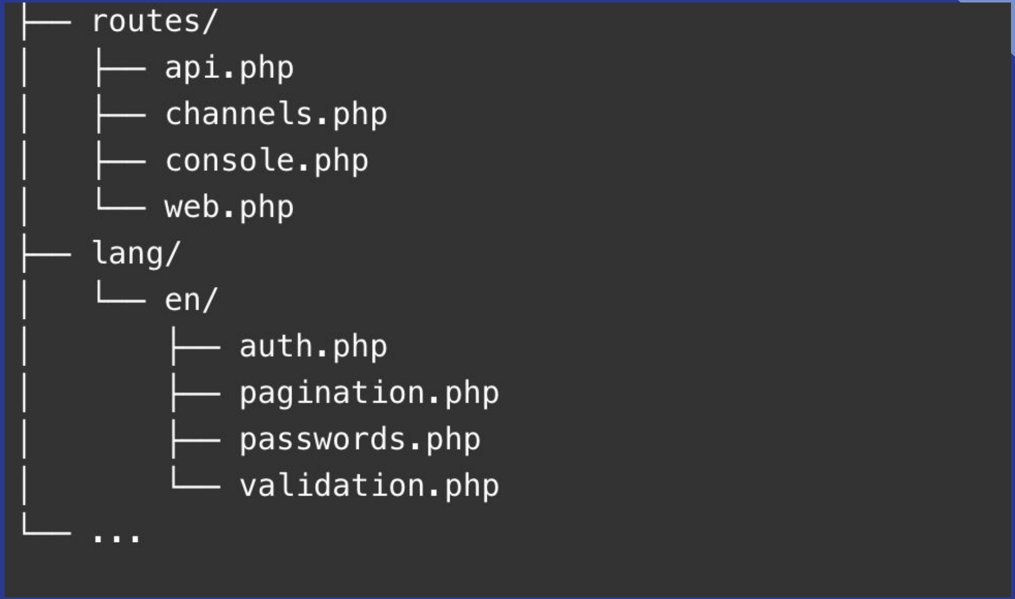
Module C

...

Cấu trúc mặc định của Laravel

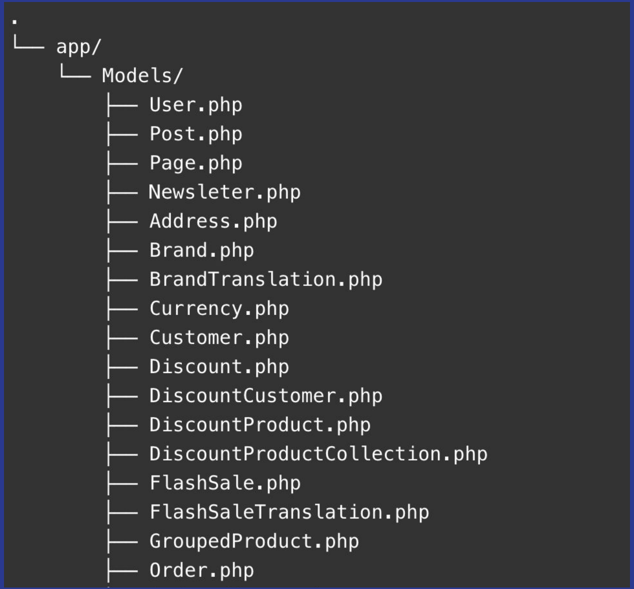


```
.
├── app/
│   ├── Console/
│   │   ├── Commands
│   │   └── Kenel.php
│   ├── Exceptions/
│   │   └── Handler.php
│   ├── Http/
│   │   ├── Controllers
│   │   ├── Requests
│   │   ├── Middleware
│   │   └── Kenel.php
│   ├── Models/
│   │   └── User.php
│   └── Providers/
│       ├── AppServiceProvider.php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

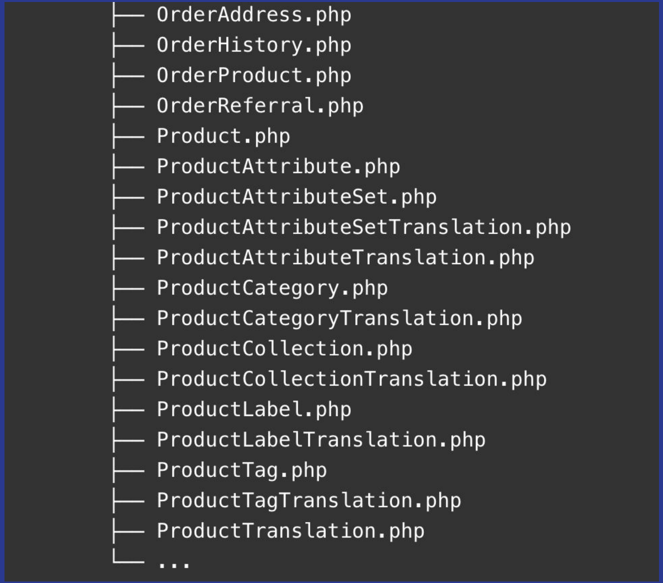


```
├── routes/
│   ├── api.php
│   ├── channels.php
│   ├── console.php
│   └── web.php
├── lang/
│   └── en/
│       ├── auth.php
│       ├── pagination.php
│       ├── passwords.php
│       └── validation.php
└── ...
```

Khó quản lý khi ứng dụng lớn dần

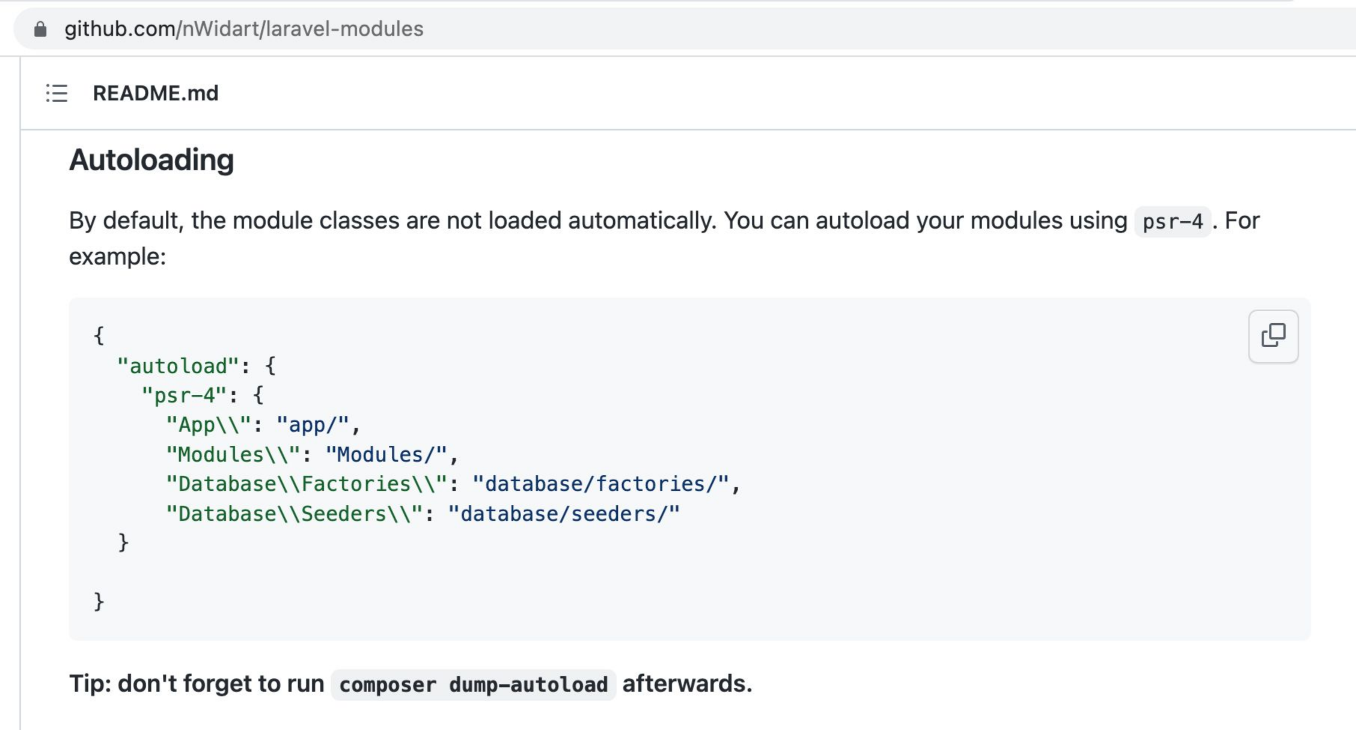


```
.
└── app/
    └── Models/
        ├── User.php
        ├── Post.php
        ├── Page.php
        ├── Newsleter.php
        ├── Address.php
        ├── Brand.php
        ├── BrandTranslation.php
        ├── Currency.php
        ├── Customer.php
        ├── Discount.php
        ├── DiscountCustomer.php
        ├── 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

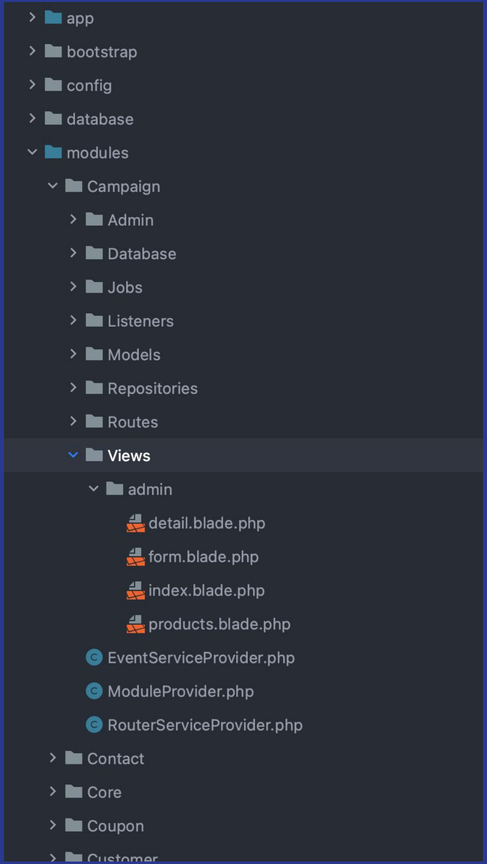
Autoloading

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/"
  }

}
```

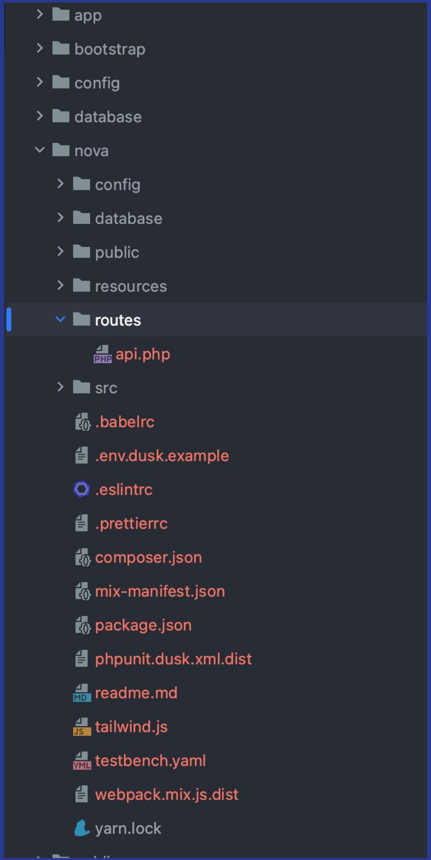
Tip: don't forget to run `composer dump-autoload` afterwards.

Một số dạng cấu trúc



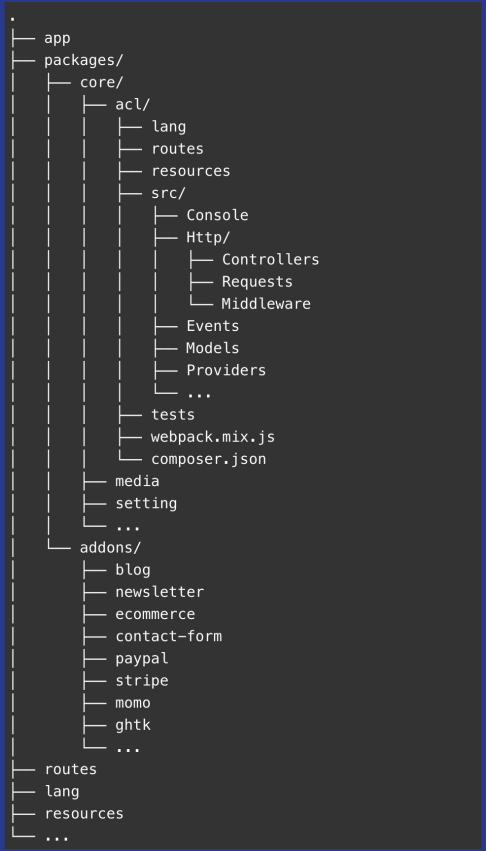
```
> app
> bootstrap
> config
> database
v modules
    v Campaign
        > Admin
        > Database
        > Jobs
        > Listeners
        > Models
        > Repositories
        > Routes
        v Views
            v admin
                detail.blade.php
                form.blade.php
                index.blade.php
                products.blade.php
        EventServiceProvider.php
        ModuleProvider.php
        RouterServiceProvider.php
    > Contact
    > Core
    > Coupon
    > Customer
```

Một số dạng cấu trúc



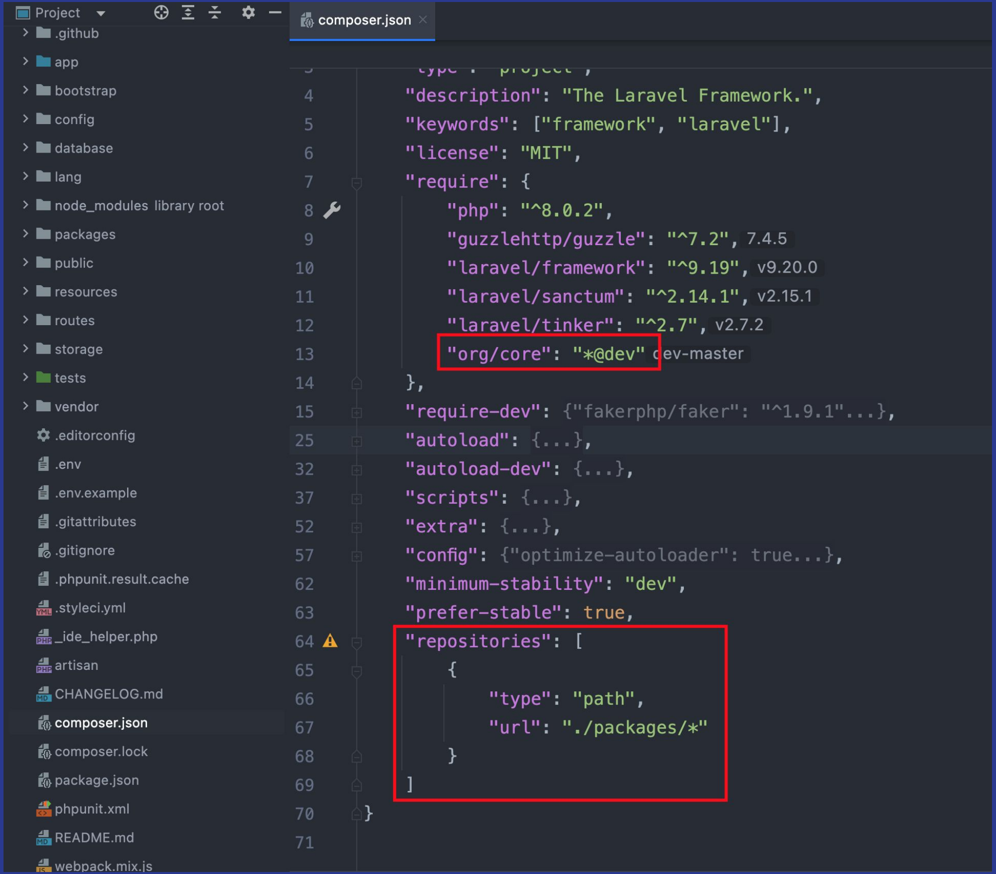
```
> app
> bootstrap
> config
> database
v nova
    > config
    > database
    > public
    > resources
    v routes
        api.php
    > src
    .babelrc
    .env.dusk.example
    .eslintrc
    .prettierrc
    composer.json
    mix-manifest.json
    package.json
    phpunit.dusk.xml.dist
    readme.md
    tailwind.js
    testbench.yaml
    webpack.mix.js.dist
    yarn.lock
```

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
└── ...
```

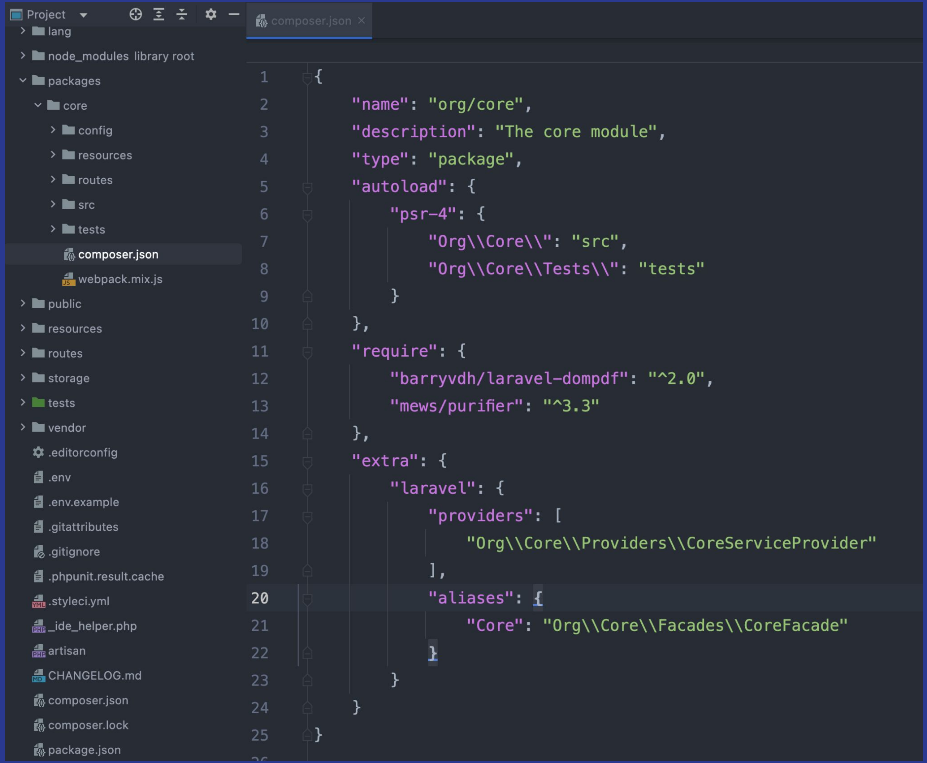
Composer.json



```
> .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
```

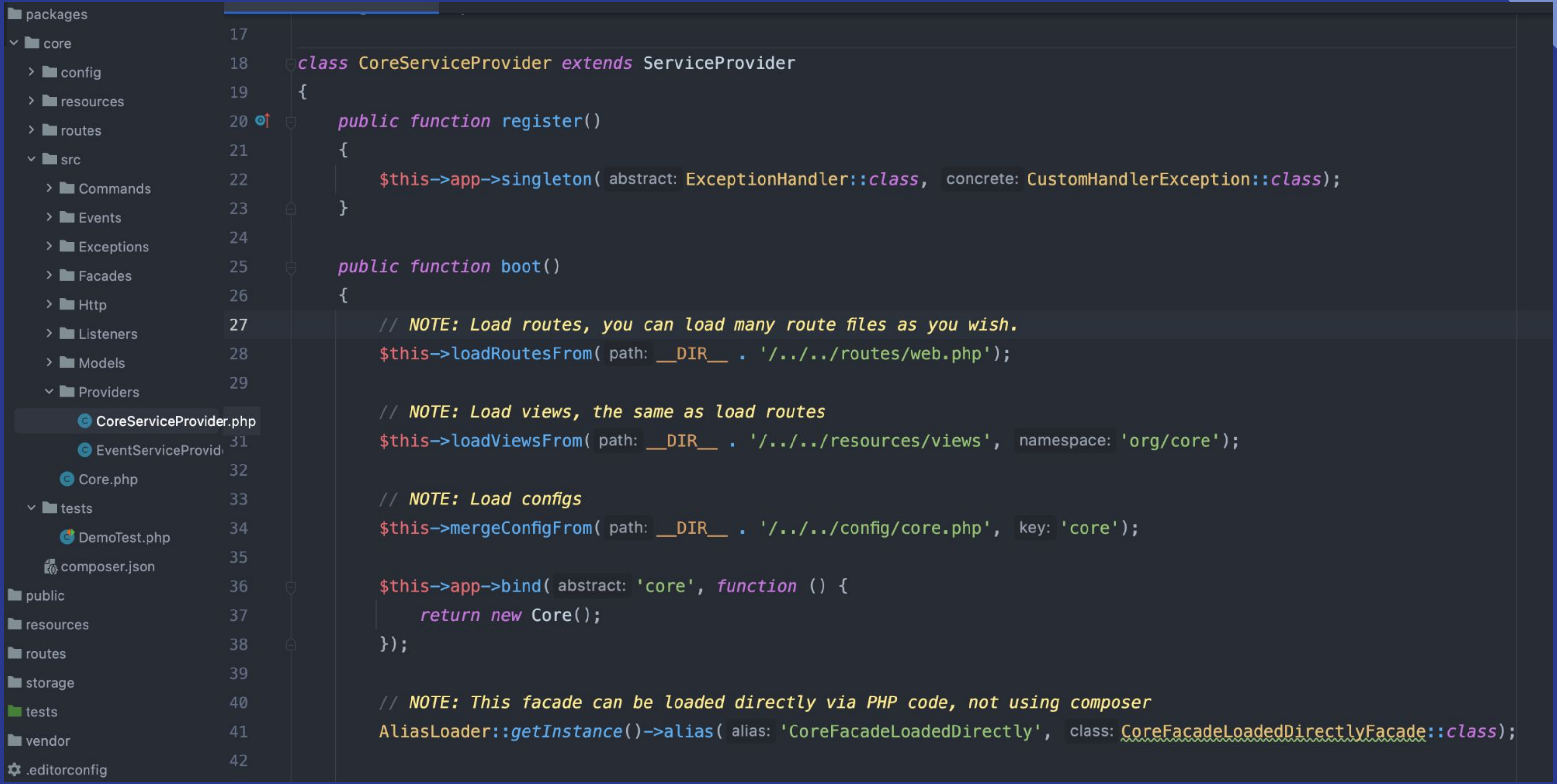
```
    "type": "project",
    "description": "The Laravel Framework.",
    "keywords": ["framework", "laravel"],
    "license": "MIT",
    "require": {
        "php": "^8.0.2",
        "guzzlehttp/guzzle": "^7.2", 7.4.5
        "laravel/framework": "^9.19", v9.20.0
        "laravel/sanctum": "^2.14.1", v2.15.1
        "laravel/tinker": "^2.7", v2.7.2
        "org/core": "*@dev" dev-master
    },
    "require-dev": {"fakerphp/faker": "^1.9.1"...},
    "autoload": {...},
    "autoload-dev": {...},
    "scripts": {...},
    "extra": {...},
    "config": {"optimize-autoloader": true...},
    "minimum-stability": "dev",
    "prefer-stable": true,
    "repositories": [
        {
            "type": "path",
            "url": "./packages/*"
        }
    ]
}
```

Composer.json



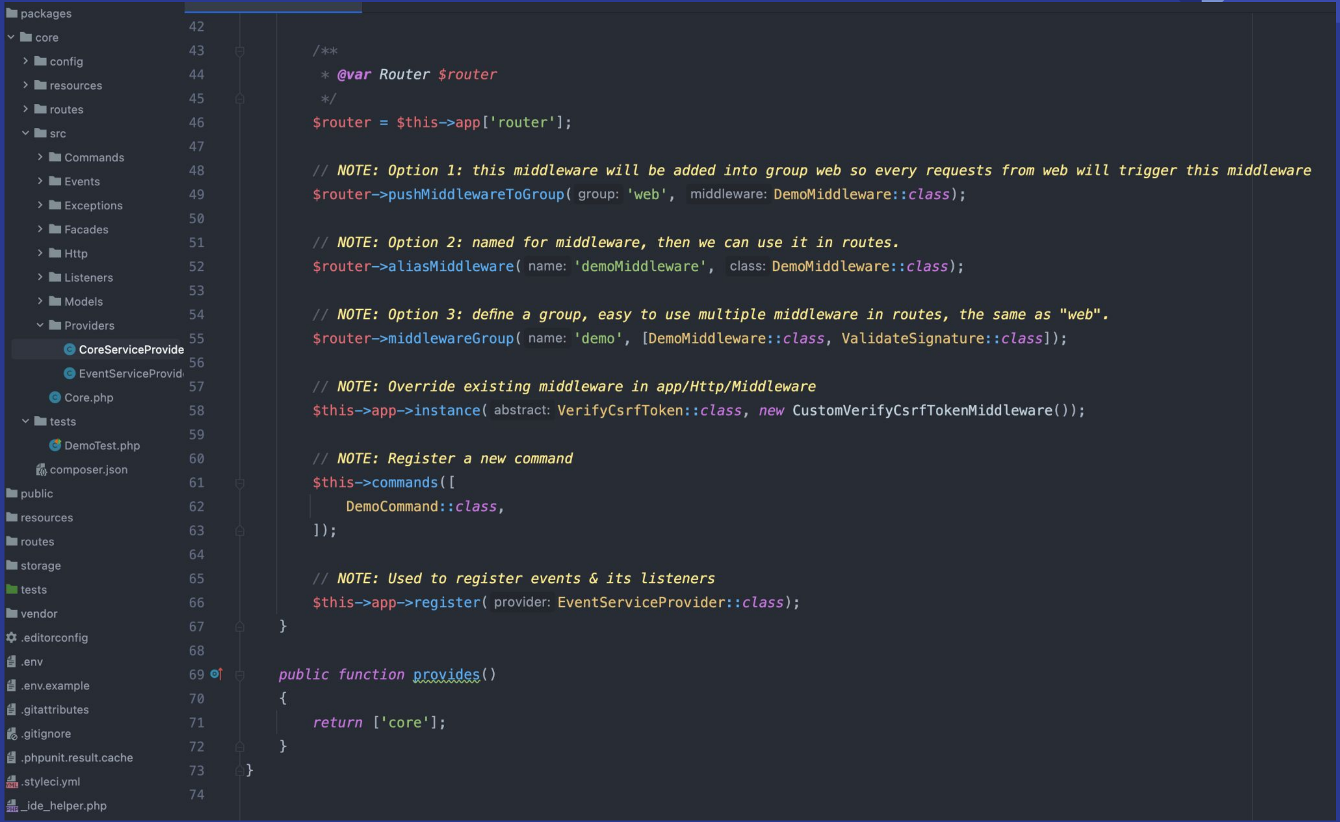
```json
{
    "name": "org/core",
    "description": "The core module",
    "type": "package",
    "autoload": {
        "psr-4": {
            "Org\\Core\\": "src",
            "Org\\Core\\Tests\\": "tests"
        }
    },
    "require": {
        "barryvdh/laravel-dompdf": "^2.0",
        "mews/purifier": "^3.3"
    },
    "extra": {
        "laravel": {
            "providers": [
                "Org\\Core\\Providers\\CoreServiceProvider"
            ],
            "aliases": {
                "Core": "Org\\Core\\Facades\\CoreFacade"
            }
        }
    }
}
```

Provider



```php

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);

```

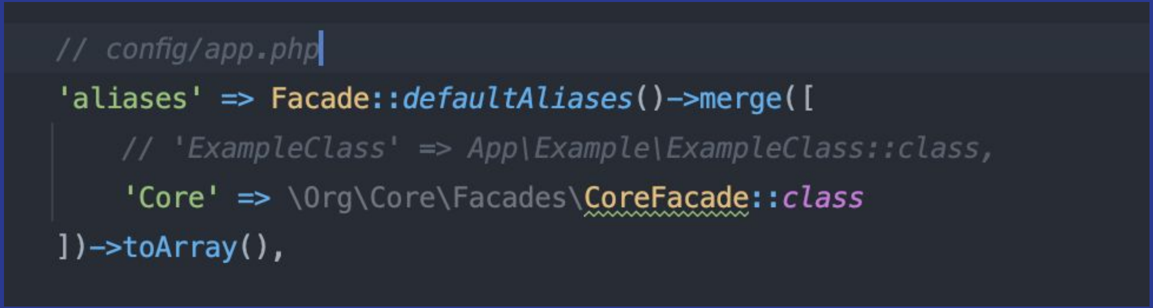
Provider



```php

        /**
         * @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'];
    }
}

```

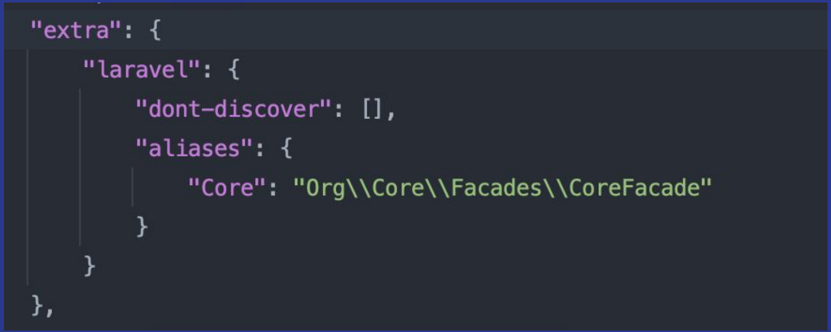
Facades

Cách thông thường



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

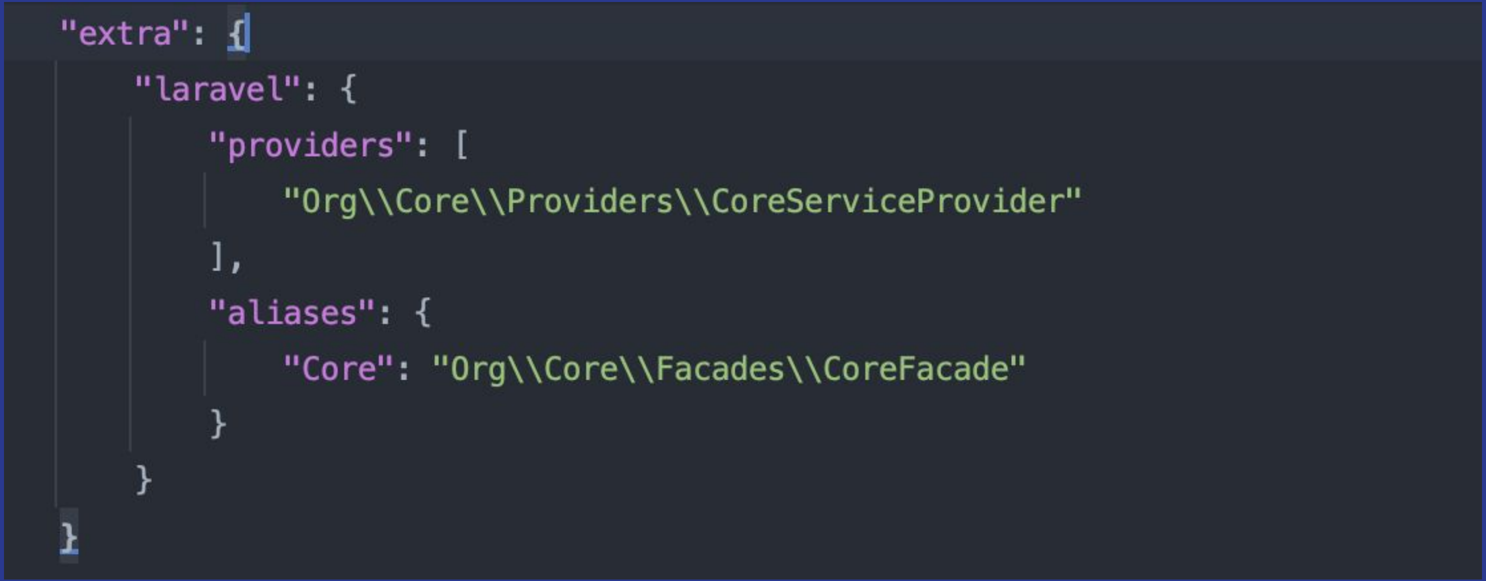
Hoặc



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

Facades

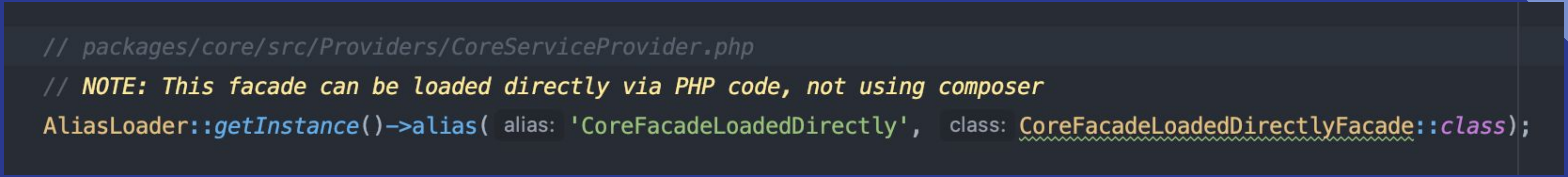
packages/core/composer.json



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

Facades

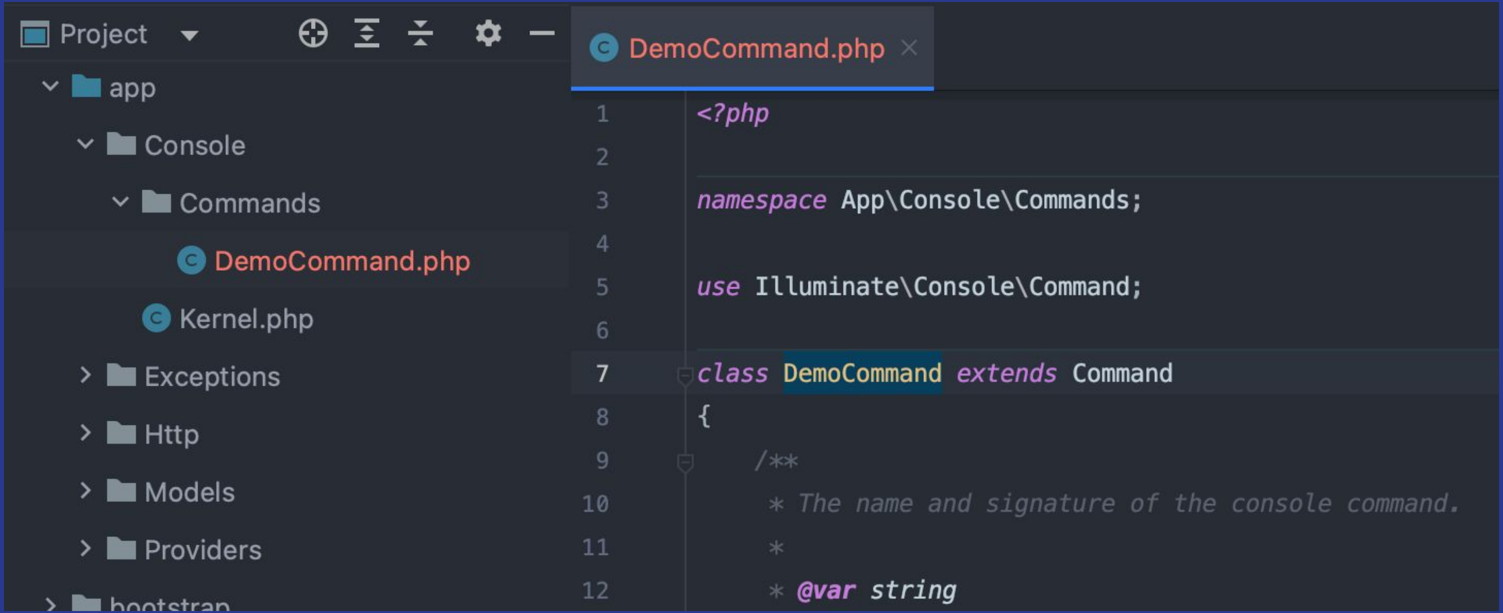
Hoặc dùng AliasLoader



```
// 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

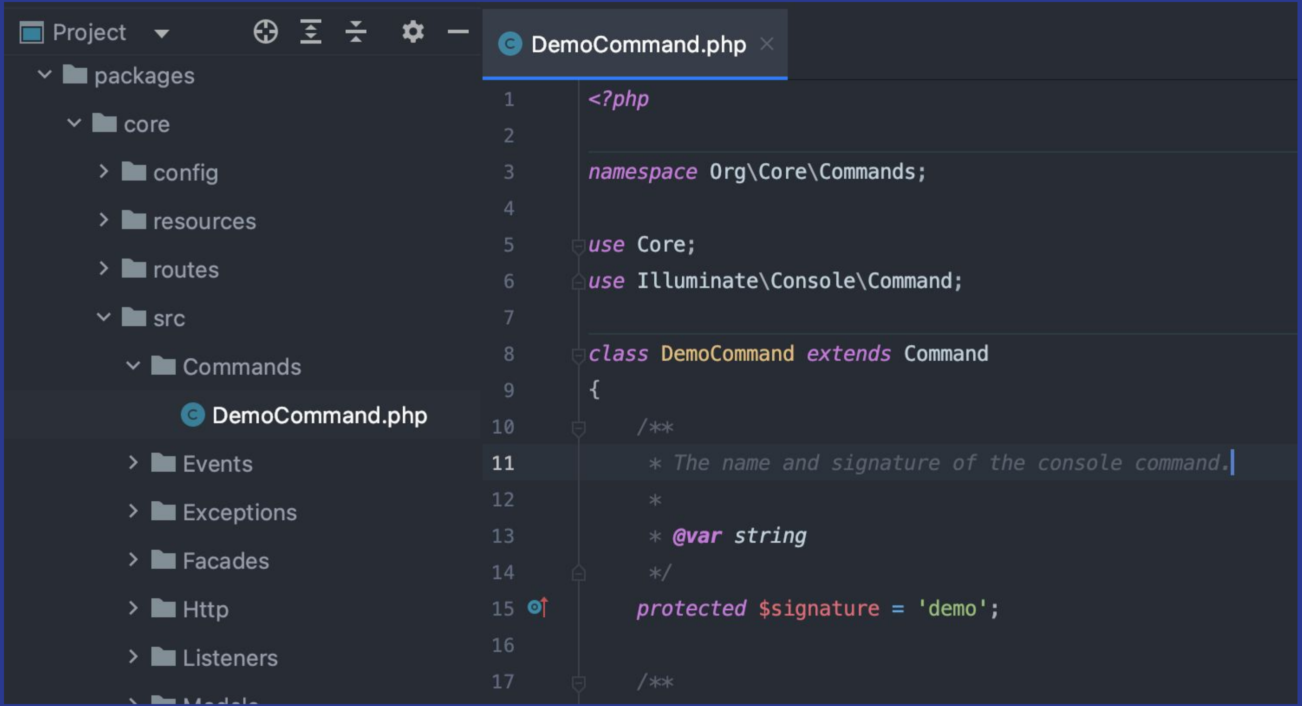
Cách thông thường



```php
<?php

namespace App\Console\Commands;

use Illuminate\Console\Command;

class DemoCommand extends Command
{
    /**
     * The name and signature of the console command.
     *
     * @var string
```

Commands

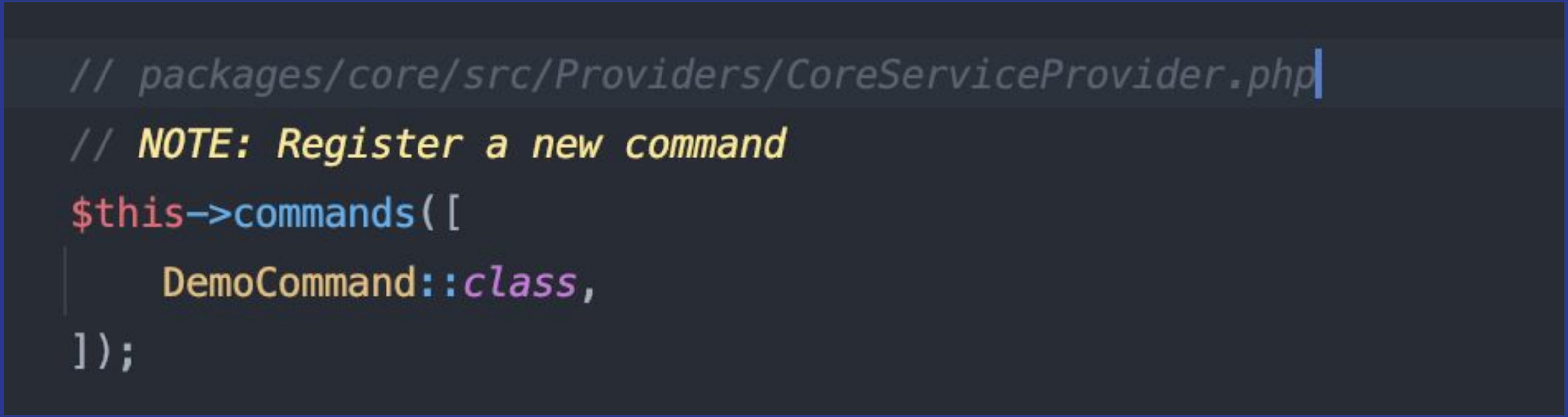
Theo dạng packages



```php
<?php

namespace Org\Core\Commands;

use Core;
use Illuminate\Console\Command;

class DemoCommand extends Command
{
    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'demo';

    /**
```

Commands

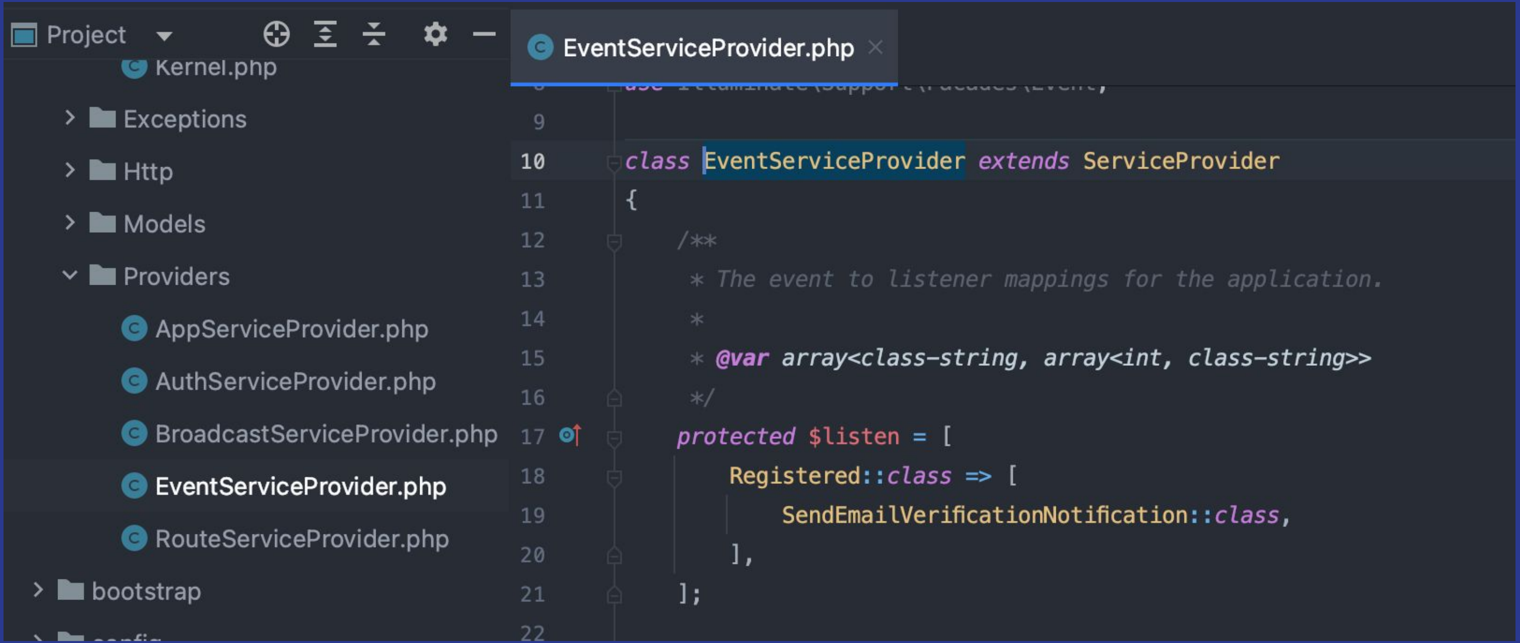
Khai báo command



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

Events

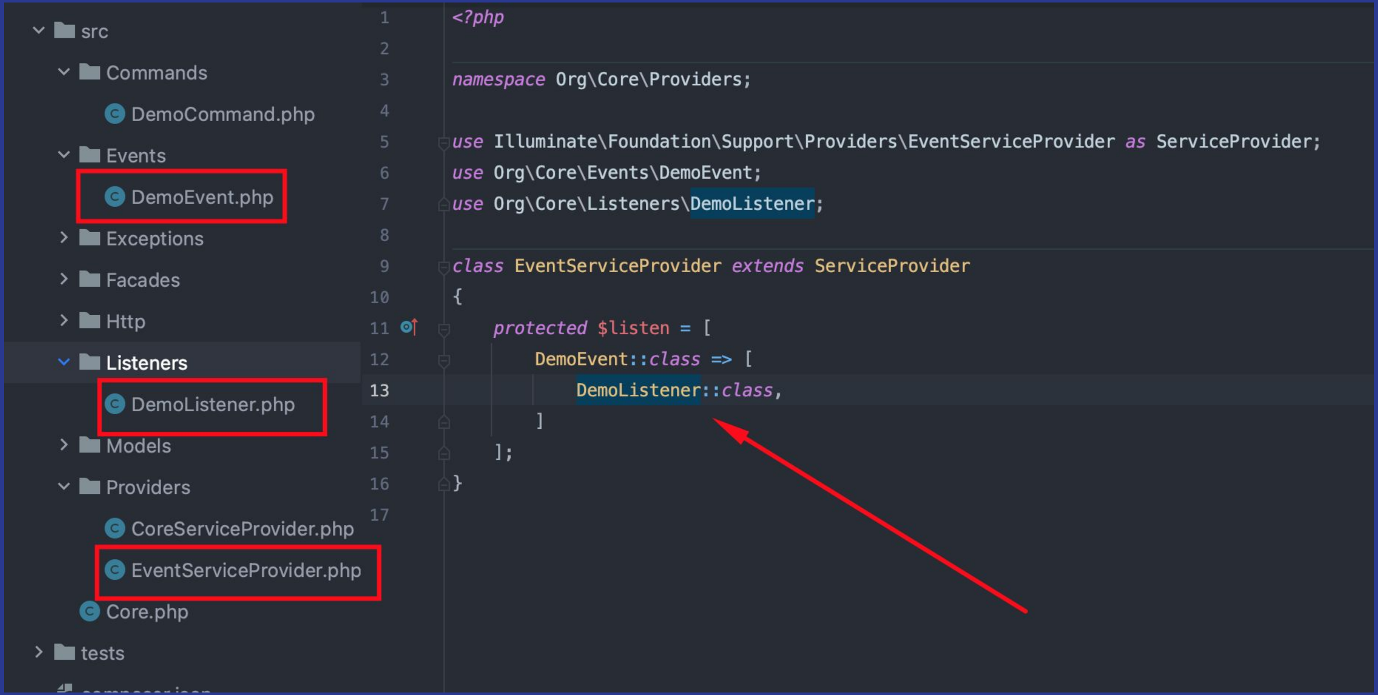
Cách thông thường



```php
use Illuminate\Support\Facades\Event;

class EventServiceProvider extends ServiceProvider
{
    /**
     * The event to listener mappings for the application.
     *
     * @var array<class-string, array<int, class-string>>
     */
    protected $listen = [
        Registered::class => [
            SendEmailVerificationNotification::class,
        ],
    ];

```

Events

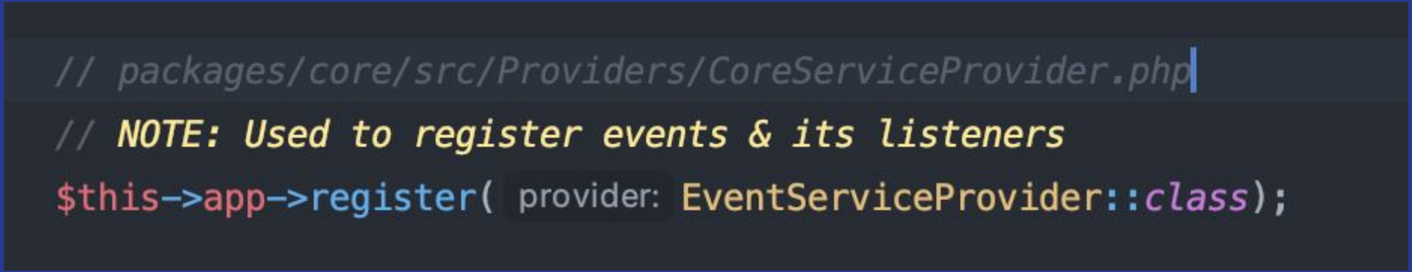
Theo dạng packages



```php
<?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,
        ]
    ];
}
```

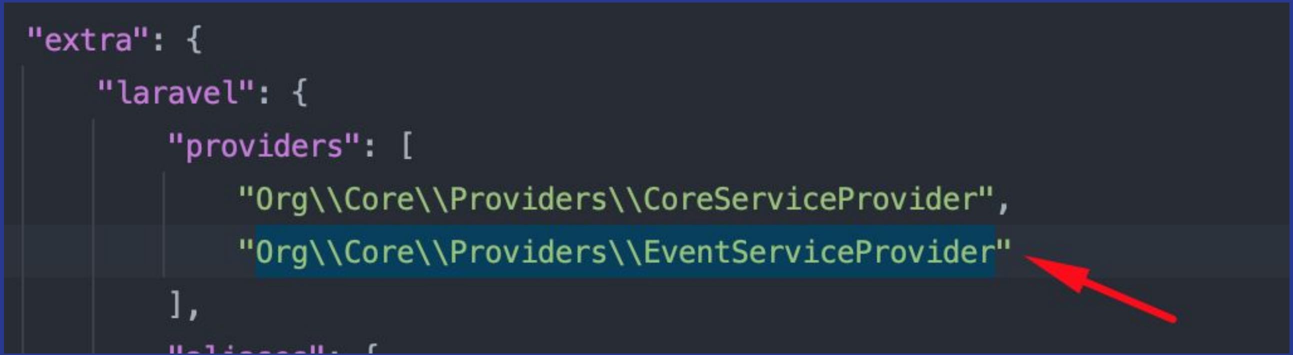
Events

Đăng ký EventServiceProvider



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

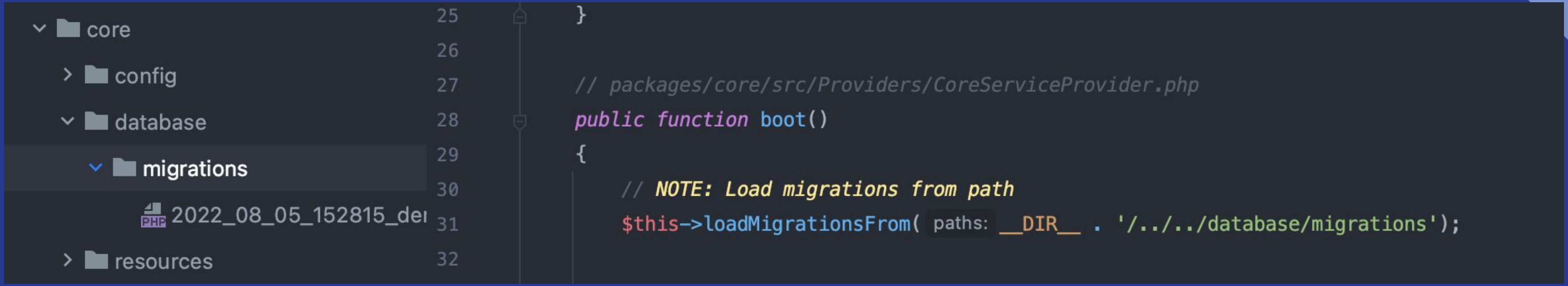
Hoặc trong composer.json



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

Migrations

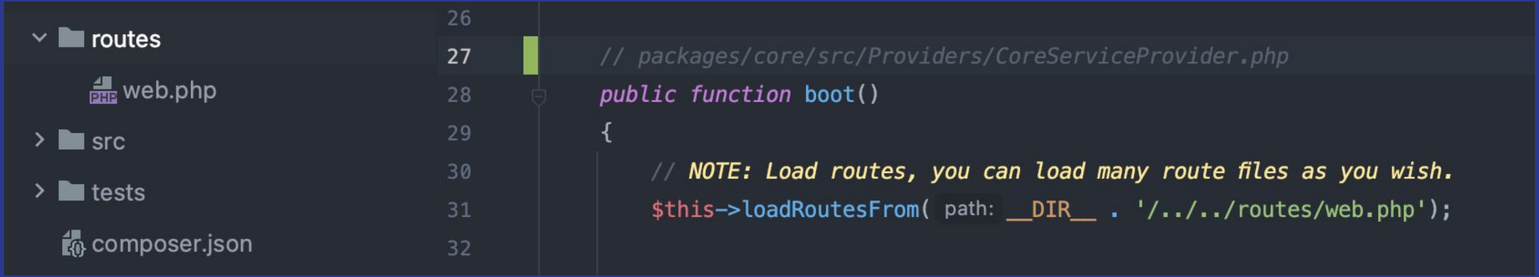
Load migrations



```php
}

// packages/core/src/Providers/CoreServiceProvider.php
public function boot()
{
    // NOTE: Load migrations from path
    $this->loadMigrationsFrom( paths: __DIR__ . '/../../database/migrations');

```

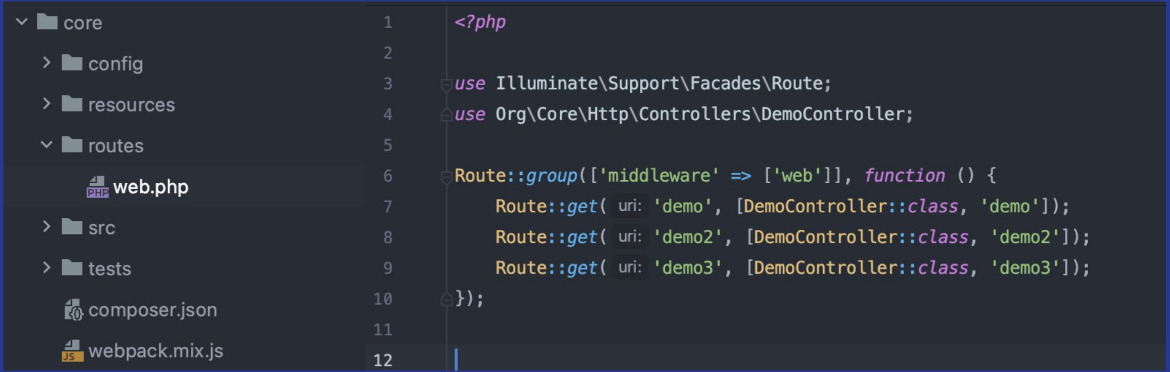
Routes

Load routes



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

```

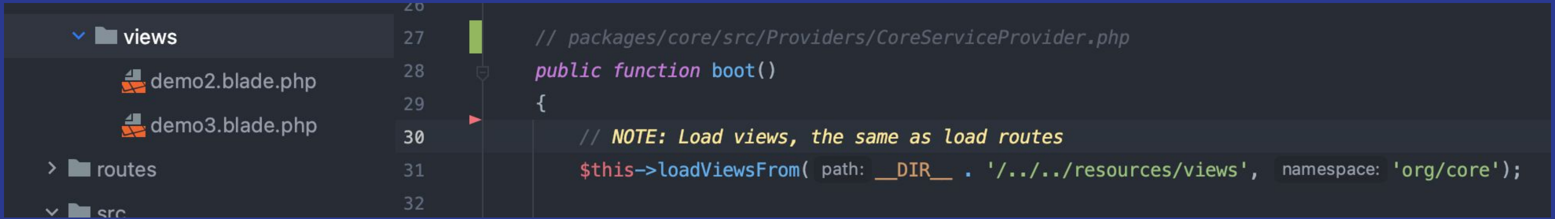
Khai báo routes



```
<?php

use Illuminate\Support\Facades\Route;
use Org\Core\Http\Controllers\DemoController;

Route::group(['middleware' => ['web']], function () {
    Route::get( uri: 'demo', [DemoController::class, 'demo']);
    Route::get( uri: 'demo2', [DemoController::class, 'demo2']);
    Route::get( uri: 'demo3', [DemoController::class, 'demo3']);
});

```

Views

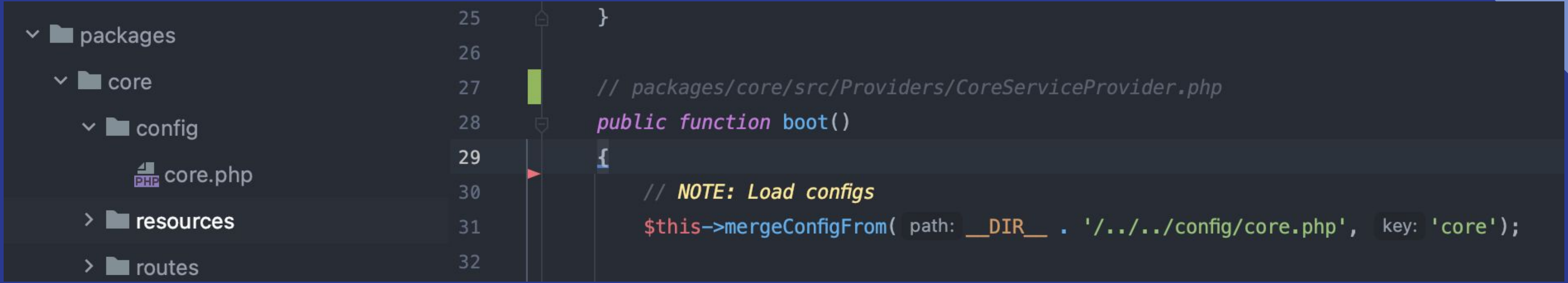
Load views



```php
// packages/core/src/Providers/CoreServiceProvider.php
public function boot()
{
    // NOTE: Load views, the same as load routes
    $this->loadViewsFrom( path: __DIR__ . '/../../resources/views',  namespace: 'org/core');
```

Config

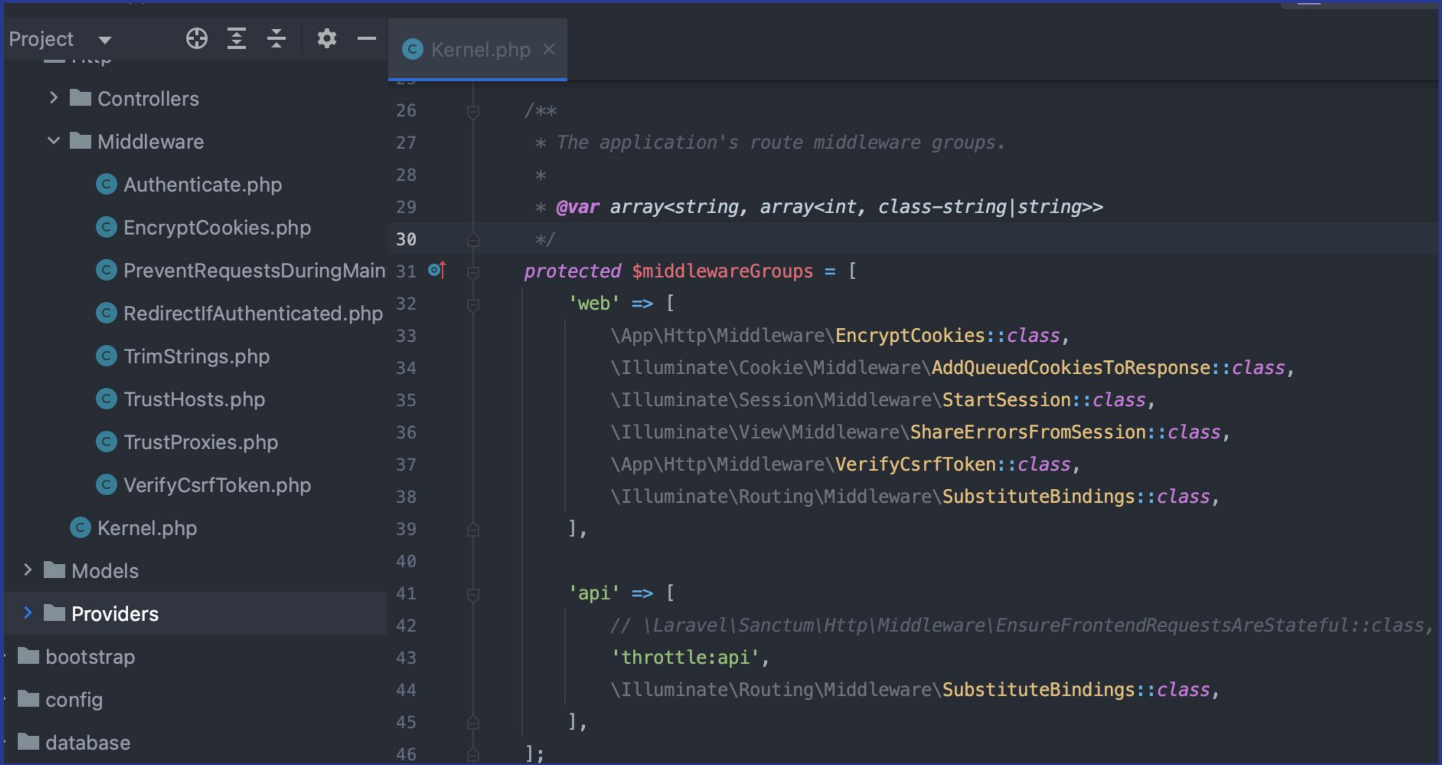
Load config



```php
}

// packages/core/src/Providers/CoreServiceProvider.php
public function boot()
{
    // NOTE: Load configs
    $this->mergeConfigFrom( path: __DIR__ . '/../../config/core.php',  key: 'core');
```

Middleware

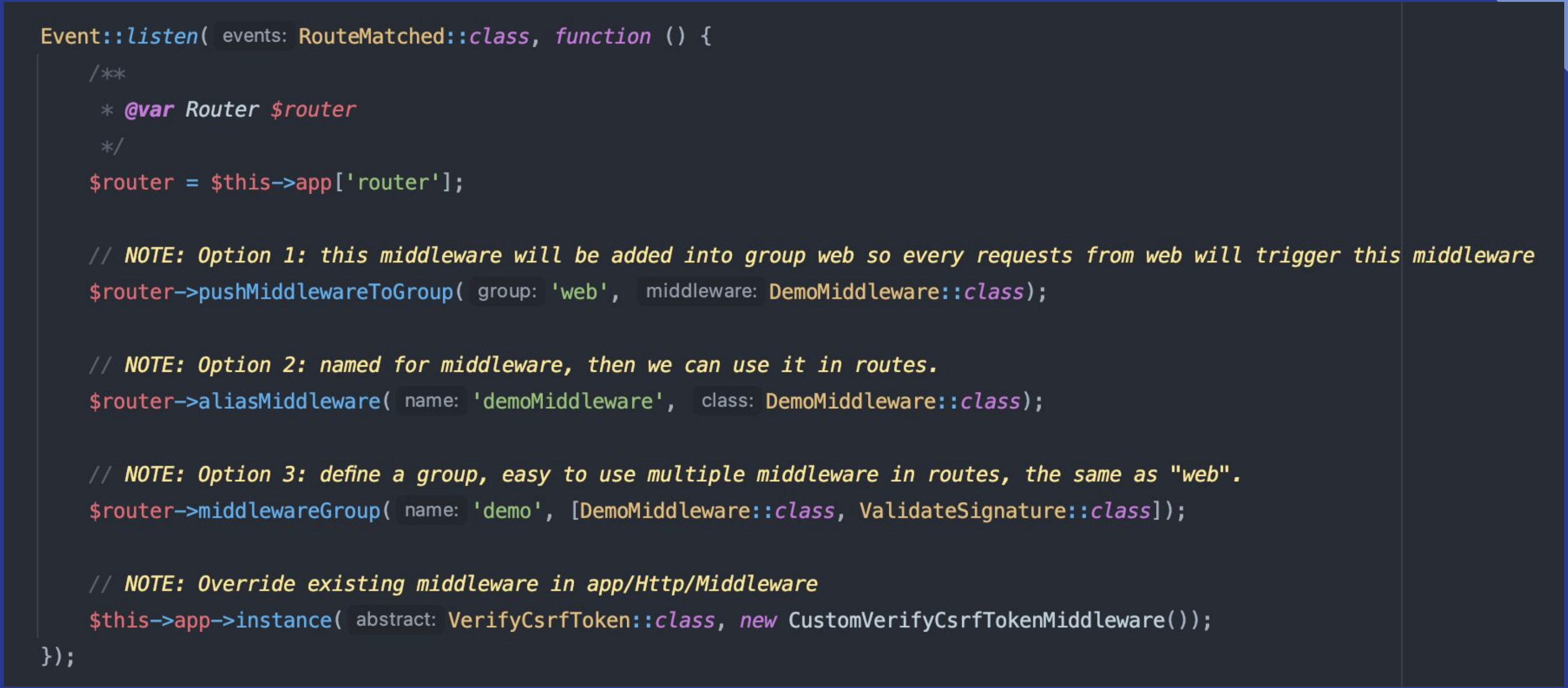
Cách thông thường



```php
    /**
     * The application's route middleware groups.
     *
     * @var array<string, array<int, class-string|string>>
     */
    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' => [
            // \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
            'throttle:api',
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],
    ];
```

Middleware

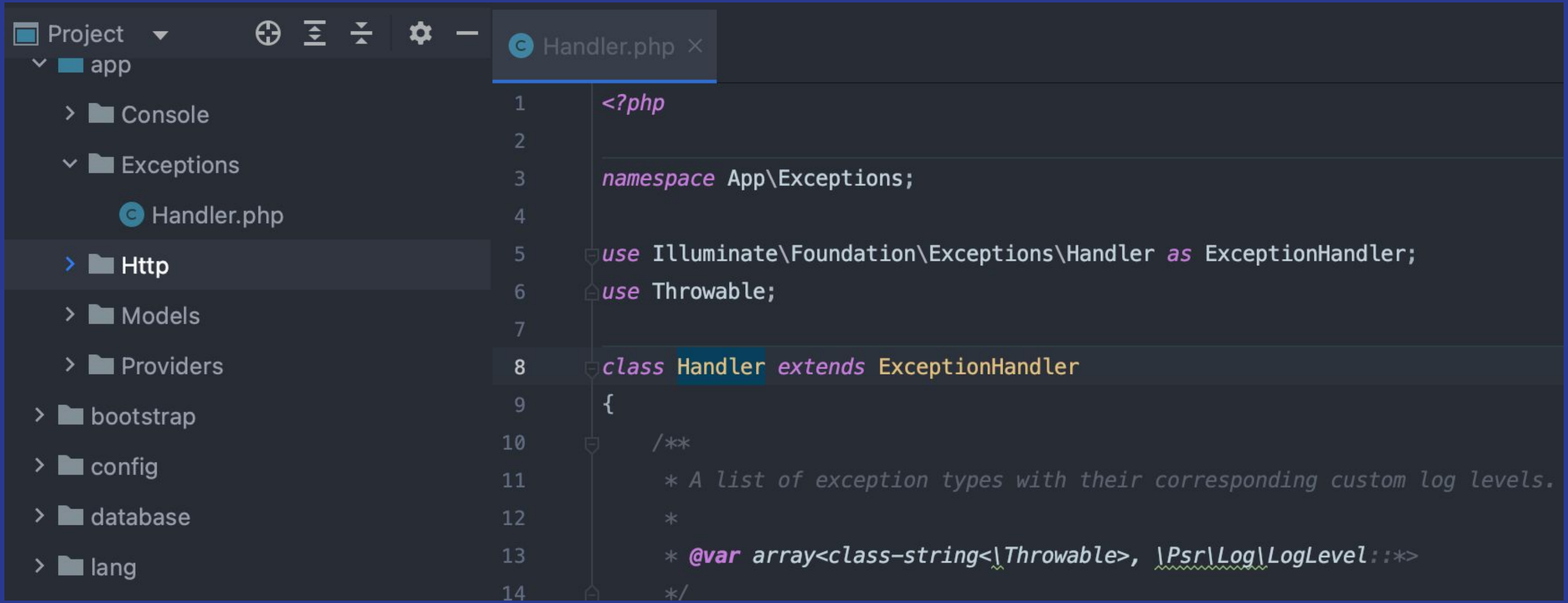
Dạng packages



```
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

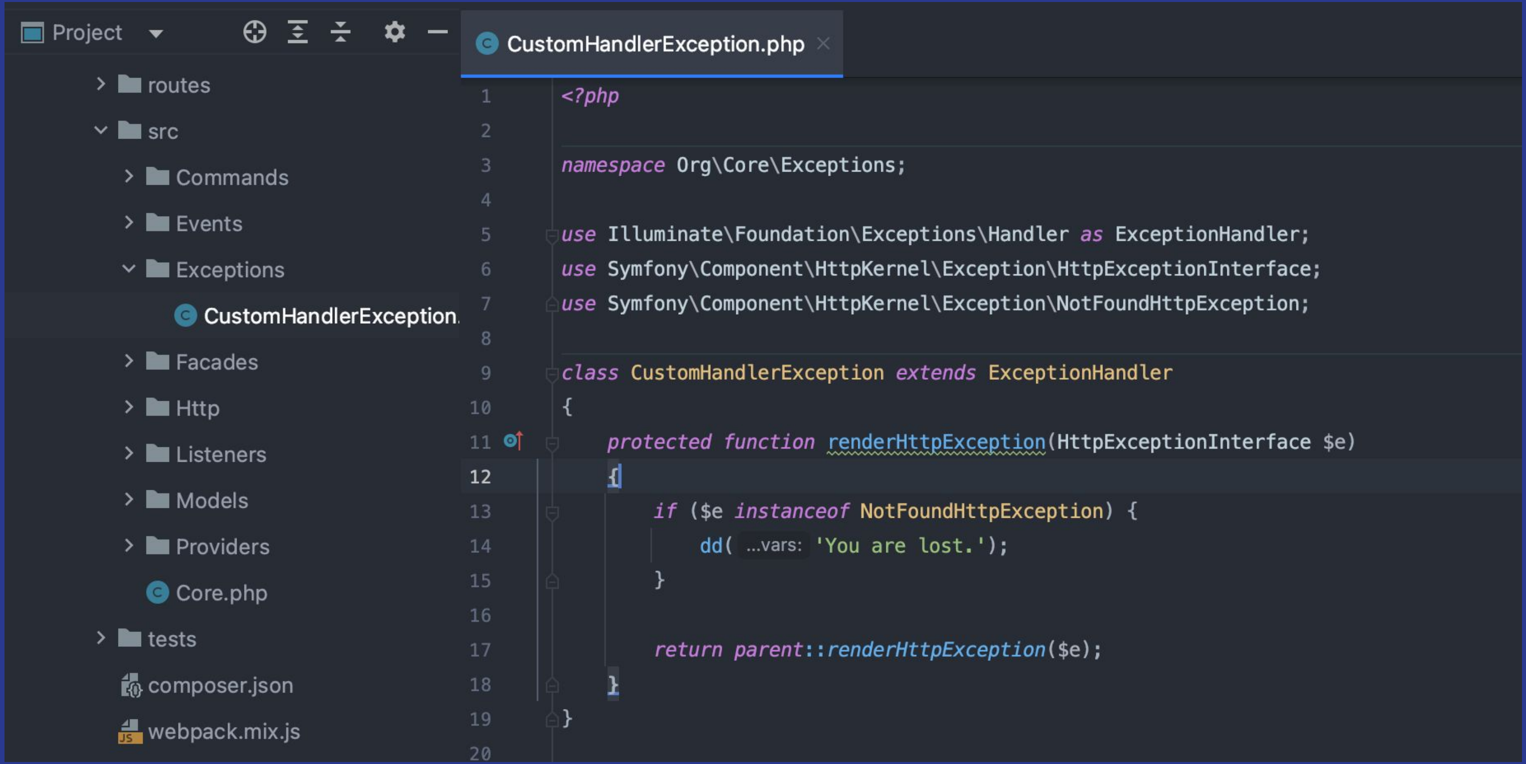
Dạng thông thường



```php
<?php

namespace App\Exceptions;

use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
use Throwable;

class Handler extends ExceptionHandler
{
    /**
     * A list of exception types with their corresponding custom log levels.
     *
     * @var array<class-string<\Throwable>, \Psr\Log\LogLevel::*>
     */
```

Handler.php

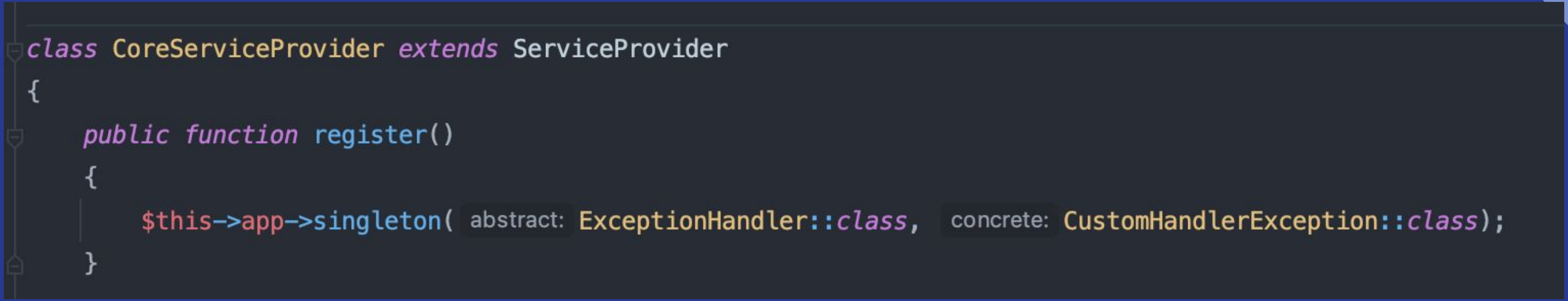
Dạng packages



```php
<?php

namespace Org\Core\Exceptions;

use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
use Symfony\Component\HttpKernel\Exception\HttpExceptionInterface;
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;

class CustomHandlerException extends ExceptionHandler
{
    protected function renderHttpException(HttpExceptionInterface $e)
    {
        if ($e instanceof NotFoundHttpException) {
            dd( ...vars: 'You are lost.');
        }

        return parent::renderHttpException($e);
    }
}
```

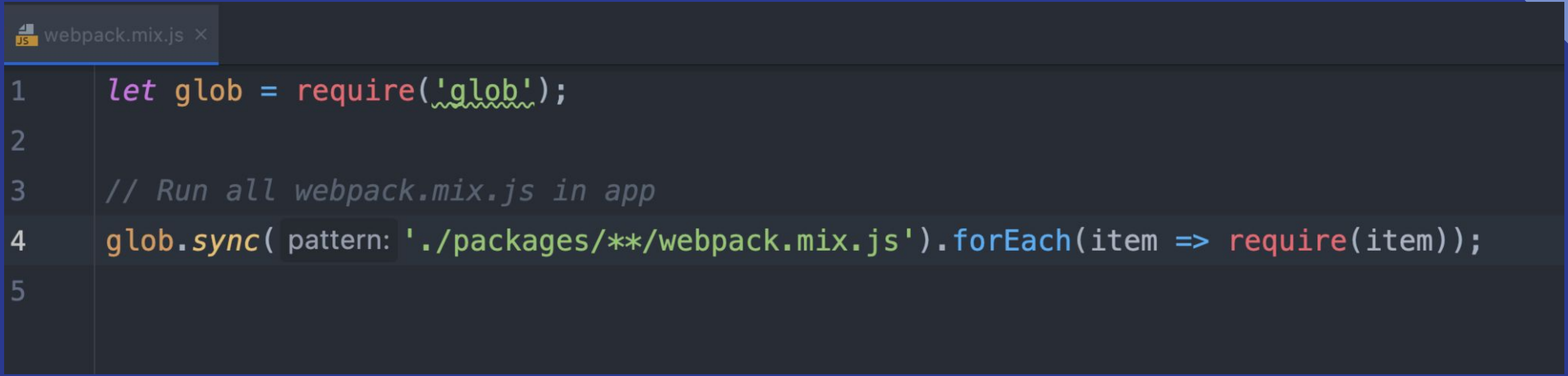
Handler.php

Khai báo



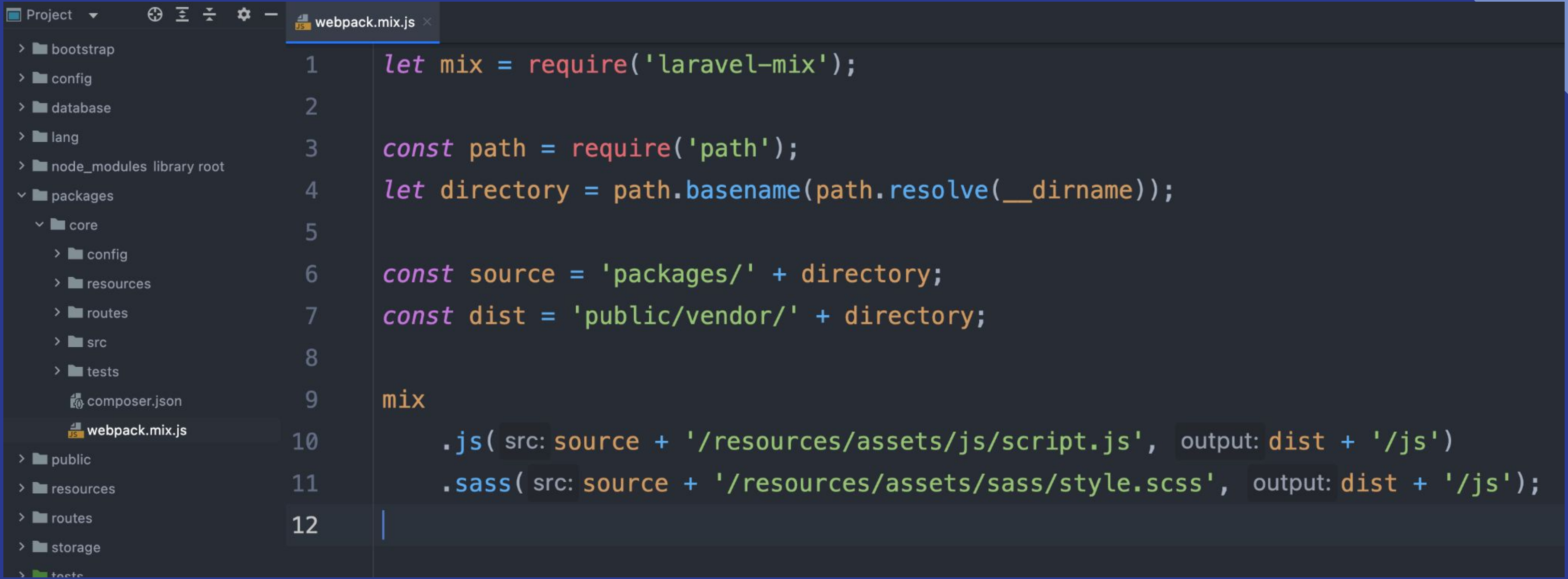
```php
class CoreServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->singleton( abstract: ExceptionHandler::class,  concrete: CustomHandlerException::class);
    }
```

Webpack



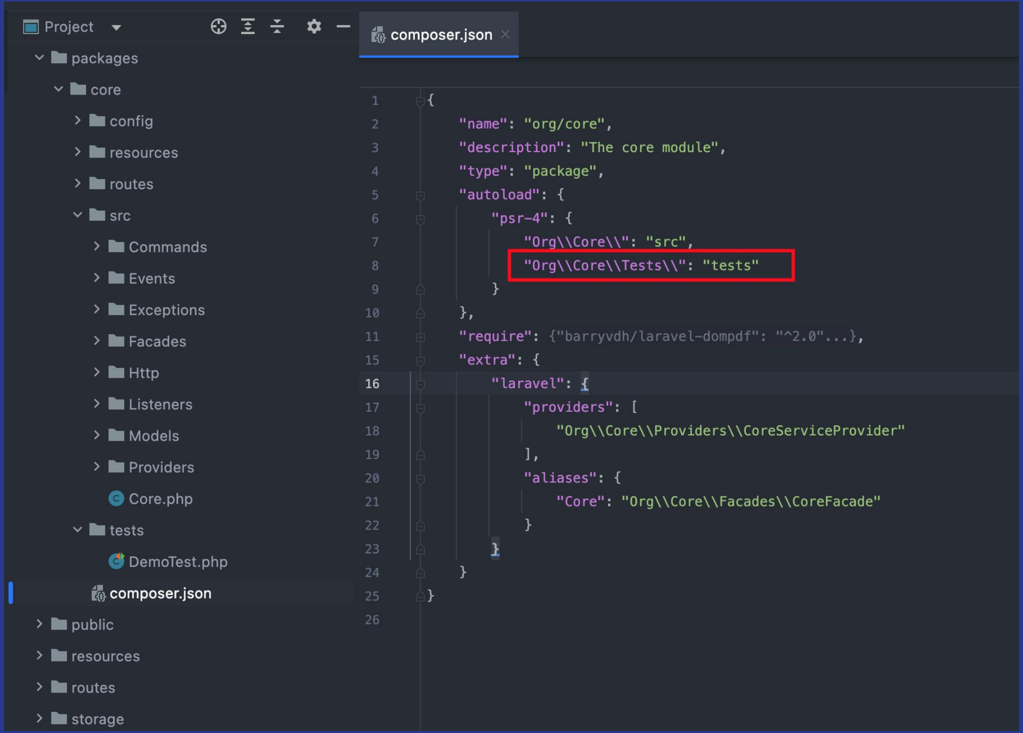
```js
let glob = require('glob');

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

Webpack



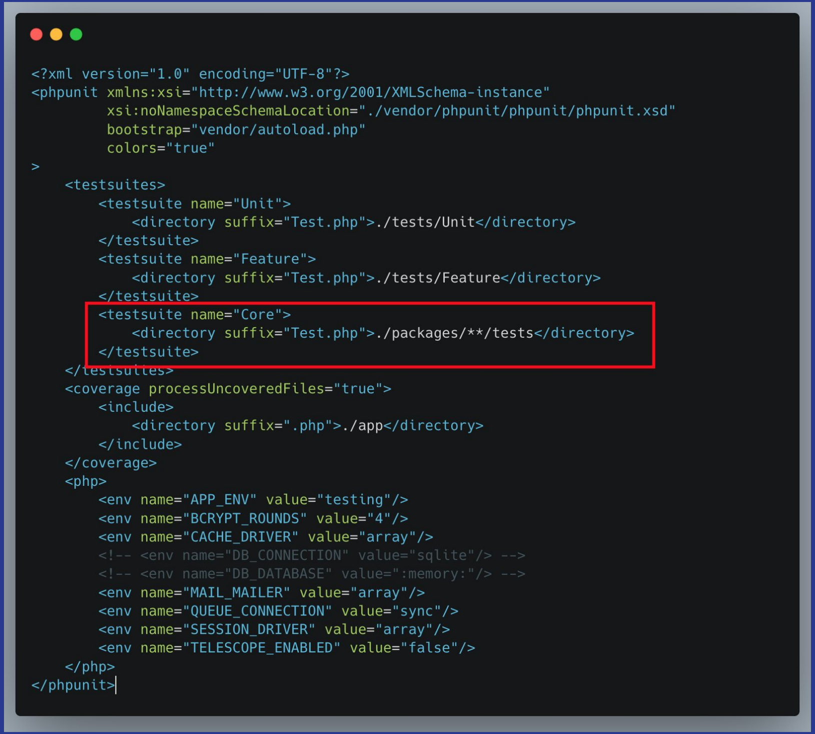
```
let mix = require('laravel-mix');

const path = require('path');
let directory = path.basename(path.resolve(__dirname));

const source = 'packages/' + directory;
const dist = 'public/vendor/' + directory;

mix
    .js( src: source + '/resources/assets/js/script.js',  output: dist + '/js')
    .sass( src: source + '/resources/assets/sass/style.scss',  output: dist + '/js');

```

Tests



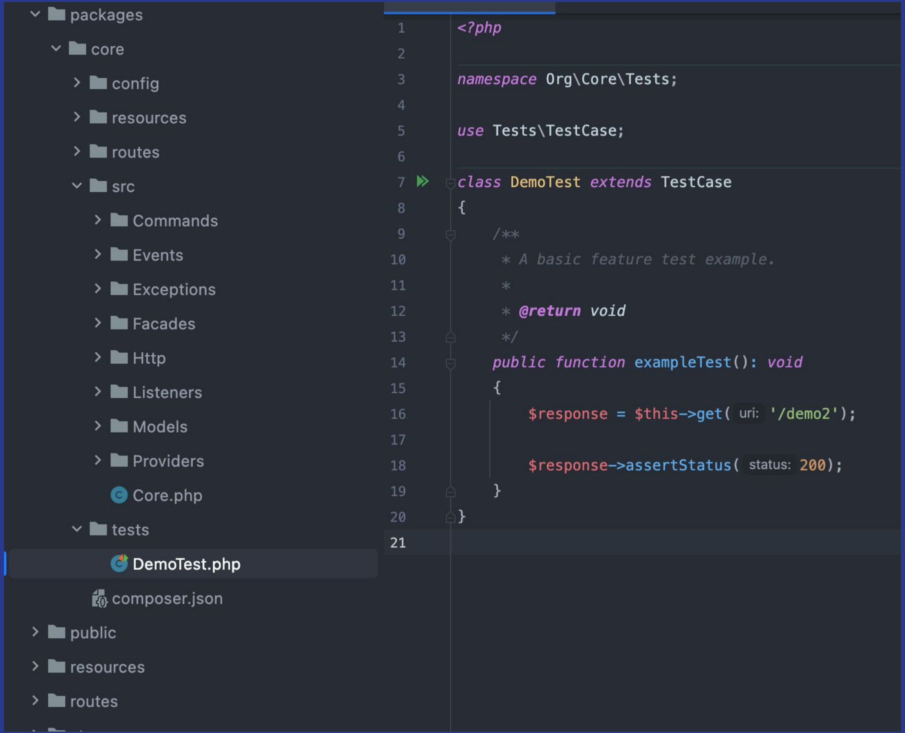
```
{
    "name": "org/core",
    "description": "The core module",
    "type": "package",
    "autoload": {
        "psr-4": {
            "Org\\Core\\": "src",
            "Org\\Core\\Tests\\": "tests"
        }
    },
    "require": {"barryvdh/laravel-dompdf": "^2.0"...},
    "extra": {
        "laravel": {
            "providers": [
                "Org\\Core\\Providers\\CoreServiceProvider"
            ],
            "aliases": {
                "Core": "Org\\Core\\Facades\\CoreFacade"
            }
        }
    }
}
```

Tests



```xml
<?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



```
packages
    core
        config
        resources
        routes
        src
            Commands
            Events
            Exceptions
            Facades
            Http
            Listeners
            Models
            Providers
            Core.php
        tests
            DemoTest.php
        composer.json
public
resources
routes
```

```php
<?php

namespace Org\Core\Tests;

use Tests\TestCase;

class DemoTest extends TestCase
{
    /**
     * A basic feature test example.
     *
     * @return void
     */
    public function exampleTest(): void
    {
        $response = $this->get( uri: '/demo2');

        $response->assertStatus( status: 200);
    }
}

```

3. Ưu và nhược điểm

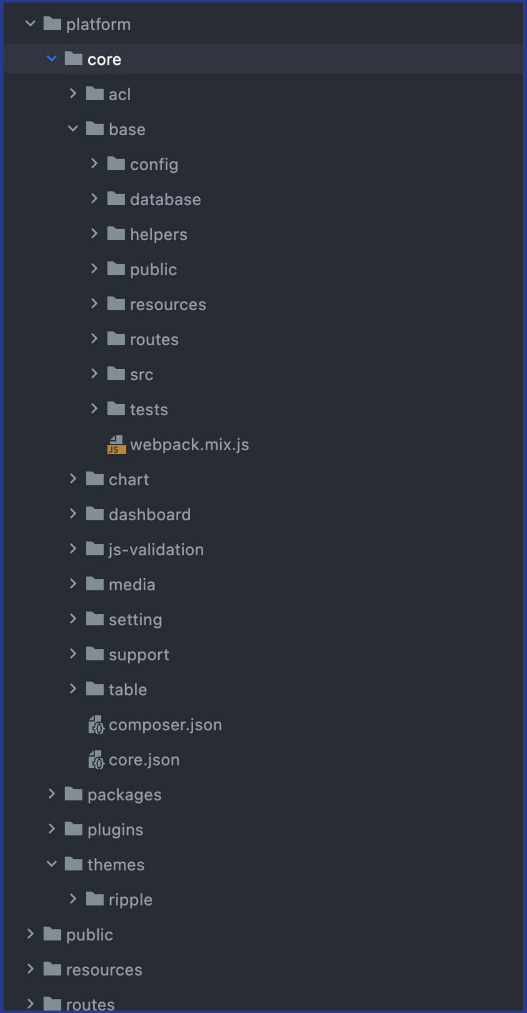
Ưu điểm

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

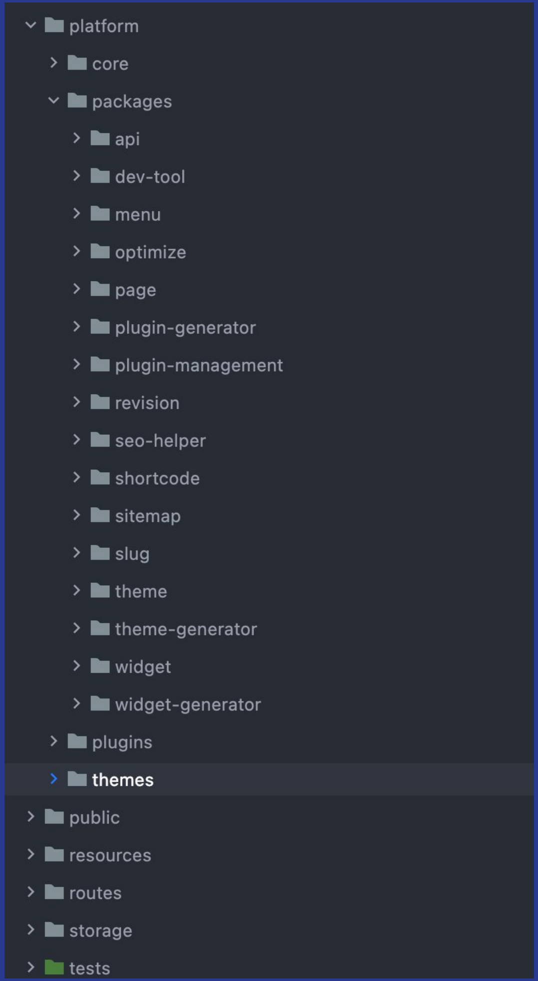
Nhược điểm

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

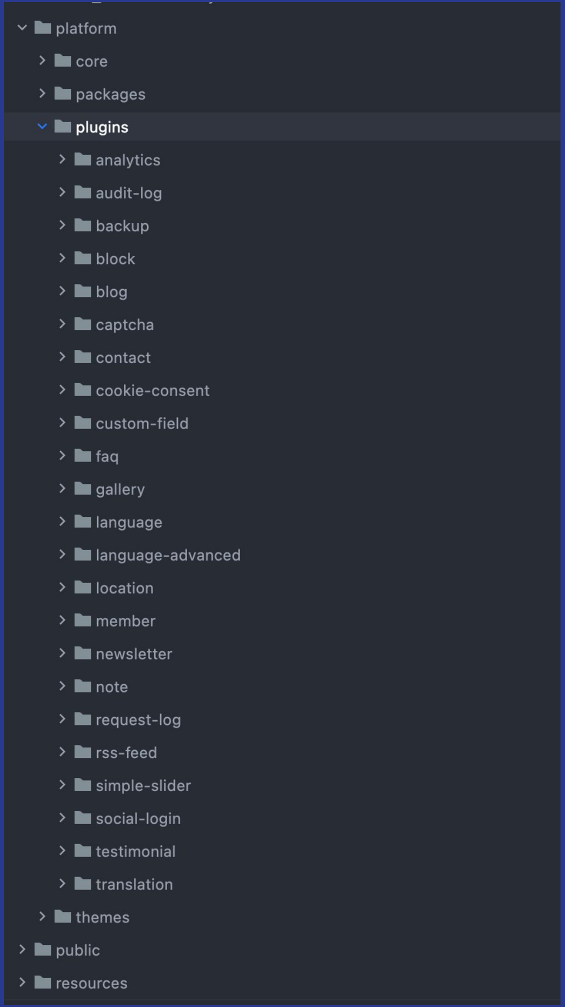
4. Demo thực tế



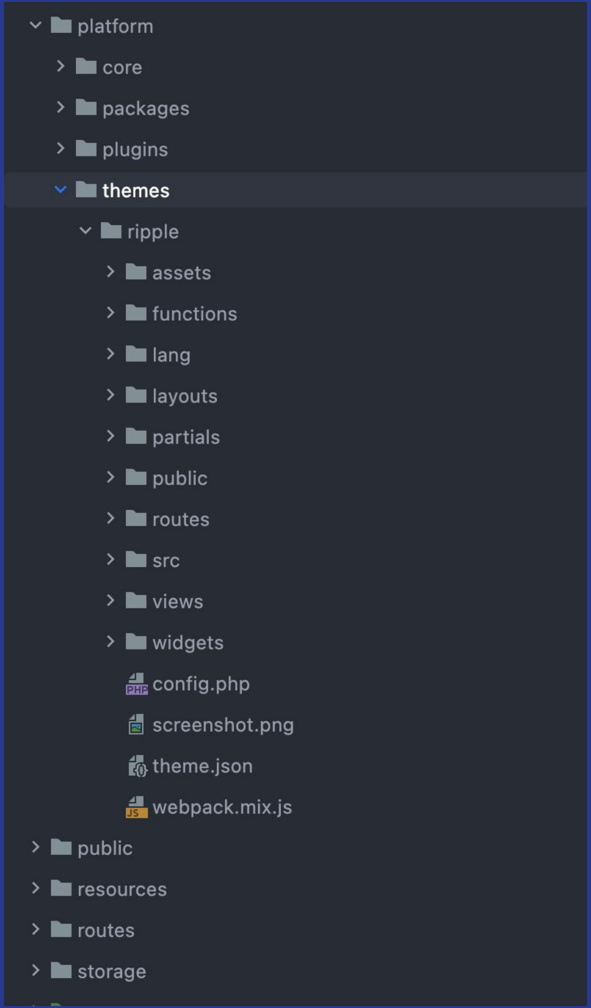
```
platform
  core
    acl
    base
      config
      database
      helpers
      public
      resources
      routes
      src
      tests
      webpack.mix.js
    chart
    dashboard
    js-validation
    media
    setting
    support
    table
    composer.json
    core.json
  packages
  plugins
  themes
    ripple
public
resources
routes
```



```
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