Stuttgarter Test-Tage
21. & 22. März 2013

Spock und Geb (WebDriver)

Wie können freie Werkzeuge zum strukturierten Testen von Web-Applications 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
- Kontakt
    - Christian.Baranowski@htwg-konstanz.de

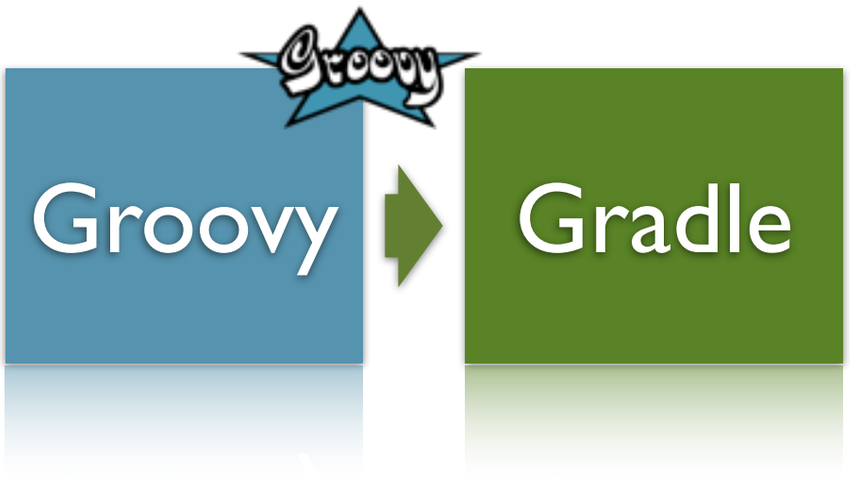
SEITENBAU

Werkzeuge

Werkzeuge

Groovy

Werkzeuge



Werkzeuge

Mockito

Groovy → Gradle → Spock

Werkzeuge

Werkzeuge

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)
- Powerd by Groovy, Tests lassen sich elegant und kurz formulieren (Groovy 99% Source-Kompatibel zu Java)
- 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)
- Kann einfach erweitert und an eigene Bedürfnisse angepasst werden
- 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

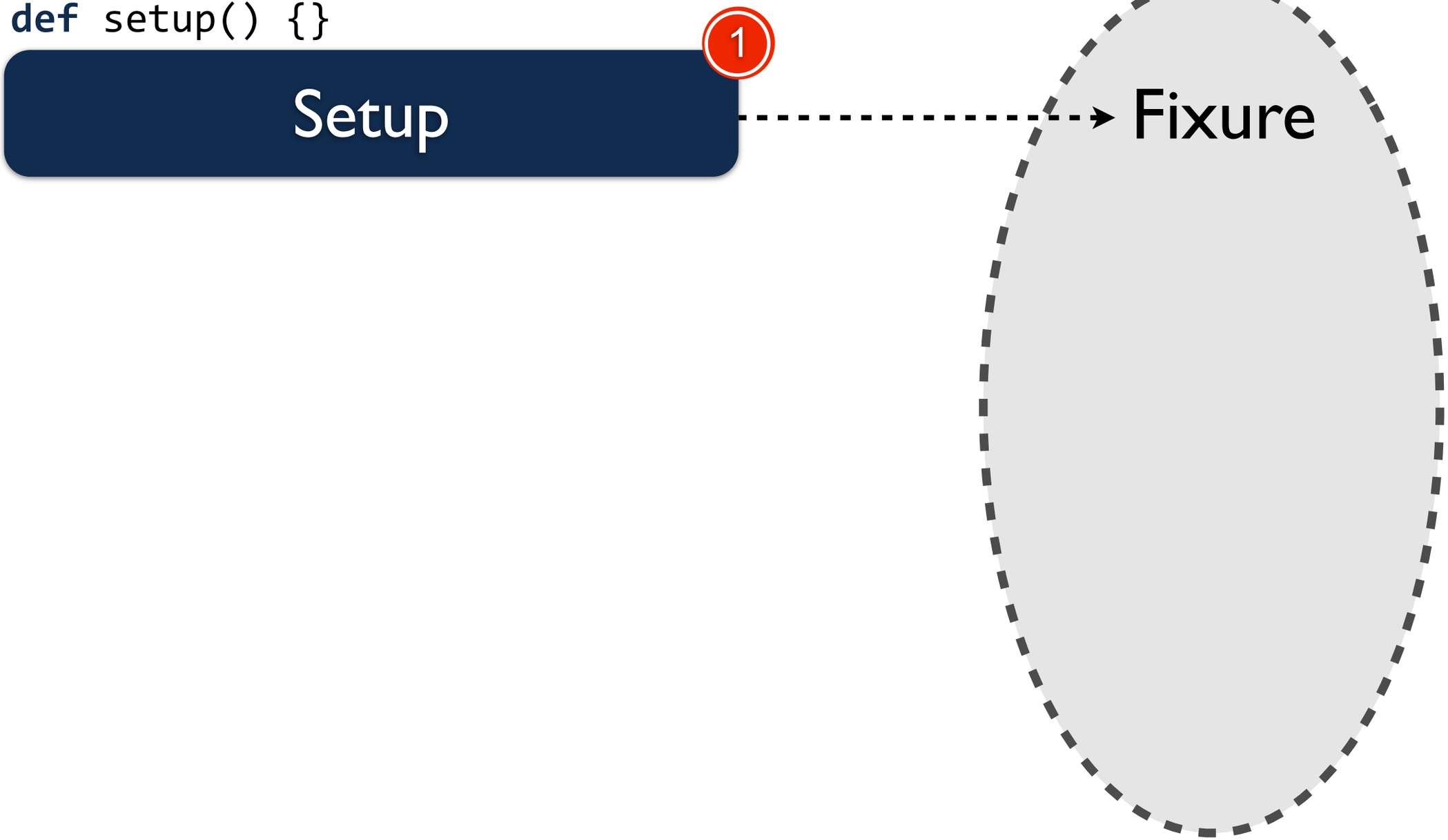
Blocks

```
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
}
```

Vier Phasen Test (Four-Phase Test)

Vier Phasen Test (Four-Phase Test)

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



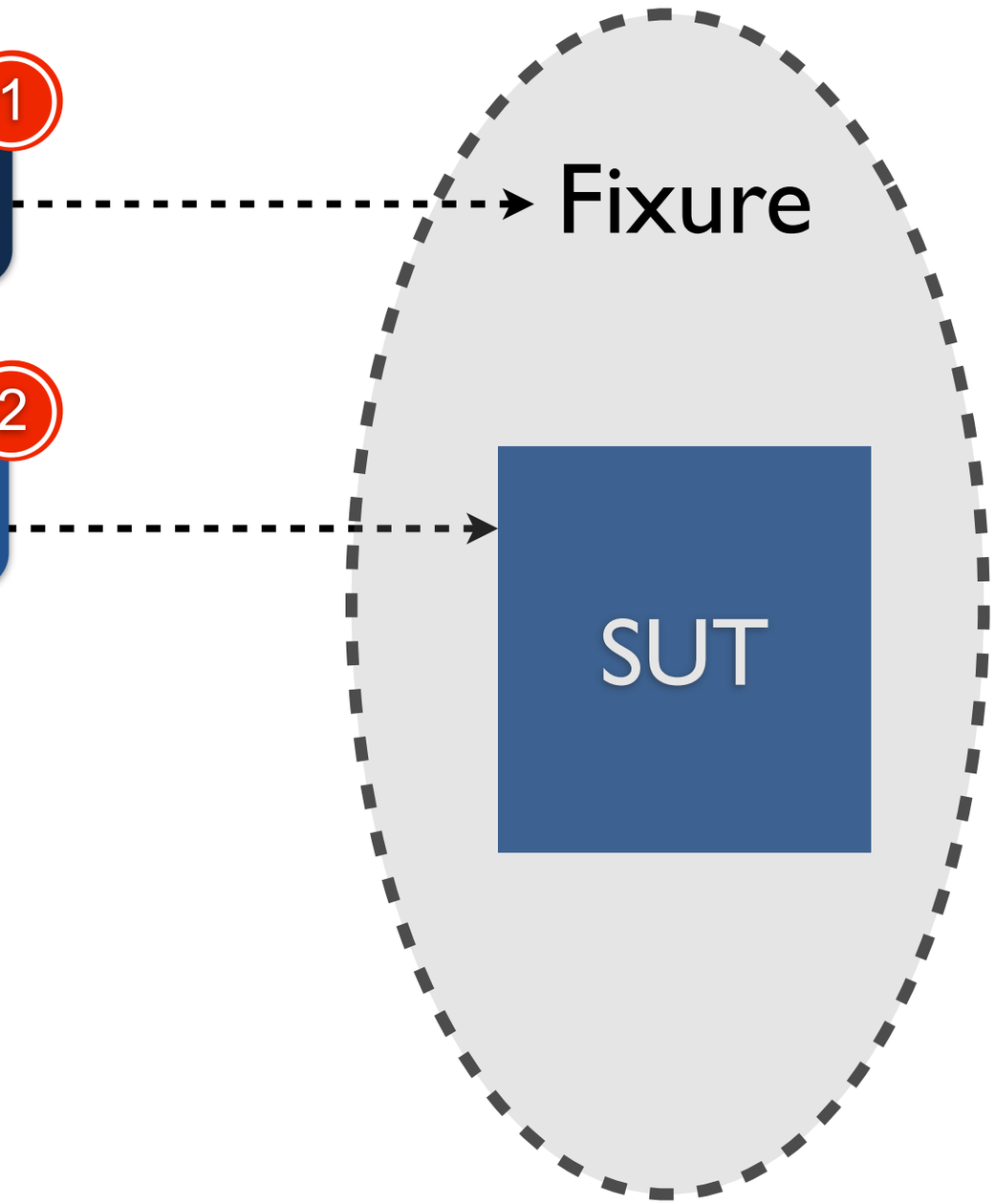
Vier Phasen Test (Four-Phase Test)

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

1 Setup

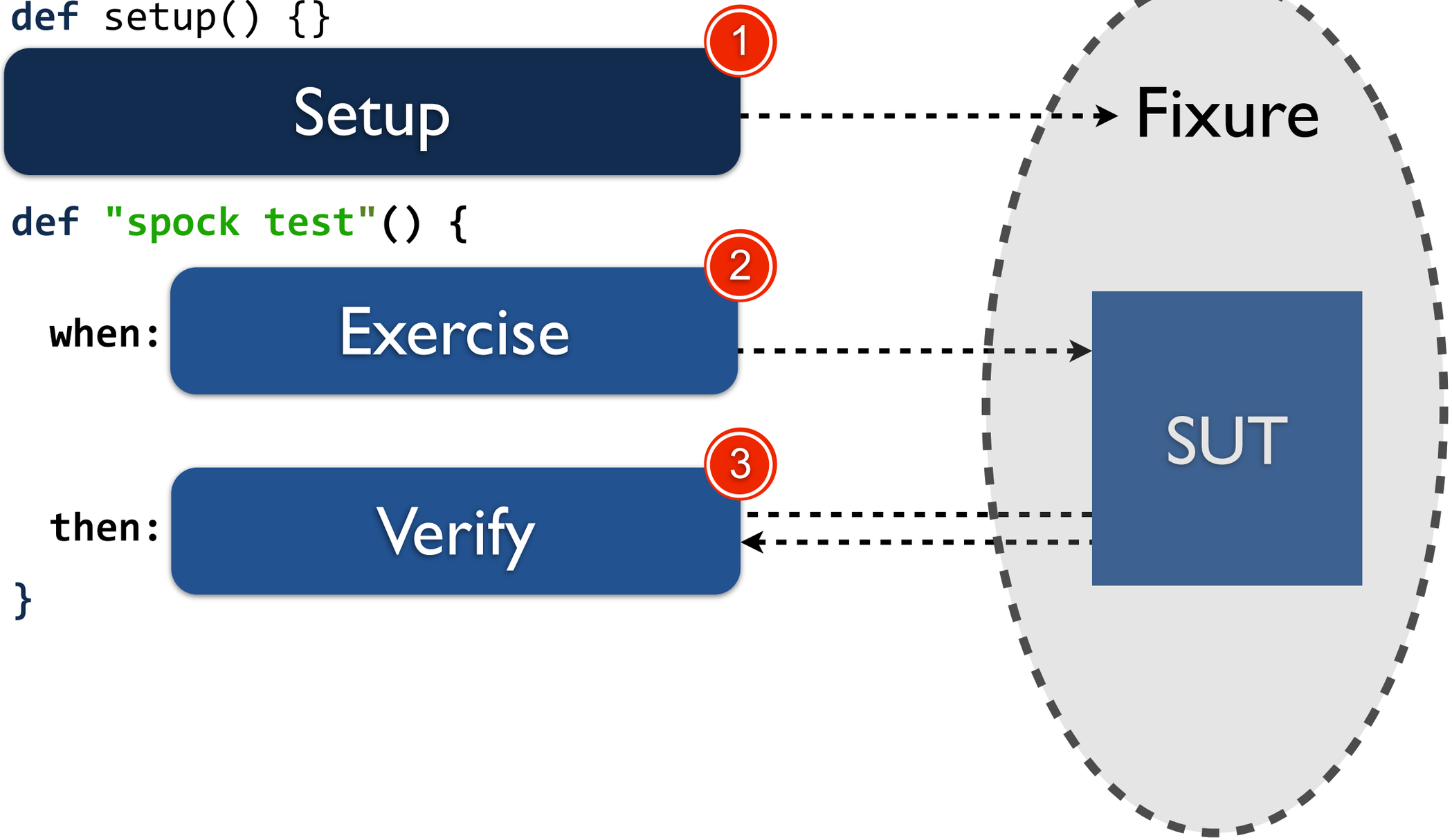
```
def "spock test"() {
```

2 `when:` Exercise

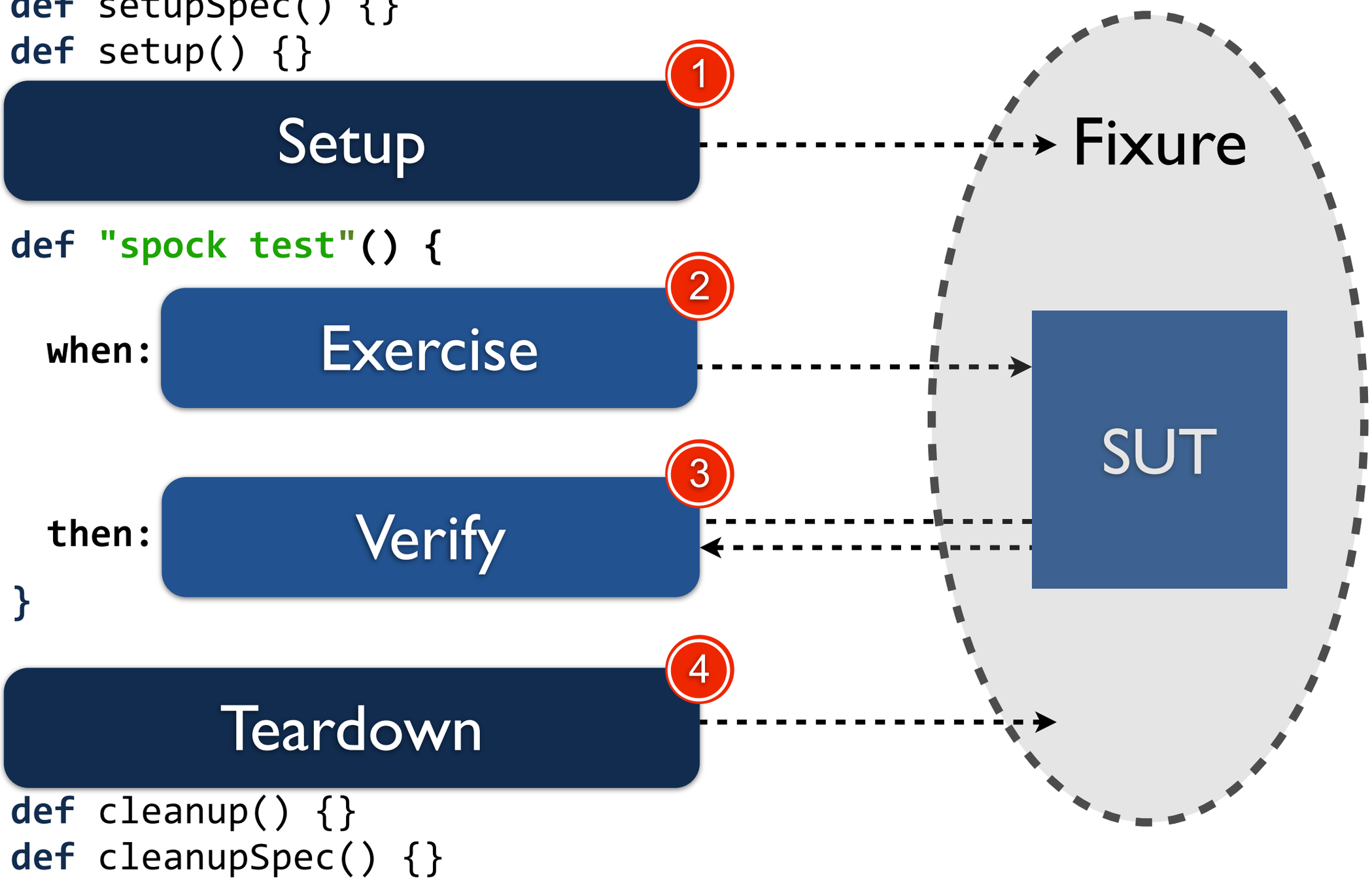


Vier Phasen Test (Four-Phase Test)

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

def "spock test"() {
  when:

  then:
}
```



Vier Phasen Test (Four-Phase Test)



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" }
}
```

Power Assertion

```
def christian = new User(id: 1, name: "Christiam")
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
|         |    6 differences (33% similarity)
|         |    (Ch)r(is)ti(am)
|         |    (Ma)r(--)ti(n-)
|         Christiam
User{id=1, basarNumber='null', name='Christiam', email='null', lastname='null'}
```

Helper Method

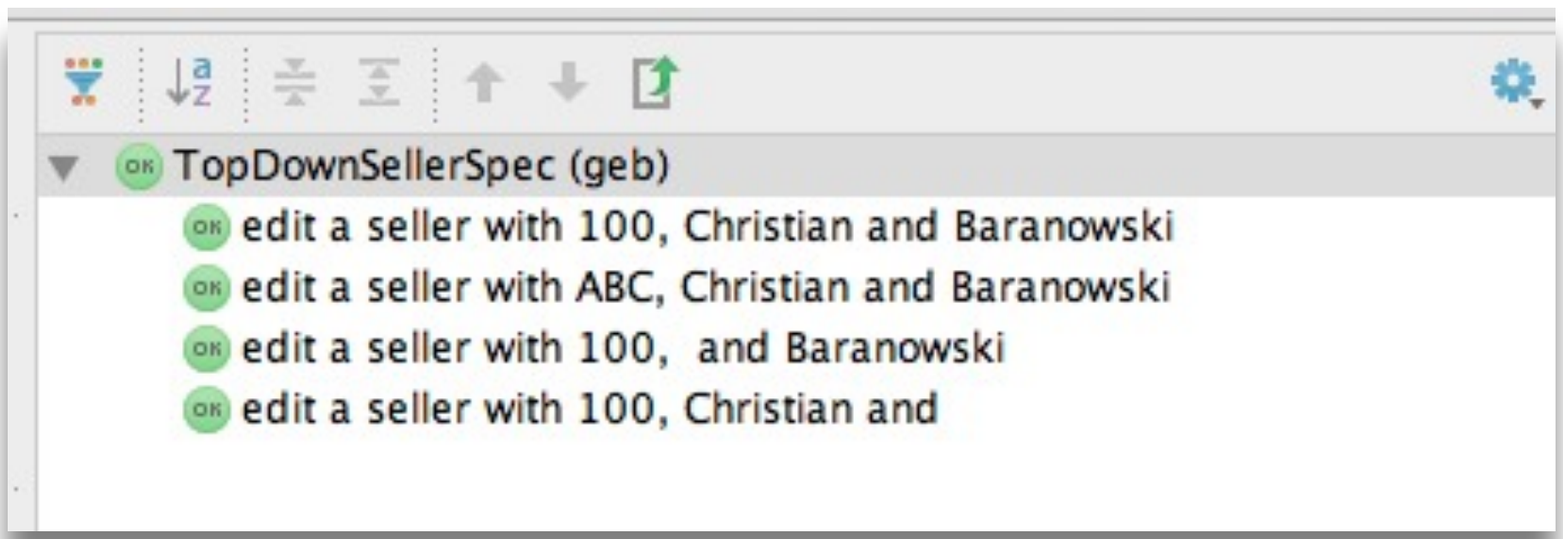
```
def "use helper method in spock test"() {
  when:
     def user = new User(name: "Christian", lastname: "Baranowski")
  then:
     referentMatches(user)
}

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"   |   ""
}
```

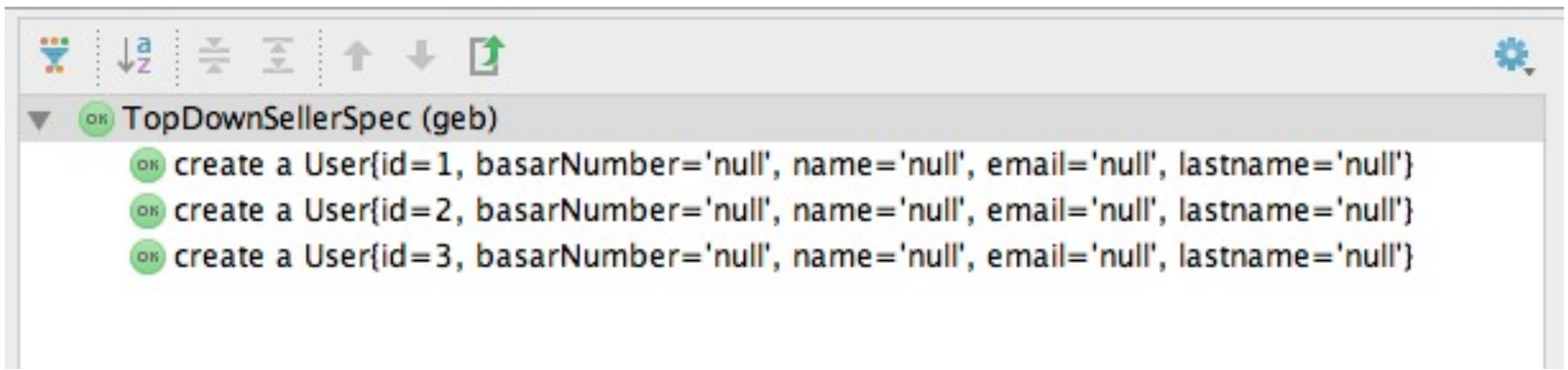
Parameterized Test

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



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)]
}
```



Warum Geb?

- Geb bietet eine Abstraktion und Vereinfachung der WebDriver API für Groovy
- Dazu werden die dyamischen Sprachfunktion von Groovy genutzt daher kann Geb nicht sinnvoll in Java Code genutzt werden
- JQuery like API für Selenium WebDriver Tests
- Geb bietet eine Mechanismus zur Seitenabstraktion der zu lesbaren Oberflächentests führt
- Einfacher waitFor{ } mir Groovy Closure für dynamische Webanwendungen
- Groovy GString bieten die Möglichkeit einfach JavaScript in Tests zu integrieren

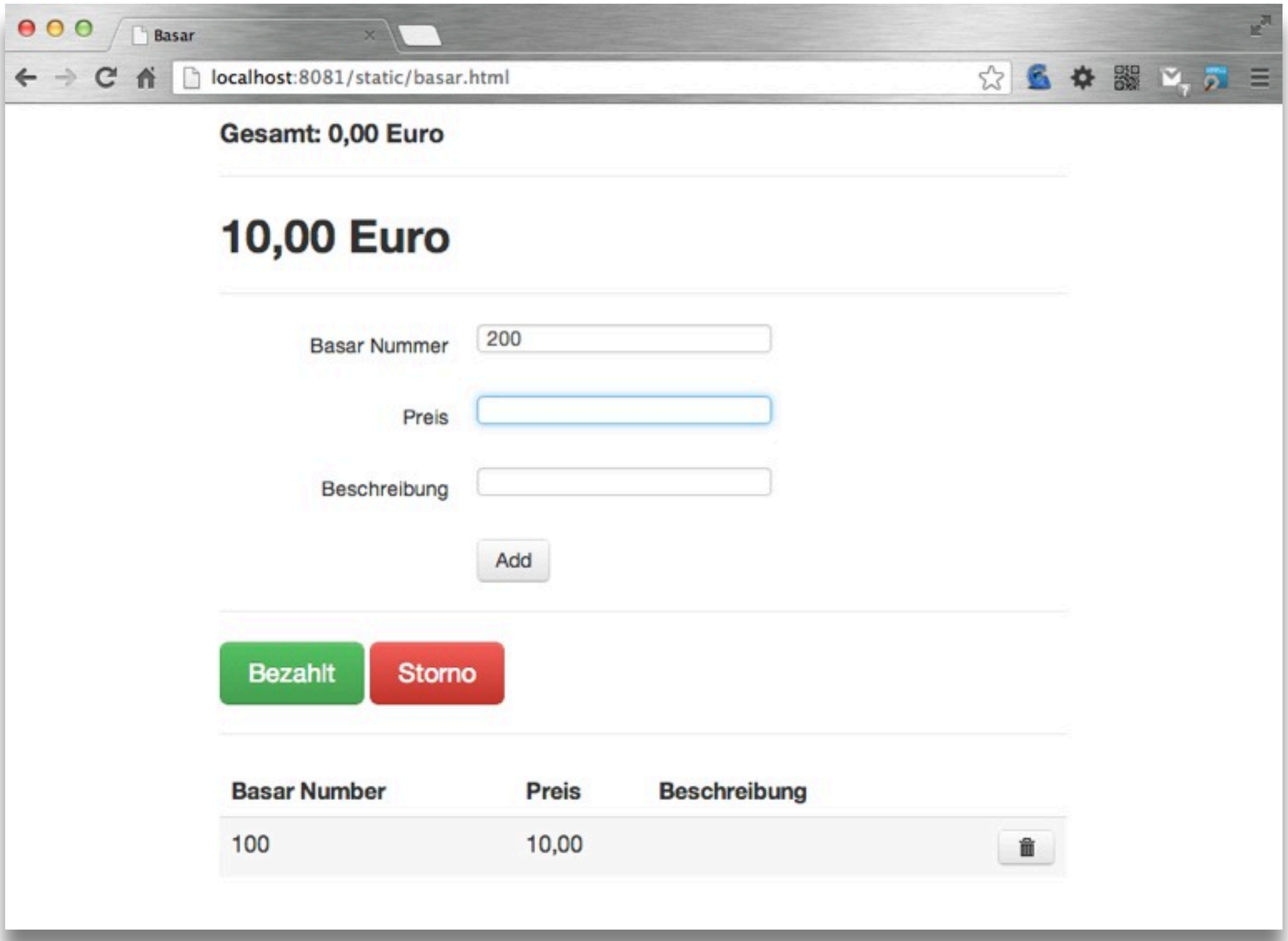
JavaScript Support in Geb


```
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 setup() {
        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)
}
```

Äquivalenzklassenbildung



Äquivalenzklassenbildung

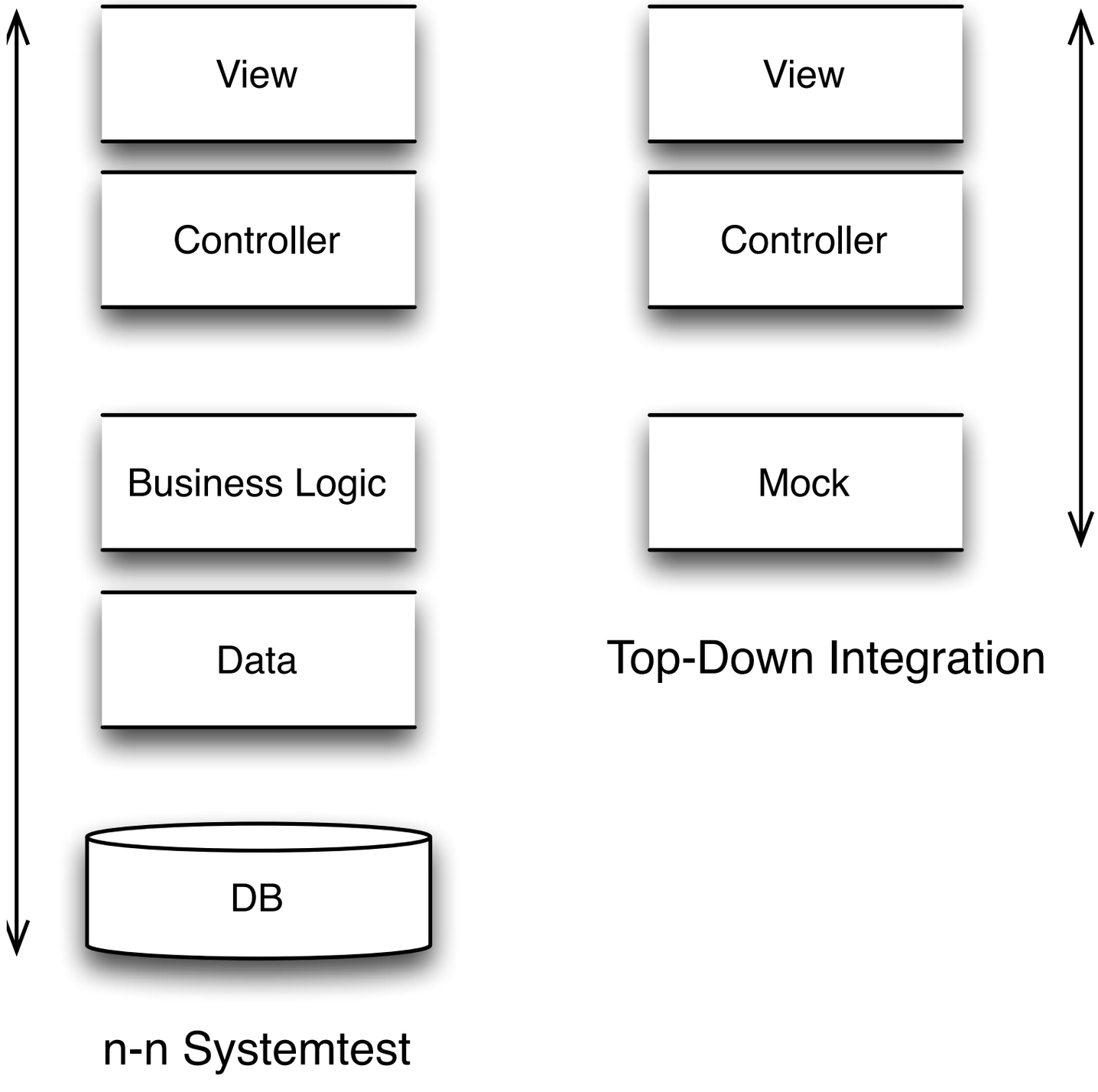
Parameter	Äquivalenzklasse	Repräsentant
BasarNummer	gÄK1_01: [0,...,MAX_LONG] uÄK1_02: [-MAX_LONG,..., 0[uÄK1_03: NaN	100 -1 „abc“
Preis	gÄK2_01: [0,...,MAX_LONG] gÄK2_02: X,50 | X = [0,..., MAX_LONG] gÄK2_03: X,5 | X = [0,..., MAX_LONG] gÄK2_04: X,00 | X = [0,..., MAX_LONG] uÄK2_05: [-MAX_LONG,...,0] uÄK2_06: X,Y | X = [-MAX_LONG,..., 0[Y = [-MAX_LONG,..., MAX_LONG] uÄK2_07: X,Y | X = [0,..., MAX_LONG] Y =]0,..., 5[uÄK2_08: X,Y | X = [0,..., MAX_LONG] Y =]5,..., 50[uÄK2_09: X,Y | X = [0,..., MAX_LONG] Y =]50,..., MAX_LONG[uÄK2_10: X,Y | X = [0,..., MAX_LONG] Y = NaN uÄK2_11: NaN uÄK2_12: X,Y | X = NaN Y = [0,..., MAX_LONG]	10 0,50 0,5 11,00 -1 -1,100 100,2 100,40 100,510 100,“abc„ “abc„ “abc„,50
Beschreibung	gÄK3_01: [NULL...TEXT]	“abc„

Live Demo

Äquivalenzklassenbildung mit Spock und Geb

Let‘s write some Groovy Spock Tests

Top-Down Integration



Top-Down Integration

```
@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
}
```

Q&A

Twitter @tux2323