**CUCUMBER – BDD Framework**

* BDD is a Behavioral Driven Development
* BDD means defining the behavior of the framework like behavior of the test cases & behavior of the use case
* Behavior means simple plain English sentence we are writing with the help of language Gherkin Keyword, Gherkin is the language to define the framework
* Gherkin is not a programming lang, it is a feature language, in which we have to define some features in the form of BDD
* In TestNG & Junit these are TDD Frameworks, here we are writing the code in the form of Annotations plus each & every thing we are writing in the form of coding, but in BDD Framework we define the behavior, on the basis of behavior we write the test cases in the background
* Cucumber with Jvm/Ruby, but Cucumber – JVM with selenium is most famous in the market.
* Another tool in BDD is JBehave(Java Behavior) is only for Java, it is complex, not user friendly, so everyone are using Cucumber
* Cucumber can be integrated with selenium, webservices(RestApi), even developers can use for unit level test cases
* Even manual testers, business Analyst, managers, Developers… etc can create feature file by using Gherkin keywords, & they give it us, we can convert into step Definitions
* Cucumber has 3 different Components:

1) **Feature file :** Feature is a .feature file, in this particular feature file we will be using Gherkin Keywords like Given, When, And, But & Then, \*…etc, with the help of these keywords we will define a scenario or scenario outline for a specific Feature, Feature is again one more keyword is available, in the feature file for that particular feature file we will define the different steps

* All the Test case steps are defined by Given, When, And ,Then, But Keywords
* Scenario will be defined by Scenario or Scenario Outline
* Feature means what exactly the Feature is lets say eg: Login Feature, HomePage Feature, Search Feature…etc, Feature means Feature name
* We can create number of scenarios in a Feature file for diff functionalities

2) **Stef Definition file :**

* After writing Feature file corresponding code will be writing into step definition file, here we are using selenium code, java lang & different Annotations

3) **TestRunner file :**

* We are writing TestRunner file in Junit
* we use TestRunner class to run our feature & to generate the o/p report, define tags

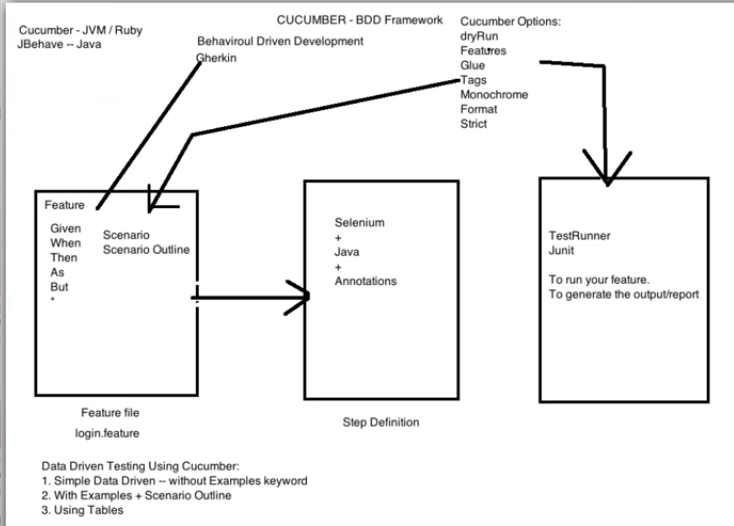
**Advantages :**

1. Entire Agile team can involve
2. Code reusable
3. Easy for reporting(user friendly)

**Disadvantages :**

1. It is very lengthy process, need to write so many steps

* To change color of keywords black to red in Feature file🡪 go to eclipse market place & install Natural plugin, if it is not found there, google it & drag the install icon to Feature file, it will install(<https://marketplace.eclipse.org/content/natural>)



Refer Url’s for some information:

1. <http://toolsqa.com/cucumber/cucumber-tutorial/>
2. <https://en.wikipedia.org/wiki/Cucumber_(software)>
3. <http://artoftesting.com/automationTesting/tddWithCucumber.html>
4. <https://www.tutorialspoint.com/cucumber/index.htm>
5. <https://www.guru99.com/introduction-to-cucumber.html>
6. <http://www.softwaretestinghelp.com/cucumber-bdd-tool-selenium-tutorial-30/>

**Imp Dependencies for Cucumber with selenium project:**

**Cucumber Dependencies: Selenium Dependencies:**

1. Cucumber-java 1) Selenium-java
2. Cucumber-jvm
3. Cucumber-junit, these 3 should be of

same version

1. Cucumber-jvm-deps
2. Cucumber-reporting
3. Gherkin
4. Junit

**Cucumber Options :**

Learn:

* What are the different Cucumber Options are available?
* How to use Cucumber Options?
* Benefits of Cucumber Options?

|  |
| --- |
|  |
|  | @CucumberOptions( |
|  | features = “classpath:features”, //the path of the feature files |
|  | glue={"stepDefinitions"}, //the path of the step definition files |
|  | format= {"pretty","html:test-outout", "json:json\_output/cucumber.json", "junit:junit\_xml/cucumber.xml"}, //to generate  different types of reporting |
|  | monochrome = true, //display the console output in a proper readable format- remove junk like numbers, brackets |
|  | strict = true, //it will check if any step is not defined in step definition file- it will throw error if any is not  implemented |
|  | dryRun = false //to check the mapping is proper between feature file and step def file – if we set dryRun = true, if any one step is not implemented in SD, it will not execute execute the remaining steps until we implement that missing step, always run with dryRun =True ,first to check whether all step definitions are defined or not. Then make it dryRun=false & run the actual execution  Dry run is used to step defs with feature file making sure all steps covered while Strict is a way of making sure all methods are not empty, If Strict is set true and a method is empty it will return an error message |
|  | tags = {"~@SmokeTest" , "~@RegressionTest", "~@End2End"} |
|  | ) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | //ORed : tags = {"@SmokeTest , @RegressionTest"} -- execute all tests tagged as @SmokeTest OR @RegressionTest |
|  | //ANDed : tags = tags = {"@SmokeTest" , "@RegressionTest"} -- execute all tests tagged as @SmokeTest AND @RegressionTest    **Cucumber Tags :** |
|  |  |

What are Cucumber Tags?

Learn:

* How to run Cucumber Tests in Groups with Cucumber Tags
* How to ignore Cucumber Tests
* Logically ANDing and ORing Tags
* Cucumber has already provided a way to organize your scenario execution by using tags in feature file. We can define each scenario with a useful tag. Later, in the runner file, we can decide which specific tag (and so as the scenario(s)) we want Cucumber to execute.
* Tag starts with “@”. After “@” you can have any relevant text to define your tag like @SmokeTests just above the scenarios you like to mark. Then to target these tagged scenarios just specify the tags names in the CucumberOptions as tags = {“@SmokeTests”}.
* Tagging not just specifically works with Scenarios, it also works with Features. Means you can also tag your features files. Any tag that exists on a Feature will be inherited by Scenario, Scenario Outline or Examples.
* In Cucumber @WIP tag indicates work in progress

**Hooks in Cucumber :**

What are Hooks in Cucumber? || JUnit Cucumber Tutorial - Scenario Hooks

* Cucumber supports hooks, which are blocks of code that run before or after each scenario. You can define them anywhere in your project or step definition layers, using the methods @Before and @After.
* Cucumber Hooks allows us to better manage the code workflow and helps us to reduce the code redundancy. We can say that it is an unseen step, which allows us to perform our scenarios or tests.
* Why Cucumber Hooks? In the world of testing, you must have encountered the situations where you need to perform the prerequisite steps before testing any test scenario.
* This prerequisite can be anything from:
* Starting a webdriver
* Setting up DB connections
* Setting up test data
* Setting up browser cookies
* Navigating to certain page or anything before the test
* In the same way there are always after steps as well of the tests like:
* Killing the webdriver
* Closing DB connections
* Clearing the test data
* Clearing browser cookies
* Logging out from the application
* Printing reports or logs
* Taking screenshots on error or anything after the test
* To handle these kind of situations, cucumber hooks are the best choice to use.

Unlike TestNG Annotaions, cucumber supports only two hooks (Before & After) which works at the start and the end of the test scenario. As the name suggests, @before hook gets executed well before any other test scenario, and @after hook gets executed after executing the scenario.

**Data driven Testing :**

* If I don’t want to hard code values in my script, then am going to use Data driven testing concept

**Cucumber BDD Framework with Data Driven Framework using Examples and Scenario Outline keywords in Feature file:**

**Learn:**

1. How to parameterize test cases with and without Examples keyword

2. Difference between Scenario and Scenario Outline in Feature file

3. Data Driven Framework in Cucumber – Selenium

4. Interview Questions on Cucumber

1. **Without scenario Outline & Examples keywords:** I**f** I have only one login functionality, then I will use this

Eg: username: Satish, password: 123456

Scenario: Free CRM create a new contact

Then user enters “Satish” and “123456”

1. **With scenario Outline & Examples keywords:** Suppose if have 10 usernames & 10 passwords, then I will use this

Eg: 1)username : SatishKumar, password: 12345

2)username : SatishReddy, password: 123456

.

.

10) username: SatishN, password: 1234567

* Here I don’t want to hard code values in scenario steps like above
* Here I will pass the values by using examples keyword, it is like an excel, we can store n number of values

Eg: Scenario Outline: Free CRM create a new contact

|  |
| --- |
|  |
|  |

Then user enters "<username>" and "<password>" 🡪this particular line parameterised

Examples:

| username | password | |🡪 pi

| Satish | 123456 |

| Satish123 | 12345678 |

|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |