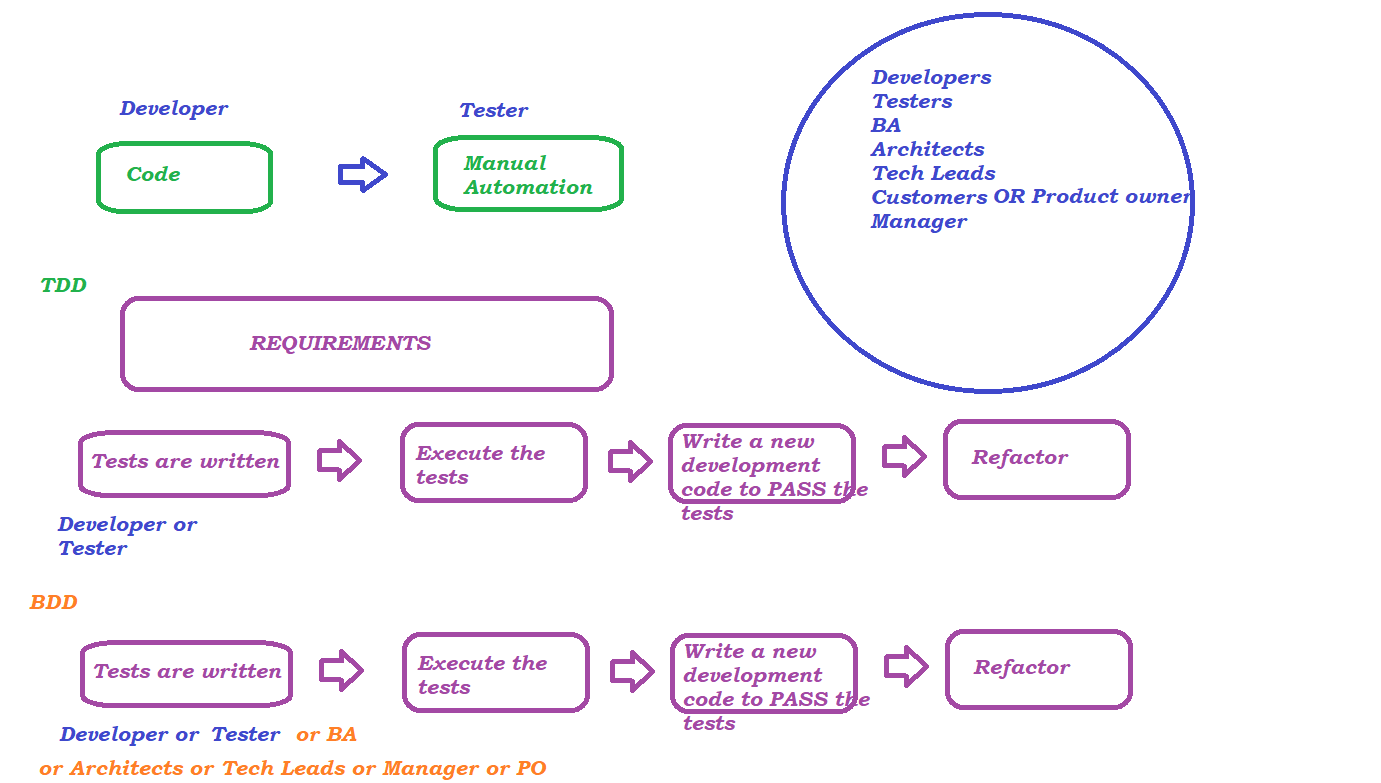
BDD with Cucumber

Introduction

****

In BDD whatever the test we are writing is in English Statements with some standard format, that is why we can involve everyone in testing activities and have more coverage in the Beginning of software development

Cucumber – Framework or Tool

Cucumber is a tool OR Framework which supports Behaviour Driven Development

Cucumber uses a language called **Gherkin**

Tests(English Statements) that are written in Gherkin language are called Feature files

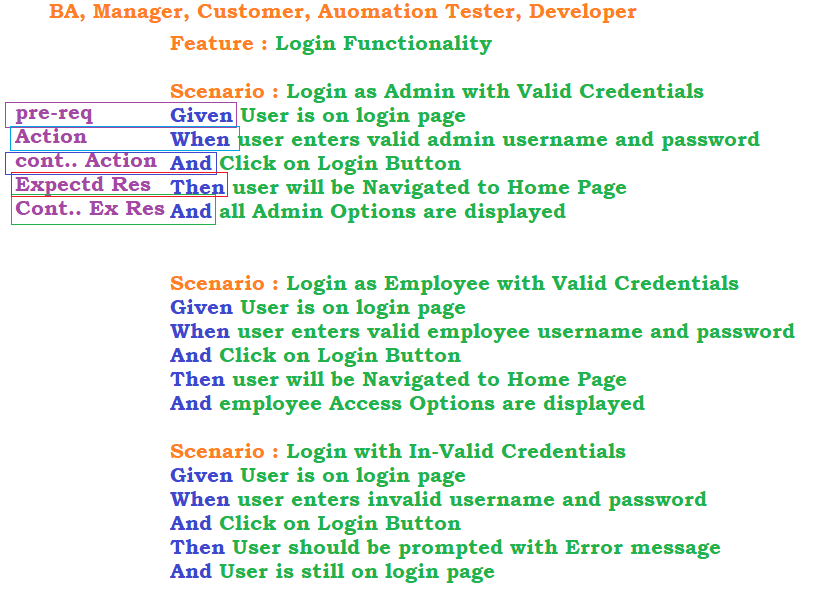
Feature files contains Keywords (Gherkin):

* Feature
* Scenario
* Given
* When
* Then
* And

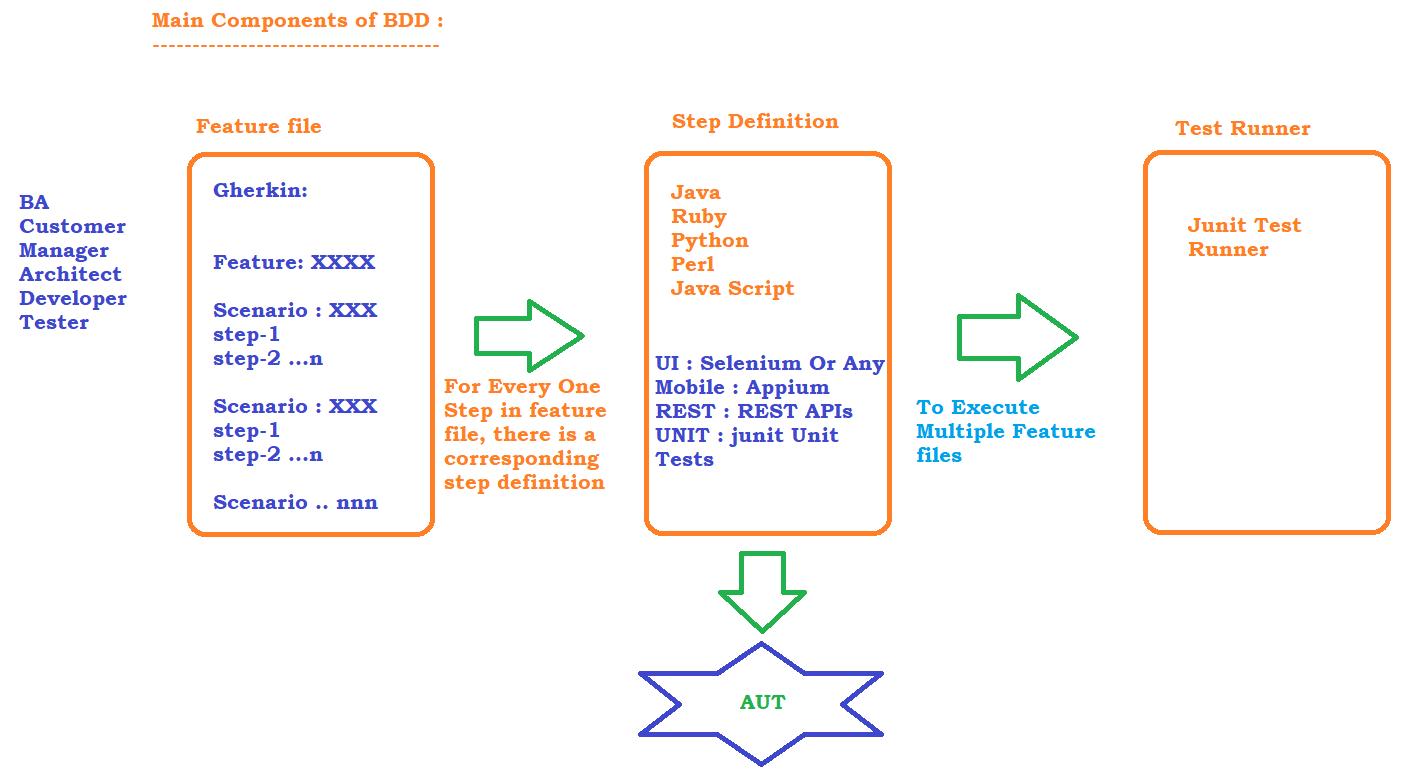
Syntax : Sample Feature file

******

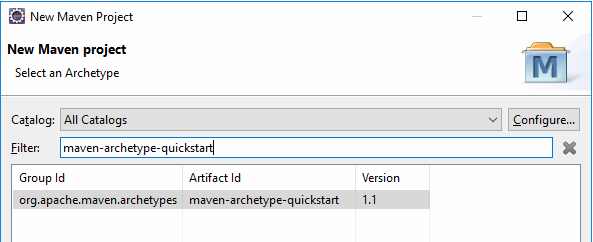
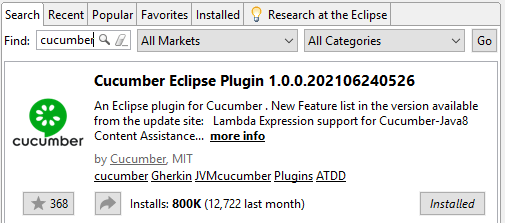
Sample Feature File



Components of Cucumber



Project Setup

1. Open Eclipse and create a Maven Project
2. Go with default workspace and click on Next
3. Select ArtifactID - maven-archetype-quickstart
   1. 
4. provide group id and Artifact id
5. click on finish
6. Verify the compiler and JRE Used for the project, If it is older version then update the complier and JRE
   1. Change the compiler from 1.5 to what is installed in PC
   2. Change the JRE to the JRE Present in JDK Location
7. Add Dependencies to pom.xml
   1. cucumber-java : 7.15.0
   2. cucumber-junit : 7.15.0
   3. junit : 4.13.2
8. Install Cucumber plugin.
   1. Help -> Eclipse Marketplace
   2. Search box -> cucumber
   3.  click on install to install the plug-in to eclipse and click on Finish to close the window
   4. Create below folders under src/test package
      1. features
      2. stepDefinitions
      3. testrunner
   5. Right click on the features folder and create a new file with .feature as a extension

First Feature File

1. create a file with .feature as extension
2. Run the feature file to get the missing step-definitions
3. copy the missing step definitions and put it in a Step definitions class and implement the step
4. Run the feature file to see the output

Test Runner

Test Runner is a Junit Class, Which has @RunWith and @CucumberOptions annotations

@RunWith holds the Cucumber Class and @CucumberOptions holds many information like,

- what is the feature file or feature files to be executed

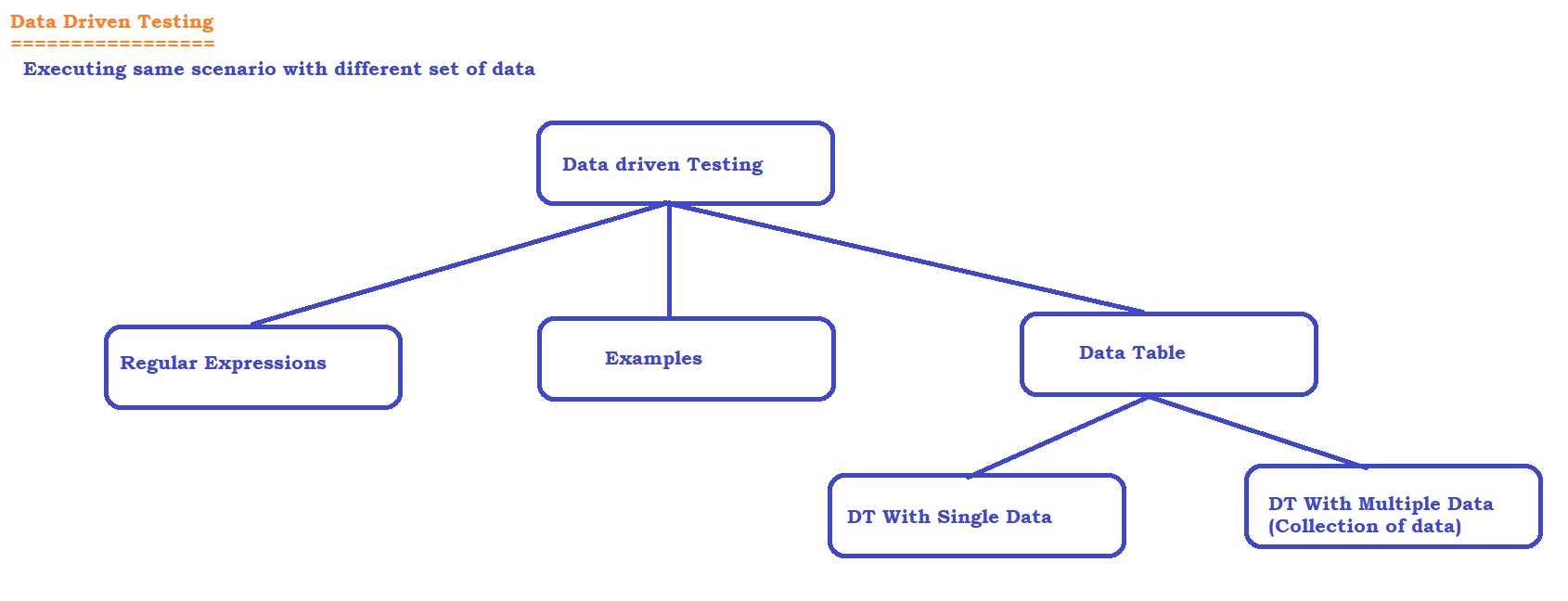
- where are the step definitions

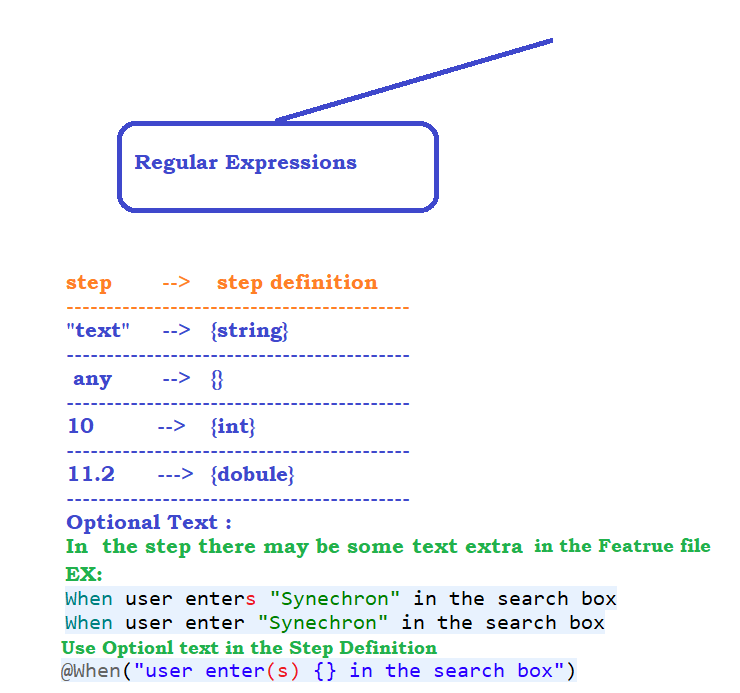
- What is the Run type – dry run or not

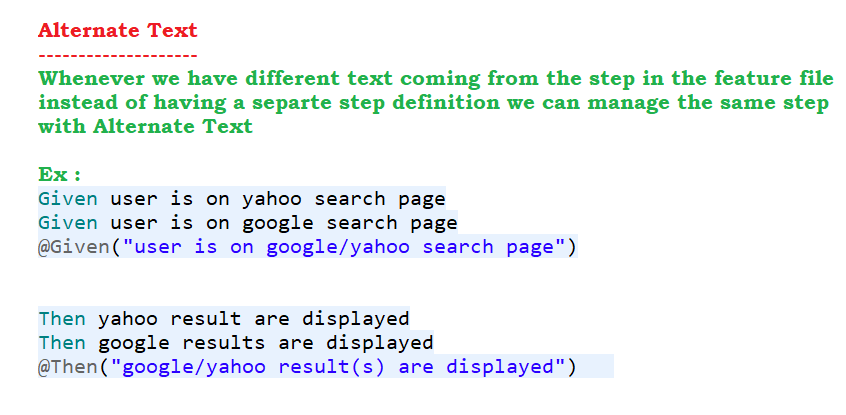
- What kind of Reports to be generated

- What all the tags to be considered while execution

Data driven Testing





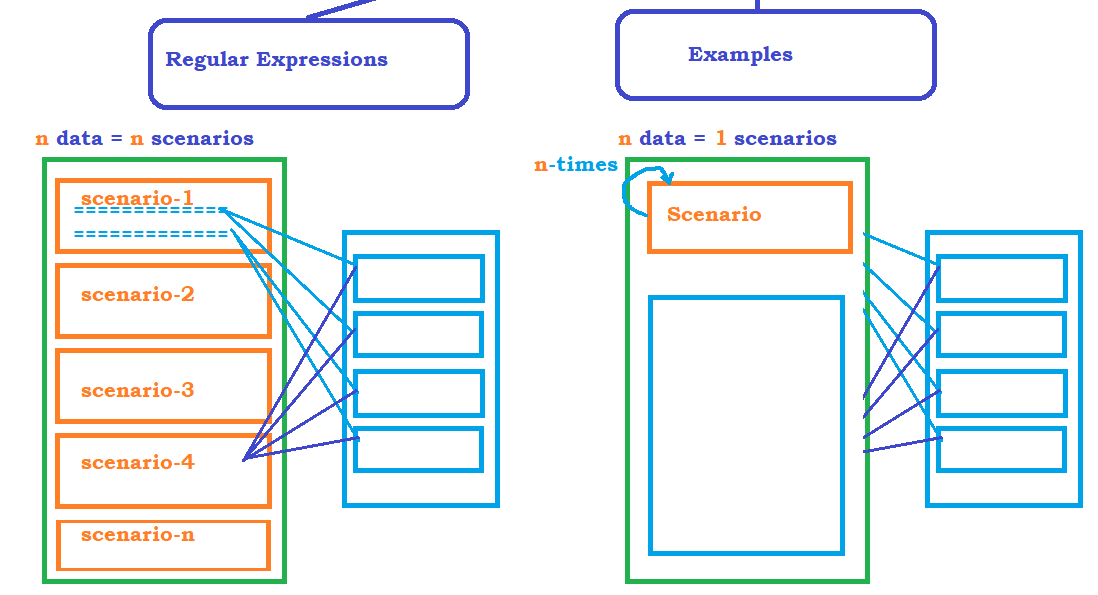


Examples in Scenario

Instead of having multiple Scenarios, each for one set of data, We can have only one scenario executed repeatedly for multiple set of test data using Example Table.

To use Example Table we have to use Scenario outline instead of Scenario

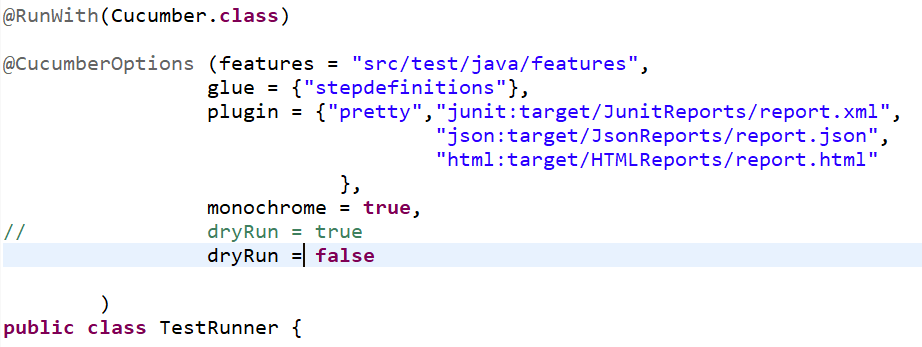
Example table is a data holder which holds the data with pipe separated



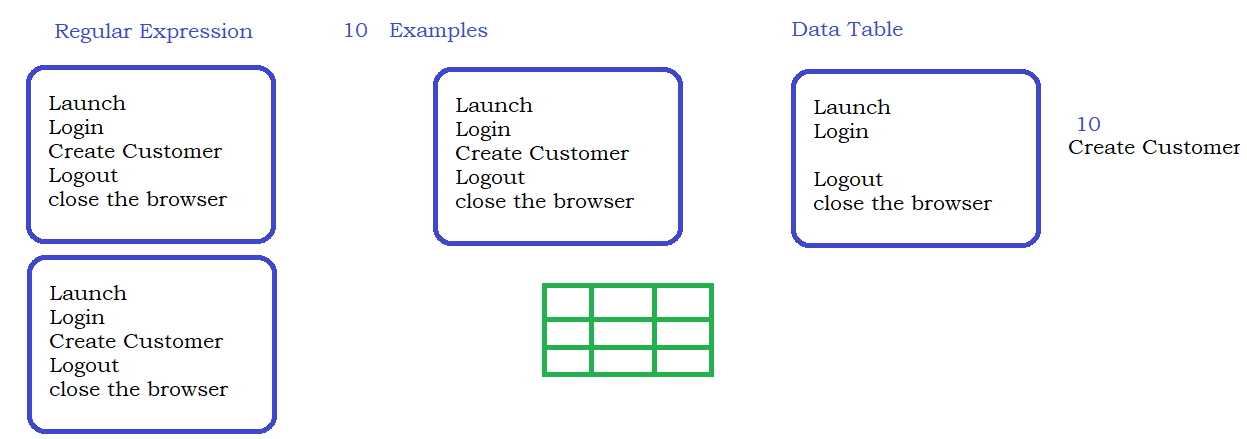
Reporter Plugin in Cucumber

Cucumber Test Runner provides a plugin using which we can generate 3 types of Reports

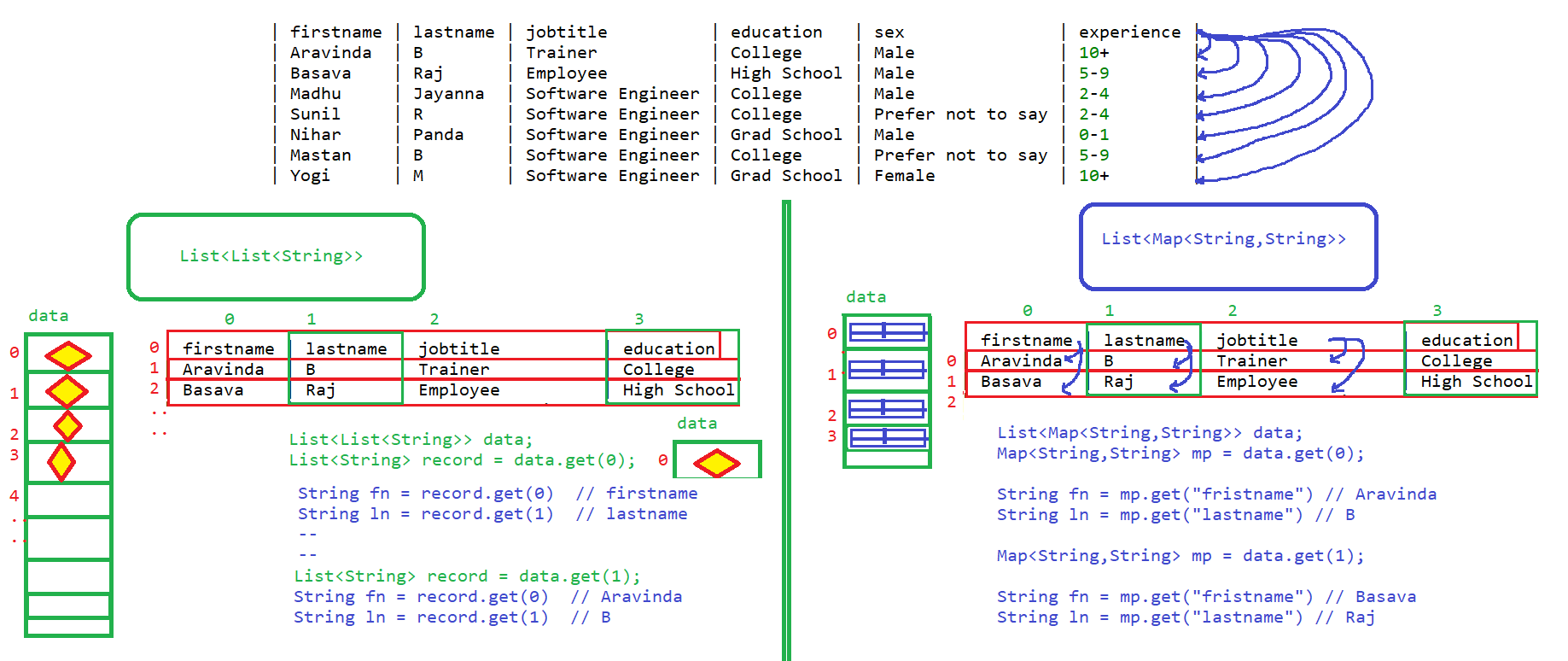
1. Junit xml Reports
2. Json Reports
3. HTML Reports



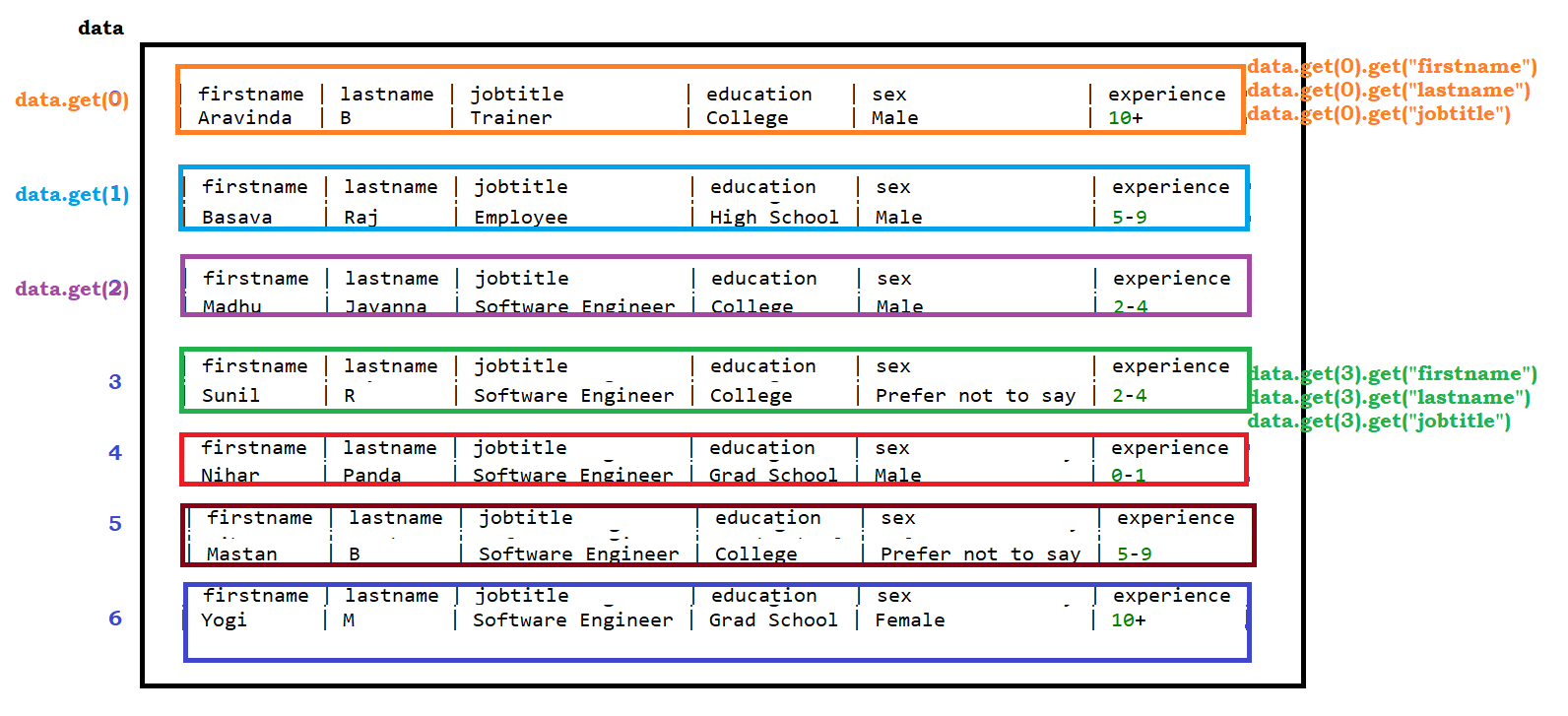
Difference Between RE / Examples / DT



Data table as List of List or List of Map



Datatable as List of Map :



Datatable as a List of List



Background in Cucumber

It is the pre-condition to the Scenario. We can avoid code duplicate if we use Background.

Any one who see the cucumber report will be able to understand there was a pre-condition executed.

Parallel Execution in Cucumber

1. Create a Maven Project and update all the required dependencies make sure to use junit-5

<!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>5.10.2</version>

</dependency>

1. Write feature files under feature folder which you created
2. Update the step definitions for the feature files
3. Write a Junit TestRunner Class to execute more than one Scenario
4. Add the **Surefire plugin** configuration to the POM under build section
5. To execute using a **Maven Failsafe plugin** include the below configuration in the build section to the POM. Rename the runner class to RunCucumberIT. You can find further details [here](https://cucumber.io/docs/community/not-cucumber/#maven-execution-plugin).
6. Use the Maven **install** or a suitable command to **execute** the **POM**.

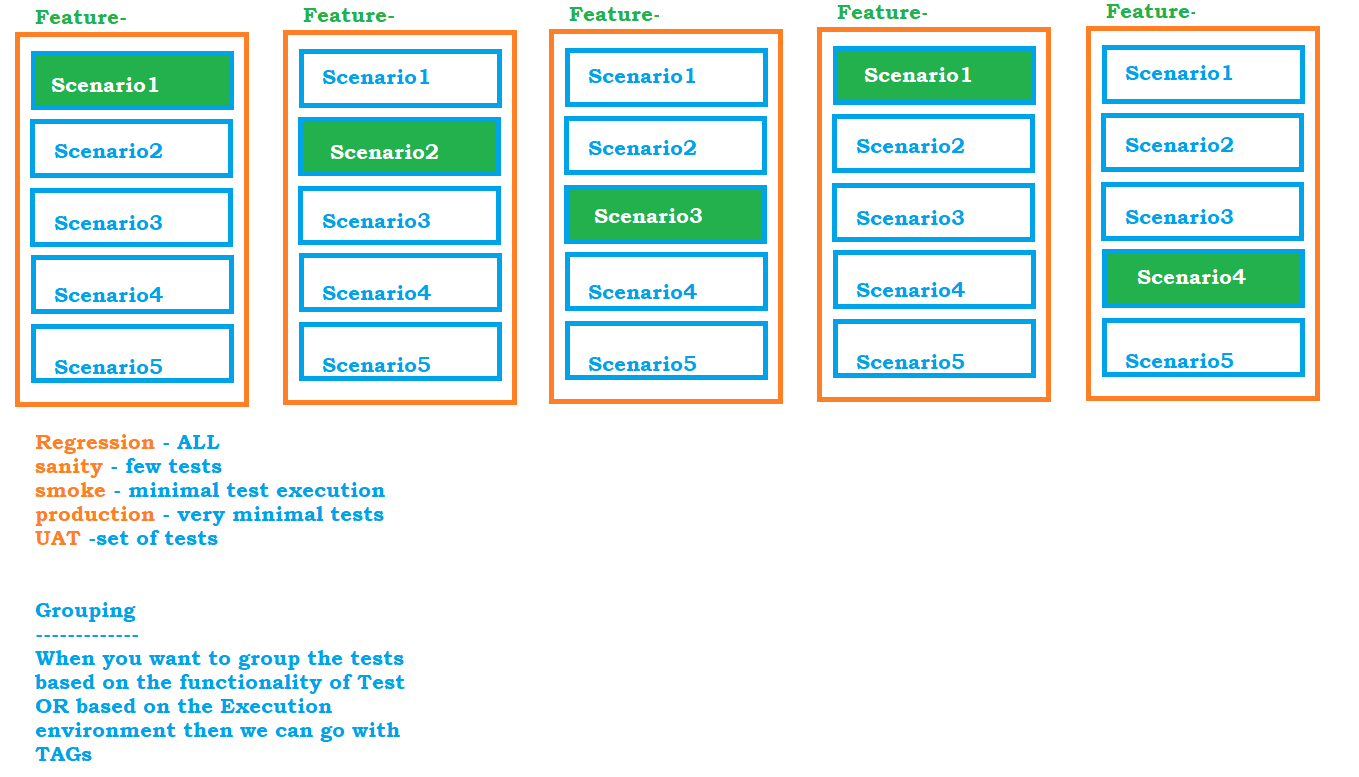
Hooks

If we want to execute any piece of code before executing or after executing a scenario then we can go with hooks,

There are different hooks supported by Cucumber

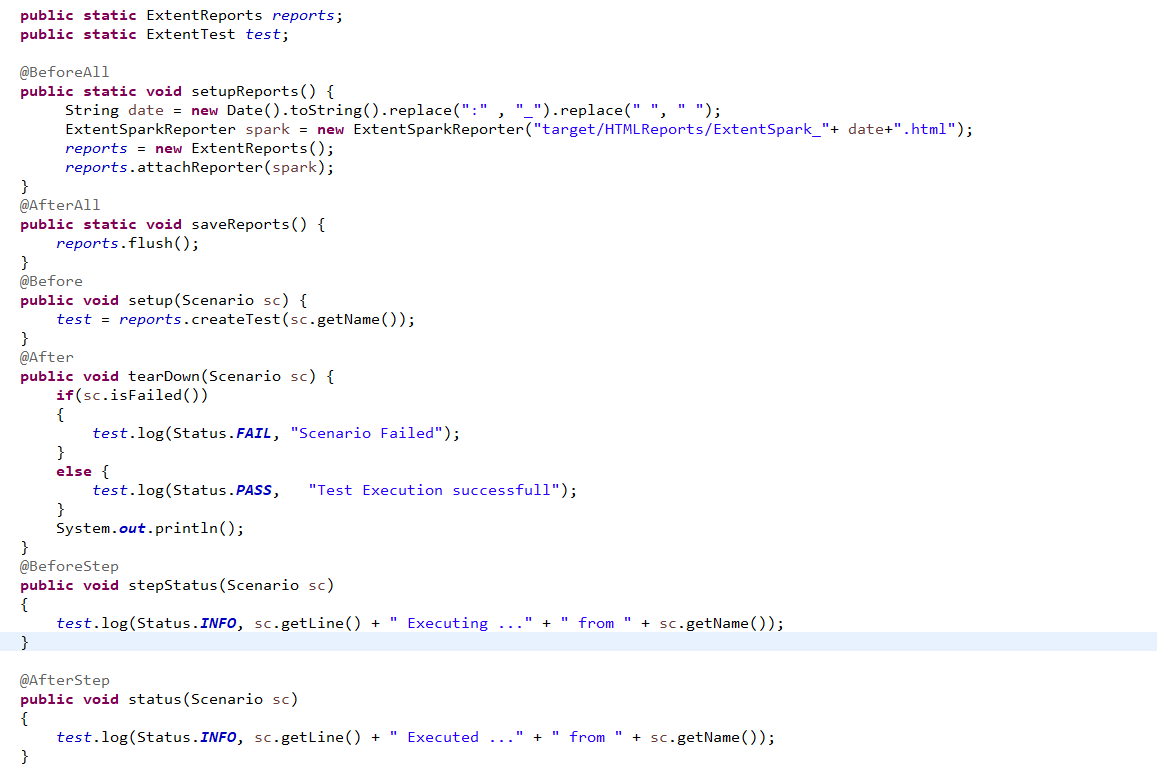
* *scenario hook*
* *step hook*
* *conditional hook*

Tags :



Extent Reports for Cucumber Project

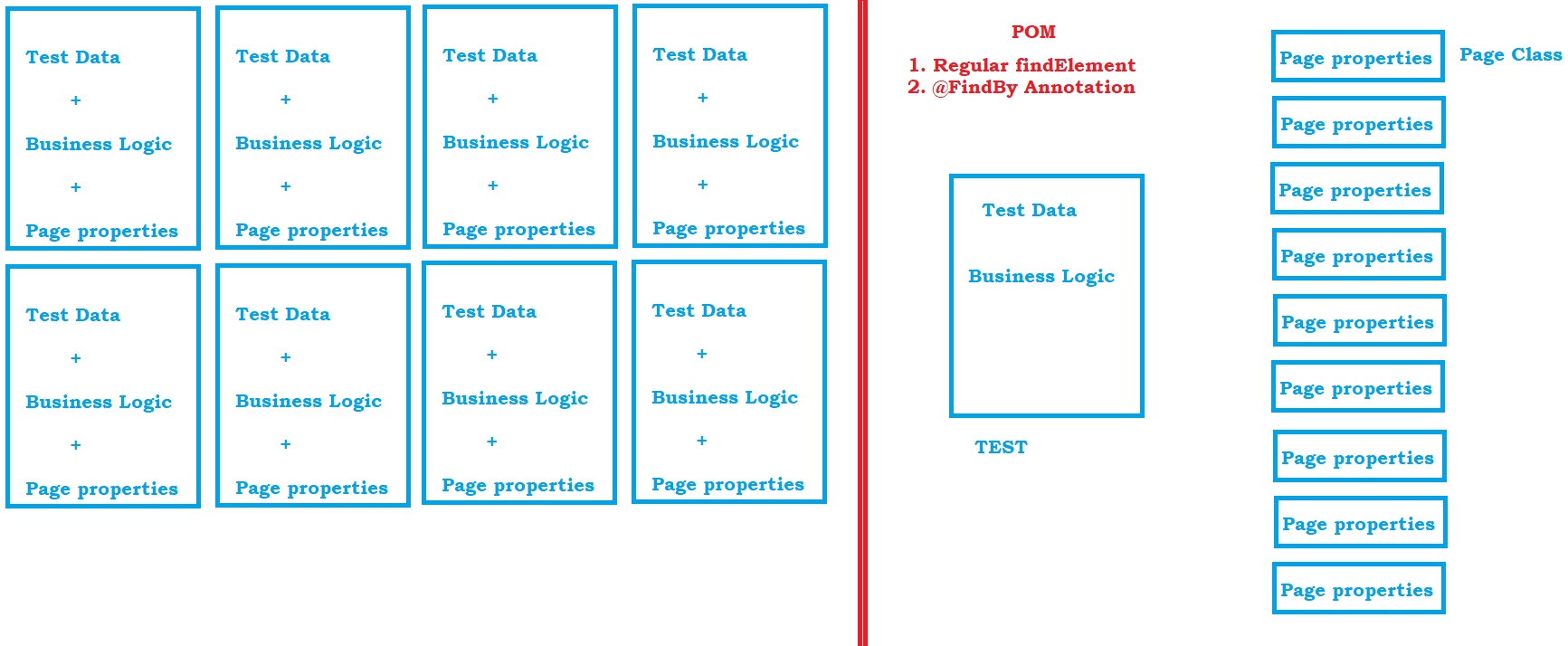
<https://www.extentreports.com/docs/versions/4/java/cucumber4.html>



Maven Cucumber Reports

* Update the maven cucumber plugin in pom.xml
* <https://github.com/damianszczepanik/maven-cucumber-reporting>
  + Update the project name – Give your project name
  + provide the html report location
  + provide the json file location
  + change the execution phase to test
  + Run Maven test to see the reports in the above specified path under target folder

POM



Capturing Screenshot

- TakeScreenshot

Batch file Execution

