



November 15-16, 2022
Seattle, WA

Achieving Gold Standard Software Quality Goals Using Groovy & Spock Testing

UNIT TESTING IN THE 2020'S








Toine Siebelink &
Lee Anjella Macabuhay



1

Purpose of Unit Tests




Purpose of Unit Tests


- ✓ **Validation:** Proves what the code does
- ✓ **Assurance:** Prevent regression
- ✓ **Design:** TDD / BDD
- ✓ **Documentation:** Describes the behavior (happy path, edge cases, error scenarios)

2 / 15


2


Misunderstandings









Common Misunderstandings

- 

“Unit test is for Code Coverage”
- 

“Unit test should test only one class”
Don't be dogmatic, be pragmatic!
- 

“Mocking is difficult”
Spock makes it intuitive!
- 

“Unit test are redundant I have good integration test”
Please read up on the testing pyramid!

3 / 15

3

Advantages of Spock and Groovy







LANGUAGE STRUCTURE AND DESCRIPTIONS PERFECTLY SUITED FOR HUMAN READABLE TEST



DATA DRIVEN MADE EASY USING PIPES OR WHERE: BLOCK



GROOVY MAGIC:
- ACCESS TO PRIVATE FIELDS
- GROOVY COLLECTIONS & MAPS
- AND MUCH MORE



SETTING UP STUBS, MOCKS AND BEHAVIOUR EASIER THEN EVER BEFORE

4 / 15

4

Method Names and Labels



Traditional junit

```
@Test
public void testCreateAndRetrieveDataspac() {
    // When a new dataspac is created

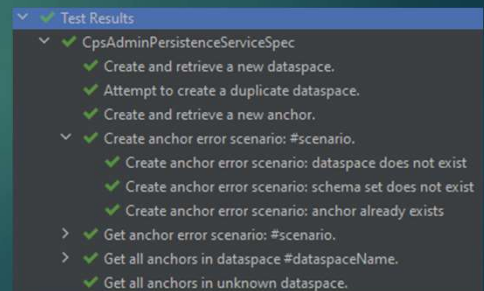
    // Then it can be retrieved
}
```



Spock & Groovy

```
def 'Create and retrieve a new dataspac.'() {
    when: 'a new dataspac is created'

    then: 'it can be retrieved'
}
```



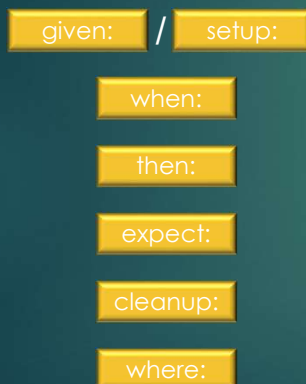
5 / 15

5

Blocks (incl. data-driven)



Blocks (labels)



Extend 'labels' with descriptions

```
when: '#x is raised to the power #y'
    def result = Math.pow(x, y)

then: 'the result is #expectedResult'
    assert result == expectedResult

where: 'the following values are used'
    x | y || expectedResult
    3 | 2 || 9
    2 | 3 || 8
    3 | 0 || 1
    4 | -1 || 0.25
```

Data Driven

6 / 15

6

Access to private members



Traditional junit

```
AdminService {
    public void setCpsCacheForTest(CpsCache cpsCache) {
        this.cpsCache = cpsCache
    }
}

AdminServiceTest {
    objectUnderTest.setCpsCacheForTest( Mock(CpsCache) )
}
```



Spock & Groovy

```
AdminServiceSpec {
    setup: { objectUnderTest.cpsCache = Mock(CpsCache) }
}
```

7 / 15

7

Easy Mocking & Invocation Checks



Spy

Stub

Mock

Defining behavior (strict)

```
def mockCpsPersistenceService = Mock(CpsPersistenceService)
def anchors = [new Anchor('a1'), new Anchor('a2')]
mockCpsPersistenceService.getAnchors('someDataspacespace') >> anchors
```

Asserting invocations (strict)

```
1 * mockCpsPersistenceService.createAnchor('someDataspacespace',
                                           'someSchemaSet',
                                           'someAnchorName')
```

Asserting invocations (relaxed)

```
0 * mockCpsPersistenceService.createAnchor(*_)
```

Define and Asserting in one (not recommended)

```
1 * mockCpsPersistenceService.getAnchors(_) >> anchors
```

8 / 15

8

If We Had More Time...



Advanced Groovy and Spock features

- old()
- with()
- GString (Groovy Strings)
- (hamcrest) matchers



Combining with other frameworks

- @SpringBootTest (injection, config)
- @WebMvcTest (REST, contract test)
- @Testcontainers (docker images)
 - PostgreSQLContainer

9 / 15

9

Shameless Plug



CPS: Configuration and Persistence Service



- Developers Landing page
 - <https://wiki.onap.org/display/DW/Configuration+Persistence+Service+Developer%27s+Landing+Page>
- Repos
 - <https://gerrit.onap.org/r/admin/repos/cps>
 - <https://gerrit.onap.org/r/admin/repos/cps/ncmp-dmi-plugin>

10 / 15

10

Tutorials



1. Basic Spock Specification
2. Data Driven Specifications
3. Expecting Exceptions
4. Mocking & Interaction Checking
5. Groovy Magic
6. Demo Test Container (CPS)



wiki.onap.org/pages/viewpage.action?pageId=153519113

11 / 15

11

Results



- All team's projects >97% coverage (SonarQube definition)
- OpenSSF Gold-Standard for Coverage
- High confidence level
- Minimal Integration Tests

Tips:

- Make Unit Test are part of **Definition-of-Done**
- Unit test coverage threshold of build **gate** (on new code)
- Include test in **review** process

SonarQube Statistics

Overall Code	
Coverage	97.1%
Lines to Cover	3,257
Uncovered Lines	63
Line Coverage	98.1%
Conditions to Cover	703
Uncovered Conditions	51
Condition Coverage	92.7%

12 / 15

12

Reference Material



- ❑ <https://www.baeldung.com/groovy-spock>
- ❑ <http://spockframework.org/spock/docs>
- ❑ <https://github.com/spockframework/spock-example>
- ❑ <http://groovy-lang.org/documentation.html>
- ❑ http://spockframework.org/spock/docs/2.3/data_driven_testing.html
- ❑ https://www.jetbrains.com/help/idea/getting-started-with-groovy.html#create_groovy_project



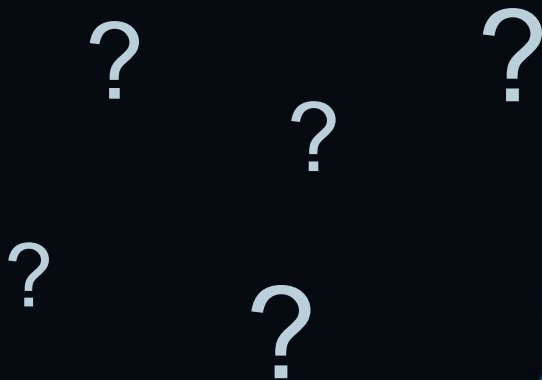
From ONAP CPS Project:

- ❑ <https://gerrit.onap.org/r/admin/repos/cps>
 - ❑ [CpsDataPersistenceServiceIntegrationSpec.groovy](#)

13 / 15

13

Questions & Answers



14

