Cucumber Automated Tests - Sample Setup

The Automation framework has been setup using:

- Cucumber JVM - https://github.com/cucumber/cucumber-jvm
- Selenium Webdriver 3 - https://www.seleniumhq.org/projects/webdriver/
- Rest Assured - https://github.com/rest-assured/rest-assured/wiki
- Java 8 - https://www.oracle.com/technetwork/java/javase/overview/java8-2100321.html
- Maven - https://maven.apache.org/

Setup Eclipse

Eclipse and a java development kit are required - How to install this is beyond the scope of this article but is detailed in the article referenced above

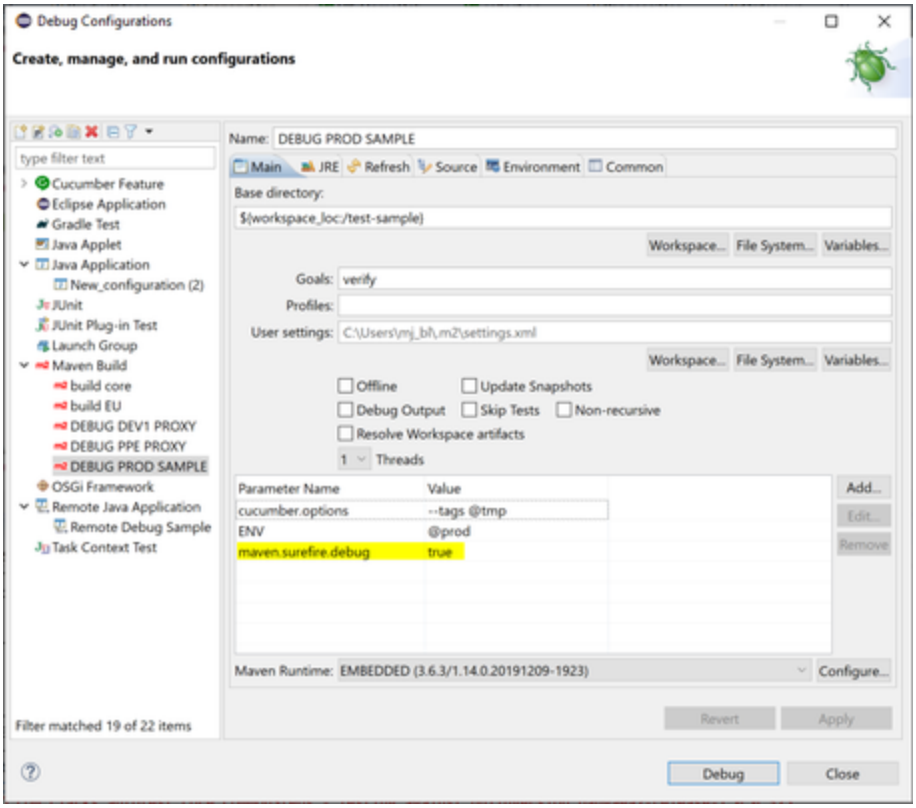
The project should be imported into eclipse via 'Import Existing projects into Workspace'

Eclipse Installation Notes:

1. Download Eclipse
    a. https://www.eclipse.org/downloads/
2. Download and installed: Eclipse IDE for Java Developers
3. After Install and Eclipse running install the Cucumber extension (NB: As of Dec 2019 you will need the latest snapshot version to support the new cucumberjvm4 version)
    a. Help Install New Software Add
        i. Name: cucumber snapshot
        ii. Location: https://cucumber.github.io/cucumber-eclipse-update-site-snapshot
        iii. Add Cucumber Eclipse Plugin, as of 03\01\2020 the version listed is: 1.0.0.201911262209
4. Adding the projects
    a. Close the Eclipse Welcome page so you have a blank workspace
    b. Right click in the left hand panel and select Import
    c. Under General select the 'Existing Projects into Workspace' option and select next
    d. Under Select root directory, select browser and navigate to where you checked out your project and select open
    e. Keep other defaults and continue

Eclipse

- Run -> Run Configurations
- Maven Build -> New
- Set Base Directory as the Workspace root, eg test-sample
- Set goals as 'test', eg:
- set ENV as '@prod'
- Set cucumber.options as '–tags <thetagtorun>'



- Click the Run Maven Build button (green play button)

Setting Up Browsers

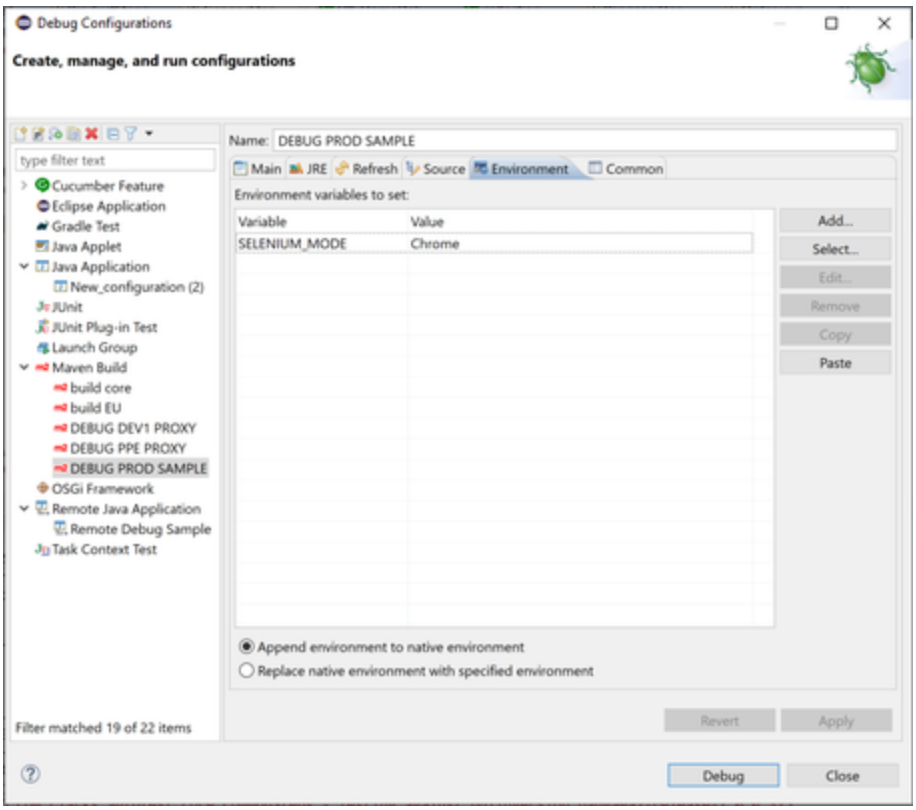
Running on a Local Browser

To run tests locally you will need:

- Chrome or Firefox

you then choose the local browser by setting the **environment variable on the 'Environment' Tab**:

SELENIUM_MODE = Chrome
OR
SELENIUM_MODE = Firefox



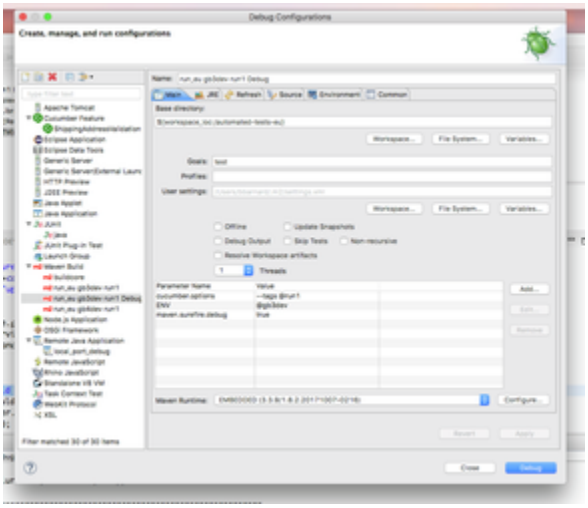
HINT: Setup a run target with the options --tags @tmp1 and add this tag to any scenario you wish to run locally, this avoids having to change the run target each time

Debugging an Eclipse run using Maven

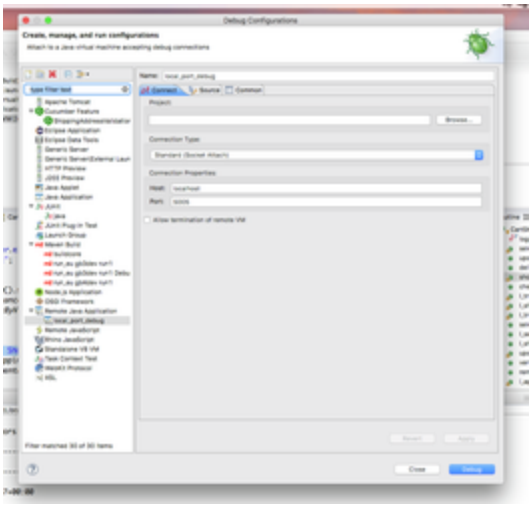
You can Debug a test in Eclipse by setting a Breakpoint at a line of code.

But when running the Debug using a Maven - > Debug Configuration the Maven is run in a separate thread and ignores the Eclipse Debug Breakpoints, ie it doesn't stop when you want it to.

To get round this define a Debug config setting the parameter: maven.surefire.debug = true



You also need to configure a Remote Java Application Config to connect to port 5005, the port the Maven surefire uses when it runs:



Run the Maven Debug Configuration and when that is running start the Remote Java Application. That will then connect to port 5005 and you can debug through the test as a normal Eclipse debug.

There is a Help page in Maven on this at:

maven-surefire-plugin/examples/debugging