**Cucumber Course Duration – 16-20 Hours**

**Cucumber Course Pre-Requisite**

* Basics of Selenium Web driver & Java

**Cucumber Course Syllabus**

**BDD-Introduction**

* Template in writing Business  Requirements
* QA role in BDD
* Template in writing test cases for Business Scenarios
* Advantages of BDD, Why it is so popular in recent time?

**Cucumber Basics**

* Why Cucumber
* When Cucumber

**Gherkin-Introduction**

* Overview

**Cucumber-Set up in Eclipse**

* Maven
* Cucumber-Junit /Junit
* Cucumber-java

**Cucumber Implementation**

1. Cucumber Tags
2. Comment in feature file
3. Annotations of cucumber
   * The methods annotated with **@Before** and **@After** are executed before and after every scenario regardless of where they are defined.

**Data driven testing with Cucumber**

* Cucumber Parameterization of test cases
* How to use Examples Keyword (Data Driven)
* How to use the Maps in Data tables

**Cucumber-Junit Introduction**

* What is Junit?
* Configuration of Junit
* Junit test Example?
* Junit Test Runner?
* Junit Test Suite
* Running Junit test from Eclipse

**Report**

* Basic Report
* Configuration with Extent Report

**Framework**

1. **Data Driven**
   * What is Data Driven Framework
   * Data driven Framework with excel
2. **Keyword Driven**
   * What is keyword driven Framework
   * Keyword-driven framework implementation
3. **POM Framework**
4. **Jenkins Installation-configuration**

**Cucumber Eclipse Plugin**

* It is not the main cucumber plugin for running the test as BDD. Even this is not a mandatory plugin but it is quite handy.
* This will helps eclipse to understand the basic *Gherkin* syntax and it works like a syntax highlighter.
* It highlights all the main syntax in the feature file which makes it more readable and clear.
* It also enables the run of the feature file independently, without the help of JUnit.

**Steps to install Cucumber Eclipse Plugin:**

* Launch the *Eclipse IDE* and from Help menu, click “**Install New Software**”.
* Type name as you wish, let’s take “**Cucumber**” and type “**http://cucumber.github.com/cucumber-eclipse/update-site**” as location. Click **OK**.

**Coding standard and Guidelines**

* You can distribute the steps in multiple java files
* Library functions can be called methods present in step definition class
* When step definition files are distributed in multiple package you need to provide the path in
  + glue={"google/","google1/"},
* you cannot have same step definition method in same package
* If can have same step definition in different class provided both the class present in different packages. But if you try to use both the packages in “glue” properties then it will give you the duplicated step def error
* If there is same scenario name in 2 feature file then only one scenario will be executed
* We cannot extends Step Defs classes if parent class having any cucumber annotation OR hooks. Normal class can be used as parent class.
* Sharing of selenium driver object we can use picocontainer dependency for dependency injection.

### Anchors

The regular expression I'm logged in matches I'm logged in and I'm logged in as an admin. To avoid ambiguous matches, use ^I'm logged in$.

The caret at the beginning anchors to the beginning of the string. The dollar at the end does the same with the end of the string. Use these with all your step definitions and you won’t have surprise matches.

**Question:**

1. Create demo file for AND & BUT annotation
2. Create demo file for multiple scenario in single feature file - Done
3. How to work with attachments
4. Do we need to read data from Excel sheet?
5. Command line execution
6. Parallel execution for cucumber feature file
7. Parallel execution using Selenium Grid
8. One feature file and multiple step def file.
9. Extents report generation code

**Cucumber**

* What is Behaviors Driven Development?
* Instead of writing your tests purely in code, with Cucumber you start by writing a human-readable user story. Then, you write code to run the story and perform test(s) based on it.
* If only coders are reading and writing the specs, forget Cucumber. If non-coders are actually participating, then Cucumber starts to make sense
* It offers a way to write tests that anybody can understand, regardless of their technical knowledge.

**What are the benefits?**

1. It is helpful to involve business stakeholders who can't easily read code
2. Cucumber focuses on end-user experience
3. Style of writing tests allow for easier reuse of code in the tests
4. Quick and easy set up and execution
5. Efficient tool for testing

**Parallel Execution in Cucumber**

Cucumber does not supportd parallel execution out of the box

1. We have to use maven's capability to invoke it in parallel. Refer [link](https://opencredo.com/running-cucumber-jvm-tests-in-parallel/)
2. Also there is a github project which uses custom plugin to execute in parallel. Refer [cucumber-jvm-parallel-plugin](https://github.com/temyers/cucumber-jvm-parallel-plugin)

Refer : <http://automationrhapsody.com/running-cucumber-tests-in-parallel/>

**How to demo parallel execution?**

* Create new java package in test/java/com
* Create multiple step defs file in the above created package.
* Make sure that your steps defs method not repeated in multiple class
* Create new package for feature files under test/resources/com
* Create multiple feature files under it and these files should use all the steps from the classes created
* Now use above paths in maven, refer above paragraph to configure maven

**Jenkins configuration**

* Plugins to install
  + Cucumber reports
  + HTML Publisher plugin
  + Ant Plugin
  + Build Timeout
  + Email Extension Plugin
  + GitHub Branch Source Plugin
  + Gradle Plugin
  + HTML Publisher plugin
  + Maven Integration plugin
  + Pipeline
  + Pipeline: GitHub Groovy Libraries
  + SSH Slaves plugin
  + Subversion Plug-in
  + TestNG Results Plugin
  + Timestamper
  + Workspace Cleanup Plugin
* Global Tool Configuration
  + Jenkins 🡪Manage Jenkins 🡪 Global Tool Configuration
    - Set Git path “C:\Program Files\Git\bin\git.exe”
    - Set name as “maven” and Maven path as “D:\Training\TrainingContent\SeleniumJars\_Software\apache-maven-3.5.0”

**Jenkins + Extent Report**

* If Extent report is not properly displayed in the browser (because of content security setting of Jenkins, please perform below steps)
  + Go to Jenkins installation directory i.e. C:\Users\admin\.jenkins
  + Open Jenkins.xml file
  + In the arguments tag please add ------hudson.model.DirectoryBrowserSupport.CSP="sandbox allow-scripts; script-src \* 'self' 'unsafe-inline' 'unsafe-eval'"
  + Save the file and restart the Jenkins from services

**TDD VS BDD**

* TDD
  + Tests written by developer
  + Run tests by dev, test failed
  + Develop code
  + Run Tests, test should pass
  + Refactor tests if required
  + Mostly written to test each unit level code/method (Unit Tests)
  + Covers low level design scenarios
* BDD
  + Stakeholders/Non-Technical defines behavior of application in simple English language
  + Design high level scenario to test expectation from application
  + Design normally for acceptance testing

<http://www.thetestroom.com/cucumber-parameter-tutorial/>

^ - This marks the beginning of the string

$ - This marks the end of the string

\ - This is used to allow using special characters in a Regex pattern such as " and .

\* - This is used to mean either zero, one of more

( ) - The text in these brackets are used to identify grouped characters

[ ] - This text is in these square brackets are used to match a single character using an or expression

. - This text will be used represent any character

|  |  |  |
| --- | --- | --- |
| **No.** | **Character Class** | **Description** |
| 1 | [abc] | a, b, or c (simple class) |
| 2 | [^abc] | Any character except a, b, or c (negation) |
| 3 | [a-zA-Z] | a through z or A through Z, inclusive (range) |
| 4 | [a-d[m-p]] | a through d, or m through p: [a-dm-p] (union) |
| 5 | [a-z&&[def]] | d, e, or f (intersection) |
| 6 | [a-z&&[^bc]] | a through z, except for b and c: [ad-z] (subtraction) |
| 7 | [a-z&&[^m-p]] | a through z, and not m through p: [a-lq-z](subtraction) |

|  |  |
| --- | --- |
| **Regex** | **Description** |
| . | Any character (may or may not match terminator) |
| \d | Any digits, short of [0-9] |
| \D | Any non-digit, short for [^0-9] |
| \s | Any whitespace character, short for [\t\n\x0B\f\r] |
| \S | Any non-whitespace character, short for [^\s] |
| \w | Any word character, short for [a-zA-Z\_0-9] |
| \W | Any non-word character, short for [^\w] |
| \b | A word boundary |
| \B | A non word boundary |

**Description:**

Do you want to write BDD style automation implementation with Cucumber, Gherkin and Java? You are at right place. We will cover starting from all basic setup and configuration required

This course will start from very basic level with no prior experience required and cover End-To-End automation with Jenkins. Cucumber is a Behavior Driven Development framework and will be used along with Gherkin, Selenium, Java, Maven, Ecllipse.

This course will also cover how you can push your written code to GIT (Version Control System) and use that code in Jenkins (CI) to run your automation and generate report. This is called as End-To-End Automation.

Basic understanding of Java or any other OOP language is required to understand the code.