

# RESPONSE

Response là các thông tin phản hồi cho người dùng. Qua đó biết được request gửi tới Server có tình trạng như thế nào. Response thường gặp nhất đó là 404 khi truy cập vào một địa chỉ website do mất kết nối thường xuất hiện 404. Trong Laravel tất cả các **route** hay **controller** đều phải trả về một **response**. Laravel có cung cấp sẵn một **Response class** để hỗ trợ trả về các loại response data một cách đơn giản nhất. Lệnh Request có thể được viết tại **Controller** hoặc **route**.

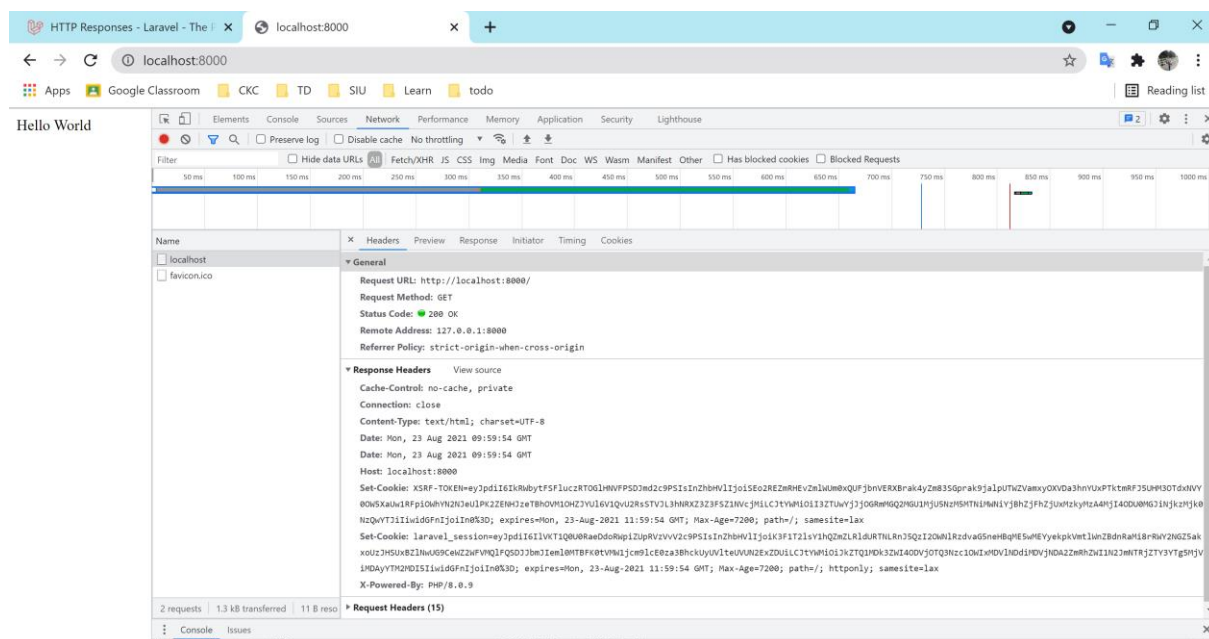
## Tạo response (Creating Response)

### Chuỗi và mảng (String and Array)

Mặc định, Khi một route, controller trả về giá trị là chuỗi thì Laravel sẽ tự động convert nó về dạng một HTTP response.

```
Route::get('/', function () {
    return 'Hello World';
});
```

### Kết quả



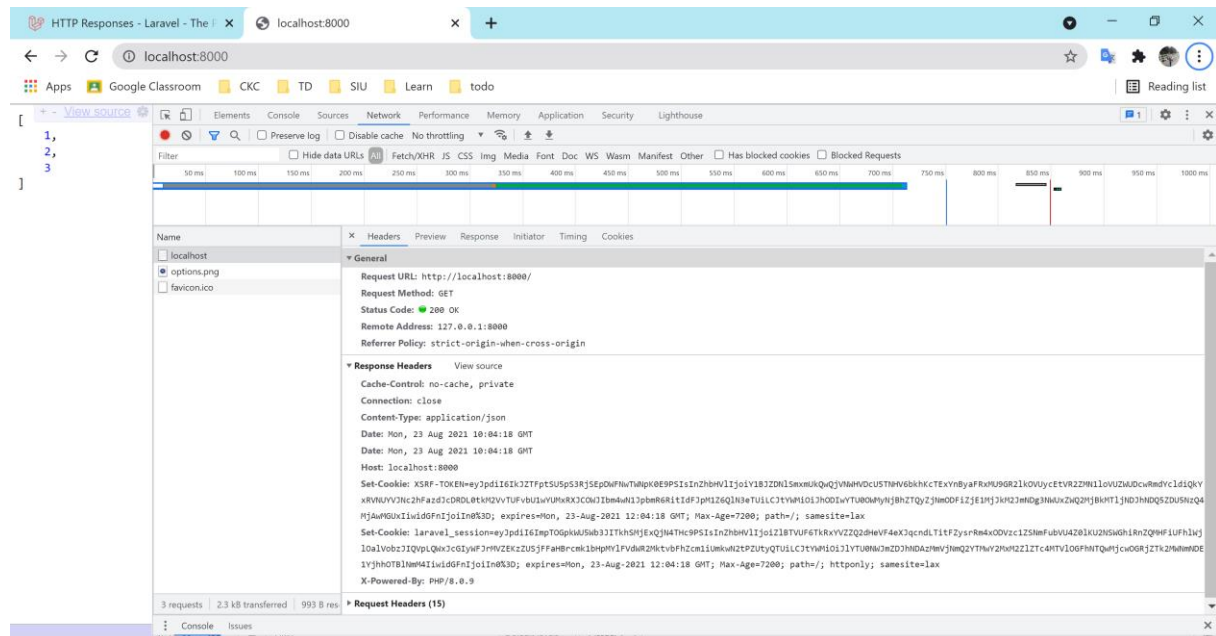
Ngoài việc trả về chuỗi từ route hoặc controller, bạn có thể trả về một mảng. Laravel sẽ tự động chuyển mảng đó về JSON response

```
Route::get('/', function () {

    return [1, 2, 3];

});
```

## Kết quả



## Đối tượng Response (Response Object)

Tuy nhiên, trong thực tế lại cần trả về response với các header, status khác nhau. Khi đó có thể sử dụng Response (`Illuminate\Http\Response`) của Laravel. Đồng thời Laravel cũng có đưa Response class vào helper. Nên hoàn toàn có thể sử dụng hàm `response()` trong helper thay cho Response class để code được ngắn gọn hơn.

```
Route::get('/home', function () {

    return response('Hello World', 200)

        ->header('Content-Type', 'text/plain');

});
```

Thiết lập status code HTTP response là 200 thông qua tham số thứ hai của method `response`. Ngoài ra còn đính kèm header `Content-Type: text/plain` bằng phương thức `header`.

## Kết quả

Tất cả các method response có thể kết nối được, cho phép xây dựng bất kỳ trường hợp response nào. Chẳng hạn có thể thêm hàng loạt header với method **header** trước khi trả về người dùng.

Hoặc thay vì kết nối nhiều method như vậy, có thể sử dụng method `withHeaders` để truyền nhiều header dưới dạng mảng.

Nếu muốn thêm cookie vào response, có thể sử dụng phương thức **cookie** với cú pháp truyền

vào tương tự như đối với hàm `setcookie` của PHP.

```
return response('Hello World')->cookie(
    'name', 'value', $minutes
);
```

The `cookie` method also accepts a few more arguments which are used less frequently. Generally, these arguments have the same purpose and meaning as the arguments that would be given to PHP's native `setcookie` method:

```
return response('Hello World')->cookie(
    'name', 'value', $minutes, $path, $domain, $secure, $httpOnly
);
```

If you would like to ensure that a cookie is sent with the outgoing response but you do not yet have an instance of that response, you can use the `Cookie` facade to "queue" cookies for attachment to the response when it is sent. The `queue` method accepts the arguments needed to create a cookie instance. These cookies will be attached to the outgoing response before it is sent to the browser:

```
use Illuminate\Support\Facades\Cookie;

Cookie::queue('name', 'value', $minutes);
```

## Generating Cookie Instances

If you would like to generate a `Symfony\Component\HttpFoundation\Cookie` instance that can be attached to a response instance at a later time, you may use the global `cookie` helper. This cookie will not be sent back to the client unless it is attached to a response instance:

```
$cookie = cookie('name', 'value', $minutes);

return response('Hello World')->cookie($cookie);
```

## Expiring Cookies Early

Nếu muốn xóa một cookie nào đó trong response, sử dụng phương thức `withoutCookie`.

```
return response('Hello World')->withoutCookie('name');
```

## Cookies & Encryption

By default, all cookies generated by Laravel are encrypted and signed so that they can't be modified or read by the client. If you would like to disable encryption for a subset of cookies generated by your application, you may use the `$except` property of the `App\Http\Middleware\EncryptCookies` middleware, which is located in the `app/Http/Middleware` directory:

```
/**
 * The names of the cookies that should not be encrypted.
 *
 * @var array
 */
protected $except = [
    'cookie_name',
];
```

## Chuyển hướng (Redirect)

Đối với trường hợp muốn trả về một redirect response, Laravel cũng có sẵn một class `Illuminate\Http\RedirectResponse` để làm điều đó. Đồng thời Laravel cũng đã đưa nó vào `helper` để tiện cho việc sử dụng. Có thể sử dụng `helper redirect` thay cho class trên.

## Chuyển hướng đến URI (Redirecting To URI)

Redirect về URL `/home/dashboard` khi người dùng vào URL `/dashboard`.

```
Route::get('/dashboard', function () {
    return redirect('home/dashboard');
});
```

Nếu muốn khi người dùng Submit Form mà bị lỗi thì chuyển về trang trước để người dùng

kiểm tra lại. Sử dụng method `back`, cùng với `withInput` để đáp ứng yêu cầu trên.

```
Route::post('/user/profile', function () {
    // Validate the request...

    return back()->withInput();
});
```

### Chuyển hướng đến route được đặt tên (Redirecting To Named Route)

When you call the `redirect` helper with no parameters, an instance of `Illuminate\Routing\Redirector` is returned, allowing you to call any method on the `Redirector` instance. For example, to generate a `RedirectResponse` to a named route, you may use the `route` method:

Cú pháp:

```
return redirect()->route('name_route');
```

Ví dụ:

```
return redirect()->route('login');

Route::get('/', function() {
    return redirect()->route('home');
});
```

Khi truy cập route `'/'` thì sẽ redirect đến route có name là `home`.

Nếu URI route chứa tham số, có thể truyền dữ liệu thông qua tham số thứ hai dưới dạng mảng.

```
// For a route with the following URI: /profile/{id}

Route::get('/', function() {
    return redirect()->route('profile', ['id' => 1]);
});

Route::get('/profile/{id}', function($id) {
```

```
return 'Profile '. $id;
})->name('profile');
```

Khi truy cập: <http://localhost:8000>. Kết quả là: Profile 1

Nếu không muốn truyền giá trị ID theo mặc định, có thể thay đổi bằng cách khai báo phương thức `getRouteKey` ở trong file model để thay đổi key ID mặc định.

```
public function getRouteKey()
{
    return $this->slug;
}
```

### Chuyển hướng đến Controller Action (Redirecting To Controller Action)

Có thể tạo chuyển hướng đến route chứa controller action thông qua method `action`. Khi đó chỉ cần truyền tên controller và tên method chứa trong route muốn chuyển đến.

[App/Http/Controllers/HomeController.php](#)

```
<?php
namespace App\Http\Controllers;
class HomeController extends Controller
{
    public function show()
    {
        return "Day la HomeController!";
    }
}
?>
```

[routes/web.php](#)

```
use App\Http\Controllers\HomeController;
Route::get('/', function() {
    return redirect()->action([HomeController::class, 'show']);
});
```

```
Route::get('/home', [HomeController::class, 'show']);
```

Khi truy cập: <http://localhost:8000>, chuyển hướng đến <http://localhost:8000/home>.  
Kết quả là: **Đây là HomeController!**

If your controller route requires parameters, you may pass them as the second argument to the `action` method:

```
return redirect()->action(
    [UserController::class, 'profile'], ['id' => 1]
);
```

### Chuyển hướng đến tên miền bên ngoài (Redirecting To External Domain)

Sometimes you may need to redirect to a domain outside of your application. You may do so by calling the `away` method, which creates a `RedirectResponse` without any additional URL encoding, validation, or verification:

```
return redirect()->away('https://www.google.com');
```

### Chuyển hướng với Flash Session (Redirecting With Flash Session)

Redirect đến một URL mới và flash session thường được thực hiện cùng một lúc. Chẳng hạn đăng ký hai route như sau:

`routes/web.php`

```
Route::get('/', function() {
    return redirect('home')->with('name', 'John');
});

Route::get('/home', function() {
    return view('home');
});
```

Tại action của route `/`, ta thực thi chuyển hướng đến URI `/home`, đồng thời thực hiện flash session với method `with`. Sau khi chuyển hướng thì flash session data được thiết lập, lúc này bạn chỉ cần lấy nó ra và sử dụng.

`resources/views/home.blade.php`



```

<h3>
    @if (session('name'))
        Welcome, {{ session('name') }}
    @else
        Welcome
    @endif
</h3>

```

Khi truy cập: <http://localhost:8000>, chuyển hướng đến <http://localhost:8000/home>.  
 Kết quả là: **John**. Nếu truy cập thẳng đến <http://localhost:8000/home>. Kết quả là: **Welcome**.

### Redirecting With Input

You may use the `withInput` method provided by the `RedirectResponse` instance to flash the current request's input data to the session before redirecting the user to a new location. This is typically done if the user has encountered a validation error. Once the input has been flashed to the session, you may easily [retrieve it](#) during the next request to repopulate the form:

```
return back()->withInput();
```

### Các loại Response khác (Other Response Types)

The `response` helper may be used to generate other types of response instances. When the `response` helper is called without arguments, an implementation of the `Illuminate\Contracts\Routing\ResponseFactory` [contract](#) is returned. This contract provides several helpful methods for generating responses.

### View Responses

Nếu cần kiểm soát **status** và **header** của response và cũng muốn trả lại nội dung của **response**, nên sử dụng method **view**:

```

return response()
    ->view('hello', $data, 200)
    ->header('Content-Type', $type);

```

- `$data`: dữ liệu truyền vào views.
- `200`: status code.

Ví dụ

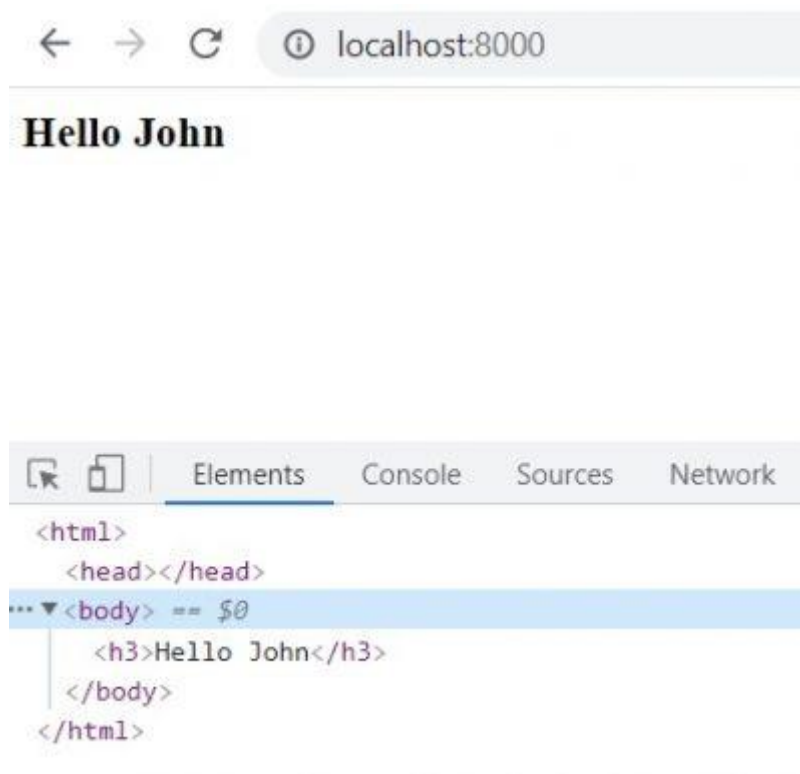
`routes/web.php`

```
Route::get('/', function() {
    return response()->view('hello', $data = ['name'=>'John'], 200)
});
```

`resources/views/hello.blade.php`

```
<h3>Hello {{ $name }}</h3>
```

Kết quả



## JSON Response

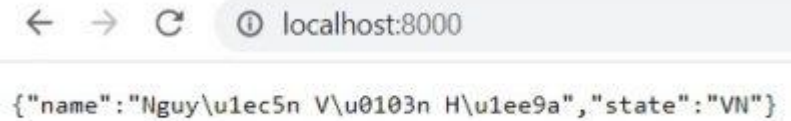
Phương thức `json` sẽ tự động set `Content-type: application/json`, chuyển đổi các mảng về dạng JSON sử dụng phương thức `json_encode` của PHP.

`routes/web.php`

```
Route::get('/', function() {
```

```
return response()->json([
    'name' => 'Nguyen Van Teo',
    'state' => 'VN'
]);
});
```

Kết quả



Nếu muốn tạo JSONP response, có thể sử dụng phương thức `json` cùng với `withCallback`.

```
Route::get('/', function(Request $request) {
    return response()
        ->json(['name' => 'Nguyen Van Teo', 'state' => 'VN'])
        ->withCallback($request->input('callback'));
});
```

## Download file

Method `download` thường được sử dụng để tạo một response bắt buộc người dùng download một file tại đường dẫn nhất định.

```
return response()->download($pathToFile);

return response()->download($pathToFile, $name, $headers);
```

Trong đó:

- `$pathToFile`: là path đến file muốn trả về.
- `$name`: là tên của file muốn trả về.
- `$headers`: là các header muốn trả về kèm theo.

Ví dụ: download file `resources/views/abc.blade.php`

```
Route::get('/download', function() {
    return response()->download('../resources/views/abc.blade.php');
```

```
});
```

Nếu muốn một cái tên khác khi download, có thể thêm tham số thứ hai như sau:

```
Route::get('/download', function() {
    return response()-
>download('../resources/views/abc.blade.php', 'other_file.php');
});
```

Có thể khai báo các header theo dạng cấu trúc mảng

```
$headers = [
    'X-Header-One' => 'Header value 1',
    'X-Header-Two' => 'Header value 2',
    // ...
];
```

Ngoài ra có thể xóa file ngay sau khi người dùng download bằng cách sử dụng method `deleteFileAfterSend`.

```
return response()->download($pathFile, $nameFile, $headers)-
>deleteFileAfterSend();
```

### Streamed download

Đôi khi muốn trả về một nội dung nào đó có thể download được, nhưng lại không muốn ghi trước khi download. Có thể sử dụng method `streamDownload` để trả về một stream data.

```
return response()->streamDownload(Closure, $nameFile, $headers);
```

Trong đó:

- **Closure**: có nhiệm vụ thực hiện xử lý để `echo` nội dung cho file.
- **\$name**: tham số thứ hai là tên file download.
- **\$headers**: tham số thứ ba (tùy chọn) là các thiết lập header.

routes/web.php

```
use App\Services\GitHub;
```

```
return response()->streamDownload(function () {

    echo GitHub::api('repo')

    ->contents()

    ->readme('laravel', 'laravel')['contents'];

}, 'laravel-readme.md');
```

*Lưu ý:* Bắt buộc phải dùng lệnh **echo** thì nội dung mới ghi được trên file download. Nội dung của file chỉ được là dạng text.

### File response

Nếu muốn trả về response của các image, pdf... để hiển thị trên trình duyệt người dùng thay vì tải về, bạn có thể sử dụng method **file**.

```
return response()->file($pathToFile);

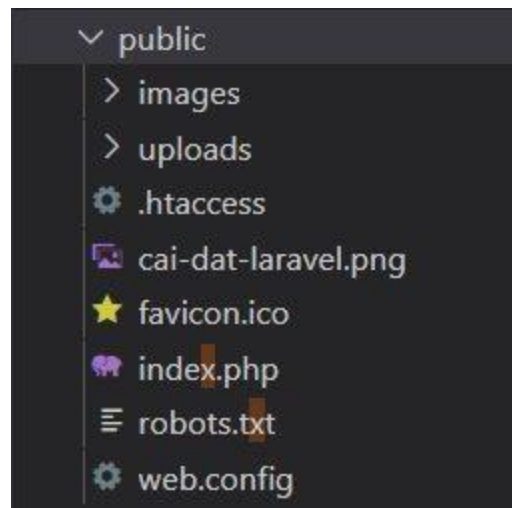
return response()->file($pathToFile, $headers);
```

Trong đó:

- **\$pathFile**: path file cần hiển thị.
- **\$headers**: tùy chọn chứa mảng thiết lập header response.

*Lưu ý:* Base path của đường dẫn file trong method **file** cũng là **public** giống như method **download**.

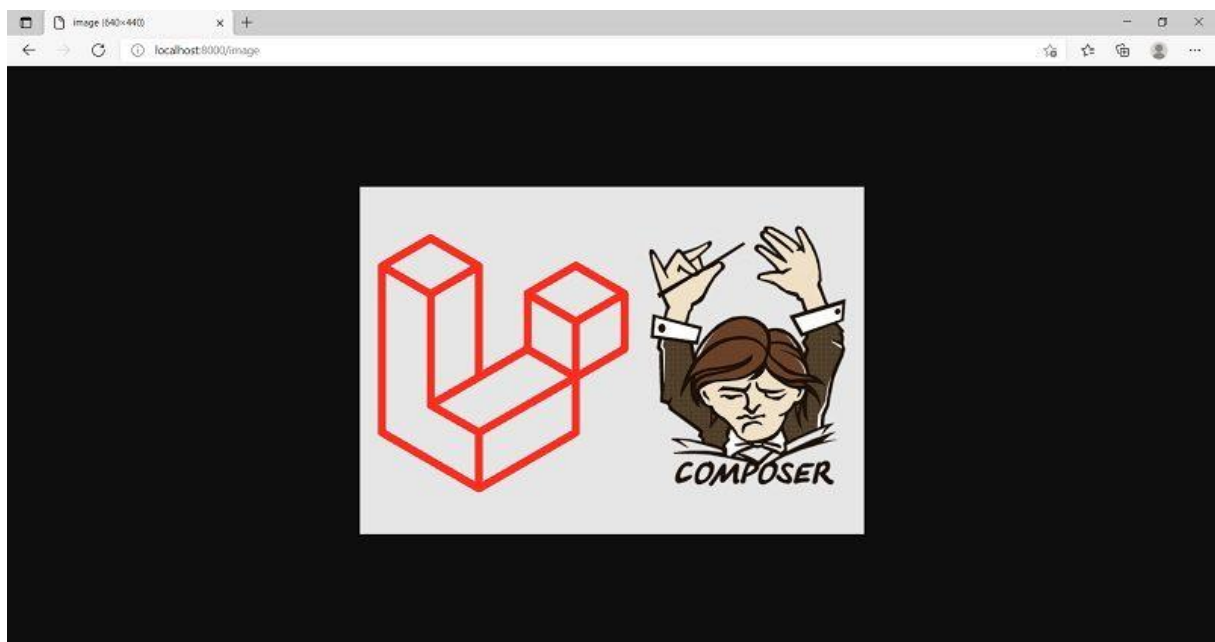
Ví dụ: Thêm một ảnh **cai-dat-laravel.png** vào folder **public**.



routes/web.php

```
Route::get('/image', function() {
    return response()->file('cai-dat-laravel.png');
});
```

Kết quả



### Response macro

Nếu muốn định nghĩa một **response** tùy chỉnh để có thể sử dụng lại trong nhiều **routes** và **controllers**, có thể sử dụng method **macro**.

Tạo một service provider với tên **ResponseMacroServiceProvider** tại method **boot** sẽ thực hiện việc định nghĩa các custom response.

Ví dụ: Tạo một **response** để có thể trả về chuỗi in hoa, mình sẽ đăng ký với **macro** như sau:

## App/providers/ResponseMacroServiceProvider

```
<?php
namespace App\Providers;
use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Response;
class ResponseMacroServiceProvider extends ServiceProvider {
    public function boot()
    {
        Response::macro('caps', function ($value) {
            return Response::make(strtoupper($value));
        });
    }
}
```

Method `Response::macro`: dùng để định nghĩa một custom response:

- Tham số thứ nhất sẽ là tên của custom response.
- Tham số thứ hai sẽ là một Closure, nhận dữ liệu truyền về qua `$value`.

Method `Response::make`: dùng để tạo response, nó sẽ nhận tham số là các chuỗi xử lý `$value` trước khi được trả về.

Liệt kê service provider vừa tạo trong `config/app.php`

```
'providers' => [
    ...
    App\Providers\ResponseMacroServiceProvider::class
],
```

Giờ đây có thể sử dụng response `caps` như những các loại response khác.

## routes/web.php

```
Route::get('caps/{str}', function($str) {
    return response()->caps($str);
})
```

Kết quả

