Pruebas de aceptación

Uso de Cucumber con Maven y Java para las pruebas de aceptación

En este documento se describen los pasos para utilizar **Cucumber** en **IntelliJ IDEA** con **Maven**, que nos ayudará a desarrollar una aplicación siguiendo el enfoque **TDD (Desarrollo guiado por Pruebas)**. Cucumber permite escribir pruebas en lenguaje natural **Gherkin** y ejecutarlas en Java.

> El **Desarrollo guiado por pruebas de software**, o Test-driven development (TDD) es una práctica de ingeniería de software que involucra otras dos prácticas: Escribir las pruebas primero (Test First Development) y Refactorización (Refactoring). Para escribir las pruebas generalmente se utilizan las pruebas unitarias (unit test en inglés). En primer lugar, se escribe una prueba y se verifica que la nueva prueba falla. A continuación, se implementa el código que hace que la prueba pase satisfactoriamente y seguidamente se refactoriza el código escrito. El propósito del desarrollo guiado por pruebas es lograr un código limpio que funcione. La idea es que los requisitos sean traducidos a pruebas, de este modo, cuando las pruebas pasen se garantizará que el software cumple con los requisitos que se han establecido. El desarrollo guiado por pruebas es una metodología ágil, **centrada en el software y en el desarrollador**.

Con Gherkin, transformamos el desarrollo guiado por pruebas en desarrollo guiado por comportamiento, gracias al uso del lenguaje natural. Siguiendo este método, primero escribiremos las pruebas en **lenguaje natural (Gherkin)**, luego implementaremos la lógica en Java para hacer que las pruebas pasen.

> **Actividad**
>
> Registra los pasos de este tutorial en una memoria.

1. Crear el Proyecto en IntelliJ con Maven

1. Abre **IntelliJ IDEA** y selecciona **New Project**.
2. Elige **Maven** como gestor de dependencias.
3. En **GroupId**, escribe: `com.ejemplo`.
4. En **ArtifactId**, escribe: `calculadora-cucumber`.
5. Haz clic en **Finish**.

El proyecto tendrá esta estructura inicial:

```
calculadora-cucumber
├── src
│   ├── main
│   │   └── java
│   │       └── com.ejemplo
│   │           └── Calculadora.java
│   ├── test
│   │   ├── java
│   │   │   └── com.ejemplo
│   │   │       ├── RunCucumberTest.java
│   │   │       ├── StepDefinitions.java
│   │   └── resources
│   │       └── features
│   │           └── calculadora.feature
├── pom.xml
├── .gitignore
```

2. Agregar Dependencias de Cucumber

Edita el archivo `pom.xml` y agrega las siguientes dependencias para **Cucumber y JUnit**:

```
<dependencies>
    <!-- Cucumber Core -->
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>7.14.0</version>
    </dependency>

    <!-- Cucumber JUnit -->
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>7.14.0</version>
        <scope>test</scope>
    </dependency>

    <!-- JUnit 5 -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter</artifactId>
        <version>5.9.2</version>
        <scope>test</scope>
    </dependency>

    <!-- Plugin para ejecutar pruebas -->
    <dependency>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.0.0-M7</version>
    </dependency>
</dependencies>
```

Ejecuta en la terminal de IntelliJ:

```
mvn clean install
```

para descargar las dependencias.

3. Escribir Escenarios de Prueba en Gherkin

Creamos el archivo `src/test/resources/features/calculadora.feature` con los escenarios en **Gherkin**:

```
Feature: Calculadora
  Como usuario, quiero realizar operaciones matemáticas básicas
  para obtener resultados correctos.

  Scenario: Sumar dos números
    Given que tengo una calculadora
    When sumo 2 y 3
    Then el resultado debe ser 5

  Scenario: Restar dos números
    Given que tengo una calculadora
    When resto 5 y 2
    Then el resultado debe ser 3
```

- `Given` establece el estado inicial.
- `When` define la acción.
- `Then` verifica el resultado esperado.

4. Implementar la Clase `Calculadora.java`

Creamos la clase en `src/main/java/com/ejemplo/Calculadora.java` :

```
package com.ejemplo;

public class Calculadora {
    public int sumar(int a, int b) {
        return a + b;
    }

    public int restar(int a, int b) {
        return a - b;
    }
}
```

5. Implementar los Step Definitions

Creamos la clase `StepDefinitions.java` en `src/test/java/com/ejemplo/StepDefinitions.java` :

```java
package com.ejemplo;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.When;
import io.cucumber.java.en.Then;
import static org.junit.jupiter.api.Assertions.*;

public class StepDefinitions {
    private Calculadora calculadora;
    private int resultado;

    @Given("que tengo una calculadora")
    public void queTengoUnaCalculadora() {
        calculadora = new Calculadora();
    }

    @When("sumo {int} y {int}")
    public void sumo(int a, int b) {
        resultado = calculadora.sumar(a, b);
    }

    @When("resto {int} y {int}")
    public void resto(int a, int b) {
        resultado = calculadora.restar(a, b);
    }

    @Then("el resultado debe ser {int}")
    public void elResultadoDebeSer(int esperado) {
        assertEquals(esperado, resultado);
    }
}
```

6. Configurar el Runner de Pruebas

Creamos la clase `RunCucumberTest.java` en `src/test/java/com/ejemplo/RunCucumberTest.java` :

```java
package com.ejemplo;

import org.junit.platform.suite.api.*;

@Suite
@IncludeEngines("cucumber")
@SelectClasspathResource("features")
@ConfigurationParameter(key = "cucumber.glue", value = "com.ejemplo")
public class RunCucumberTest {
}
```

7. Ejecutar las Pruebas con Maven

Para ejecutar las pruebas, usa el siguiente comando en la terminal:

```
mvn test
```

Si todo está configurado correctamente, deberías ver un resultado similar a este:

```
[INFO] Running com.ejemplo.RunCucumberTest
Feature: Calculadora

  Scenario: Sumar dos números
    Given que tengo una calculadora
    When sumo 2 y 3
    Then el resultado debe ser 5

  Scenario: Restar dos números
    Given que tengo una calculadora
    When resto 5 y 2
    Then el resultado debe ser 3

[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
```

Esto indica que las pruebas han pasado correctamente.

> **Actividad**
>
> Realiza el siguiente tutorial oficial y haz una memoria
>
> - Tutorial de 10 minutos -se tarda más-

Principales anotaciones de Cucumber

Anotación	Descripción
`@Given`	Define el contexto inicial.
`@When`	Describe la acción que ocurre.
`@Then`	Verifica el resultado esperado.