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 gherkin scripts with the TF ecosystem, the test script is the feature file itself. The Java implementation of the gherkin scripts must be compatible with the cucumber framework.

The runner (Maven based) enables to run scripts from an IDE, command line, TF Server but also from **Squash T**est **M**anagment (**Squash TM**).

Differents reporting are available, and in the case where the execution launcher is Squash Test Management (Squash TM), runtime test status and final report are sent back to Squash TM.

The runner allows either to run the features either to check if the features are runable (dryrun) , ie if the automation work was done which means that a java implementation of each step exists.

CHAPTER 2

Contents

2.1 Usage

The users wishing to execute features from SquashTM will refer to the corresponding TM documentation.

2.1.1 Prerequisites

Plateforme

You must have installed on platforme:

- jdk (version >= 8)
- maven (version >= 3.5)

Java project

In case you already have a project gherkin/cucumber and want to run it with TF, just add this build and repository tags to your POM file, update the plugin's version (${ta.cucumber.runner.version}) and adjust the settings like a project from scratch.

```
<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 -->
```

(continues on next page)

(continued from previous page)

```
                <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.commons.exporter.html.
↪HtmlSuiteResultExporter"/>
                    <!--
                        <exporter implementation="org.squashtest.ta.gherkin.exporter.
↪HtmlGherkinSuiteResultExporter" />
                    -->
                </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>

<!-- Squash TA maven plugin repository -->
<pluginRepositories>
    <pluginRepository>
    <id>org.squashtest.plugins.release</id>
    <name>squashtest.org</name>
```

(continues on next page)

(continued from previous page)

```
    <url>http://repo.squashtest.org/maven2/releases</url>
    <snapshots>
    <enabled>false</enabled>
    </snapshots>
    <releases>
    <enabled>true</enabled>
    </releases>
    </pluginRepository>
</pluginRepositories>
```

In case you are starting from scratch, it is recommended to create the test project with the squash-ta-cucumber-archetype. Change the version of the archetype below if necessary (cf http://repo.squashtest.org), and type the following line in a terminal:

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

You must structure your project as follows:

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

A very simple example of a feature and this implementation is provided with the archetype.

The implementation must also conform to the Cucumber framework for java. In particular, there should not be multiple implementations for the same step.

2.1.2 How to use

General

To start the execution of all the tests (features) of the project, use the following commands by replacing /path/to/project with the valid path:

- for the goal "run":

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

- for the goal "dryrun":

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

Options

"ta.feature" parameter

You can restrict 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 list of features. The contents of the json file must look like this exampl**
  {“test”:[{“id”:””,”script”:”pathRelativeToSquashRoot/TEST1.feature”,”param”:””},{“id”:””,”script”:”pathRelativeToSquas

where the given path for the feature must 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.

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

<squashRoot>

In the project POM file, there is this line:

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

The value of this tag point out the root folder wherer TF 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 must 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. If you do not start running from TM, you can change the value of squashRoot or disable it by commenting the line.

<additionnalDryRunChecks>

Boolean used only if the goal is ‘dryrun’, ignored otherwise. If activate (true), the Gherkin plugin TF 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();”

Common TF options

The blocks

- <gkconfigurers>: configuration of the Squash TF Server and squash TM link
- <gkexporters>: selecting reports (html, Junit, surfire)

are respectively identical to blocks <configurers> and <exporters> of the TF framework. Refer TF documentation for these topics.

See also the TF documentation about logger.

Specific reporting for Gherkin’s tests

You can replace the standard TF hml report with a more specific to Gherkin tests html report. To do, comment the line about HtmlSuiteResultExporter in the POM file and uncomment the line about HtmlGherkinSuiteResultExporter.

```
<!-- <exporter implementation="org.squashtest.ta.commons.exporter.html.
↪HtmlSuiteResultExporter"/> -->
<exporter implementation="org.squashtest.ta.gherkin.exporter.
↪HtmlGherkinSuiteResultExporter" />
```

2.1.3 Expected results

The table below gives you the TF status 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.