Course - Laravel Framework

Browser Test

Laravel Dusk cung cấp một API kiểm tra và tự động hóa trên trình duyệt nhanh chóng, dễ sử dụng. Theo mặc định, Dusk không yêu cầu bạn cài đặt JDK hoặc Selenium trên máy tính cục bộ của bạn. Thay vào đó, Dusk sử dụng ChromeDriver. Tuy nhiên, bạn có thể tự do sử dụng bất kỳ driver tương thích Selenium nào khác mà bạn muốn.

Tags: browser test, laravel

Giới thiệu

Laravel Dusk cung cấp một API kiểm tra và tự động hóa trên trình duyệt nhanh chóng, dễ sử dụng. Theo mặc định, Dusk không yêu cầu bạn cài đặt JDK hoặc Selenium trên máy tính cục bộ của bạn. Thay vào đó, Dusk sử dụng ChromeDriver. Tuy nhiên, bạn có thể tự do sử dụng bất kỳ driver tương thích Selenium nào khác mà bạn muốn.

Bạn có thể tìm hiểu sâu về Dusk tại đây

Bắt đầu

Cài đặt

Để bắt đầu bạn phải cài đặt thư viện `laravel/dusk`.

```
composer require --dev laravel/dusk
```

Sau khi cài đặt package Dusk, hãy thực hiện lệnh Artisan `dusk:install`. Lệnh `dusk:install` sẽ tạo một thư mục *tests/Browser* và một ví dụ về Dusk test:

```
php artisan dusk:install
```

Tạo bài kiểm tra

Để tạo bài kiểm tra Dusk, hãy sử dụng lệnh Artisan `dusk:make`. Bài kiểm tra đã tạo sẽ được đặt trong thư mục *tests/Browser*:

```
php artisan dusk:make LoginTest
```

Tích hợp cơ sở dữ liệu

Hầu hết các bài kiểm tra bạn viết sẽ tương tác với các trang lấy dữ liệu từ cơ sở dữ liệu của ứng dụng của bạn; tuy nhiên, các bài kiểm tra Dusk của bạn không bao giờ được sử dụng trait `RefreshDatabase`. Trait `RefreshDatabase` tận dụng các giao dịch cơ sở dữ liệu sẽ không thể áp dụng hoặc không khả dụng trên các yêu cầu HTTP request. Thay vào đó, hãy sử dụng trait `DatabaseMigrations`, đặc điểm này sẽ tích hợp lại cơ sở dữ liệu cho mỗi lần kiểm tra:

```php
<?php

namespace Tests\Browser;
use App\Models\User;
use Illuminate\Foundation\Testing\DatabaseMigrations;
use Laravel\Dusk\Chrome;
use Tests\DuskTestCase;

class ExampleTest extends DuskTestCase
{
  use DatabaseMigrations;
}
```

> **Chú ý:** Cơ sở dữ liệu trong bộ nhớ SQLite có thể không được sử dụng khi thực hiện các bài kiểm tra Dusk. Vì trình duyệt thực thi trong quy trình của chính nó, nó sẽ không thể truy cập cơ sở dữ liệu trong bộ nhớ của các quy trình khác.

Chạy bài kiểm tra

Để chạy thử nghiệm trình duyệt của bạn, hãy thực hiện lệnh Artisan `dusk`:

```
php artisan dusk
```

Nếu bạn gặp lỗi kiểm tra vào lần cuối cùng bạn chạy lệnh `dusk`, bạn có thể tiết kiệm thời gian bằng cách chạy lại bài kiểm tra thất bại trước bằng lệnh `dusk:fails`:

```
php artisan dusk:fails
```

Lệnh `dusk` chấp nhận bất kỳ đối số nào thường được chương trình chạy bài kiểm tra PHPUnit chấp nhận, chẳng hạn như cho phép bạn chỉ chạy bài kiểm tra cho một nhóm nào đó:

```
php artisan dusk --group=foo
```

> Nếu bạn đang sử dụng Laravel Sail để quản lý môi trường phát triển cục bộ của mình, vui lòng tham khảo tài liệu Sail về cách cấu hình và chạy thử nghiệm Dusk.

Khởi động thủ công ChromeDriver

Theo mặc định, Dusk sẽ tự động cố gắng khởi động ChromeDriver. Nếu điều này không hiệu quả với hệ thống cụ thể của bạn, bạn có thể khởi động ChromeDriver theo cách thủ công trước khi chạy lệnh `dusk`. Nếu bạn chọn khởi động ChromeDriver theo cách thủ công, bạn nên comment dòng sau của tập tin *tests/DuskTestCase.php* của bạn:

```
/**
 * Prepare for Dusk test execution.
 *
 * @beforeClass
 * @return void
 */
public static function prepare()
{
  // static::startChromeDriver();
}
```

Ngoài ra, nếu bạn khởi động ChromeDriver trên một cổng khác 9515, bạn nên sửa đổi phương thức `driver` của cùng một class để phản ánh đúng cổng:

```
/**
 * Create the RemoteWebDriver instance.
 *
 * @return \Facebook\WebDriver\Remote\RemoteWebDriver
 */
protected function driver()
{
  return RemoteWebDriver::create(
    'http://localhost:9515', DesiredCapabilities::chrome()
  );
}
```