**PROJECT—Root Element ===>modelVersion—4.0.0 ===>groupID===>artifactId===>version (artifactID version)**

**BUILD LIFE CYCLE: VALIDATE: validate the project is correct and all necessary information is available.**

**COMPILE: compile the source code of the project. TEST: compile all the Test from your project. PACKAGE: create jar/war file & share the file to somebody. VERIFY: run & checks on results of integration tests to ensure quality criteria are met. INSTALL: deploy JAR file in local repository. Before install it create a package .m2/repository. DEPLOY: deploy JAR file in remote location like JENKINS. We need to specify the target.**

**MAVEN PLUGIN: clean, compiler, deploy, failsafe, install, resources, site, surefire, verifier.**

**Run the Junit integration tests in an isolated class loader.**

**package testRunner;**

**import org.junit.runner.RunWith;**

**import io.cucumber.junit.Cucumber;**

**import io.cucumber.junit.CucumberOptions;**

**@RunWith(Cucumber.class)**

**@CucumberOptions(features, glue, plugin, snippets = CAMELCASE/UNDERSCORE**

**dryRun = true, (It doesn't going to run anything or none of the steps is executed. We**

**are just running to find out those scenarios don’t have step definitions. It also gives**

**missing snippets).**

**strict = true, (It is mandatory to add step definition for all the scenarios in all the**

**features files. Will fail execution if there are undefined or pending steps).**

**monochrome = true, (Display the console output in much readable way).**

**tags = {"@SmokeTest, @RegressionTest"}---execute all tests tagged as @SmokeTest OR**

**@RegressionTest**

**= {"@SmokeTest", "@RegressionTest"} --execute all tests tagged as @SmokeTest AND**

**@RegressionTest**

**(Managing the execution, which specific scenario will be run associate with the tag.**

**//name = {"Logo"} (which execute those scenarios which have 'logo' word inside the**

**scenarios)) Tags are Inherited by child element. Running a subset of scenarios**

**public class TestRunner { }**

**HOOKS: Cucumber hooks** are blocks of code that runs before or after each scenario.

**1. GLOBAL HOOKS will run once before any scenario is run or after all scenario have been run.**@Before All run before any scenario is run and @After All run after all scenarios have been executed.

**2. SCENARIO HOOKS runs for every scenario. Before** hooks run before the first step of each scenario. After hooks run after the last step of each scenario, even when the step result is failed, undefined, pending, or skipped.

**3. STEP HOOKS Step hooks invoked before and after a step. @BeforeStep @AfterStep**

**4. CONDITIONAL HOOKS with tags for more specifically execution.** Associate a Before or After hook with a tag expression. @After(“@browser and not @headless).

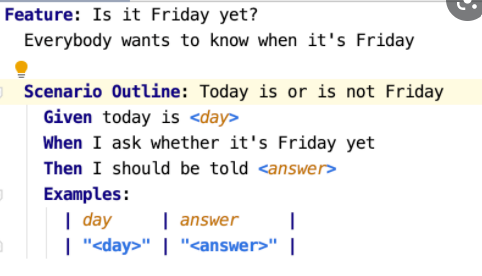
**5. ORDER PRIORITIZING HOOKS: @Before(order=0).**

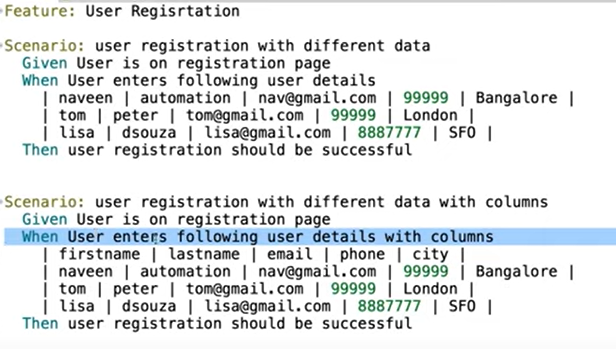
* Cucumber tests are written in plain-text English called Gherkin, so people with no or less technical skills can also write scenarios. It allows us to involve business stakeholders who can’t easily read a code. It **helps to bridge any understanding gap between business stakeholders and the developers.**
* High reusability of code in the tests. End-user experience is a priority.
* It can be integrated with Selenium and other testing frameworks like JUnit & TestNG

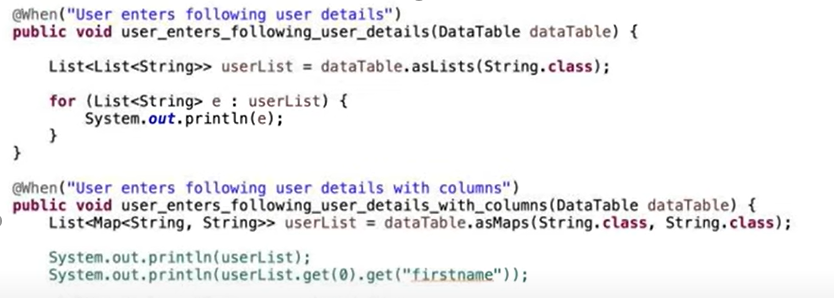
SCENARIO **includes all the possible circumstances of the feature and test scripts for** these circumstances. **SCENARIO OUTLINE: When particular scenario is run with more than one dataset in multiple combinations. It is like DATADRIVEN.**

Selenium can work independently of Cucumber. Cucumber depends on Selenium or Appium for step-definition implementation. Selenium is used for automated UI testing, while Cucumber is used for acceptance testing. DATATABLE: When defining Cucumber Secnarios, we often inject test data used by the rest of the scenario:

**Set the cucumber.execution.parallel.enabled configuration parameter to true e.g. in junit-platform.properties.**

****

****

****