2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Spock und Geb in Action

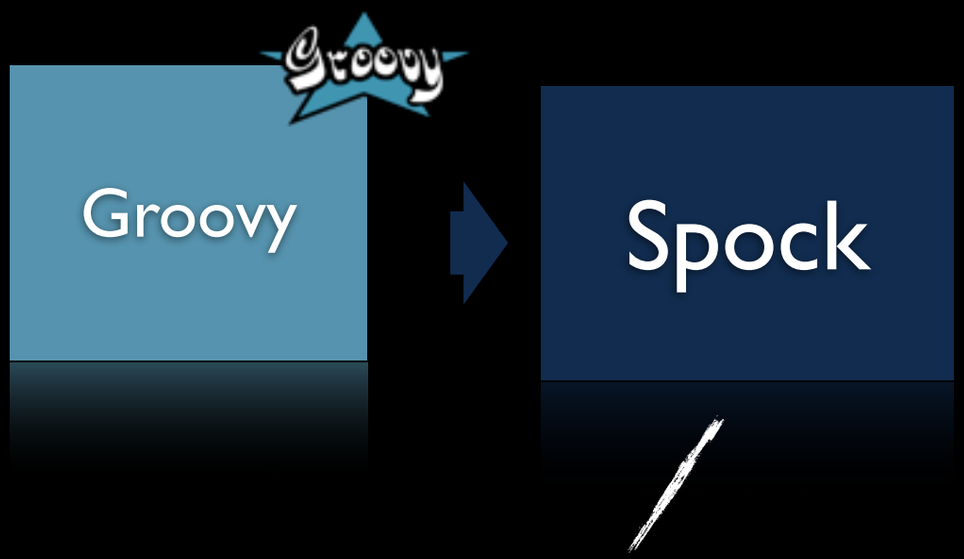
Behavior Driven Development von Web-Applikationen mit Groovy, Spock und Geb

Christian Baranowski

SEITENBAU GmbH

Werkzeuge

Werkzeuge



www.spockframework.org

Werkzeuge

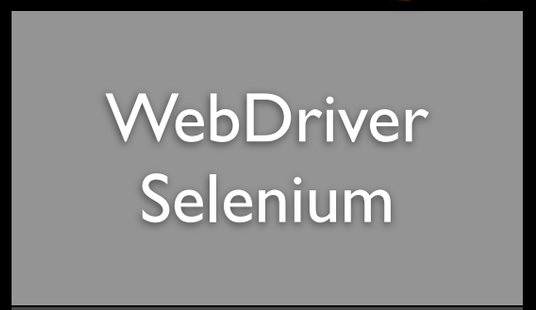
www.gebish.org

Groovy → Spock → Geb

www.spockframework.org

Werkzeuge

www.gebish.org



www.spockframework.org

Werkzeuge

Selenium WebDriver
Walery Strauch
Q34: Donnerstag, 5. 9., 15:40 – 16:50 Uhr

www.gebish.org

Spock

Geb

WebDriver
Selenium

www.spockframework.org

Warum Spock?

Warum Spock?

1. Spock ist ein einfaches BDD Werkzeug für die JVM

weitere Argumente ...

weitere Argumente ...

2. Spock biete ausdrucksstarke DSL zur Spezifikation von Tests, insbesondere für Parametrisierte Tests und Mocking

weitere Argumente ...

2. Spock biete ausdrucksstarke DSL zur Spezifikation von Tests, insbesondere für Parametrisierte Tests und Mocking
3. Spock kann sowohl für Unit- wie Systemtests genutzt werden

weitere Argumente ...

2. Spock biete ausdrucksstarke DSL zur Spezifikation von Tests, insbesondere für Parametrisierte Tests und Mocking
3. Spock kann sowohl für Unit- wie Systemtests genutzt werden
4. JUnit Kompatibel - Zur Ausführung wird JUnit genutzt, Integration in IDEs, Build-Tools (Ant, Maven, Gradle...) und CI (Jenkins)

weitere Argumente ...

2. Spock biete ausdrucksstarke DSL zur Spezifikation von Tests, insbesondere für Parametrisierte Tests und Mocking
3. Spock kann sowohl für Unit- wie Systemtests genutzt werden
4. JUnit Kompatibel - Zur Ausführung wird JUnit genutzt, Integration in IDEs, Build-Tools (Ant, Maven, Gradle...) und CI (Jenkins)
5. Spock vereint die besten Features aus bewährten Tools wie JUnit und RSpec

Spock Given When Then

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
}
```

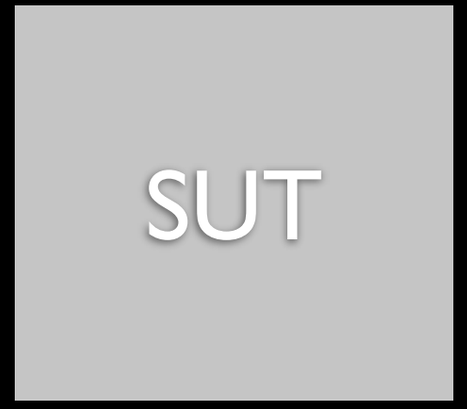
Spock Blocks

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

Spock Blocks in Action

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

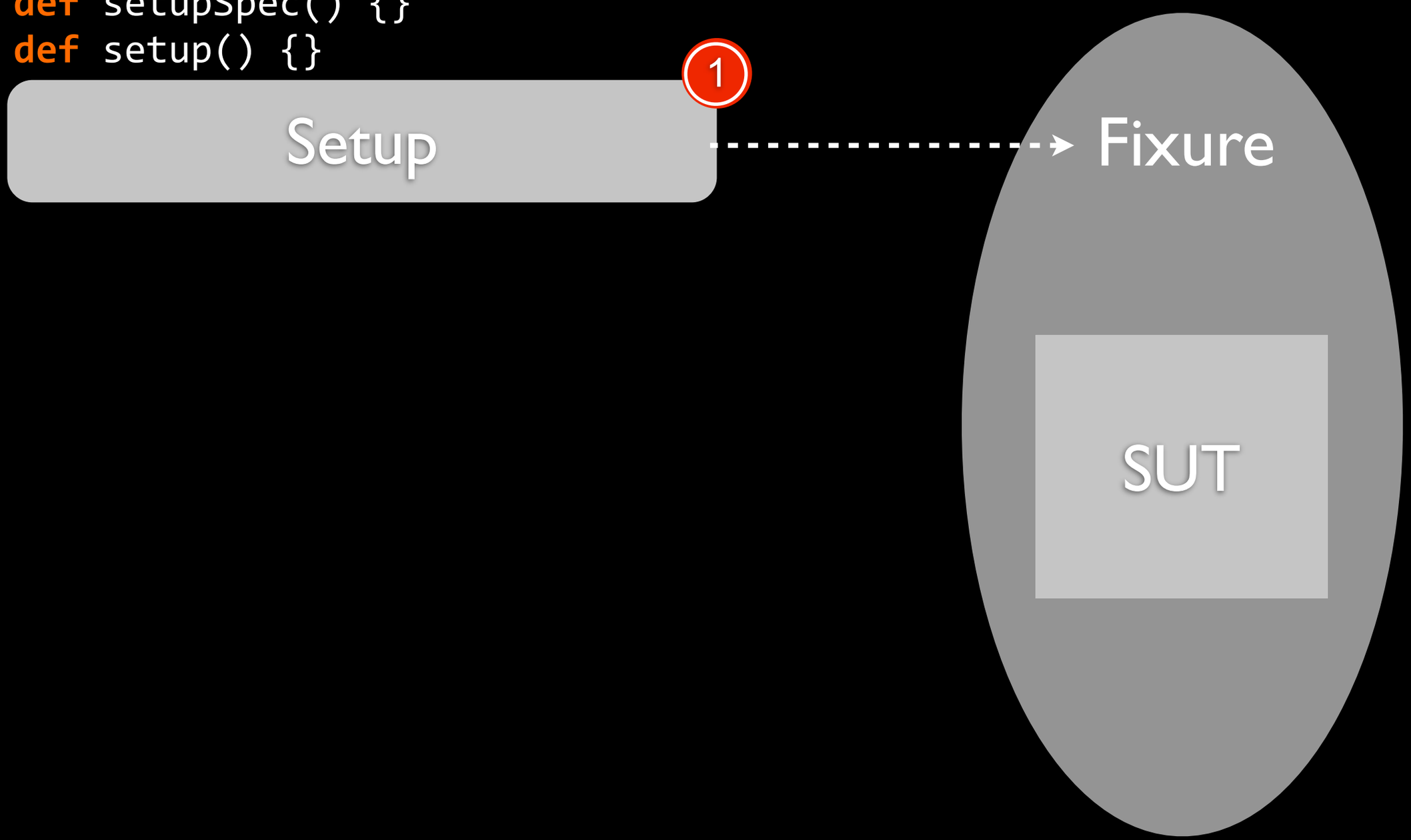
Vier Phasen Test (Four-Phase Test)



xUnit Test Patterns - Gerard Meszaros

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

Fixure

SUT

Vier Phasen Test (Four-Phase Test)

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

def "spock test"() {

  when:

  then:
}
```

Setup

Exercise

Verify

1

2

3

Fixure

SUT

Vier Phasen Test (Four-Phase Test)

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

Setup (1)

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

```
  when:
```

Exercise (2)

```
  then:
```

Verify (3)

```
}
```

Teardown (4)

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

Fixure

SUT

Spock Lifecycle

```
class LifecycleSpec extends Specification {
    def setupSpec() {  println "01 - setup Spec" }
    def setup() { println "02 - setup" }
    def "simple spock test"() {
        given:
          println "02 - setup test"
        expect:
            def data = []
            data == []
        cleanup:
            println "04 - cleanup test"
    }
    def cleanup() { println "04 - cleanup" }
    def cleanupSpec() { println "04 - cleanup Spec" }
}
```

Groovy Power Assertion

```
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'}
```

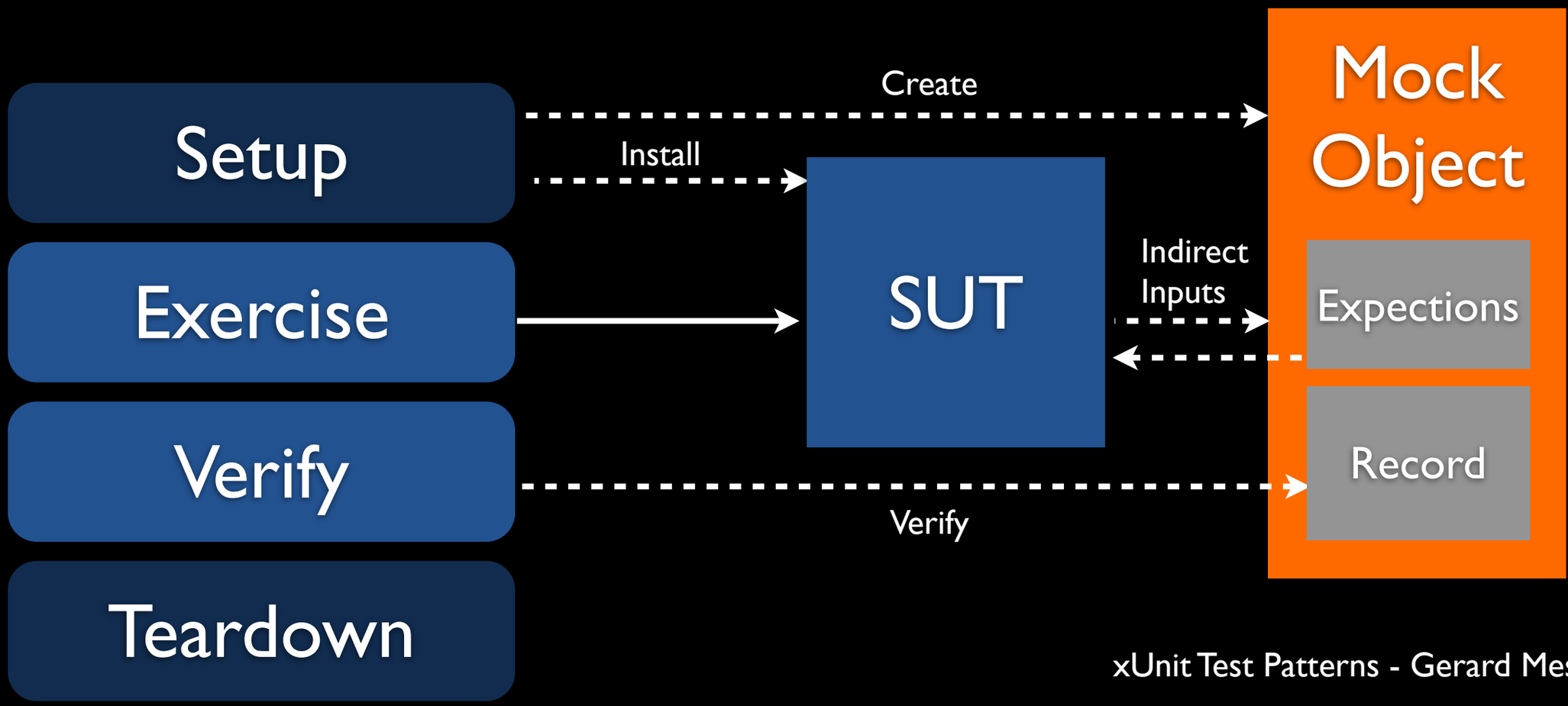
Groovy Power Assertion

```
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'}
```

Im then: block Kurzschreibweise möglich ohne assert

Mocking



xUnit Test Patterns - Gerard Meszaros

Mocking

```
def mockService = Mock(SellerService)
1 *       mockService.findByName( 'max' )
```

Mocking

```
def mockService = Mock(SellerService)
1 *      mockService.findByName( 'max' )
```

cardinality

```
    1 *      mockService.findByName( 'max' )
```

cardinality

```
(0..1) * mockService.findByName( 'max' )
(1.._) * mockService.findByName( 'max' )
_ * mockService.findByName( 'max' )
```

Mocking

```
def mockService = Mock(SellerService)
1 *      mockService.findByName( 'max' )
```

cardinality

method constraint

cardinality

```
1 *      mockService.findByName( 'max' )
(0..1) * mockService.findByName( 'max' )
(1.._) * mockService.findByName( 'max' )
_ * mockService.findByName( 'max' )
```

method constraint

```
1 *      mockService._( *_ )
0 *      mockService._
```

Mocking

```
def mockService = Mock(SellerService)
1 *      mockService.findByName( 'max' )
```

cardinality — method constraint — argument constraint

cardinality

```
1 *      mockService.findByName( 'max' )
(0..1) * mockService.findByName( 'max' )
(1..*) * mockService.findByName( 'max' )
* * mockService.findByName( 'max' )
```

method constraint

```
1 *      mockService._( *_ )
0 *      mockService._
```

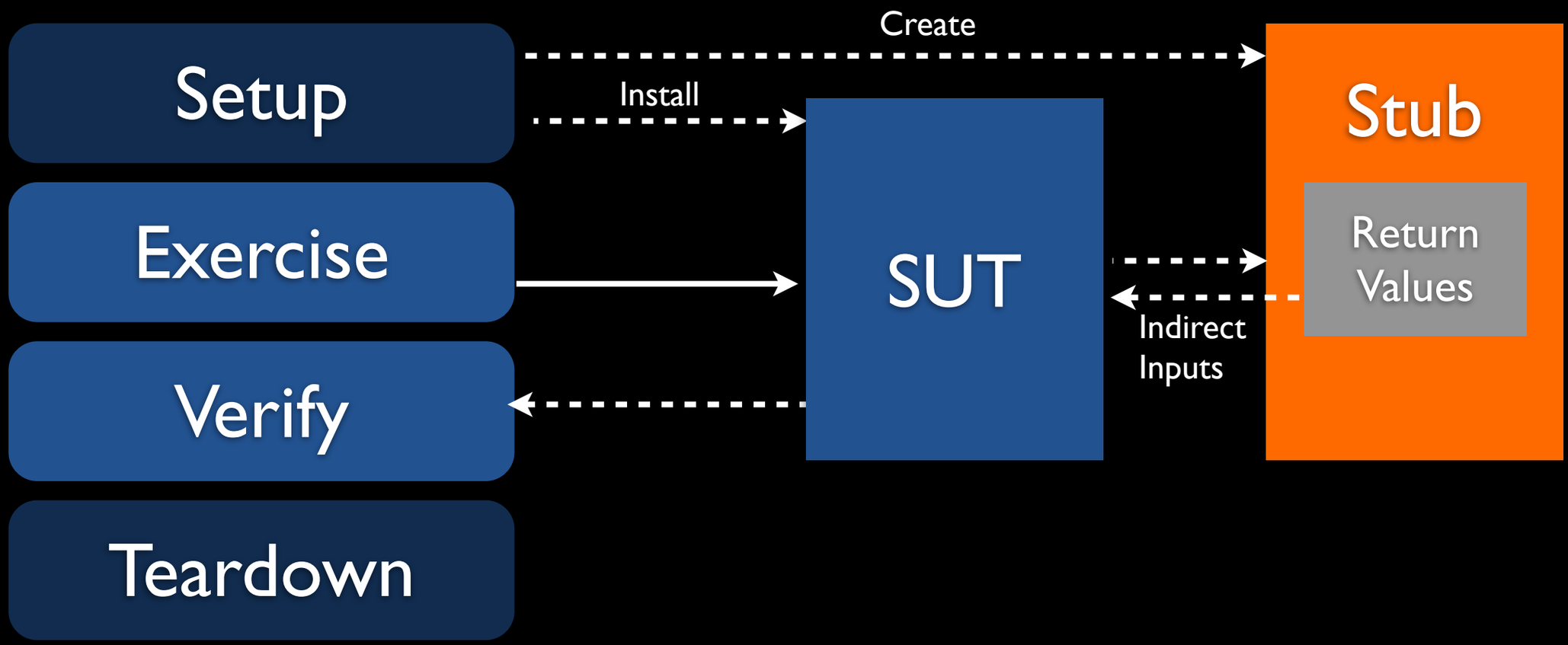
argument constraint

```
1 *      mockService.findByName( _ )
1 *      mockService.findByName( !null )
1 *      mockService.findByName( !'max' )
1 *      mockService.findByName( { it.size() > 0 } )
```

Mocking Example

```
def "only the first call should be forwarded"() {
  given:
    def mockService = Mock(SellerService)
    def cache = new SimpleCache(target: mockService)
  when: "invoke two times the cached method"
    cache.findByName('max')
    cache.findByName('max')
  then: "target method was invoked one time"
    1 * mockService.findByName('max')
}
```

Stubbing



xUnit Test Patterns - Gerard Meszaros

Stubbing

```
def mockService = Mock(SellerService)

mockService.findByName( 'max' ) >> 'OK'
```

Stubbing

```
def mockService = Mock(SellerService)

mockService.findByName( 'max' ) >> 'OK'
```

method constraint

Stubbing

```
def mockService = Mock(SellerService)

mockService.findByName( 'max' ) >> 'OK'
```

method constraint

argument constraint

Stubbing

```
def mockService = Mock(SellerService)

mockService.findByName( 'max' ) >> 'OK'
```

method constraint

argument constraint

response generator

response generator

```
mockService.findByName( 'max' ) >> 'OK'
mockService.findByName( 'max' ) >> 'OK' >> 'ERR'
mockService.findByName( 'max' ) >>> ['OK', 'ERROR', 'ups']
mockService.findByName( 'max' ) >> { throw new InternalError() }

mockService.findByName( _ ) >> 'OK'
mockService.findByName( _ ) >> {name -> name.size() > 1 ? 'OK' : 'ERR'}
```

Stubbing Example

```
def "cache should return result of the target"() {
  given: "a mock service object that returns OK"
  def mockService = Mock(SellerService)
  mockService.findByName('max') >> 'OK' >> { throw exception() }
  and: "cache with the mock object as target"
  def cache = new SimpleCache(target: mockService)
  when: "invoke cache the first time"
  def result = cache.findByName('max')
  then: "result is OK"
  result == 'OK'
  when: "invoke cache the second time"
  result = cache.findByName('max')
  then: "result is OK"
  result == 'OK'
}
```

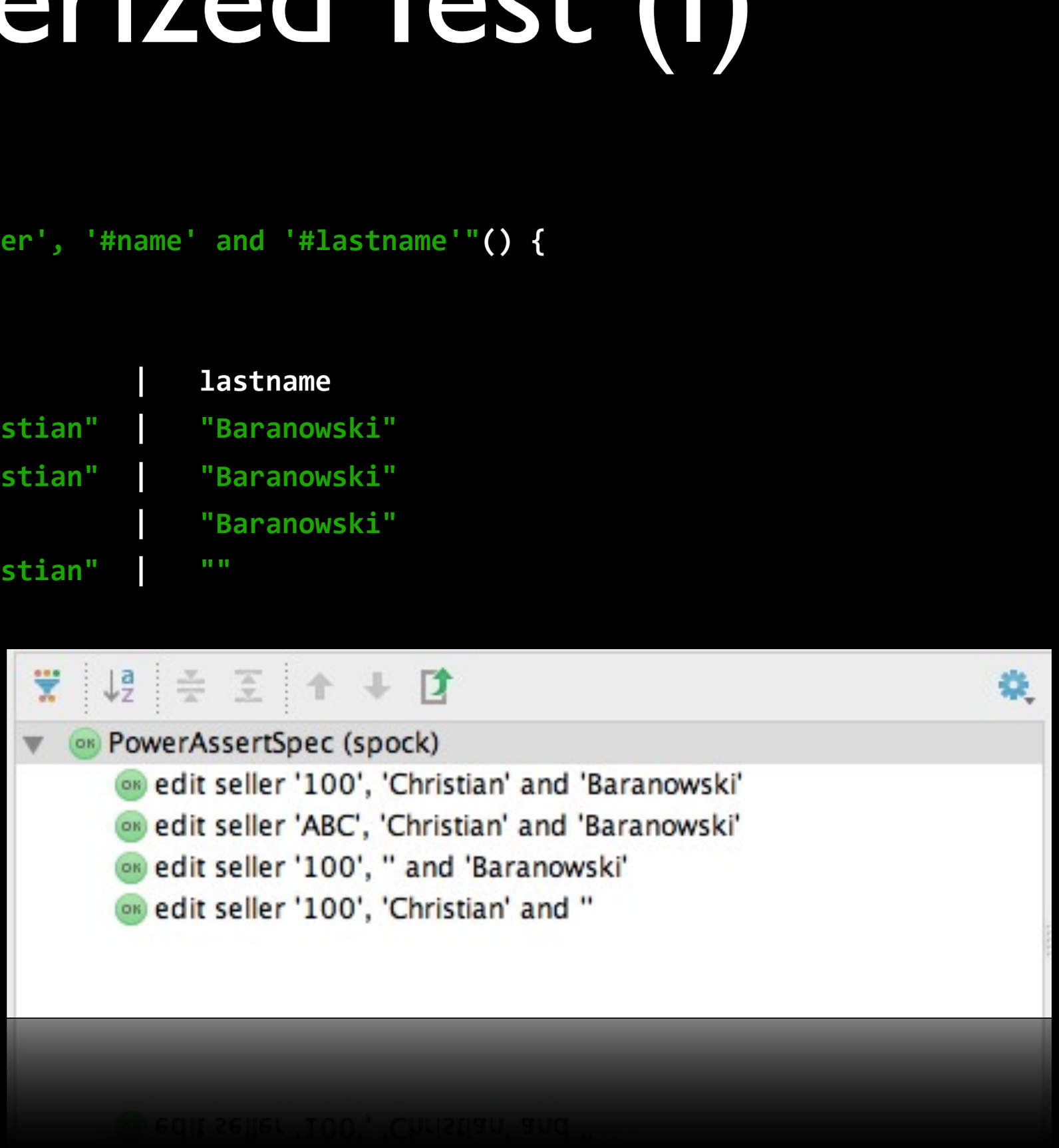
Parameterized Test (I)

Parameterized Test (I)

```
@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 (I)

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



Parameterized Test (II)

```
@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
- 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
- einfache JavaScript Integration für Testanbindung

Geb „JQuery like API“

```
$(«css selector», «index or range», «attribute / text matchers»)
```

Geb „JQuery like API“

```
$(«css selector», «index or range», «attribute / text matchers»)
```

selector

```
$('div.span8 p')
$('#newUserButton')
$('.error')
```

Geb „JQuery like API“

```
$(«css selector», «index or range», «attribute / text matchers»)
```

selector

```
$('div.span8 p')
$('#newUserButton')
$('.error')
```

index or range

```
$('div', 2, class: 'span2')
$('div', 0..2, class: 'span2')
```

Geb „JQuery like API“

```
$(«css selector», «index or range», «attribute / text matchers»)
```

selector

```
$('div.span8 p')
$('#newUserButton')
$('.error')
```

index or range

```
$('div', 2, class: 'span2')
$('div', 0..2, class: 'span2')
```

attribute / text matchers

```
$('button', class: 'btn')
$('td', text: '559')
$('td', text: contains('55'))
$('input', value: ~/.*/)
```

Geb „JQuery like API“

```
$(«css selector», «index or range», «attribute / text matchers»)
```

selector

```
$('div.span8 p')
$('#newUserButton')
$('.error')
```

index or range

```
$('div', 2, class: 'span2')
$('div', 0..2, class: 'span2')
```

attribute / text matchers

```
$('button', class: 'btn')
$('td', text: '559')
$('td', text: contains('55'))
$('input', value: ~/.*/)
```

Finding

```
$('div').find('.span8')
```

Geb „JQuery like API“

Traversing

```
$('div').parent()
$('div').next()
$('div').siblings()
$('div').children()

$('td', text: '559').parent().find('td')
$('td', text: '559').parent().children()
```

Geb „JQuery like API“

Accessing Attributes

```
$('#newUser').@id          == 'newUser'
$('#newUser').attr('id') == 'newUser'
$('#newUser').text()       == 'Neuen Nutzer anlegen'
$('#newUser').tag()        == 'button'
$('#newUser').classes()  == ['btn', 'updateUserButton']
$('button')*.text()        == ['Löschen', 'Neuen Nutzer']
```

clicking

```
$('#newUser').click()
```

input values

```
$('input', name: 'basarNumber').value('100')
```

Sending keystrokes

```
$('div.desc') << 'X'
$('input', name: 'basarNumber') << '100'
```

Geb waitFor API

```
// use default wait configuration
waitFor {
    sellerCountBefore < sellerCount()
}

// wait for up to 10 seconds, using the default retry interval
waitFor(10) {
    sellerCountBefore < sellerCount()
}
waitFor(timeout=20) { $("#newUser") }

// wait for up to 10 seconds, waiting half a second between retries
waitFor(10, 0.5) {
    sellerCountBefore < sellerCount()
}

// custom message
waitFor(message: 'Button not found', timeout=2) {
    $("#NewUserButton")
}
```

Page Object Pattern

```
import geb.Page

class BasarPage extends Page {
    static url = "static/basar.html"
    static at = { title == "Basar" }
    static content = {
        basarForm { $("form") }
        addButton (to: BasarPage) { basarForm.addCartItem() }
    }
}
```

Page Object Pattern

```
import geb.Page

class BasarPage extends Page {
    static url = "static/basar.html"
    static at = { title == "Basar" }
    static content = {
        basarForm { $("form") }
        addButton (to: BasarPage) { basarForm.addCartItem() }
    }
}
```

```
// Test (Test Logik)
Browser.drive {
    to BasarPage
    assert at(BasarPage)
    basarForm.with {
        basarNumber = "100"
        price = "10,50"
    }
    addButton.click()
}
```

Action Test

```
def "add a item to the cart"() {
        given:
        BasarBind basar = start BasarBind

        when:
        basar {
            enter '100' into basarNumberField
            enter '0,50' into priceField
            click addButton
        }

        then:
        basar.sum == '0,50'
}
```

https://github.com/tux2323/gebbind

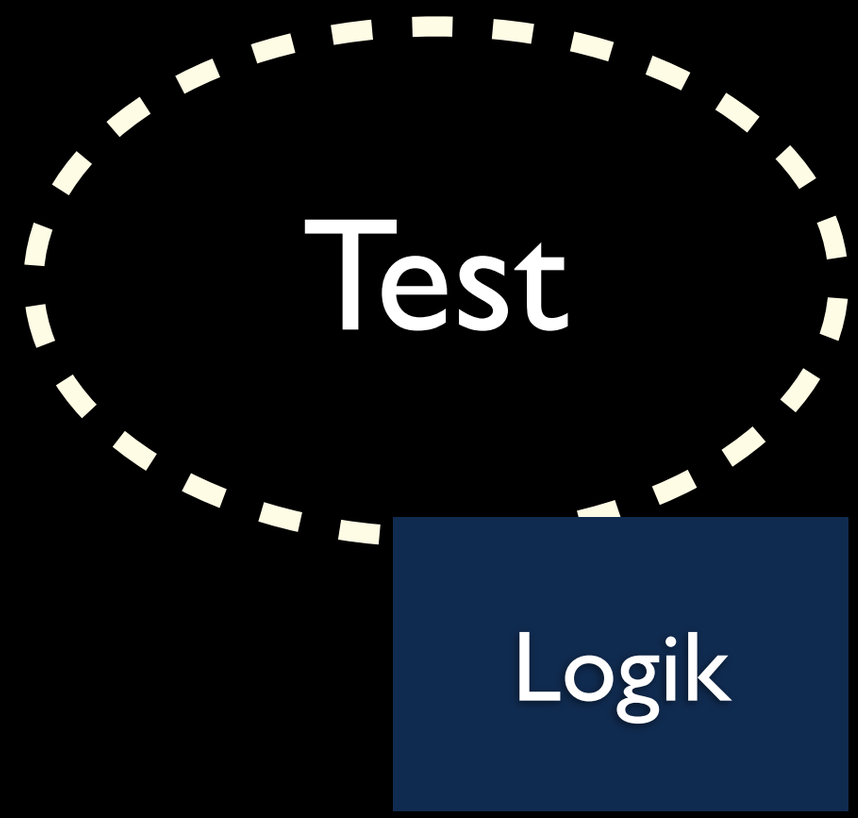
Action Test

```
class BasarBind extends Bind {

    def start() {
        go "/static/basar.html"
        waitFor { title == "Basar" }
    }

    InputText basarNumberField = inputText { $("#basarNumber") }
    InputText priceField = inputText { $("#price") }

    Button addButton = button onClick: {
        def beforeClickCartSize = $('tr').size()
        $('#addCartItem').click()
        waitFor {
            $('tr').size() > beforeClickCartSize ||
            $('.text-error').size() > 0
        }
    }

    Text sum = text { $('#sum') }
}
```

https://github.com/tux2323/gebbind

Test Design

Test Design

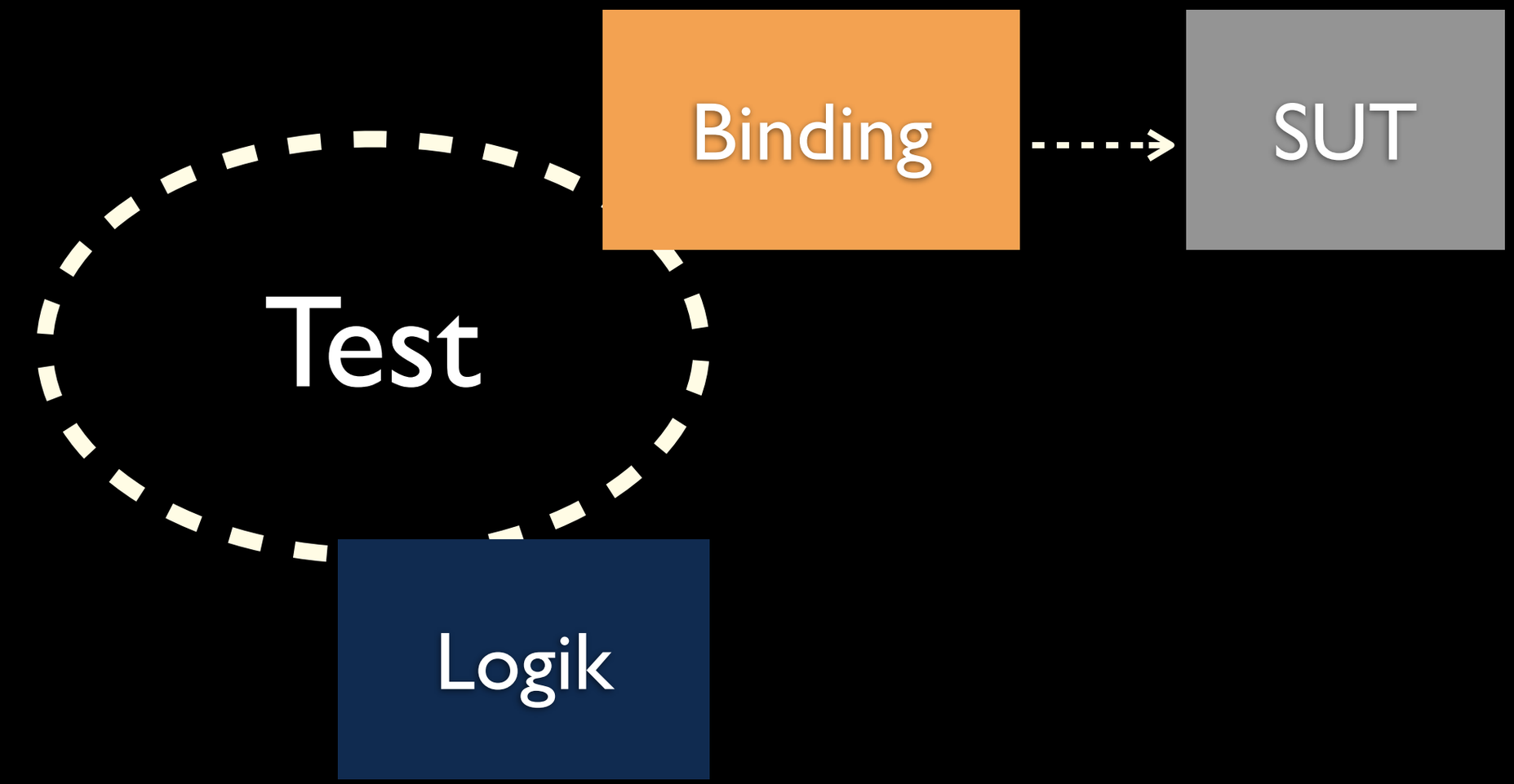


Test Design

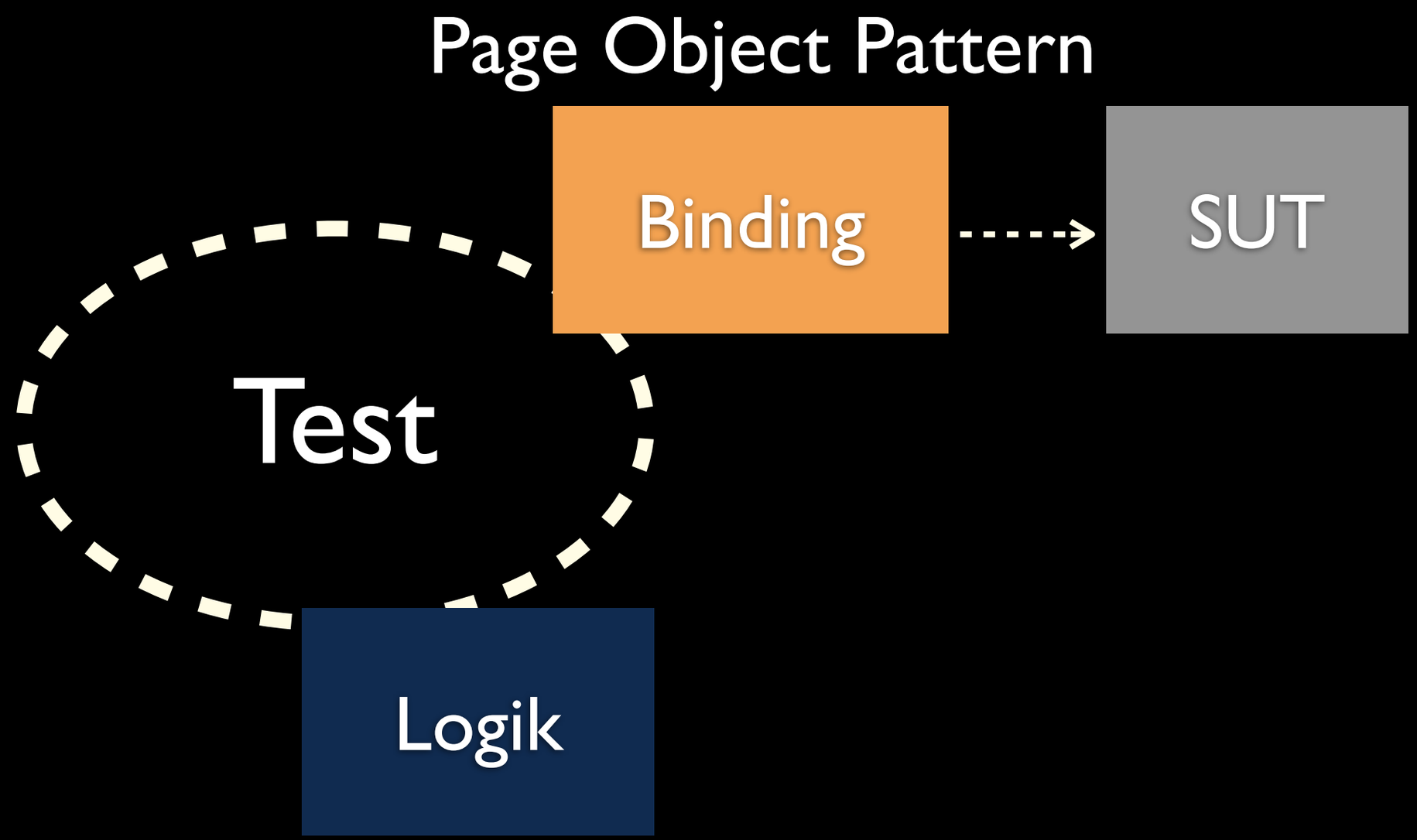
Logik

Spock Framework

Test Design

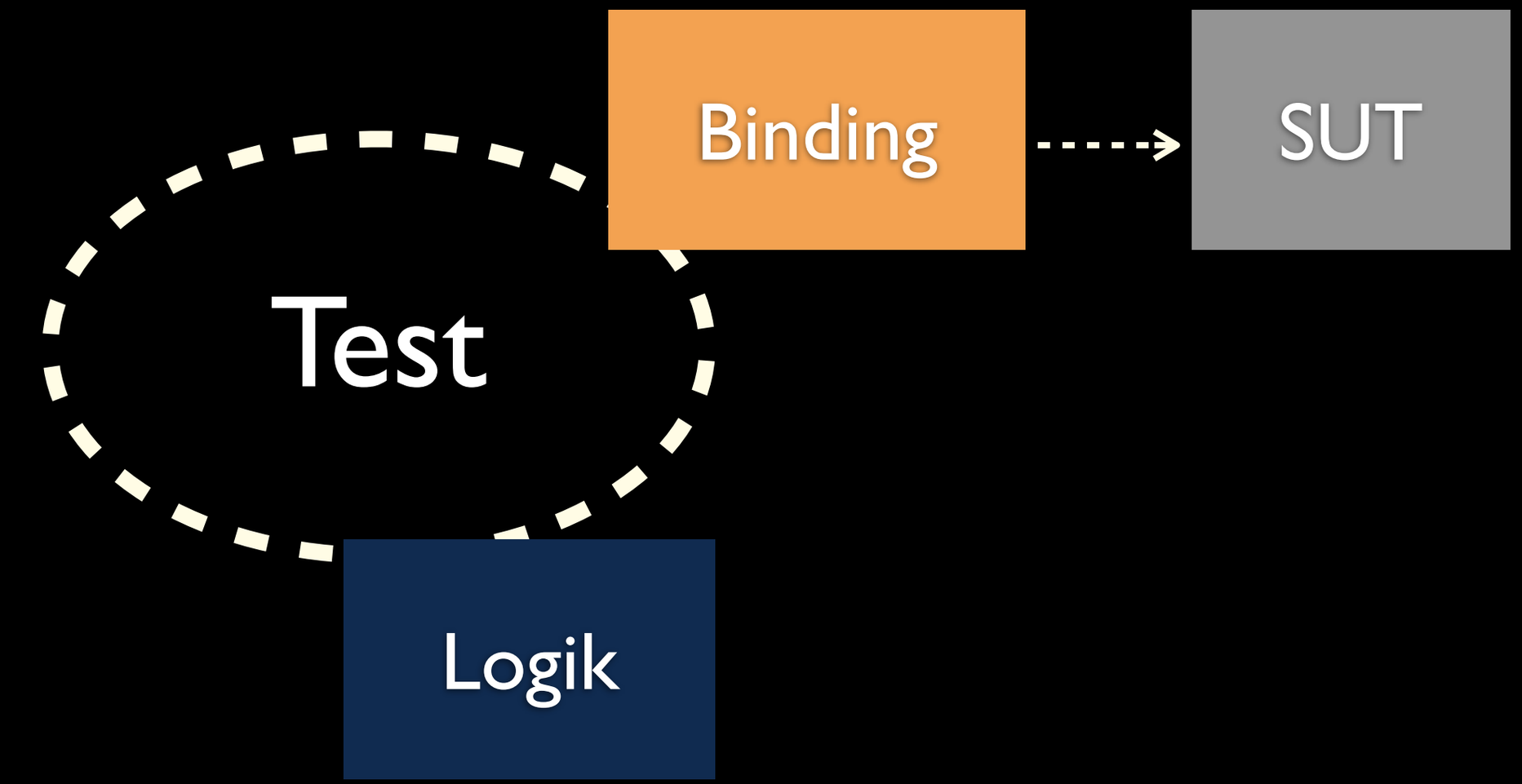


Test Design


Page Object Pattern
Binding
SUT
Test
Logik
Spock Framework

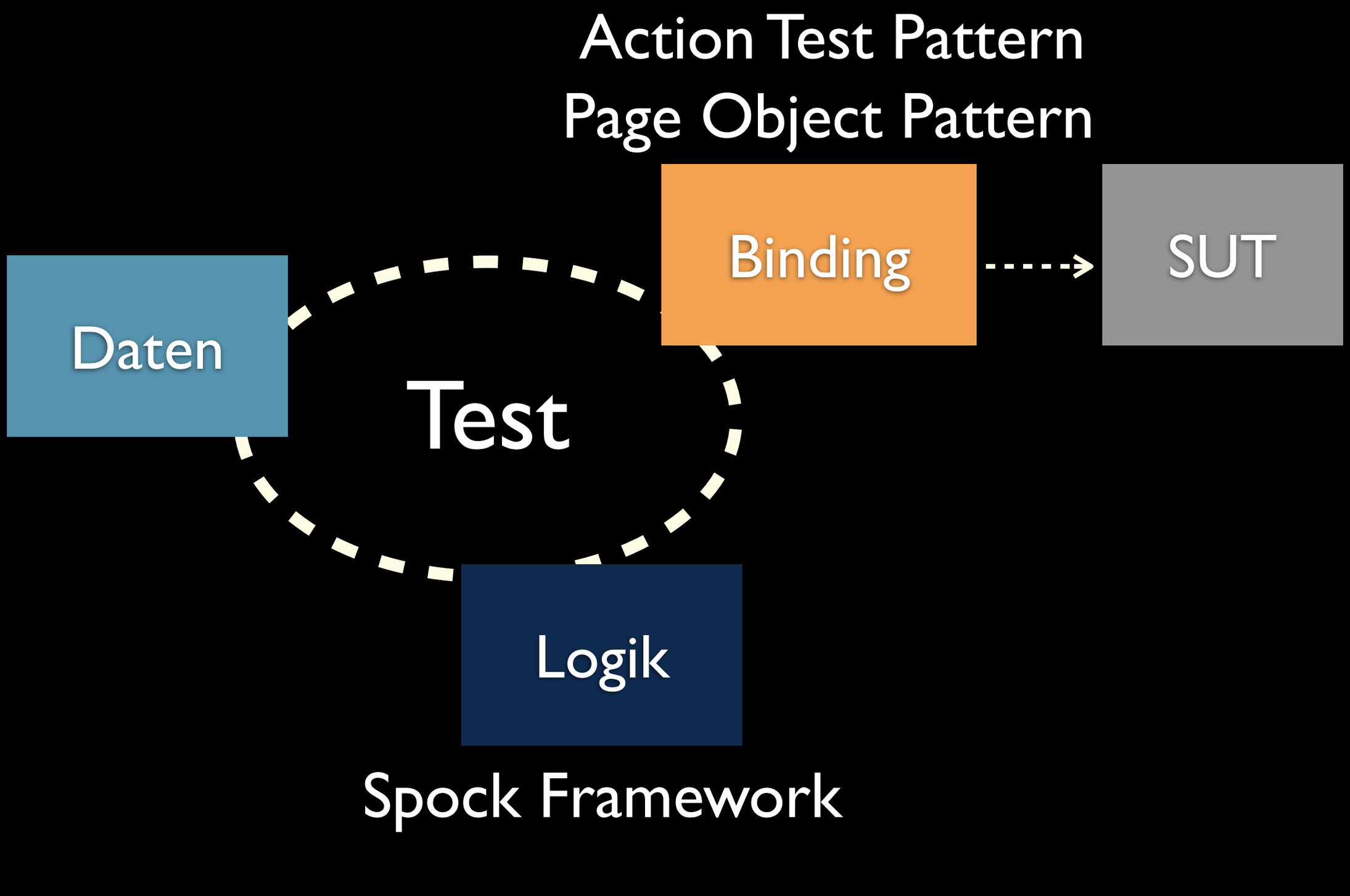
Test Design

Action Test Pattern
Page Object Pattern



Spock Framework

Test Design

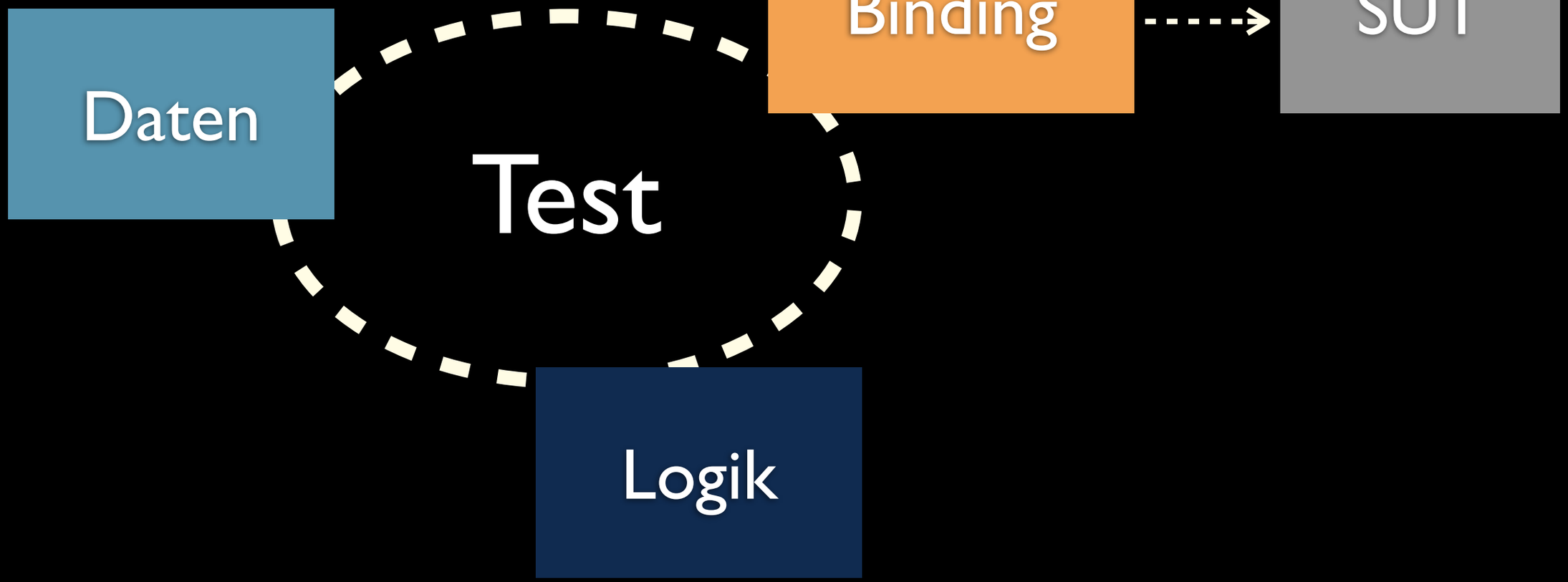

Action Test Pattern
Page Object Pattern
Binding
SUT
Daten
Test
Logik
Spock Framework

Test Design

Parameterized Test

```
basarNumber  | name         ||  lastname
"100"        | "Christian"  ||  "Baranowski"
"ABC"        | "Christian"  ||  "Baranowski"
"100"        | ""           ||  "Baranowski"
"100"        | "Christian"  ||  ""
```

Action Test Pattern
Page Object Pattern



Spock Framework

2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

https://github.com/tux2323/basar-spockgeb-demo

Vielen Dank!

Christian Baranowski

SEITENBAU Gmbh