

Day 1: Architecture Concepts

1. Request lifecycle

- Request là gì?
 - + Request (hay HTTP request) là thông tin được gửi từ client lên server để xác định mong muốn và truy cập tài nguyên. Khi có request phát sinh, client sẽ dùng URL hay bộ định vị tài nguyên đồng nhất có chứa thông tin cần thiết để truy cập vào nguồn tài nguyên của máy chủ. ([định nghĩa request](#))
VD: khi đang ở trang web Laravel.com, nếu ta gõ what is request?? trên thanh tìm kiếm, thì ngay lập tức trình web sẽ dựa vào keyword để gửi yêu cầu tìm kiếm, lúc này bên phía server sẽ phân tích yêu cầu, gửi luồng xử lý tới vị trí lưu trữ của mã nguồn PHP và nhiệm vụ của mã nguồn là tiếp nhận yêu cầu, phân tích request và trả về response cho client ([ví dụ request](#))
 - + HTTP request gồm 3 phần: ([cấu trúc request](#), [cấu trúc request_1](#))
 - Request Line: gồm 3 phần:
 - Method (phương thức): xác định phương thức (GET, POST, PUT, DELETE,...) để server biết xử lý thông tin tài nguyên
 - Path (đường dẫn): chứa URL của request để tìm tài nguyên máy chủ
 - HTTP Version (phiên bản giao thức): xác định phiên bản giao thức HTTP. Ví dụ: HTTP/1.0 hoặc HTTP/1.1
 - Headers: truyền đạt nhiều thông tin hơn về request, giúp server hiểu cách xử lý thông tin được yêu cầu bởi client. Chẳng hạn: cookie, thông tin ủy quyền, tác nhân người dùng.
 - Message body: chứa các dòng yêu cầu, thông tin, dòng trống, tiêu đề, nội dung

```
-----Request-----
Request Line: GET /utilities/weatherfull/city/hyderabad HTTP/1.1
Request Method: GET

Request Time: 2017-08-27 13:30:30

Accept-Encoding: gzip, deflate

Host address: restapi.demoqa.com

Client IP: 84.245.10.101

Client Port: 42092

HTTP Protocol Version: HTTP/1.1

Connection: close

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.101 Safari/537.36

Accept-Language: en-GB,en-US;q=0.8,en;q=0.6

Request body:
```

[Kiến trúc hệ thống trên Laravel – phần 6 \(viblo.asia\)](#)

[Kiến trúc hệ thống trên Laravel – phần 9 \(viblo.asia\)](#)

[Tập 5: Vòng đời request Laravel \(Request lifecycle Laravel\) \(viblo.asia\)](#)

[Khoá học Laravel Framework 8.x - Bài 2: Cấu trúc thư mục - Vòng đời Request trong Laravel \(youtube.com\)](#)

[Request Lifecycle - Laravel 11.x - The PHP Framework For Web Artisans](#)

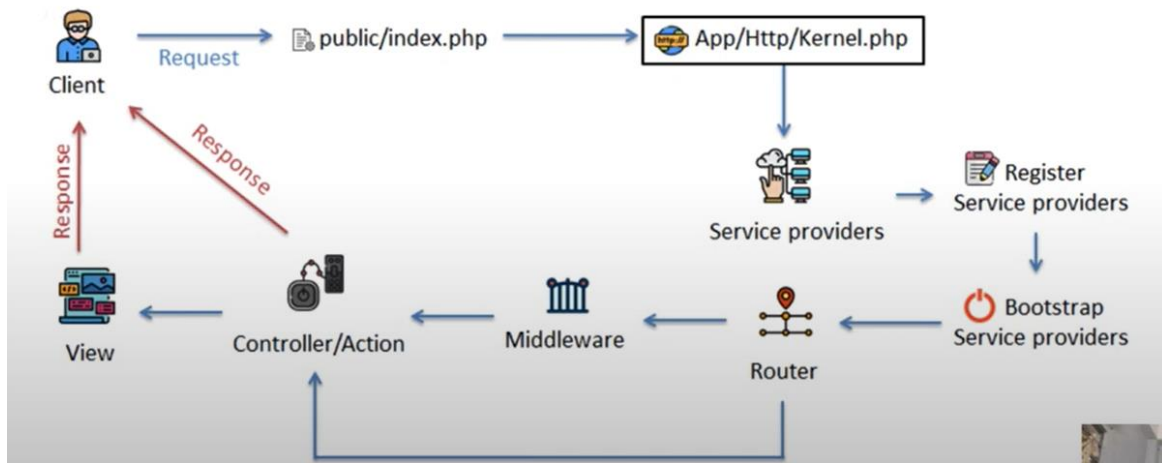
[Tìm Hiểu Về Vòng Đời Request Trong Laravel - Học Laravel \(hoclaravel.vn\)](#)

[Vòng đời của Laravel \(Request lifecycle in laravel\) \(hozitech.com\)](#)

- Vòng đời của request:

- + Bootstrap: Khi client gửi một request đến server. Đầu tiên, request sẽ gửi đến publish/index.php, đây là điểm đầu tiên mà tất cả các request từ client đều phải đi qua.
- + HTTP Kernel: Sau đó chuyển đến app/http/Kernel.php (đây được gọi là trung tâm request, nơi chứa tất cả các request từ client trước khi request được xử lý bởi ứng dụng) hoặc console kernel (xử lý các request command).
- + Service Providers: Sau đó, request được chuyển đến Service provider (đây chính là trung tâm khởi động để chạy các service).
- + Router: Trước khi request được điều hướng đến View hay Controller cụ thể nào đó, nó phải thông qua Router. Router được ví như chốt chặn kiểm tra. Router có nhiệm vụ kiểm tra tất cả route đã được khai báo trong các file thư mục routes (so với request được gửi đến)
- + Middleware: Sau đó, request sẽ qua Middleware, đây là nơi kiểm tra xem request có thỏa mãn điều kiện hay không, nếu có sẽ cho đi tiếp đến Controller hoặc không thì sẽ dừng lại hoặc chuyển hướng sang một thứ khác.
- + Finishing up: Request khi đến Controller sẽ có 2 hướng: 1 là sẽ đến view rồi trả response cho client; 2 là sẽ trả trực tiếp response cho client.

Request Lifecycle Laravel



2. Service Container

[Tìm hiểu về Service Container trong Laravel \(viblo.asia\)](https://viblo.asia)

[Service Container - Laravel 11.x - The PHP Framework For Web Artisans](#)

[Dependency Injection và Service Container trong Laravel \(giangmd.net\)](#)

[Tìm hiểu về Service Container trong Laravel \(viblo.asia\)](https://viblo.asia)

[php - What is the concept of Service Container in Laravel? - Stack Overflow](#)

- Service Container là một phần quan trọng của hệ thống, giúp quản lý các phụ thuộc (dependencies) của các class và thực hiện dependency injection. Dùng để quản lý việc tạo và phân phối các service trong ứng dụng.
- Dependency injection: cho phép các dependences được tiêm vào class thông qua constructor hoặc qua các phương thức setter.
- Lợi ích của Service Container:
 - + Tự động giải quyết phụ thuộc: nếu một class không có phụ thuộc hoặc chỉ phụ thuộc vào class cụ thể (không phải interface), SC tự động giải quyết phụ thuộc đó.
 - + Tiện lợi và không cần cấu hình: các class như Controller, event listener, middleware và nhiều class khác trong Laravel tự động nhận phụ thuộc từ Service Container
- Trong vd dưới, ta có class UserController cần truy xuất dữ liệu của FooService:
 - + Ta có 1 class FooService

```
<?php
namespace App\Services;

class FooService
{
    public function __construct()
    {
        ...
    }
    public function doSomething()
    {
        // Code for Something.
    }
}

?>
```

- + Giờ ta phải binding class FooService vào container với cái tên FooService

```
$this->app->bind('FooService', \App\Services\FooService::class);
```

- + Sau khi class được register vào container. Chúng ta có thể truy xuất nó từ bất kì vị trí nào của ứng dụng

```
// sử dụng $this->app->make để lấy ra instance của class
$fooService = $this->app->make('FooService');
$fooService->doSomething();

// ở vị trí mà code không truy cập được biến $app thì có thể
// sử dụng hàm helper global resolve
$fooService = resolve('FooService');
$fooService->doSomething();

// Hoặc có thể viết trên 1 dòng mã như sau
app()->make('FooService')->doSomething();
```

3. Service Providers

Service providers(nhà cung cấp dịch vụ) là một khái niệm quan trọng khi sử dụng dịch vụ các framework như là laravel. Giúp khởi tạo và đăng kí các service trong ứng dụng. Được sử dụng để quản lí và cấu hình các dịch vụ, gắn kết các class, thực hiện các tác vụ khởi tạo.

Chức năng của Service Providers:

- + Đăng kí service: SP cho phép đăng kí các dịch vụ vào container của Laravel
- + Cấu hình ứng dụng: cấu hình các dịch vụ, middleware và các tùy chọn khác thông qua SP
- + Tạo các singleton: SP giúp tạo ra các singleton, đảm bảo chỉ có một phiên bản duy nhất của class được tạo ra.

Trong Laravel đã tích hợp sẵn một số Service Providers, nhưng mình có thể tạo riêng cho ứng dụng. Cú pháp để tạo một Service Providers, sử dụng terminal:

```
php artisan make:provider MyServiceProvider
```

4. Facades

[Facades - Laravel 11.x - The PHP Framework For Web Artisans](#)

Facades giúp tạo ra một giao diện đơn giản để truy cập các đối tượng từ container dịch vụ của ứng dụng. Facades cung cấp một giao diện tĩnh cho các class có sẵn trong container dịch vụ của Laravel, giúp viết mã ngắn gọn và dễ nhớ mà không cần nhớ tên dài của các class.

Lợi ích của facades:

- + Cung cấp cú pháp ngắn gọn và dễ nhớ
- + Dễ kiểm thử vì không cần inject hoặc cấu hình các class thủ công
- + Tuy nhiên, cần chú ý không để class trở nên quá lớn khi sử dụng quá nhiều facades.

Nói chung, facades giúp viết mã ngắn gọn và dễ nhớ khi truy cập các tính năng của framework mà không cần phải nhớ đến tên dài của các class

```
use Illuminate\Support\Facades\Cache;
use Illuminate\Support\Facades\Route;

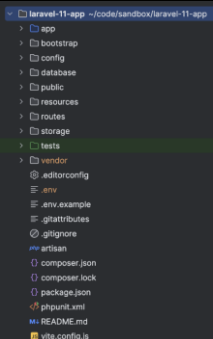

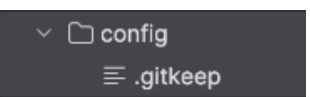
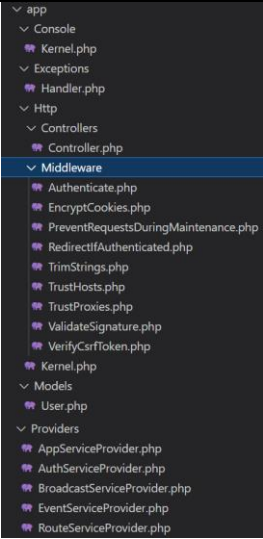
Route::get('/cache', function () {
    return Cache::get('key');
});
```

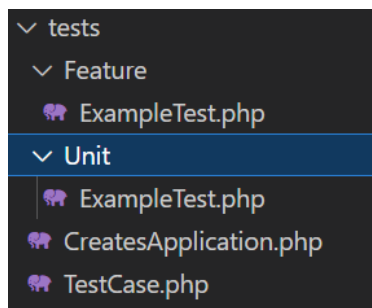
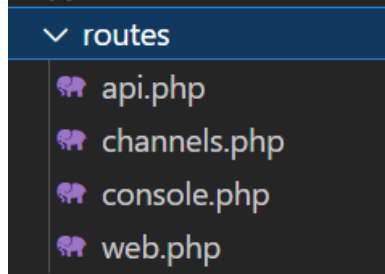
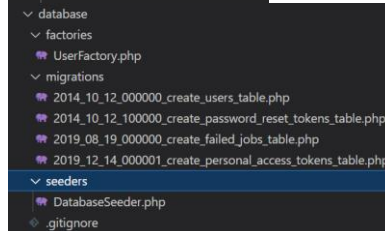
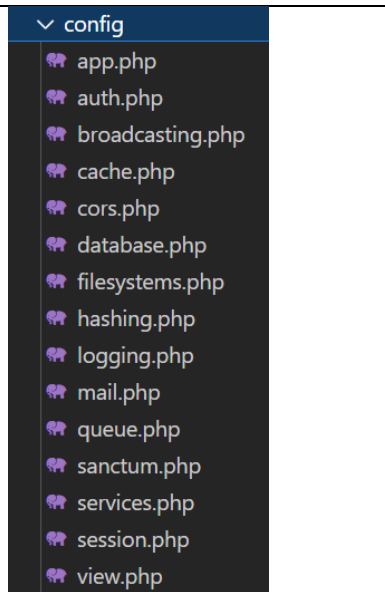
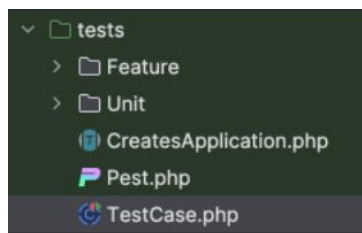
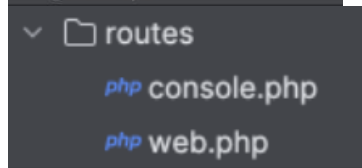
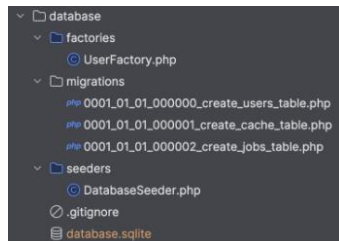
Cú pháp của facades: **Illuminate\Support\Facades\Tên_class**.

5. Application structure: Cấu trúc thư mục của Laravel 11

[Cấu trúc project Laravel | Laravel | Hướng dẫn học | Học web chuẩn \(hocwebchuan.com\)](#)

[Dive into the Streamlined Directory Structure in Laravel 11 - Laravel News \(laravel-news.com\)](#)

	 		<p>+ app: chứa tất cả cốt lõi trong ứng dụng. Hầu hết các class trong project được tạo tại đây.</p> <ul style="list-style-type: none"> • http: <ul style="list-style-type: none"> ○ http/Controllers: Chứa Controller của project ○ http/Middleware: chứa các tập tin lọc và ngăn chặn các requests. • Models: chính là phần Model trong MVC • Providers: chứa các file thực hiện việc khai báo service và bind vào trong Service Container. • Console(Laravel_10): chứa các tập tin định nghĩa các câu lệnh trên artisan • Exceptions(Laravel_10): chứa các tập tin quản lý, điều hướng lỗi.
--	--	---	---



- + **bootstrap**: chứa những file khởi động của framework và những file cấu hình auto loading, route, file cache.
- + **config**: chứa tất cả những file cấu hình.
- + **database**: chứa 2 folder: migration (tạo và thao tác với database) và seeds (Tạo dữ liệu mẫu), tiện lợi để lưu trữ dữ liệu sau này.
 - **Factories**: chứa các file định nghĩa các cột bảng dữ liệu để tạo ra các dữ liệu mẫu
 - **Migrations**: chứa các file tạo và chỉnh sửa dữ liệu
 - **Seeds**: chứa các file tạo dữ liệu thêm vào CSDL
- + **publish**: chứa file index.php, là cổng cho tất cả các request vào project, bên trong thư mục còn chứa file JavaScript và CSS.
- + **resources**: chứa những file view và raw, các file biên soạn như LESS, SASS, JavaScript. Ngoài ra còn chứa tất cả file lang trong project. Folder views chính là View trong MVC
- + **routes**: chứa tất cả các điều khiển route (đường dẫn) trong project.
 - **Api.php**: điều khiển các route của ứng dụng, như route của ứng dụng User (đăng nhập, đăng kí)
 - **Web.php**: điều khiển các route của view, như route của trang sản phẩm,...
 - **Console.php**:
 - **Channels.php**:
- + **storage**: chứa các file biên soạn blade template, gồm file based

			<p>sessions, fide caches và những file sinh ra từ project.</p> <ul style="list-style-type: none">+ tests: chứa những file test như PHPUnit test.+ vendor: chứa các thư viện của Composer và các file core.+ .env: chứa các config chính của Laravel. File cấu hình, thiết lập môi trường.+ .gitattributes, .gitignore: file dành cho xử lý git.+ artisan: file thực hiện lệnh của Laravel.+ composer.json, composer.lock, composer-setup.php: file của composer+ package.json: chứa các package cần dùng cho project.+ phpunit.xml: của phpunit dùng để testing project+ vite.config.js:
--	--	--	---