Xây dựng RESTful API với Laravel

Với các ứng dụng web như hiện nay, việc xây dựng API cho các ứng dụng là không thể thiếu

Laravel cũng hỗ trợ lập trình viên rất mạnh việc xây dựng API

HTTP METHOD

Để xây dựng RESTful API, chúng ta sẽ tìm hiểu về các loại HTTP Method và ý nghĩa từng loại

HTTP Method	**Ý nghĩa**
GET	Được sử dụng để lấy thông tin từ Server theo URI đã cung cấp
HEAD	Giống với GET nhưng Response trả về không có Body, chỉ có Header
POST	Gửi thông tin tới Server, bao gồm Header, Body, Form Data,...
PUT	Ghi đè tất cả thông tin của đối tượng với những gì được gửi lên
PATCH	Ghi đè các thông tin được thay đổi của đối tượng
DELETE	Xoá tài nguyên trên Server
CONNECT	Thiết lập một kết nối tới Server theo URI
OPTIONS	Mô tả các tùy chọn giao tiếp cho Resource
TRACE	Thực hiện một bài Test Loop - Back theo đường dẫn đến Resource

Router RESTful API

Khi xây dựng API với Laravel, chúng ta sẽ làm việc với file `routes/api.php`

Khi các route được khai báo trong file này sẽ có những đặc điểm sau:

- URL sẽ có thêm tiền tố `api`
- Bỏ qua CSRF Token
- Middleware Group mặc định sẽ là api (Tìm hiểu trong file `app/Http/Kernel.php`)

Sau đây là ví dụ CURD

```
// Lấy danh sách sản phẩm
Route::get('products', [ProductsController::class,
'index'])->name('products.index');

// Lấy thông tin sản phẩm theo id
Route::get('products/{id}', [ProductsController::class,
'show'])->name('products.show');

// Thêm sản phẩm mới
Route::post('products', [ProductsController::class,
'store'])->name('products.store');

// Cập nhật thông tin sản phẩm theo id
# Sử dụng put nếu cập nhật toàn bộ các trường
Route::put('products/{id}', [ProductsController::class,
'update'])->name('products.update');

# Sử dụng patch nếu cập nhật 1 vài trường
Route::patch('products/{id}', [ProductsController::class,
'update'])->name('products.update');

// Xóa sản phẩm theo id
Route::delete('products/{id}',
[ProductsController::class,
```

```
'destroy'])->name('products.destroy');
```

Mặc định router đã được gán **middleware bindings**, nếu muốn sử dụng **model binding** trong controller thì chúng ta sửa lại tham số trong router như sau:

```
Route::get('products/{product}',
[ProductsController::class,
'show'])->name('products.show');

Route::put('products/{product}',
[ProductsController::class,
'update'])->name('products.update');

Route::patch('products/{product}',
[ProductsController::class,
'update'])->name('products.update');

Route::delete('products/{product}',
[ProductsController::class,
'destroy'])->name('products.destroy');
```

Ngoài ra, bạn có thể sử dụng **apiResource**

```
Route::apiResource('products',
ProductsController::class);
```

Nếu không muốn sử dụng toàn bộ method trong apiResource mọi người có thể chỉ định sử dụng 1 vài method bằng hàm **only()**

```
Route::apiResource('products',
ProductsController::class)->only(['index', 'show']);
```

Hoặc loại bỏ method không dùng bằng hàm **except()**

```
Route::apiResource('products',
ProductsController::class)->except(['show', 'update']);
```

Controller Resource

Để tạo ra Resource Controllers chúng ta chạy lệnh sau

```
php artisan make:controller ProductsController --api
```

Lúc này file `ProductsController.php` được tạo ra với nội dung như sau:

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

class ProductsController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
```

```
     */
    public function store(Request $request)
    {
        //
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
```

```
    {
        //
    }
}
```

Ngoài ra nếu muốn sử dụng **model binding** khi tạo **Resource Controllers** thì dùng lệnh bên dưới

```
php artisan make:controller ProductsController --api
--model=Models/Product
```

Eloquent Resources

Khi xây dựng API, bạn có thể cần `transform` dữ liệu từ controller trước khi trả về cho người dùng ứng dụng của bạn, laravel cũng đã hỗ trợ điều này với `Eloquent Resources`

Để tạo ra 1 class chuyển đổi chúng ta chạy lệnh sau

```
php artisan make:resource Product
```

File `app/Http/Resources/Product.php` sẽ có nội dung như sau

```
<?php

namespace App\Http\Resources;

use Illuminate\Http\Resources\Json\JsonResource;

class Product extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
```

```
     * @param  \Illuminate\Http\Request  $request
     * @return array
     */
    public function toArray($request)
    {
        return parent::toArray($request);
    }
}
```

Chúng ta sẽ tùy chỉnh dữ liệu trả về là chỉ có **title** và **price**

```
<?php

namespace App\Http\Resources;

use Illuminate\Http\Resources\Json\JsonResource;

class Product extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return array
     */
    public function toArray($request)
    {
        return [
            'title' => $this->title,
            'price' => $this->price,
        ];
    }
}
```

Để sử dụng bên Controller, chúng ta cần sửa lại bên Controller như sau:

```php
//Gọi namespace
use App\Http\Resources\Product as ProductResource;

//Tại vị trí cần sử dụng
ProductResource::collection($products);
```

Ngoài ra, Laravel hỗ trợ rất nhiều thứ. Các bạn tham khảo thêm tại đây: https://laravel.com/docs/8.x/eloquent-resources

HTTP Response Code

Code	Ý nghĩa
200	Ok. Mã cơ bản có ý nghĩa là thành công trong hoạt động
201	Đối tượng được tạo, được dùng trong hàm store
204	Không có nội dung trả về. Hoàn thành hoạt động nhưng sẽ không trả về nội dung
206	Trả lại một phần nội dung, dùng khi sử dụng phân trang
400	Lỗi. Đây là lỗi cơ bản khi không vượt qua được xác nhận yêu cầu từ server
401	Unauthorized. Lỗi do yêu cầu authentication
403	Forbidden. Lỗi này người dùng vượt qua authentication, nhưng không có quyền truy cập.
404	Not found. Không tìm thấy yêu cầu tương tứng
500	Internal server error
503	Service unavailable