Squash TF Cucumber Java Runner Documentation

squashtest

Apr 02, 2020

Contents

CHAPTER 1

Overview

The **Squash Test Factory (Squash TF) Cucumber Java Runner** allows you to run your tests written in Gherkin language with the TF ecosystem. The Java implementation of your Gherkin scripts must just be compatible with the cucumber framework.

The runner (Maven based) enables to run your scripts (a selection of feature files) from an IDE, command line, TF Server but also from **Squash T**est **M**anagement (**Squash TM**).

As with TF framework, different reporting are available. But with Cucumber Java Runner you can also have an HTML report specific to Gherkin scripts with among other informations, the result of the execution at each level (suite, feature, scenario, dataset, step), as well as the step by step feature display with the matching java methods executed.

In the case where the execution launcher is Squash Test Management (Squash TM), and simply by configuring TF, the runtime test status and the final report can be automatically forward to Squash TM.

The runner allows either to run the features either to check if the features are runnable (dryrun) , ie if the automation work was done which means that a java implementation of each step exists, is not empty and not reduce to throw an 'cucumber.api.PendingException'.

CHAPTER 2

Contents

The purpose of this section is to describe how to create a test project from scratch or how to configure an existing Gherkin/Java project so it can be launched by the TF Runner. This includes of course the case where feature files have been written with squashTM and extract from a GIT repository.

Note: Please, check your Maven and JDK installations first as explains here

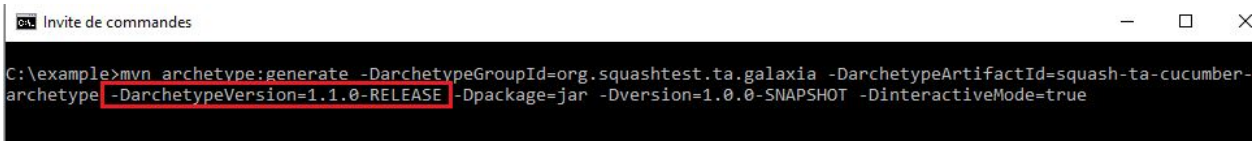
2.1 Java project

2.1.1 > From scratch

In the case you are starting from scratch, it is recommended to create the test project with the squash-ta-cucumber-archetype. Before typing the line below in a terminal, please update the version of the archetype if necessary.

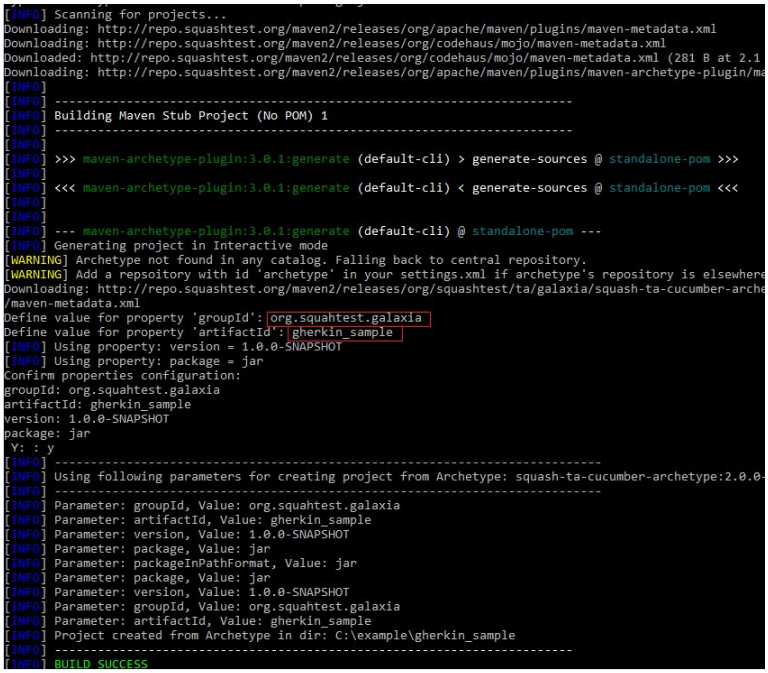
```
mvn archetype:generate -DarchetypeGroupId=org.squashtest.ta.galaxia -
↪DarchetypeArtifactId=squash-ta-cucumber-archetype -DarchetypeVersion=1.1.0-RELEASE -
↪Dpackage=jar -Dversion=1.0.0-SNAPSHOT -DinteractiveMode=true
```

The version of the archetype is 1.1.0-RELEASE in this sample,



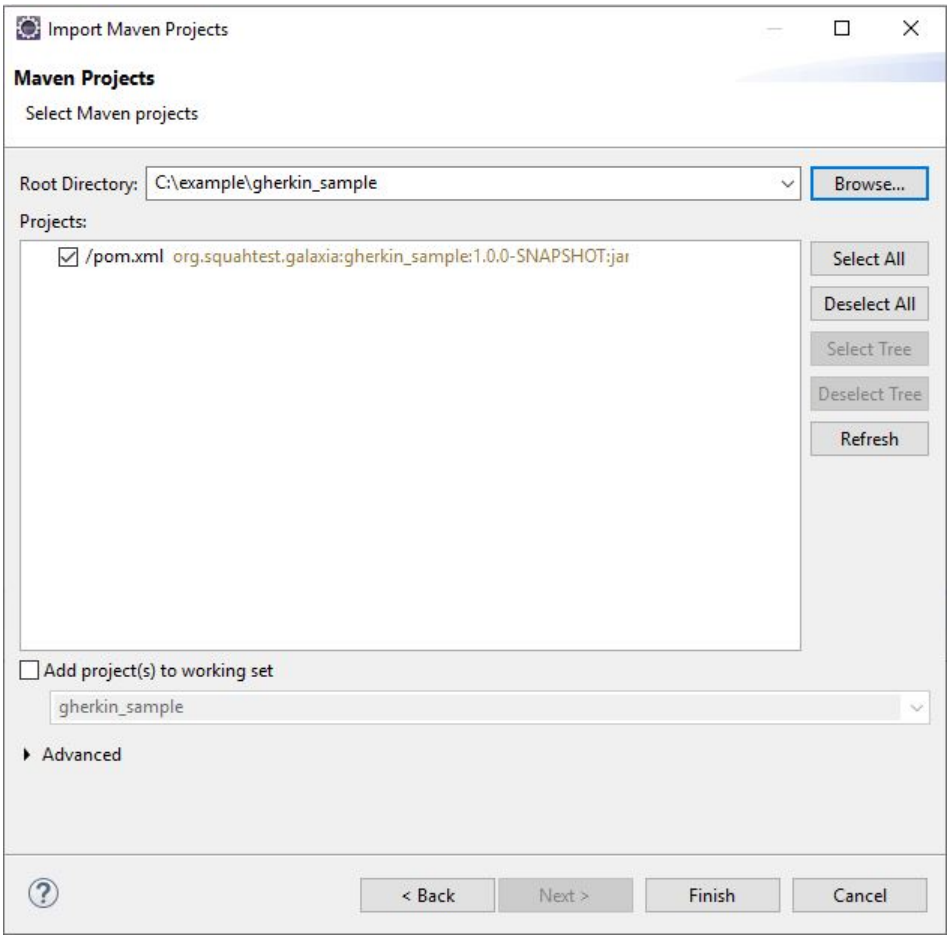
and you can find all available versions in our repository: http://repo.squashtest.org/maven2/release/archetype-catalog.xml

After typing the command line on top and supplying a groupId and an artifactId ('org.squashtest.galaxia' and 'gherkin_sample' in the example above), you should have 'BUILD SUCCESS'.

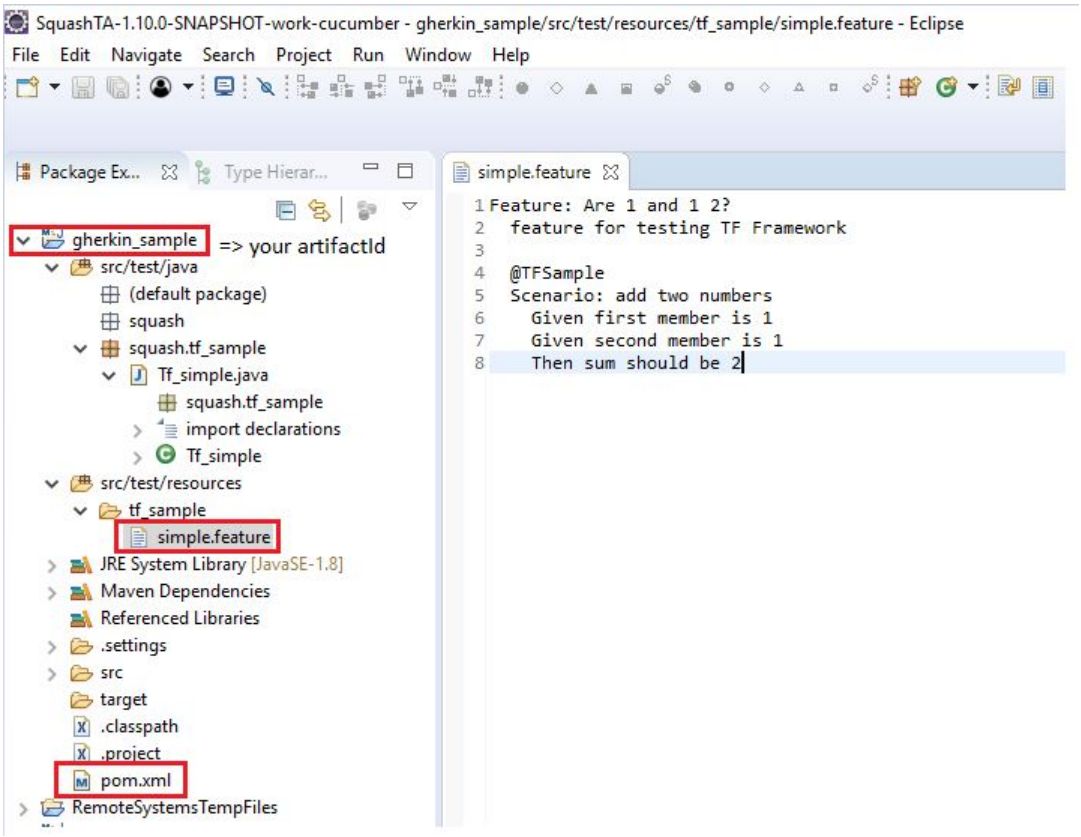


If not, please read the note about *environment-configuration*. The most common causes of errors are: Maven is not installed or not in path, the access to our repository are not declared in the Maven's setting.xml file, the TF archetype version does not exist in our repository.

Now you just have to open the newly created project in an IDE and start writing your tests. With Eclipse:



Note that an example of a feature with its java code is provided with the archetype.



See *here* for how to run it.

See *here* for how to implement your own tests.

See *here* for how to change TF runner's default options.

2.1.2 > Existing project

In the case you already have a test project gherkin/cucumber and want to run it with TF, add or modify these 4 blocks to your pom.xml file (located at the root of your project) as shown hereafter.

Add the property for runner's version (and the sources's encoding if necessary) like this

```
<properties>
        <!-- Squash-TF-cucumber runner version used by the project -->
        <ta.cucumber.runner.version>1.1.0-RELEASE</ta.cucumber.runner.version>
        <!-- optional source encoding -->
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

Add this <plugin> block above in your build/plugins block.

```
<build>
        <plugins>
                <!-- Configuration of the Squash TA framework used by the project -->
                <plugin>
                        <groupId>org.squashtest.ta.galaxia</groupId>
                        <artifactId>squash-tf-gherkin-maven-plugin</artifactId>
                        <version>${ta.cucumber.runner.version}</version>
                        <configuration>
                                <featuresList>${ta.feature}</featuresList>
                                <squashRoot>squash</squashRoot>
                                <!-- DryRunOptions for dryrun goal only -->
                                <additionnalDryRunChecks>true</
↪additionnalDryRunChecks>

                                <!-- Define exporters -->
                                <gkexporters>
                                        <exporter
                                                implementation="org.squashtest.ta.
↪commons.exporter.surefire.SurefireSuiteResultExporter">
                                                <jenkinsAttachmentMode>${ta.jenkins.
↪attachment.mode}</jenkinsAttachmentMode>
                                        </exporter>

                                        <exporter
                                                implementation="org.squashtest.ta.
↪gherkin.exporter.HtmlGherkinSuiteResultExporter" />
```

(continues on next page)

(continued from previous page)

```
                                </gkexporters>

                                <!-- Define configurers -->
                                <gkconfigurers>
                                        <configurer implementation="org.squashtest.ta.
↪maven.TmCallBack">
                                                <!-- <tmCallBack> -->
                                                <endpointURL>${status.update.events.
↪url}</endpointURL>
                                                <executionExternalId>${squash.ta.
↪external.id}</executionExternalId>
                                                <jobName>${jobname}</jobName>
                                                <hostName>${hostname}</hostName>
                                                <endpointLoginConfFile>${squash.ta.
↪conf.file}</endpointLoginConfFile>
                                                <reportBaseUrl>${ta.tmcallback.
↪reportbaseurl}</reportBaseUrl>
                                                <jobExecutionId>${ta.tmcallback.
↪jobexecutionid}</jobExecutionId>
                                                <reportName>${ta.tmcallback.
↪reportname}</reportName>
                                                <!-- </tmCallBack> -->
                                        </configurer>
                                </gkconfigurers>
                        </configuration>

                        <executions>
                                <execution>
                                        <goals>
                                        <!-- to execute feature files -->
                                                <goal>run</goal>
                                                <!-- to check feature files are␣
↪runable (DryRun) -->
                                                <goal>dryrun</goal>
                                        </goals>
                                </execution>
                        </executions>
                </plugin>
        </plugins>
</build>
```

Add the this block for the path to our repository:

```
<!-- Squash TA maven plugin repository -->
        <pluginRepositories>
                <pluginRepository>
                <id>org.squashtest.plugins.release</id>
                <name>squashtest.org</name>
                <url>http://repo.squashtest.org/maven2/releases</url>
                <snapshots>
                <enabled>false</enabled>
                </snapshots>
                <releases>
                <enabled>true</enabled>
                </releases>
                </pluginRepository>
```

(continues on next page)

(continued from previous page)

```
        </pluginRepositories>
```

Add these dependencies:

```
<dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>4.0.0</version>
</dependency>

<dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>4.0.0</version>
</dependency>

<dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
</dependency>
```

Except for the TF-TM link parameters and eventually the runner's version, you can use all default settings. See *Runner's options* to modify them.

2.2 Scripting / Coding

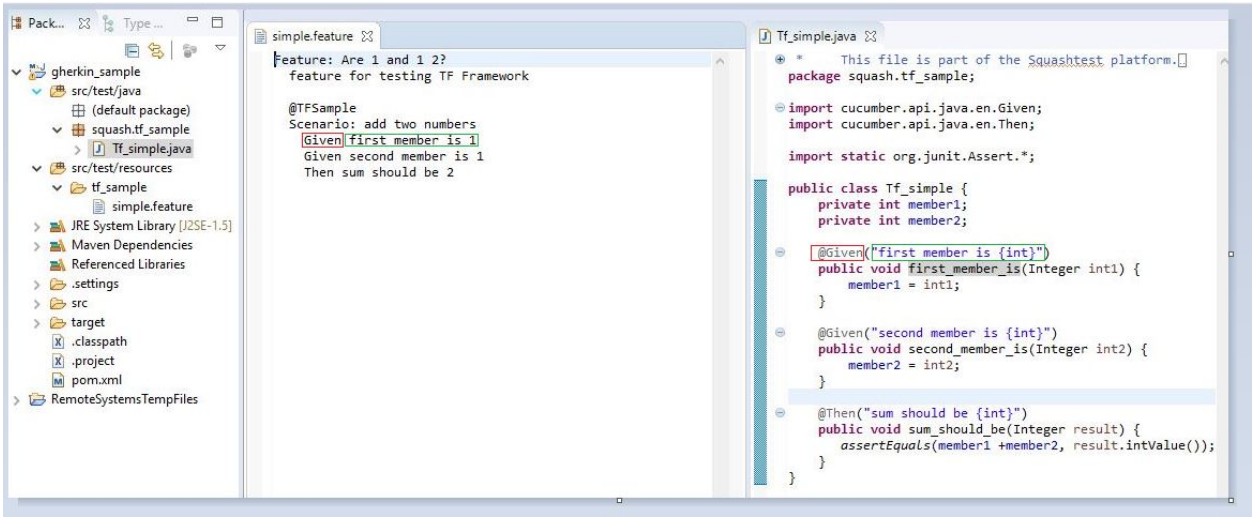
2.2.1 > Overview

Note: Your project should be based on the Cucumber framework. Your tests written in Gherkin language should be implemented in java language.

We recommend that you structure your project as follows:

- the feature files should be in src/test/resources (or any subdirectory)
- the Java implementation in src/test/java

Below the very simple example of a feature and its implementation provided with the TF archetype.



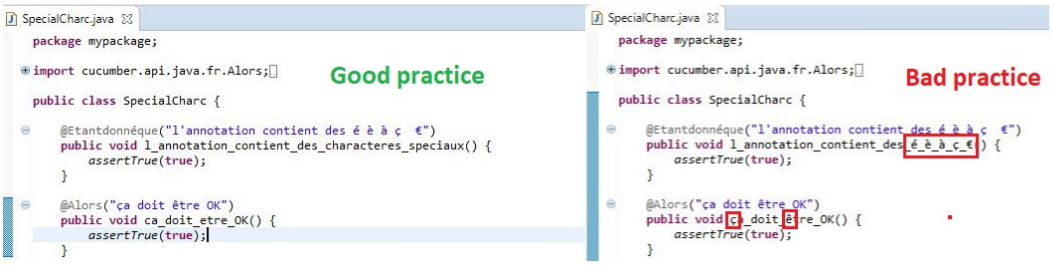
See:

- https://docs.cucumber.io/gherkin/reference/ for *Gherkin Language Reference*
- https://docs.cucumber.io/cucumber/step-definitions/ for how to write your *Step Definitions*, ie the java methods that link them to steps of your Gherkin script.

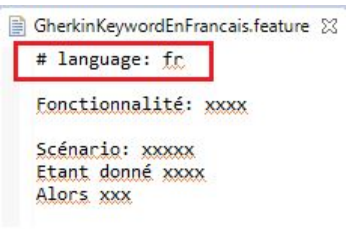
2.2.2 > Advices

About the encoding and the language used:

- If you use the TF artifact to create your project, the encoding is set to UTF-8 (java/feature). Of course, you can modify it through the property 'project.build.sourceEncoding' in the POM.xml file.
- If you declare an encoding in your feature files (#encoding: utf-8 by example), make sure there is no inconsistency with what is declared in your POM.xml file.
- If your feature files are written with UTF-8 (or any special encoding), a good practice is to name your java methods only with ASCII characters.

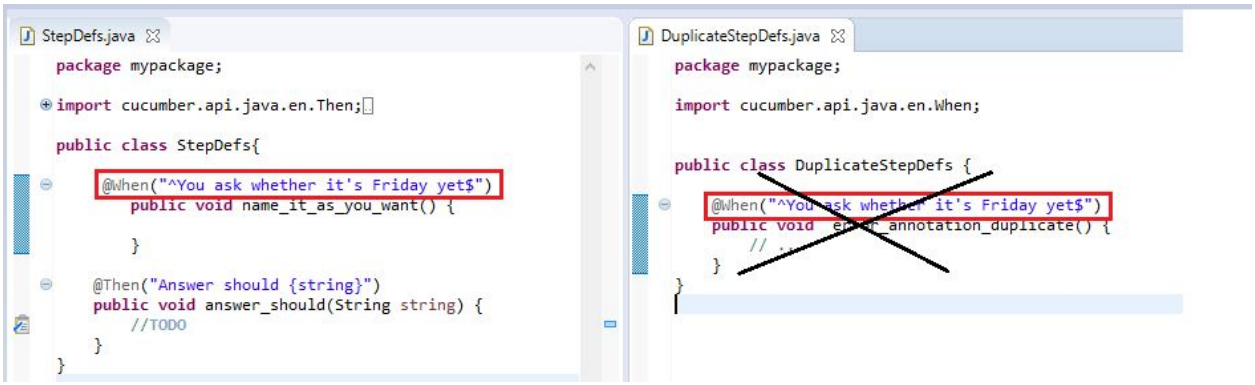


- You can of course use keywords in a given language by declaring it in your feature files (check only the consistency of implied encoding)



About the glue:

All the classes of the project will be used to find the methods matching with a step of the features. There should not be multiple implementations for the same step. That is, each annotation that identifies a step must be on a single java method (even if the method with the duplicate annotation is in a different class or classpath).



2.3 Execute (run, dryrun)

Note: Please, if you want to start feature files execution from TM, refer to the Squash TM corresponding documentation. If you want to start execution from Squash TF execution Server, use the template involved.

2.3.1 > Command line

To start the execution of all the tests (feature files) of the project from a terminal, go to the directory of your project (where the POM.xml file is located) and use one the following commands by replacing /path/to/project with the valid path:

- for the goal “run”:

```
mvn squash-tf-gherkin:run -Dta.feature=/path/to/project/src/test/resources
```

- for the goal “dryrun”:

```
mvn squash-tf-gherkin:dryrun -Dta.feature=/path/to/project/src/test/resources
```

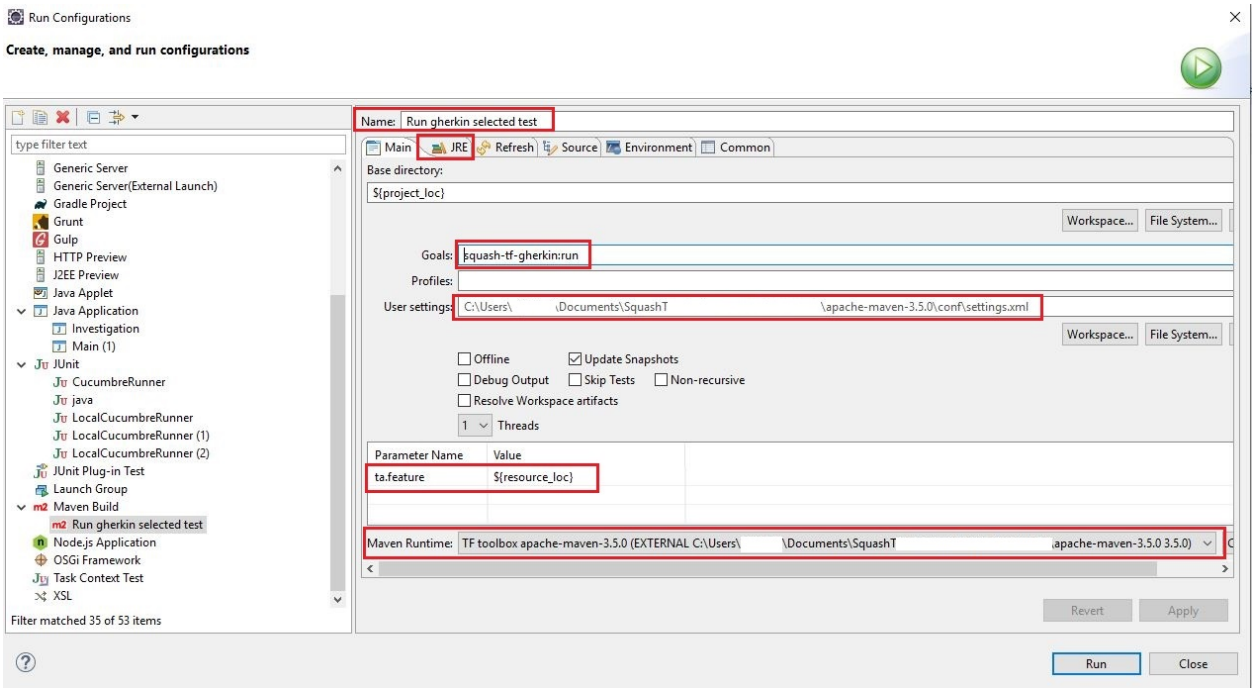
With our sample and for the goal dryrun

```
C:\example\gherkin_sample>mvn squash-tf-gherkin:dryrun -Dta.feature=C:\example\gherkin_sample\src\test\resources
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building gherkin_sample 1.0.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- squash-tf-gherkin-maven-plugin:1.0.0-RELEASE:dryrun (default-cli) @ gherkin_sample ---
[INFO] Launching Squash TA C3PO edition.
[INFO] SquashTAGherkinTestRunableMojo (Maven Mojo to check if feature filesare runable)starting ...  with additionnalDryRunChecks to: tru
[INFO] Listing sqhashTA engine component packages
[INFO] Loading plugin configuration for: org.squashtest.ta.plugin.cucumber
[INFO] Loading plugin configuration for: org.squashtest.ta.plugin.local.process
[INFO] Loading plugin configuration for: org.squashtest.ta.enginecore
[INFO] Loading plugin configuration for: org.squashtest.ta.plugin.commons-component
[INFO] Loading XML bean definitions from Byte array resource [Computed squashTA engine configuration]
[INFO] Refreshing org.springframework.context.support.GenericXmlApplicationContext@430fa4ef: startup date [Fri Apr 12 15:55:23 CEST 2019]
oot of context hierarchy
[INFO] JSR-330 'javax.inject.Inject' annotation found and supported for autowiring
[WARNING] The endpoint URL is set to its default value: "file://dev/null", so the call back is not activated
[INFO] Beginning execution of ecosystem FeatureList
[INFO] Beginning execution of test Eco.setup
[INFO] Beginning execution of test tf_sample.simple.feature
[INFO] Exporting results
[INFO] SquashTA Gherkin Mojo: class of exporter processing: org.squashtest.ta.commons.exporter.surefire.SurefireSuiteResultExporter
[INFO] SquashTA Gherkin Mojo: class of exporter processing: org.squashtest.ta.gherkin.exporter.HtmlGherkinSuiteResultExporter
[INFO] Cleaning resources
[INFO] Squash TA : build complete.
[INFO] All the files from C:\Users\CJUILL~1\AppData\Local\Temp\Squash_TA were properly deleted.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.284 s
[INFO] Finished at: 2019-04-12T15:55:24+02:00
[INFO] Final Memory: 25M/271M
[INFO] ------------------------------------------------------------------------
```

See *Selecting tests* for how to select the tests to execute or *Expected results*

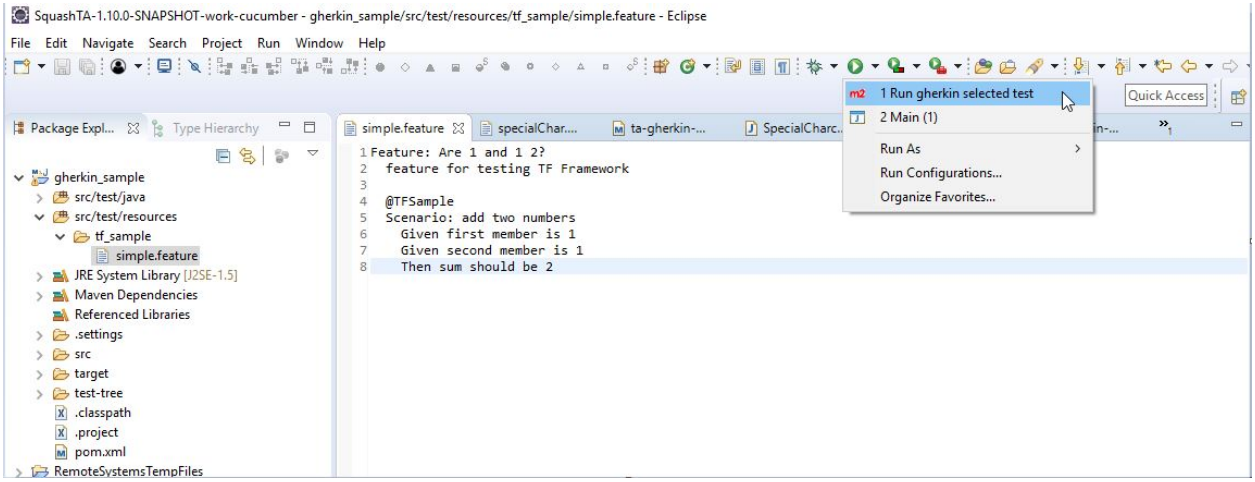
2.3.2 > Eclipse launcher

If you use Eclipse, you can create specific launch configurations (one to run and one to dryrun). Select the menu “run” then “run Configuration…” and create a launch like this for a run goal:



Check your Maven and JDK (JRE tab) configurations.

Then, select a feature file or a subdirectory of src/test/resources and click run / Run gherkin selected test



2.4 Runner's options (POM)

Some options or parameters can be configured with the pom.xml file of your test project.

2.4.1 > Runner's version

```
<properties>
        <!-- Squash-TF-cucumber runner version used by the project -->
        <ta.cucumber.runner.version>1.1.0-RELEASE</ta.cucumber.runner.version>
        ...
</properties>
```

See http://repo.squashtest.org/maven2/release/org/squashtest/ta/galaxia/squash-ta-cucumber/ for all available versions.

2.4.2 > Encoding

```
<properties>
        ...
        <!-- optional: source encoding -->
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

Before changing the encoding, please read *About encoding*

2.4.3 > Selecting tests

```
<featuresList>${ta.feature}</featuresList>
```

You can choose the features to be processed by changing the ta.feature parameter above for both goals. The supported values are:

- /path/to/project/src/test/resources/**../subfolder** => for all feature files in the given subfolder of src/test/resources.
- /path/to/project/src/test/resources/**../myTest.feature** => for a single feature.
- {file:path/to/JSONfile/MyFeatureList.json} => for a file containing a list of feature files. The contents of the json file should look like this example:

```
{
  "test": [
    {
      "id": "",
      "script": "pathRelativeToSquashRoot/TEST1.feature",
      "param": ""
    },
    {
      "id": "",
      "script": "pathRelativeToSquashRoot/TEST2.feature",
      "param": ""
    }
  ]
}
```

where the given path for the feature should be relative to the src/test/resources/squash folder. You can change this relative path by changing the <squashRoot> value in the project POM file.

- a json containing a list of feature files like:

```
{
  "test": [
    {
      "script": "pathRelativeToSrcTestResources/test1.feature"
    },
    {
      "script": "pathRelativeToSrcTestResources/test2.feature"
    }
  ]
}
```

Warning: the Gherkin files should have a ".feature" extension otherwise they will be ignored.

2.4.4 > TM squashRoot

```
<squashRoot>squash</squashRoot>
```

This parameter specifies a subpath for the feature files when the tests suite execution is requested by SquashTM. If you do not start running from Squash TM, you can change the value of squashRoot or disable it by commenting the line. But if you start running from TM, do not change this parameter otherwise all the feature files will be "NOT_FOUND".

The value of this tag point out the root folder where TF runner should look for the gherkin files to run. It is taken into account only if the parameter "ta.feature" is a JSON file (ie ignored otherwise). The value should be a directory and

its relative path from src/test/resources folder. For example and for the JSON above, the runner will look for the file TEST1.feature in /pathToProject/src/test/resources/squash/pathRelativeToSquashRoot.

2.4.5 > Additional DryRun Checks

```
<additionnalDryRunChecks>true</additionnalDryRunChecks>
```

Boolean used only if the goal is 'dryrun', ignored otherwise. If activate (true), the TF Gherkin plugin will help you improve the 'dryrun' diagnosis of Cucumber by a search:

- for empty methods
- for methods reduce to the skeleton cucumber: "throw new cucumber.api.PendingException();"

2.4.6 > Reports selection

The same reports as for a TA scripts are available (Html, Junit, Surfire). You can request them by configuring the corresponding exporters. Refer to TA documentation for these topics, only replace the <exporters> tag's name for a SquashTA project with <gkexporters> and add exporter(s) as you want.

If you used our artifact, the specific HTML reporting for the Gherkin tests suite is configured by default as follow :

```
<gkexporters>
        <exporter implementation="org.squashtest.ta.commons.exporter.surefire.
↪SurefireSuiteResultExporter">
                <jenkinsAttachmentMode>${ta.jenkins.attachment.mode}</
↪jenkinsAttachmentMode>
        </exporter>

        <exporter implementation="org.squashtest.ta.gherkin.exporter.
↪HtmlGherkinSuiteResultExporter" />
</gkexporters>
```

Note that you can only have one HTML reporting, either SquashTA ("org.squashtest.ta.commons.exporter.html.HtmlSuiteResultExporter or Cucumber's runner ("org.squashtest.ta.gherkin.exporter.HtmlGherkinSuiteResultExporter")

2.4.7 > TF-TM link

The <gkconfigurers> block of the POM.xml file contains the items for the configuration of the Squash TF Server and squash TM link. It is identical to the <configurers> blocks of other TF projects (same subfields). Refer to TF documentation for this topic.

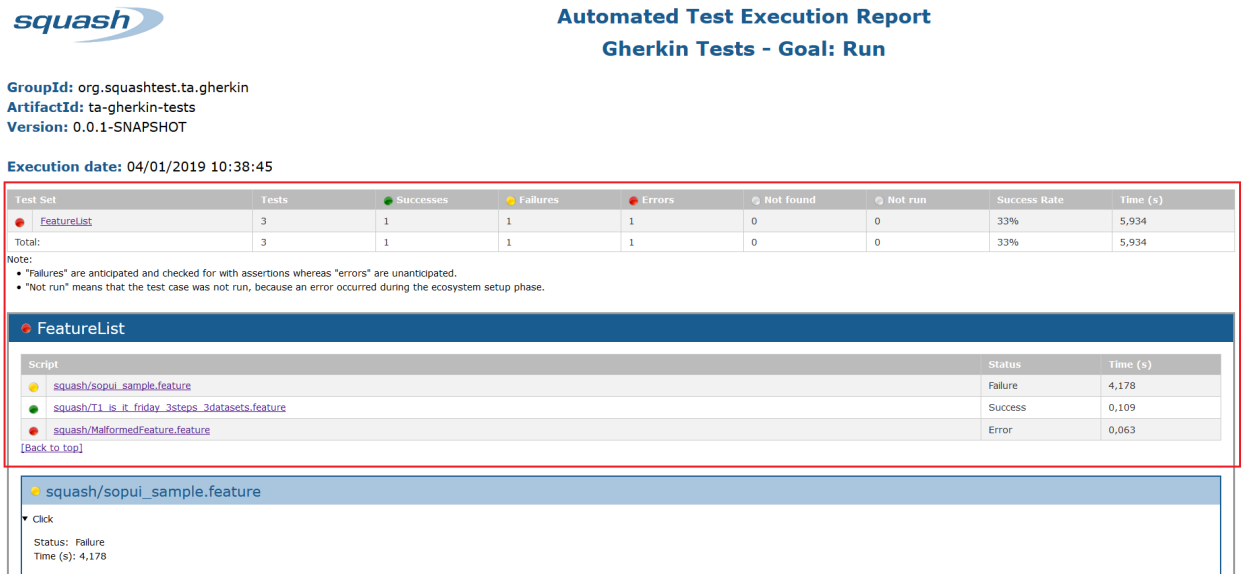
2.5 Expected results

2.5.1 > overview

All TF reports are in the **target/squashTA** subfolder from your java project. We recommend you the most complete reporting for the Gherkin Suite Tests namely to use the TF HTML Gherkin exporter. If you did not create the project

with the squash-ta-cucumber-archetype, check you have configured it (line with 'HtmlGherkinSuiteResultExporter' in the POM.xml file).

This reporting, the file **target/squashTA/html-reports/squash-ta-report.html** starts like this:

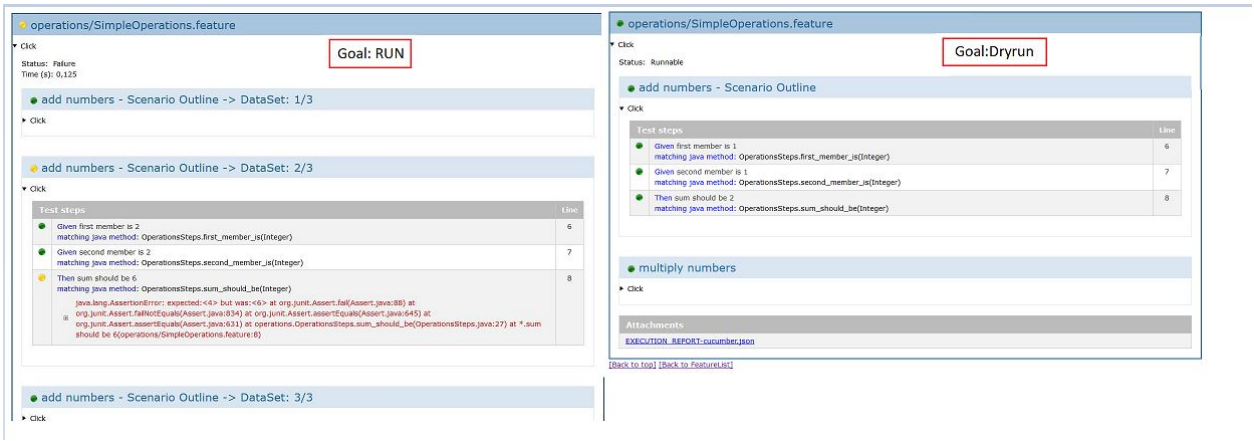


The status of the test suite gives you a summary of the execution of each feature files of the suite (framed in red above).

Except if feature file is NOT_RUN or NOT_FOUND, you can watch for details for each feature file in corresponding dedicated HTML block.

Each block representing a feature contains a sub-block for each scenario.

For a run goal each execution of a outline scenario with a dataset is displayed as a scenario. For a dryrun goal, each scenario outline is displayed only once.



If a result of a scenario is failure or error the following scenarios are executed. For the run goal, the step following the step in error are obviously not run, but for the dryrun goal all the steps are always checked.

Please note that if there is a technical or configuration error by example during compilation or during parsing a malformed feature file, technical informations will be displayed instead of the feature file.

2.5.2 > Definition of Status

The table below gives you the TF status of the feature file/scenario/step for different cases of error. In the case where a feature file contains several scenarios and/or outline scenarios (same scenario with several datasets), the status of the feature is the result is the compilation of any intermediate results.

	Test Status
Ecosystem's setup:	...
no compiling project	NOT_RUN
existing duplicate implementation	(for all tests)
Feature loading:	...
Feature file not found	NOT_FOUND
Feature file not ending by .feature	ERROR
Malformed feature (not parsable file)	ERROR
Executing goal 'run':	...
Step not implemented (no matching)	ERROR
Runtime error (/0, nullPOinter,..)	ERROR
Cucumber.runtime.* exception	ERROR
JUNIT assert	FAILURE
Executing goal 'dryrun':	...
Step not implemented (no matching)	FAILURE
Step matching with empty java method	FAILURE (1)
cucumber.api.PendingException()	FAILURE (1)

(1): only if <additionnalDryRunChecks> is set to true, else status is SUCCESS.