**Laravel Begin Step By Step**

****

**Hình 1 : Laravel framework**

Mục lục

[I. Tóm tắt: 6](#_Toc99977225)

[II. Giới thiệu 6](#_Toc99977226)

[III. Cách cài đặt 7](#_Toc99977227)

[IV. Cơ bản 9](#_Toc99977228)

[1. Routing 9](#_Toc99977229)

[- **Routing là gì ?** 9](#_Toc99977230)

[- **Dependency Injection(tiêm phụ thuộc):** 11](#_Toc99977231)

[- **CSRF Protection:** 11](#_Toc99977232)

[- **Route Parameters:** 11](#_Toc99977233)

[- **Named Routes:** 12](#_Toc99977234)

[- **Route Middleware:** 12](#_Toc99977235)

[- **Route Controller:** 12](#_Toc99977236)

[- **Route Prefixes:** 13](#_Toc99977237)

[2. Middleware 13](#_Toc99977238)

[- **Middleware là gì ?** 13](#_Toc99977239)

[- **Tạo Middleware:** 14](#_Toc99977240)

[- **Before-after-Middleware:** 14](#_Toc99977241)

[3. Controller 17](#_Toc99977242)

[- **Controller là gì ?** 17](#_Toc99977243)

[- **Controller middleware** 18](#_Toc99977244)

[- **Resource Controller là gì ?** 19](#_Toc99977245)

[- **Route Resource:** 19](#_Toc99977246)

[- **Các method và action mà Route-Resource trỏ đến:** 19](#_Toc99977247)

[4. Request 20](#_Toc99977248)

[- **Request là gì ?** 20](#_Toc99977249)

[- **Sử dụng Request** 20](#_Toc99977250)

[- **Lấy tất cả giá trị Request:** 21](#_Toc99977251)

[- **Lấy giá trị Request từ chuỗi truy vấn:** 21](#_Toc99977252)

[- **Lấy giá trị Request thông qua thuộc tính động :** 21](#_Toc99977253)

[5. Responses 22](#_Toc99977254)

[- **Reponse là gì ?** 22](#_Toc99977255)

[- **Tạo Reponse:** 22](#_Toc99977256)

[- **Reponse Redirect:** 23](#_Toc99977257)

[- **Reponse Json:** 23](#_Toc99977258)

[- **Reponse view with value:** 23](#_Toc99977259)

[6. Database 23](#_Toc99977260)

[- **Giới thiệu:** 24](#_Toc99977261)

[- **Truy vấn tất cả dữ liệu với Query Builder** 24](#_Toc99977262)

[- **Truy vấn 1 hàng dữ liệu với Query Builder** 24](#_Toc99977263)

[- **Tổng hợp** 24](#_Toc99977264)

[- **Xác định nếu giá trị đó tồn tại trong database** 24](#_Toc99977265)

[- **Cụ thể mệnh đề select** 25](#_Toc99977266)

[- **Mệnh đề Where đơn giản** 25](#_Toc99977267)

[- **Mệnh đề orWhere** 25](#_Toc99977268)

[- **Mệnh đề whereBetween** 26](#_Toc99977269)

[- **Mệnh đề whereNotBetween** 26](#_Toc99977270)

[- **Mệnh đề orderBy** 26](#_Toc99977271)

[- **Mệnh đề lastest & oldest** 26](#_Toc99977272)

[- **Where Update** 27](#_Toc99977273)

[- **Where Delete** 27](#_Toc99977274)

[7. Database: Migration 27](#_Toc99977275)

[- **Migration là gì ?** 27](#_Toc99977276)

[- **Generating Migrations** 27](#_Toc99977277)

[- **Migrations Structure** 28](#_Toc99977278)

[- **Setting The Migration Connection** 28](#_Toc99977279)

[- **Running Migration** 29](#_Toc99977280)

[- **Rolling Back Migrations** 29](#_Toc99977281)

[- **Rolling Back and Migrate using a single command** 29](#_Toc99977282)

[- **Drop all Table and Migrate** 29](#_Toc99977283)

[- **Create Table** 29](#_Toc99977284)

[- **Checking For Table Column Existence** 29](#_Toc99977285)

[- **Create Column** 30](#_Toc99977286)

[- **Column Modifiers** 30](#_Toc99977287)

[- **Dropping Columns** 30](#_Toc99977288)

[8. Eloquent 31](#_Toc99977289)

[- **Giới thiệu** 31](#_Toc99977290)

[- **Generating Model Classes** 31](#_Toc99977291)

[- **Table Names** 31](#_Toc99977292)

[- **Primary Keys** 32](#_Toc99977293)

[- **Increment** 32](#_Toc99977294)

[- **keyType** 33](#_Toc99977295)

[- **Timestamps** 33](#_Toc99977296)

[- **Retrieving Models** 34](#_Toc99977297)

[- **Building Queries** 34](#_Toc99977298)

[- **Retrieving Single Models** 34](#_Toc99977299)

[- **Not Found Exception** 35](#_Toc99977300)

[- **Retrieving Or Creating Models** 35](#_Toc99977301)

[- **Retrieving Aggregates** 36](#_Toc99977302)

[- **Inserts** 36](#_Toc99977303)

[- **Update** 36](#_Toc99977304)

[- **Delete Model** 37](#_Toc99977305)

[- **Delete Model By Primary Key** 37](#_Toc99977306)

[- **Soft Deleting** 37](#_Toc99977307)

[- **Restoring Soft Deleted Models** 38](#_Toc99977308)

[9. Eloquent Relationship 39](#_Toc99977309)

[- **Giới thiệu** 39](#_Toc99977310)

[- **One to One** 39](#_Toc99977311)

[- **One to One Inverse** 39](#_Toc99977312)

[- **One to Many** 40](#_Toc99977313)

[- **One to Many Inverse** 40](#_Toc99977314)

[- **Querying Belongs To Relationships** 41](#_Toc99977315)

[- **Many to Many Relationship** 41](#_Toc99977316)

[- **Many To Many Inverse** 42](#_Toc99977317)

[10. Code Example 42](#_Toc99977318)

[- **Giới thiệu** 42](#_Toc99977319)

[- **Chuẩn bị** 42](#_Toc99977320)

[- **Controller** 43](#_Toc99977321)

[- **Model** 43](#_Toc99977322)

[- **Route** 43](#_Toc99977323)

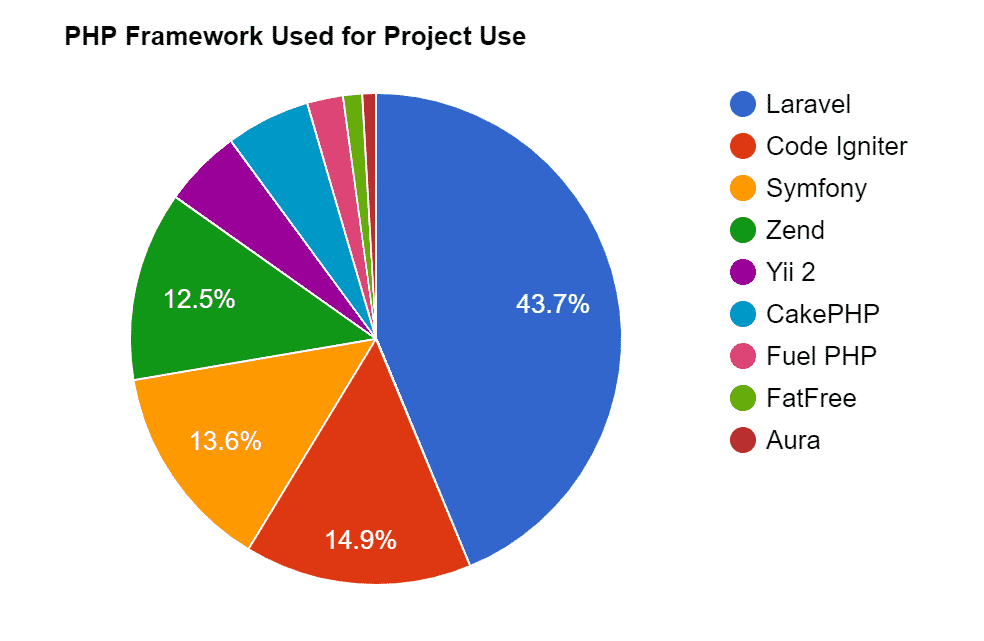
[- **Kết :** 44](#_Toc99977324)

# Tóm tắt:

* Tài liệu mô tả các bước để beginner tiếp cận với Laravel Framework, tài liệu có mô tả và giải thích các khái niệm cơ bản của Laravel Framework , có hình ảnh kèm code và giải thích ở github.
* Link github :

<https://github.com/tuancutedeptrai/Laravel-Beginer-Step-By-Step>

# Giới thiệu



**Hình 2 : Biểu đồ các Framework mô tả các Framework PHP thường được sử dụng trong các project 2021**

* Laravel là 1 trong những Framework được sử dụng nhiều nhất trong các Framework PHP tính đến hiện tại .
* Laravel là gì ? Laravel là 1 khung ứng dụng web với cú pháp tốt , đơn giản . Laravel cung cấp 1 cấu trúc và 1 điểm bắt đầu để ta dễ dàng tạo 1 ứng dụng web, Laravel được viết ra để người sử dụng chúng có thể tập trung vào phát triển những thứ thú vị và không cần quan tâm chi tiết quá về những điều có sẵn trong Laravel.
* Laravel có nhiều cộng đồng người dùng đông đảo với những forum để khi ta có câu hỏi thì luôn luôn có thể nhanh chóng tìm được câu trả lời.

<https://laravel.io/forum> : Forum laravel.io .

<https://laracasts.com/> : Laracast được ví tựa như Stackoverflow của Laravel khi có bất kì câu hỏi nào ta có thể tìm kiếm ở đây .

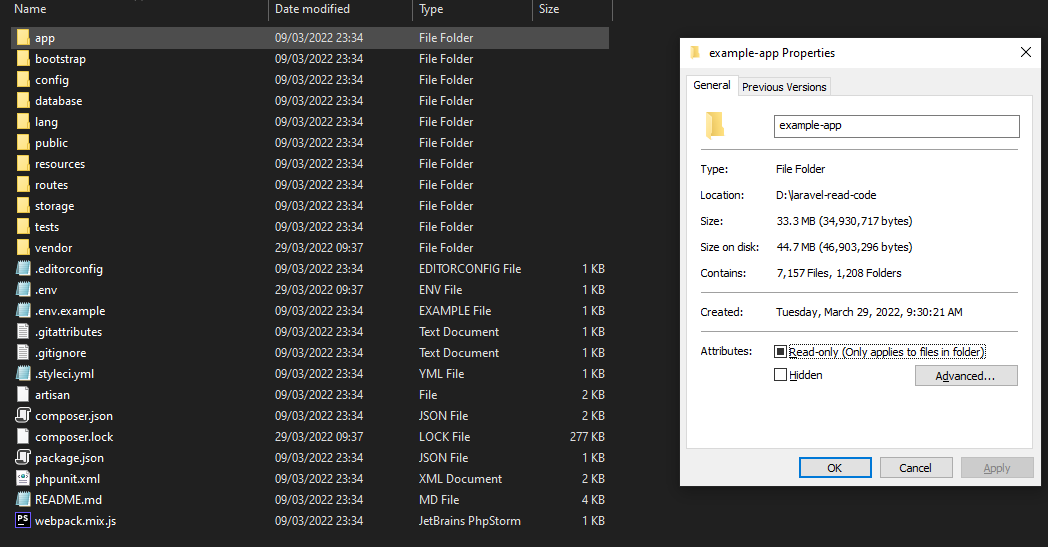
# Cách cài đặt

* Ta có thể lên github tìm source Laravel github : <https://github.com/laravel/laravel>
* Tuy nhiên ở đây ta không cài qua github mà thông qua cách cài được giới thiệu ở trang chủ của Laravel.
* Ta sẽ cài đặt Laravel thông qua Composer , để có thể cài đặt Laravel thông qua Composer thì máy tính phải có sẵn Composer . Ta có thể tải và cài đặt Composer thông qua trang chủ của nó ở đây : <https://getcomposer.org/>
* Tiếp đến ta có thể dùng lệnh cài Laravel thông qua Composer được giới thiệu ở trang chủ của Laravel .
* Mở CMD(command Prompt) của Window hoặc terminal ở MAC và chạy lệnh :



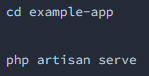
**Hình 3 : Câu lệnh tải Laravel bằng Composer**

* Khi tải về ta sẽ được các thư mục như hình dưới và khi mở thông tin kiểm tra ta thấy project có chứa khoảng 7157 file .

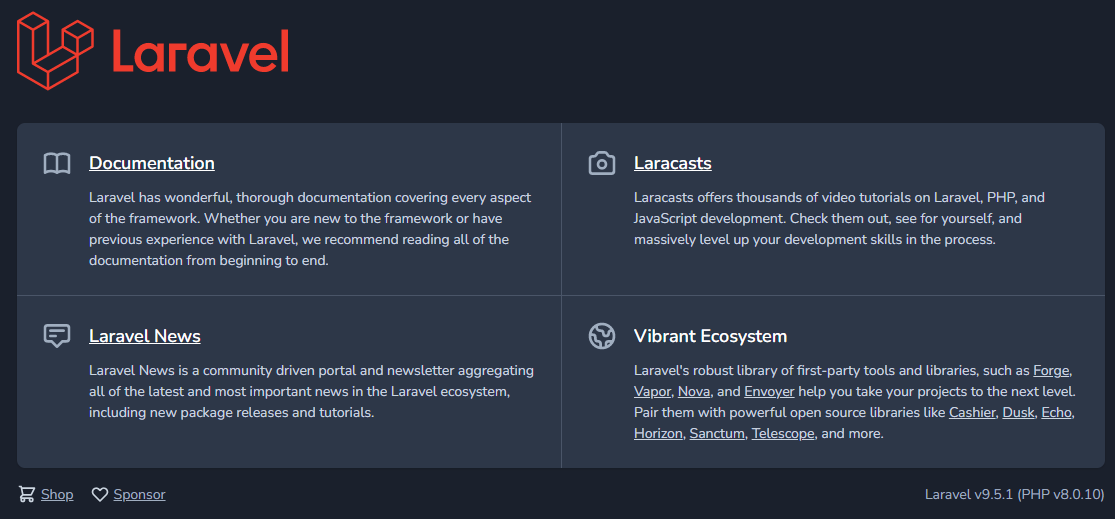


**Hình 4: Thư mục và số file sau khi tải về**

**-**Ở thư mục hiện tại ta mở CMD(command Prompt) chạy 2 câu lệnh sau và mở trình duyệt lên sẽ thấy kết quả sau khi tải về .



**Hình 5: 2 câu lệnh chạy project bằng CMD.**



**Hình 6: Kết quả hiển thị ở trình duyệt.**

# Cơ bản

## Routing

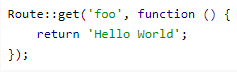
* **Routing là gì ?**

Ví dụ muốn đi từ A -> B thì ta sẽ đi qua những tuyến đường nào đó. Route là tuyến đường . Để 1 người dùng có thể vào và tương tác với trang web của chúng ta thì chúng ta cũng cần phải thiết kế các tuyến đường để người dùng có thể vào.



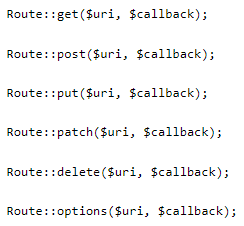
**Hình 7: Route**

* Tất cả các định tuyến Laravel được định nghĩa tại tập tin **app/Http/routes**, nó sẽ được Framework tự động tải. Định tuyến Laravel cơ bản nhất đơn giản chấp nhận một URI và một Closure (một hàm hay một tham chiếu đến một hàm cùng với môi trường tham chiếu), cung cấp một cách rất đơn giản và ý nghĩa cho việc định tuyến.



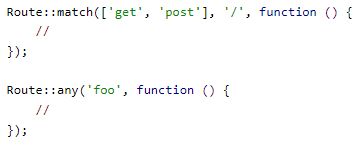
**Hình 8:Ví dụ cách đăng kí 1 route**

* Tất cả các định tuyến được định nghĩa trong thư mục routes đều được tự động load bởi **App\Providers\RouteServiceProvider.** File **web.php** sẽ định nghĩa tất cả route cho web interface. File **api.php** sẽ định nghĩa các định tuyến cho các API(Dành cho những website tạo ra các api cho người dùng hoặc những web có sử dụng những công nghệ như React,Vue….).
* Các phương thức định tuyến có sẵn : get , post , put , patch , delete , options



**Hình 9:** **Các phương thức định tuyến có sẵn**

* Đôi khi bạn có thể phải đăng ký nhiều định tuyến mà đáp ứng nhiều phương thức HTTP. Bạn có thể làm như vậy bằng cách sử dụng các phương thức **match**. Hoặc, bạn thậm chí có thể đăng ký một định tuyến mà đáp ứng tất cả các phương thức HTTP bằng cách sử dụng phương thức **any**:



**Hình 10:** **Ví dụ phương thức match – any**

### 

### **Dependency Injection(tiêm phụ thuộc):**

Nếu đặt 1 phụ thuộc vào callback của route. Khi route loading Laravel sẽ tự động load phụ thuộc đó .

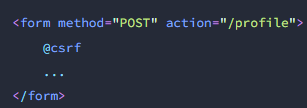
Ví dụ : Phụ thuộc Request sẽ tự động load kèm theo route

**Route::get('/users', function (Request $request) { // ... });**

**Hình 11:** **Ví dụ Laravel sẽ load phụ thuộc Request kèm theo route**

### **CSRF Protection:**

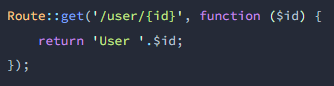
Đối với Form HTML có các method như Post, Put, Patch, Delete thì Laravel bắt buộc phải có CSRF token để tránh lỗi bảo mật CSRF .



**Hình 12:** **Ví dụ CSRF method Post**

### **Route Parameters:**

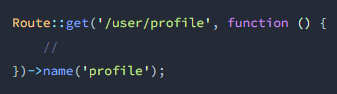
Đôi khi ta cần phải thêm tham số bắt buộc vào route để chúng ta có thể có dữ liệu để làm việc . Ví dụ ta cần thêm ID vào để thực hiện 1 truy vấn nào đó với ID user .



**Hình 13:** **Ví dụ đặt id ở tham số route và thực hiện gọi id ở callback của route**

### **Named Routes:**

Các tuyến đường được đặt tên cho phép tạo URL hoặc chuyển hướng thuận tiện cho các tuyến đường cụ thể. Bạn có thể chỉ định tên cho một tuyến bằng cách xâu chuỗi **name** phương thức vào định nghĩa tuyến:



**Hình 14:** **Ví dụ đặt tên cho route để redirect khi cần .**

### **Route Middleware:**

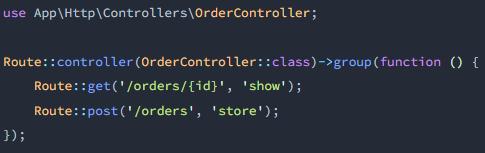
Để gán middleware vào 1 nhóm route , bạn có thể sử dụng middleware phương thức trước khi định nghĩa nhóm route . Middleware sẽ thực thi lần lượt theo thứ tự đặt trong Array.



**Hình 15:** **Ví dụ đặt middleware cho 1 nhóm route**

### **Route Controller:**

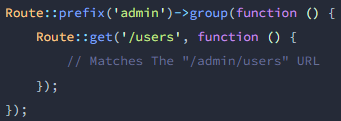
Nếu 1 nhóm route sử dụng cùng 1 controller chúng ta có thể sử dụng phương thức controller để định nghĩa 1 controller cho group.Sau đó khi định nghĩa từng route ta chỉ cần cung cấp method của controller đó cho phù hợp với route.



**Hình 16:** **Route Controller**

### **Route Prefixes:**

Nếu 1 nhóm route sử dụng cùng 1 tiền tố phía trước chúng ta có thể sử dụng phương thức **prefix** để định nghĩa cho route group.Sau đó ta định nghĩa từng route ta chỉ cần cung cấp URL sau tiền tố và định nghĩa callback cho các route đó .

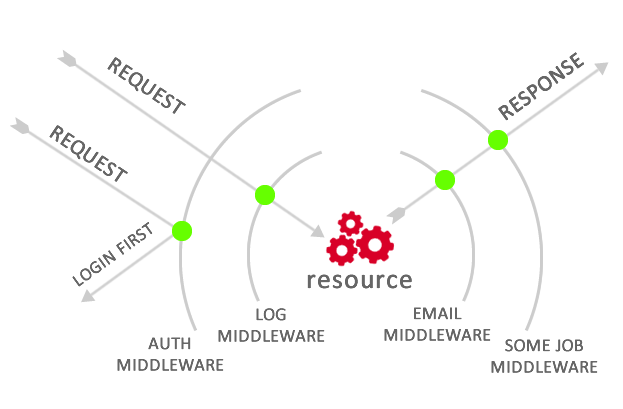


**Hình 17:** **Route Prefixes**

## Middleware

### **Middleware là gì ?**

Middleware cung cấp 1 cơ chế thuận tiện để kiểm tra và lọc các yêu cầu http xâm nhập vào ứng dụng của bạn . Ví dụ Laravel có middleware xác minh người dùng đã đăng nhập hay chưa . Nếu người dùng không được xác thực, middleware sẽ chuyển hướng người dùng đến màn hình đăng nhập ứng dụng của bạn. Tuy nhiên, nếu người dùng được xác thực, middleware sẽ cho phép yêu cầu tiến sâu hơn vào ứng dụng.

** Hình 18:** **Middleware**

### **Tạo Middleware:**

Để tạo middleware ta có thể dùng câu lệnh như sau : *php artisan make:<tenMiddleware>*

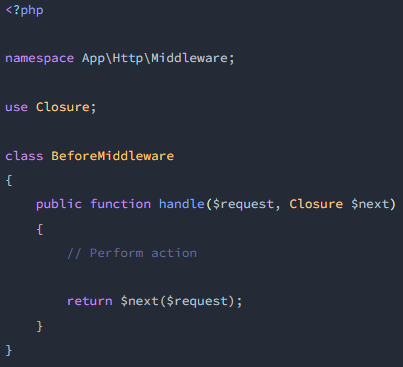


**Hình 19: Tạo** **Middleware**

### **Before-after-Middleware:**

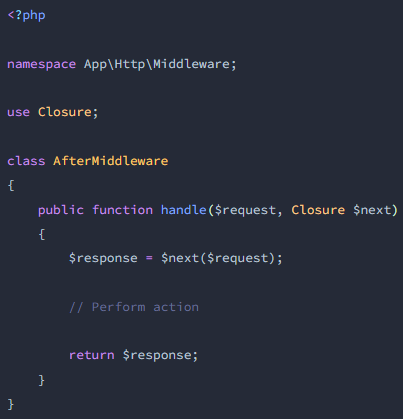
1 Middleware có thể chạy trước hoặc sau khi ứng dụng đã xử lý 1 Request .

**+ Before Middleware:**



**Hình 20: Before** **Middleware**

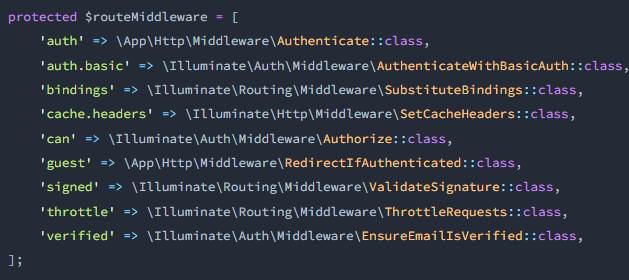
**+ After Middleware:**



**Hình 21: After Middleware**

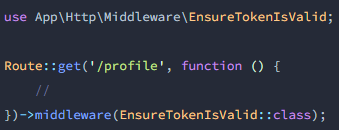
* **Register Middleware:** 1 Middleware để có thể chạy trước tiên ta cần phải đăng kí nó

**+ Register Middleware for a specific route:** Ta tìm đến app/Http/Kernel.php trong ứng dụng của bạn và tìm đến biến **$routeMiddleware** và thêm key của middleware chúng ta vừa tạo.



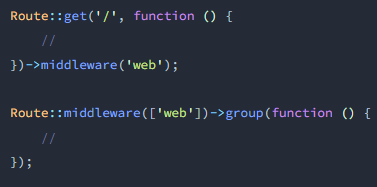
**Hình 22: Register Middleware**

**+ Use Middleware for a specific route:** Để sử dụng middleware trên 1 route



**Hình 23: Sử dụng Middleware trên 1 route**

**+ Use Middleware for a group route:** Đôi khi ta cần sử dụng middleware trên 1 nhóm route.



**Hình 24: Sử dụng Middleware trên 1 group route**

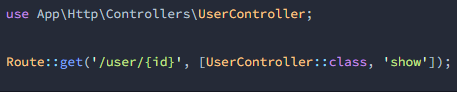
## Controller

* **Controller là gì ?**

Thay vì viết tất cả logic xử lý yêu cầu dưới dạng callback ở route, thì chúng ta có thể viết xử lý logic ở Controller và gom nhóm các xử lý chung 1 nhóm thành 1 Controller.



**Hình 25: Controller**



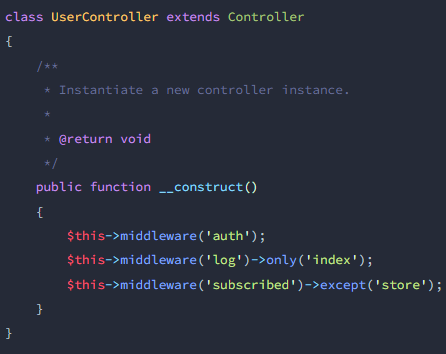
**Hình 26: Sử dụng Controller**

* **Controller middleware**

Xử dụng middleware trong Controller



**Hình 27: Sử dụng Controller**



**Hình 28: Sử dụng middleware trong hàm construct**

* **Resource Controller là gì ?**

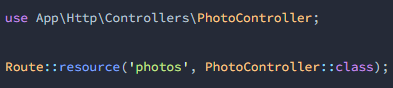
Resource Controller là 1 Controller khi tạo sẽ đầy đủ các phương thức của 1 Controller cho 1 Restful API(xem, thêm , xóa , sửa …)



**Hình 29: Ví dụ câu lệnh tạo 1 resource Controller**

* **Route Resource:**

Route-Resource là route chứa đủ các phương thức Get,Post,Put,Patch,Delete method được sử dụng theo tên Resource Controller mà route đó gọi ra.



**Hình 30:Route Resource**

* **Các method và action mà Route-Resource trỏ đến:**

Route-Resource là route chứa đủ các phương thức **Get,Post,Put,Patch,Delete** method được sử dụng theo tên Resource Controller mà route đó gọi ra.



**Hình 31:Bảng method hỗ trợ trong Route Resource**

## Request

* **Request là gì ?**

Request là đối tượng để tương tác với HTTP request hiện tại được xử lý bởi ứng dụng cũng như những giá trị được truyền vào yêu cầu bao gồm file , cookie , input…

* **Sử dụng Request**

Ta có thể sử dụng giá trị của Request thông qua biến Request ở tham số của function, tham số này sẽ được Laravel tự động tiêm phụ thuộc .



**Hình 32:Ví dụ sử dụng giá trị của Request**

* **Lấy tất cả giá trị Request:**



**Hình 33:Lấy tất cả giá trị request**

* **Lấy giá trị Request từ chuỗi truy vấn:**

Ta có thể lấy giá trị input truy xuất từ chuỗi truy vấn trên URL , lấy từng giá trị theo cách sau :



**Hình 34:Lấy giá trị 1 biến query trên URL**

Nếu giá trị đó không có , đối số thứ 2(giá trị mặc định) sẽ được trả về .



**Hình 35:Giá trị thứ 2 sẽ trả về nếu như truy vấn giá trị name không tồn tại**

* **Lấy giá trị Request thông qua thuộc tính động :**

Lấy giá trị value bằng cách trỏ tên thuộc tính của input



**Hình 36:Lấy giá trị name một cách động .**

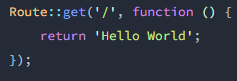
## Responses

* **Reponse là gì ?**

+ Response là phản hồi khi ta xử lý xong 1 yêu cầu hoặc có lỗi gì đó cần phải trả về thông báo cho user

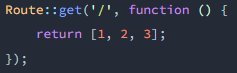
* **Tạo Reponse:**

+ Tạo 1 response đơn giản trả về string :



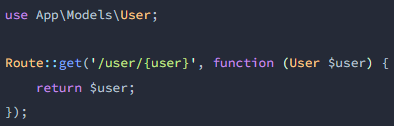
**Hình 37:Trả về 1 string value**

+ Tạo 1 response đơn giản trả về array : Dữ liệu trả về kiểu array sẽ tự động được Laravel chuyển về kiểu Json .



**Hình 38:Trả về 1 Array value**

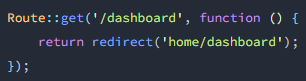
+ Tạo 1 response trả về kiểu Object : Dữ liệu trả về kiểu Object ví dụ Eloquent sẽ tự động được Laravel chuyển về kiểu Json .



**Hình 39:Trả về 1 Object**

* **Reponse Redirect:**

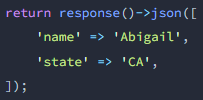
Đôi khi bạn muốn tạo 1 redirect người dùng tới 1 route khác . Khi đó ta có thể dùng Response Redirect .



**Hình 40:Return redirect**

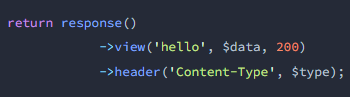
* **Reponse Json:**

Đôi khi bạn muốn return 1 json value ta có thể return theo cách sau :



**Hình 41:Return redirect**

* **Reponse view with value:**



**Hình 42:Return view with data**

## Database

* **Giới thiệu:**

Laravel cung cấp các công cụ để ta truy vấn database đồng thời chống các lỗi bảo mật trong database như Eloquent , Query Builder.

* **Truy vấn tất cả dữ liệu với Query Builder**

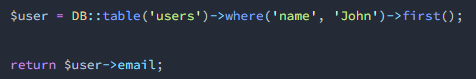
Ta có thể dùng DB Façade cung cấp tên table và dùng method get để get tất cả row từ table đó .



**Hình 43:Truy vấn tất cả dữ liệu từ bảng users**

* **Truy vấn 1 hàng dữ liệu với Query Builder**

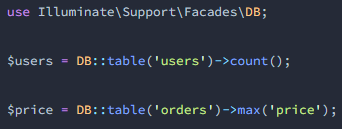
Nếu bạn chỉ cần truy xuất một hàng từ bảng cơ sở dữ liệu, bạn có thể sử dụng phương thức first của lớp DB Facade.



**Hình 44:Truy vấn 1 hàng dữ liệu**

* **Tổng hợp**

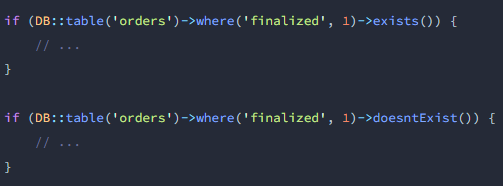
Trình tạo truy vấn cũng cung cấp nhiều phương pháp để truy xuất các giá trị tổng hợp như **count** và . Bạn có thể gọi bất kỳ phương thức nào sau đây sau khi tạo truy vấn của mình **max min avg sum**



**Hình 45:Ví dụ tổng hợp giá trị**

* **Xác định nếu giá trị đó tồn tại trong database**

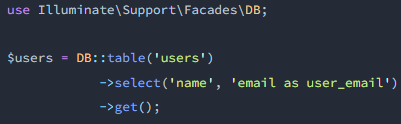
Nếu cần xác định 1 biến có tồn tại hay không ta có thể dùng hàm exists và doesntExist



**Hình 46:Xác định giá trị tồn tại hoặc không trong database**

* **Cụ thể mệnh đề select**

Bạn có thể không muốn select tất cả cột từ database . Sử dụng **select** method bạn có thể điều chỉnh câu lệnh select theo ý mình .



**Hình 47:Select cụ thể giá trị trong bảng**

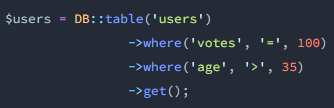
Ta có thể select những giá trị riêng biệt bằng hàm **distinct**



**Hình 48:Select những giá trị riêng biệt**

* **Mệnh đề Where đơn giản**

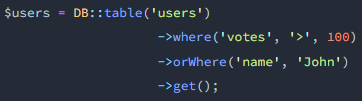
Bạn có thể dùng mệnh đề where để lựa chọn các giá trị theo yêu cầu . Mệnh đề where có 3 tham số . Tham số 1 là tên cột để so sánh , tham số thứ 2 là toán tử , tham số thứ 3 là giá trị để so sánh .



**Hình 49:Sử dụng mệnh đề where**

* **Mệnh đề orWhere**

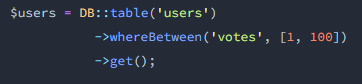
Bạn có thể dùng mệnh đề orWhere để so sánh có thể hoặc



**Hình 50:Sử dụng mệnh đề orWhere**

* **Mệnh đề whereBetween**

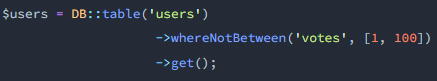
Bạn có thể dùng mệnh đề whereBetween để lấy các giá trị khoảng giữa theo điều kiện.



**Hình 51:Sử dụng mệnh đề whereBetween**

* **Mệnh đề whereNotBetween**

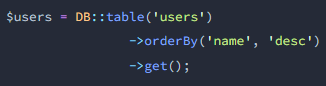
Bạn có thể dùng mệnh đề whereNotBetween để lấy các giá trị nằm ngoài khoảng nào đó theo điều kiện.



**Hình 52:Sử dụng mệnh đề whereNotBetween**

* **Mệnh đề orderBy**

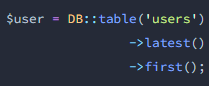
Đôi khi bạn muốn sắp xếp kết quả truy vấn theo một cột nhất định , theo một hướng nhất định(tăng dần , giảm dần ) ta có thể dùng mệnh đề orderBy



**Hình 53:Sử dụng mệnh đề orderBy giảm dần**

* **Mệnh đề lastest & oldest**

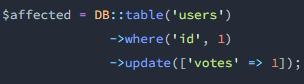
Mệnh đề lastest và oldest dựa vào cột created\_at để đưa ra kết quả gần nhất hoặc cũ nhất.



**Hình 53:Sử dụng mệnh đề lastest**

* **Where Update**

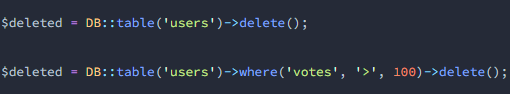
Ta có thể dùng where để lựa chọn đúng đối tượng cần Update và sau đó Update giá trị của chúng.



**Hình 54:Update Where**

* **Where Delete**

Ta có thể dùng where để lựa chọn đúng đối tượng cần xóa và sau đó xóa.



**Hình 54:Delete Where**

## Database: Migration

* **Migration là gì ?**

Migration giống như quản lí version cho database , cho phép teammate định nghĩa và share database cấu trúc cho mọi người trong team.

* **Generating Migrations**

Chúng ta có thể dùng lệnh này để tạo cú pháp tạo 1 bảng trên database

Cú pháp : ***php artisan make:migration create\_<tenbang>\_table***



**Hình 55:Tạo migration database**

* **Migrations Structure**

Migration có 2 method là up và down . Method up sẽ chạy khi tạo mới 1 bảng , ngược lại method down sẽ chạy khi xóa bảng đó khỏi database



**Hình 55.1:Migration flight trong database**

* **Setting The Migration Connection**

Nếu migration của bạn cần tương tác với database khác database được setting mặc định của ứng dụng của bạn , bạn có thể cài đặt biến **$connection** trong migration của bạn .



**Hình 56:** **Setting The Migration Connection**

* **Running Migration**

Để chạy migration trên database ta có thể dùng lệnh ***php artisan migrate***

* **Rolling Back Migrations**

Để rollback migration mới nhất bạn có thể dùng lệnh rollback. Lệnh này quay trở lại bản cuối cùng của migration bao gồm nhiều file .

Cú pháp : ***php artisan migrate:rollback***

* **Rolling Back and Migrate using a single command**

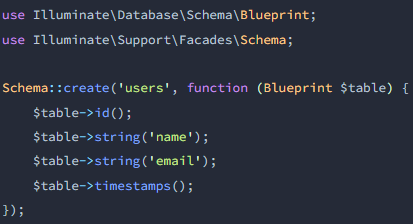
Để rollback và migrate bằng cách chỉ sử dụng 1 lệnh ta có thể sử dụng lệnh ***php artisan migrate:refresh***

* **Drop all Table and Migrate**

Để drop tất cả bảng và migrate lại chúng ta có thể sử dụng lệnh ***php artisan migrate:fresh***

* **Create Table**

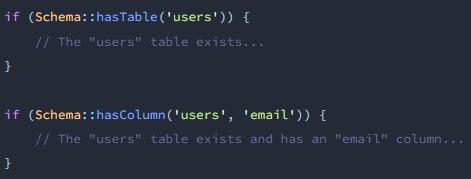
Để tạo bảng mới trong database sử dụng method **create** . Method create có 2 tham số , 1 là tên bảng , 2 là callback để định nghĩa bảng có những column nào và column có thuộc tính ra sao .



**Hình 57:** **Sử dụng method create**

* **Checking For Table Column Existence**

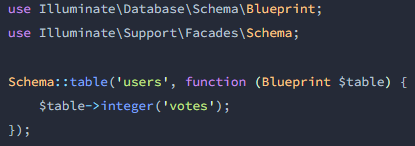
Để kiểm tra có bảng đó trong database hoặc có cột đó trong database ta có thể dùng method **hasTable** hoặc method **hasColumn**



**Hình 58:** **Sử dụng method hasTable và hasColumn**

* **Create Column**

Ta có thể tạo cột trong database bằng method **table**. Giống như method create , table method nhận 2 tham số : 1 là tên bảng , 2 là 1 closure nhận thực thể bạn muốn bạn có thể dùng nó để add column vào bảng .

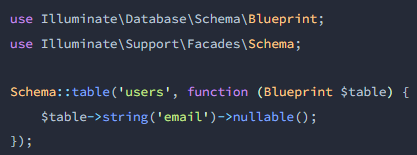


**Hình 59:** **Sử dụng method table để tạo column**

* **Column Modifiers**

Ta có thể thêm định nghĩa cho column khi thêm 1 cột mới vào database

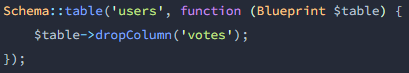
Các modifiers bạn có thể đọc trên docs của Laravel



**Hình 60:** **Sử dụng method nullable để tạo column có thể null**

* **Dropping Columns**

Để xóa 1 cột bạn có thể sử dụng method **dropColumn** trên Schema builder blueprint của closure method table



**Hình 61:** **Sử dụng method dropColumn**

## Eloquent

* **Giới thiệu**

Laravel bao gồm Eloquent , 1 object-relation mapper để chúng ta dễ dàng tương tác với database. Khi sử dụng Eloquent , mỗi database table có 1 Eloquent tương ứng để dễ dàng tương tác với bảng . Ngoài ra có thể sử dụng Eloquent để dễ dàng truy vấn , thêm , xem , xóa , sửa với bảng ghi dễ dàng .

* **Generating Model Classes**

Để tạo 1 Model ta có thể dùng lệnh ***php artisan make:model <TenModel>***

Model tạo bằng lệnh sẽ kế thừa lớp Model gốc của Laravel .

Model tạo ra sẽ nằm trong thư mục app/Models



**Hình 62:** **Sử dụng lệnh để tạo Model**

Ta có thể tạo thêm những thứ liên quan như migration , controller … bằng các option của nó . Vui lòng đọc các option này trên Laravel docs



**Hình 62:** **Tạo model và thêm option migration**

* **Table Names**

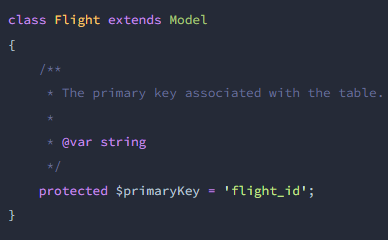
Laravel dựa vào tên class Model sẽ mặc định tạo ra kết nối với bảng trên database khi mặc định setting trong Laravel . Ví dụ ta có model **Flight** , Laravel sẽ tự động tìm bảng **flights** trên cơ sở dữ liệu và tự động kết nối . Nếu tên bảng có dấu gạch dưới , với mỗi từ đầu trong gạch dưới của tên bảng Laravel yêu cầu chữ cái đầu tiên đó sẽ là viết hoa . Ví dụ **AirTrafficController** sẽ tự động kết nối với bảng **air\_traffic\_controllers** .Nếu như những tự động kết nối của Laravel không đúng với tên bảng trên database ta có thể định nghĩa 1 biến **table** trong Model . Khi ta đưa dữ liệu vào biến này Laravel sẽ kết nối với tên bảng của biến đó .



**Hình 63:** **Cấu hình kết nối bảng cho Model qua biến table**

* **Primary Keys**

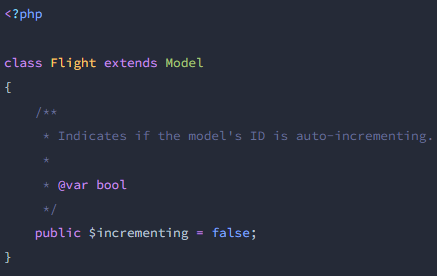
Mặc định Laravel sẽ hiểu column có tên **id** là khóa chính. Tuy nhiên nếu cần ta có thể định nghĩa lại biến **$primaryKey** . Khi ta định nghĩa lại biến này Laravel sẽ hiểu Model đó có khóa chính là cột có tên dữ liệu trong biến



**Hình 64:** **Cấu hình khóa chính cho 1 Model**

* **Increment**

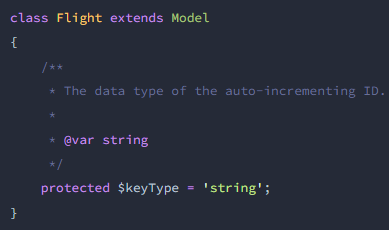
Laravel sẽ mặc định hiểu khóa chính là khóa tự động tăng . Tuy nhiên nếu cần ta có thể chỉnh sửa biến **$incrementing** của Laravel trong Model để Laravel hiểu khóa chính của chúng ta không tự động tăng . Biến này chỉ có 2 giá trị là **true** hoặc **false**



**Hình 65:** **Cấu hình khóa chính tự động tăng hoặc không**

* **keyType**

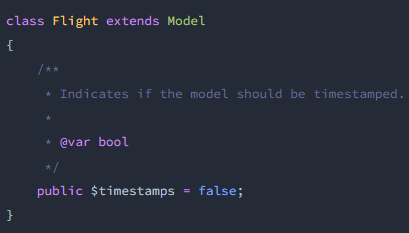
Mặc định Laravel sẽ hiểu khóa chính là **integer** tuy nhiên nếu khóa chính của bạn không phải integer ta sẽ phải cấu hình biến **$keyType** trong Model của Laravel .



**Hình 66:** **Cấu hình loại dữ liệu của khóa chính**

* **Timestamps**

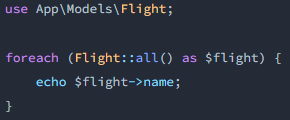
Mặc định Laravel sẽ hiểu và tự động update 2 trường dữ liệu **created\_at** và **update\_at** khi có sự kiện update hoặc tạo ra 1 dữ liệu mới trong bảng bằng Eloquent . Nếu như trong bảng của bạn không có 2 thuộc tính này , ta có thể chỉnh sửa biến **$timestamps** của Model là **true** hoặc **false** . Laravel sẽ hiểu và update hoặc không update 2 trường dữ liệu theo biến $timestamps



**Hình 67:** **Cấu hình created\_at và update\_at**

* **Retrieving Models**

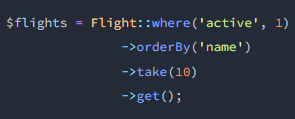
Một khi đã kết nối Model với bảng database thành công . Ta có thể dễ dàng gọi dữ liệu từ database.Đơn giản ta có method all để truy cập tất cả dữ liệu của bảng Model đó kết nối .



**Hình 68:** **Gọi tất cả dữ liệu qua method all**

* **Building Queries**

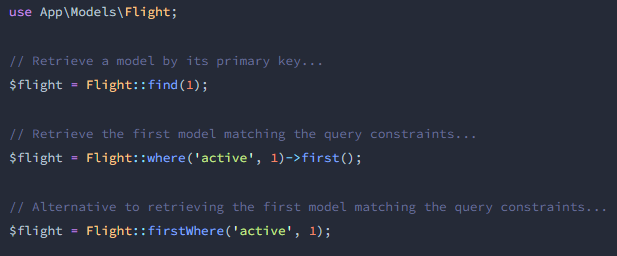
Ta có thể xem Model đã kết nối giống như 1 Query Builder đối với bảng đó. Nó có đầy đủ phương thức để ta thực hiện query



**Hình 69:** **Building Query in Eloquent**

* **Retrieving Single Models**

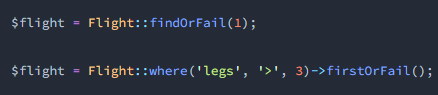
Những method như all , get sẽ trả về 1 mảng dữ liệu . Để trả về 1 hàng dữ liệu ta có thể dùng những method như **find , first , firstWhere**



**Hình 70:** **Sử dụng first , find , firstWhere**

* **Not Found Exception**

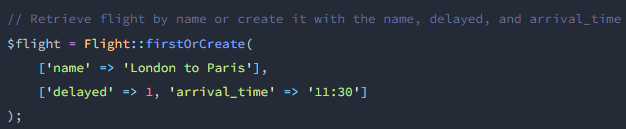
Đôi khi sẽ xảy ra lỗi Not Found Exception nếu model đó không tìm thấy . Những hàm như **findOrFail** , **firstOrFail** sẽ gọi kết quả đầu tiên và kiểm tra nếu không tìm thấy sẽ ném ra 1 lỗi **ModelNotFoundException** . Nếu không bắt lỗi này nó sẽ ném lỗi 404



**Hình 71:** **Sử dụng firstOrFail , findOrFail**

* **Retrieving Or Creating Models**

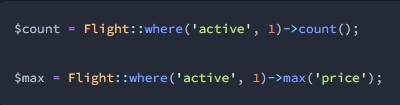
Đôi khi ta muốn gọi dữ liệu ra nhưng nếu không có dữ liệu đó ta muốn tạo luôn dữ liệu đó . Laravel cung cấp các hàm như **firstOrCreate, firstOrNew** . Sau đó bắt buộc ta phải gọi hàm **save** để Laravel lưu vào database .



**Hình 71:** **Sử dụng firstOrCreate**

* **Retrieving Aggregates**

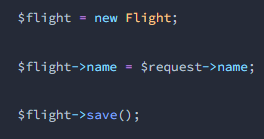
Đôi khi ta muốn gọi dữ liệu với các hàm đếm ta có thể dùng các hàm **count,sum,max**



**Hình 72:** **Sử dụng count , max**

* **Inserts**

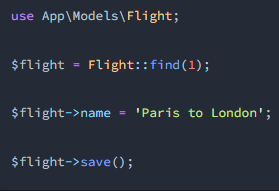
Ta có thể dễ dàng **insert** 1 hàng dữ liệu mới vào database bằng cách tạo đối tượng mới qua Eloquent và trỏ đến các field để điền dữ liệu sau đó dùng hàm **save** để tạo mới đối tượng. Khi insert bằng cách này chú ý biến **$fillable** ở Model biến này sẽ chịu trách nhiệm đảm bảo các field được thêm vào database



**Hình 73:** **Insert bằng Eloquent**

* **Update**

Ta có thể dễ dàng update đối tượng đã tồn tại trong database . Để update đối tượng ta có thể dùng hàm find hoặc where để trỏ đến đúng đối tượng cần update và gọi thuộc tính cần update và sửa thuộc tính . Sau đó dùng hàm **save** để lưu đối tượng



**Hình 74:** **Update bằng Eloquent**

* **Delete Model**

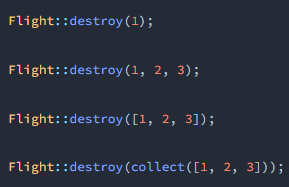
Ta có thể delete model bằng cách trỏ đến đối tượng và dùng hàm **delete .**



**Hình 75:** **Sử dụng hàm delete**

* **Delete Model By Primary Key**

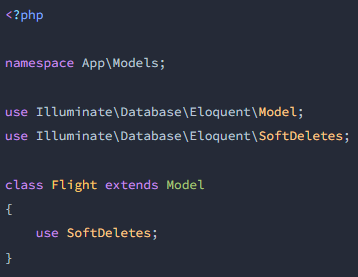
Nếu bạn biết primary key và muốn delete chúng theo primary key ta có thể dùng hàm **destroy**



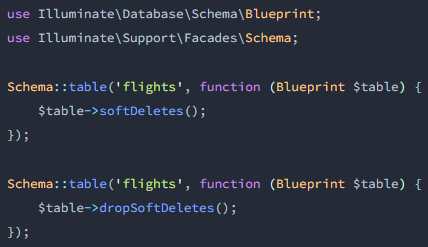
**Hình 76:** **Sử dụng hàm destroy để delete theo id**

* **Soft Deleting**

Nếu bạn muốn delete nhưng không muốn xóa chúng trong database ta có thể dùng hàm Soft Delete. Soft Delete sẽ không thật sự xóa chúng trong database nhưng sẽ active cột **delete\_at** trong database. Để sử dụng Soft Delete ta phải cấu hình Model bằng thuộc tính **use SoftDeletes** . Đồng thời phải tạo thêm cột delete\_at trên database để Laravel sử dụng .



**Hình 77:** **Cấu hình SoftDeletes ở Model**



**Hình 78:** **Tạo cột delete\_at ở database**

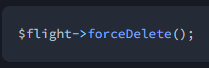
* **Restoring Soft Deleted Models**

Đôi khi bạn muốn khôi phục 1 đối tượng đã delete . Để khôi phục bạn có thể gọi phương thức **restore** . Method này sẽ làm cột delete\_at trở thành null



**Hình 79:** **Khôi phục 1 đối tượng đã Soft Delete**

Đôi khi bạn muốn hoàn toàn delete 1 đối tượng trong database ta sẽ dùng method **forceDelete**



**Hình 80:** **Delete hoàn toàn 1 đối tượng**

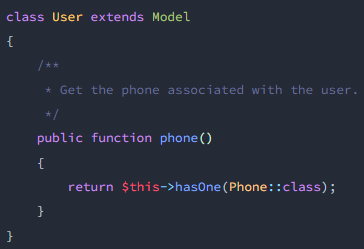
## Eloquent Relationship

* **Giới thiệu**

Database thường liên quan các bảng với nhau. Vì vậy Laravel Eloquent cũng hỗ trợ liên kết các bảng với nhau

* **One to One**

1-1 là định nghĩa cơ bản nhất của mối quan hệ database . Ví dụ : 1 user có 1 phone . Để định nghĩa ta có thể dùng method **hasOne** trong Model



**Hình 81:** **Định nghĩa quan hệ 1-1 trong Model**

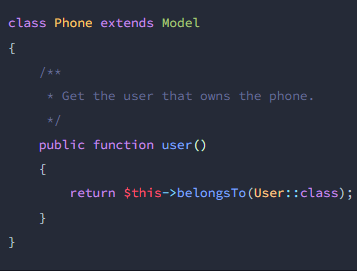


**Hình 82:** **truy vấn số phone của user qua Eloquent**

Eloquent quyết định khóa ngoại dựa trên parent model . Trong trường hợp này class Phone phải có column user\_id . Nếu như table Phone không có user\_id ta phải thêm tham số thứ 2 vào hàm **hasOne** để Laravel hiểu khóa ngoại không phải là user\_id.

* **One to One Inverse**

Ta có thể access Phone Model qua User Model bằng cách định nghĩa mối quan hệ 1-1 nghịch đảo . Dùng belongsTo để định nghĩa nghịch đảo one to one của Phone



**Hình 83:** **Đặt quan hệ 1-1 ngược từ Phone Model**

* **One to Many**

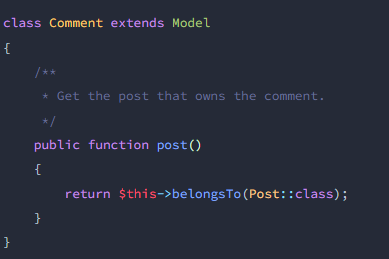
1 to Many là quan hệ định nghĩa 1 parent Model có nhiều child ví dụ 1 blog có nhiều comment



**Hình 84:** **Định nghĩa 1 Post có nhiều comment**

* **One to Many Inverse**

1 to Many đảo ngược là quan hệ định nghĩa Model có nhiều field có khóa ngoại trùng với khóa ngoại của bảng cha.



**Hình 85:** **Định nghĩa 1 Post có nhiều comment đảo ngược**

* **Querying Belongs To Relationships**

Ta có thể dùng method **whereBelongsTo** của Laravel và đưa tham số là id của parent để truy vấn ngược lại một cách dễ dàng.



**Hình 86:** **Định nghĩa 1 Post có nhiều comment đảo ngược**

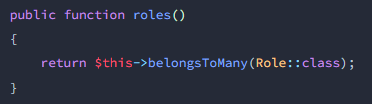
Mặc định Laravel sẽ định nghĩa method sử dụng qua tên Model được bind vào , tuy nhiên nếu cần ta có thể tự định nghĩa method được sử dụng qua tham số thứ 2 của hàm .



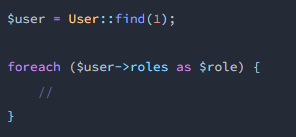
**Hình 87:** **Sử dụng whereBelongsTo với 2 tham số**

* **Many to Many Relationship**

Để định nghĩa quan hệ Many to Many ở Model của class ta sử dụng hàm **belongsToMany**



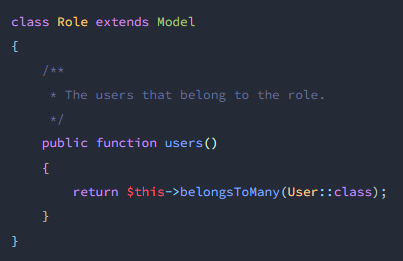
**Hình 88:** **Định nghĩa Many To Many trong Model**



**Hình 89:** **Sử dụng Many To Many để truy vấn**

* **Many To Many Inverse**

Để định nghĩa quan hệ đảo ngược Many to Many ở Model của class ta sử dụng hàm **belongsToMany**



**Hình 90: Định nghĩa belongsToMany Inverse**

## Code Example

* **Giới thiệu**

Sau tất cả các bài cơ bản , bây giờ chúng ta sẽ đi tạo CURD bằng Laravel cho 1 TodoList

* **Chuẩn bị**

+ 1 Model

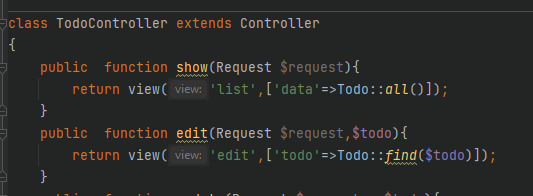
+ 1 Controller

+ 4 route cho 4 method(thêm,xem,xóa,sửa)

+ layout

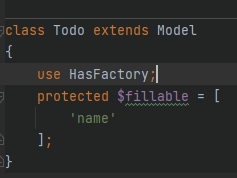
+ Migration

* **Controller**



**Hình 91: Định nghĩa Controller**

* **Model**



**Hình 92: Định nghĩa Model**

* **Route**



**Hình 93: Định nghĩa Route**

* **Kết :**

Các bạn có thể đọc thêm về code mẫu ở github : <https://github.com/tuancutedeptrai/Laravel-Beginer-Step-By-Step>