Utilisation de Behat dans Laravel

Behat is a PHP framework for BDD (Behavior-Driven Development). It allows you to write human-readable descriptions of your application's behavior and then automate the testing of those descriptions. Integrating Behat with Laravel can help ensure your application meets its specifications from the perspective of its stakeholders.

```
    Then I should see the "div#block-sl
    Then I should see the "div#block-m
    Then I should see the "div#breadcr

Feature: Home Page

  Scenario: View Homepage Features
    Given I am on the homepage
    Then I should see the "div#block-b
    Then I should see the "div#block-w
    Then I should see the "section#blo

19 scenarios (19 passed)
86 steps (86 passed)
0m19.81s (11.43Mb)
```

Setting Up Behat with Laravel

```
DEMO_PROJECT
  bin
    behat
  features/bootstrap
    FeatureContext.php
  vendor
  composer.json
  composer.lock
```

1. **Install Behat**
2. **Configure Behat**
3. **Write Feature Files**
4. **Write Context Classes**
5. **Run Behat Tests**

1. Install Behat

```
"require": {
    "php": ">=5.5.9",
    "laravel/framework": "5.2.*",
    "guzzlehttp/guzzle": "^6.1",
    "laralib/l5scaffold": "dev-master",
    "behat/mink": "^1.7",
    "behat/behat": "^3.1",
    "behat/mink-extension": "dev-master",
    "laracasts/behat-laravel-extension": "^1.0",
    "behat/mink-selenium2-driver": "^1.3",
    "league/flysystem-aws-s3-v3": "~1.0",
    "laracasts/utilities": "~2.0"
```

First, install Behat and its dependencies using Composer:

```
composer require --dev behat/behat
```

Then, initialize Behat:

```
vendor/bin/behat --init
```

This command will create the initial directory structure for Behat in your project.

2. Configure Behat

```
dow  Help
default:
    suites:
        home_ui:
            paths: [ %paths.base%/features/home]
            contexts: [ HomePageUiContext ]
    extensions:
        Laracasts\Behat:
            env_path: .env.behat
        Behat\MinkExtension:
            default_session: laravel
            base_url: https://recipes.dev
            laravel: ~
            selenium2:
              wd_host: "http://selenium.dev:4444/wd/hub"
            browser_name: chrome

travis:
  extensions:
    Laracasts\Behat:
      env_path: .env.travis
    Behat\MinkExtension:
      base_url: http://localhost:8000
      default_session: laravel
      laravel: ~
      selenium2:
        wd_host: 'http://127.0.0.1:4444/wd/hub'
      browser_name: chrome
```

Behat uses a configuration file (`behat.yml`) to define settings for your tests. Create this file in the root of your Laravel project:

```
default:
    suites:
```

```
        default:
            contexts:
                - FeatureContext
    extensions:
        Behat\MinkExtension:
            base_url: http://localhost
            goutte: ~
            selenium2: ~
```

This configuration sets up Behat with the Mink extension, which allows browser interaction. Adjust the `base_url` to match your local development environment.

3. Write Feature Files

Feature files describe the behavior of your application in Gherkin syntax. Create a directory for your feature files (`features`) and add a new feature file (`login.feature`):

```
Feature: User Login
  In order to use the application
  As a registered user
  I want to log in to my account

  Scenario: Successful login
    Given I am a registered user with email "john@example.com" and password "passw
    When I go to the login page
    And I fill in "email" with "john@example.com"
    And I fill in "password" with "password"
    And I press "Login"
```

```
    Then I should be redirected to the dashboard
    And I should see "Welcome, John Doe"
```

4. Write Context Classes

Context classes contain the PHP code that implements the steps in your feature files. Behat generated a `FeatureContext.php` file when you ran the `--init` command. Open this file and add the necessary step definitions:

```php
<?php

use Behat\Behat\Context\ClosuredContextInterface,
    Behat\Behat\Context\TranslatedContextInterface,
    Behat\Behat\Context\BehatContext,
    Behat\Behat\Exception\PendingException;
use Behat\Gherkin\Node\PyStringNode,
    Behat\Gherkin\Node\TableNode;

use Behat\MinkExtension\Context\MinkContext;

/**
 * Features context.
 */
class FeatureContext extends MinkContext
{
}
```

```php
<?php

use Behat\Behat\Context\Context;
use Behat\Behat\Tester\Exception\PendingException;
use Behat\MinkExtension\Context\MinkContext;
use App\Models\User;
use Illuminate\Support\Facades\Hash;

class FeatureContext extends MinkContext implements Context
{
    /**
     * @Given I am a registered user with email :email and password :password
     */
    public function iAmARegisteredUserWithEmailAndPassword($email, $password)
    {
        User::create([
            'name' => 'John Doe',
            'email' => $email,
            'password' => Hash::make($password),
```

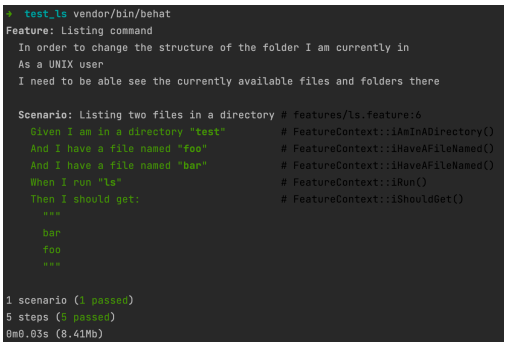
```php
        ]);
    }

    /**
     * @Then I should be redirected to the dashboard
     */
    public function iShouldBeRedirectedToTheDashboard()
    {
        $this->assertPageAddress('/dashboard');
    }

    /**
     * @Then I should see :text
     */
    public function iShouldSee($text)
    {
        $this->assertPageContainsText($text);
    }
}
```

5. Run Behat Tests

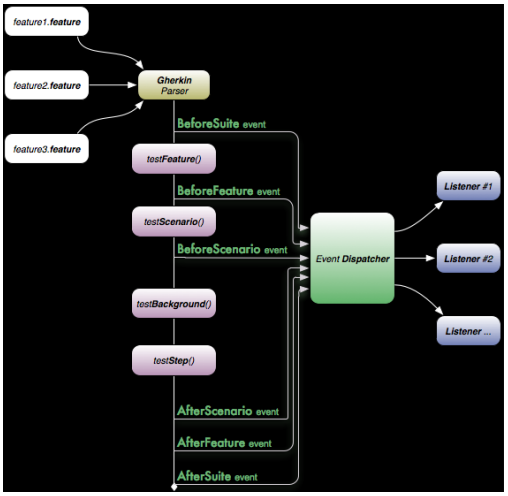
Before running your tests, ensure your Laravel application is running. Then, execute Behat:




```
vendor/bin/behat
```

Example Complete Workflow

1. **Write a Feature File**: Describe the behavior in Gherkin syntax.
2. **Implement Step Definitions**: Add methods to the context class to match each step.
3. **Run Behat**: Execute Behat to run your tests.
4. **Develop and Refactor**: Implement the necessary functionality in Laravel, refactor your code, and ensure all tests pass.



Tips for Integration

- **Database Transactions**: Use Laravel's `DatabaseTransactions` trait in your context class to ensure each scenario runs in isolation.
- **Browser Testing**: For more complex browser interactions, consider using Selenium with Mink.
- **Mocking and Stubbing**: Use Laravel's testing helpers to mock and stub external services.

Example `behat.yml` with Laravel Specific Configurations

```yaml
default:
    extensions:
        Behat\MinkExtension:
            base_url: http://localhost:8000
            default_session: laravel
            laravel: ~
            selenium2: ~
        Laracasts\Behat\ServiceContainer\BehatExtension: ~
    suites:
        default:
            paths: [ %paths.base%/features ]
            contexts:
                - FeatureContext
```

Conclusion

By following these steps and examples, you can effectively integrate Behat into your Laravel project, enabling you to practice BDD and ensure your application behaves as expected from the user's perspective.