SYNTAX
TECHNOLOGIES

Cucumber

Class 1

Agenda

What is Cucumber ? Benefits of Cucumber In Selenium?

Gherkin Language Usage in Cucumber

Feature File

Step Definitions

Runner Class

Cucumber Tags

Cucumber Introduction

Cucumber is a testing tool that supports Behavior Driven Development (BDD) framework.

In very simple terms, BDD or behavior-driven development, is a technique where your specifications or test cases are written in plain English like sentences.

With this approach, the non-technical team members find it easy to understand the flow and collaborate more in the process of software development.

It defines application behavior from end user point of few using simple English text , defined by a language called Gherkin.

It allows automation of functional validation in easily readable and understandable format (like plain English) to Business Analysts, Developers, Testers, etc.

Cucumber was initially implemented in Ruby and then extended to Java framework. Both the tools support native JUnit.

Benefits of BDD

Collaboration:

This process involves multiple stakeholders and their understanding of the project. Having all the stakeholders involved from the beginning and throughout the process fosters strong communication and engage in the product development cycle.

Focus on End Users:

BDD puts the customer's vision in mind first. BDD focuses on the user's view on how the application should behave. By focusing on the user's view, this takes out the "what-if's" from all the different stakeholders to deliver the product in a shorter software development life cycle.

Business Values:

Implementing BDD automates redundant processes and creates structured scenarios by writing test scenarios in an easily-decipherable language – Gherkin. Eliminating lag time in between processes means that businesses can focus on client priorities and deliver results faster.

Gherkin Language

The purpose of the Cucumber test framework is to execute examples expressed in Gherkin.

Gherkin is the high level language where you express examples in plain language using **Given/When/Then** to setup your environment, use the system, and verify the result.

The purpose with Gherkin is to be able to express examples that describe a behavior of a system in such a way that anyone with domain knowledge can understand what works and how it is supposed to work.

Gherkin is not necessarily used to write automated tests. It is primarily used to write structured tests which can later be used as project documentation.

Gherkin keywords:

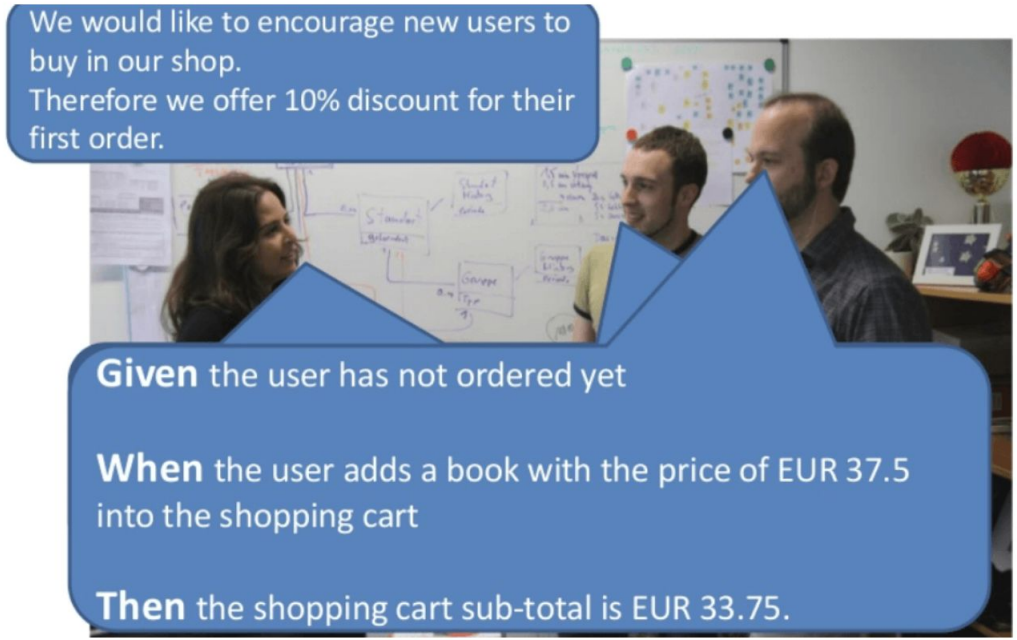
Feature, Background, Scenario, Given, When, Then, And

Gherkin Language

Before Gherkin

Gherkin Language

After Gherkin



What is the usage of each keyword

Feature: Each Gherkin file begins with a Feature keyword. Feature defines the logical test functionality you will test in this feature file, basically what we are going to test.

Background: Background keyword is used to define steps which are common to all the tests in the feature file.

Scenario: Each Feature will contain some number of tests to test the feature. Each test is called a Scenario and is described using the Scenario keyword.

Given : Given defines a precondition to the test.

When : When keyword defines the test action that will be executed. By test action we mean the user input action.

Then : Then keyword defines the Outcome of previous steps.

And : And keyword is used to add conditions to your steps

But : But keyword is used to add negative type comments. It is not a hard & fast rule to use but only for negative conditions

Benefits of Cucumber

There are number of benefits but from Selenium perspective, major advantages of Cucumber are :

- It gives the ability to produce Cucumber Reports of execution
- Cucumber BDD is open source and hence, its free to use
- With Cucumber, we can write our test scripts in multiple languages such as **Java, Ruby, .NET, Python etc**
- Cucumber easily integrates with Selenium and other web based testing tools
- Cucumber is one of the most widely used BDD tools. It is very easy to implement data driven testing using Cucumber

BDD can be implemented through free and open source frameworks like RSPEC, Cucumber, and Specflow.

Feature file

The file, in which Cucumber tests are written, is known as feature files.
A Feature File is an entry point to the Cucumber tests. This is a file where you will describe your tests in Gherkin format.
The extension of the feature file is ".feature".
A feature usually contains a list of scenarios.

Feature File consist of following components:

- Feature
- Scenario
- Scenario Outline
- Given
- When
- Then

```
1 Feature: Login
2
3 Scenario: Valid Login
4  Given I navigated to the HRMS application
5  When I enter valid username
6  And I enter valid password
7  And I click login button
8  Then I am successfully logged in
```

Step Definition File

Features files contains one or more scenarios per feature file.

Every scenario can contain n-number of test steps using the Gherkin keywords

For each and every step we have to provide the implementations this implementation is nothing but the methods in java but decorated with the test step detail.

Cucumber step definition class is a normal java class where we can store step definition methods. And a step definition method is a java method that is linked to a step in the scenario in feature file.

Step Definition File

```java
package com.steps;

import io.cucumber.java.en.*;

public class LoginSteps {

    @Given("I navigated to the HRMS application")
    public void i_navigated_to_the_HRMS_application() {
        System.out.println("I am on the hrms app");
    }

    @When("I enter valid username")
    public void i_enter_valid_username() {
        System.out.println("Username is entered");
    }

    @When("I enter valid password")
    public void i_enter_valid_password() {
        System.out.println("Password is entered");
    }

    @When("I click login button")
    public void i_click_login_button() {
        System.out.println("Clicked login button");
    }

    @Then("I am successfully logged in")
    public void i_am_successfully_logged_in() {
        System.out.println("Successfully logged in");
    }
}
```