Stuttgarter Test-Tage
21. & 22. März 2013

Spock und Geb (WebDriver)

Wie können freie Werkzeuge zum strukturierten Testen von Web-Applicationen eingesetzt werden?

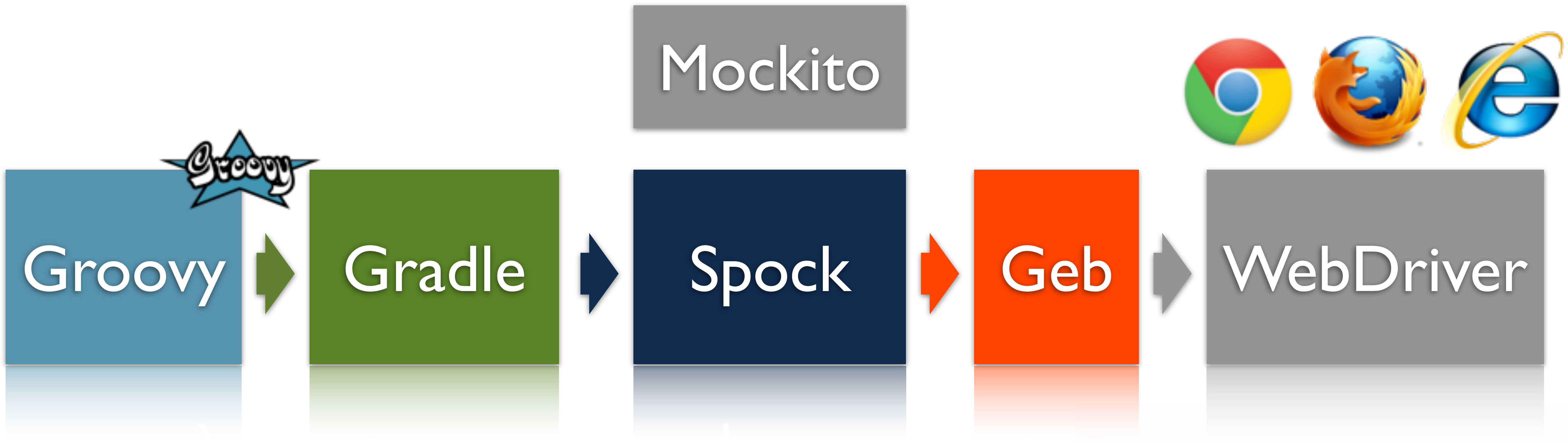
Christian Baranowski

Willkommen

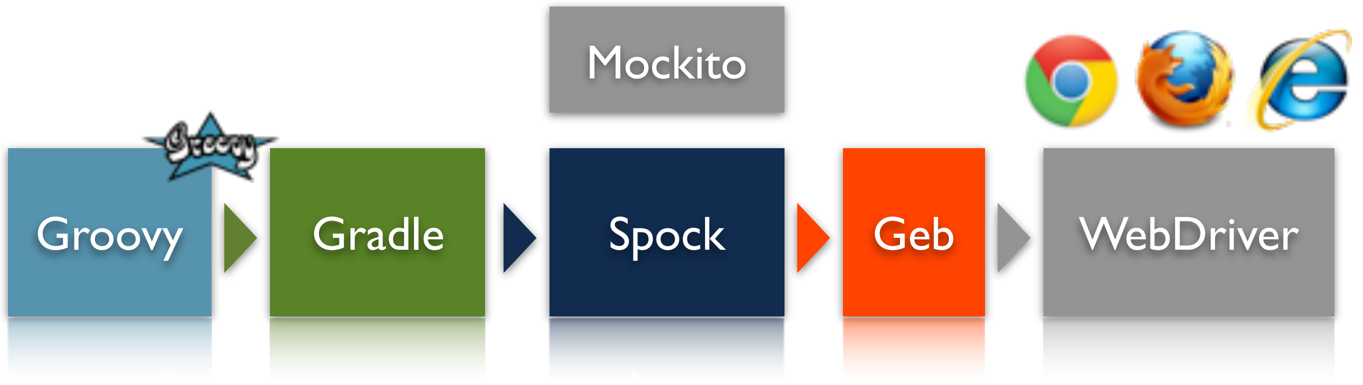
- Christian Baranowski
- Software Qualitätssicherung @ SEITENBAU GmbH Konstanz (DE)
    - Custom Software Solutions
    - E-Government Solutions
    - Identity Management and SSO Solutions
    - www.seitenbau.de
- Vorstand OSGi Users' Forum Germany
    - Co-lead (mit Jochen Hiller) German Enterprise Working Group.
    - OSGi Code Camp

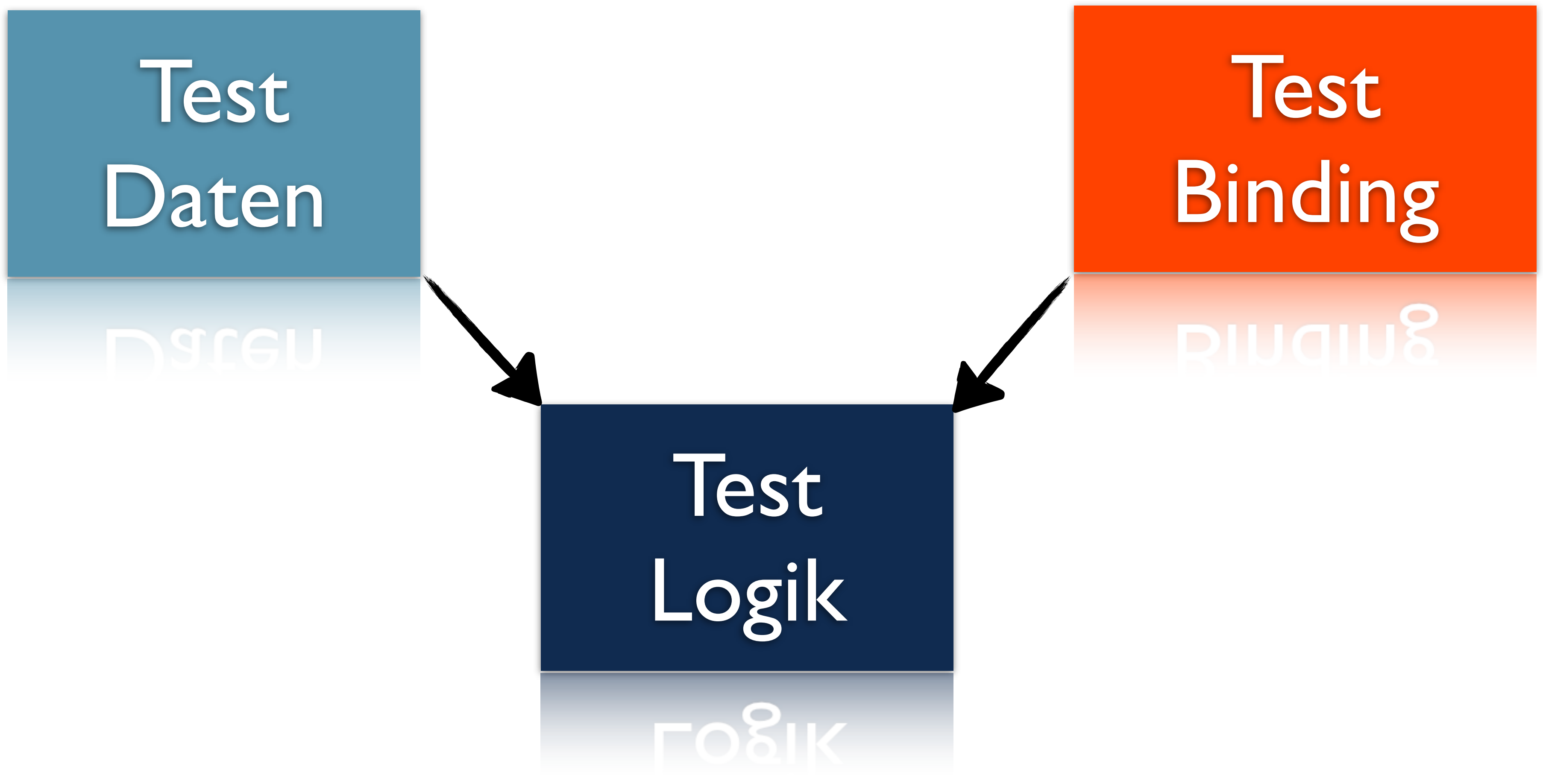
SEITENBAU

Werkzeuge



Test Design


Mockito
Groovy
Gradle
Spock
Geb
WebDriver


Test Daten
Test Binding
Test Logik

Warum Spock?

- Sehr einfaches **BDD** Werkzeug für die JVM, kann schnell erlernt werden
- Biete eine ausdrucksstarke DSL zur Spezifikation von Tests, insbesondere für Parametrisierte Tests (Data Driven Tests)
- Spock kann sowohl für Unit- wie Systemtests genutzt werden
- JUnit Kompatibel - Zur Ausführung wird JUnit genutzt, Integration in IDEs, Build-Tools (Ant, Maven, Gradle...) und CI (Jenkins)
- Spock vereint die besten Features aus bewährten Tools wie JUnit, JMock und RSpec

Spock Given When Then

```
def "spock test with given when then block"() {
   given: "Array with one element"
       def data = ["Some Data"]
   when: "Pop a element from the array"
       data.pop()
   then: "Size of the array is zero"
       data.size() == 0
}
```

Blocks

given:	Vorbedingung, Data Fixtures, Setup
when:	Zustand SUT wird verändert
then:	Assertions, Prüfung des neuen Zustands
expect:	Kurzvariante für when & then
and:	Unterteilung in weitere Blöcke
setup:	Alias für den given Block
cleanup:	Cleanup innerhalb eines Tests

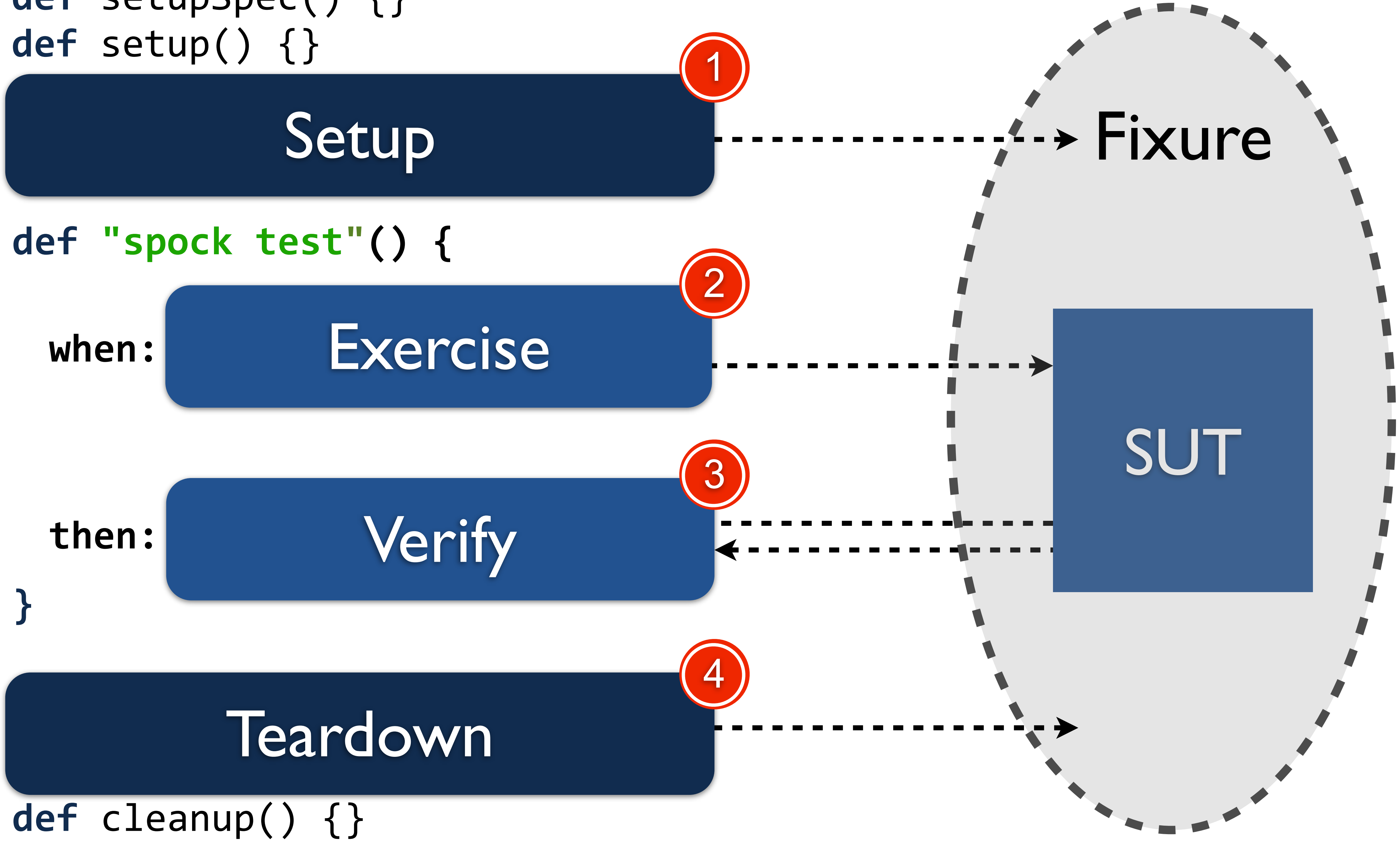
Blocks

```
def "spock test with some blocks"() {
        given:
            def basar = mock(Basar)
            when(basar.getTotal()).thenReturn(100L)
        when:
            def total = basar.getTotal()
        then:
            total == 100L
        and:
            def user = basar.findUserWithId(100)
        then:
            user == null
        cleanup:
            basar = null
}
```

Lifecycle

```
class LifecycleSpec extends Specification {

    def setupSpec() {  println "01 - setup Spec" }
    def setup() { println "02 - setup" }

    def "simple spock test"() {
        expect:
            def data = []
            data == []
    }
    def cleanup() { println "04 - cleanup" }
    def cleanupSpec() { println "04 - cleanup Spec" }
}
```

Vier Phasen Test (Four-Phase Test)



```
def setupSpec() {}
def setup() {}

def "spock test"() {
  when:

  then:
}

def cleanup() {}
def cleanupSpec() {}
```

Power Assertion

```groovy
def christian = new User(id: 1, name: "Christian")
def martin = new User(id:  1, name: "Martin")
assert christian.name == martin.name
```

```
christian.name == martin.name
|         |    |  |      |
|         |    |  |      Martin
|         |    |  User{id=1, basarNumber='null', name='Martin', email='null', lastname='null'}
|         |    false
|         |    5 differences (44% similarity)
|         |    (Ch)r(is)ti(a)n
|         |    (Ma)r(--)ti(-)n
|         Christian
User{id=1, basarNumber='null', name='Christian', email='null', lastname='null'}
```

Helper Method

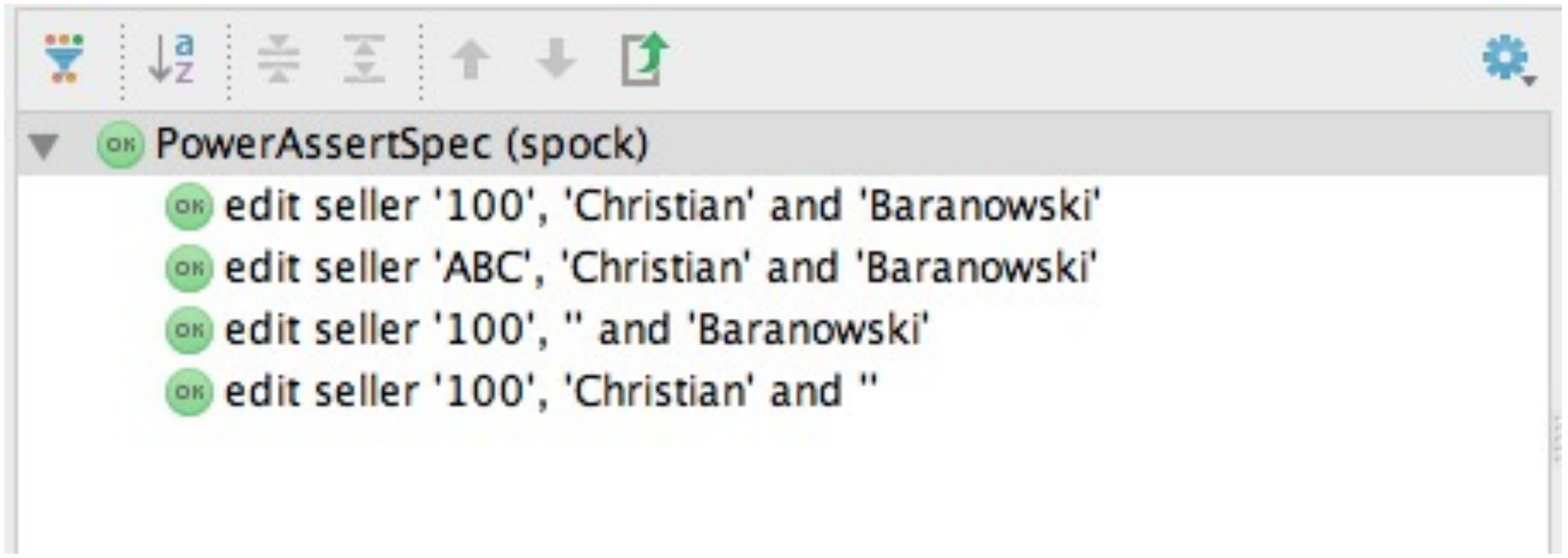
```
def "use helper method in spock test"() {
  when:
     def user = new User(name: "Christian", lastname: "Baranowski")
  then:
     referentMatches(user)
}

def referentMatches(user) {
  assert user.name == "Christian"
  assert user.lastname == "Baranowski"
}
```

Parameterized Test

```
@Unroll
def "edit seller '#basarNumber', '#name' and '#lastname'"() {
   when:
     def updatedUser = updateUser(basarNumber, name, lastname)
   then:
     updatedUser.basarNumber == basarNumber
     updatedUser.name == name
     updatedUser.lastname == lastname
   where:
          basarNumber  | name          |   lastname
          "100"        | "Christian"   |   "Baranowski"
          "ABC"        | "Christian"   |   "Baranowski"
          "100"        | ""            |   "Baranowski"
          "100"        | "Christian"   |   ""
}
```

Parameterized Test

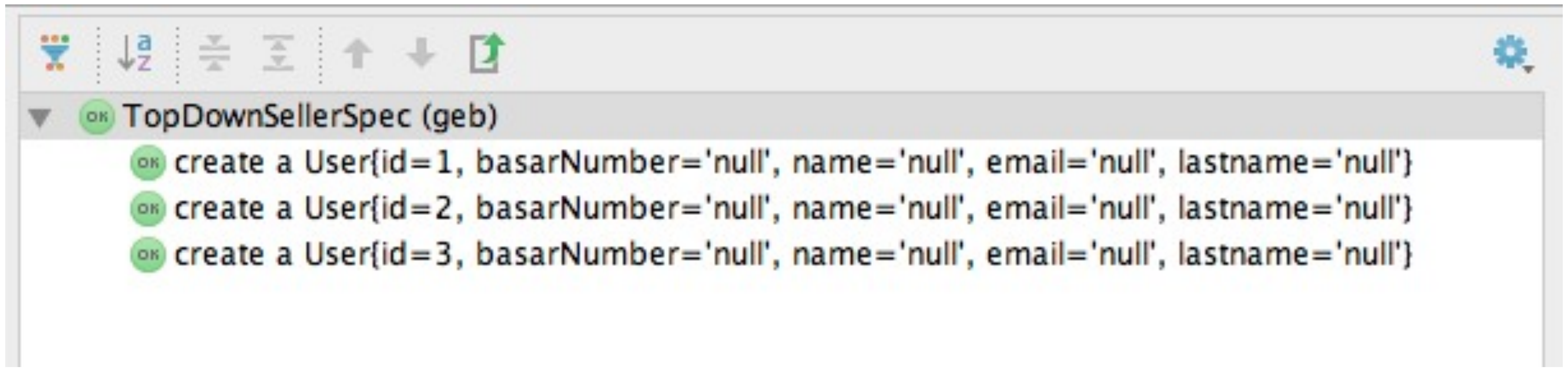
```
@Unroll
def "edit seller '#basarNumber', '#name' and '#lastname'"() {
  ...
  where:
        basarNumber   | name          |   lastname
        "100"         | "Christian"   |   "Baranowski"
        "ABC"         | "Christian"   |   "Baranowski"
        "100"         | ""            |   "Baranowski"
        "100"         | "Christian"   |   ""
}
```



```
PowerAssertSpec (spock)
    edit seller '100', 'Christian' and 'Baranowski'
    edit seller 'ABC', 'Christian' and 'Baranowski'
    edit seller '100', '' and 'Baranowski'
    edit seller '100', 'Christian' and ''
```

Parameterized Test

```
@Unroll
def "create a #user"() {
   when:
     basar.saveUser(user)
   then:
     basar.findUserWithId(user.id) == user
   where:
     user << [new User(id: 1), new User(id: 2), new User(id: 3)]
}
```



```
TopDownSellerSpec (geb)
    create a User{id=1, basarNumber='null', name='null', email='null', lastname='null'}
    create a User{id=2, basarNumber='null', name='null', email='null', lastname='null'}
    create a User{id=3, basarNumber='null', name='null', email='null', lastname='null'}
```

Geb (pronounced "jeb")

Warum Geb?

- Geb bietet eine Abstraktion und Vereinfachung der WebDriver API für Groovy
- Dazu werden die dyamischen Sprachfunktionen von Groovy genutzt.
- JQuery like API für Selenium WebDriver
- Geb bietet einen Mechanismus zur Seitenabstraktion
  ⇒ lesbare Oberflächentests
- Einfacher waitFor{ } mir Groovy Closure für dynamische Web-Anwendungen
- Groovy GString bietet einfache JavaScript Integration in Tests

Geb „JQuery like API“

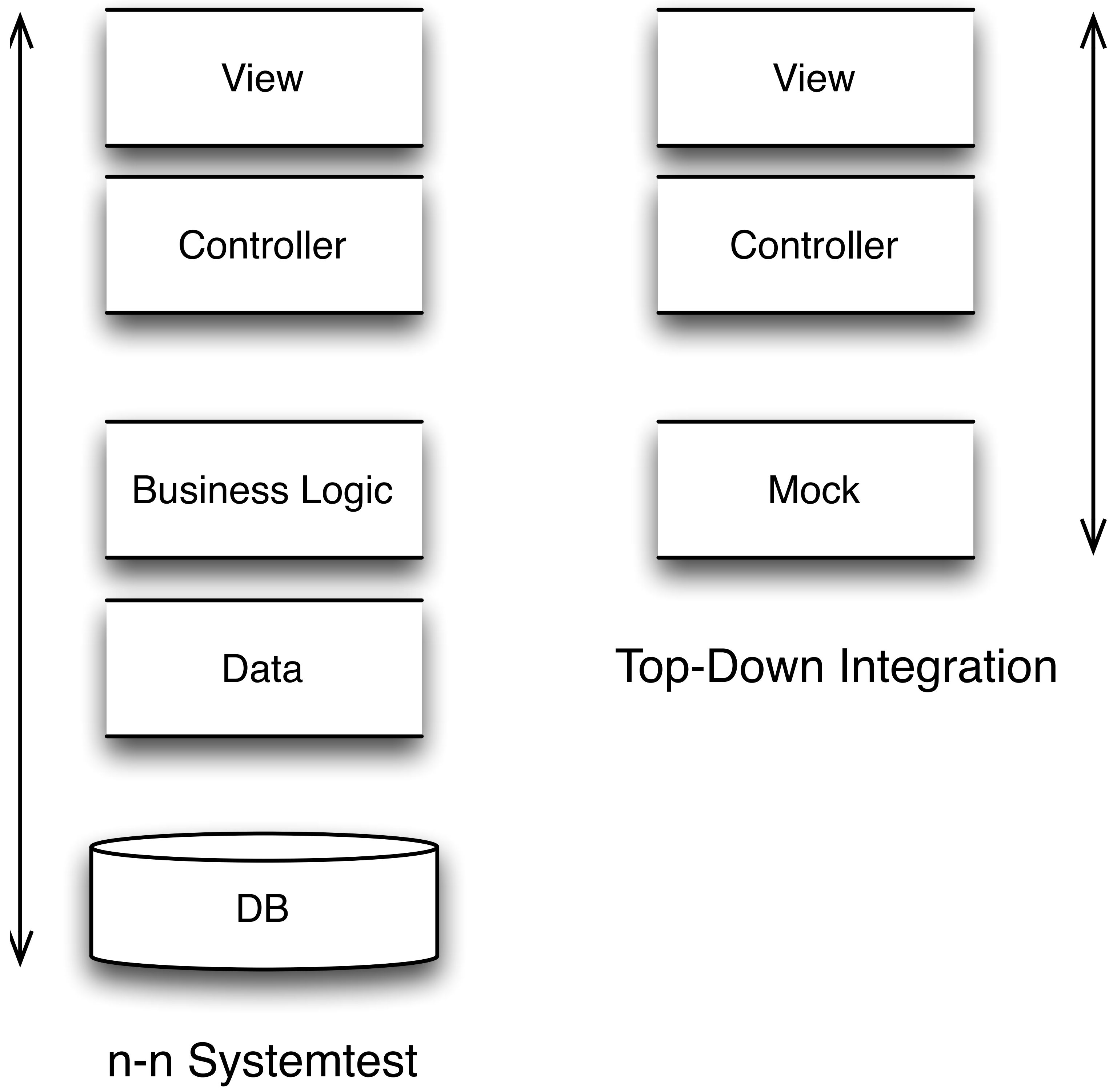
```
// CSS 3 selectors
$("div.some-class p:first[title='something']")

// Find via index and/or attribute matching
$("h1", 2, class: "heading")
$("p", name: "description")
$("ul.things li", 2)

// 'text' is special attribute for the element text content
$("h1", text: "All about Geb")

// Use builtin matchers and regular expressions
$("p", text: contains("Geb"))
$("input", value: ~/\d{3,}-\d{3,}-\d{3,}/)

// Chaining
$("div").find(".b")
$("div").filter(".c").parents()
$("p.c").siblings()
```

Page Objects

```
class BasarPage extends Page {

    static url = "static/basar.html"

    static at = { title == "Basar" }

    static content = {
        basarForm { $("form") }
        addButton { $("#addCartItem") }
    }
}
```

```
to BasarPage
at BasarPage
basarForm.with {
  basarNumber = number
  price = preis
}
addButton.click()
```

Top-Down Integration



Top-Down Integration

```groovy
@Autowired
Basar basarMock

def "create a new seller"() {
        given:
            def user = [basarNumber: "100", name: "Christian"]
            when(basarMock.findAllUsers()).thenReturn([])
        when:
            go "$basarUrl/static/sellers.html"
            waitFor { $("#newUser") }
            $("#newUser").click()
            waitFor { $("#basarNumber") }
            $("#basarNumber").value(user.basarNumber)
            $("#name").value(user.name)
            $("#saveUser").click()
            waitFor { $("#successfullCreated") }
        then:
            ArgumentCaptor<User> userArgumentCaptor = ArgumentCaptor.forClass(User)
            verify(basarMock).saveUser(userArgumentCaptor.capture())
        and:
            User newUser = userArgumentCaptor.value
            newUser.basarNumber == user.basarNumber
            newUser.name == user.name
}
```

JavaScript Support in Geb

```
def users = js.exec('''
                       var users = []
                       var rows = $("#usersBody tr")
                       rows.each(function() {
                            var cells = $(this).children().not(".rightCell")
                            var user = {
                                 basarNumber: $(cells[0]).text(),
                                 vorname:     $(cells[1]).text(),
                                 nachname:    $(cells[2]).text(),
                                 email:       $(cells[3]).text()
                            }
                            users.push(user)
                       })
                       return users
''')
then:
  users == [[basarNumber:"100", vorname: "Christian", nachname: "", email: ""],
            [basarNumber:"101", vorname: "Martin",    nachname: "", email: ""]]
```

Firebug Support

```
def firebug = getClass()
    .getResource("/firebug-1.11.2-fx.xpi")
def profile = new FirefoxProfile();
profile.addExtension(new File(firebug.file));
browser.driver = new FirefoxDriver(profile)
```

Q&A
Twitter @tux2323

Let‘s write some Groovy Spock Geb ...