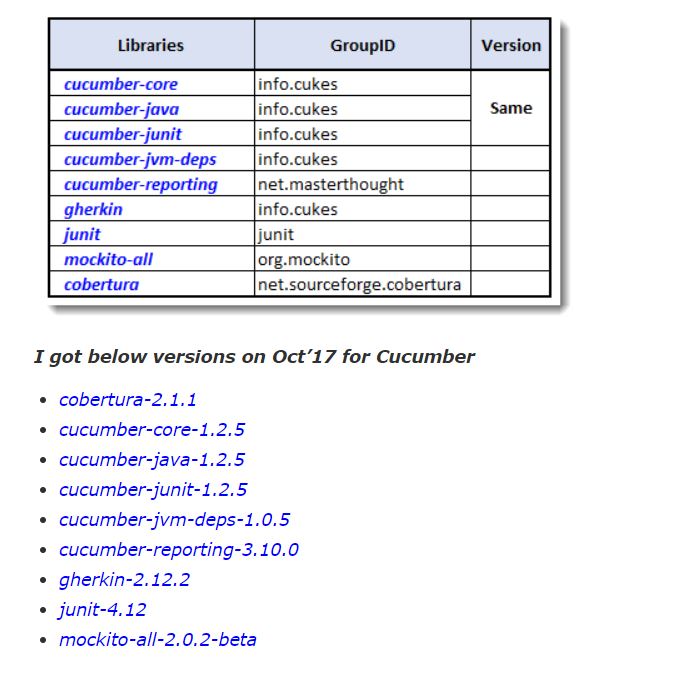
Cucumber is a software tool used by computer programmers that supports behavior-driven development. Central to the Cucumber BDD approach is its plain language parser called Gherkin. It allows expected software behaviors to be specified in a logical language that customers can understand.

**Requirements**

1. Download JDK
2. Setup Environment Variables:
   1. Go to Control Panel\System and Security\System
   2. Click Advanced System Settings > Environment Variables
   3. Click on New button from System variables section
   4. Enter JAVA\_HOME in Variable Name field and JDK folder location in Variable value field and click OK button
   5. Again Open Environment Variables section
   6. System variables > Double click “Path”
   7. Add JDK bin location as New item in the list
   8. Verify both these actions by using commands “javac -version” and “where javac”
3. **Not needed if dependencies are added in pom file.** Download Cucumber jar files from “<https://search.maven.org>” as below.



1. Download Selenium with all jar files

**Create new project and execute**

1. Open Eclipse and create new Maven Project
2. Delete default packages under src folders
3. Add dependencies in the pom.xml file - Maven dependencies will be added to the library
4. Click on src/main/java folder and create New > Package as “Features”
5. Right click on Features package and create New > File - feature file Ex: Login.feature (check if Natural 0.7.6 is installed in the eclipse - Refer below for installation details)
6. Add test case to feature file
7. Click on src/main/java folder and create New > Package as “stepDefinitions”
8. Right click on this package and create New > File Ex: LoginStep
9. Add Selenium code to this class
10. Click on src/main/java folder and create New > Package as “runner”
11. Right click on this package and create New > File Ex: TestRunner
12. Add runner code to test runner class
13. Run the runner file - Once done, missing steps will be displayed in the console window
14. Copy the steps from the console window and paste in the LoginStep file
15. Now update glue step in test runner class as below

*@CucumberOptions(features=\*\*path of feature file\*\*, glue=\*\*path of steps file\*\*)*

1. Run the test runner class to view the execution and results.

**Natural Plugin Installation**

1. Go to Eclipse Marketplace and search for “Natural”
2. If it is listed - click install
3. If not - download from “marketplace.eclipse.org”
4. Just drag and drop the plugin in the eclipse

**Cucumber Options**

* dryRun - check the mapping of feature file and step definition
* features - path of the feature file
* glue - path of the step definition
* format - to get the output in required format
* strict - fail the run if any step is not defined in step definition
* monochrome - to get output in proper readable format
* tags - to sun particular scenario from the feature file
  + Or : tags = {“@SmokeTest , @RegressionTest”}
  + And : tags = {“@SmokeTest” , “@RegressionTest”}
  + Add ~ before test which needs to be ignored

**DDF Using Cucumber**

1. Without Examples keyword - giving credentials directly in the feature file, Selenium code for expressions should be **\"(.\*)\"**

Ex:

@Then("^User enters \"(.\*)\" and \"(.\*)\"$")

public void user\_enters\_Username\_and\_Password(String username, String password)

1. With Examples + Scenario Outline

Ex:

Feature: Home Page

Scenario Outline: Successful Login with Valid Credentials

Given User enters the Academy URL

When User clicks Login button

Then User enters "<username>" and "<password>"

Then User clicks Submit button

Then User should be taken to homepage

Then Closes the browser

Examples:

| username | password |

| admin | fl33t |

| jalizha | test123 |

| admin | fl33t |

1. Using Tables - not much needed

**Hooks in Cucumber**

@Before and @After are the hooks in Cucumber

Both will executed for each scenario and each iteration

**TaggedHooks**

@Before(“@tagname”)

@After(“@tagname”)

**Priority**

Global hooks can be executed in priority order as below

@Before(order=0)

@After(order=0)