Unit Testing using Spock

Imran Mir

Imran Mir
Technology Lead
To The New
Twitter: imirimran
LinkedIn: www.linkedin.com/in/imran-mir-81254010

Highlights

- Spock and its advantages
- Spock basics
- Data driven testing
- Spock Mocking
- Extensions

Unit testing always takes a back seat

- Does not provide short term gratification
- Consumes resources -Time and Money
- Tooling is not great
  - Verbosity of code
  - Verbosity in reports

Spock

- A testing framework for Java and Groovy applications
- Based on Groovy
- Compatible with JUnit
- Helps writing expressive tests in less time

Setup

- For java apps
  - Groovy
  - Spock
  - cglib

Do I need to learn Groovy to learn spock

- Very small learning curve for Java developers
- You just need to get used to a different tune of Java

Groovy

Powerful optionally typed, dynamic language, with static-typing and static compilation capabilities

Spock Basics

Spec

```
import spock.lang.Specification

class UserSpec extends Specification {

}
```

Fixture methods

```
def setup(){}      //execute just before a test
def cleanup(){}    //execute just after a test
def setupSpec(){}  //execute just once before first test
def cleanupSpec(){}  //execute just once after the last test
```

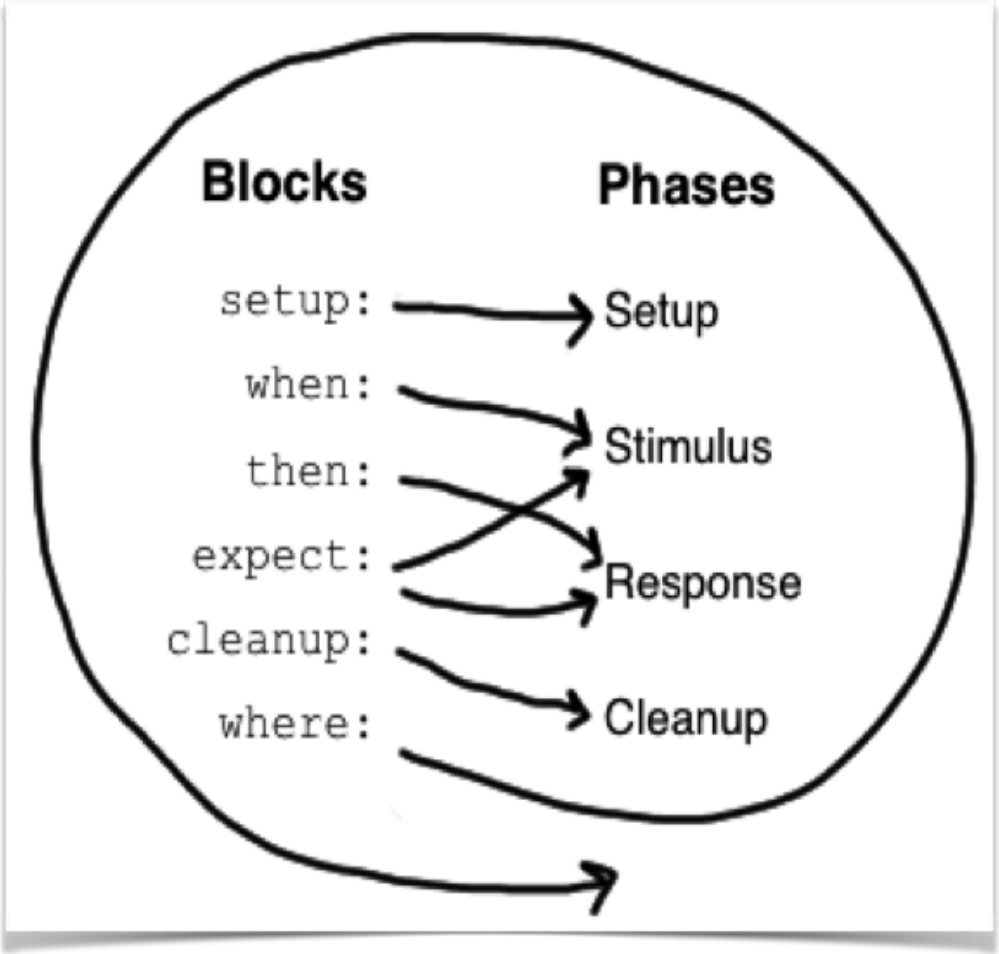
JUnit Test case

```
@Test
public void testGetDisplayNameForAdultMale() throws Exception {
```

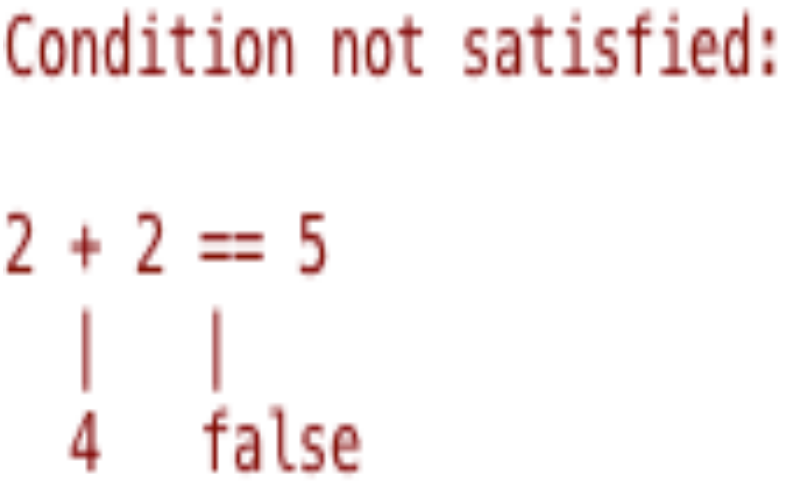
Feature Methods

```
import spock.lang.Specification

class ExampleSpec extends Specification {

    void "State the feature to be tested"() {
        //Test Code
    }

}
```

Blocks


Blocks
Phases
setup:
Setup
when:
Stimulus
then:
expect:
Response
cleanup:
Cleanup
where:

Reports



Scalene triangle has none of the side equal to each other

```
Condition not satisfied:

triangleType == "Scalene1"
|            |
Scalene      false
             1 difference (87% similarity)
             Scalene(-)
             Scalene(1)

        at com.gr8conf.demo.TriangleTest.Scalene triangle has none of the side equal to each other(TriangleTest.groovy:22)
```

Best Documentation

```
def "The newly bought kettle makes tea"() {
    setup: "Need some tea leaves"
    // code goes here

    and: "and some  water"
    // code goes here

    and: "and the kettle"
    // code goes here

    when: "contents are boiled"
    // code goes here

    then: "we taste some delicious tea"
    // compare the result
}
```

Testing exception conditions

- An exception is thrown
- An exception is not thrown

Data Driven Testing

Spock separates data from code

Spock Extensions

Hook into the spec's lifecycle to enrich or alter its behavior:

- IgnoreRest
- Ignore
- IgnoreIf
- Requires
- PendingFeature
- Stepwise
- ConfineMetaClassChanges
- Subject

Interaction Based Testing

Explores how the object(s) under specification interact by way of method calls

Mocks and Stubbing

- Focus is on the behaviour rather than the state of objects
- Mocks listen to the interactions

```
setup:

def emailService = Mock(EmailService)
EmailService emailService2 = Mock()
```

Stubbing

```
PointsAnalyzerService analyzerService = Mock(PointsAnalyzerService)
analyzerService.analyze(account) >> PointsBand.GOLD
```

Partial mocking

- Spy

Extending Spock

We can create our own extensions

QUESTIONS ?

References

- http://spockframework.org
- https://meetspock.appspot.com/
- https://github.com/imranmir/gr8conf_spock_demo
- https://github.com/kensipe/spock-javaone2014