Cucumber Testing Documentation

Setting Up Cucumber

Cucumber is introduced as a robust testing tool supporting Behavior-Driven Development (BDD) by enabling tests to be composed in natural language. The integration of Cucumber into a Spring Boot project is outlined in a series of steps to facilitate seamless collaboration between development and testing teams.

Step 1: Add Cucumber Dependencies

Dependencies for Maven and Gradle are specified to incorporate Cucumber into the project. The necessary dependencies include Spring Boot Starter Web, Spring Boot Starter Test, and Cucumber dependencies with specific version numbers.

Step 2: Create Cucumber Feature Files

Feature files are to be crafted in the src/test/resources directory, containing test scenarios expressed in Gherkin language. An example feature file, greeting.feature, is provided with two scenarios: one involving a custom name and another with a default name.

gherkin

Feature: Greeting API

Scenario: User greets with a custom name

Given user has a greeting API

When user sends the data as "<name>"

Then the response should be "Hello <name>!"

Scenario: User greets with default name

Given user has a greeting API

When user sends the data as "world"

Then the response should be "Hello world!"

Step 3: Implement Step Definitions

Step definition classes, residing in the src/test/java directory, are introduced to map Gherkin steps to Java code. The example GreetingStepDefinitions.java file illustrates the mapping for steps related to the greeting API scenarios.

Java package com.example.restservice;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class GreetingStepDefinitions {

@Given("user has a greeting API")

public void givenStep() {

// Implementation goes here

}

@When("user sends the data as {string}")

public void whenStep(String data) {

// Implementation goes here

}

@Then("the response should be {string}")

public void thenStep(String expectedResponse) {

// Implementation goes here

}

}

Step 4: Configure Cucumber Options

Configuration options are to be set up through a runner class or a cucumber.properties file, specifying features, glue (package containing step definitions), plugins, and more. An example cucumber.properties file is presented to illustrate the configuration.

The completion of these steps establishes the Spring Boot project with Cucumber for writing and executing tests. The guidance recommends running the test runner class or employing Maven/Gradle commands to execute Cucumber tests and reviewing the generated reports for detailed results.

Test Case 1: User greets with a custom name

- Given user has a greeting API:

- Establishes the initial conditions, indicating user access to the greeting API.

- When user sends the data as "<name>":

- Describes the user action, sending data to the API with a custom name represented by "<name>".

- Then the response should be "Hello <name>!":

- Specifies the expected outcome, asserting that the API response should be a greeting message containing the custom name provided.

Test Case 2: User greets with default name

- Given user has a greeting API:

- Similar to the first case, sets up the initial conditions, indicating user access to the greeting API.

- When user sends the data as "world":

- Describes the user action, sending data to the API with the default name "world."

- Then the response should be "Hello world!":

- Specifies the expected outcome, asserting that the API response should be a greeting message containing the default name "world."

Purpose and Significance:

- Variety of Scenarios:

- The test cases cover different scenarios, ensuring the API responds correctly to both custom and default names, validating its versatility.

- Behavior Verification:

- Leveraging Cucumber's Given-When-Then structure enhances communication between technical and non-technical stakeholders, providing a clear understanding of expected behavior.

- Readability and Clarity:

- Gherkin language promotes readability, allowing stakeholders to comprehend and validate expected behavior without delving into intricate technical details.

- Easy Maintenance:

- The separation of scenarios in Gherkin files and step definitions allows for easy maintenance and modification of test cases without impacting the underlying code.

In summary, these test cases showcase the flexibility and readability benefits of Cucumber, facilitating collaboration, verification, and maintenance of the greeting API's behavior across various input scenarios.