SPOCK

MARIO ZUPAN - @MZUPZUP

ABOUT ME

- **@netconomy for 4 years**
- **~2.5 years of Java**
- **~1.5 years of JavaScript**
- **Software Craftsmanship**

AGENDA

- **Overview of Spock's Capabilities**
- **Code-Examples**
- **How to use it**
- **Impulse to try it**

SPOCK

SPOCK FACTS

- **around since 2008**
- **1.0 just got released**
- **created by an Austrian (@pniederw)**
- **It uses Groovy!**

- **Simple**
- **Syntactic Sugar**
- **Functional Features**
- **List / Map Literals**
- **...**

SPOCK FEATURES

- **Learning Curve**
- **Unrolling**
- **Data-Driven Testing**
- **Interaction Based Testing**
- **Framework Integration**
- **DSL**

TEST CLASS

```
import spock.lang.*;
class HelloSpock extends Specification {
    ...
}
```

TEST CASE

```
import spock.lang.*;
class HelloSpock extends Specification {
    def "adds two numbers" () {
        ...
    }
}
```

WHOLE TEST

```
import spock.lang.*;
class HelloSpock extends Specification {
    def "adds two numbers" () {
        expect:
        add(a, b) == c

        where:
        a | b || c
        1 | 7 || 8
        4 | 4 || 8
        7 | 1 || 8
    }
}
```

BLOCKS - SETUP

```
import spock.lang.*;
class HelloSpock extends Specification {
    def "adds to the wallet" () {
        setup:
        Wallet wallet = new Wallet()
    }
}
```

BLOCKS - GIVEN

```
import spock.lang.*;
class HelloSpock extends Specification {
    def "adds to the wallet" () {
        given:
        Wallet wallet = new Wallet()
    }
}
```

BLOCKS - WHEN

```
import spock.lang.*;
class HelloSpock extends Specification {
    def "adds to the wallet" () {
        given:
        Wallet wallet = new Wallet()

        when:
        wallet.add(100)
    }
}
```

BLOCKS - THEN

```
import spock.lang.*;
class HelloSpock extends Specification {
    def "adds to the wallet" () {
        given:
        Wallet wallet = new Wallet()

        when:
        wallet.add(100)

        then:
        wallet.get() == 100
    }
}
```

BLOCKS - AND

```
import spock.lang.*;
class HelloSpock extends Specification {
    def "adds to the wallet" () {
        given: "there is a wallet"
        Wallet wallet = new Wallet()

        when: "100 euros are added"
        wallet.add(100)

        and: "another 100 euros are added"
        wallet.add(100)

        then: "there are 200 euros in the wallet"
        wallet.get() == 200
    }
}
```

BLOCKS - CLEANUP

```
import spock.lang.*;
class HelloSpock extends Specification {
    def "adds to the wallet" () {
        given:
        Wallet wallet = new Wallet()

        when:
        wallet.add(100)

        then:
        wallet.get() == 100

        cleanup:
        wallet.delete()
    }
}
```

BLOCKS - WHERE

```
import spock.lang.*;
class HelloSpock extends Specification {
    def "computing the maximum of two numbers"() {
      expect:
      Math.max(a, b) == c

      where:
      a << [5, 3]
      b << [1, 9]
      c << [5, 9]
    }
}
```

FAILURE REPORTING

```
Condition not satisfied:

max(a, b) == c
|    |  |  |  |
3    1  3  |  2
          false
```

FAILURE REPORTING

```
class UnrollingTest extends Specification {
    @Unroll
    def "maximum of #left and #right
         is #maximum" (int left, int right, int maximum) {

        expect:
        Math.max(left, right) == maximum

        where:
        left | right | maximum
        1 | 3 | 2
    }
}
```

FAILURE REPORTING

```
- maximum of 1 and 3 is 2   FAILED

 Condition not satisfied:

Math.max(left, right) == maximum
     |   |     |       |   |
     3   1     3       |   2
                     false
```

EXCEPTIONS

```
class StackSpec extends Specification {
    def "throws on pop when the stack is empty" () {
        given:
        stack = new Stack()

        when:
        stack.pop()

        then:
        thrown EmptyStackException
    }
}
```

SHARED RESOURCES

```
@Shared resource = new ExpensiveResource()
```

DATA TABLES

```
class DataDriven extends Specification {
    def "maximum of two numbers"() {
        expect:
        Math.max(a, b) == c

        where:
        a | b || c
        3 | 5 || 5
        7 | 0 || 7
        0 | 0 || 0
    }
}
```

DATA TABLES

```
class DataDriven extends Specification {
    def "maximum of two numbers"() {
        expect:
        Math.max(a, b) == c

        where:
        a << [3, 7, 0]
        b << [5, 0, 0]
        c << [5, 7, 0]
    }
}
```

DATA TABLES

```
class DataDriven extends Specification {
    def "maximum of two numbers"() {
        expect:
        Math.max(a, b) == c

        where:
        a = 3
        b = Math.random() * 100
        c = a > b ? a : b
    }
}
```

DATA TABLES

```
class DataDriven extends Specification {
    def "transform numbers" () {
        given:
        NumTransformer trans = new NumTransformer()

        expect:
        trans.transform(arr) == res

        where:
        arr          || res
        null         || null
        []           || []
        [1]          || [2]
        [1, 2, 3]    || [2, 3, 4]
    }
}
```

INTERACTION BASED TESTING

```groovy
class InteractionTest extends Specification {
    def "" () {
        given:
        ProductService productService = new ProductService()
        Cache cache = Mock()
        productService.setCache(cache)

        when:
        productService.getProduct('123')

        then:
        1 * cache.get('products', '123')
    }
}
```

CARDINALITY

```
5 * cache.get('products', '123')
0 * cache.get('products', '123')

(1..5) * cache.get('products', '123')
(1.._) * cache.get('products', '123')
(_..5) * cache.get('products', '123')

_ * cache.get('products', '123')
```

CONSTRAINTS

```
1 * _.get()
1 * cache./get.*s/("123") // getProducts, getArticles
1 * cache._('products', '123') // any method on cache

1 * cache.get('products', '123') // normal
1 * cache.get('products', '!123') // not '123'
1 * cache.get('products', _ as String) // any string
1 * cache.get('products', {it.size() > 2}) // any string longer than 2

1 * cache.get('products', _) // any code
1 * cache.get(*_) // any list of arguments
```

ORDER

```
when:
productService.getArticle('123')

then:
1 * cache.get('products', '123')

then:
1 * cache.get('images', '123')
```

SCOPE

```
when:
productService.getProduct('123')

then:
cache.get('products', '123')

when:
productService.getProduct('789')

then:
cache.get('products', '789')
```

STUBBING

```
cache.get(_) >> new Product('123')
cache.get('123') >> new Product('123')
cache.get('789') >> new Product('789')

cache.get(_) >> [new Product('789'), new Product('123')] // multiple

// compute return value
cache.get(_) >> { args -> args[0] == '123' ? new Product('123') : null}

cache.get(_) >> { throw new NoCacheActiveException() }
// closures can be chained as well
```

SPIES

```
class InteractionTest extends Specification {
    def "calls real methods" () {
        given:
        cache = Spy(Cache)

        when:
        productService.getProduct('123') // calls the real cache inside

        then:
        1 * cache.get('products', '123')
    }
}
```

GROOVY SPECIALS

- **Mock Constructors**
- **Mock Statics**
- **All-Instance-Mocking**
- **For Java -> Powermock**

EXTENSIONS

- **@Ignore / @IgnoreRest / @IgnoreIf()**
- **@Requires()**
- **@Stepwise**
- **@AutoCleanup**
- **@Timeout()**
- **@Subject / @Issue / @Title / @Narrative**
- **Spock-Genesis**

INTEGRATIONS

- **Spring Framework**
- **Tapestry**
- **Guice**
- **Grails**
- **Eclipse**
- **Intellij**

WRAP-UP

THANK YOU

- **@mzupzup**
- **https://github.com/zupzup/jug-spock-talk**