User Manual

Instance-Based
Cucumber Testing

Matthew Walker

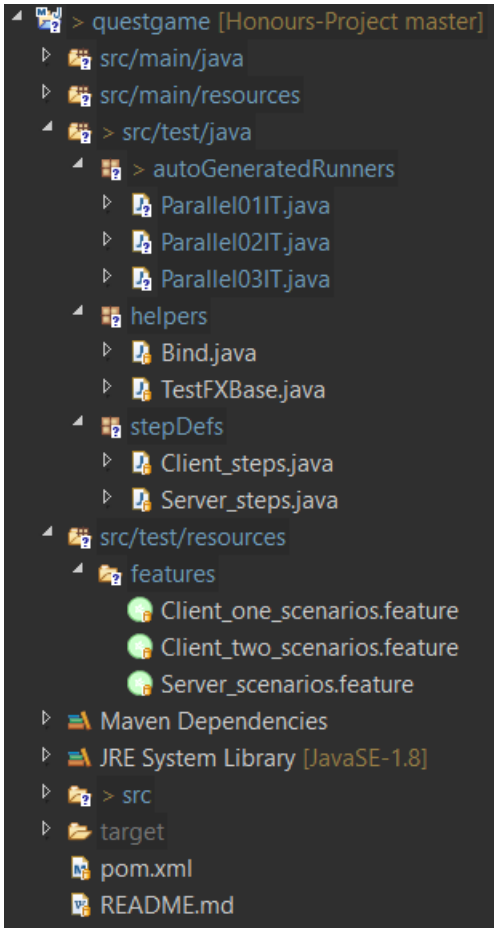
Table of Contents

Importing into Eclipse

To import the project into Eclipse:

1. Open Eclipse
2. Click on File -> Open Projects from File System…
3. Click on Directory in the window that pops up
4. Navigate to the project folder and click on the 'Honours-Project' folder
5. Click Finish
6. After the Maven Dependencies have downloaded,
   right click on the project -> Maven -> Update Project… and click OK

Package Structure



As seen in this image, there are two source code folders: main and test. The main folder contains all the code for running the game, and the test folder contains all the code used to perform the instance-based Cucumber testing. The first folder, 'autoGeneratedRunners', contains the runners needed to run the Cucumber threads in parallel, which were generated automatically. The 'helpers' folder contains two helper classes needed by the step definition classes. The 'Bind' file is the class created to take a server object upon creation, and then allow the ability to fetch actual instances from the server object as it is running. The 'TestFXBase' file is need by the client step definitions file, as it is what allows for the access of nodes within the client windows, making automation of the clients possible. The 'stepDefs' folder contains the client and server step definitions, which correspond to the scenarios outlined in the feature files. Lastly, the 'features' folder, which is also stored within the 'resources' folder, contains the Cucumber feature files, each with unique scenarios used to run together as a test of the entire IUT.

***NOTE: The runners will need to be automatically re-generated after importing this project onto a different machine, as the file directories will be different. This will be outline in the 'Auto-Generating Parallel Runners' section, but it is possible to just replace the existing file directories written in the runners, with the directories used by your machine.

Java 8 Dependency

As seen in the package structure, this project is dependent on java version 8. This is due to dependencies of the IUT, and the maven-compiler need to perform these tests. Other versions of java were tried, but this version appears to be the only one which works for this implementation.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.0</version>
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <fork>true</fork>
        <executable>C:\Program Files\Java\jdk1.8.0_201\bin\javac</executable>
    </configuration>
</plugin>
```

In the maven pom file, the maven compiler plugin can be found in the plugin section, as seen in the image above. To allow it to use version 8 of the JDK, it must be setup like this, with the <executable> pointing to the location of the JDK on your machine (The path seen here is what it looks like on my machine, but it may be different for you). In addition to this, the project must also be using version 8 of the JRE, as seen in the package structure image. To do this, right click on the project in eclipse, select properties, click on the java compiler tab, and set the compliance to version 8.

Auto-Generating Runners

Upon importing the project into eclipse, it will already contain runners, however they will not function properly unless the directories referenced within them are correct. When importing the project onto another machine, these runners will either need to be re-generated, or have the directories manually changed to point to the locations on your machine.

Parallel01IT.java

```
1  package autoGeneratedRunners;
2
3  import org.junit.runner.RunWith;
7
8  @RunWith(Cucumber.class)
9  @CucumberOptions(
10         strict = true,
11         features = {"C:/Users/Thinkpad/Documents/GitHub/Honours-Project/Honours-Project/src/test/resources/features/Server_scenarios.feature"},
12         plugin = {"json:C:/Users/Thinkpad/Documents/GitHub/Honours-Project/Honours-Project/target/cucumber-parallel/1.json"},
13         monochrome = false,
14         tags = {},
15         glue = {"stepDefs"})
16 public class Parallel01IT {
17 }
18
```

As seen in the image above, the features attribute points to the specific Cucumber feature file that this runner will use. The plugin attribute is the json file containing the outcome of the feature, that will be created after it is ran, placed in the that directory. And lastly, the glue attribute points to the folder containing the glue code corresponding to the feature file.

```
<plugin>
    <groupId>com.github.temyers</groupId>
    <artifactId>cucumber-jvm-parallel-plugin</artifactId>
    <version>5.0.0</version>
    <executions>
        <execution>
            <id>generateRunners</id>
            <phase>generate-test-sources</phase>
            <goals>
                <goal>generateRunners</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <glue>stepDefs</glue>
        <outputDirectory>src/test/java/autoGeneratedRunners</outputDirectory>
        <featuresDirectory>src/test/resources/features</featuresDirectory>
        <cucumberOutputDir>target/cucumber-parallel</cucumberOutputDir>
        <format>json</format>
        <parallelScheme>FEATURE</parallelScheme>
    </configuration>
</plugin>
```

This is the plugin which generates the runners, seen above. Note that the <outputDirectory> points to the autoGeneratedRunners folder, and the <featuresDirectory> points to the features folder.

To run this plugin and generate the runners:

1. Right click on the maven pom file
2. Click on Run As -> Maven Build…
3. In the 'Name' space, enter: *Generate Runners*
4. In the 'Goals' space, enter this: *com.github.temyers:cucumber-jvm-parallel-plugin:5.0.0:generateRunners*
5. Then press the Run button at the bottom
6. Wait for the build to finish
7. Refresh the 'autoGeneratedRunners' folder
8. The runners have been created

***NOTE: It may occur that the runners will not contain their proper package declaration (>package autoGeneratedRunners), in which case it must be added to each of the runner classes.

Running Parallel Cucumber Threads

To run in parallel each of the Cucumber feature files, or more specifically the runners for each of the feature files, maven surefire must be used. For this project a different version on maven surefire, known as maven failsafe was used, due to it handling the failure of multiple threads in a more elegant fashion. This plugin in shown in the image to the right, with the configuration being the only part that must be customized. The <forkCount> corresponds to the number of threads, or in this case the number of Cucumber runners that will be running in parallel (in this case 3). The <reuseForks> is to be used if threads will be starting and stopping at different times, however in this case it can be set to false. And the <includes> section identifies the name structure of the runners that will be used to run in parallel with each other (in this case, each auto-generated runner has the same naming structure, ending in 'IT.class').

To then run the Cucumber runners in Parallel with maven failsafe:

1. Ensure the runners have been generated
2. Right click on the maven pom file
3. Click Run As -> Maven Test (Compiles everything)
4. Click Run As -> Maven Build…
5. In the 'Name' space, enter: *Parallel Test*
6. In the 'Goals' space, enter this:
   org.apache.maven.plugins:maven-failsafe-plugin:3.0.0-M3:integration-test
7. Then press the Run button at the bottom

***NOTE: After doing this once, to repeat it more easily: right click on the pom file, select configurations, and the *Parallel Test* build will be shown in a list, to then be run again. Also always remember to run a maven test before each parallel test, since it must first compile any and all changes to the source code.

Also, to fix the log4j failure, add a text file on your C drive call 'log4j-application.log'

Package Versions Used

The easiest way to see all the package versions used is to open the maven pom file with a generic text editor in Eclipse, as all the dependencies and plugins are shown there with identification about what each of them are, but here I will also list everything used:

Package Name	Package Version	Package URL
junit-jupiter-api	5.4.0-M1	https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api/5.4.0-M1
junit-jupiter-params	5.4.0-M1	https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-params/5.4.0-M1
junit-jupiter-engine	5.4.0-M1	https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-engine/5.4.0-M1
junit-vintage-engine	5.4.0-M1	https://mvnrepository.com/artifact/org.junit.vintage/junit-vintage-engine/5.4.0-M1
junit-platform-launcher	5.4.0-M1	https://mvnrepository.com/artifact/org.junit.platform/junit-platform-launcher/1.5.0-M1
junit-platform-runner	5.4.0-M1	https://mvnrepository.com/artifact/org.junit.platform/junit-platform-runner/1.5.0-M1
openjfx-monocole	1.8.0_20	https://mvnrepository.com/artifact/org.testfx/openjfx-monocle/1.8.0_20
hamcrest-all	1.3	https://mvnrepository.com/artifact/org.hamcrest/hamcrest-all/1.3
testFx	3.1.2	https://mvnrepository.com/artifact/org.loadui/testFx/3.1.2
testFx-core	4.0.15-alpha	https://mvnrepository.com/artifact/org.testfx/testfx-core/4.0.15-alpha
testFx-junit	4.0.15-alpha	https://mvnrepository.com/artifact/org.testfx/testfx-junit/4.0.15-alpha
log4j	1.2.17	https://mvnrepository.com/artifact/log4j/log4j/1.2.17
cucumber-core	4.2.0	https://mvnrepository.com/artifact/io.cucumber/cucumber-core/4.2.0
cucumber-java	4.2.0	https://mvnrepository.com/artifact/io.cucumber/cucumber-java/4.2.0
cucumber-jvm	4.2.0	https://mvnrepository.com/artifact/io.cucumber/cucumber-jvm/4.2.0

cucumber-junit	4.2.0	https://mvnrepository.com/artifact/io.cucumber/cucumber-junit/4.2.0
cucumber-jvm-deps	1.0.5	https://mvnrepository.com/artifact/info.cukes/cucumber-jvm-deps/1.0.5
gherkin	5.1.0	https://mvnrepository.com/artifact/io.cucumber/gherkin/5.1.0
cucumber-jvm-parallel-plugin	5.0.0	https://mvnrepository.com/artifact/com.github.temyers/cucumber-jvm-parallel-plugin/5.0.0
maven-compiler-plugin	3.8.0	https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-compiler-plugin/3.8.0
maven-failsafe-plugin	3.0.0-M3	https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-failsafe-plugin/3.0.0-M3