Script Instrumentation

Table of contents

1 About Instrumentation

The Grinder allows a script to mark the parts of the script code that should be recorded. This is called *instrumentation*.

Code is instrumented for a Test (../g3/scripts.html#tests) . When instrumented code is called, the test's statistics are updated. The standard statistics record the time taken, number of calls, and number of errors. Advanced scripts can add additional custom statistics (../g3/statistics.html) .

We've seen an example of using instrumentation in the introduction (../g3/scripts.html#tests) . To recap, you instrument an object by using a `Test` to modify the Java byte code of the object. Here's the example code again.

```
from net.grinder.script import Test
from net.grinder.script.Grinder import grinder

test1 = Test(1, "Log method")

# Instrument the info() method with our Test.
test1.record(grinder.logger.info)

class TestRunner:
    def __call__(self):
        log("Hello World")
```

Each time "Hello World" is written to the log file, the time taken will be recorded by The Grinder.

Instrumentation can be *nested*. For example, you might instrument a method with Test 1, and the method code might call two `HTTPRequest`s that are instrumented with Test 2 and Test 3. The code instrumented by Tests 2 and 3 is nested within the Test 1 code. The time recorded against the Test 1 will be greater than the total time recorded for Tests 2 and 3. It will also include any time spent in the function itself, for example calls to `grinder.sleep()`.

2 Supported targets

A wider range of target objects can be instrumented.

Java instance	Each call to a non-static method is recorded, including calls to super classes methods. Instances of arrays and primitive types cannot be instrumented.
Java class	Each call made to a constructor or a static method declared by the class is recorded. Calls of non-static methods or static methods defined by super classes are not recorded.
Jython instance	Each call to an instance method is recorded.
Jython function or method	Each call of the function or method is recorded.
Jython class	Each call made to the Jython class (i.e. constructor calls) is recorded.
Clojure function	Each call of the function is recorded.

JVM classes loaded in the bootstrap classloader, and classes from The Grinder's `net.grinder.engine.process` implementation package cannot be instrumented.

3 Selective instrumentation

The Grinder 3.7 adds an overloaded version of record (../../g3/script-javadoc/net/grinder/script/Test.html#record(java.lang.Object, net.grinder.script.Test.InstrumentationFilter)) that allows the target object to be instrumented selectively.

Selective instrumentation is useful for instrumenting instances of the HTTPRequest (../../g3/script-javadoc/net/grinder/plugin/http/HTTPRequest.html) class, which has ancillary methods that typically need to be called without affecting test statistics. Here's an example of how to use selective instrumentation.

```
from net.grinder.script import Test
from net.grinder.plugin.http import HTTPRequest

test = Test(1, "my test")

class GetAndPostFilter(Test.InstrumentationFilter):
  def matches(self, method):
    return method.name in ["GET", "POST"]

request = HTTPRequest(url="http://grinder.sourceforge.net")
test.record(request, GetAndPostFilter())

class TestRunner:
    def __call__(self):
        # GET() is instrumented, so call statistics are reported.
        request.GET()

        # getUrl() is not instrumented, no call statistics are reported.
        print "Called %s" % request.url
```

4 Troubleshooting Instrumentation

The instrumentation relises on Dynamic Code Redefinition, a Java 6 feature.

When you start an agent process, you will normally see a line like this in the worker process log file (../g3/getting-started.html#Output) .

```
    16/11/09 08:02:18 (process paston01-0): instrumentation agents:
    byte code transforming instrumenter for Jython 2.1/2.2; byte code transforming
 instrumenter for Java
```

If you see the following line, you should check you are using a Java 6 JVM.

```
     16/11/09 07:59:42 (process paston01-0): instrumentation agents: NO INSTRUMENTER
COULD BE LOADED
```