January 1, 2015 Cucumber By Lakshay Sharma

So far in the series of Cucumber tutorial we have covered *Feature files, Gherkins, Step Definitions, Annotations, Test Runner Class* and many other things. There is no doubt that you cannot set up *BDD framework* until you know all the concepts but there are still few more areas which are very important to know in the life of Cucumber Automation such as *Cucumber Options, Regular Expressions, Page Object factory* and few others. Let's start with *Cucumber Options*.

us to do all the things that we could have done if we have used cucumber command line. This is very helpful and of utmost importance if we are using IDE such eclipse only to execute our project. You must have noticed that we set few options in the '***TestRunner'*** class in the previous chapter.

TestRunner Class

```
1  package cucumberTest;
2
3  import org.junit.runner.RunWith;
4  import cucumber.api.CucumberOptions;
5  import cucumber.api.junit.Cucumber;
6
7  @RunWith(Cucumber.class)
8  @CucumberOptions(
9          features = "Feature"
10         ,glue={"stepDefinition"}
11         )
12
13 public class TestRunner {
14
15 }
```

So in the above example we have just set two different *Cucumber Options*. One is for *Feature File* and other is for *Step Definition* file. We will talk about it in detail now but with this we can say that *@CucumberOptions* are used to set some specific properties for the *Cucumber* test.

Following Main Options are available in Cucumber:

Options Type	Purpose	Default Value
dryRun	true: Checks if all the Steps have the Step Definition	false
features	set: The paths of the feature files	{}
glue	set: The paths of the step definition files	{}
tags	instruct: What tags in the features files should be executed	{}
monochrome	true: Display the console Output in much readable way	false
format	set: What all report formaters to use	false
strict	true: Will fail execution if there are undefined or pending steps	false

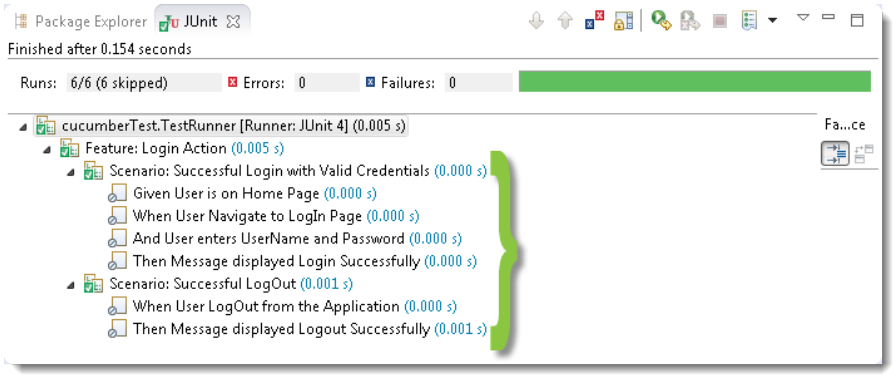
Dry Run

give us the message. For practice just add the code `***dryRun = true***` in ***TestRunner*** class:

TestRunner Class

```
1  package cucumberTest;
2
3  import org.junit.runner.RunWith;
4  import cucumber.api.CucumberOptions;
5  import cucumber.api.junit.Cucumber;
6
7  @RunWith(Cucumber.class)
8  @CucumberOptions(
9          features = "Feature"
10         ,glue={"stepDefinition"}
11         ,dryRun = true
12         )
13
14 public class TestRunner {
15
16 }
```

Now give it a run by Right Click on ***TestRunner*** class and Click ***Run As > JUnit Test.*** *Cucumber* will run the script and the result will be shown in the left hand side *project explorer window* in *JUnit* tab.



Take a look at the time duration at the end of the every *Steps*, it is (***0.000s***). It means none of the *Step* is executed but still *Cucumber* has made sure that every Step have the corresponding method available in the *Step Definition* file. Give it a try, remove the `***@Given("^User is on Home Page$")***` statement from the ***Test_Steps*** class and run the ***TestRunner*** class again. You would get the following message:

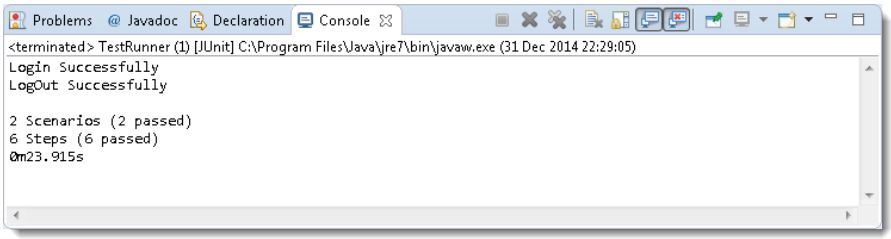
Monochrome

This option can either set as ***true*** or ***false***. If it is set as *true*, it means that the *console output* for the *Cucumber test* are much more readable. And if it is set as *false*, then the *console output* is not as readable as it should be. For practice just add the code '***monochrome = true***' in ***TestRunner*** class:

TestRunner Class

```
1  package cucumberTest;
2
3  import org.junit.runner.RunWith;
4  import cucumber.api.CucumberOptions;
5  import cucumber.api.junit.Cucumber;
6
7  @RunWith(Cucumber.class)
8  @CucumberOptions(
9          features = "Feature"
10         ,glue={"stepDefinition"}
11         ,monochrome = false
12         )
13
14 public class TestRunner {
15
16 }
```

Now give it a run by Right Click on ***TestRunner*** class and Click ***Run As*** > ***JUnit Test.*** *Cucumber* will run the script and Console Output will display like this:



This time change the value from *true* to *false* and run the ***TestRunner*** class again. This time the *Console Output* will look like this:

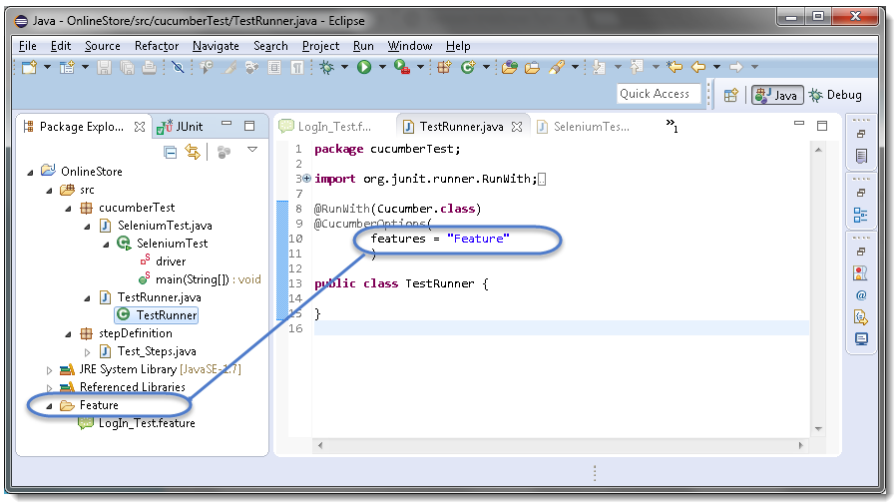
Features

Features Options helps *Cucumber* to locate the *Feature file* in the project folder structure. You must have notices that we have been specifying the *Feature Option* in the ***TestRunner*** class since the first chapter. All we need to do is to specify the folder path and *Cucumber* will automatically find all the `***.features***` extension files in the folder. It can be specified like:

features = "Feature"

Or if the Feature file is in the deep folder structure

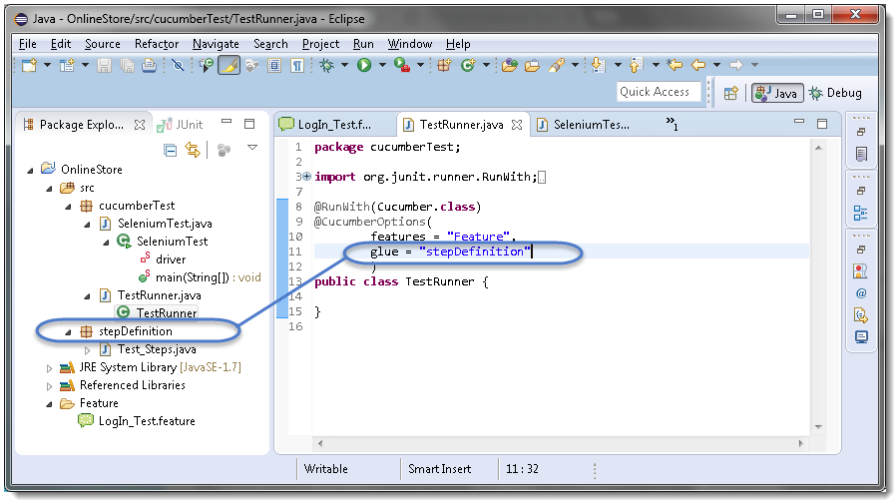
features = "src/test/features"



Glue

It is almost the same think as *Features Option* but the only difference is that it helps *Cucumber* to locate the ***Step Definition file.*** Whenever *Cucumber* encounters a *Step*, it

glue = "src/test/stepDeinition"



Format

Format Option is used to specify different formatting options for the output reports. Various options that can be used as for-matters are:

Pretty: Prints the *Gherkin* source with additional colours and stack traces for errors. Use below code:

format = {"pretty"}

HTML: This will generate a HTML report at the location mentioned in the for-matter itself. Use below code:

format = {"html:Folder_Name"}

JSON: This report contains all the information from the gherkin source in JSON Format. This report is meant to be

JUnit: This report generates XML files just like Apache Ant's JUnit report task. This XML format is understood by most Continuous Integration servers, who will use it to generate visual reports. use the below code:

format = { "junit:Folder_Name/cucumber.xml"}

For more updates on Cucumber Tutorial, please Subscribe to our Newsletter.

Please ask questions on ForumsQA, in case of any issues or doubts.

Cucumber

About the author

Lakshay Sharma

I'M LAKSHAY SHARMA AND I'M A TEST AUTOMATION ENGINEER. Have passed 11 years playing with automation in mammoth projects like O2 (UK), Sprint (US), TD Bank (CA), Canadian Tire (CA), NHS (UK) & ASOS(UK). Currently I am working with **BLOOMREACH** as SDET. I am passionate about designing

SUBSCRIBE TO NEWSLETTER

Enter your email address:

Subscribe

GOT SELENIUM PROBLEMS ?

TOOLSQA

wife and a lovely daughter. Please connect with me at **LinkedIn** or follow me on **Instagram**.

Related posts

Background in Cucumber

October 8, 2017

Execution Order of Hooks

October 7, 2017

Tagged Hooks in Cucumber

October 4, 2017

Cucumber Hooks

October 3, 2017

Cucumber Tags

September 30, 2017

Maps in Data Tables

December 15, 2015

RECENT POST

- Automated Testing with the Selenium Automation Tool
- Run Cucumber Test from Command Line / Terminal
- Cucumber Extent Report
- Cucumber Reports
- Share data between steps in Cucumber using Scenario Context
- Handle Ajax call Using JavaScriptExecutor in Selenium?
- Data Driven Testing using Json with Cucumber
- How to use Hooks in Selenium Cucumber Framework

Got a Question?

Site Links

Jobs in India
Selenium Training
Corporate Training
OPT Training & Placements
Video Tutorials
About US

Tutorials

Software Testing Tutorial
Selenium Tutorial in Java
Selenium Tutorial in C#
Cucumber Tutorial Java

Author

I'M LAKSHAY SHARMA AND I'M A

Contact US

SITEMAP

TestNg Tutorial

JUnit Unit Testing

Maven Tutorial

Core Java Tutorial

automation in mammoth projects like O2 (UK), Sprint (US), TD Bank (CA), Canadian Tire (CA), NHS (UK) & ASOS(UK). Currently I am working with **BLOOMREACH** as SDET.

I am passionate about designing Automation Frameworks that are effective and easy to maintain. For automating websites my weapons are **QTP** and **Selenium (Webdriver)**. I live in Amsterdam(NL), with my wife and a lovely daughter.

Please connect with me at **LinkedIn** or follow me on **Instagram**.