The JUnit Plug-in

Table of contents

1 What's it for?

Why would you want to do use the JUnit plug-in? Three possible reasons spring to mind:

- To write a simple test case around arbitrary Java code, rather than creating a new plug-in. This is the recommended way to replace the functionality of the old `TraderEJBPlugin` that was shipped with early versions of The Grinder. (As an added bonus you end up with some JUnit test cases which you can use to impresses your boss.
- To thrash the heck out of your code in an attempt to discover race conditions (bugs).
- To investigate the statistical effects of subatomic particles passing through your hardware.

Looking for race conditions requires that each instance of the JUnit tests should run against a common fixture - how to achieve in general this is left as an exercise. Please send any useful patterns to grinder-use (mailto:grinder-use@lists.sourceforge.net) . Additionally, because The Grinder currently runs each test cycle in the same order, most race conditions will be hidden. Perhaps a better approach would be to ditch The Grinder 2 and the JUnit plugin and instead use The Grinder 3 (../g3/) to run scripts that exercise the test cases appropriately.

2 JUnit plug-in class

To use the JUnit (http://junit.org/) plug-in, specify:

```
grinder.plugin=net.grinder.plugin.junit.JUnitPlugin
```

3 JUnit plug-in properties

This table lists the JUnit plug-in properties that you can set in `grinder.properties` in addition to the core properties (../g3/properties.html) .

`grinder.plugin.parameter.testSuite`	The fully qualified name of the JUnit test suite class. The class can be anything you can normally pass to a JUnit `TestRunner`. A quick summary: it can either have a method, `public static junit.framework.Test suite()` which returns the `TestSuite` *or* it can define a number of tests methods which will be discovered through introspection - the method names must begin with `test`. (Confusingly, it matters not whether the class directly implements `junit.framework.TestSuite`; this is a JUnit thing, so take any complaints there).
`grinder.plugin.parameter.logStackTra`	Set to `true` to produce stack traces for failures and errors in the error log. The default is `false`.
`grinder.plugin.parameter.initialTest`	The test number used for the first test, subsequent tests are numbered sequentially. This property is useful if you want to use several worker processes to run different test suites against the same console.The default is `0`.

The tests to run are automatically sucked out of the test suite class - you shouldn't specify them individually in grinder.properties. However, you can tweak with individual tests' sleep time as normal. For example, if you want to wait a second before the 6th test in the test suite, you should say

```
grinder.test5.sleepTime=1000
```

JUnit has the concept of *failures* (which occur due to assertions failing) and *errors* (which occur when tests throw exceptions). The Grinder *errors* count for a test is incremented by one if the test causes either a failure or an error.