

TCK

Introduction

The TCK is the standard test suite for verifying that a Groovy implementation complies with the Groovy Language Specification. It is analogous to the JCK (Java Compatibility Kit) test suite used to verify compliance with the Java Language Specification.

The TCK may also be used as a non-normative specification for the Groovy language, or as a part of a functional test suite for a Groovy implementation.

Structure and Workings

On disk, the TCK consists of an Ant script (build.xml), source code for the controlling logic (src directory), source code for the TCK tests (test directory), and the various libraries required to support the running of the TCK (lib directory). To run the tests, the jar file containing the Groovy implementation under test must be placed into the testlib directory, or by default it will use Reference Implementation (RI) libraries for testing.

To run the tests, the Ant script is started (the RI will invoke Ant via Maven). The Ant script:

1. compiles the controlling logic,
2. runs the controlling logic over the TCK test source to generate the raw test source (into the gentest directory),

Step #2, in which TCK test source is converted into raw test source, is the source of much unintended confusion. For clarity, it is important to keep in mind the distinction between controlling unit tests and raw unit tests.

and if you are running outside of the RI maven build, the Ant script:

1. starts a controlling JVM that runs each of the raw tests in a test JVM, using the Groovy implementation under test, and
2. post-processes the resulting JUnit report into HTML pages.

What are all these lines marked '@fail' and '@pass'?

Each Groovy source file in the test directory contains a GroovyTestCase class. Markers (containing the characters '@fail') may be inserted into this file to indicate lines that would cause the test case to fail if included.

There are also '@pass' markers to indicate lines that would not cause the test case to fail if included.

Step #2 in the ant script converts each Groovy test case class into a raw test case class. The raw test case class is a Java JUnit TestCase subclass. Each test method in the raw test case class runs the entire Groovy test case, but with a different one of the '@fail' or '@pass' lines included. The raw test class also includes a version of the Groovy test case with none of the '@fail' or '@pass' lines included.

To recap, if a Groovy test case (from the test directory) has five '@fail' lines and two '@pass' lines, the resulting raw test case (in the gentest directory) will have eight test methods:

- one test method that runs the Groovy test case as-is, expecting it to pass,
- five test methods run the Groovy test case including one of the @fail lines and expecting it to fail, and
- two methods running the Groovy test case including one of the @pass lines and expecting it to pass.

Interpreting the TCK Results

When reading the TCK results, keep in mind that each individual test consists of an entire run of the Groovy test case. Generally, a 'pass' indicates that every test in the Groovy test case passed; if a single test fails, the entire test is marked as failed. For tests including an '@fail' line, a pass indicates that there was at least one failure, though it may not have been the failure that the author intended.

What does this mean? If the TCK results show 100% tests pass, then the Groovy implementation under test has passed the TCK. If less than 100% pass, you should be careful when interpreting the result: a 75% pass rate does not necessarily indicate that the Groovy implementation under test is 75% complete.

For tests within tests, the TCK now reports the errors back up to the top level, be it a maven build or a test report, so no test failures should be missed, please let us know if this isn't the case...

What I Haven't Told You

- There are two kinds of '@fail' lines: plain '@fail' and also '@fail:parse'. When a test case includes '@fail:parse' it is expected to not compile, while '@fail' indicates that the test case should compile but not complete normally.
- You'll have to look at a test to see the full syntax for @pass and @fail lines.
- The TCK results pages contain a link to the System.out output of each test run. This shows the full source of the Groovy test case run by each raw test, as well as highlighting compilation failures.

TCK Limitations

There are many aspects of a Groovy implementation that are not tested by the TCK, such as stability under load. A successful TCK run does not imply that the Groovy implementation under test is suitable for any particular use.