**Day 2:**

**MVC, Routing, Middleware, Request, Response, HTTP method()**

1. MVC

[App Architecture - Understanding Frontend, Backend and Web Servers (youtube.com)](https://www.youtube.com/watch?v=d1Gd-MGaleE&list=PLUU3EzfPr915ebZONvUVHKm8Bls6D7EgA)

[(79) App Architecture - Understanding Frontend, Backend and Web Servers - YouTube](https://www.youtube.com/watch?v=d1Gd-MGaleE&list=PLUU3EzfPr915ebZONvUVHKm8Bls6D7EgA)

* Laravel sử dụng kiến trúc MVC (Model – View – Controller)
* Model: đây là phần tương tác trực tiếp với dữ liệu, chịu trách nhiệm quản lí dữ liệu được truyền từ cơ sở dữ liệu và giao diện người dùng (View). Model truy xuất dữ liệu từ cơ sở dữ liệu, sau đó thực hiện một số thao tác mà ứng dụng phải thực hiện, rồi lại lưu trữ dữ liệu đó lại cơ sở dữ liệu.
* View: Là giao diện người dùng, chứa mọi thứ mà người dùng có thể nhìn tháy được trên máy tính (tức là phần front-end)
* Controller: Là nơi xử lí các yêu cầu người dùng thông qua view, là cầu nối của Model và View. Controller giúp tương tác với component Model để lấy dữ liệu từ database, rồi chuyển dữ liệu đó về component View để có được output mong muốn và được hiển thị trên màn hình View. Tương tự, khi người dùng thực hiện các thao tác trên màn hình View, sẽ tạo một số dữ liệu, Controller sẽ tìm nạp dữ liệu đó, sau đó sẽ thực hiện một số thao tác hoặc chèn dữ liệu đó vào cơ sở dữ liệu thông qua Model.

A diagram of a system

Description automatically generated

* Luồng xử lí trong MVC:
* Khi một request từ client gửi đến server. Thì bị Controller chặn lại để xem đó là URL request hay sự kiện
* Sau đó, Controller xử lí input của user, sau đó gửi request data cho Model.
* Model chuẩn bị data bằng cách truy xuất dữ liệu từ database và gửi lại response data cho Controller.
* Cuối cùng, khi xử lí xong request thì Controller gửi lại dữ liệu trở về View và hiển thị cho người dùng thông qua màn hình View.
* Ưu, nhược điểm của MVC

|  |  |
| --- | --- |
| **Ưu điểm** | **Nhược điểm** |
| * **Tổ chức mã nguồn tốt hơn** * MVC tách biệt logic hiển thị View và logic xử lí dữ liệu Model, giúp mã nguồn dễ đọc và dễ bảo trì hơn * Có thể thay đổi giao diện View mà không ảnh hưởng đến logic xử lí Model * **Dễ dàng phân chia công việc** * MVC cho phép phân chia công việc giữa các thành viên trong nhóm phát triển * Developer có thể làm việc độc lập trên từng phần của ứng dụng * **Tích hợp dễ dàng** * Các thành phần của MVC có thể được thay thế hoặc mở rộng mà không ảnh hưởng đến các thành phần khác * Dễ dàng tích hợp các thư viên bên ngoài | * **Phức tạp hóa** * MVC đòi hỏi developer phải hiểu rõ về cấu trúc và quy tắc của nó * Cần có thời gian để học và làm quen với cách làm việc của MVC * **Khó khăn trong việc debug** * Khi có lỗi, việc debug sẽ trở nên phức tạp hơn do phải theo dõi từng phần của MVC * **Tăng thời gian phát triển phần mềm** * Việc phân chia ứng dụng thành các phần riêng biệt yêu cầu thời gian và công sức hơn so với việc viết tất cả mã trong 1 file. |

1. Route

[Routing trong Laravel, cách tạo Route và sử dung AZ (freetuts.net)](https://freetuts.net/routing-trong-laravel-5644.html)

[Cùng nhau học Router Laravel 10.x (phần 1) (viblo.asia)](https://viblo.asia/p/cung-nhau-hoc-router-laravel-10x-phan-1-Ny0VGdPD4PA)

[Laravel 11 đã ra mắt! Dưới đây là tất cả những thay đổi lớn và tính năng mới. (hoangpham.space)](https://hoangpham.space/laravel-11)

[Tìm hiểu về routing trong Laravel (viblo.asia)](https://viblo.asia/p/tim-hieu-ve-routing-trong-laravel-3P0lPaPo5ox)

[Routing - Laravel 11.x - The PHP Framework For Web Artisans](https://laravel.com/docs/11.x/routing#the-default-route-files)

[Blade template trong Laravel 8 (phần 1) (toidicode.com)](https://toidicode.com/blade-template-trong-laravel-8-phan-1-446.html)

* Route là gì??
* Routing là một cơ chế quản lí URL, nó cho phép định tuyến các URL các Request của client đến các các hàm xử lí tương ứng. Khi client gửi request đến một URL, Laravel sẽ tìm xem URL đó có được định tuyến trong file Routes hay không. Nếu tìm thấy, Laravel sẽ gọi hàm xử lí tương ứng với URL đó.
* VD: khi client truy cập vào website với đường dẫn Laravel.com/routinggg, thì lúc này Route sẽ nhận 1 phần trong URL là /routing và nó sẽ xử lí request đó, gọi đến Controller Routinggg (nếu sử dụng phương thức GET) hoặc một thao tác khác tùy vào coder.
* Cấu trúc thư mục của Routes trong laravel: để định nghĩa route, ta viết trong thư mục Routes
* Routes/console.php:
* Routes/Web.php: định nghĩa các route cho giao diện web, được gán vào nhóm Middleware web (bao gồm quản lí phiên bản và bảo vệ CSRF
* Routes/api.php: định nghĩa các route cho api, được gán vào nhóm Middleware api (stateless và hỗ trợ xác thực qua token)
* Route/channels.php:
* So sánh về routes của Larevel 11 và 10

|  |  |  |
| --- | --- | --- |
|  | Laravel 11 | Laravel 10 |
|  |  | A screenshot of a computer program  Description automatically generated |
| Về cấu trúc thư mục | * L11 sử dụng các tệp web.php và api.php để định nghĩa routes | * L10 vẫn sử dụng 2 tệp đó để định nghĩa routes. Nhưng cấu hình được định nghĩa ở: App/Providers/RouterServiceProvider.php * Để thêm 1 route, ta sẽ thêm 1 router trong file App/Providers/RouterServiceProvider.php, vd nó sẽ có đường dẫn là routes/web2.php     Sau đó vào thư mục routes, thêm 1 file web2.php |
| Middleware group mặc định | * Các routes trong web.php được gán vào nhóm Middleware web, bao gồm quản lí phiên và bảo vệ CSRF | * Tương tự như L11 |
| API Routes | * Sử dụng api.php để định nghĩa các routes cho api * Có cài đặt sẵn Laravel Sanctum để hỗ trợ xác thực qua token | * không có sự hỗ trợ của Laravel Sanctum |
| Helthy - check | * có một route mới để kiểm tra trạng thái của ứng dụng helthy-check, cho phép thực hiện kiểm tra khác nhau cho các phần khác nhau của ứng dụng và đảm bảo mọi thứ đnag hoạt động một cac suôn sẻ. * Route này được xác thực trong bootstrap/app.php |  |

* Các phương thức trong Route: có 6 phương thức
* **Route::get($uri, $callback)**: dùng để truy cập đến tài nguyên của server mà không làm thay đổi dữ liệu server
* **Route::post($uri, $callback)**: dùng để tạo dữ liệu mới
* **Route::put($uri, $callback)**: dùng để cập nhập dữ liệu
* **Route::path($uri, $callback)**: dùng để gửi dữ liệu mà cần sửa đổi và không ảnh hưởng đến các phần khác của dữ liệu
* **Route::delete($uri, $callback)**: dùng đề xóa dữ liệu
* **Route::options($uri, $callback)**:

Ngoài ra còn có các phương thức khác:

* Match: dùng để định nghĩa nhiều phương thức với chung một response trả về
* Any: cũng giống như match, nó sử dụng với tất cả các phương thức trên
* View (đây gọi là route tĩnh): nếu chỉ cần trả ra 1 view, ta có thể sử dụng phương thức này

VD:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

Khi xác định nhiều route cùng chia sẻ 1 URL, ta phải xác định các phương thức trong 6 phương thức trên trước khi xác định phương thức match, any hay view.

* Định nghĩa route
* Bảo vệ khỏi CSRF ([Tìm hiểu về Laravel Framework: cơ bản về Routing (viblo.asia)](https://viblo.asia/p/tim-hieu-ve-laravel-framework-co-ban-ve-routing-bJzKmx8w59N))
* Dependency ijection
* Chuyển hướng (redirect routes)
* Tham số trong route
* Tiền tố (prefix) cho nhóm route
* Group route

1. Middleware

[Middleware trong Laravel là gì? Tìm hiểu về cách sử dụng Middleware (hozitech.com)](https://hozitech.com/middleware-trong-laravel)

[Tìm hiểu về Middleware trong Laravel (viblo.asia)](https://viblo.asia/p/tim-hieu-ve-middleware-trong-laravel-ByEZk2PoKQ0)

[Middleware trong Laravel là gì? Tìm hiểu về cách sử dụng Middleware (hozitech.com)](https://hozitech.com/middleware-trong-laravel)

[Middleware trong Laravel - w3seo kiến thức cơ bản trong laravel (websitehcm.com)](https://websitehcm.com/middleware-trong-laravel/)

* Middleware là gì??
* Middleware đóng vai trò quan trọng trong việc xử lí các request http. Giúp kiểm soát request đến trang web một các linh hoạt và dễ dàng
* Là một tầng trung gian giữ request và response được gửi qua laravel.
* Cho phép thực hiện các xử lí trước khi request được xử lí bởi controller và cũng cho phép thực hiện các xử lí sau khi request đã được xử lí xong,
* Middleware trong laravel được sử dụng để thực hiện các xử lí
* Kiểm tra quyền truy cập của client (phân quyền)
* Xác thực dữ liệu đầu vào (xác thực người dùng)
* Kiểm tra dữ liệu đầu vào có hợp lí hay không
* Thực hiện các xử lí trước và sau khi request được xử lí bởi controller.
* Ghi log
* Xử lí CORS

VD về middleware:

* Xác thực người dùng: laravel cung cấp sẵn một middleware xác thực người dùng để đăng nhập vào hệ thống, nếu client chưa đăng nhập. Middleware sẽ chuyển hướng về màn hình login để đăng nhập. Nếu client đăng nhập thành công, ứng dụng sẽ cho phép request tiếp tục vào ứng dụng
* Định nghĩa Middleware
* Tất cả Middleware đều được đặt trong thư mục app/http/Middleware
* Để tạo một Middleware, ta sử dụng lệnh:

**Php artisan make:middleware <tên\_middleware>**

* File Middleware mặc định có cấu trúc như sau:

**A white paper with black text

Description automatically generated**

* Hàm handle(): là nơi viết các xử lí cho Middleware, hàm này nhận vào 2 tham số:
* $request: Đối tượng Illuminate\Http\Request đại diện cho request đến ứng dụng
* $next: Hàm Closure đại diện cho Middleware tiếp theo
* Return $next($request): có chức năng chuyển tiếp request đến Middleware tiếp theo hoặc đến controller nếu không có Middleware nào còn lại
* Before and after Middleware

|  |  |
| --- | --- |
| **Before Middleware** | **After Middleware** |
| Thực hiện xử lí trước khi request được xử lí | Thực hiện xử lí sau khi request được xử lí |

* Cấu hình Middleware
* Để cấu hình Middleware, chỉ cần mở file app/http/kernel.php, trong mảng $routeMiddleware (đây là nơi chứa định nghĩa Middleware). Sau đó thêm Middleware
* Mymiddleware: tên định danh cho Middleware, và \App...: đường dẫn đến file Middleware vừa tạo

A computer screen shot of a black background

Description automatically generated

* Sử dụng Middleware
* Để chỉ định Middleware cho Route, sử dụng cú pháp sau

**A black background with green and white text

Description automatically generated**

* Để chỉ định nhiều Middleware cho Route, sử dụng cú pháp sau:

**A close up of words

Description automatically generated**

* Ngoài ra, có thể sử dụng Middleware trong các Controller bằng sử dụng phương thức Middleware, sử dụng cú pháp sau

**$this -> middleware(‘mymiddleware’)**

* Middleware group
* Để tạo Middleware group, ta sử dụng phương thức middlewareGroup trong file app/Http/Kernel.php

A screen shot of a computer program

Description automatically generated

* Trong đó, web và api là tên của middleware group
* Để sử dụng Middleware, ta chỉ cần đăng kí tên group middleware khi sử dụng trong Route hoặc Controller. Trong ví dụ dưới, Middleware group web đươc sử dụng để đăng khí Middleware cho Route. Các Middleware được đăng kí sẽ được áp dụng cho tất cả Route trong group này

A computer code with black text

Description automatically generated

1. Request

[HTTP Requests - Laravel 11.x - The PHP Framework For Web Artisans](https://laravel.com/docs/11.x/requests#main-content)

[Cơ bản về Request trong Laravel (viblo.asia)](https://viblo.asia/p/co-ban-ve-request-trong-laravel-bWrZnER9Kxw)

* Request là gì? (xem lại day1)
* Các phương thức Request HTTP
* GET: dùng để truy cập dữ liệu từ máy chủ cụ thể
* POST: dùng khi muốn gửi thông tin đến máy chủ, cập nhập tài nguyên. Thông tin lưu trữ ở phần thân của http request
* PUT: cập nhập và truyền tải tài nguyên. Tuy nhiên, request PUT không cố đinh, kết quả không đổi dù có request PUT nhiều lần
* DELETE: client có thể xóa tài nguyên trên server
* HEAD: không thông báo trong nội dung, dùng để đánh giá tính khả dụng của API tại điểm cuối
* PATCH: cập nhập thông tin từ máy chủ, có áp dụng sửa lỗi một phần
* TRACE: cách kiểm tra sự lặp lại theo URL của tài nguyên đích, dùng để chạy thử nghiệm, gỡ lỗi và thực hiện các thao tác chuẩn đoán API
* CONNECT: tạo kết nối đến máy chủ thông qua HTTP và tham số URL
* OPTIONS: mô tả các tùy chọn giao tiếp cho resource
* Để lấy đối tượng của HTTP Request hiện tại thông quan dependency injection Request trong Controller, ta khai báo câu lệnh sau tại phần đầu Controller, đối tượng của Request sẽ được tự động inject vào bởi service container.



A screenshot of a computer program

Description automatically generated

* Các phương thức hỗ trợ của Request

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **$request -> path()** | Trả về đường dẫn của request |  |
| **$request -> is()** | Kiểm tra xem đường dẫn request có khớp với mẫu hay không, sử dụng \* để kiểm tra trùng khớp tất cả |  |
| **$request -> url()** | Trả về đường dẫn URL không chứa query string |  |
| **$request -> fullUrl()** | Trả về đầy đủ đường dẫn URL |  |
| **$request -> method()** | Trả về hành động của request: get, post,... |  |
| **$request -> isMethod()** | Xác định xem hành động của request phải get, post,... hay không |  |
| **$request -> route() -> getName()** | Trả về tên của Route nếu nó được đặt tên |  |
| **$request -> ip()** | Trả về địa chỉ ip của người dùng |  |
| **$request -> server()** | Trả về các thông tin liên quan đến server |  |
| **$request -> header()** | Trả về thông tin header của request: thông tin về trình duyệt sử dụng user-agent, thông tin dữ liệu cookie, thông tin về host request |  |
| **$request -> route(‘category’)** | Trả về instance của category |  |

1. Response

[HTTP Responses - Laravel 11.x - The PHP Framework For Web Artisans](https://laravel.com/docs/11.x/responses#main-content)

[HTTP Responses trong Laravel (viblo.asia)](https://viblo.asia/p/http-responses-trong-laravel-1VgZvwJplAw)

* Response là dữ liệu mà server trả về cho client
* Cấu trúc của Response: cũng có 3 phần
* Status Line: gồm 3 phần
* Phiên bản HTTP
* Mã trạng thái: là một con số có 3 chữ số, do server cung cấp để phản hồi request của client. Gồm 5 lớp tiêu chuẩn, được xác định bởi chữ số đầu tiên
* 1xx: Thông tin – Request đã được tiếp nhận, quá trình tiếp tục
* 2xx: thành công – Request đã được tiếp nhận, hiểu và chấp nhận thành công
* 3xx: chuyển hướng – cần thực hiện thêm hành động để hoàn thành request
* 4xx: lỗi client – request chứa cú pháp không đúng hoặc không thể thực hiện được
* 5xx: lỗi server – server không thể thực hiện được mặc dù request hợp lệ
* Reason Phrase (Status text): mô tả trạng thái
* Response Header: dùng để truyền thông tin bổ sung cho client. Đối với Request có thể có 0 dòng header nhưng đối với response rất ít khi có trường hợp này.
* Response body: chứa dữ liệu tài nguyên được máy khách yêu cầu.

A screenshot of a computer

Description automatically generated

1. HTTP method