Course - Laravel Framework

Cấu tạo URL

Laravel cung cấp một số hàm trợ giúp để hỗ trợ bạn trong việc tạo URL cho ứng dụng của bạn. Những hàm trợ giúp này chủ yếu giúp tạo liên kết trong các template và response API của bạn hoặc khi tạo response chuyển hướng đến phần khác của ứng dụng.

Tags: cau tao url, laravel

Căn bản

Tạo ra URL

Hàm trợ giúp `url` có thể được sử dụng để tạo các URL cho ứng dụng của bạn. URL được tạo sẽ tự động sử dụng lược đồ (HTTP hoặc HTTPS) và máy chủ lưu trữ từ request hiện tại đang được ứng dụng xử lý:

```
$post = App\Models\Post::find(1);

echo url("/posts/{$post->id}");

// http://example.com/posts/1
```

Truy cập URL hiện tại

Nếu không có đường dẫn nào được cung cấp cho hàm trợ giúp `url`, một phiên bản `Illuminate\Routing\UrlGenerator` sẽ được trả về, cho phép bạn truy cập thông tin về URL hiện tại:

```
// Get the current URL without the query string...
echo url()->current();

// Get the current URL including the query string...
echo url()->full();

// Get the full URL for the previous request...
echo url()->previous();
```

Mỗi phương thức trên cũng có thể được truy cập thông qua facade `URL`:

```
use Illuminate\Support\Facades\URL;

echo URL::current();
```

URL cho named route (route đã được đặt tên)

Hàm trợ giúp `route` có thể được sử dụng để tạo URL cho các route được đặt tên. Các route được đặt tên cho phép bạn tạo các URL mà không bị trùng lặp với URL thực được xác định trên route. Do đó, nếu URL trên route thay đổi, thì không cần thay đổi các thông tin trên hàm `route`. Ví dụ: hãy tưởng tượng ứng dụng của bạn chứa một route được xác định như sau:

```
Route::get('/post/{post}', function (Post $post) {
  //
})->name('post.show');
```

Để tạo một URL cho route này, bạn có thể sử dụng hàm trợ giúp `route` như sau:

```
echo route('post.show', ['post' => 1]);

// http://example.com/post/1
```

Tất nhiên, hàm trợ giúp `route` cũng có thể được sử dụng để tạo URL cho các route có nhiều tham số:

```
Route::get('/post/{post}/comment/{comment}', function (Post $post, Comment $comment) {
  //
})->name('comment.show');

echo route('comment.show', ['post' => 1, 'comment' => 3]);

// http://example.com/post/1/comment/3
```

Bất kỳ phần tử nào trong mảng bổ sung mà không tương ứng với các tham số đã được khai báo trong route sẽ được thêm vào chuỗi truy vấn của URL:

```
echo route('post.show', ['post' => 1, 'search' => 'rocket']);

// http://example.com/post/1?search=rocket
```

Model Eloquent

Bạn thường sẽ tạo URL bằng cách sử dụng route key (thường là primary key) của các model Eloquent. Vì lý do này, bạn có thể chuyển các model Eloquent làm giá trị tham số. Hàm trợ giúp `route` sẽ tự động trích xuất route key của model:

```
echo route('post.show', ['post' => $post]);
```

Các signed URL (URL đã được đánh dấu)

Laravel cho phép bạn dễ dàng tạo các signed URL đến các route được đặt tên. Các URL này có thêm một mã hash "signatured" vào chuỗi truy vấn cho phép Laravel xác minh rằng URL không bị sửa đổi kể từ khi nó được tạo. Các signed URL cực kỳ hữu ích cho các route có thể truy cập công khai nhưng cần một lớp bảo vệ chống lại việc thao túng URL.

Ví dụ: bạn có thể sử dụng các URL đã đánh dấu để triển khai liên kết "hủy đăng ký" công khai được gửi qua email cho khách hàng của bạn. Để tạo một URL đã được đánh dấu cho một route được đặt tên, hãy sử dụng phương thức `signedRoute` của facade `URL`:

```
use Illuminate\Support\Facades\URL;

return URL::signedRoute('unsubscribe', ['user' => 1]);
```

Nếu bạn muốn tạo một route URL chỉ đánh dấu tạm thời và sẽ hết hạn sau một khoảng thời gian nhất định, bạn có thể sử dụng phương thức `temporarySignedRoute`. Và khi Laravel kiểm tra, nó sẽ đảm bảo rằng thời hạn này chưa hết thời hạn:

```
use Illuminate\Support\Facades\URL;

return URL::temporarySignedRoute(
  'unsubscribe', now()->addMinutes(30), ['user' => 1]
);
```

Kiểm tra signed route

Để xác minh một request gửi đến đã được đánh dấu một cách hợp lệ, bạn nên gọi phương

thức `hasValidSignature` trên request gửi đến:

```
use Illuminate\Http\Request;

Route::get('/unsubscribe/{user}', function (Request $request) {
  if (! $request->hasValidSignature()) {
    abort(401);
  }

  // ...
})->name('unsubscribe');
```

Ngoài ra, bạn có thể gán middleware `Illuminate\Routing\Middleware\ValidateSignature` cho route. Nếu nó chưa xuất hiện, bạn nên gán cho middleware này một cái key trong mảng `routeMiddleware` của HTTP kernel của bạn:

```
/**
 * The application's route middleware.
 *
 * These middleware may be assigned to groups or used individually.
 *
 * @var array
 */
protected $routeMiddleware = [
  'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
];
```

Khi bạn đã đăng ký middleware trong kernel của mình, bạn có thể đính kèm nó vào một route nào đó. Nếu request gửi đến không được đánh dấu một cách hợp lệ, thì middleware sẽ tự động trả về phản hồi HTTP 403:

```
Route::post('/unsubscribe/{user}', function (Request $request) {
  // ...
})->name('unsubscribe')->middleware('signed');
```

Phản hồi signed route không hợp lệ

Khi ai đó truy cập vào signed URL đã hết hạn, họ sẽ nhận được trang lỗi với mã trạng thái HTTP 403. Tuy nhiên, bạn có thể tùy chỉnh bằng cách tạo ra một hàm nặc danh với exception `InvalidSignatureException` trong hàm xử lý exception của bạn. Hàm này sẽ trả về một HTTP response:

```
use Illuminate\Routing\Exceptions\InvalidSignatureException;

/**
 * Register the exception handling callbacks for the application.
 *
 * @return void
 */
public function register()
{
  $this->renderable(function (InvalidSignatureException $e) {
    return response()->view('error.link-expired', [], 403);
  });
}
```

URL cho controller action

Hàm `action` sẽ tạo một URL cho controller action nào đó:

```
use App\Http\Controllers\HomeController;

$url = action([HomeController::class, 'index']);
```

Nếu phương thức của controller chấp nhận các tham số route, thì bạn có thể truyền một mảng các tham số route làm đối số thứ hai cho hàm:

```
$url = action([UserController::class, 'profile'], ['id' => 1]);
```

Các giá trị mặc định

Đối với một số ứng dụng, bạn có thể muốn chỉ định các giá trị mặc định cho toàn bộ request với tham số URL. Ví dụ: hãy tưởng tượng nhiều route của bạn chứa tham số `{locale}`:

```
Route::get('/{locale}/posts', function () {
  //
})->name('post.index');
```

Thật là rườm rà khi bạn luôn phải khai báo ngôn ngữ mỗi khi bạn gọi hàm route. Vì vậy, bạn có thể sử dụng phương thức `URL::defaults` để xác định giá trị mặc định cho tham số này sao cho nó sẽ luôn được áp dụng trong request hiện tại. Bạn có thể muốn gọi phương thức này từ một middleware route để bạn truy cập vào request hiện tại:

```
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\URL;

class SetDefaultLocaleForUrls
{
  /**
   * Handle the incoming request.
   *
   * @param  \Illuminate\Http\Request  $request
   * @param  \Closure  $next
   * @return \Illuminate\Http\Response
   */
  public function handle($request, Closure $next)
  {
    URL::defaults(['locale' => $request->user()->locale]);

    return $next($request);
  }
}
```

Khi giá trị mặc định cho thông số `{locale}` đã được đặt, thì bạn không còn bắt buộc phải

truyền giá trị của nó mỗi khi tạo URL thông qua hàm `route` nữa.

Giá trị mặc định và thứ tự ưu tiên

Việc đặt các giá trị mặc định của URL có thể cản trở việc xử lý các ràng buộc model ngầm định của Laravel. Do đó, bạn nên ưu tiên cho middleware đặt các giá trị mặc định URL sao cho sẽ được thực thi trước middleware `SubstituteBindings` của chính Laravel. Bạn có thể thực hiện điều này bằng cách đảm bảo rằng middleware của bạn xuất hiện trước middleware `SubstituteBindings` trong thuộc tính `$middlewarePriority` của HTTP kerrnel trong ứng dụng của bạn.

Thuộc tính `$middlewarePriority` được định nghĩa trong lớp `Illuminate\Foundation\Http\Kernel` cơ sở. Bạn có thể sao chép khai báo của nó từ class đó và viết lại nó trong HTTP kernel của ứng dụng để sửa đổi nó:

```
/**
 * The priority-sorted list of middleware.
 *
 * This forces non-global middleware to always be in the given order.
 *
 * @var array
 */
protected $middlewarePriority = [
  // ...
  \App\Http\Middleware\SetDefaultLocaleForUrls::class,
  \Illuminate\Routing\Middleware\SubstituteBindings::class,
  // ...
];
```