INTRO TO SPOCK AND GEB

Craig Atkinson | Chief Technologist | Object Partners

OBJECT PARTNERS

ABOUT ME

- Craig Atkinson
- Chief Technologist, Object Partners (OPI)
- Using Spock & Geb past 4 years
- Very minor Geb contributor

AGENDA

- Intro to Spock
- Data-driven testing
- Mocks for unit testing
- Intro to Geb
- Combining Geb and Spock

WHAT IS SPOCK?

Behavior-style test framework in Groovy with support for easy data-driven testing

BEHAVIOR-STYLE TESTING

Test cases separated into three main sections

- given (setup)
- when (execute method under test)
- then (verify results)

TEST CLASS NAME

Test class name ends in **Spec** or **Specification** and class extends **spock.lang.Specification**

```
import spock.lang.Specification

class BankAccountSpec extends Specification {

}
```

TEST CASE NAME

Test case method names can be descriptive sentences

```
def "when depositing 10 dollars into new account then balance should be 10"() {

}
```

Much more descriptive than

```
void depositTestCase1() {

}
```

TEST CASE BODY

```
class BankAccountSpec extends Specification {

  def "when depositing 10 dollars in new account then balance should be 10"() {
    given:
    BankAccount bankAccount = new BankAccount()

    when:
    bankAccount.deposit(10)

    then:
    assert bankAccount.balance == 10
  }
}
```

ONE-LINE TEST

```
def "new account should have 0 balance"() {
  expect:
  assert new BankAccount().balance == 0
}
```

SETUP METHOD

Run code before each test method

```
void setup() {
  // Setup code goes here
}
```

CLEANUP METHOD

Run code after each test method

```
void cleanup() {
  // Cleanup code goes here
}
```

SETUP / CLEANUP PER TEST CLASS

```
void setupSpec() {
  // Runs once before test methods in class
}

void cleanupSpec() {
  // Runs once after test methods in class
}
```

SETUP / CLEANUP AND INHERITANCE

- Might have a hierarchy of test classes for DRY common test code
- Spock runs the base class **setup** first
- Then goes down the inheritance chain
- **cleanup** is the reverse, starting at test class then going up to base class

DATA-DRIVEN TESTS

Run **same test body** with **multiple sets** of test **inputs** and expected **outputs**

WHERE BLOCK

```
where:
input1 | input2 || output
4      | 6      || 5
12     | 18     || 15
20     | 14     || 17
```

TEST WITH WHERE BLOCK

```
def 'deposits should increase balance'() {
  given:
  BankAccount bankAccount = new BankAccount()

  when:
  bankAccount.deposit(amount)

  then:
  assert bankAccount.balance == expectedBalance

  where:
  amount || expectedBalance
  10     || 10
  25     || 25
  50     || 50
}
```

IDENTIFY FAILURES

- By default, each line in **where:** block combined into single test
- Makes it slower to identify which iteration failed
- Spock provides **@Unroll** to separate out test cases
- Also include inputs and outputs from **where:** block in tests results

@UNROLL

```
import spock.lang.Unroll

@Unroll
def 'depositing #amount should increase balance to #expectedBalance'() {
  given:
  BankAccount bankAccount = new BankAccount()

  when:
  bankAccount.deposit(amount)

  then:
  assert bankAccount.balance == expectedBalance

  where:
  amount || expectedBalance
  10     || 10
  25     || 25
  50     || 50
}
```

GROOVY POWER ASSERT

Prints out descriptive and useful failure message when an assertion fails

```
def 'x plus y equals z'() {
  when:
  int x = 4
  int y = 5
  int z = 10

  then:
  assert x + y == z
}
```

DESCRIPTIVE FAILURE MESSAGE

```
Condition not satisfied:

x + y == z
| | | |  |
4 9 5 |  10
      false
```

MOCKING OBJECTS

- Spock has powerful built-in object mocking capabilities
- Very helpful for mocking interactions with other classes

CLASS TO TEST

```
class BankAccount {
  AuditService auditService

  BigDecimal balance

  BankAccount() {
    auditService = new AuditService()

    balance = 0
  }

  void deposit(BigDecimal amount) {
    balance += amount

    auditService.record('deposit', amount)
  }
}
```

TEST USING MOCK OBJECT

```
class BankAccountSpec extends Specification {
  def "should record audit event when making deposit"() {
    given:
    BankAccount bankAccount = new BankAccount()

    // Create Spock mock object
    AuditService auditService = Mock()

    // Use mock object in class-under-test
    bankAccount.auditService = auditService

    when:
    bankAccount.deposit(100)

    then:
    1 * auditService.record('deposit', 100)

    and:
    assert bankAccount.balance == 100
  }
}
```

MOCK RETURN VALUE

```
then:
1 * accountService.calculateBalance(account) >> 20
```

FLEXIBLE ARGUMENT MATCHING

Can use closure to match method arguments

```
1 * userService.sendWelcomeEmail({ User user ->
  user.email == 'jim@test.com' && user.name == 'Jim Smith'
})
```

THROW EXCEPTIONS

```
1 * userService.sendWelcomeEmail(user) >> {
  throw new IllegalStateException()
}
```

USE ARGUMENTS IN RESULT

```
1 * userService.createUser(_, _) >> { String email, String name ->
  new User(email: email, name: name)
}
```

MORE MOCK OBJECT CAPABILITIES

- Spock interaction docs
- One-page Spock mock cheatsheet
- GitHub project with Spock mock examples

HELPFUL ANNOTATIONS

- **@Ignore** to skip test method or whole class
- Fastest way to fix a failing build

RUN SINGLE TEST METHOD

```
import spock.lang.IgnoreRest

@IgnoreRest
def 'only run this test method'() {
}
```

SPOCK SUMMARY

- Write readable tests with given/when/then syntax
- Avoid copy-paste with data-driven testing
- Write focused unit tests with flexible built-in mock objects

GEB FUNCTIONAL TESTING

WHAT IS FUNCTIONAL TESTING?

- Interact with the full system the way a user does
- For web applications, using a browser

WHY FUNCTIONAL TESTS?

- Confidence that application really works for your users
- Verify all parts of application work correctly together
- Front-end code, back-end code, database, messaging, caching, etc.

EXAMPLE FUNCTIONAL TEST

Verify that user can sign up

- User goes to sign up form
- Fills in all required fields on form
- Clicks submit button
- Verify welcome page displayed
- Verify user data saved to database

GEB INTRODUCTION

- Groovy wrapper around Selenium testing library
- jQuery/CSS-style selectors for finding elements on page
- Powerful built-in page object support
- Write tests with Spock, JUnit, or TestNG
- Run tests using any browser Selenium supports
- Created by Luke Daley, currently lead by Marcin Erdmann

SIMPLE EXAMPLE TEST

Search for Geb homepage using Google

```
go "http://www.google.com"

$("input", name: "q").value("Geb")
$("button", name: "btnG").click()

waitFor { $("#search").displayed }

assert $("#search").text().contains("gebish.org")
```

DEMO

TEST DESIGN

- Tests with embedded page structure (HTML) are less readable
- Multiple places to update if page structure changes
- Can we abstract page structure out of tests?

PAGE OBJECT PATTERN

- Abstract page-specific details into helper classes
- Page objects re-used across tests
- Single point of maintenance
- Tests easier to read

EXAMPLE TEST, REVISITED

```
to GoogleHomePage

searchBox = "Geb"
searchButton.click()

assert searchResults.text().contains("gebish.org")
```

PAGE OBJECT

```
class GoogleHomePage extends geb.Page {
  static url = "http://www.google.com"

  static content = {
    searchBox { $("input", name: "q") }
    searchButton { $("button", name: "btnG") }
    searchResults { $("#search") }
  }
}
```

WHAT'S IN A GEB PAGE OBJECT?

- Elements on page and how to find them
- URL to go directly to page
- How to verify currently on page
- Helper methods to simplify page interaction

CONTENT BLOCK

- Defines elements on the page that tests will interact with
- Includes **selectors** that tell Geb how to find the element on the page

COMMON SELECTORS

```
static content = {
  depositButtonById    { $("#deposit-button") }

  depositButtonByClass { $(".deposit-button") }

  depositButtonByName  { $("input", name: "deposit") }

  depositButtonByText  { $("input", text: "Deposit") }
}
```

MANY MORE SELECTORS

- **Full list of all types of selectors in Geb manual**

GO DIRECTLY TO PAGE

- Define **url** in Page Object
- Use **to(PageClass)** method in test
- Speed up test by skipping preliminary pages

EXAMPLE

```
class LoginPage extends Page {
  static url = "/app/login"
}
```

```
// In test
LoginPage loginPage = to(LoginPage)
```

VERIFY ON EXPECTED PAGE

- Geb can verify destination page when page changes
- Calling **to()** method, clicking link, etc.
- Geb uses **at** checker in page object

DEFINE 'AT' CHECKER

```
class LoginPage extends Page {
  static at = { title == 'Login to my app' }

  static url = "/app/login"
}
```

DYNAMIC CONTENT

- Content changes without a page reload (ajax, websockets, etc.)
- Wait for element displayed, content values, etc.
- **waitFor** method available in tests and page objects
- Waits until closure returns true

WAITFOR EXAMPLES

```
waitFor {
  $("div.alert").displayed
}

assert $("div.alert").text() == "Error creating user"
```

```
waitFor {
  $("div.message").text() == "Update successful"
}
```

CONFIGURATION

- Waiting, reporting, browsers, etc.
- Configuration read from GebConfig.groovy file
- **List of options in Geb manual**

CROSS-BROWSER TESTING

Can use same test code across different browsers

CONFIGURING BROWSER IN GEB

- Browser Selenium driver dependency
- Additional OS-specific executable
- Section in GebConfig.groovy

SELENIUM DRIVER DEPENDENCIES

```
def seleniumVersion = "2.53.1"

testCompile "org.seleniumhq.selenium:selenium-chrome-driver:${seleniumVersion}"
testCompile "org.seleniumhq.selenium:selenium-firefox-driver:${seleniumVersion}"
testCompile "org.seleniumhq.selenium:selenium-ie-driver:${seleniumVersion}"
```

SWITCHING BROWSERS

GebConfig.groovy

```
import org.openqa.selenium.chrome.ChromeDriver
import org.openqa.selenium.firefox.FirefoxDriver

// Default browser
driver = { new FirefoxDriver() }

environments {
  chrome {
    // Assumes OS-specific library already downloaded on machine running tests
    System.setProperty('webdriver.chrome.driver', '/path/to/chromedriver')

    driver = { new ChromeDriver() }
  }
}
```

BROWSER REQUIREMENTS

- Firefox <= 47: Selenium driver dependency only
- Firefox >= 48: Also requires OS-specific library
- Chrome & IE: Selenium driver and OS-specific library

OS-DRIVER AUTOMATIC DOWNLOAD

- Could download/manage these OS-specific executables by hand, or ...
- Use **WebdriverManager library** by Boni Garcia

WEBDRIVERMANAGER DEPENDENCY

```
testCompile("io.github.bonigarcia:webdrivermanager:1.4.1")
```

CHROME GEBCONFIG EXAMPLE

```
import io.github.bonigarcia.wdm.ChromeDriverManager
import org.openqa.selenium.chrome.ChromeDriver

environments {
  chrome {
    // Downloads driver for the current OS and does all necessary configuration
    ChromeDriverManager.getInstance().setup()

    driver = { new ChromeDriver() }
  }
}
```

COMBINING GEB WITH SPOCK

- **given / when / then** clearly separate test case sections
- Descriptive, full-sentence test names
- Data-driven testing with **where:** block

EXAMPLE GEB / SPOCK TEST

```
class AccountDepositGebSpec extends geb.spock.GebReportingSpec {
  def "should deposit amount into bank account"() {
    given:
    AccountPage accountPage = to(AccountPage)

    DepositPage depositPage = accountPage.clickDepositLink()

    when:
    depositPage.depositAmount(100)

    then:
    waitFor { depositPage.successMessage.displayed }
  }
}
```

TEST SUPERCLASS

- Extend **GebReportingSpec**
- Automatically takes screenshots and HTML dumps
- Very helpful for debugging test failures, especially in CI

DATA-DRIVEN GEB / SPOCK TEST

```
@Unroll
def "should get error message when logging in with invalid user #username"() {
  when: "logging in as invalid user"
  LoginPage loginPage = loginAsUser(username)

  then: "should show error message"
  assert loginPage.errorMessage == expectedErrorMessage

  where:
  username       || expectedErrorMessage
  'disabledUser' || 'Sorry, your account is disabled'
  'lockedUser'   || 'Sorry, your account is locked'
  'missingUser'  || 'Sorry, we could not find that account'
}
```

WRAPUP

SPOCK RESOURCES

- Spock documentation
- Spock web console
- Official Spock example project
- Spock mock examples
- Spock User mailing list

GEB RESOURCES

- Geb manual
- Geb User mailing list
- Official Geb/Grails example
- Official Geb/Gradle example
- Geb/Spring Boot example
- Geb/Grails 3 example
- Geb/Grails 2 example
- Geb/Gradle example

GEB TIPS & TRICKS PRESENTATION

Law School 235 after lunch

Q&A

- @craigatk1
- craig.atkinson@objectpartners.com