EBook Gratis

APRENDIZAJE behat

Free unaffiliated eBook created from
Stack Overflow contributors.

#behat

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: behat

It is an unofficial and free behat ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official behat.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con behat

Observaciones

Esta sección proporciona una descripción general de qué es behat y por qué un desarrollador podría querer usarlo.

También debe mencionar cualquier tema grande dentro de behat, y vincular a los temas relacionados. Dado que la Documentación para behat es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Examples

Pruebas funcionales como historias de usuario.

Las pruebas funcionales se describen mejor como pruebas para sus historias de usuario. Si has tratado con historias de usuario antes, normalmente siguen el siguiente patrón:

```
As a [role], I want to [desire], so that [benefit/desired outcome]
```

Para los siguientes ejemplos usaremos esta historia de usuario como ejemplo:

```
As a Dungeon Master, I want to ensure harmony and mutual trust, so that
the party can work together as a team
```

Los dos marcos de prueba más populares para pruebas funcionales en PHP son Behat y PHPSpec .

Comenzando con Behat

Behat proporciona sintaxis de Gherkin, que es un formato legible por humanos. Te permite describir fácilmente tus historias de usuario.

Para comenzar con Behat, debe instalarlo con Composer y luego inicializar sus archivos de prueba:

```
$ composer require --dev behat/behat="^3.0.5"

$ ./vendor/bin/behat --init

+d features # place your *.feature files here
+d features/bootstrap # place your context classes here
+f features/bootstrap/FeatureContext.php # place your definitions, transformations and hooks
here
```

Por defecto, coloca sus archivos de prueba en la carpeta de `features/` y tiene la extensión `.feature` .

Cada archivo de prueba debe definir una característica particular de la aplicación. Una característica se divide en un montón de escenarios y contiene una serie de pasos que deben realizarse con éxito para que el escenario pase. Cada escenario debe pasar para que una característica pase.

```
# features/PartyHarmony.feature
Feature: Party Harmony
    As a Dungeon Master, I want to ensure harmony and mutual trust, so that
    the party can work together as a team

    Scenario: Teach members to respect each others property
        Given that the Wizard has 10 cookies
        And the Bard eats 1 cookie
        Then the Bard is mysteriously on fire
```

Para ejecutar tus pruebas, ejecutas el binario de Behat directamente. Opcionalmente, podemos especificar qué archivo de características se ejecutará (de lo contrario, se ejecutan todas las pruebas). Este archivo de características fallará con errores de pasos no definidos (porque no hemos definido qué significan esos pasos):

```
$ ./vendor/bin/behat features/PartyHarmony.feature
Feature: Party Harmony
    As a Dungeon Master, I want to ensure harmony and mutual trust, so that
    the party can work together as a team

  Scenario: Teach members to respect each others property # features/PartyHarmony.feature:6
    Given that the Wizard has 10 cookies
    And the Bard eats 1 cookie
    Then the Bard is mysteriously on fire

1 scenario (1 undefined)
3 steps (3 undefined)
0m0.01s (10.49Mb)

--- FeatureContext has missing steps. Define them with these snippets:

    /**
     * @Given that the Wizard has :arg1 cookies
     */
    public function thatTheWizardHasCookies($arg1)
    {
        throw new PendingException();
    }

    /**
     * @Given the Bard eats :arg1 cookie
     */
    public function theBardEatsCookie($arg1)
    {
        throw new PendingException();
    }

    /**
     * @Then the Bard is mysteriously on fire
     */
    public function theBardIsMysteriouslyOnFire()
    {
```

```
        throw new PendingException();
    }
```

Cada paso en un escenario ejecuta un fragmento de código de un archivo PHP de contexto (diferentes pruebas de características pueden cargar diferentes contextos). Podemos copiar los ejemplos sugeridos por Behat o crear los nuestros. Los pasos se comparan con una comprobación de expresión regular. Así que si implementamos

```
<?php
#
class FeatureContext {
    /**
     * @Given that the wizard has :num cookies
     */
    public function wizardHasCookies($num) {
        // $this->wizard is a pre-existing condition.... like syphilis
        $this->wizard->setNumberOfCookies($num);
    }

    /**
     * @Given the Bard eats :num cookie
     */
    public function theBardEatsCookie($num)
    {
        $this->bard->consumeCookies($num);
    }

    /**
     * @Then the Bard is mysteriously on fire
     */
    public function theBardIsMysteriouslyOnFire() {
        PHPUnit_Framework_Assert::assertTrue(
            $this->bard->isBardOnFire()
        );
    }
}
```

Notarás el uso de `PHPUnit_Framework_Assert` . Behat no tiene su propio sistema de afirmación, por lo que puede utilizar el que desee.

Ahora, ejecutar las pruebas ejecutará el código real y podemos probar si todo pasa:

```
$ ./vendor/bin/behat features/PartyHarmony.feature
Feature: Party Harmony
    As a Dungeon Master, I want to ensure harmony and mutual trust, so that
    the party can work together as a team

  Scenario: Teach members to respect each others property # features/PartyHarmony.feature:6
    Given that the Wizard has 10 cookies                  #
FeatureContext::thatTheWizardHasCookies()
    And the Bard eats 1 cookie                            #
FeatureContext::theBardEatsCookie()
    Then the Bard is mysteriously on fire                 #
FeatureContext::theBardIsMysteriouslyOnFire()

1 scenario (1 passed)
3 steps (3 passed)
```

```
0m0.01s (10.59Mb)
```

Extendiendo Behat con Visón

Mink proporciona una interfaz para los controladores web (como Goutte y Selenium), así como un `MinkContext` que, cuando se amplía, proporciona un lenguaje web adicional para nuestros pasos.

Para instalar Mink (y el controlador Goutte predeterminado):

```
$ composer require --dev behat/mink-extension="^2.0"
$ composer require --dev behat/mink-goutte-driver="^1.0"
```

Luego extiende tu contexto con `MinkContext` :

```
<?php
use Behat\MinkExtension\Context\MinkContext;

class FeatureContext extends MinkContext … {
    …
}
```

Puede ver la lista completa de sintaxis disponible en su instalación de Behat con el siguiente comando:

```
$ ./vendor/bin/behat -dl

Given /^(?:|I )am on "(?P<page>[^"]+)"$/
 When /^(?:|I )reload the page$/
 When /^(?:|I )move backward one page$/
 When /^(?:|I )move forward one page$/
 When /^(?:|I )press "(?P<button>(?:[^"]|\\")*)"$/
 When /^(?:|I )follow "(?P<link>(?:[^"]|\\")*)"$/
 When /^(?:|I )fill in "(?P<field>(?:[^"]|\\")*)" with "(?P<value>(?:[^"]|\\")*)"$/
```

A continuación, debe configurar Mink para indicar dónde se encuentra el sitio web que desea probar y qué controladores web utilizar (Goutte de forma predeterminada):

```
# ./behat.yml
default:
    extensions:
        Behat\MinkExtension:
            base_url: "[your website URL]"
            sessions:
                default:
                    goutte: ~
```

Aquí hay un ejemplo de un escenario que usa solo los pasos provistos por Mink:

```
# ./features/Authentication.feature
Feature: Authentication
    As a security conscious developer I wish to ensure that only valid users can access our
website.
```

```
    Scenario: Login in successfully to my website
        When I am on "/login"
        And I fill in "email" with "my@email.com"
        And I fill in "password" with "my_password"
        And I press "Login"
        Then I should see "Successfully logged in"

    Scenario: Attempt to login with invalid credentials
        When I am on "/login"
        And I fill in "email" with "my@email.com"
        And I fill in "password" with "not_my_password"
        And I press "Login"
        Then I should see "Login failed"
```

Ahora puede probar esto ejecutando la función a través de Behat:

```
./vendor/bin/behat features/Authentication.feature
```

Puede crear sus propios pasos utilizando `MinkContext` para pasos comunes (por ejemplo, el inicio de sesión es una operación muy común):

```
Feature: Authentication
    As a security conscious developer I wish to ensure that only valid users can access our
website.

    Scenario: Login in successfully to my website
        Given I login as "my@email.com" with password "my_password"
        Then I should see "Successfully logged in"

    Scenario: Attempt to login with invalid credentials
        Given I login as "my@email.com" with password "not_my_password"
        Then I should see "Login failed"
```

Necesitará extender su archivo de contexto con `MinkContext` para obtener acceso a los controladores web y las interacciones de la página:

```
<?php
use Behat\MinkExtension\Context\MinkContext;

class FeatureContext extends MinkContext {
    /**
      * @Given I login as :username with password :password
      */
    public function iLoginAsWithPassword($username, $password) {
        $this->visit("/login");
        $this->fillField("email", $username);
        $this->fillField("password", $password);
        $this->pressButton("Login");
    }
}
```

Mink también proporciona selectores de CSS en la mayoría de sus llamadas proporcionadas previamente, lo que le permite identificar elementos en la página mediante construcciones como esta:

```
When I click on "div[id^='some-name']"
And I click on ".some-class:first"
And I click on "//html/body/table/thead/tr/th[first()]"
```

Probando JavaScript con Mink y Selenium

Si queremos probar JavaScript en un sitio web, tendremos que usar algo un poco más potente que Goutte (que es simplemente cURL a través de Guzzle). Hay un par de opciones como ZombieJS , Selenium y Sahi . Para este ejemplo usaré selenio.

Primero necesitarás instalar los controladores para Mink:

```
$ composer require --dev behat/mink-selenium2-driver="^1.2"
```

Y también deberá descargar el archivo jar de Selenium standalone server e iniciarlo:

```
$ java -jar selenium-server-standalone-2.*.jar
```

También tendremos que decirle a Behat que cuando usamos la etiqueta `@javascript` para usar el controlador Selenium y proporcionar la ubicación del servidor independiente Selenium.

```
# ./behat.yml
default:
    # …
    extensions:
        Behat\MinkExtension:
        base_url: "[your website URL]"
        sessions:
            # …
            javascript:
                selenium2:
                    browser: "firefox"
                    wd_host: http://localhost:4444/wd/hub
```

Luego, para cada prueba que desee ejecutar utilizando la emulación del navegador, solo necesita agregar una `@javascript` (o `@selenium2`) al comienzo de la función o escenario.

```
# ./features/TestSomeJavascriptThing.feature
@javascript # or we could use @selenium2
Feature: This test will be run with browser emulation
```

La prueba se puede ejecutar a través de Behat (como cualquier otra prueba). La única diferencia es que, cuando se ejecuta la prueba, debe aparecer una ventana del navegador en la computadora que ejecuta el servidor independiente Selenium que luego realizará las pruebas descritas.

Configuración de datos de prueba

Con las pruebas funcionales se suelen modificar los datos. Esto puede hacer que las ejecuciones subsiguientes del conjunto de pruebas falle (ya que los datos podrían haber cambiado desde el

estado original en el que se encontraba).

Si ha configurado su origen de datos utilizando un ORM o un marco que admita la migración o la inicialización (como Doctrine , Propel , Laravel), puede usar esto para crear una nueva base de datos de prueba completa con datos de Fixture en cada ejecución de prueba.

Si actualmente no usa uno de estos (o equivalente), puede usar herramientas como Phinx para configurar rápidamente una nueva base de datos de prueba o preparar una base de datos existente para cada ejecución de prueba (limpiar las entradas de prueba, restablecer los datos a su estado original)).

```
# Install Phinx in your project
$ php composer.phar require robmorgan/phinx

$ php vendor/bin/phinx init
Phinx by Rob Morgan - https://phinx.org. version x.x.x
Created ./phinx.xml
```

Agregue sus credenciales de base de datos a `./phinx.xml` .

```
$ php vendor/bin/phinx create InitialMigration
```

Puede especificar cómo se crean y rellenan las tablas de la base de datos utilizando la sintaxis que se proporciona en la documentación .

Luego, cada vez que ejecutas tus pruebas ejecutas un script como este:

```
#!/usr/bin/env bash

# Define the test database you'll use
DATABASE="test-database"

# Clean up and re-create this database and its contents
mysql -e "DROP DATABASE IF EXISTS $DATABASE"
mysql -e "CREATE DATABASE $DATABASE"
vendor/bin/phinx migrate

# Start your application using the test database (passed as an environment variable)
# You can access the value with $_ENV['database']
database=$DATABASE php -d variables_order=EGPCS -S localhost:8080

# Run your functional tests
vendor/bin/behat
```

Ahora sus pruebas funcionales no deben fallar debido a cambios en los datos.

Capturando correos electrónicos

Las pruebas funcionales también pueden incluir procesos de prueba que abandonan su entorno, como llamadas a API externas y correos electrónicos.

Como ejemplo, imagine que está probando funcionalmente el proceso de registro para su sitio

web. El último paso de este proceso consiste en enviar un correo electrónico con un enlace de activación. Hasta que se visite este enlace, la cuenta no está completamente registrada. Usted querría probar ambos:

1. Que el correo electrónico se envíe correctamente (formato, reemplazo de marcador de posición, etc.) y,
2. Que el enlace de activación funcione.

Ahora puede probar el envío de correo electrónico pero utilizando un cliente IMAP o POP para recuperar el correo electrónico enviado desde el buzón, pero esto significa que también está probando su conexión a Internet, el servidor de correo electrónico remoto y cualquier problema que pueda surgir durante la entrega (detección de spam) por ejemplo).

Una solución más simple es usar un servicio local que atrapa las conexiones SMTP salientes y vuelca el correo electrónico enviado al disco.

Un par de ejemplos son:

smtp-sink - Un programa de utilidad que viene incluido con Postfix.

```
# Stop the currently running service
sudo service postfix stop

# Dumps outgoing emails to file as "day.hour.minute.second"
smtp-sink -d "%d.%H.%M.%S" localhost:2500 1000

# Now send mails to your local SMTP server as normal and they will be
# dumped to raw text file for you to open and read

# Run your functional tests
vendor/bin/behat
```

No se olvide de eliminar `smtp-sink` y reinicie su servicio de postfix después:

```
# Restart postfix
sudo service postfix start
```

FakeSMTP : un cliente basado en Java que atrapa el correo saliente

```
# -b = Start without GUI interface
# -o = Which directory to dump your emails to
$ java -jar fakeSMTP.jar -b -o output_directory_name
```

Alternativamente, puede usar un servicio remoto que proporciona este servicio como mailtrap pero luego sus pruebas dependen del acceso a Internet.

Instalación o configuración

Behat / Mink

Instale usando composer (para otros métodos, mire) behat.org Si usa linux, asegúrese de haber

instalado php-curl (la instalación normal de rizos no funcionará)

Linux

```
sudo apt-get install php5-curl
```

Si está utilizando **Windows** , asegúrese de tener PHP, Curl y Git instalados. Puedes encontrarlos en los siguientes enlaces:

- PHP (Xampp): https://www.apachefriends.org/de/index.html
- Curl: http://curl.haxx.se/latest.cgi?curl=win64-nossl
- Git: http://git-scm.com/download/win

Tu compositor.json contendría lo siguiente:

behat - composer.json

```
{
  "require": {
    "behat/behat": "dev-master",
    "behat/mink": "dev-master",
    "behat/mink-extension": "dev-master",
    "behat/mink-selenium2-driver": "dev-master",
    "phpunit/php-code-coverage": "dev-master",
    "phpunit/phpunit-mock-objects": "dev-master",
    "phpunit/phpunit": "dev-master"
  },
  "minimum-stability": "dev",
  "config": {
    "bin-dir": "bin/"
  }
}
```

(Al guardar el archivo composer.json en Windows, debe elegir "Todos los archivos" como tipo de archivo y codificación "ANSI")

Luego ejecuta los siguientes comandos:

```
$ curl http://getcomposer.org/installer | php
$ php composer.phar install
```

Después de esta extensión de Behat, Mink y Behat-Mink se instalan, para ejecutar behat

ejecutar behat

```
$ bin/behat
```

Para activar el uso de la extensión Behat-Mink: behat.yml cree un archivo "behat.yml" con el siguiente contenido

behat.yml

```
default:
  suites:
    default:
      paths:
        features: %paths.base%/features/
        bootstrap: %paths.base%/features/bootstrap/
      contexts:
        - FeatureContext
  extensions:
    Behat\MinkExtension:
      base_url: 'http://www.startTestUrl.de'
      selenium2:
        browser: firefox
        wd_host: "http://localhost:4444/wd/hub"
```

Este archivo estará en el mismo directorio que contiene el directorio bin y el enlace a behat. También tenga en cuenta que en el archivo yml, no use pestañas para la sangría. utilizar espacios. Para obtener una lista de comandos disponibles en behat-mink, use

```
$ bin/behat -di
```

Haz que behat sea parte de tu sistema.

Linux

Vaya a su Homedirectory y haga lo siguiente:

```
$ sudo vi .bashrc
```

Y añade estas líneas al final del directorio.

```
export BEHAT_HOME=/home/*user*/path/to/behat
export PATH=$BEHAT_HOME/bin:$PATH
```

Reinicie la consola o escriba "source .bashrc"

Windows

Repase las configuraciones del sistema y agregue la ruta de acceso de behat / bin a las variables de entorno

También se deben instalar **otros controladores** sobre controladores como Selenium, phantomjs, goutte, etc.

Lea Empezando con behat en línea: https://riptutorial.com/es/behat/topic/9136/empezando-con-behat

Creditos

S. No	Capítulos	Contributors
1	Empezando con behat	Community, dasmelch, Tom