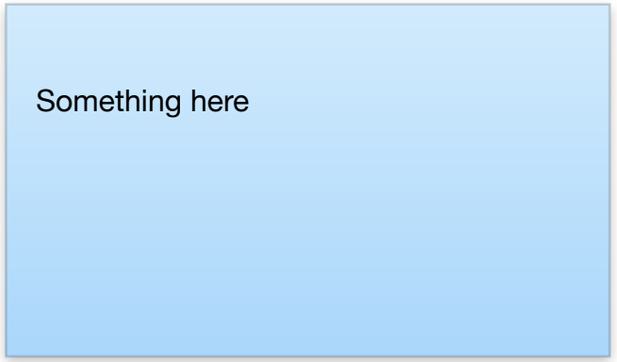
# Spock Unit Testing

By Imran Mir

# Highlights

- Introduction to Unit Testing
- Introduction to Spock
- Basics of Spock
- Demo



# Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individu endently scrutinized for p n.

Discuss Unit Testing with the group.
And clarify doubts if any
How it differs from Integeration and Functional testing

# What do we test ?

What a program is supposed to do

==

What the program actually does

# Motivation ?

- People are not perfect
  - We make errors in design and code
  - And we need to deliver high quality software consistently

# contd..

- Testing is an investment
  - Over the time as tests build, the early investment in writing the test cases pays dividends later as the size of the application grows

# A way of thinking

- Design and code are creative
- Testing is destructive. The primary aim is to break the software
- Most often unit testing is done by the same developer who writes the code
- Needs split personality: when you start testing, become paranoid and malicious

# Surprisingly hard to do

People don't like finding out that they make mistakes

- **Unit Tests** execute a unit of software with the intent of finding bugs and errors
- **Good Unit Tests** have high probability of finding bugs and errors
- **Successful Unit Tests** detect bugs and errors

Unit Testing is the integral part of software development

# Understanding Unit testing

- Unit testing is a method by which individual units of source code are tested to determine if they are fit for use
  - A unit is the smallest testable part of an application
  - Each test case is independent from the others: substitutes like method stubs, mock objects, can be used to assist testing a module in isolation.
  - A unit test provides a strict, written contract that the piece of code must satisfy
  - It tests individual methods or blocks of code without considering for surrounding infrastructure

# Disadvantages of writing Unit tests

- Test cases -written to suit programmer's implementation (not necessarily specification)
- The actual database or external file is never tested directly by TDD
- It is highly reliant on Refactoring and Programmer skills

# Advantages Of Unit Testing

- Facilitates change
- Simplifies Integration
- Serves as documentation
- Evolve designs
- Are index of the Quality of a Software

# Why we need Mocking ?

- To isolate a piece of code under test from its dependencies
- If real objects are impractical to incorporate
- In short mocks simulate the behaviour of real objects

# Stubbing

- We simulate a complex execution by replacing the actual behaviour with a dummy behaviour

# Spock

- Testing and specification framework for Java and Groovy applications
- Beautiful and highly expressive specification language
- One of the big reasons G ning popular

Can you name one other reason for Groovy popularity

# Specification

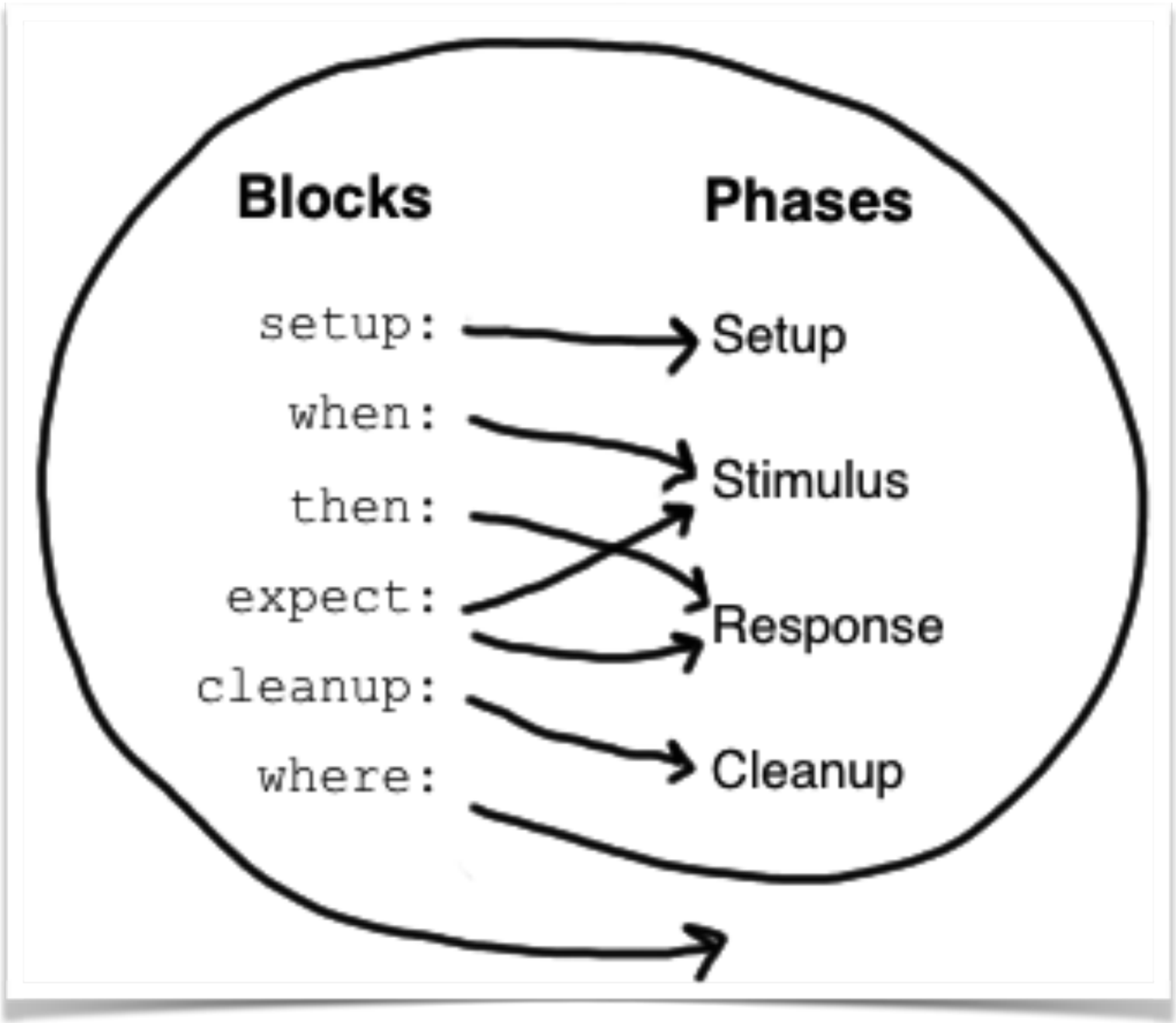
- spock.lang.Specification
- Gives a number of useful methods for writing specifications

# Fixture Methods

- def setup() {} // run before every feature method
- def cleanup() {} // run after every feature method
- def setupSpec() {} // run before the first feature method
- def cleanupSpec() {} // run after the last feature method

# Feature Methods

```
def "pushing an element on the stack"() {

  // blocks go here

}
```


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

# Specification as Documentation

# Living Documentation

- def “our 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

# contd..

- given: “plane water"

  // ...

- when: “lemon juice is added"

  // ...

- then: “you get a lemonade"

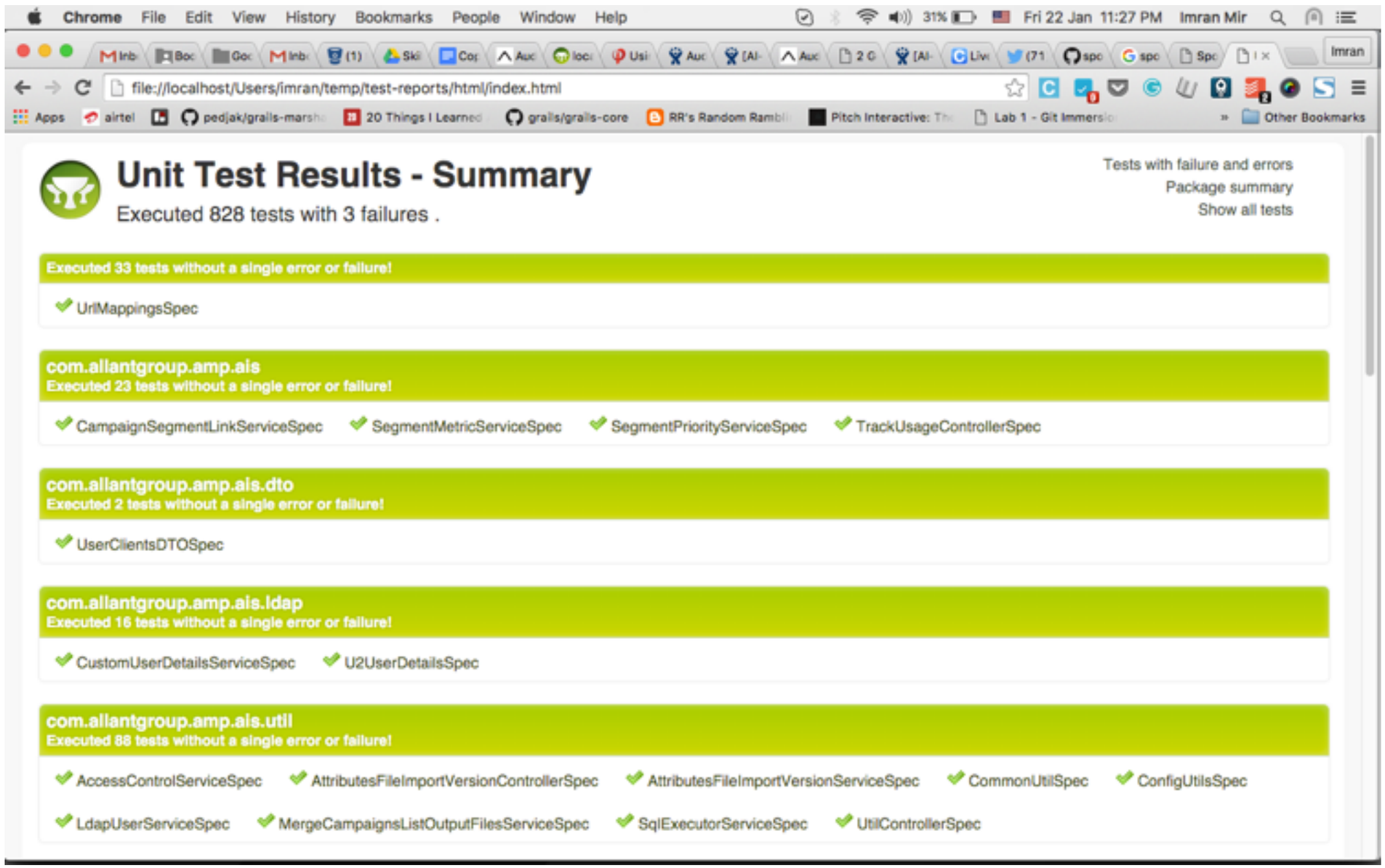
# ..contd.

```
given: "an empty bank account"
// ...

when: "the account is credited $10"
// ...

then: "the account's balance is $10"
// ...
```

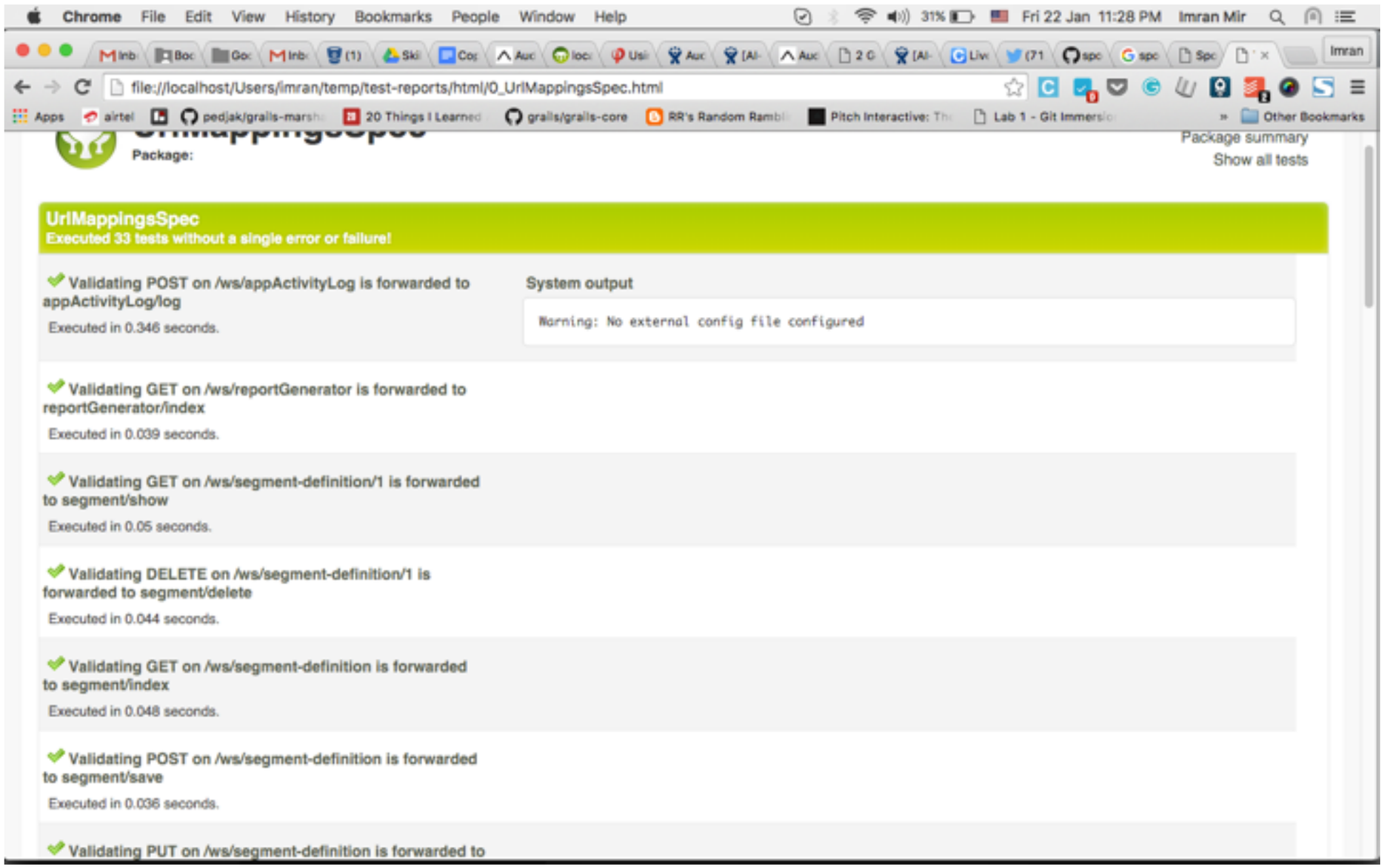
# Enough of theory

```
when:
stack.push(elem)

then:
!stack.empty
stack.size() == 1
stack.peek() == elem
```

# Test Report


Chrome File Edit View History Bookmarks People Window Help
Fri 22 Jan 11:27 PM Imran Mir
file://localhost/Users/imran/temp/test-reports/html/index.html
Unit Test Results - Summary
Executed 828 tests with 3 failures .
Tests with failure and errors
Package summary
Show all tests
Executed 33 tests without a single error or failure!
UrlMappingsSpec
com.allantgroup.amp.ais
Executed 23 tests without a single error or failure!
CampaignSegmentLinkServiceSpec
SegmentMetricServiceSpec
SegmentPriorityServiceSpec
TrackUsageControllerSpec
com.allantgroup.amp.ais.dto
Executed 2 tests without a single error or failure!
UserClientsDTOSpec
com.allantgroup.amp.ais.ldap
Executed 16 tests without a single error or failure!
CustomUserDetailsServiceSpec
U2UserDetailsSpec
com.allantgroup.amp.ais.util
Executed 88 tests without a single error or failure!
AccessControlServiceSpec
AttributesFileImportVersionControllerSpec
AttributesFileImportVersionServiceSpec
CommonUtilSpec
ConfigUtilsSpec
LdapUserServiceSpec
MergeCampaignsListOutputFilesServiceSpec
SqlExecutorServiceSpec
UtilControllerSpec

# ..cont.


Chrome File Edit View History Bookmarks People Window Help
Fri 22 Jan 11:28 PM Imran Mir
file://localhost/Users/imran/temp/test-reports/html/0_UrlMappingsSpec.html
Package:
Package summary
Show all tests
UrlMappingsSpec
Executed 33 tests without a single error or failure!
Validating POST on /ws/appActivityLog is forwarded to appActivityLog/log
Executed in 0.346 seconds.
System output
Warning: No external config file configured
Validating GET on /ws/reportGenerator is forwarded to reportGenerator/index
Executed in 0.039 seconds.
Validating GET on /ws/segment-definition/1 is forwarded to segment/show
Executed in 0.05 seconds.
Validating DELETE on /ws/segment-definition/1 is forwarded to segment/delete
Executed in 0.044 seconds.
Validating GET on /ws/segment-definition is forwarded to segment/index
Executed in 0.048 seconds.
Validating POST on /ws/segment-definition is forwarded to segment/save
Executed in 0.036 seconds.
Validating PUT on /ws/segment-definition is forwarded to

# When something goes wrong

Chrome File Edit View History Bookmarks People Window Help 30% Fri 22 Jan 11:30 PM Imran Mir

file://localhost/Users/imran/temp/test-reports/html/com/allantgroup/amp/ais/ws/subscriber/85_QueryGeneratorSpec.html

Apps airtel pedjak/grails-marsh 20 Things I Learned grails/grails-core RR's Random Ramb Pitch Interactive: Th Lab 1 - Git Immersi Other Bookmarks

test to check the getCriterionQuery() throws exception for #Error:descrption

Executed in 0.072 seconds.

createDemographicSegmentQuery Select generated valid sql When attributes have more than on attribute value

Executed in 0.176 seconds.

Condition not satisfied: query == result | | | | | ((column2 = ('c2Value1') OR column2 = ('c2Value2')) AND (column1 = ('c1Value1') OR column1 = ('c1Value2'))) | false | 8 differences (92% similarity) | ((column(1) = ('c(1)Value1') OR column(1) = ('c(1)Value2')) AND (column(2) = ('c(2)Value1') OR column(2) = ('c(2)Value2'))) | ((column(2) = ('c(2)Value1') OR column(2) = ('c(2)Value2')) AND (column(1) = ('c(1)Value1') OR column(1) = ('c(1)Value2'))) ((column1 = ('c1Value1') OR column1 = ('c1Value2')) AND (column2 = ('c2Value1') OR column2 = ('c2Value2')))

```
junit.framework.AssertionFailedError: Condition not satisfied:

query == result
|     |  |
|     |  ((column2 = ('c2Value1') OR column2 = ('c2Value2')) AND (column1 = ('c1Value1') OR
column1 = ('c1Value2')))
|     false
|     8 differences (92% similarity)
|     ((column(1) = ('c(1)Value1') OR column(1) = ('c(1)Value2')) AND (column(2) = ('c(2)Value1')
OR column(2) = ('c(2)Value2')))
|     ((column(2) = ('c(2)Value1') OR column(2) = ('c(2)Value2')) AND (column(1) = ('c(1)Value1')
OR column(1) = ('c(1)Value2')))
((column1 = ('c1Value1') OR column1 = ('c1Value2')) AND (column2 = ('c2Value1') OR column2 =
('c2Value2')))

        at com.allantgroup.amp.ais.ws.subscriber.QueryGeneratorSpec.createDemographicSegmentQuery
Select generated valid sql #description(QueryGeneratorSpec.groovy:192)
```

System output

```
|((column1 = ('c1Value1') OR column1 = ('c1Value2')) AND (column2 = ('c2Value1') OR column2 =
('c2Value2')))|
```

createDemographicSegmentQuery Select generated valid sql When criteria has just one attribute

Executed in 0.096 seconds.

System output

```
|((column1 = ('c1Value1') OR column1 = ('c1Value2')))|
```

# Let us try to read the failures

```
Condition not satisfied:

stack.size() == 2
|      |       |
|      1       false
[push me]
```

# Using Data Tables

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

        where:
        a | b | c
        1 | 3 | 3
        7 | 4 | 4
        0 | 0 | 0
    }
}
```

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

  where:
  a << [3, 5, 9]
  b << [7, 4, 9]
  c << [7, 5, 9]
}
```

```
when:
stack.pop()

then:
thrown(EmptyStackException)
stack.empty
```

Assume stack is empty

Type of Exception

# Spot the difference here

```
def "HashMap accepts null key"() {
  setup:
  def map = new HashMap()

  when:
  map.put(null, "elem")

  then:
  notThrown(NullPointerException)
}
```

# A special mention for @Unroll

```
@Unroll
def "maximum of two numbers"() { ... }
```

```
maximum of two numbers[0]   PASSED
maximum of two numbers[1]   FAILED
```

# Few Extensions

- @Ignore
- @IgnoreRest

# Mocks

- Mocks have not behaviour
- Calling methods on them is allowed but has no effect other than returning the default value for the method's return type (false, 0, or null)
- A mock object is only equal to itself, has a unique hash code, and a string representation that includes the name of the type it represents
- This default behavior is overridable by stubbing the methods

# Mock Example

```
class Transaction {
   def emailService
   void cancelSale(Product product, User user) {
 String productName = product.name
 user.balance += (product.price - calculateDiscount(product, user))
 user.cancelPurchase(product)
 emailService.sendCancellationEmail(user, productName)
}
```

```
  def "Email is send when a sale is cancelled"(){
 given: "A product"
 Product product = new Product(name: 'p1', price: 100)

 and: "A customer"
 User user = new User()

 and: "A sale"
 Transaction transaction = new Transaction()

 and: "An email service mock"
 def emailService = Mock(EmailService)
 transaction.emailService = emailService


 when: "Cancel save is called"
 transaction.cancelSale(product, user)

 then: "Validate email service is called"
 1 * emailService.sendCancellationEmail(user, _ as String)
}
```

# Stub

- Stubbing is the act of making collaborators respond to method calls in a certain way
- When stubbing a method, you don’t care if and how many times the method is going to be called; you just want it to return some value, or perform some side effect

# Stub example

```
Boolean encyryptPassword(String pwd) {
    String encryptedPassword = passwordEncrypterService.encrypt(pwd)
    return encryptedPassword
 }
```

```
def "Valid password is encrypted" ( ) {
    given: "A user"
    User user = new User()

    and: "A passwordEncrypterMock"
    def passwordEncrypterService = Mock(PasswordEncrypterService)
    passwordEncrypterService.encrypt(_ as String) >>  "drowssap"
    user.passwordEncrypterService = passwordEncrypterService
    passwordEncrypterService.encrypt(_ as String) >> "drowssap"

    when: "encryptPassword is called"
    String encryptedPwd = user.encyryptPassword("password")

    then:
    encryptedPwd == "drowssap"
}
```

# Understanding Stub syntax

```
subscriber.receive(_) >> "ok"
```

```
subscriber.receive(_) >> "ok"
|           |        |      |
|           |        |      response generator
|           |        argument constraint
|           method constraint
target constraint
```

# ..cont.

```
subscriber.receive("message1") >> "ok"
subscriber.receive("message2") >> "fail"
```

```
subscriber.receive(_) >>> ["ok", "error", "error", "ok"]
```

# Combining Mocking and Stubbing

```
1 * subscriber.receive("message1") >> "ok"
1 * subscriber.receive("message2") >> "fail"
```

# Spy

- A spy is always based on a real object
- Method calls on a spy are automatically delegated to the real object
- After creating a spy, you can listen in on the conversation between the caller and the real object underlying the spy
- Think twice before using this feature. It might be better to change the design of the code under specification

```
List<String>getSortedInterestedInCategories(){
    List<String> interestedInCategories = getInterestedInCategories()
    interestedInCategories.sort()
}

List<String>getInterestedInCategories(){    //assumed to be a very complex method
    sleep(10000)
}
```

```
def "SortedInterestedInCategories can be retrieved"(){
    given: "A user spy"
    def user = Spy(User)
    user.getInterestedInCategories() >> ["z","x","a","s"]

    when:
    def sortedCategories = user.getSortedInterestedInCategories()

    then:
    sortedCategories == ["a","s","x","z"]
}
```

Questions ?