BDD Tools Cucumber

***What is Cucumber?***

***Cucumber*** is a testing framework which supports ***Behavior Driven Development (BDD).***It lets us define application behavior in plain meaningful English text using a simple grammar defined by a language called ***Gherkin***. Cucumber itself is written in ***Ruby***, but it can be used to “test” code written in *Ruby* or other languages including but not limited to *Java*, *C#* and *Python.*

  It also allows to write specification in human readable [***Gherkin format***](http://toolsqa.wpengine.com/cucumber/gherkin-business-driven-development-bdd-language/).

Why BDD Framework?

Let’s assume there is a requirement from a client for an E-Commerce website to increase the sales of the product with implementing some new features on the website. The only challenge of the development team is to convert the client idea in to something that actually delivers the benefits to client.

The original idea is awesome. But the only challenge here is that the person who is developing the idea is not the same person who has this idea. If the person who has the idea happens to be a talented software developer, then we might be in luck: the idea could be turned into working software without ever needing to be explained to anyone else. Now the idea needs to be communicated and has to travel from Business Owners(Client) to the development teams or many other people.

***Gherkin***

## A language above is called *Gherkin* and it implements the principles of *Business readable domain specific language(BRDSL)*. Domain specific language gives you the ability to describe your application behavior without getting into details of implementation. What does that mean? If we go back to our tutorial in [*TDD*](http://toolsqa.wpengine.com/cucumber/test-driven-development-tdd/) we saw that we wrote test code before writing any application code. In a way we described what is the expected behavior of our application in terms of tests.

## Cucumber Introduction:

* Cucumber is a framework which supports BDD-Behavior Driven Development
* In BDD Automation programs are created based on the behavior of the application
* Cucumber was initially implemented with Ruby, later it was extended to Java, C#

## Cucumber Feature File:

A ***Feature File*** is an entry point to the *Cucumber* tests. This is a file where you will describe your tests in Descriptive language (Like English). It is an essential part of Cucumber, as it serves as an automation test script as well as live documents. A feature file can contain a scenario or can contain many scenarios in a single feature file but it usually contains a list of scenarios

In Cucumber the automation programs are created based on a file called Feature File.

In this feature file the task to be automated is written in plain English statements which are connected to the selenium code which performs those activities. The advantage of this process is the flow of the automation programs can be understood by Non-Technical people like Stake-Holders or Business Analyst.

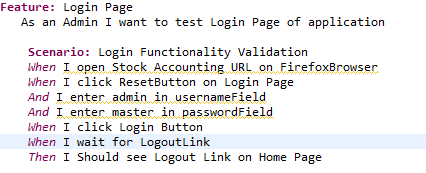
Note: Cucumber uses its own language called as Gherkin

## Cucumber Keywords:

Feature: This represents the module or functionality that is under tests

Scenario: This represents the test case that is been automated in a particular feature

Note: one Feature have one or more Scenarios.



Given: This represents the pre-condition of the test case

When: This represents the exact action that is performed in the test case

And: This represents any additional actions that should be performed on the test case

Then: This represents the outcome of the test case

Background: Whenever any step is required to perform in each scenario then those steps need to be placed in Background. For Instance: If a user needs to clear database before each scenario then those steps can be put in a background.

A Step Definition is a small piece of *code* with a *pattern* attached to it or in other words a Step Definition is a java method in a class with an annotation above it. An annotation followed by the pattern is used to link the *Step Definition* to all the matching *Steps*, and the *code* is what *Cucumber* will execute when it sees a *Gherkin Step*. *Cucumber* finds the *Step Definition* file with the help of Glue code in ***Cucumber Options***.

## Cucumber Test Runner Class:

With a cucumber-based framework, you cannot run a feature file on its own. You will need to create a java class, which in turn will run this cucumber feature file. We call this class as cucumber test runner class.

Cucumber test runner class is one of the many mechanisms using which you can run Cucumber feature file. The test runner class also acts as an inter-link between feature files and **step definition classes**. It is in test runner class, that you provide the path for both feature file and **step defs class**.

With a test runner class, you have the option to run either a single feature file, or multiple feature files as well. For now, we will focus on running a single feature file

Example TestRunner:

**import** org.junit.runner.RunWith;

**import** cucumber.api.CucumberOptions;

**import** cucumber.api.junit.Cucumber;

**import** cucumber.api.testng.AbstractTestNGCucumberTests;

@RunWith(Cucumber.**class**)

@CucumberOptions(features="FeatureFiles",

glue= "stepDefinitions",

tags= {"@CreationTest"},

plugin = {"com.cucumber.listener.ExtentCucumberFormatter:Reports/report.html","pretty", "html:target/cucumber-reports"},

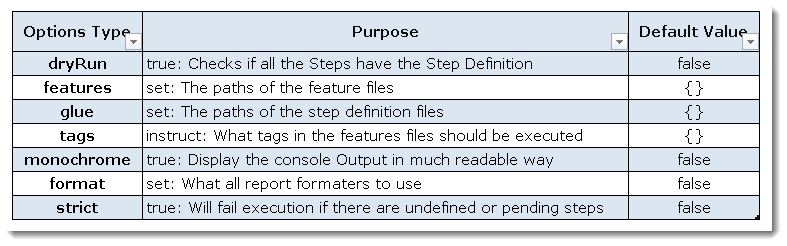
//monochrome = **true**

//,dryRun=true

)

**public** **class** Runner **extends** AbstractTestNGCucumberTests {

}



A Step Definition is a small piece of *code* with a *pattern* attached to it or in other words a Step Definition is a java method in a class with an annotation above it. An annotation followed by the pattern is used to link the *Step Definition* to all the matching *Steps*, and the *code* is what *Cucumber* will execute when it sees a *Gherkin Step*. *Cucumber* finds the *Step Definition* file with the help of Glue code in ***Cucumber Options***.

Create FeatureFiles folder at the project level and create one feature file in it

Create testRunner package and create a Runner.java under \src\test\java

Now run the test runner and observe the missing steps generated by cucumber

Create stepDefinitions package and create a StepDefinitions.java class under \src\test\java

Add the missing steps generated by cucumber to StepDefinitions.java class

Create a new source folder and name it as \src\test\resources and put driver files in it

Create Reports folder at the project level to store extent reports

Update the pom.xml file with Apache poi dependencies

Create TestData excel workBook in \src\test\resources

Create utility package in \src\test\java and create an excel Handler Class to get data from a cell and sheet

Create DataHelper class to transfer data to cucumber line in Create utility package in \src\test\java

**public** **class** DataHelper {

**public** **static** String getData(String rowSheetIndex,String colHdr) **throws** Throwable {

ExcelHandler tdtf=**new** ExcelHandler();

//String hypen="11-Login";

**int** hypenIndex=rowSheetIndex.indexOf("-");

String number=rowSheetIndex.substring(0, hypenIndex);

**int** rowNumber=Integer.*parseInt*(number);

String sheet=rowSheetIndex.substring(hypenIndex+1, rowSheetIndex.length());

//System.out.println(number+" "+sheet);

**int** rqColNumber=-1;

**for**(**int** i=0;i<tdtf.colCount(sheet, 0);i++) {

**if**(tdtf.getData(sheet,0,i).equalsIgnoreCase(colHdr)) {

rqColNumber=i;

}

}

**return** tdtf.getData(sheet, rowNumber, rqColNumber);

}

}

Create a scenario like below

Scenario Outline: Login Functionality Validation

When I open Stock Accounting URL on "Browser" from "<Row-SheetIndex>"

When I click ResetButton

And I enter "UserName" in usernameField from "<Row-SheetIndex>"

And I enter "Password" in passwordField from "<Row-SheetIndex>"

When I click Login Button

When I wait for LogoutLink

Then I Should see Logout Link

When I close the browser

Examples:

|Row-SheetIndex|

|2-Login|

|3-Login|

|4-Login|