**CUCUMBER WITH SELENIUM**

**BDD:**  Behavioral Driven Development is based on TDD(Test Driven Devp) and it fill the gap b/w B.A & dev team. And manual testers &automation testers.

BDD seems like plain text, but it has syntax based on different tools.

**GIVEN: (What we have to perform action)- prerequisites**

**WHEN: Performs action- Action**

**THEN: The desired outcome- Validation**

BDD Support Tools:

* Cucumber 🡪 ruby, python, java, js, php
* Jbehave
* Nbehave
* SpecFlow 🡪 BDD with C# people

But all above tools can use “ Gherkin”.

* It is a business readable like plain text
* Has some spaces and indentation to define structure.
* Has very few syntax which make parser(tool) to behave based on structure.

**Syntax of Gherkin:**

* **Feature**
* **Background**
* **Scenario**
* **Given**
* **When**
* **Then**
* **And**
* **But**
* **Scenario outline**
* **Scenario Templates**
* **Examples**

|  |  |
| --- | --- |
| BDD | Traditional Automation |
| Easy to understand by everyone like ba, productonr etc..bcz Plain text | Its in Coding, only d ev and qa understand |
| BDD can share with every one even stakeholders | Only who knows coding |
| Easy to imple ment | More Coding knowledge required |
|  |  |

**Installation pre-Requisite:**

* IDE
* Jar Files:
  + Cucumber-core
  + Cucumber-java
  + Cucumber-java-deps
  + Cucumber-jvm
  + Gherkin
  + Selenium(its related jars)

**How we are going to install or reference the JAR:**

Visual studio, C# 🡪 Nuget Repository.

In java 🡪 Maven Public Repository

**2 files required to execute a Cucumber test scenario?**

* Features
* Step Definition File

**Feature File:**

* This is plain text file
* It saves with .feature extension

**What is step definition?**

* Step definition files are java class files which hold all the steps written with feature file.
* In specflow each stepdefination file will be decorated with [Binding] attribute, in cucumber its not required.

**What Kind of reports can we generate in cucumber:**

* **HTML**
* **Jason**
* **Junit**

**Shortcut Keys:**

Ctrl + F1 == To get more undefined step refernces.

Alt + Enter == To create step definition

Ctrl + Shft + A == To increase font size

**Cucumber Options:**

* Dry Run
* Features
* Glue
* Tags 🡪 We Define in Feature file
* Mono Chrome
* Format
* Strict

**These all define in Test runner.**

Dry Run:

Is to check all steps from feature file is mentioned in stepdfntn filr or not.

Mapping b/w feature file and step definition file

Dryrun = true

Dryrun = false

Feature:

Location of feature file

features = "C:\\Users\\RITHWIK\\new\\Java\_MyWork\\Frmwrk\\src\\main\\java\\feature"

Glue:

Location of Step Definition File

Glue = {“Stepdefitn”}

Mono Chrome:

Display the console output more readable format

Monochrome = true

Format:

For Proper Format

format= {"pretty","html:test-outout"}

**To Generate different types of reporting format**

To create Jason , junit folders in project:

format= {"pretty","html:test-outout",”Jason:jason\_output/cucumbr.xml”,”junit:junit\_xml/cucmbr.xml”}

New Version:

plugin = { "html:target/cucumber-html-report",

"json:target/cucumber.json", "pretty:target/cucumber-pretty.txt",

"usage:target/cucumber-usage.json", "junit:target/cucumber-results.xml" },

Strict:

It will check any step is not defined in step definition file

It fails the execution, if any pending files are there

Strict = true,

**Data Driven Testing Using Cucumber**

**1) Simple Data Driven (Without Example Key Word)**

**2) With Example + Scenario Outline Keyword**

**3) Using Tables**

**1.Without Example: ( In Eclipse > Cucumber> srcmain > freecrm)**

Here we use “Scenario”

Regular Expression:

**\”(.\*)\”**

**\”([^”\]\*)\”**

**2. With Example Keyword**

Here we use “Scenario Outline”

<username> and <password>

Examples:

|username|Password|

|prathap456|sriBright@123|

|prathp876|abcdef567|

--This uses Example keyword to define the test data for the Scenario

--This works for the whole test.

--Cucumber automatically run the complete test the number of times equal to the number of data in the Test Set.

Test Data: --No keyword is used to define the test data. --This works only for the single step, below which it is defined.

--A separate code is need to understand the test data and then it can be run single or multiple times but again just for the single step, not for the complete test.

**Data Tables(With List): (In Eclipse: NewTask)**

The main difference between with example keyword and without is

**In example**: we write all data driven in tha last after example keyword like above

**W/o example (data driven)**: we write data driven here after every step like below .

In data Tables we use “Scenario” only.

**Its not a great approach, because if we have 2 sets of data, here we have to write code again and again. With Example key word no need to write code again.**

In this:

* We have to initialize data table for methods
* We have to create variable for data table ex; credentials
* Then we have to initialize above variable(credentials) with raw method

**credentials. Raw();**

* Raw() method give you List<List<String>>
* Then we have to store Raw() method like
* **List<List<String>> data = credentials. Raw();**
* Raw() method gives you complete data table access.

*Then* User entered Username and Password

**|naveenk |test@123 |**

**(0th row, 0th column, 0th row column 1)**

|  |  |
| --- | --- |
| **User** | **pass** |
| **(0,0)** | **(0,1)** |
| **(1,0)** | **(1,1)** |

sendKeys(data.get(0).get(0));

**Data Tables (With Map Objects):**

It is a good approach, it will give clear idea for which key we are giving value. It is similar to “Data Tables”.

**Here we use Key and Value pair Format**

Ex: |Username(Key) |Password(Key) |

|Naveen k (Value) | test@123(Value)|

|prathap B(value) | test@456(value) |

Here we use **.asMaps** objecthere. Here map object is “map<string,string>”.

**Here we have to use for Loop to parameterize the values.**

Ex: for(map<string,string> data : credentials.asmap(String.class, string.class)

**Cucumber Tags**

We have to use **tags** option to execute particular tag steps,

Ex: tags= {“@SmokeTest”}

**OR**

Here we use **,** as like **“or “** operator

Ex:tags= {“@SmokeTest,@RegressionTest”}

To execute all tests tagged as both mentioned tags

**AND**

Here also we use , only but we mentione like below

Tags = {“{“@SmokeTest”,” @RegressionTest ”}

To ignore particular tag we use “ **~** ”

Ex: tags= {“~@smoketest”, “@RegressionTest”}

tags= {“~@smoketest”, “~@RegressionTest”,”E2ETest”}

**HOOKS in Cucumber**

**@Before**

**@After**

It will help to reduce the code to write again and again for every scenario.

Ex:

@before

Login to homepage with username and password

@After

Close the browser

Scenario: Create new deal

Code

Scenario: Create New Case

Code..

**Tagged Hooks**

**I)Global Hooks**

These hooks applicable for all scenarios in the test case, like

@before, @after

II) Loacl Variables:

These hooks for particular Scenario only.

Tagged hooks are if we wn executeany pre condition for specific scenario, we have to tagge hooks.

Ex: Feature

@first

Scenario

@second

Scenario

@third

Scenario

…………………………………..

Ex: StepDefination

@before (order=0) Global

@after (order=0)

@before (order=1)

@after(order=1)

We can give order for hooks also, When we have to execute 2 or more pre conditions.

Order=0 will execute firest then order=1.

@before(“@first”)

Pre condition

@after(“@first”)

Precondition

…Write code for above featutres

**POM with Cucumber**

This feature will use to create configure and properties file, which are going to use again and again in our framework