Japanese Using JMockit with Groovy

JMockit is a single class with a small set of static methods, which allow arbitrary methods and constructors of any other class to be replaced by mock implementations at runtime. It has the following features:

- no particular design must be followed by code under test, e.g.:
  - you don't need to have interfaces everywhere
  - you don't need to avoid static method calls
  - you don't need to use dependency injection, i.e. you can have `new SomeClass()` calls throughout your code
  - you don't need to worry about `final` classes
- legacy code can be unit tested without the need for any adaptation

Since JMockit depends on the JVM class redefinition mechanism exposed by java.lang.instrumentation, Groovy, JUnit or TestNG tests that use it must be run under a Java SE 5 VM. However, application and test code can still be compiled to older versions of the language.

The sections below illustrate using JMockit for the mocking parts of Using Testing Frameworks with Groovy.

The Item Storer Example

We are going to consider how you might use JMockit as part of testing the Item Storer Example.

First we define the following class in a file called `MockReverser.groovy`:

```
class MockReverser implements Reverser {
    private testData = [123.456:-123.456, hello:'olleh']
    def reverse(value) { testData[value] }
}
```

We place this in a separate file rather than straight in the same script as the instrumentation classes in the JVM (which JMockit relies upon) don't know about classes compiled in memory by Groovy. They will look for replaced classes on the classpath.

Now, here is how we can test `Storer`:

```
// require(url:'https://jmockit.dev.java.net', jar='jmockit.jar')
// require(url:'https://jmockit.dev.java.net', jar='jmockit-asm2.jar')
// needs to be run with "-javaagent:jmockit.jar"
import mockit.*

def checkReverse(storer, value, reverseValue) {
    storer.put(value)
    assert value == storer.get()
    assert reverseValue == storer.getReverse()
}

Mockit.redefineMethods(GroovyReverser, MockReverser)
def storer = new Storer()
checkReverse(storer, 123.456, -123.456)
checkReverse(storer, 'hello', 'olleh')
```

If you recall from the example, `Storer` does a `new GroovyReverser()` call, so we use `Mockit.redefineMethods()` to replace the original version with our mock version.