Specification Testing

Idaho State University | Computer Science

Isaac Griffith

CS 2263
Department of Informatics and Computer Science
Idaho State University

ROAR

Outcomes

After today's lecture you will be able to:

- Describe the basic concepts of specification testing
- Use the Spock framework to implement and execute specification tests
- Understand how Spock is useful for BDD

Inspiration

“The tools that are most effective are fairly simple, but have enough subtlety and nuance that they can be very powerful”

Why not just use JUnit or TestNG?

- JUnit and TestNG are good testing frameworks, but
  - often need an additional mocking or stubbing library
  - syntax generally limited to Java
- Spock has some built-in advantages:
  - mocking and stubbing part of core
  - succinct syntax for data driven testing
  - leverages Groovy syntax

Do we need another framework?

- Spock incorporates the best concepts from JUnit, RSpec, jMock, and Mockito
- The answer is obviously yes; otherwise there wouldn't be this lecture, right?
- Plus Spock makes testing fun!

How Spock Measures Up

- Unit Testing
  - JUnit
  - TestNG
- Mocking and Stubbing
  - EasyMock
  - jMock
  - Mockito
  - PowerMock
  - jMockit
- BDD
  - Cucumber
  - JBehave
- Spock does all of this

Comparison to JUnit Concepts

JUnit	Spock
Test Class	Specification
Test	Feature
Test Method	Feature method
@Before	setup()
@After	cleanup()
Assertion	Condition
@TEST(expected...)	Exception condition

Simple Broken Tests

JUnit

```
public class CalculatorTest {
  Calculator calculator;

  @Before
  public void setup() {
    calculator = new Calculator();
  }

  @Test
  public void testSimpleAddition() {
    assertEquals("2+2=4", 5,
        calculator.add(2, 2));
  }
}
```

Spock

```
class CalculatorSpec extends Specification {
  def calculator

  def setup() {
    calculator = new Calculator()
  }

  def "Test Simple Addition"() {
    expect:
      calculator.add(2, 2) == 5
  }
}
```

Simple Broken Test Results

JUnit Failed Test Output

```
CalculatorTest > testSimpleAdditon FAILED
  java.lang.AssertionError: 2+2=4 expected:<5> but was:<4>
    at org.junit.Assert.fail(Assert.java:88)
    at org.junit.Assert.failNotEquals(Assert.java:834)
    at org.junit.Assert.assertEquals(Assert.java:645)
    at CalculatorTest.testSimpleAddition(CalculatorTest.java:15)
```

Spock Failed Test Output

```
CalculatorSpec > Test Simple Addition FAILED
    Condition not satisfied:

    calculator.add(2,2) == 5
    |          |        |
    |          4        false
    Calculator@11dd18e
        at CalculatorSpec.Test Simple Additon(CalculatorSpec.groovy:13)
```

Spruced up using Spock Parameterization

```
class CalculatorSpec extends Specification {
  @Unroll
  def "#a + #b = #c"() {
    setup:
      def calculator = new Calculator()
    expect:
      calculator.add(a, b) == c
    where:
      a | b || c
      2 | 2 || 4
      2 | 2 || 5
  }
}
```

Parameterized Test Output

Spock Passed Test Output:

```
CalculatorSpec > 2 + 2 = 4 PASSED
```

Spock Failed Test Output:

```
CalculatorSpec > 2 + 2 = 5 FAILED
    Condition not satisfied:

    calculator.add(a,b) == c
    |          |   | |  |  |
    |          4   2 2  |  5
    Calculator@11dd18e  false
        at CalculatorSpec.#a + #b = #c(CalculatorSpec.groovy:19)
```

Spock and Mocking

```
def "Mocked calculator"() {
  setup:
    def calculator = Mock(Calculator)
  when:
    calculator.add(2,2)
  then:
    1 * calculator.add(2,2)
}
```

Spock and Mocking

```
def "Mocked calculator"() {
  setup:
    def calculator = Mock(Calculator)
  when:
    calculator.add(2,2)
  then:
    1 * calculator.add(2,2)
    0 * calculator.add(_,_)
}
```

Spock and Stubbing

```
def "Stubbed calculator"() {
  setup:
    def calcultor = Stub(Calculator) {
      add(2,2) >> 4
    }
  expect:
    calculator.add(a,b) == c
  where:
    a | b || c
    2 | 2 || 4
}
```

Spock and Stubbing

```
def "Stubbed calculator"() {
  setup:
    def calcultor = Stub(Calculator) {
      add(2,2) >> 4
      add(_,_) >> {x, y -> x + y}
    }
  expect:
    calculator.add(a,b) == c
  where:
    a | b || c
    2 | 2 || 4
    3 | 3 || 6
    4 | 3 || 7
}
```

Spock and BDD

Classic Example of BDD:
Scenario: Multiple Givens
Given: one thing
And: another thing
And: yet another thing
When: I open my eyes
Then: I see something
But: I don't see something else

Valid Spock Code

```
def "Multiple Givens"() {
  given: "one thing"
  and: "another thing"
  and: "yet another thing"
  when: "I open my eyes"
  then: "I see something"
  and: "I don't see something else"
}
```

Idaho State University
Computer Science

Spock plus BDD and Parameterization

```
@Unroll
def "BDD: #a + #b = #c"() {
  given: "a new calculator"
    def calculator = new Calculator()
  and: "nothing is done to the calculator before addition"
  when: "adding two values together"
    def sum = calculator.add(a, b)
  then: "the result is the expected sum"
    c == sum
  where:
    a | b || c
    2 | 2 || 4
    3 | 2 || 5
}
```


ROAR

Will Spock play with my IDE

- Spock has a built in JUnit runner called Sputnik that makes it transparent to most Java IDE's and build tools

Are there any questions?