**Cucumber 3.0**

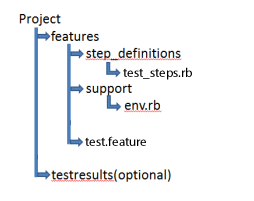
**Ruby 2.4.1 jruby 1.7**

**cucumber framework**

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

**Advantages of Cucumber**

* You can inolve business stakeholders who can not code
* End user experience is priority

[](http://www.swtestacademy.com/wp-content/uploads/2016/03/base_folder1.png)

Test Driven Development

**TDD** is an iterative development process. Each iteration starts with a set of tests written for a new piece of functionality. These tests are supposed to fail during the start of iteration as there will be no application code corresponding to the tests. In the next phase of the iteration Application code is written with an intention to pass all the tests written earlier in the iteration. Once the application code is ready tests are run.

Any failures in the test run are marked and more Application code is written/re-factored to make these tests pass. Once application code is added/re-factored the tests are run again. This cycle keeps on happening till all the tests pass. Once all the tests pass we can be sure that all the features for which tests were written have been developed.

## Benefits of TDD

1. Unit test proves that the code actually works
2. Can drive the design of the program
3. Refactoring allow to improve the design of the code
4. Low Level regression test suite
5. Test first reduce the cost of the bugs

## Drawbacks of TDD

1. Developer can consider it as a **waste of time**
2. The test can be targeted on verification of classes and methods and not on what the code really should do
3. Test become part of the maintenance overhead of a project
4. Rewrite the test when requirements change

Behavior Driven Development

Behavior Driven testing is an extension of TDD. In BDD also we write tests first and the add application code.

*Tests are written in plain descriptive English type grammar*

* *Tests are explained as behavior of application and are more user focused*
* *Using examples to clarify requirements*

This difference brings in the need to have a language which can define, in an understandable format.

BDD(Tests,applicaton code---***Gherkin***)----cucumber(runs the tests---ruby)

BDD(Tests,applicaton code---***Gherkin***)----specflow(runs the t

ests---.Net)

## What is 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 ***Plain English line with simple grammar*** ). 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. Let’s create one such file.

Feature

Scenario 0

Given When Then And

Scenario 1

Given When Then And

Feature file in cucumber consist of parameters or conditions required for executing code, they are

* Feature
* ***Background***
* Scenario

Scenario Outline Same scenario can be executed for multiple sets of data using scenario outline.  The data is provided by a tabular structure separated by (I   I).

* Given
* When
* Then
* ***And***
* ***But***
* ***\****

**Feature: Defines what feature you will be testing in the tests below(TITLE)**

**Background** 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***

Each scenario/test can be basically broken down into three parts:

* **Given**: **pre-condition of the test** It specifies the context of the text to be executed.
* **When**: "When" specifies the test action that has to performed
* **Then**: The expected outcome of the test can be represented by "Then"
* **And: Defines additional conditions of the test**

***Feature****: LogIn Action Test*  
*Description: This feature will test a LogIn and LogOut functionality*

***Scenario****: Successful Login with Valid Credentials*  
***Given****User is on Home Page*  
***When****User Navigate to LogIn Page*  
***And****User enters UserName and Password*  
***Then****Message displayed LogIn Successfully*

## But Keyword

***But***keyword is used to add negative type comments. It makes sense to use But when you will try to add a condition **which is opposite to the premise your test is trying to set**.

***Feature****: LogIn Action Test*  
*Description: This feature will test a LogIn and LogOut functionality*

***Scenario****: Unsuccessful Login with InValid Credentials*  
***Given****User is on Home Page*  
***When****User Navigate to LogIn Page*  
***And****User enters UserName and Password*  
***But****The user credentials are wrong*  
***Then****Message displayed Wrong UserName& Password*

## \* Keyword

This keyword defies the whole purpose of having Given, When, Then and all the other keywords

that all the keywords can be replaced by the***\* keyword*** and your test will just work fine.

**step definition**

A step definition is the actual code implementation of the feature mentioned in feature file

Step definition maps the test case steps in the feature filesto code, which executes and checks the outcomes from the system under test. For a step definition to be executed, it must match the given component in a feature.

Step definition is defined in ruby files under "features/step\_definitions/\*\_steps.rb".

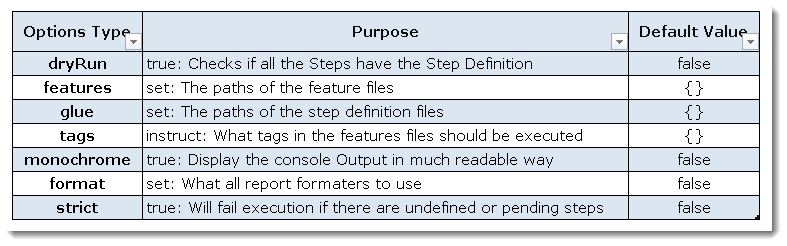
***Step Definition: Test\_Steps Class***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | package stepDefinition;    import java.util.concurrent.TimeUnit;    import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;    **import cucumber.api.java.en.Given;**  **import cucumber.api.java.en.Then;**  **import cucumber.api.java.en.When;**    public class Test\_Steps {  public static WebDriver driver;  @Given("^User is on Home Page$")  public void user\_is\_on\_Home\_Page() throws Throwable {          driver = new FirefoxDriver();          driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);          driver.get("http://www.store.demoqa.com");  }    @When("^User Navigate to LogIn Page$")  public void user\_Navigate\_to\_LogIn\_Page() throws Throwable {  driver.findElement(By.xpath(".//\*[@id='account']/a")).click();  }    @When("^User enters UserName and Password$")  public void user\_enters\_UserName\_and\_Password() throws Throwable {  driver.findElement(By.id("log")).sendKeys("testuser\_1");      driver.findElement(By.id("pwd")).sendKeys("Test@123");      driver.findElement(By.id("login")).click();  }    @Then("^Message displayed Login Successfully$")  public void message\_displayed\_Login\_Successfully() throws Throwable {  System.out.println("Login Successfully");  }    @When("^User LogOut from the Application$")  public void user\_LogOut\_from\_the\_Application() throws Throwable {  driver.findElement (By.xpath(".//\*[@id='account\_logout']/a")).click();  }    @Then("^Message displayed Logout Successfully$")  public void message\_displayed\_Logout\_Successfully() throws Throwable {          System.out.println("LogOut Successfully");  } |

Cucumber finds the Step Definition file with the help of Glue code in ***Cucumber Options***

*@Cucumber Options* are used to set some specific properties for the *Cucumber* test.

Following Main Options are available in Cucumber:



Feature: Yahoo search Action Test

Description: This feature will test a search functionality in Yahoo page

Scenario: Search on yahoo

Given user is on yahoo home page

When user enters in search box about “motorcycle sales”

And user clicks on the search button

Then user will find the results

env.rb

 require ‘selenium-webdriver’

Before do

$driver = Selenium : : WebDriver.for :firefox

end

After do

$driver.quit

End

Featurename.feature

Feature: Yahoo Searching Action Test

I can search for ‘ motor cycle sales’ on yahoo.com and get relevant results for the search and

find the best deals near me

Scenario: Search on Yahoo

Given user is on yahoo home page

When user enters “motorcycle sales” in search box

And user clicks on the search button

Then user will find the results

Stepdef.rb

Given(/^ user is on Yahoo Home Page$/) do

$driver.get<http://www.yahoo.com>

End

When(/^ user enters in search box about “([^”]\*)”$/) do |arg1|

$driver.find\_element(:id, “uh-search-box”).send\_keys(“motorcycle sales”)

End

When(/^user clicks on the search button$/) do

$driver.find\_element(:id, “uh-search-box”).click

Wait = Selenium: :WebDriver: : Wait.new(:timeout =>15)

End

Then(/^user will find the results$/) do

Sleep 5

str=$driver.find\_element(:xpath,”//\*[@id=’web’] ..”).text

puts str

end