Course - Laravel Framework

Làm việc với Response

Đối tượng Response sẽ được Laravel dùng làm nền tảng cho việc gửi dữ kiện về ngược lại cho người dùng sau khi họ đã gửi request đến cho ứng dụng Laravel.

Tags: response, laravel

Tạo ra Response

Chuỗi và mảng

Tất cả các route và controller phải trả về một phản hồi để được gửi ngược trở lại trình duyệt của người dùng. Laravel cung cấp một số cách khác nhau để trả lại phản hồi. Phản hồi cơ bản nhất là trả về một chuỗi từ một route hoặc controller. Framework sẽ tự động chuyển đổi chuỗi thành một phản hồi HTTP đầy đủ:

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

Ngoài việc trả về các chuỗi từ các route và controller của bạn, bạn cũng có thể trả về các mảng. Framework sẽ tự động chuyển đổi mảng thành phản hồi JSON:

```
Route::get('/', function () {
  return [1, 2, 3];
});
```

> Bạn có biết rằng bạn cũng có thể trả lại một bộ Eloquent collection từ các route hoặc controller của mình không? Chúng sẽ tự động được chuyển đổi thành JSON. Cho nó một shot!

Đối tượng Response

Thông thường, bạn sẽ không chỉ trả về các chuỗi hoặc mảng đơn giản như vậy từ các action của route. Thay vào đó, bạn sẽ còn trả về đầy đủ đối tượng `Illuminate\Http\Response` hoặc view của nó.

```
Route::get('/home', function () {
  return response('Hello World', 200)->header('Content-Type', 'text/plain');
});
```

Eloquent Model vs Collection

Bạn cũng có thể trả về các model và collection của Eloquent ORM trực tiếp với từ route và controller của mình. Khi bạn làm như vậy, Laravel sẽ tự động chuyển đổi các model và collection thành phản hồi JSON trong khi vẫn giữ được các thuộc tính ẩn của model:

```
use App\Models\User;

Route::get('/user/{user}', function (User $user) {
  return $user;
});
```

Đính kèm header vào response

Hãy ghi nhớ rằng hầu hết các phương thức phản hồi đều có thể làm việc với nhau, cho phép xây dựng hoàn thiện các đối tượng `response`. Ví dụ: bạn có thể sử dụng phương thức `header` để thêm một nhóm header vào response trước khi gửi lại cho người dùng:

```
return response($content)
  ->header('Content-Type', $type)
  ->header('X-Header-One', 'Header Value')
  ->header('X-Header-Two', 'Header Value');
```

Hoặc, bạn có thể sử dụng phương thức `withHeaders` để khai báo một mảng header sẽ được thêm vào response:

```
return response($content)
  ->withHeaders([
    'Content-Type' => $type,
    'X-Header-One' => 'Header Value',
    'X-Header-Two' => 'Header Value',
  ]);
```

Middleware Cache-Control

Laravel có một middleware `cache.headers`, nó có thể được sử dụng để nhanh chóng thiết lập header `Cache-Control` cho một nhóm các route. Các chỉ thị phải được cung cấp bằng cách sử dụng "snake case" tương đương với chỉ thị cache-control tương ứng và phải

được tách biệt bằng dấu chấm phẩy. Nếu `etag` được mô tả trong danh sách chỉ thị, thì mã hash MD5 của nội dung response sẽ tự động được đặt làm định danh ETag:

```
Route::middleware('cache.headers:public;max_age=2628000;etag')->group(function () {
  Route::get('/privacy', function () {
    // ...
  });

  Route::get('/terms', function () {
    // ...
  });
});
```

Đính kèm cookie vào trong response

Bạn có thể đính kèm cookie vào đối tượng `Illuminate\Http\Response` gửi đi bằng phương thức `cookie`. Bạn nên truyền tên, giá trị và thời hạn cookie (dạng phút) hợp lệ cho phương pháp này:

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

Phương thức `cookie` cũng cho truyền một số đối số ít khi được sử dụng. Nói chung, các đối số này có cùng mục đích và ý nghĩa với các đối số sẽ được truyền vào phương thức `setcookie` của PHP:

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

Nếu bạn muốn đảm bảo rằng một cookie được gửi cùng với response, nhưng bạn chưa có đối tượng của response đó, bạn có thể sử dụng facade `Cookie` để bố trí cookie cho việc đính kèm vào response khi nó được gửi đi. Phương thức `queue` cho phép truyền các đối số cần thiết để tạo một đối tượng cookie. Các cookie này sẽ được đính kèm vào response gửi đi trước khi nó được gửi đến trình duyệt:

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

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

Tạo ra các đối tượng cookie

Nếu bạn muốn tạo một đối tượng `Symfony\Component\HttpFoundation\Cookie` có thể được đính kèm vào một đối tượng response sau này, thì bạn có thể sử dụng hàm trợ giúp toàn cục `cookie`. Cookie này sẽ không được gửi lại cho người dùng trừ khi nó được đính kèm với một đối tượng response:

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

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

Kết thúc cookie

Bạn có thể xóa một cookie bằng cách làm nó hết hạn với phương thức `withoutCookie` của một response gửi đi:

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

Nếu bạn chưa có đối tượng response để gửi đi, thì bạn có thể sử dụng phương thức `expire` của facade `Cookie` để kết thúc cookie:

```php
Cookie::expire('name');
```

Cookie và Mã hoá

Mặc định, tất cả cookie do Laravel tạo ra đều được mã hóa và đánh dấu để phía máy khách không thể sửa đổi hoặc đọc được chúng. Nếu bạn muốn tắt mã hóa cho một phần dữ liệu cookie được tạo bởi ứng dụng của mình, thì bạn có thể sử dụng thuộc tính `$except` của middleware `App\Http\Middleware\EncryptCookies`, nằm trong thư mục `app/Http/Middleware`:

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

Chuyển hướng trang

Các phản hồi chuyển hướng trang là các đối tượng của class `Illuminate\Http\RedirectResponse`, nó chứa các header thích hợp để chuyển hướng người dùng đến một URL khác. Có nhiều cách để tạo ra một đối tượng `RedirectResponse`. Phương pháp đơn giản nhất là sử dụng hàm trợ giúp toàn cục `redirect`:

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

Đôi khi bạn có thể muốn chuyển hướng người dùng đến vị trí trước đó của họ, chẳng hạn như khi HTML form đã gửi không hợp lệ. Bạn có thể làm như vậy bằng cách sử dụng hàm trợ giúp toàn cục `back`. Vì tính năng này sử dụng session, nên hãy đảm bảo rằng route gọi hàm `back` đang sử dụng nhóm middleware `web`:

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

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

Chuyển hướng bằng tên route

Khi bạn gọi hàm `redirect` mà không có tham số, một đối tượng của `Illuminate\Routing\Redirector` sẽ được trả về, cho phép bạn gọi bất kỳ phương

thức nào trên đối tượng `Redirector`. Ví dụ: để tạo `RedirectResponse` cho một route đã được đặt tên, thì bạn có thể sử dụng phương thức `route`:

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

Nếu route của bạn có các tham số, thì bạn có thể truyền chúng vào đối số thứ hai cho phương thức `route`:

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

return redirect()->route('profile', ['id' => 1]);
```

Tham số route với các model Eloquent

Nếu bạn đang chuyển hướng đến một route có tham số `"ID"` đang tham chiếu đến một model Eloquent, thì bạn có thể truyền chính model đó. ID sẽ được trích xuất tự động:

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

return redirect()->route('profile', [$user]);
```

Nếu bạn muốn tùy chỉnh giá trị được đặt trong tham số route, bạn có thể chỉ định cột dữ liệu khi khai báo route có tham số (Ví dụ: `/profile/{id:slug}`) hoặc bạn có thể ghi đè phương thức `getRouteKey` trên Eloquent model của mình:

```
/**
 * Get the value of the model's route key.
 *
 * @return mixed
 */
public function getRouteKey()
{
  return $this->slug;
}
```

Chuyển hướng đến action

Bạn cũng có thể chuyển hướng trang đến các action của controller. Để làm được như vậy, hãy truyền controller và tên action cho phương thức `action`:

```
use App\Http\Controllers\UserController;

return redirect()->action([UserController::class, 'index']);
```

Nếu route sử dụng controller yêu cầu các tham số, thì bạn có thể truyền vào các tham số làm đối số thứ hai cho phương thức `action`:

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

Chuyển hướng đến tên miền bên ngoài

Đôi khi bạn có thể cần phải chuyển hướng đến một tên miền bên ngoài ứng dụng của mình. Bạn có thể làm như vậy bằng cách gọi phương thức `away`, phương thức này tạo ra `RedirectResponse` mà không cần mã hóa, xác thực hoặc xác minh URL bổ sung nào:

```
return redirect()->away('https://www.example.ext');
```

Chuyển hướng với dữ liệu flash session

Chuyển hướng đến một URL mới và flash dữ liệu đến session thường được thực hiện cùng một lúc. Thông thường, điều này được thực hiện sau khi thực hiện thành công một hành động khi bạn gửi thông báo thành công cho session. Để thuận tiện, bạn có thể tạo một đối tượng `RedirectResponse` và flash dữ liệu cho session trong một chuỗi phương thức đơn nhất, liên tục:

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

```
    return redirect('dashboard')->with('status', 'Profile updated!');
});
```

Sau khi người dùng được chuyển hướng, bạn có thể hiển thị thông báo trong flash session. Ví dụ, sử dụng cú pháp của Blade:

```
@if (session('status'))
    <div class="alert alert-success">
        {{ session('status') }}
    </div>
@endif
```

Chuyển hướng với dữ liệu đầu vào

Bạn có thể sử dụng phương thức `withInput` được cung cấp bởi đối tượng `RedirectResponse` để flash dữ liệu đầu vào của request hiện tại vào session trước khi chuyển hướng người dùng đến một vị trí mới. Điều này thường được thực hiện nếu người dùng gặp lỗi xác thực. Khi đầu vào đã được chuyển vào session, bạn có thể dễ dàng truy xuất nó trong lần request tiếp theo để tái hiện lại HTML form:

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

Các kiểu response khác

Hàm trợ giúp toàn cục `response` có thể được sử dụng để tạo các loại đối tượng Response khác nhau. Khi hàm `response` được gọi mà không có đối số, việc triển khai đối tượng hợp đồng `Illuminate\Contracts\Routing\ResponseFactory` sẽ được trả về. Hợp đồng này cung cấp một số phương thức hữu dụng để tạo phản hồi.

Hiển thị response

Nếu bạn cần kiểm soát status và header của response nhưng cũng cần trả lại phần hiển thị dưới dạng nội dung của response, thì bạn nên sử dụng phương thức `view`:

```
return response()
```

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

Tất nhiên, nếu bạn không cần truyền mã status HTTP tùy chỉnh hoặc các header tùy chỉnh, thì bạn có thể sử dụng hàm trợ giúp toàn cục `view`.

Response dữ liệu JSON

Phương thức `json` sẽ tự động đặt header Content-Type thành `application/json`, cũng như chuyển đổi mảng đã cho thành dữ liệu JSON bằng cách sử dụng hàm `json_encode` của PHP:

```
return response()->json([
    'name' => 'Abigail',
    'state' => 'CA',
]);
```

Nếu bạn muốn tạo ra một JSONP response, thì bạn có thể sử dụng phương thức `json` kết hợp với phương thức `withCallback`:

```
return response()
    ->json(['name' => 'Abigail', 'state' => 'CA'])
    ->withCallback($request->input('callback'));
```

Download tập tin

Phương thức download có thể được sử dụng để tạo ra một response buộc trình duyệt của người dùng tải xuống tập tin theo đường dẫn nào đó. Phương thức download cho phép truyền tên tập tin làm đối số thứ hai cho phương thức, sẽ xác định tên tập tin nào được nhìn thấy bởi người dùng trong lúc đang tải tập tin. Cuối cùng, bạn có thể truyền một mảng HTTP header làm đối số thứ ba cho phương thức:

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

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

> **Chú ý:** Symfony HttpFoundation, quản lý tải xuống tập tin, yêu cầu tập tin được tải xuống phải có tên dạng ASCII.

Download với stream

Đôi khi bạn có thể muốn chuyển response dạng chuỗi của một hành động nào đó thành một response có thể tải xuống được mà không cần phải ghi nội dung của hành động vào đĩa. Bạn có thể sử dụng phương thức `streamDownload` trong trường hợp này. Phương thức này chấp nhận một hàm callback, tên tập tin và một mảng header tùy chọn làm đối số của nó:

```
use App\Services\GitHub;

return response()->streamDownload(function () {
  echo GitHub::api('repo')
    ->contents()
    ->readme('laravel', 'laravel')['contents'];
}, 'laravel-readme.md');
```

Response tập tin

Phương thức `file` có thể được sử dụng để hiển thị tập tin, chẳng hạn như hình ảnh hoặc PDF lên trực tiếp trình duyệt của người dùng thay vì bắt đầu tải xuống. Phương thức này chấp nhận đường dẫn đến tập tin làm đối số đầu tiên và một mảng header làm đối số thứ hai:

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

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

Nếu bạn muốn xác định một response tùy chỉnh mà bạn có thể sử dụng lại nhiều lần với các route và controller của mình, bạn có thể sử dụng phương thức `macro` trên facade `Response`. Thông thường, bạn nên gọi phương thức này từ phương thức `boot` của một trong các nhà cung cấp dịch vụ ứng dụng của bạn, chẳng hạn như nhà cung cấp dịch vụ `App\Providers\AppServiceProvider`:

```php
<?php

namespace App\Providers;

use Illuminate\Support\Facades\Response;
use Illuminate\Support\ServiceProvider;

class AppServiceProvider extends ServiceProvider
{
  /**
   * Bootstrap any application services.
   *
   * @return void
   */
  public function boot()
  {
    Response::macro('caps', function ($value) {
      return Response::make(strtoupper($value));
    });
  }
}
```

Hàm `macro` cho phép truyền một tên của macro làm đối số đầu tiên của nó và một hàm nặc danh làm đối số thứ hai của nó. Việc hàm macro sẽ được thực thi khi gọi tên macro từ việc thực thi `ResponseFactory` hoặc hàm trợ giúp `response`:

```php
return response()->caps('foo');
```