Course - Laravel Framework

Xử lý lỗi

Khi bạn bắt đầu một dự án Laravel mới, việc xử lý lỗi và exception đã được cài đặt cho bạn. class App\Exceptions\Handler là nơi tất cả các ngoại lệ do ứng dụng của bạn đưa ra đều được ghi lại và sau đó được hiển thị cho người dùng.

Tags: error, laravel

Cấu hình

Tùy chọn gỡ lỗi trong tệp cấu hình `config/app.php` của bạn xác định lượng thông tin về lỗi sẽ thực sự được hiển thị cho người dùng. Mặc định, tùy chọn này được đặt để kết nối đến giá trị của biến môi trường `APP_DEBUG`, được lưu trữ trong tệp `.env` của bạn.

Trong quá trình phát triển project tại máy tính cục bộ, bạn nên đặt biến môi trường `APP_DEBUG` thành `true`. Trong môi trường production của bạn, giá trị này phải luôn luôn là `false`. Nếu giá trị được đặt thành `true` trong phiên bản production, bạn có nguy cơ để lộ các thông tin cấu hình nhạy cảm cho người dùng của ứng dụng của mình.

Xử lý Exception

Hiển thị Exception

Tất cả các Exception được xử lý bởi lớp `App\Exceptions\Handler`. Class này chứa một phương thức `register` nơi bạn có thể đăng ký báo cáo exception và hiển thị callback. Chúng ta sẽ xem xét chi tiết từng khái niệm này. Báo cáo exception được sử dụng để ghi lại các exception hoặc gửi chúng đến một dịch vụ bên thứ ba như Flare, Bugsnag hoặc Sentry. Mặc định, các exception sẽ được ghi lại dựa trên cấu hình logging của bạn. Tuy nhiên, bạn có thể tự do ghi lại các ngoại lệ theo cách bạn muốn.

Ví dụ: nếu bạn cần báo cáo các loại exception khác nhau theo những cách khác nhau, bạn có thể sử dụng phương thức `reportable` để đăng ký một hàm nặc danh sẽ được thực thi khi một exception của một loại đã cho cần được báo cáo. Laravel sẽ dò ra loại exception nào mà hàm nặc danh sẽ báo cáo bằng cách kiểm tra kiểu đã được khai báo trong hàm nặc danh:

```
use App\Exceptions\InvalidOrderException;

/**
 * Register the exception handling callbacks for the application.
 *
 * @return void
 */
public function register()
{
  $this->reportable(function (InvalidOrderException $e) {
    //
```

```
    });
}
```

Khi bạn đăng ký một callback báo cáo exception tùy chỉnh bằng phương thức `reportable`, Laravel sẽ vẫn ghi lại exception bằng cách sử dụng cấu hình ghi nhật ký mặc định cho ứng dụng. Nếu bạn muốn dừng bố trí exception cho ngăn xếp logging mặc định, bạn có thể sử dụng phương thức `stop` khi chỉ định callback báo cáo của mình hoặc trả về `false` từ callback:

```
$this->reportable(function (InvalidOrderException $e) {
    //
})->stop();

$this->reportable(function (InvalidOrderException $e) {
    return false;
});
```

> **Chú ý:** Để tùy chỉnh báo cáo exception cho một exception nào đó, bạn cũng có thể vận dụng các ngoại lệ có thể báo cáo..

Ghi chép tóm lượt

Nếu có, Laravel sẽ tự động thêm ID của người dùng hiện tại vào mọi thông điệp ghi chép của exception dưới dạng dữ liệu ghi chép tóm lượt. Bạn có thể xác định dữ liệu ghi chép tóm lượt của riêng mình bằng cách khai báo lại phương thức context của class `App\Exceptions\Handler` của ứng dụng. Thông tin này sẽ được bao gồm trong mọi thông điệp ghi chép của exception được viết bởi ứng dụng của bạn:

```
/**
 * Get the default context variables for logging.
 *
 * @return array
 */
protected function context()
{
    return array_merge(parent::context(), [
```

```
    'foo' => 'bar',
  ]);
}
```

Ghi chép trong exception

Mặc dù việc thêm phạm vị, bối cảnh vào mọi thông điệp ghi chép có thể có ích, nhưng đôi khi một exception nào đó có thể có một bối cảnh thống nhất mà bạn muốn đưa vào ghi chép của mình. Bằng cách chỉ định phương thức `context` trên một trong các exception tùy chỉnh của ứng dụng, bạn có thể chỉ định bất kỳ dữ liệu nào liên quan đến exception đó mà sẽ được thêm vào mục ghi chép của exception:

```
<?php
namespace App\Exceptions;
use Exception;
class InvalidOrderException extends Exception
{
  // ...
  /**
   * Get the exception's context information.
   *
   * @return array
   */
  public function context()
  {
    return ['order_id' => $this->orderId];
  }
}
```

Hàm `report`

Đôi khi bạn có thể cần phải báo cáo một exception nhưng vẫn tiếp tục xử lý yêu cầu hiện tại. Hàm `report` cho phép bạn nhanh chóng báo cáo một exception với chương trình xử lý exception mà không hiển thị trang lỗi cho người dùng:

```
public function isValid($value)
```

```
{
  try {
    // Validate the value...
  } catch (Throwable $e) {
    report($e);
    return false;
  }
}
```

Bỏ qua exception

Khi xây dựng ứng dụng của bạn, sẽ có một số loại exception mà bạn chỉ muốn bỏ qua và không bao giờ báo cáo. Chương trình xử lý exception của ứng dụng của bạn có chứa thuộc tính `$dontReport` được khởi tạo với một mảng trống. Bất kỳ class nào mà bạn thêm vào thuộc tính này sẽ không bao giờ được báo cáo; tuy nhiên, chúng vẫn có thể có logic hiển thị tùy chỉnh:

```
use App\Exceptions\InvalidOrderException;

/**
 * A list of the exception types that should not be reported.
 *
 * @var array
 */
protected $dontReport = [
  InvalidOrderException::class,
];
```

> Ở phía sau, Laravel đã bỏ qua một số loại lỗi cho bạn, chẳng hạn như các trường hợp exception do lỗi 404 HTTP "không tìm thấy" hoặc 419 HTTP response được tạo do nó có mã token CSRF không hợp lệ.

Trình bày exception

Mặc định, chương trình xử lý exception của Laravel sẽ chuyển đổi exception thành HTTP response cho bạn. Tuy nhiên, bạn có thể tự do đăng ký hàm nặc danh nào đó để trình bày

các trường hợp exception. Bạn có thể thực hiện điều này thông qua phương thức `renderable` của chương trình xử lý exception của bạn.

Hàm nặc danh được truyền cho phương thức `renderable` sẽ trả về một đối tượng của `Illuminate\Http\Response` mà có thể đã được tạo với hàm `response`. Laravel sẽ dò ra kiểu exception nào mà hàm nặc danh sẽ hiển thị bằng cách kiểm tra kiểu exception đã được khai báo trên hàm nặc danh:

```
use App\Exceptions\InvalidOrderException;
/**
 * Register the exception handling callbacks for the application.
 *
 * @return void
 */
public function register()
{
  $this->renderable(function (InvalidOrderException $e, $request) {
    return response()->view('errors.invalid-order', [], 500);
  });
}
```

Bạn cũng có thể sử dụng phương `renderable` để tạo lại cách thức trình bày cho các exception của Laravel hoặc Symfony chẳng hạn như `NotFoundHttpException`. Nếu hàm nặc danh được truyền cho phương thức `renderable` không trả về giá trị, việc trình bày exception theo cách của Laravel sẽ được áp dụng:

```
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
/**
 * Register the exception handling callbacks for the application.
 *
 * @return void
 */
public function register()
{
  $this->renderable(function (NotFoundHttpException $e, $request) {
    if ($request->is('api/*')) {
      return response()->json([
```

```
            'message' => 'Record not found.'
        ], 404);
    }
  });
}
```

Các exception có thể ghi và có thể hiển thị

Thay vì kiểm tra kiểu exception trong phương thức `register` của chương trình xử lý exception, bạn có thể khai báo các phương thức `report` và `render` trực tiếp lên các exception tùy chỉnh của mình. Khi các phương thức này tồn tại, chúng sẽ được tự động gọi bởi framework:

```
<?php
namespace App\Exceptions;
use Exception;
class InvalidOrderException extends Exception
{
  /**
   * Report the exception.
   *
   * @return bool|null
   */
  public function report()
  {
    //
  }
  /**
   * Render the exception into an HTTP response.
   *
   * @param  \Illuminate\Http\Request  $request
   * @return \Illuminate\Http\Response
   */
  public function render($request)
  {
    return response(...);
  }
```

```
}
```

Nếu exception của bạn mở rộng một exception đã được tuỳ chỉnh trước đó, chẳng hạn như exception được tích hợp sẵn trong Laravel hoặc Symfony, thì bạn có thể trả về `false` từ phương thức `render` của exception để trình bày HTTP response mặc định của exception:

```
/**
 * Render the exception into an HTTP response.
 *
 * @param  \Illuminate\Http\Request  $request
 * @return \Illuminate\Http\Response
 */
public function render($request)
{
  // Determine if the exception needs custom rendering...
  return false;
}
```

Nếu exception của bạn chứa logic báo cáo tùy chỉnh mà chỉ áp dụng khi phải đáp ứng các điều kiện nào đó, thì bạn có thể cần phải cho Laravel biết khi nào báo cáo exception bằng cách sử dụng cấu hình xử lý exception mặc định. Để thực hiện điều này, bạn có thể trả về `false` từ phương thức `report` của exception:

```
/**
 * Report the exception.
 *
 * @return bool|null
 */
public function report()
{
  // Determine if the exception needs custom reporting...

  return false;
}
```

Bạn có thể khai báo bất kỳ thư viện nào trên phương thức `report` và chúng sẽ tự

động được đưa vào trong phương thức bởi service container của Laravel.

Các exception HTTP

Một số exception mô tả mã lỗi HTTP từ máy chủ. Ví dụ: đây có thể là lỗi "không tìm thấy trang" (404), "lỗi trái phép" (401) hoặc thậm chí lỗi ứng dụng 500 do nhà phát triển tạo ra. Để tạo các response này ở bất kỳ đâu trong ứng dụng của bạn, bạn có thể sử dụng hàm `abort`:

```
abort(404);
```

Tuỳ biến trang hiện lỗi

Laravel giúp dễ dàng hiển thị các trang lỗi tùy biến cho các mã HTTP status khác nhau. Ví dụ: nếu bạn muốn tùy biến trang lỗi cho mã HTTP 404, hãy tạo một template view *resources/views/errors/404.blade.php*. Bản view này sẽ được hiển thị tất cả các lỗi 404 do ứng dụng của bạn gặp phải. Các bản view trong thư mục này phải được đặt tên sao cho khớp với mã HTTP status mà chúng muốn thể hiện. Đối tượng `Symfony\Component\HttpKernel\Exception\HttpException` do hàm `abort` đưa lên sẽ được truyền vào bản view với biến `$exception`:

```
<h2>{{ $exception->getMessage() }}</h2>
```

Bạn có thể tạo nhanh các template trang lỗi mặc định của Laravel trong dự án của mình bằng cách dùng lệnh Artisan `vendor:publish`. Khi các template đã được đưa vào dự án, bạn có thể tùy chỉnh chúng theo ý thích của mình:

```
php artisan vendor:publish --tag=laravel-errors
```