Serenity BDD y Alternativas en Python

Introducción a Serenity BDD

Serenity BDD es un framework de automatización de pruebas basado en Java que facilita la escritura de pruebas BDD (Behavior-Driven Development) utilizando herramientas como JUnit, Cucumber y Selenium. Su principal objetivo es proporcionar informes detallados y estructurados de las pruebas realizadas.

Características principales de Serenity BDD

- Soporte para Cucumber y JUnit.
- Integración con Selenium para pruebas web.
- Generación de informes automáticos detallados.
- Facilita la gestión de pruebas de aceptación y regresión.

Cómo utilizar Serenity BDD

Serenity BDD se puede configurar en un proyecto Java utilizando **Maven** y **JUnit** o **Cucumber**. A continuación, se muestra cómo crear un proyecto básico con Serenity BDD.

1. Crear un Proyecto con Serenity BDD

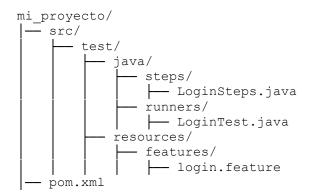
Instalación de dependencias

Para empezar, crea un proyecto **Maven** y agrega las siguientes dependencias en el archivo pom.xml:

```
<dependencies>
   <!-- Serenity core -->
   <dependency>
       <groupId>net.serenity-bdd</groupId>
       <artifactId>serenity-core</artifactId>
       <version>3.5.0
   </dependency>
   <!-- Serenity con Cucumber -->
   <dependency>
       <groupId>net.serenity-bdd</groupId>
       <artifactId>serenity-cucumber6</artifactId>
       <version>3.5.0
   </dependency>
   <!-- WebDriver para Selenium -->
   <dependency>
       <groupId>org.seleniumhq.selenium
       <artifactId>selenium-java</artifactId>
       <version>4.6.0
```

```
</dependency>
```

2. Estructura del Proyecto



3. Escribir un Escenario en Gherkin

```
En src/test/resources/features/login.feature:
```

```
Feature: Login en la aplicación web

Scenario: Usuario inicia sesión con éxito
Given el usuario abre el navegador en la página de login
When ingresa su usuario y contraseña
Then debería ver su panel de control
```

4. Definir los Pasos en Java

Archivo src/test/java/steps/LoginSteps.java:

```
import net.serenitybdd.core.steps.UIInteractionSteps;
import net.thucydides.core.annotations.Step;
import org.openqa.selenium.By;
import org.openga.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class LoginSteps extends UIInteractionSteps {
   WebDriver driver;
    @Step("Abrir el navegador en la página de login")
    public void abrirNavegador() {
        driver = new ChromeDriver();
        driver.get("https://example.com/login");
    }
    @Step("Ingresar usuario y contraseña")
    public void ingresarCredenciales() {
        driver.findElement(By.id("username")).sendKeys("usuario demo");
        driver.findElement(By.id("password")).sendKeys("password123");
        driver.findElement(By.id("login-button")).click();
```

```
@Step("Validar panel de control")
public void validarDashboard() {
    assert driver.getTitle().contains("Dashboard");
    driver.quit();
}
```

5. Ejecutar las Pruebas en Serenity BDD

Ejecuta las pruebas con:

```
mvn clean verify
```

Configurar Eclipse para BDD

Eclipse no tiene soporte nativo para archivos **.feature** de BDD, por lo que se recomienda instalar el plugin **Cucumber Eclipse Plugin**:

- 1. Ir a Help > Eclipse Marketplace.
- 2. Buscar Cucumber Eclipse Plugin e instalarlo.
- 3. Reiniciar Eclipse.
- 4. Ahora los archivos **.feature** serán reconocidos y se podrá ejecutar BDD correctamente.

Alternativas en Python: Behave y Pytest-BDD

Estructura del Proyecto para Alternativas en Python

Contenido de behave.ini

```
[behave]

default_tags = ~@wip  # Excluye escenarios marcados con @wip (Work In Progress)

logging_level = INFO  # Nivel de logging

show timings = false  # No mostrar tiempos de ejecución
```

Contenido de conftest.py

```
import pytest
from selenium import webdriver

@pytest.fixture(scope="session")
def browser():
    driver = webdriver.Chrome()
    yield driver
    driver.quit()
```

Ejemplo con Behave

```
Archivo features/login.feature:
Feature: Login en la aplicación web
  Scenario: Usuario inicia sesión con éxito
    Given el usuario abre el navegador en la página de login
    When ingresa su usuario y contraseña
    Then debería ver su panel de control
Archivo features/steps/login steps.py:
from behave import given, when, then
from selenium import webdriver
@given('el usuario abre el navegador en la página de login')
def step open browser(context):
    context.driver = webdriver.Chrome()
    context.driver.get("https://example.com/login")
@when('ingresa su usuario y contraseña')
def step login(context):
    context.driver.find element by id("username").send keys("usuario demo")
    context.driver.find_element_by_id("password").send_keys("password123")
    context.driver.find element by id("login-button").click()
@then('debería ver su panel de control')
def step verify dashboard(context):
    assert "Dashboard" in context.driver.title
    context.driver.quit()
```

Ejecutar pruebas en Behave

behave

Ejecutar pruebas en Pytest-BDD

pytest

Consideraciones Finales

Ambas opciones en Python permiten la automatización de pruebas web con Selenium y seguir la metodología BDD de manera efectiva. 🚜