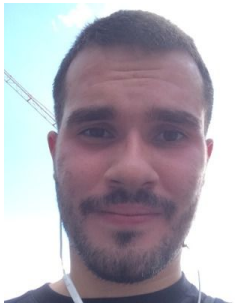
# Tests mit Kotlin


# Wie testen Spaß machen kann

János Gericke, Christof Vollrath

JavaLand Brühl, 2019

# Über uns



## János Gericke

Junior Software Engineer
Spring-Media

Test-focused Software Engineer, Frontend- & Backend-Testing, Krav Maga Practitioner

janos.gericke@spring-media.de
https://www.xing.com/profile/Janos_Gericke
https://www.linkedin.com/in/janos-gericke

## Christof Vollrath

Software- und System-Architekt
Spring-Media

Software Architect, Java Developer, Android App Creator, Kotlin Fan, Tai Chi Practitioner

christof.vollrath@spring-media.de
https://www.xing.com/profile/Christof_Vollrath
https://www.linkedin.com/in/christof-vollrath-72b3a05/
www.twitter.com/cvollrath

# Was wir machen

# Spring-Media

spring.

Spring-Media = Tochter von Axel-Springer

SPRING Axel Springer Digital News Media GmbH & Co. KG

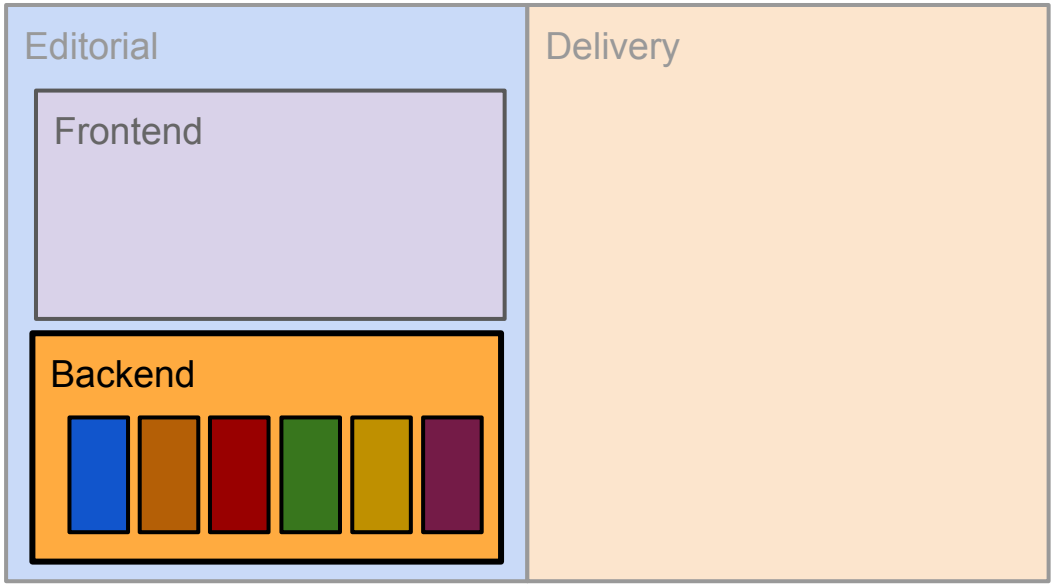
=> Digitale Produkte im Bereich News Media

Unser Team: Content Management System

Lean CMS

> 500 Redakteure

# Architektur LeanCms




Microservices, Spring Boot, Java / Kotlin, Swagger / OpenAPI, MongoDB, PostgreSQL, Elasticsearch

# LeanCms Entwicklungsprozess

- Github Flow (Branch based)
- Kurzlebige Feature Branches (½ - 3 Tage Lebensdauer)
- Continuous Deployment
- Merge triggert Deployment Pipeline -> Live
- Mehrere Deployments am Tag

**Umfangreiche automatisierte Tests!**

# Das Problem

JUnit und Cucumber

# JUnit


- Gleiche Programmiersprache wie Projekt
- Einfach nutzbar, wenig Einarbeitung
- Leicht in den Buildprozess integrierbar
- Gute Assertion-Frameworks

# JUnit - Beispiel

```
public void readDocument_for_test_article_with_fragments()   throws ApiException {
        //read existing article and add some Fragments to the existing RichText
        DocumentBaseMutable documentBaseMutable =  documentDao.findCurrentDocument( cmsIdWithTextProperty);
        assertThat(documentBaseMutable,  notNullValue());
        assertThat(documentBaseMutable.getMeta().getFragments(),   notNullValue());

        //add some Fragments to "text" Property
        ExternalLinkFragment externalLinkFragment = createExternalLinkFragment(  "1234_1", "http://www.bild.de", "http://www.bild.de/", OffsetDa
        documentBaseMutable.getMeta().addFragment( "text", externalLinkFragment);
        EmbedFragment embbedFragment = createEmbedFragment(  "1234_3", EmbedFragment.Type. BILD,"http://www.bild.de/myembed", OffsetDateTime. now
        embbedFragment.setId( "1234_3");
        embbedFragment.setModCounter( 1L);
        embbedFragment.setLastModified(OffsetDateTime.  now());
        ...
        documentBaseMutable.getMeta().addFragment( "text", embbedFragment);

        //and store
        DocumentBaseMutable storedDocument =  documentDao.updateOne(documentBaseMutable.getMeta().getId(), documentBaseMutable);
        assertNotNull(storedDocument);

        //Read and Test the Swagger API....
        de.bild.backend.contentapi.client.model.Document document =   documentApi.readDocument( cmsIdWithTextProperty.toString());
        assertThat(document,  notNullValue());
        assertThat(document.getCmsId(),  is(cmsIdWithTextProperty.toString()));
        assertThat(document.getDocumentType(),  equalTo(de.bild.backend.contentapi.client.model.Document.DocumentTypeEnum.  ARTICLE));
        assertThat(document, IsInstanceOf. instanceOf(de.bild.backend.contentapi.client.model.Article.  class));
        ...
```

# JUnit - Probleme

- Viel Boiler-Plate-Code
- Testbeschreibung in Kommentaren wegen Begrenzung Methodennamen
- Strukturierung in prepare / execute / assert nur durch Kommentare
- Übersichtlichkeit erfordert Disziplin (Auslagern in Methoden)
- IDE-Support begrenzt

# JUnit - Unabhängige -Tests ??

“Eine wichtige Eigenschaft von Tests ist, dass sie voneinander unabhängig sind. “

Java ist auch eine Insel, Christian Ullenboom (2010)
http://www.tutego.de/blog/javainsel/2010/04/junit-4-tutorial-java-tests-mit-junit/

Junit 5: “By default, test methods will be ordered using an algorithm that is deterministic but intentionally nonobvious.”

https://junit.org/junit5/docs/snapshot/user-guide/#writing-tests-test-execution-order

# Die Grenzen von JUnit - Beispiel CRUD

**C**reate

**R**ead

**U**pdate

**D**elete

# CRUD - Warum nicht so?

| | |
|---|---|
| ***C*** | **`Wenn`** ein Datensatz angelegt wird |
| ***R*** | **`Dann`** kann er gelesen werden |
| ***U*** | **`Wenn`** danach der Datensatz geändert wird |
| ***R*** | **`Dann`** kann die Änderung gelesen werden |
| ***D*** | **`Wenn`** danach der Datensatz gelöscht wurde |
| ***R*** | **`Dann`** kann er nicht mehr gelesen werden |

# Cucumber: Feature-Files

- Gut lesbare, parametrisierbare, wiederverwendbare Testschritte

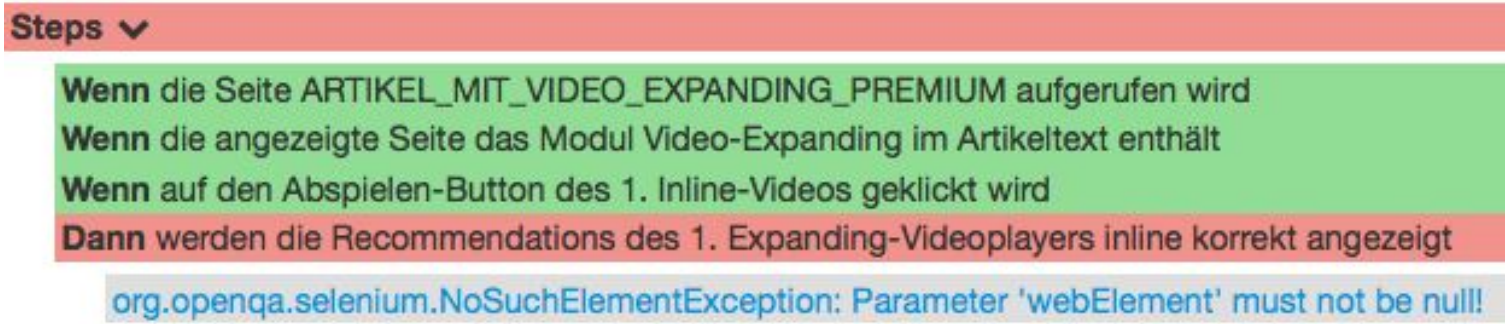
**Szenariogrundriss:** Artikel - User klickt auf Bild mit internem Verweis
**Wenn** die Seite <*Seite*> aufgerufen wird
**Wenn** die angezeigte Seite das Modul <*Modul*> im Artikeltext enthält
**Wenn** der User auf das <*Modul*> mit Verweis klickt
**Dann** öffnet sich die Seite mit der URL für <*Ziel*>

**Beispiele:**

| *Seite* | *Modul* | *Ziel* |
| --- | --- | --- |
| **ARTIKEL_MIT_BILD_VERWEIS** | **Bild** | **ULTIMA_HOME** |

# Cucumber: Report

- Keine Programmierkenntnisse nötig, um Testergebnisse zu verstehen



# Cucumber: Glue Code

- Testspezifikation und -Code nicht in derselben Datei

**Szenariogrundriss:** Artikel - User klickt auf Bild mit internem Verweis
  **Wenn** die Seite <*Seite*> aufgerufen wird
  **Wenn** die angezeigte Seite das Modul <*Modul*> im Artikeltext enthält
  **Wenn** der User auf das <*Modul*> mit Verweis klickt
  **Dann** öffnet sich die Seite mit der URL für <*Ziel*>

```
@Dann("^öffnet sich die Seite mit der URL für (\\w*)$")
public void pageForURLLoads(String page) {
   String testPageUrl = pageManager.getPageUrl(page, ProtocolType.WEB, getScenarioContext()
         .isProductionEnvironment()).trim();
   LOG.debug("URL for page " + page + " is: " + testPageUrl);
   assertNotNull("Page url for page '" + page + "' must not be null", testPageUrl);
   assertTrue("URL should be: " + testPageUrl + " but was: " + getIWebSitePO().getUrl(), getIWebSitePO().getUrl()
         .contains(testPageUrl));
}
```

# Das Problem: JUnit ↔ Cucumber

JUnit

+ Vertraut für Entwickler
- Geringe Lesbarkeit
- Keine Dokumentation der Testfälle

Cucumber

+ Gut Lesbar
+ Verständliche Dokumentation der Testfälle
- Unbeliebt bei Entwicklern
- Trennung von Spezifikation und Code

# Einschub

DSL

# DSL

Domain Specific Language

= kleine einfache Sprache, maßgeschneidert auf eine bestimmte Problemstellung

- Ausdrucksstark
- Schnell zu lernen
- Übersichtlich

# Real DSL - Beispiel Gherkin

```
Feature: Refund item
  Scenario: Jeff returns a faulty microwave
    Given Jeff has bought a microwave for $100
    And he has a receipt
    When he returns the microwave
    Then Jeff should be refunded $100
```

# Real DSL - GraphQL

```
{
  user(id: 1) {
    name
    age
    friends {
      name
    }
  }
}
```

# Keine DSL - maven

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>Maven Quick Start Archetype</name>
    <url>http://maven.apache.org</url>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.8.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

# Kuckucks DSL

Nachteile echter DSLs:

- Compiler muss gebaut werden
- IDE und Tooling
- Features fehlen
- Einarbeitungsaufwand

# Kuckucks (interne) DSL

Statt eigene Sprache zu entwickeln auf bestehende Sprache aufsetzen

- Erbt alle Features der Sprache
- Nur Delta umsetzen
- Geringere Einarbeitung
- Unterstützung durch die IDE

Anforderungen an die Sprache:

- Kompakte, mächtige Syntax
- Operator Overloading
- Lambdas

# Kuckucks DSL - Beispiele

Host-Sprache Groovy: sehr mächtig, schwach Typisiert

- Gradle (Groovy)
- Spok (Groovy)
- Gatling (Scala)

Host-Sprache Kotlin: mächtige Sprache mit strenger Typisierung

- Gradle Kotlin DSL
- Spek

# Kuckucks DSL - Warum Kotlin

- Ausdrucksstark und mächtig
- Typsicher und Null-Sicher
- Lambdas
- Operator-Overloading (eingeschränkt)
- Infix-Notation für Funktionen
- Extension Functions (fremde Klassen erweitern, auch final)
- Builder-Pattern (typsicher!) = Lambdas with Receiver
- Läuft in der JVM
- Sehr gute Integration mit Java

# Die Lösung

Spek

# Spek - Grundlegende Annahmen

- Ein grüner Test sagt nicht, dass die Spezifikation korrekt implementiert wurde
- Tests sollen prüfen…
  - ...ob Code ausgeführt wird und funktioniert
  - **...ob Code die zugrundeliegenden Anforderungen erfüllt**
- Idee: Testcode den Spezifikationen gegenüberstellen

# Spek - Specification Framework

- Bedingung - Aktion - erwartetes Ergebnis

```
object CalculatorSpec: Spek({
    given("a calculator") {
        val calculator = SampleCalculator()
        on("addition") {
            val sum = calculator.sum(2, 4)
            it("should return the result of adding the first number to the second number") {
                assertEquals(6, sum)
            }
        }
        on("subtraction") {
            val subtract = calculator.subtract(4, 2)
            it("should return the result of subtracting the second number from the first
number") {
                assertEquals(2, subtract)
            }
        }
    }
})
```

Source: http://spekframework.org/docs/latest/

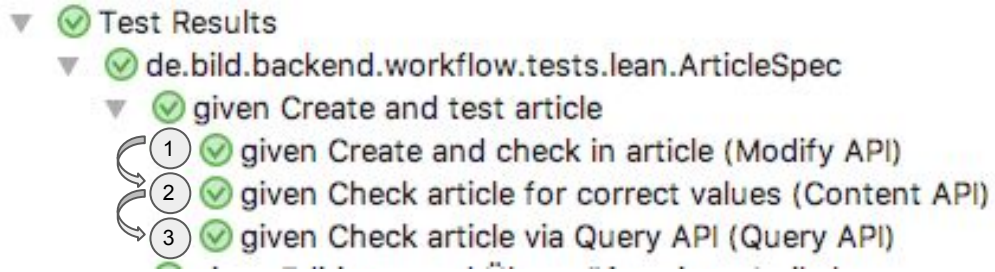
# Spek bei Spring-Media / IntelliJ Spek Plugin

```
given( description: "Create and test article") {
    var articleCmsId = ""
    val authorValue = "AutorQA"
    val metaDescriptionValue = "MetaDescription"
    val displayDateValue = "2017-03-15T23:00Z"
    val tenSecondsAgo = OffsetDateTime.now(ZoneOffset.UTC).minusSeconds( seconds: 10).toEpochSecond()
    var articleCoremediaId = ""

    given( description: "Create and check in article (Modify API)") {...}

    given( description: "Check article for correct values (Content API)") {...}
```

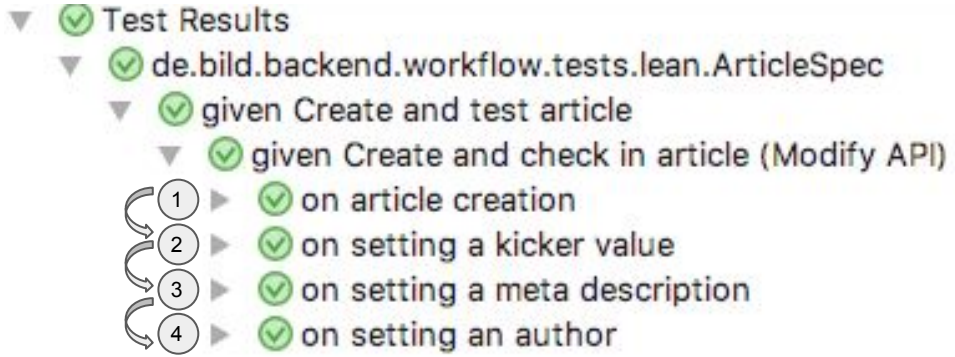
---



# Spek bei Spring-Media / IntelliJ Spek Plugin

```
given( description: "Create and check in article (Modify API)") {
    on( description: "article creation") {...}
    on( description: "setting a kicker value") {...}
    on( description: "setting a meta description") {...}
    on( description: "setting an author") {...}
```

---



# Spek bei Spring-Media / IntelliJ Spek Plugin

```
given( description: "Create and check in article (Modify API)") {

    on( description: "article creation") {
        val modifyApiResponse = createDocument(DocumentTypeEnum.ARTICLE)
        val creationResponse = modifyApiResponse.data

        it( description: "should return a newly created CMS ID") {
            creationResponse.cmsId `should not equal` null
            creationResponse.cmsId.`should not be empty`()
            articleCmsId = creationResponse.cmsId
        }
    }
```

Kluent

---

- Test Results
  - de.bild.backend.workflow.tests.lean.ArticleSpec
    - given Create and test article
      - given Create and check in article (Modify API)
        - on article creation
          - it should return a newly created CMS ID

# Gegenüberstellung **Spek** <-> JUnit

```
describe("Specification of a mutable list") {
    given("a mutable list") {
        val mutableList = mutableListOf<String>()
        it("should be empty") {
            mutableList.`should be empty`()
        }
        on("appending an element") {
            mutableList.add("element")
            it("should contain this element") {
                mutableList `should contain` "element"
            }
        }
        on("deleting this element") {
            mutableList.remove("element")
            it("should be empty again") {
                mutableList.`should be empty`()
            }
        }
    }
```

# Gegenüberstellung Spek <-> **JUnit**

```
@TestMethodOrder(OrderAnnotation.class)
class ListTest {
    List<String> mutableList = new ArrayList();

    @Test @Order(1)
    void shouldBeEmpty() {
        assertThat(mutableList, empty());
    }
    @Test @Order(2)
    void afterAdddingAnElementItShouldContainThisElement() {
        mutableList.add("element");
        assertThat(mutableList, hasItem("element"));
    }
    @Test @Order(3)
    void afterDeletingThisElementItShouldBeEmptyAgain() {
        mutableList.remove("element");
        assertThat(mutableList, empty());
    }
}
```

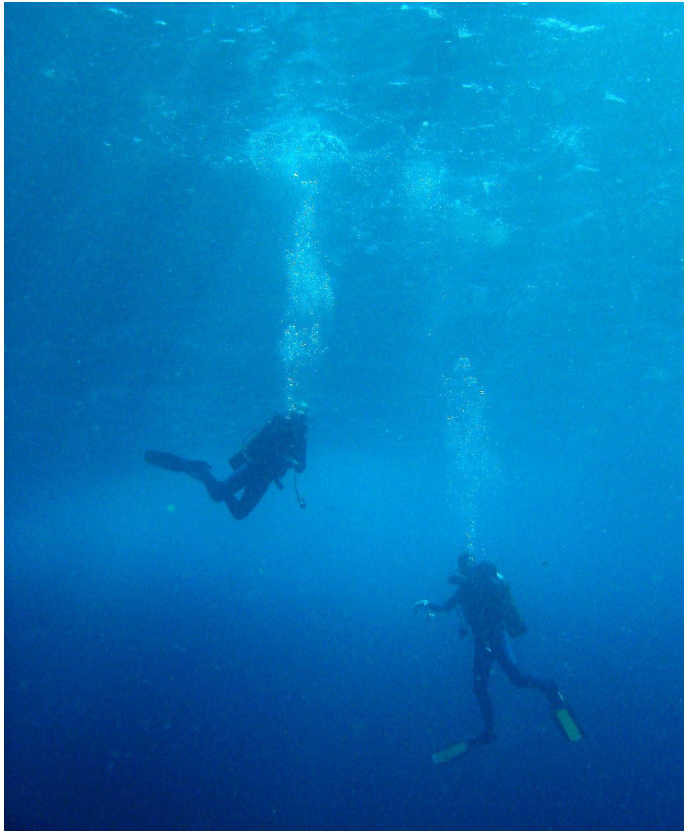
# Gegenüberstellung Kluent <-> Hamcrest

```
list `should equal` listOf(1, 2, 3)
assertThat(list, equalTo(asList(1, 2, 3)))

list `should contain` 1
assertThat(list, hasItem(1))

"Kluent is great!" `should match` """[Kk]luent.*is\sgreat!"""
assertThat("Hamcrest also", matchesPattern("Hamcrest\\salso"));
```

# Programmieren = Tiefseetauchen



Software  Java, Kotlin

Zwischenstopp:
Spezifikation mit Spek

Fachdomäne

# Spek und Spring Framework

Programmiermodelle passen nicht gut zusammen - Spring Initialisierung!

- Akzeptanztests ✔
- Spring-Integrationstests ✖
- Integrationstests (Spring-Light) ✔
- Unit-Tests ✔

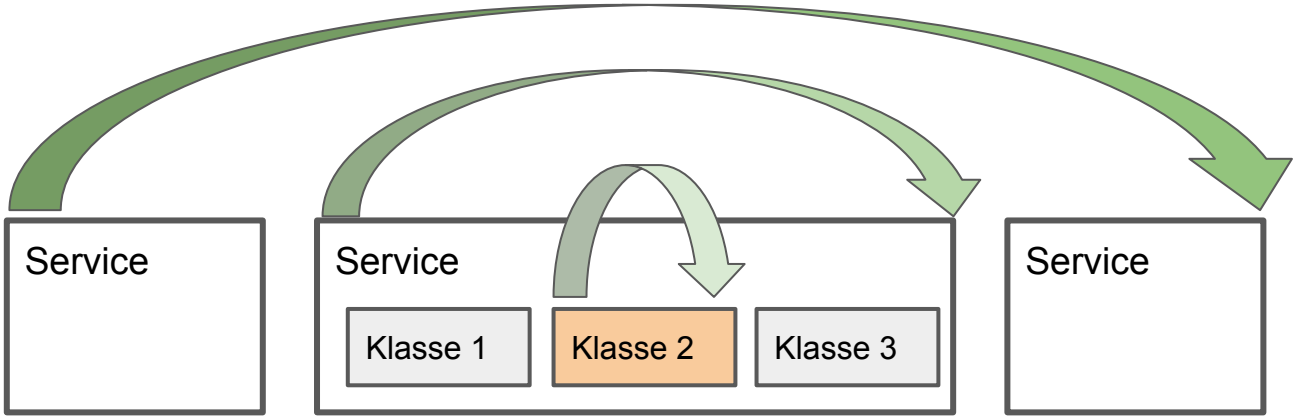
Wie kann man damit umgehen:

- Spring-Light
- KotlinTest
- Kein Spring: Ktor, kein Spring Data JPA: Expose

# Entwicklungsprozess

mit Spek

# Geschachtelte Test-Zyklen




[3] Akzeptanz-Tests -> Dev-Environment
[1] Integrations-Tests -> Embedded Datenbank
[2] Unit-Tests

# Unsere Deployment Pipeline

| clone repository | build docker image | clone repository | init kubetools | deploy dev | run workflow tests | deploy stg | deploy prd |
|---|---|---|---|---|---|---|---|
| 5s | 1min 44s | NaNy NaNd | 8s | 1min 7s | 3min 22s | 1min 25s | 1min 20s |
| 4s | 1min 45s | 5s | 6s | 1min 53s | 3min 3s | 51s | 51s |

# Zusammenfassung

- Keine Cucumber-Tests mehr
- Tests sind selbsterklärend und wartbar
- Entwickler verstehen die Akzeptanztests und entwickeln sie weiter
- Tester und Entwickler sind glücklich
- ~~Störrische~~ Java-Entwickler
- Neue Projekte: Kotlin

# Ausblick

Spek 2

- Multiplatform-Unterstützung (JVM, JS, Native)
- Änderungen im Teststyle (Breaking Changes)

JUnit 5

- @DisplayName, @Nested

Kotlin-Test

- Kann jetzt Given/When/Then
- Spring Framework Unterstützung (Spring-Extension)
- Data-Driven-Tests, Property-Based-Tests

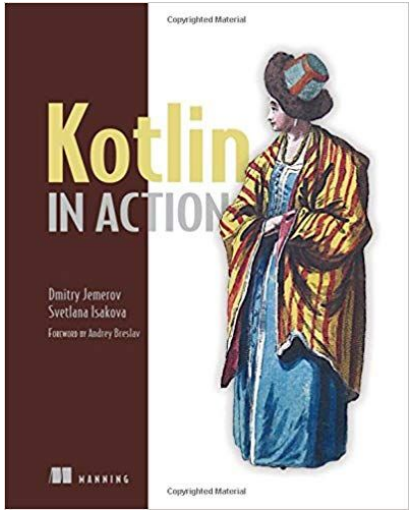
# Links und Literatur

https://www.spring-media.de/

https://spekframework.org/

https://github.com/kotlintest/kotlintest

https://kotlinlang.org/ https://play.kotlinlang.org




Kotlin in Action, Dmitry Jemerov, Svetlana Isakova