Course - Laravel Framework

Console Test

Ngoài việc đơn giản hóa kiểm tra HTTP, Laravel cung cấp một API đơn giản để kiểm tra các lệnh console tùy biến của ứng dụng của bạn.

Tags: console test, kiem tra lenh console, laravel

Giới thiệu

Ngoài việc đơn giản hóa kiểm tra HTTP, Laravel cung cấp một API đơn giản để kiểm tra các lệnh console tùy biến của ứng dụng của bạn.

Theo định hướng thành công / thất bại

Để bắt đầu, hãy khám phá cách xác nhận assertion liên quan đến mã thoát (exit code) của lệnh Artisan. Để thực hiện điều này, chúng ta sẽ sử dụng phương thức `artisan` để gọi một lệnh Artisan từ bài kiểm tra của chúng ta. Sau đó, chúng ta sẽ sử dụng phương thức `assertExitCode` để xác nhận rằng lệnh đã hoàn thành với một mã thoát nhất định:

```
/**
 * Test a console command.
 *
 * @return void
 */
public function test_console_command()
{
  $this->artisan('inspire')->assertExitCode(0);
}
```

Bạn có thể sử dụng phương thức `assertNotExitCode` để khẳng định rằng lệnh không thoát với một mã thoát (exit code) nhất định:

```
$this->artisan('inspire')->assertNotExitCode(1);
```

Tất nhiên, tất cả các lệnh đầu cuối thường thoát với mã trạng thái là `0` khi chúng thành công và mã thoát (exit code) khác `0` khi chúng không thành công. Do đó, để thuận tiện, bạn có thể sử dụng `assertSuccessful` và `assertFailed` để khẳng định rằng một lệnh nhất định đã thoát bằng mã thoát (exit code) thành công hay không:

```
$this->artisan('inspire')->assertSuccessful();

$this->artisan('inspire')->assertFailed();
```

Theo định hướng input / output

Laravel cho phép bạn dễ dàng "mô phỏng" đầu vào của người dùng cho các lệnh trên bảng điều khiển của bạn bằng phương thức mong đợiQuestion. Ngoài ra, bạn có thể chỉ định mã thoát và văn bản mà bạn muốn xuất ra bằng lệnh console bằng cách sử dụng các phương thức confirmExitCode và inheritOutput. Ví dụ: hãy xem xét lệnh console sau:

```
Artisan::command('question', function () {
  $name = $this->ask('What is your name?');

  $language = $this->choice('Which language do you prefer?', [
    'PHP',
    'Ruby',
    'Python',
  ]);

  $this->line('Your name is '.$name.' and you prefer '.$language.'.');
});
```

Bạn có thể kiểm tra lệnh này bằng bài kiểm tra sau, nó sử dụng các phương thức expectsQuestion, expectsOutput, doesntExpectOutput và assertExitCode:

```
/**
 * Test a console command.
 *
 * @return void
 */
public function test_console_command()
{
  $this->artisan('question')
    ->expectsQuestion('What is your name?', 'Taylor Otwell')
    ->expectsQuestion('Which language do you prefer?', 'PHP')
    ->expectsOutput('Your name is Taylor Otwell and you prefer PHP.')
    ->doesntExpectOutput('Your name is Taylor Otwell and you prefer Ruby.')
    ->assertExitCode(0);
}
```

Xác nhận định hướng

Khi viết một lệnh yêu cầu xác nhận dưới dạng câu trả lời "có" hoặc "không", bạn có thể sử dụng phương thức `expectsConfirmation`:

```
$this->artisan('module:import')
  ->expectsConfirmation('Do you really wish to run this command?', 'no')
  ->assertExitCode(1);
```

Bảng định hướng

Nếu lệnh của bạn hiển thị một bảng thông tin bằng cách sử dụng phương thức `table` của Artisan, thì việc viết các kỳ vọng đầu ra cho toàn bộ bảng có thể rất phức tạp. Thay vào đó, bạn có thể sử dụng phương thức `expectsTable`. Phương thức này chấp nhận tiêu đề của bảng làm đối số đầu tiên và dữ liệu của bảng làm đối số thứ hai:

```
$this->artisan('users:all')
  ->expectsTable([
    'ID',
    'Email',
  ], [
    [1, 'taylor@example.com'],
    [2, 'abigail@example.com'],
  ]);
```