

JUG Ostfalen, 26.09.2024

Ich packe meinen Testtoolkoffer und nehme  
mit...  
Testwerkzeuge für den Entwickleralltag

Sandra Parsick  
@SandraParsick  
 [@sparsick@mastodon.social](mailto:@sparsick@mastodon.social)  
 [mail@sandra-parsick.de](mailto:mail@sandra-parsick.de)

# Wer bin ich?

- Sandra Parsick
- Freiberuflicher Softwareentwickler und Consultant im Java-Umfeld
- Schwerpunkte:
  - Java Enterprise Anwendungen
  - Agile Methoden
  - Software Craftmanship
  - Automatisierung von Entwicklungsprozessen
- Trainings
- Workshops

✉️ mail@sandra-parsick.de

Ⓜ️ @sparsick@mastodon.social

RSS https://www.sandra-parsick.de

🎧 https://ready-for-review.dev





Das kann man  
lokal nicht testen.

Das ist zu  
aufwendig zu testen.

Das kann man  
nicht testen.



**CHALLENGE ACCEPTED**

# Agenda

Integrierte Tests

equals, hashCode,  
toString

Testcode-  
Duplikierung

Testdaten

Concurrency

Schlecht lesbare  
Assertions

Tests, die nur unter bestimmten  
Bedingungen laufen

Problem: Diese Tests können nur unter bestimmten Bedingungen laufen.

# JUnit 5 – Conditional Test Execution

```
3 // Operating System Conditions
4
5 @Test
6 @EnabledOnOs(OS.LINUX)
7 void testForLinux(){
8     assertTrue(true);
9 }
10
11 @Test
12 @EnabledOnOs(OS.WINDOWS)
13 void testForWindows(){
14     assertTrue(true);
15 }
16
17 @Test
18 @EnabledOnOs({OS.LINUX, OS.WINDOWS})
19 void testForLinuxAndWindows(){
20     assertTrue(true);
21 }
22
23 @Test
24 @DisabledOnOs(OS.LINUX)
25 void testNotForLinux(){
26     assertTrue(true);
27 }
```

```
29 // Java Runtime Environment Conditions
30
31 @Test
32 @EnabledOnJre(JRE.JAVA_8)
33 void onlyOnJava8() {
34     assertTrue(true);
35 }
36
37 @Test
38 @EnabledOnJre({ JRE.JAVA_9, JRE.JAVA_10 })
39 void onJava9Or10() {
40     assertTrue(true);
41 }
42
43 @Test
44 @DisabledOnJre(JRE.JAVA_8)
45 void notOnJava8() {
46     assertTrue(true);
47 }
```

# JUnit 5 – Conditional Test Execution

```
50     // System Property Conditions
51
52     @Test
53     @EnabledIfSystemProperty(named = "os.arch", matches = ".*64.*")
54     void onlyOn64BitArchitectures() {
55         assertTrue(true);
56
57     }
58
59     @BeforeEach
60     void setup(){
61         System.setProperty("ci-server", "true");
62     }
63
64     @Test
65     @DisabledIfSystemProperty(named = "ci-server", matches = "true")
66     @EnabledIfSystemProperty(named = "ci-server", matches = "true")
67     void notOnCiServer() {
68         assertTrue(true);
69     }
```

# JUnit 5 – Conditional Test Execution

```
71     // Environment Variable Conditions
72     // variable can be set in idea run configuration
73     // ENV1=staging-server
74     // ENV2=local.development
75
76     @Test
77     @EnabledIfEnvironmentVariable(named = "ENV1", matches = "staging-server")
78     void onlyOnStagingServer() {
79         assertTrue(true);
80     }
81
82     @Test
83     @DisabledIfEnvironmentVariable(named = "ENV2", matches = ".*development.*")
84     void notOnDeveloperWorkstation() {
85         assertTrue(true);
86     }
87
```

# Problem: Testcode Duplizierung

```
public class Position {  
  
    private int xCoordinate;  
    private int yCoordinate;  
  
    public Position(int xCoordinate, int yCoordinate) {  
        this.xCoordinate = xCoordinate;  
        this.yCoordinate = yCoordinate;  
    }  
  
    public Position moveBackward(Direction facingDirection) {  
        int newX = xCoordinate;  
        int newY = yCoordinate;  
        switch (facingDirection) {  
            case NORTH:  
                newY--;  
                break;  
            case SOUTH:  
                newY++;  
                break;  
            case WEST:  
                newX++;  
                break;  
            case EAST:  
                newX--;  
                break;  
        }  
        return new Position (newX, newY);  
    }  
  
    public Position moveForward(Direction facingDirection) {  
        int newX = xCoordinate;  
        int newY = yCoordinate;  
  
        switch (facingDirection) {  
            case NORTH:  
                newY++;  
                break;  
            case SOUTH:  
                newY--;  
                break;  
            case WEST:  
                newX--;  
                break;  
            case EAST:  
                newX++;  
                break;  
        }  
  
        return new Position(newX, newY);  
    }  
}
```

# Naiver Ansatz

```
7  class PositionNaiveTest {  
8  
9      @Test  
10     void moveBackward_toNorth() {  
11         Position positionUnderTest = new Position(10, 10);  
12  
13         Position newPosition = positionUnderTest.moveBackward(Direction.NORTH);  
14  
15         assertThat(newPosition).isEqualTo(new Position(10, 9));  
16     }  
17  
18     @Test  
19     void moveBackward_toEast() {  
20         Position positionUnderTest = new Position(10, 10);  
21  
22         Position newPosition = positionUnderTest.moveBackward(Direction.EAST);  
23  
24         assertThat(newPosition).isEqualTo(new Position(9, 10));  
25     }  
26 }
```

```
~~~  
27     @Test  
28     void moveBackward_toSouth() {  
29         Position positionUnderTest = new Position(10, 10);  
30  
31         Position newPosition = positionUnderTest.moveBackward(Direction.SOUTH);  
32  
33         assertThat(newPosition).isEqualTo(new Position(10, 11));  
34     }  
35  
36     @Test  
37     void moveBackward_toWest() {  
38         Position positionUnderTest = new Position(10, 10);  
39  
40         Position newPosition = positionUnderTest.moveBackward(Direction.WEST);  
41  
42         assertThat(newPosition).isEqualTo(new Position(11, 10));  
43     }  
44 }  
45
```

# Parametrisierte Tests mit JUnit 4

```
1 @RunWith(Parameterized.class)
2 public class PositionMoveBackwardJUnit4Test {
3
4     private final Direction direction;
5     private final Position expectedPosition;
6
7     public PositionMoveBackwardJUnit4Test(Direction direction, Position expectedPosition) {
8         this.direction = direction;
9         this.expectedPosition = expectedPosition;
10    }
11
12    @Parameterized.Parameters
13    public static Collection moveBackwardParameter() {
14        return Arrays.asList(new Object[][]{
15            {Direction.NORTH, new Position(10, 9)},
16            {Direction.EAST, new Position(9, 10)},
17            {Direction.SOUTH, new Position(10, 11)},
18            {Direction.WEST, new Position(11, 10)}
19        });
20    }
21
22    @Test
23    public void moveBackward() {
24        Position positionUnderTest = new Position(10, 10);
25        Position newPosition = positionUnderTest.moveBackward(direction);
26        assertThat(newPosition).isEqualTo(expectedPosition);
27    }
28 }
```

```
1 @RunWith(Parameterized.class)
2 public class PositionMoveForwardJUnit4Test {
3
4     private final Direction direction;
5     private final Position expectedPosition;
6
7     public PositionMoveForwardJUnit4Test(Direction direction, Position expectedPosition) {
8         this.direction = direction;
9         this.expectedPosition = expectedPosition;
10    }
11
12    @Parameterized.Parameters
13    public static Collection moveBackwardParameter() {
14        return Arrays.asList(new Object[][]{
15            {Direction.NORTH, new Position(10, 11)},
16            {Direction.EAST, new Position(11, 10)},
17            {Direction.SOUTH, new Position(10, 9)},
18            {Direction.WEST, new Position(9, 10)}
19        });
20    }
21
22    @Test
23    public void moveForward() {
24        Position positionUnderTest = new Position(10, 10);
25
26        Position newPosition = positionUnderTest.moveForward(direction);
27
28        assertThat(newPosition).isEqualTo(expectedPosition);
29    }
30}
```

# Parametrisierte Test mit JUnit5

```
1 class PositionJUnit5Test {
2
3     @ParameterizedTest
4     @MethodSource("createMoveBackwardParameter")
5     void moveBackward(Direction direction, Position expectedPosition) {
6         Position positionUnderTest = new Position(10, 10);
7         Position newPosition = positionUnderTest.moveBackward(direction);
8         assertThat(newPosition).isEqualTo(expectedPosition);
9     }
10
11    private static Stream<Arguments> createMoveBackwardParameter() {
12        return Stream.of(
13            Arguments.of(Direction.NORTH, new Position(10, 9)),
14            Arguments.of(Direction.EAST, new Position(9, 10)),
15            Arguments.of(Direction.SOUTH, new Position(10, 11)),
16            Arguments.of(Direction.WEST, new Position(11, 10))
17        );
18    }
19
20    @ParameterizedTest
21    @MethodSource("createMoveForwardParameter")
22    void moveForward(Direction direction, Position expectedPosition) {
23        Position positionUnderTest = new Position(10, 10);
24        Position newPosition = positionUnderTest.moveForward(direction);
25        assertThat(newPosition).isEqualTo(expectedPosition);
26    }
27
28    private static Stream<Arguments> createMoveForwardParameter() {
29        return Stream.of(
30            Arguments.of(Direction.NORTH, new Position(10, 11)),
31            Arguments.of(Direction.EAST, new Position(11, 10)),
32            Arguments.of(Direction.SOUTH, new Position(10, 9)),
33            Arguments.of(Direction.WEST, new Position(9, 10))
34        );
35    }
36}
```

# Weitere Möglichkeiten in JUnit5

```
@ParameterizedTest
@ValueSource(strings = {"a", "b", "c"})
void singleLetter(String candidate) {
    assertTrue(isSingleLetter(candidate));
}

private boolean isSingleLetter(String candidate) {
    return candidate.length() == 1;
}

// type for value sources:
// short
// byte
// int
// long
// float
// double
// char
// java.lang.String
// java.lang.Class

@ParameterizedTest
@EnumSource(TimeUnit.class)
void timeUnit(TimeUnit timeUnit) {
    assertThat(timeUnit).isNotNull();
}
```

```
@ParameterizedTest
@CsvSource({ "foo, 1", "bar, 2",
            "'baz, qux', 3" })
// @CsvFileSource(resources = "/test-data.csv", numLinesToSkip = 1)
void testWithCsvSource(String first, int second) {
    assertNotNull(first);
    assertEquals(0, second);
}
```

```
@ParameterizedTest
@ArgumentsSource(MyArgumentsProvider.class)
void testWithArgumentsSource(String argument) {
    assertNotNull(argument);
}
```

---

```
public class MyArgumentsProvider implements ArgumentsProvider {
    @Override
    public Stream<? extends Arguments> provideArguments(
            ExtensionContext extensionContext) {
        return Stream.of("foo", "bar").map(Arguments::of);
    }
}
```

# Parametrisierte Test mit Spock

```
1 class PositionSpockTest extends Specification {
2
3     def "move backward" (Direction direction, Position expectedPosition){
4         Position positionUnderTest = new Position(10, 10)
5         Position newPosition = positionUnderTest.moveBackward(direction)
6
7         expect:
8             assert newPosition == expectedPosition
9
10        where:
11            direction | expectedPosition
12            Direction.NORTH | new Position(10, 9)
13            Direction.EAST | new Position(9, 10)
14            Direction.SOUTH | new Position(10, 11)
15            Direction.WEST | new Position(11, 10)
16    }
17
18    def "move forward" (Direction direction, Position expectedPosition){
19        Position positionUnderTest = new Position(10, 10)
20        Position newPosition = positionUnderTest.moveForward(direction)
21
22        expect:
23            assert newPosition == expectedPosition
24
25        where:
26            direction | expectedPosition
27            Direction.NORTH | new Position(10, 11)
28            Direction.EAST | new Position(11, 10)
29            Direction.SOUTH | new Position(10, 9)
30            Direction.WEST | new Position(9, 10)
31    }
32}
```

# Problem: Testdaten

<https://www.generatedata.com>



	Data Type	Property Name	Examples	Options
1	Names	<input type="text" value="name"/>	John Smith	<input type="text" value="MaleName Surname"/>
2	Phone / Fax	<input type="text" value="phone"/>	Germany	<input type="text" value="(0xx) xxxxxxxx"/> <input type="text" value="(0xxx) xxxxxxxx"/> <input type="text" value="(0xxxx) xxxxxxxx"/> <input type="text" value="(03xxxx) xxxxxxxx"/>
3	Fixed Number of Words	<input type="text" value="text"/>	No examples available.	Generate <input type="button" value="13"/> WORDS

Add

JSON

```
1 [
2   {
3     "name": "Judah Jacobson",
4     "phone": "(04027) 6123545",
5     "text": "odio, auctor vitae, aliquet nec, imperdiet nec, leo. Morbi neque tellus, imperdiet non,"
6   },
7   {
8     "name": "Oren Cooper",
9     "phone": "(0563) 84983771",
10    "text": "vehicula risus. Nulla eget metus eu erat semper rutrum. Fusce dolor quam, elementum"
11  },
12  {
13    "name": "Kibo Gardner",
14    "phone": "(0947) 60752555",
15    "text": "eget, ipsum. Donec sollicitudin adipiscing ligula. Aenean gravida nunc sed pede. Cum sociis"
16  },
17  {
18    "name": "George Dalton",
19    "phone": "(07746) 9809882",
20    "text": "consectetuer euismod est arcu ac orci. Ut semper pretium neque. Morbi quis urna."
21  },
22  {
23    "name": "Ishmael Giles",
```



Help

Contact



GRID



PREVIEW



SAVE



GENERATE

## Daten-Typen auswählen

### Human-Daten

Namen

Namen, regional

Telefon / Fax

Telefon / Fax, Regional

E-mail

Datum

Firma

SIRET

Chilean RUT number

Persönliche Nummer

Organisation Anzahl

### Geo

Straße

City

Postleitzahl / PLZ

Staat / Provinz / Kreis

Land

Breite / Länge

PIN

CW

Track 1

Track 2

### Text

Feste Anzahl der Worte

Beliebige Anzahl von Wörtern

### Numerisch

Alphanumerisch

Auto-Inkrement

Number Range

GUID

Currency

### Mathe

Standardabweichung

### Andere

Konstante

Zusammengesetzt

Tree (übergeordnete Zeile ID)

Benutzerdefinierte Liste

SETTINGS

PREVIEW



Format

SQL



Database table name

myTable

Misc Options



Statement Type

 INSERT INSERT IGNORE UPDATE

batch size 10

```
1  DROP TABLE IF EXISTS `myTable`;
2
3  CREATE TABLE `myTable` (
4      `id` mediumint(8) unsigned NOT NULL auto_increment,
5      `name` varchar(255) default NULL,
6      `phone` varchar(100) default NULL,
7      `text` TEXT default NULL,
8      PRIMARY KEY (`id`)
9  ) AUTO_INCREMENT=1;
10
11  INSERT INTO `myTable` (`name`, `phone`, `text`)
12  VALUES
13      ("Dane Mann", "(0697) 69336736", "Phasellus at augue id ante dictum cursus. Nunc mauris elit, dictum eu, eleifend"),
14      ("Jonah Newton", "(06845) 5796135", "nibh vulputate mauris sagittis placerat. Cras dictum ultricies ligula. Nullam enim. Sed nulla"),
15      ("Rigel Turner", "(099) 73612680", "lobortis ultrices. Vivamus rhoncus. Donec est. Nunc ullamcorper, velit in aliquet lobortis, nisi"),
16      ("Macaulay Cervantes", "(03373) 065114", "molestie pharetra nibh. Aliquam ornare, libero at auctor ullamcorper, nisl arcu iaculis enim,"),
17      ("Steven Gross", "(0871) 35898863", "erat volutpat. Nulla facilisis. Suspenisse commodo tincidunt nibh. Phasellus nulla. Integer vulputate, risus"),
18      ("Berk Burris", "(08061) 2623971", "accumsan sed, facilisis vitae, orci. Phasellus dapibus quam quis diam. Pellentesque habitant morbi"),
19      ("Scott Thomas", "(031838) 380727", "cursus vestibulum. Mauris magna. Duis dignissim tempor arcu. Vestibulum ut eros non enim"),
20      ("Malik Durham", "(042) 66835127", "id ante dictum cursus. Nunc mauris elit, dictum eu, eleifend nec, malesuada ut,"),
21      ("Alden House", "(03426) 3667808", "tristique ac, eleifend vitae, erat. Vivamus nisi. Mauris nulla. Integer urna. Vivamus molestie"),
22      ("Joel Barnett", "(0417) 76425813", "In mi pede, nonummy ut, molestie in, tempus eu, ligula. Aenean euismod mauris");
23  INSERT INTO `myTable` (`name`, `phone`, `text`)
24  VALUES
25      ("Brody Horton", "(0663) 39334541", "ac turpis egestas. Fusce aliquet magna a neque. Nullam ut nisi a odio"),
26      ("Dominic Griffith", "(037635) 785644", "eu nibh vulputate mauris sagittis placerat. Cras dictum ultricies ligula. Nullam enim. Sed"),
27      ("Sylvester Meyer", "(031251) 571434", "nibh dolor, nonummy ac, feugiat non, lobortis quis, pede. Suspenisse dui. Fusce diam"),
28      ("Raphael Weeks", "(0086) 54448754", "sed pede. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus"),
29      ("Graham Dejesus", "(035582) 121633", "Suspenisse aliquet molestie tellus. Aenean egestas hendrerit neque. In ornare sagittis felis. Donec"),
30      ("Caesar Pearson", "(0108) 48343683", "Sed auctor odio a purus. Duis elementum, dui quis accumsan convallis, ante lectus"),
31      ("Asher Leblanc", "(0253) 98561128", "vulputate, risus a ultricies adipiscing, enim mi tempor lorem, eget mollis lectus pede"),
32      ("Gage McGowan", "(082) 27065449", "aliquet libero. Integer in magna. Phasellus dolor elit, pellentesque a, facilisis non, bibendum"),
33      ("Cole Durham", "(016) 40512714", "dignissim. Maecenas ornare egestas ligula. Nullam feugiat placerat velit. Quisque varius. Nam porttitor"),
34      ("Orlando Andrews", "(033324) 111567", "sem. Pellentesque ut ipsum ac mi eleifend egestas. Sed pharetra, felis eget varius");
35
```

CLOSE PANEL

# Mit LLM Modelle?

Generiere mir bitte Testdaten mit Adressen aus den USA?



Natürlich! Hier sind einige Beispieladressen aus den USA, die du als Testdaten verwenden kannst:

1. **John Doe** 123 Maple Street Springfield, IL 62704 USA
2. **Jane Smith** 456 Oak Avenue Los Angeles, CA 90001 USA
3. **Michael Johnson** 789 Pine Road Dallas, TX 75201 USA
4. **Emily Davis** 101 Elm Boulevard New York, NY 10001 USA
5. **William Brown** 202 Cedar Lane Miami, FL 33101 USA
6. **Olivia Wilson** 303 Birch Drive Seattle, WA 98101 USA
7. **David Miller** 404 Walnut Court Denver, CO 80201 USA
8. **Sophia Anderson** 505 Cherry Street San Francisco, CA 94101 USA
9. **James Taylor** 606 Maplewood Terrace Chicago, IL 60601 USA
10. **Isabella Martinez** 707 Pinecrest Way Austin, TX 73301 USA

Falls du spezifischere Details benötigst oder besondere Anforderungen an die Testdaten hast, lass es mich wissen!

Wandele sie mir in JSON um



Natürlich! Hier sind die oben genannten Adressen im JSON-Format:

json

Code kopieren

```
[  
  {  
    "name": "John Doe",  
    "address": {  
      "street": "123 Maple Street",  
      "city": "Springfield",  
      "state": "IL",  
      "zip": "62704",  
      "country": "USA"  
    }  
  },  
  {  
    "name": "Jane Smith",  
    "address": {  
      "street": "456 Oak Avenue",  
      "city": "Los Angeles",  
      "state": "CA",  
      "zip": "90001",  
      "country": "USA"  
    }  
  },  
  {  
    "name": "Michael Johnson",  
    "address": {  
      "street": "789 Pine Street",  
      "city": "Phoenix",  
      "state": "AZ",  
      "zip": "85001",  
      "country": "USA"  
    }  
  }]
```

Und wenn es doch lieber Code sein soll?

# Testdaten mit DataFaker

```
@Test  
void simpleFaker() {  
    Faker dataFaker = new Faker();  
    Person person = new Person();  
    person.setFirstName(dataFaker.name().firstName());  
    person.setLastName(dataFaker.name().lastName());  
    person.setJobTitle(dataFaker.job().title());  
  
    // more test code  
}
```

- Built-In Faker (Auszug)
  - Business
  - Commerce
  - Name
  - Lorem
- Local Support (Auszug)
  - de (in vielen Varianten)
  - en (in vielen Varianten)

# ObjectMother Pattern

```
class PersonTestData { // ObjectMother pattern
    static Person newPersonWithoutJobTitle() {
        Faker dataFaker = new Faker();
        Person person = new Person();
        person.setFirstName(dataFaker.name().firstName());
        person.setLastName(dataFaker.name().lastName());
        return person;
    }

    static Person newPerson() {
        Faker dataFaker = new Faker();
        Person person = new Person();
        person.setFirstName(dataFaker.name().firstName());
        person.setLastName(dataFaker.name().lastName());
        person.setJobTitle(dataFaker.job().title());
        return person;
    }
}
```

```
@Test
void objectMother(){
    Person personWithoutJobTitle =
        PersonTestData.newPersonWithoutJobTitle();
    Person fullPerson =
        PersonTestData.newPerson();

    //more test code
}
```

# TestDataBuilder Pattern

```
class PersonTestDataBuilder { // test data builder pattern
    private String firstName;
    private String lastName;
    private String jobTitle;

    PersonTestDataBuilder withFirstName(String firstName){
        this.firstName = firstName;
        return this;
    }

    PersonTestDataBuilder withLastName(String lastName){
        this.lastName = lastName;
        return this;
    }

    PersonTestDataBuilder withJobTitle (String jobTitle){
        this.jobTitle = jobTitle;
        return this;
    }

    Person build() {
        Person person = new Person();
        person.setFirstName(firstName);
        person.setLastName(lastName);
        person.setJobTitle(jobTitle);
        return person;
    }
}
```

```
@Test
void testDataBuilder(){
    Faker dataFaker = new Faker();
    PersonTestDataBuilder personBuilder = new PersonTestDataBuilder();
    personBuilder.withFirstName(dataFaker.name().firstName())
        .withLastName(dataFaker.name().lastName())
        .withJobTitle(dataFaker.job().title());
    Person person = personBuilder.build();
}
```

Problem: Schlecht lesbare Assertion

# Built-In Assertion in JUnit

```
@Test  
void builtInAssertion() {  
    Hero hero = new Hero("Batman", "Bruce Wayne");  
  
    assertEquals(hero.getName(), "Batman");  
    assertEquals(hero.getRealName(), "Bruce Wayne");  
}
```

# AssertJ – Fluent Assertion API

```
@Test
void hasField(){
    assertThat(new Hero ("Superman", "Clark Kent"))
        .hasFieldOrPropertyWithValue("realName", "Clark Kent")
        .hasFieldOrPropertyWithValue("name", "Superman");
}
```

```
@Test
void basic() {
    assertThat("The Lord of the Rings").isNotNull()
        .startsWith("The")
        .contains("Lord")
        .endsWith("Rings");
}
```

# AssertJ – List Assertion

```
@Test
void basic(){
    List<String> heros = List.of("Batman", "Superman");

    assertThat(heros)
        .hasSize(2)
        .contains("Batman")
        .containsExactly("Batman", "Superman")
        .containsAnyOf("Batman", "Superman", "Wonder woman");

    // containsNull
    // negotiation
}

@Test
void extractionSample(){
    List<Hero> heros = List.of(new Hero("Batman", "Bruce Wayne"), new Hero("Superman", "Clark Kent"));

    assertThat(heros)
        .extracting("realName")
        .contains("Bruce Wayne", "Clark Kent");
}
```

# AssertJ – Exception Assertion

```
@Test
void bddStyle(){
    // GIVEN
    String[] names = { "Pier ", "Pol", "Jak" };
    // WHEN
    Throwable thrown = catchThrowable(() -> System.out.println(names[9]));
    // THEN
    assertThat(thrown)
        .isInstanceOf(ArrayIndexOutOfBoundsException.class)
        .hasMessageContaining("9");
}

@Test
void assertThatThrownByExample(){
    assertThatThrownBy(() -> { throw new Exception("boom!"); })
        .isInstanceOf(Exception.class)
        .hasMessageContaining("boom")
        .hasMessage("boom!");
}
```

# AssertJ – Exception Assertion

```
@Test
void assertThatExceptionOfTypeExample(){
    assertThatExceptionOfType(IOException.class)
        .isThrownBy(() -> { throw new IOException("boom!"); })
        .withMessage("%s!", "boom")
        .withMessageContaining("boom")
        .withNoCause();

    //This later syntax has been enriched for common exceptions :
    //    assertThatNullPointerException
    //    assertThatIllegalArgumentException
    //    assertThatIllegalStateException
    //    assertThatIOException
}

@Test
void assertThatNoExceptionIsThrown(){
    assertThatCode(() -> {
        // code that should NOT throw an exception
    }).doesNotThrowAnyException();
}
```

# AssertJ - Assumption

```
@Test
void assume(){
    assumeThat("Bonn").isEqualTo("Bonn");
}

@Test
void assumeMoreComplex(){
    assumeThat(new File("/starwars-testdata/star-wars-logo.png"))
        .isFile()
        .exists();
    assumeThat(List.of("Hello", "World"))
        .hasSize(2)
        .contains("Hello");
}
```

# Speziell für DB-Tests: AssertJ-DB

```
@Test
void savePerson() {
    Person person = new Person();
    person.setFirstName("Matt");
    person.setLastName("Clark");
    person.setJobTitle("actor");

    repositoryUnderTest.save(person);

    var personTable = new Table(dataSource, "person");
    assertThat(personTable)
        .column("first_name").containsValues("Matt")
        .column("last_name").containsValues("Clark")
        .column("job_title").containsValues("actor");
}
```

Gibt es auch speziell  
für Neo4j

# JUnit 5 – Grouped Assertion

```
@Test
void standardAssertions() {
    assertEquals(2, 2);
    assertEquals(4, 4, "The optional assertion message is now the last parameter.");
    assertTrue('a' < 'b', () -> "Assertion messages can be lazily evaluated -- "
               + "to avoid constructing complex messages unnecessarily.");
}

@Test
void groupedAssertions() {
    // In a grouped assertion all assertions are executed, and any
    // failures will be reported together.
    assertAll("person",
              () -> assertEquals("John", "John"),
              () -> assertEquals("Doe", "Doe"),
              () -> assertThat(Lists.list("foo")).isNotEmpty());
}
```

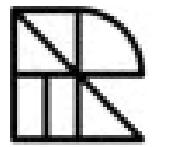
# Exkurs: Migration auf JUnit 5

```
<properties>
    <junit.jupiter.version>5.4.2</junit.jupiter.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-engine</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.junit</groupId>
            <artifactId>junit-bom</artifactId>
            <version>${junit.jupiter.version}</version>
            <scope>import</scope>
            <type>pom</type>
        </dependency>
    </dependencies>
</dependencyManagement>
```

# Code Migration mit OpenRewrite



# Rewrite

Large-Scale Automated Source Code  
Refactoring



```
<plugin>
  <groupId>org.openrewrite.maven</groupId>
  <artifactId>rewrite-maven-plugin</artifactId>
  <version>4.38.0</version>
  <configuration>
    <activeRecipes>
      <recipe>org.openrewrite.java.spring.boot2.SpringBoot2JUnit4to5Migration</recipe>
    </activeRecipes>
  </configuration>
  <dependencies>
    <dependency>
      <groupId>org.openrewrite.recipe</groupId>
      <artifactId>rewrite-spring</artifactId>
      <version>4.29.0</version>
    </dependency>
  </dependencies>
</plugin>
```



```
$ mvn rewrite:run
[INFO] Using active recipe(s) [org.openrewrite.java.spring.boot2.SpringBoot2JUnit4to5Migration]
[INFO] Using active styles(s) []
[INFO] Validating active recipes...
[INFO] Project [petclinic] Resolving Poms...
[INFO] Project [petclinic] Parsing Source Files
[INFO] Running recipe(s)...
[WARNING] Changes have been made to pom.xml by:
[WARNING]   org.openrewrite.java.spring.boot2.SpringBoot2JUnit4to5Migration
[WARNING]     org.openrewrite.java.testing.junit5.JUnit4to5Migration
[WARNING]       org.openrewrite.maven.ExcludeDependency: {groupId=junit, artifactId=junit}
[WARNING]       org.openrewrite.maven.ExcludeDependency: {groupId=org.junit.vintage, artifactId=junit-vintage-engine}
[WARNING] Changes have been made to src/test/java/org/springframework/samples/petclinic/owner/OwnerControllerTests.java by:
[WARNING]   org.openrewrite.java.spring.boot2.SpringBoot2JUnit4to5Migration
[WARNING]     org.openrewrite.java.testing.junit5.JUnit4to5Migration
[WARNING]       org.openrewrite.java.testing.junit5.UpdateBeforeAfterAnnotations
[WARNING]       org.openrewrite.java.testing.junit5.UpdateTestAnnotation
[WARNING]     org.openrewrite.java.spring.boot2.UnnecessarySpringRunWith
[WARNING]       org.openrewrite.java.testing.junit5.RunnerToExtension: {runners=
[org.springframework.test.context.junit4.SpringRunner, org.springframework.test.context.junit4.SpringJUnit4ClassRunner],
extension=org.springframework.test.context.junit.jupiter.SpringExtension}
[WARNING]       org.openrewrite.java.spring.boot2.UnnecessarySpringExtension
[WARNING] Changes have been made to src/test/java/org/springframework/samples/petclinic/owner/VisitControllerTests.java by:
[WARNING]   org.openrewrite.java.spring.boot2.SpringBoot2JUnit4to5Migration
[WARNING]     org.openrewrite.java.testing.junit5.JUnit4to5Migration
[WARNING]       org.openrewrite.java.testing.junit5.UpdateBeforeAfterAnnotations
[WARNING]       org.openrewrite.java.testing.junit5.UpdateTestAnnotation
[WARNING]     org.openrewrite.java.spring.boot2.UnnecessarySpringRunWith
[WARNING]       org.openrewrite.java.testing.junit5.RunnerToExtension: {runners=
[org.springframework.test.context.junit4.SpringRunner, org.springframework.test.context.junit4.SpringJUnit4ClassRunner],
extension=org.springframework.test.context.junit.jupiter.SpringExtension}
[WARNING]       org.openrewrite.java.spring.boot2.UnnecessarySpringExtension
```

```
[INFO] --- org.springframework:spring-framework-test-context:junit:jupiter:spring-testContext-junit5
```

```
[WARNING]      org.openrewrite.java.spring.boot2.UnnecessarySpringExtension
[WARNING] Changes have been made to src/test/java/org/springframework/samples/petclinic/vet/VetControllerTests.java by:
[WARNING]      org.openrewrite.java.spring.boot2.SpringBoot2JUnit4to5Migration
[WARNING]      org.openrewrite.java.testing.junit5.JUnit4to5Migration
[WARNING]      org.openrewrite.java.testing.junit5.UpdateBeforeAfterAnnotations
[WARNING]      org.openrewrite.java.testing.junit5.UpdateTestAnnotation
[WARNING]      org.openrewrite.java.spring.boot2.UnnecessarySpringRunWith
[WARNING]          org.openrewrite.java.testing.junit5.RunnerToExtension: {runners=
[org.springframework.test.context.junit4.SpringRunner, org.springframework.test.context.junit4.SpringJUnit4ClassRunner],
extension=org.springframework.test.context.junit.jupiter.SpringExtension}
[WARNING]      org.openrewrite.java.spring.boot2.UnnecessarySpringExtension
[WARNING] Changes have been made to src/test/java/org/springframework/samples/petclinic/vet/VetTests.java by:
[WARNING]      org.openrewrite.java.spring.boot2.SpringBoot2JUnit4to5Migration
[WARNING]      org.openrewrite.java.testing.junit5.JUnit4to5Migration
[WARNING]          org.openrewrite.java.testing.junit5.UpdateTestAnnotation
[WARNING] Changes have been made to src/test/java/org/springframework/samples/petclinic/service/ClinicServiceTests.java by:
[WARNING]      org.openrewrite.java.spring.boot2.SpringBoot2JUnit4to5Migration
[WARNING]      org.openrewrite.java.testing.junit5.JUnit4to5Migration
[WARNING]          org.openrewrite.java.testing.junit5.UpdateTestAnnotation
[WARNING]      org.openrewrite.java.spring.boot2.UnnecessarySpringRunWith
[WARNING]          org.openrewrite.java.testing.junit5.RunnerToExtension: {runners=
[org.springframework.test.context.junit4.SpringRunner, org.springframework.test.context.junit4.SpringJUnit4ClassRunner],
extension=org.springframework.test.context.junit.jupiter.SpringExtension}
[WARNING]      org.openrewrite.java.spring.boot2.UnnecessarySpringExtension
[WARNING] Please review and commit the results.
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  36.321 s
[INFO] Finished at: 2022-12-13T12:17:55+01:00
[INFO] -----
```



```
$ git diff
```

```
diff --git a/pom.xml b/pom.xml
index cd0f967..c8db6dc 100644
--- a/pom.xml
+++ b/pom.xml
@@ -62,6 +62,16 @@
     <groupId>org.springframework.boot</groupId>
     <artifactId>spring-boot-starter-test</artifactId>
     <scope>test</scope>
+    <exclusions>
+        <exclusion>
+            <groupId>junit</groupId>
+            <artifactId>junit</artifactId>
+        </exclusion>
+        <exclusion>
+            <groupId>org.junit.vintage</groupId>
+            <artifactId>junit-vintage-engine</artifactId>
+        </exclusion>
+    </exclusions>
</dependency>
```

```
diff --git a/src/test/java/org/springframework/samples/petclinic/PetclinicIntegrationTests.java b/src/test/java/org/springframework/samples/petclinic/PetclinicIntegrationTests.java
index 669dbdd..1db15a8 100644
--- a/src/test/java/org/springframework/samples/petclinic/PetclinicIntegrationTests.java
+++ b/src/test/java/org/springframework/samples/petclinic/PetclinicIntegrationTests.java
@@ -16,15 +16,12 @@
package org.springframework.samples.petclinic;

-import org.junit.Test;
-import org.junit.runner.RunWith;
+import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.samples.petclinic.vet.VetRepository;
-import org.springframework.test.context.junit4.SpringRunner;

-@RunWith(SpringRunner.class)
@SpringBootTest
public class PetclinicIntegrationTests {

@@ -32,7 +29,7 @@ public class PetclinicIntegrationTests {
    private VetRepository vets;

    @Test
-   public void testfindAll() throws Exception {
+   void testfindAll() throws Exception {
        vets.findAll();
        vets.findAll(); // served from cache
    }
}
```

Problem: Zwei Objekte sind gleich, obwohl sie nicht gleich sein sollten

```
1 public final class Person {  
2  
3     private String firstName;  
4     private String lastName;  
5     private String jobTitle;  
6  
7     // more code like getter and setter  
8  
9     @Override  
10    public boolean equals(Object o) {  
11        if (this == o) return true;  
12        if (o == null || getClass() != o.getClass()) return false;  
13        Person person = (Person) o;  
14        return Objects.equals(firstName, person.firstName) &&  
15                Objects.equals(lastName, person.lastName) &&  
16                Objects.equals(jobTitle, person.jobTitle);  
17    }  
18  
19    @Override  
20    public int hashCode() {  
21        return Objects.hash(firstName, lastName, jobTitle);  
22    }  
23 }
```

# equals() Contract

- It is *reflexive*: for any non-null reference value  $x$ ,  $x.equals(x)$  should return true.
- It is *symmetric*: for any non-null reference values  $x$  and  $y$ ,  $x.equals(y)$  should return true if and only if  $y.equals(x)$  returns true.
- It is *transitive*: for any non-null reference values  $x$ ,  $y$ , and  $z$ , if  $x.equals(y)$  returns true and  $y.equals(z)$  returns true, then  $x.equals(z)$  should return true.
- It is *consistent*: for any non-null reference values  $x$  and  $y$ , multiple invocations of  $x.equals(y)$  consistently return true or consistently return false, provided no information used in equals comparisons on the objects is modified.
- For any non-null reference value  $x$ ,  $x.equals(null)$  should return false.

Zur Erinnerung

# *hashCode()* Contract

- *hashCode()* must return same integer consistently for the same object.
- *hashCode()* for two `a.equals(b)` object must be same, ie `a.hashCode()` is same as `b.hashCode()`.
- *hashCode()* for two `not equals()` object may not be distinct.

Zur Erinnerung

# EqualsVerifier

```
1  @Test
2  void equalsContractVerySimple(){
3      Person person1 = createPerson();
4      Person person2 = createPerson();
5      person2.setJobTitle("Consultant");
6
7      assertThat(person1).isNotEqualTo(person2);
8      assertThat(person1.hashCode()).isNotEqualTo(person2.hashCode());
9  }
```

```
12  @Test
13  void equalsContractMuchBetter(){
14      EqualsVerifier.forClass(Person.class)
15          .suppress(Warning.NONFINAL_FIELDS).verify();
16  }
```



```
java.lang.AssertionError: EqualsVerifier found a problem in class Person.  
-> Mutability: equals depends on mutable field firstName.
```

For more information, go to: <http://www.jqno.nl>equalsverifier/errormessages>

```
at nl.jqno.equalsverifier.EqualsVerifierApi.verify(EqualsVerifierApi.java:305)  
at com.github.sparsick.test.tool.database.PersonTest.equalsContractMuchBetter(PersonTest.java:26)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)  
at  
java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  
at java.base/java.lang.reflect.Method.invoke(Method.java:566)  
at org.junit.platform.commons.util.ReflectionUtils.invokeMethod(ReflectionUtils.java:675)  
at org.junit.jupiter.engine.execution.MethodInvocation.proceed(MethodInvocation.java:60)  
at  
org.junit.jupiter.engine.execution.InvocationInterceptorChain$ValidatingInvocation.proceed(InvocationInterceptorChain.java:125)  
at org.junit.jupiter.engine.extension.TimeoutExtension.intercept(TimeoutExtension.java:132)  
at org.junit.jupiter.engine.extension.TimeoutExtension.interceptTestableMethod(TimeoutExtension.java:124)  
at org.junit.jupiter.engine.extension.TimeoutExtension.interceptTestMethod(TimeoutExtension.java:74)  
at  
org.junit.jupiter.engine.execution.ExecutableInvoker$ReflectiveInterceptorCall.lambda$ofVoidMethod$0(ExecutableInvoker.java:115)  
at org.junit.jupiter.engine.execution.ExecutableInvoker.lambda$invoke$0(ExecutableInvoker.java:105)  
at  
org.junit.jupiter.engine.execution.InvocationInterceptorChain$InterceptedInvocation.proceed(InvocationInterceptorChain.java:104)  
.
```

Was ist mit *toString()*?

Problem: *toString()* beinhaltet nicht alle Klassenfelder

```
1 public final class Person {  
2  
3     private String firstName;  
4     private String lastName;  
5     private String jobTitle;  
6  
7     // more code like getter, setter, hashCode and equals  
8  
9  
10    @Override  
11    public String toString() {  
12        return "Person{" +  
13                "firstName='" + firstName + '\'' +  
14                ", lastName='" + lastName + '\'' +  
15                ", jobTitle='" + jobTitle + '\'' +  
16                '}';  
17    }  
18}
```

# To String Verifier

```
1  @Test
2  void toStringVerifier(){
3      ToStringVerifier.forClass(Person.class)
4          .withClassName(Style.SIMPLE_NAME)
5          .verify();
6  }
```



```
java.lang.AssertionError:  
  
Failed verification:  
com.github.sparsick.test.tool.database.Person  
  
Expected auto generated toString:  
Person{firstName='4b2b5d96-7d9f-46a3-a7ce-dd416c46706c', lastName='bb28f281-5cf4-43be-90ba-9dffaac274e6',  
jobTitle='a000c70c-1107-4a30-b9b9-5176222a4da6'}  
  
To start with class name: com.github.sparsick.test.tool.database.Person  
  
at com.jparams.verifier.tostring.ToStringVerifier.verify(ToStringVerifier.java:362)  
at com.github.sparsick.test.tool.database.PersonTest.toStringVerifier(PersonTest.java:33)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)  
at  
java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  
at java.base/java.lang.reflect.Method.invoke(Method.java:566)  
at org.junit.platform.commons.util.ReflectionUtils.invokeMethod(ReflectionUtils.java:675)  
at org.junit.jupiter.engine.execution.MethodInvocation.proceed(MethodInvocation.java:60)  
at  
org.junit.jupiter.engine.execution.InvocationInterceptorChain$ValidatingInvocation.proceed(InvocationInterceptorChain.java:125)  
at org.junit.jupiter.engine.extension.TimeoutExtension.intercept(TimeoutExtension.java:132)  
at org.junit.jupiter.engine.extension.TimeoutExtension.interceptTestableMethod(TimeoutExtension.java:124)  
at org.junit.jupiter.engine.extension.TimeoutExtension.interceptTestMethod(TimeoutExtension.java:74)  
at  
org.junit.jupiter.engine.execution.ExecutableInvoker$ReflectiveInterceptorCall.lambda$ofVoidMethod$0(Executable
```

# To String Verifier

- NameStyle
- Preset
  - Eclipse
  - IntelliJ
  - Guava
  - ApacheToStringBuilder
- Weitere Verifier (Auswahl)
  - withHashCode
  - withIgnoreFields
  - withNullField
  - withOnlyTheseFields

Problem: Wie soll ich Nebenläufigkeit testen?

# Awaitility

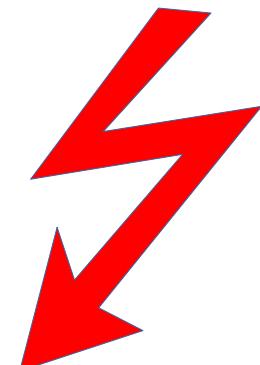


- Java DSL, um asynchrone Operationen zu synchronisieren
- Support:
  - Groovy DSL
  - Kotlin DSL
  - Scala DSL

**Wichtig !**

Awaitility macht nichts, um Thread-Sicherheit oder Thread-Synchronisierung zu gewährleisten.

It's your job!



# Beispiel

```
@Test
void awaitilityDemo() {
    var demoUnderTest = new ConcurrencyDemo();
    demoUnderTest.addItem("Hello World");

    await().until(demoUnderTest::hasNewItem);
    assertThat(demoUnderTest.allItems()).contains("Hello World");
}
```

# Weitere Möglichkeiten (Auswahl)

Abfragen von Feldern:

```
await().until( fieldIn(object).ofType(int.class), equalTo(2) );
```

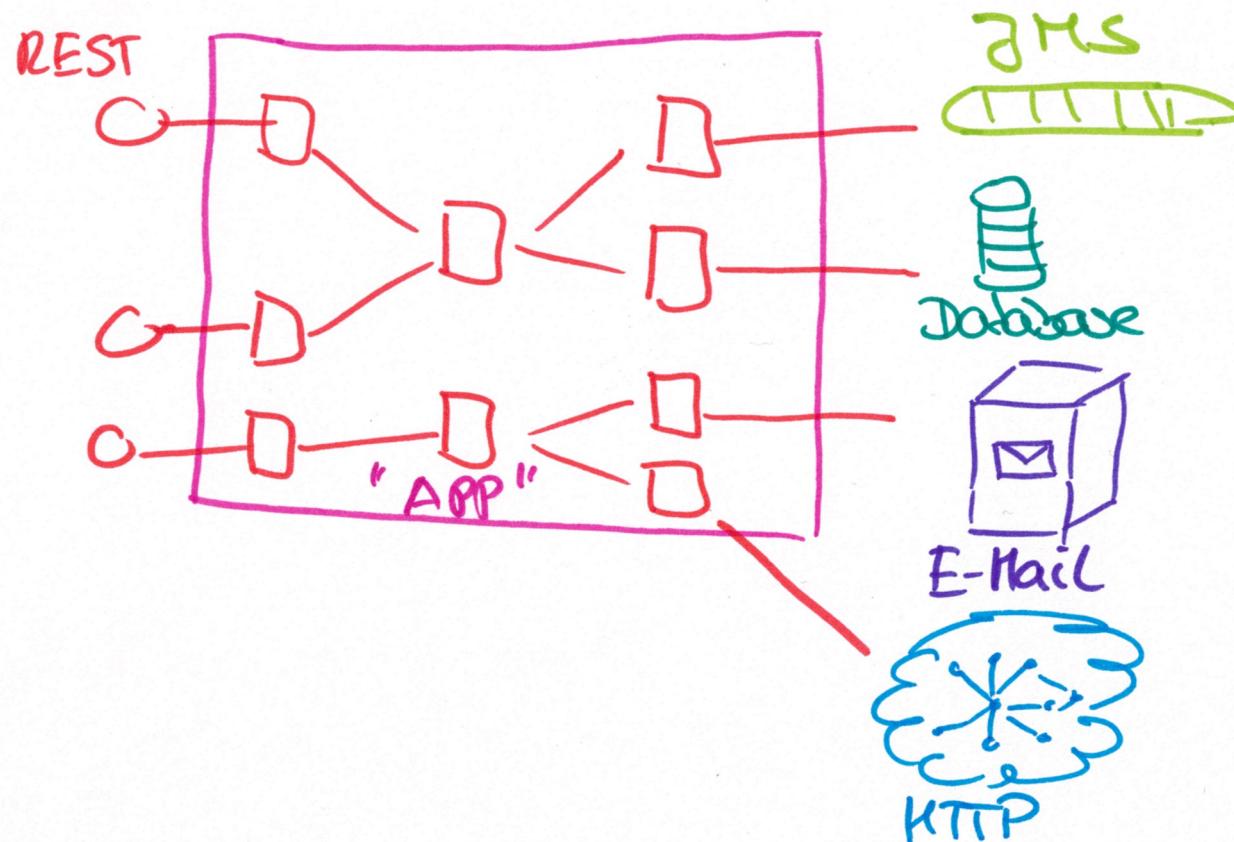
Ignorieren von Exceptions:

```
given().ignoreExceptions().await().until(() -> SocketUtils.findAvailableTcpPort(x,y));
```

Abwarten einer Mindestzeit:

```
await().atLeast(1, SECONDS).and().atMost(2, SECONDS).until(value(), equalTo(1));
```

# Problem: Integrierte Tests



# Problembeschreibung

- Interaktion mit Infrastruktur wird recht spät getestet
  - Feedback bei Fehlern recht spät
- Abhängig von einer bestimmten Infrastruktur
  - Manchmal auch schon beim Build (Bad smell)
  - False negative Fehlerquote recht hoch
  - Aufwändiges Setup
  - Testausführung langsam

# Integration vs Integrated Tests

## Integrated Tests

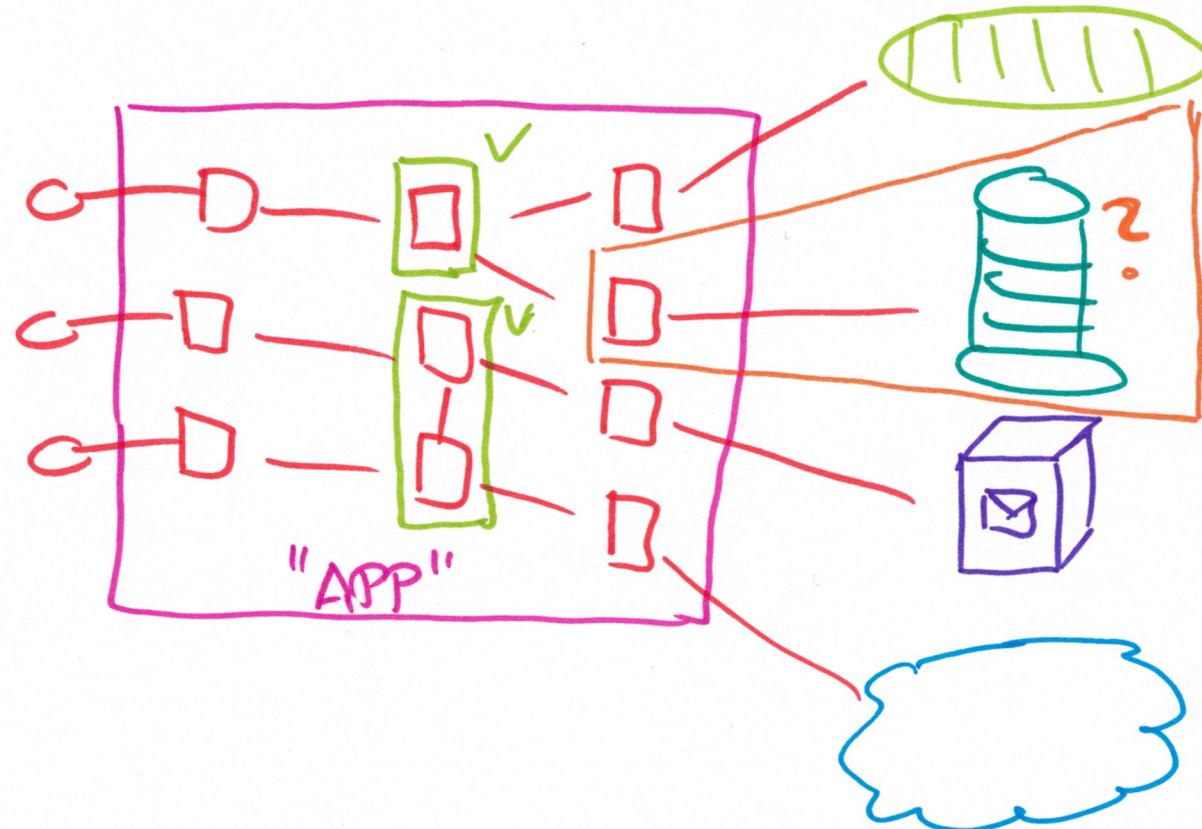
A test that will pass or fail based on the correctness of another system.

*J.B.Rainsberger*

Signs for having Integrated Tests:

- We spin up other services in a local testing environment
- We test against other services in a shared testing environment
- Changes to your system breaks tests for other systems

*from Spotify Blog Post*



# Anforderungen

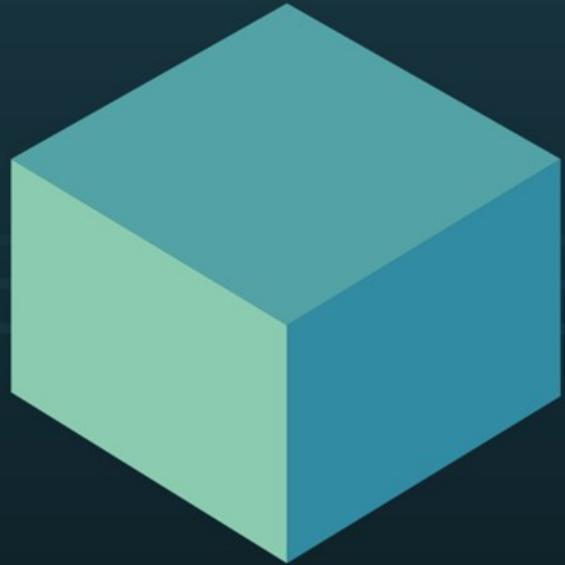
- Build muss unabhängig von anderen Systemen sein
  - Minimierung der false negative Fehler
  - Entkopplung zwischen den Entwicklern
- Feedback bei Fehlern so schnell wie möglich
- Testausführung beschleunigen



Datenbank

# Datenbanken

- Embedded Datenbanken
  - H2, Derby
  - Nicht Produktionsnah
  - Nicht alles testbar
- Standalone Datenbanken
  - Abhangigkeit zur bestimmten Infrastruktur
- Shared Datenbanken
  - Abhangigkeiten zwischen Entwickler
  - Hohe false negative Fehlerrate



TESTCONTAINERS

```
@Testcontainers
class PersonRepositoryJUnit5Test {

    @Container
    private PostgreSQLContainer postgres = new PostgreSQLContainer();

    private PersonRepository repositoryUnderTest;

    @BeforeEach
    void setup(){
        HikariConfig hikariConfig = new HikariConfig();
        hikariConfig.setJdbcUrl(postgres.getJdbcUrl());
        hikariConfig.setUsername(postgres.getUsername());
        hikariConfig.setPassword(postgres.getPassword());

        HikariDataSource ds = new HikariDataSource(hikariConfig);
        Flyway flyway = Flyway.configure().dataSource(ds).load();
        flyway.migrate();

        repositoryUnderTest = new PersonRepository(ds);
    }

    @Test
    void saveAndFindAllPerson() {
        Person person = new Person();
        person.setFirstName("firstName");
        person.setLastName("lastName");
        person.setJobTitle("jobTitle");

        repositoryUnderTest.save(person);

        List<Person> persons = repositoryUnderTest.findAllPersons();
        assertThat(persons).hasSize(1).contains(person);
    }
}
```

```
@Testcontainers
class DbMigrationJUnit5Test {

    @Container
    private MySQLContainer mysqlDb = new MySQLContainer();

    @Test
    void testDbMigrationFromTheScratch(){
        Flyway flyway = Flyway.configure()
            .dataSource(mysqlDb.getJdbcUrl(),
                       mysqlDb.getUsername(),
                       mysqlDb.getPassword()).load();

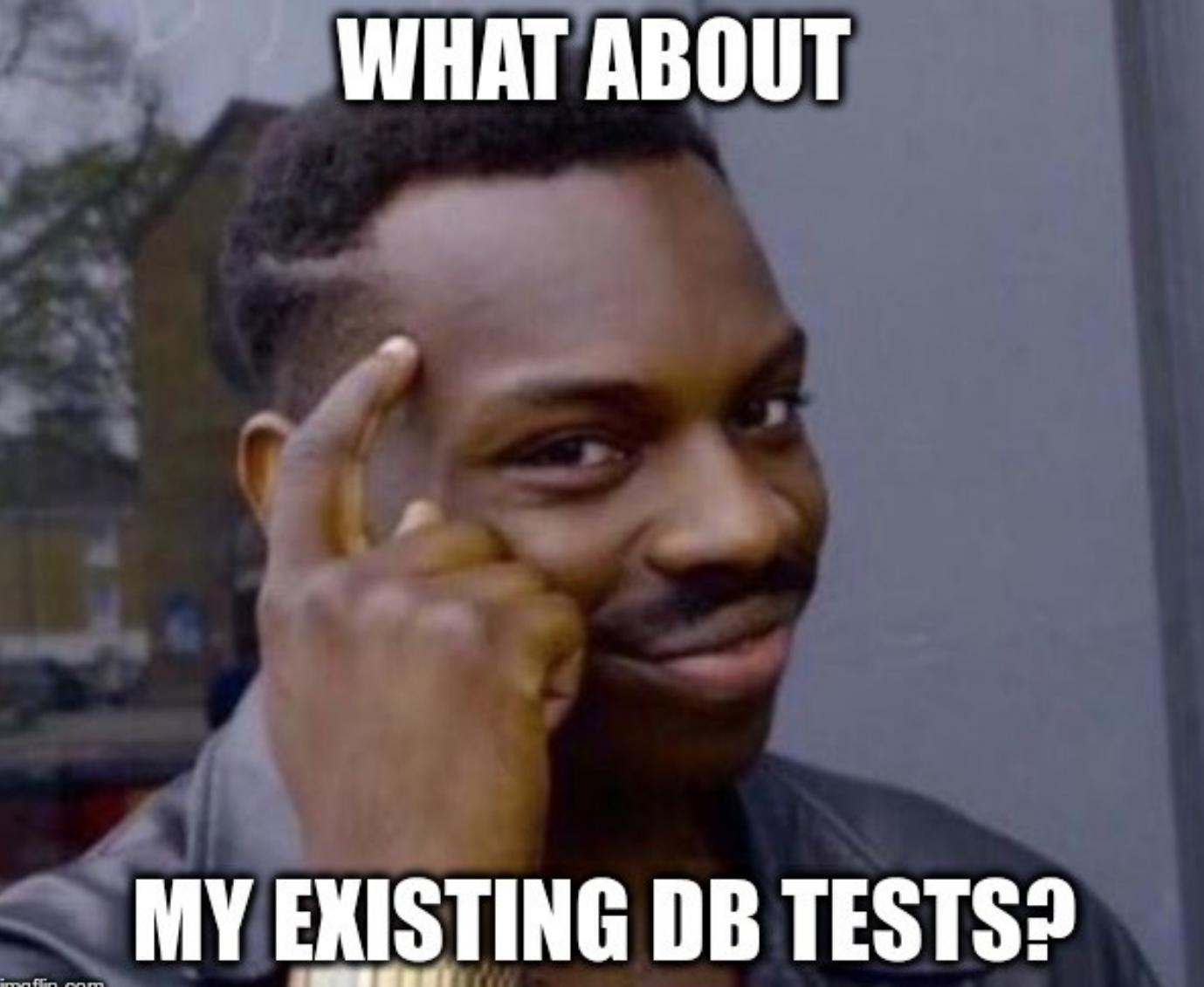
        flyway.migrate();
    }
}
```

## TESTS

```
-----  
Running db.migration.DbMigrationITest  
INFO - RegistryClientProviderStrategy - Found docker client settings from environment  
INFO - DockerClientProviderStrategy - Found Docker environment with Environment variables, system properties and defaults. Resolved:  
  dockerHost=unix:///var/run/docker.sock  
  apiVersion='{UNKNOWN_VERSION}'  
  registryUrl='https://index.docker.io/v1/'  
  registryUsername='sparsick'  
  registryPassword='null'  
  registryEmail='null'  
  dockerConfig='DefaultDockerClientConfig[dockerHost=unix:///var/run/docker.sock, registryUsername=sparsick, registryPassword=<null>, registryEmail=<null>]'  
INFO - DockerClientFactory      - Docker host IP address is localhost  
INFO - DockerClientFactory      - Connected to docker:  
  Server Version: 17.05.0-ce  
  API Version: 1.29  
  Operating System: Linux Mint 18.2  
  Total Memory: 19511 MB  
    i Checking the system...  
    ✓ Docker version is newer than 1.6.0  
    ✓ Docker environment has more than 2GB free  
    ✓ File should be mountable  
    ✓ Exposed port is accessible  
INFO - [mysql:latest]           - Creating container for image: mysql:latest  
INFO - [mysql:latest]           - Starting container with ID: 2668be66c2631e49b5bcb4e180665d223525ec896ea78034326076d5f9063d53  
INFO - [mysql:latest]           - Container mysql:latest is starting: 2668be66c2631e49b5bcb4e180665d223525ec896ea78034326076d5f9063d53  
INFO - [mysql:latest]           - Waiting for database connection to become available at jdbc:mysql://localhost:32769/test using query 'SELECT  
INFO - [mysql:latest]           - Obtained a connection to container (jdbc:mysql://localhost:32769/test)  
INFO - [mysql:latest]           - Container mysql:latest started  
INFO - VersionPrinter          - Flyway 4.0.3 by Boxfuse  
INFO - DbSupportFactory         - Database: jdbc:mysql://localhost:32769/test (MySQL 5.7)  
INFO - DbValidate               - Successfully validated 2 migrations (execution time 00:00.011s)  
INFO - MetaDataTableImpl        - Creating Metadata table: `test`.`schema_version`  
INFO - DbMigrate                - Current version of schema `test`: <> Empty Schema <>  
INFO - DbMigrate                - Migrating schema `test` to version 1.0.0 - create person table  
INFO - DbMigrate                - Migrating schema `test` to version 2.0.0 - add column job title  
INFO - DbMigrate                - Successfully applied 2 migrations to schema `test` (execution time 00:00.133s).  
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 13.9 sec
```

# Testcontainers

- Temporary database containers - spezielle MySQL, PostgreSQL, Oracle XE und Virtuoso container
- Webdriver containers - Dockerized Chrome oder Firefox browser für Selenium/Webdriver Operationen mit automatischer Videoaufnahme
- Weitere spezifische Container – Elasticsearch, Kafka, Apache Pulsar, Mockserver, Toxiproxy, Nginx, Hashicorp Vault
- Generic containers – irgendein Docker Container
- Docker compose – Wiederverwendung von Docker Compose YAML Datei
- Dockerfile containers – Container direkt von einem Dockerfile



**WHAT ABOUT**

**MY EXISTING DB TESTS?**

# Container Via JDBC URL

```
public class PersonRepositoryJdbcUrlTestContainerTest {  
    private PersonRepository repositoryUnderTest;  
    private Flyway flyway;  
  
    @BeforeEach  
    void setup(){  
        HikariConfig hikariConfig = new HikariConfig();  
        hikariConfig.setJdbcUrl("jdbc:tc:postgresql:9.6.8:///persondb");  
        hikariConfig.setUsername("postgres");  
        hikariConfig.setPassword("");  
  
        DataSource ds = new HikariDataSource(hikariConfig);  
        flyway = Flyway.configure().dataSource(ds).load();  
        flyway.migrate();  
  
        repositoryUnderTest = new PersonRepository(ds);  
    }  
  
    @AfterEach  
    void cleanUp(){  
        flyway.clean();  
    }  
}
```

# Docker Maven Plugin (DMP)

- fabric8io/docker-maven-plugin (<http://dmp.fabric8.io> )
- Kann Docker Image bauen
- Aber auch Container starten und stoppen

# Maven Phase: Integration-Test

- Ausgangspunkt:
  - Es gibt Integrationstests gegen embedded oder standalone Datenbank
- Improvement:
  - DMP startet Container vor den Tests (*pre-integration-test*)
  - DMP stoppt Contianer nach den Tests (*post-integration-test*)



# Fazit

Integration Tests  
Mockserver, Wiremock,  
Testcontainers

equals, hashCode, toString  
EqualsVerifier, ToStringVerifier

Gut lesbare Assertions  
AssertJ, AssertJ-DB, JUnit5  
Group Assertion

Concurrency  
Awaitility

Testdaten  
generatedata.com  
JavaFaker  
ObjectMother, TestDataBuilder

Parametrisierte  
Tests  
JUnit5, Spock

Tests, die nur unter bestimmten  
Bedingungen laufen  
JUnit5 ConditionsTest

Fragen?

[mail@sandra-parsick.de](mailto:mail@sandra-parsick.de)

[@sparsick@mastodon.social](https://@sparsick@mastodon.social)

<https://github.com/sparsick/test-tool-talk>

# Literatur

- <https://www.martinfowler.com/bliki/ObjectMother.html>
- <http://natpryce.com/articles/000714.html>
- <http://coding-is-like-cooking.info/2018/04/pre-tested-integration-back-to-the-basis-of-ci/>
- <http://blog.thecodewhisperer.com/permalink/integrated-tests-are-a-scam>
- <http://blog.thecodewhisperer.com/permalink/clearing-up-the-integrated-tests-scam>
- [https://bee42.com/de/blog/The\\_dark\\_age\\_of\\_container\\_testing/](https://bee42.com/de/blog/The_dark_age_of_container_testing/)

# Literatur

- <https://labs.spotify.com/2018/01/11/testing-of-microservices/>
- <https://martinfowler.com/bliki/IntegrationTest.html>
- <https://codewithoutrules.com/2016/07/31/verified-fakes/>