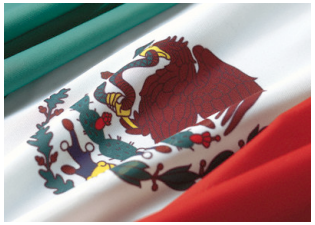
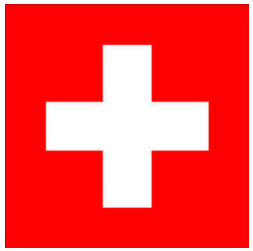
SPOCK'S NEW TRICKS

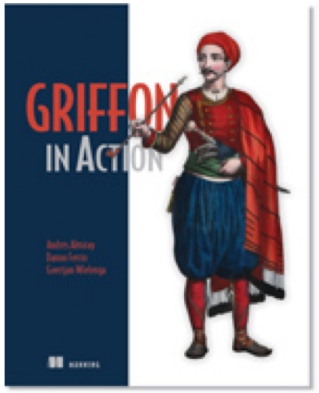
ANDRES ALMIRAY

@AALMIRAY

ANDRESALMIRAY.COM







JAVA™
CHAMPIONS

ORACLE
Developer
Champion

@aalmiray

JCP Executive Committee Associate Seat

Committer

Committer

JSR377 Specification Lead

@aalmiray

SPOCK 1.0
(2015-03-02)

@aalmiray

FEATURES

- Data Tables
- Integration with Guice, Spring, Tapestry, Unitils, Grails
- Compatible with JUnit4 extensions
- Custom mocking framework (Java/Groovy)
- Extensible

PARAMETERIZATION

```
@Unroll
class HelloServiceSpecification extends Specification {
    def "Invoking sayHello('#input') yields '#output'"() {
        given:
        HelloService service = new DefaultHelloService()

        when:
        String result = service.sayHello(input)

        then:
        result == output

        where:
        input   | output
        ''      | 'Howdy stranger!'
        'Test'  | 'Hello Test'
    }
}
```

MOCK + INTERACTIONS

```
class SampleSpec extends Specification {
    def "A mocking example"() {
        given:
        Collaborator collaborator = Mock()
        1 * collaborator.foo() >> 'Spock'
        Component component = new Component(collaborator)
        when:
        String output = component.doit('is Groovy!')
        then:
        output == 'Spock is Groovy!’
    }
}
```

CARDINALITY

```
1 * subscriber.receive("hello")
0 * subscriber.receive("hello")
(1..3) * subscriber.receive("hello")
(1.._) * subscriber.receive("hello")
(_..3) * subscriber.receive("hello")
_ * subscriber.receive("hello")
```

TARGET & METHOD

```
1 * subscriber.receive("hello")
1 * _.receive("hello")
1 * subscriber._("hello")
1 * subscriber./r.*e/("hello") )
1 * _._("hello")
```

ARGUMENTS

```
1 * subscriber.receive("hello")
1 * subscriber.receive(!"hello")
1 * subscriber.receive()
1 * subscriber.receive(_)
1 * subscriber.receive(*_)
1 * subscriber.receive(!null)
1 * subscriber.receive(_ as String)
1 * subscriber.receive({ it.size() > 3 })
```

ASCII ART?

```
(_.._) * _._(*_) >> _
```

SPOCK 1.1
(2017-05-01)

@aalmiray

MULTIPLE ASSERTIONS

- **Known in AssertJ as “Soft Assertions”**

```
then:
verifyAll {
    a == something
    b == anotherValue
}
```

@PENDINGFEATURE

```
@spock.lang.Unroll
class Pending extends spock.lang.Specification {
    @spock.lang.PendingFeature
    def "Should implement this later"() {
        expect:
        num1 + num2 == result

        where:
        num1 | num2 || result
        2    | 2    || 4       // passes!
        2    | 2    || 3       // fails!
    }
}
```

DATA TABLES

```
@spock.lang.Unroll
class CalculatorSpec extends spock.lang.Specification {
    def "Sum of two numbers"() {
        expect:
        num1 + num2 == result

        where:
        num1 | num2 || result
        2    | 2    || 4
        2    | num1 || 4
        2    | 2    || num1 + num2
    }
}
```

DETACHED MOCKS (SPRING)

- **There are two choices for creating mocks**
  - DetachedMockFactory
  - SpockMockFactoryBean

DETACHED MOCKS (SPRING)

```
class DetachedJavaConfig {
  def mockFactory = new DetachedMockFactory()

  @Bean GreeterService serviceMock() {
    return mockFactory.Mock(GreeterService)
  }

  @Bean GreeterService serviceStub() {
    return mockFactory.Stub(GreeterService)
  }

  @Bean GreeterService serviceSpy() {
    return mockFactory.Spy(GreeterServiceImpl)
  }
}
```

DETACHED MOCKS (SPRING)

```
class DetachedJavaConfig {
  @Bean
  FactoryBean<GreeterService> alternativeMock() {
    return new SpockMockFactoryBean(GreeterService)
  }
}
```

SPOCK 1.2
(2018-09-23)

@aalmiray

DETACHED MOCKS (SPRING)

- **Two additional choices inspired in Spring Boot's @MockBean**
  - @SpringBean
  - @SpringSpy

DETACHED MOCKS (SPRING)

```
@org.springframework.test.context.ContextConfiguration
class SpringExample extends spock.lang.Specification {
  @org.spockframework.spring.SpringBean
  Service1 service1 = Mock()

  @org.spockframework.spring.SpringBean
  Service2 service2 = Stub() {
    generateQuickBrownFox() >> "blubb"
  }

  // continued in next slide
```

DETACHED MOCKS (SPRING)

```
  def "injection with stubbing works"() {
    expect:
    service2.generateQuickBrownFox() == "blubb"
  }

  def "mocking works was well"() {
    when:
    def result = service1.generateString()

    then:
    result == "Foo"
    1 * service1.generateString() >> "Foo"
  }
}
```

DETACHED MOCKS (GUICE)

- Similar to Spring's support, you may use DetachedMockFactory with Guice's Injector

JUKITO

```java
@RunWith(JukitoRunner.class)
public class AppControllerTest {
    @Inject private AppController controller;
    @Inject private AppModel model;

    @Test
    public void happyPath(HelloService service) {
        // given:
        String input = "Test";
        String output = "Hello Test";
        when(service.sayHello(input)).thenReturn(output);

        // when:
        model.setInput(input);
        controller.sayHello();

        // then:
        assertThat(model.getOutput(), equalTo(output));
        verify(service, only()).sayHello(input);
    }
}
```

SPOCK

```
@UseModules(TestModule)
class AppControllerSpec extends Specification {
    @Inject private AppController controller
    @Inject private AppModel model
    @Inject private HelloService service

    def happyPath() {
        given: "The HelloService mock is configured to return 'Hello \$input'"
        1 * service.sayHello('Test') >> 'Hello Test'

        when: "The input is set to 'Test'"
        model.input = 'Test'

        and: "The sayHello action is invoked on the controller"
        controller.sayHello()

        then: "The output should be 'Hello Test'"
        model.output == 'Hello Test'
    }


    // continued in next slide
```

SPOCK

```
    // continued from last slide


    static class TestModule extends AppModule {
        private final MockFactory mockFactory = new DetachedMockFactory()

        @Override
        protected void bindHelloService() {
            bind(HelloService).toInstance(mockFactory.Mock(HelloService))
        }
    }
}
```

@AUTOATTACH

- Use this feature if you need to attach mocks without leveraging Spring or Guice

@AUTOATTACH

```
class AutoAttachSpec extends spock.lang.Specification {
  def mockFactory = new spock.mock.DetachedMockFactory()

  @spock.mock.AutoAttach
  List<String> mock = mockFactory.Mock(List)

  def "Auto attaches mock to spec"() {
    when:
    mock.add("foo")

    then:
    1 * mock.add(_)
  }
}
```

Vladimír Oraný

@musketyr Follows you

father, husband, groovy & dsl enthusiast, creator of Spreadsheet Builder and Dru & Gru test frameworks, test facilitator @agorapulse

Prague, Czech republic

medium.com/@musketyr/

@aalmiray

@RETRY

```
@spock.lang.Unroll
class RetryExample extends spock.lang.Specification {
    @spock.lang.Retry
    def bar() {
        expect:
        test

        where:
        test << [false, true, true]
    }
}
```

ADDITIONAL INTERFACES

```
class AdditionalInterfaces extends spock.lang.Specification {
  def "java stubs"() {
    given:
    def stub = Stub(List, additionalInterfaces: [Closeable])

    expect:
    stub instanceof List
    stub instanceof Closeable
  }
}
```

ADDITIONAL INTERFACES

```
class AdditionalInterfaces extends spock.lang.Specification {
  def "java mocks"() {
    given:
    def mock = Mock(List, additionalInterfaces: [Closeable])

    expect:
    mock instanceof List
    mock instanceof Closeable
  }
}
```

REPORTS

SPOCK-REPORTS

- https://github.com/renatoathaydes/spock-reports

- **Generates developer friendly reports.**
- **Generates customer friendly reports, following BDD conventions**

Specification run results

Specifications summary:

Created on Tue May 29 10:28:30 CEST 2018 by aalmiray

Total	Passed	Failed	Feature failures	Feature errors	Success rate	Total time
8	8	0	0	0	100.0%	1.626 seconds

Specifications:

Name	Features	Failed	Errors	Skipped	Success rate	Time
CalculatorSpec	3	0	0	0	100.0%	0.028 seconds
Pending	0	0	0	1	100.0%	0.006 seconds
S1Spec	1	0	0	0	100.0%	0.002 seconds
S2Spec	1	0	0	0	100.0%	0.001 seconds
SampleSpec	1	0	0	0	100.0%	0.007 seconds
org.kordamp.javatrove.hello.pmvc.FunctionalSpec	2	0	0	0	100.0%	1.552 seconds
org.kordamp.javatrove.hello.pmvc.controller.AppControllerSpec	1	0	0	0	100.0%	0.023 seconds
org.kordamp.javatrove.hello.pmvc.service.HelloServiceSpecification	2	0	0	0	100.0%	0.007 seconds

SPOCK-REPORTS

```
def happyPath() {
    given: "The HelloService mock is configured to return 'Hello \$input'"
    1 * service.sayHello('Test') >> 'Hello Test'

    when: "The input is set to 'Test'"
    model.input = 'Test'

    and: "The sayHello action is invoked on the controller"
    controller.sayHello()

    then: "The output should be 'Hello Test'"
    model.output == 'Hello Test'
}
```

Report for org.kordamp.javatrove.hello.pmvc.controller.AppControllerSpec

Summary:

Created on Tue May 29 10:28:28 CEST 2018 by aalmiray

Executed features	Failures	Errors	Skipped	Success rate	Time
1	0	0	0	100.0%	0.023 seconds

Features:

- happyPath

happyPath

Return

Given: The HelloService mock is configured to return 'Hello $input'

When: The input is set to 'Test'

And: The sayHello action is invoked on the controller

Then: The output should be 'Hello Test'

MIGRATION

JUNIT2SPOCK

- Lukasz Opaluch wrote Junit2Spock
- https://github.com/opaluchlukasz/junit2spock

@aalmiray

HTTP://ANDRESALMIRAY.COM/NEWSLETTER

HTTP://ANDRESALMIRAY.COM/EDITORIAL

THANK YOU!

ANDRES ALMIRAY
@AALMIRAY

ANDRESALMIRAY.COM