Cucumber-BDD Framework

Testleaf
Always Ahead

Summary

Behaviour-Driven Development (BDD) is the software development process that Cucumber was built to support.

BDD is a way for software teams to work that closes the gap between business people and technical people by encouraging collaboration across roles to build shared understanding of the problem to be solved

Pre-requisite

Pre requirement for Cucumber

- Cucumber Plugin
- Natural Plugin
- Dependencies

```
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-
testng</artifactId>
         <version>7.1.0</version>
    </dependency>
   <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>7.1.0</version>
   </dependency>
```

Abbreviations

- BDD-Behaviour DrivenDevelopment framework
- TDD- Test Driven Development
- ATDD-Acceptance Test Driven Development

Layers in Cucumber

Feature: Includes Gherkin Syntax to communicate the behavior of the application

Step Definition: Actual implementation of feature steps with java

Runner: For executing the code by mapping the feature file and Step definition

Gherkin Keywords

Feature: Represents the high-level description of a software feature

Scenario: Represents the all possible features/functionality of the application

Scenario Outline: run the same scenario multiple times, with different combinations of values.

Examples: Represents the dynamic test data(list of values) to be passed in to the application

Background: Holds the common steps of the feature file.

Given: Represents the Pre-Condition steps in the application

When: Represents the test condition of the application

Then: Represents the expected positive outcome of the test scenario

But: Represents the expected negative outcome of the test scenario

And: To replace successive Given's , When's and Then's - to make the feature step more readable

Testleaf
Always Ahead

Step Implementation

Includes actual Java Code implementation for the feature file.

Annotations to integrate/map with feature file

- @Given
- @When
- @Then
- @And
- @But

Hooks Implementation: To have the common lines of code in the project specific method/Base Class

- @Before -PreCondition steps
- @After - PostCondition steps

Runner Class

Uses abstract class AbstractTestNgCucumberTests for the execution which internally have @Test annotation

@CucumberOptions- to Configuring the Cucumber Execution in Runner class which is imported from

import io.cucumber.testng.CucumberOptions;

and the options it includes

features: To set the path of the feature file to be executed

glue: To map with the actual step implementation with feature (set the path for the step definition

monochrome: To remove the unwanted junk characters in the console. Default value= false

publish: To generate the cucumber report .Default value= false

snippets: To set the snippet method signature . snippets=SnippetType.*CAMELCASE*

dryRun : To compile the feature file with the step definition.default value= false

tags : To categorize the feature files. tags=@Login