JavaLand, 16.03.2021

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

Sandra Parsick
@SandraParsick
mail@sandra-parsick.de

Zu meiner Person

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

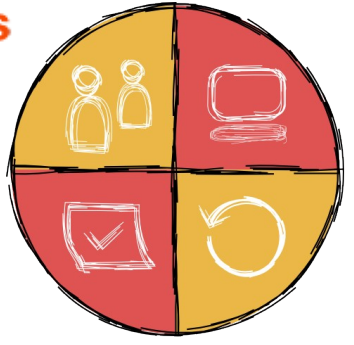
mail@sandra-parsick.de

@SandraParsick

xing.to/sparsick

https://www.sandra-parsick.de

Oracle Groundbreakers




Das kann man lokal nicht testen.
Das ist zu aufwendig zu testen.
Das kann man nicht testen.


CHALLENGE ACCEPTED
imgflip.com
quickmeme.com

Agenda

Testdaten

Integrierte Tests

Testcode-
Duplizierung

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

```java
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  }
```

Parametrizierte Tests mit JUnit 4

```java
@RunWith(Parameterized.class)
public class PositionMoveBackwardJUnit4Test {

    private final Direction direction;
    private final Position expectedPosition;

    public PositionMoveBackwardJUnit4Test(Direction direction, Position expectedPosition) {
        this.direction = direction;
        this.expectedPosition = expectedPosition;
    }

    @Parameterized.Parameters
    public static Collection moveBackwardParameter() {
        return Arrays.asList(new Object[][]{
                {Direction.NORTH, new Position(10, 9)},
                {Direction.EAST, new Position(9, 10)},
                {Direction.SOUTH, new Position(10, 11)},
                {Direction.WEST, new Position(11, 10)}
        });
    }

    @Test
    public void moveBackward() {
        Position positionUnderTest = new Position(10, 10);
        Position newPosition = positionUnderTest.moveBackward(direction);
        assertThat(newPosition).isEqualTo(expectedPosition);
    }
}
```

```java
@RunWith(Parameterized.class)
public class PositionMoveForwardJUnit4Test {

    private final Direction direction;
    private final Position expectedPosition;

    public PositionMoveForwardJUnit4Test(Direction direction, Position expectedPosition) {
        this.direction = direction;
        this.expectedPosition = expectedPosition;
    }

    @Parameterized.Parameters
    public static Collection moveBackwardParameter() {
        return Arrays.asList(new Object[][]{
                {Direction.NORTH, new Position(10, 11)},
                {Direction.EAST, new Position(11, 10)},
                {Direction.SOUTH, new Position(10, 9)},
                {Direction.WEST, new Position(9, 10)}
        });
    }

    @Test
    public void moveForward() {
        Position positionUnderTest = new Position(10, 10);

        Position newPosition = positionUnderTest.moveForward(direction);

        assertThat(newPosition).isEqualTo(expectedPosition);
    }
}
```

Parametrisierte Test mit JUnit5

```java
class PositionJUnit5Test {

    @ParameterizedTest
    @MethodSource("createMoveBackwardParameter")
    void moveBackward(Direction direction, Position expectedPosition) {
        Position positionUnderTest = new Position(10, 10);
        Position newPosition = positionUnderTest.moveBackward(direction);
        assertThat(newPosition).isEqualTo(expectedPosition);
    }

    private static Stream<Arguments> createMoveBackwardParameter() {
        return Stream.of(
                Arguments.of(Direction.NORTH, new Position(10, 9)),
                Arguments.of(Direction.EAST, new Position(9, 10)),
                Arguments.of(Direction.SOUTH, new Position(10, 11)),
                Arguments.of(Direction.WEST, new Position(11, 10))
        );
    }

    @ParameterizedTest
    @MethodSource("createMoveForwardParameter")
    void moveForward(Direction direction, Position expectedPosition) {
        Position positionUnderTest = new Position(10, 10);
        Position newPosition = positionUnderTest.moveForward(direction);
        assertThat(newPosition).isEqualTo(expectedPosition);
    }

    private static Stream<Arguments> createMoveForwardParameter() {
        return Stream.of(
                Arguments.of(Direction.NORTH, new Position(10, 11)),
                Arguments.of(Direction.EAST, new Position(11, 10)),
                Arguments.of(Direction.SOUTH, new Position(10, 9)),
                Arguments.of(Direction.WEST, new Position(9, 10))
        );
    }
}
```

Weitere Möglichkeiten in JUnit5

```java
@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();
}
```

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

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

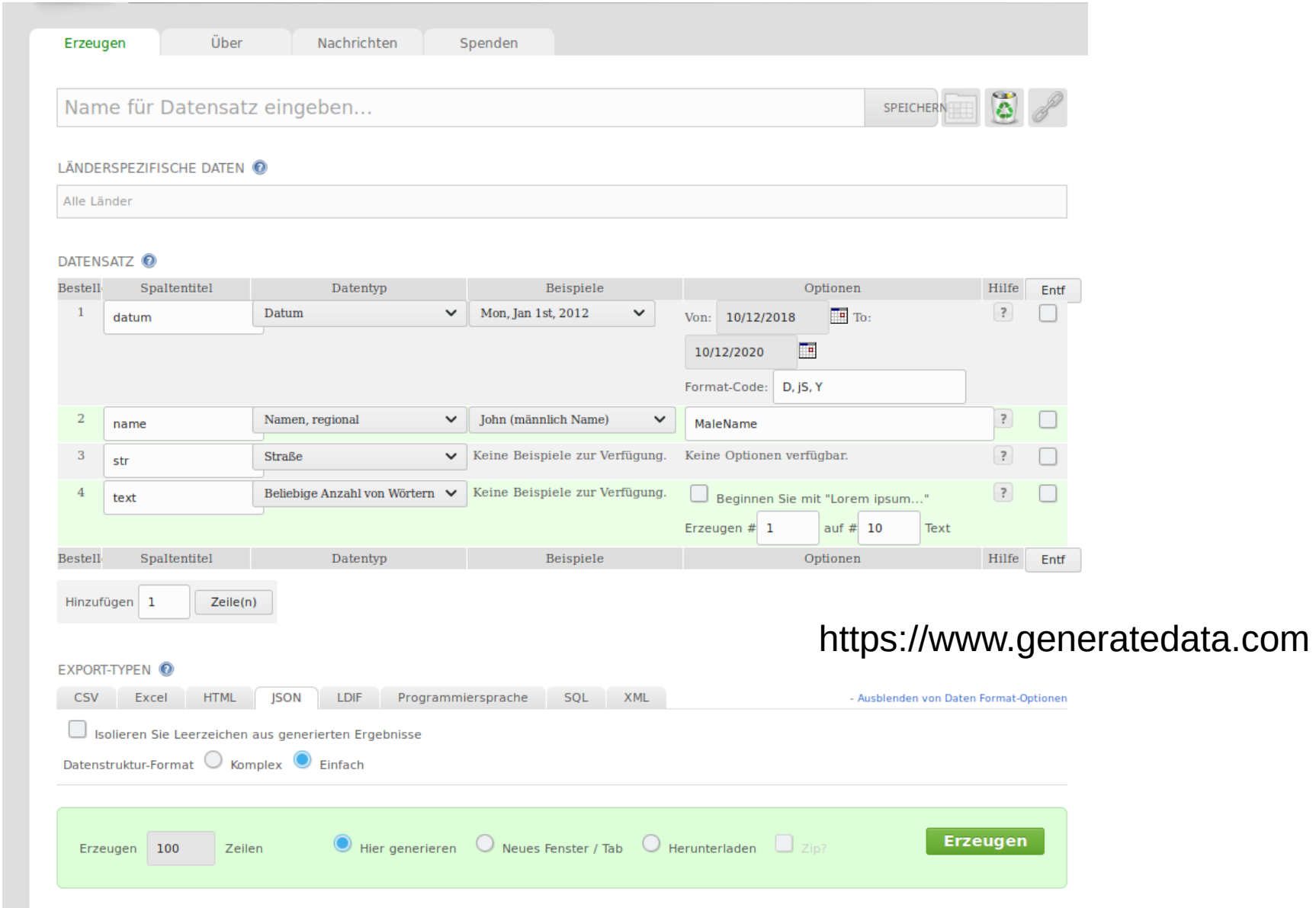
```java
public class MyArgumentsProvider implements ArgumentsProvider {
    @Override
    public Stream<? extends Arguments> provideArguments(
                        ExtensionContext extensionContext) {
        return Stream.of("foo", "bar").map(Arguments::of);
    }
}
```

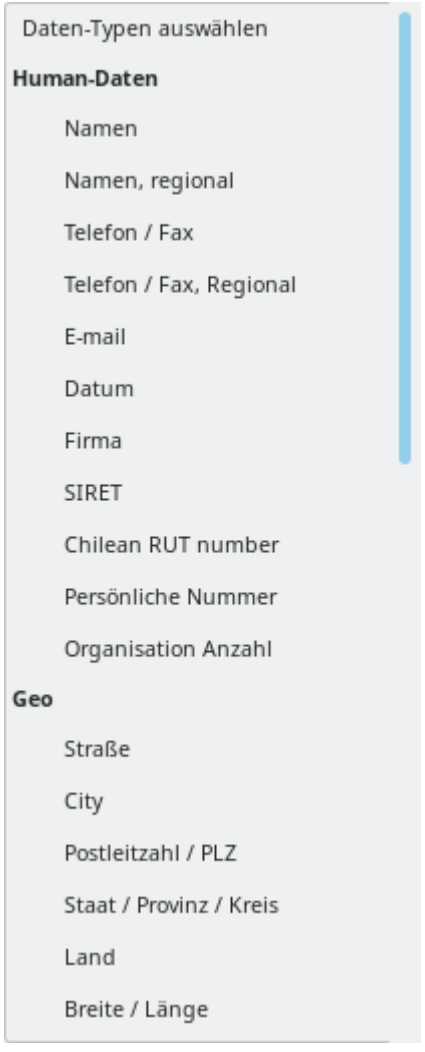
Parametrisierte Test mit Spock

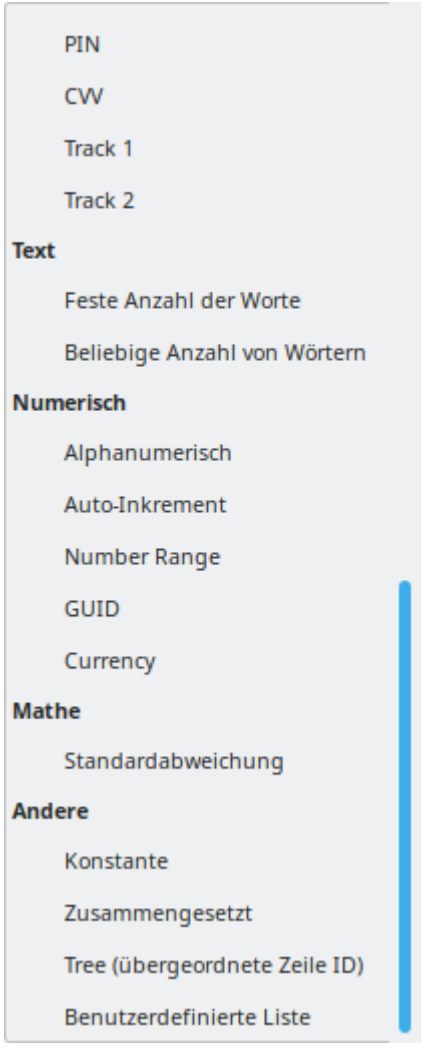
```groovy
class PositionSpockTest extends Specification {

    def "move backward" (Direction direction, Position expectedPosition){
        Position positionUnderTest = new Position(10, 10)
        Position newPosition = positionUnderTest.moveBackward(direction)

        expect:
        assert newPosition == expectedPosition

        where:
        direction | expectedPosition
        Direction.NORTH | new Position(10, 9)
        Direction.EAST | new Position(9, 10)
        Direction.SOUTH | new Position(10, 11)
        Direction.WEST |new Position(11, 10)
    }

    def "move forward" (Direction direction, Position expectedPosition){
        Position positionUnderTest = new Position(10, 10)
        Position newPosition = positionUnderTest.moveForward(direction)

        expect:
        assert newPosition == expectedPosition

        where:
        direction | expectedPosition
        Direction.NORTH | new Position(10, 11)
        Direction.EAST | new Position(11, 10)
        Direction.SOUTH | new Position(10, 9)
        Direction.WEST |new Position(9, 10)
    }
}
```

Problem: Testdaten

https://www.generatedata.com


Erzeugen
Über
Nachrichten
Spenden
Name für Datensatz eingeben...
SPEICHERN
LÄNDERSPEZIFISCHE DATEN
Alle Länder
DATENSATZ
Bestell
Spaltentitel
Datentyp
Beispiele
Optionen
Hilfe
Entf
1
datum
Datum
Mon, Jan 1st, 2012
Von: 10/12/2018 To:
10/12/2020
Format-Code: D, jS, Y
2
name
Namen, regional
John (männlich Name)
MaleName
3
str
Straße
Keine Beispiele zur Verfügung.
Keine Optionen verfügbar.
4
text
Beliebige Anzahl von Wörtern
Keine Beispiele zur Verfügung.
Beginnen Sie mit "Lorem ipsum..."
Erzeugen # 1 auf # 10 Text
Hinzufügen 1 Zeile(n)
https://www.generatedata.com
EXPORT-TYPEN
CSV
Excel
HTML
JSON
LDIF
Programmiersprache
SQL
XML
- Ausblenden von Daten Format-Optionen
Isolieren Sie Leerzeichen aus generierten Ergebnisse
Datenstruktur-Format Komplex Einfach
Erzeugen 100 Zeilen
Hier generieren
Neues Fenster / Tab
Herunterladen
Zip?
Erzeugen


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
CVV
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

Erzeugen | Über | Nachrichten | Spenden

Erstellt **100** **100** Ergebnisse

abbrechen


```
[
    {
        "datum": "Wed, 1st, 2020",
        "name": "Zachary",
        "str": "Ap #213-6985 Molestie  Straße",
        "text": "nec mauris blandit mattis. Cras eget"
    },
    {
        "datum": "Fri, 10th, 2019",
        "name": "Macaulay",
        "str": "Ap #891-1080 Curabitur  Ave",
        "text": "hendrerit neque. In ornare sagittis felis. Donec tempor, est ac"
    },
    {
        "datum": "Sun, 24th, 2019",
        "name": "Jonas",
        "str": "Ap #942-7252 Quam  Avenue",
        "text": "Nam tempor diam dictum sapien. Aenean massa. Integer vitae nibh."
    },
    {
        "datum": "Thu, 4th, 2019",
        "name": "Adrian",
        "str": "6794 Egestas,  Rd.",
        "text": "turpis. Aliquam adipiscing lobortis risus."
    },
    {
        "datum": "Mon, 3rd, 2018",
        "name": "Lev",
        "str": "Ap #105-239 Arcu  Straße",
        "text": "Sed id risus"
    },
```


« Regenerieren

A A A

Und wenn es doch lieber Code sein soll?

Testdaten mit JavaFaker

```java
@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

```java
@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
    // negotation
}

@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

```java
@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

```java
@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

```java
@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");
}
```

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

```xml
<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>
```

```xml
<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>
```

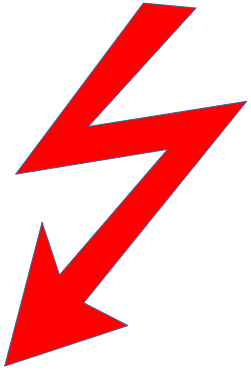
Problem: Wie soll ich Nebenläufigkeit testen?

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) );
```

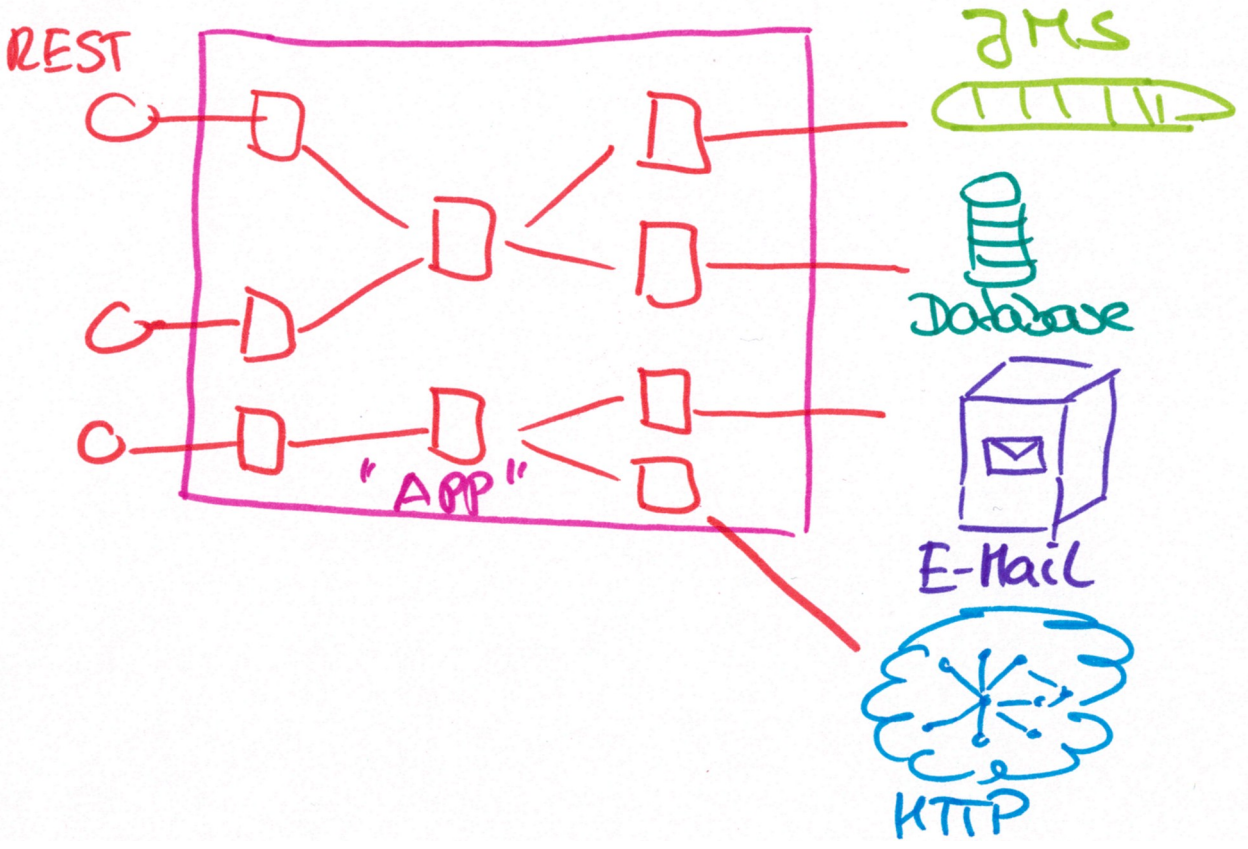
Ignorieren von Exceptions:

```
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


REST
JMS
Database
"APP"
E-Mail
HTTP

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

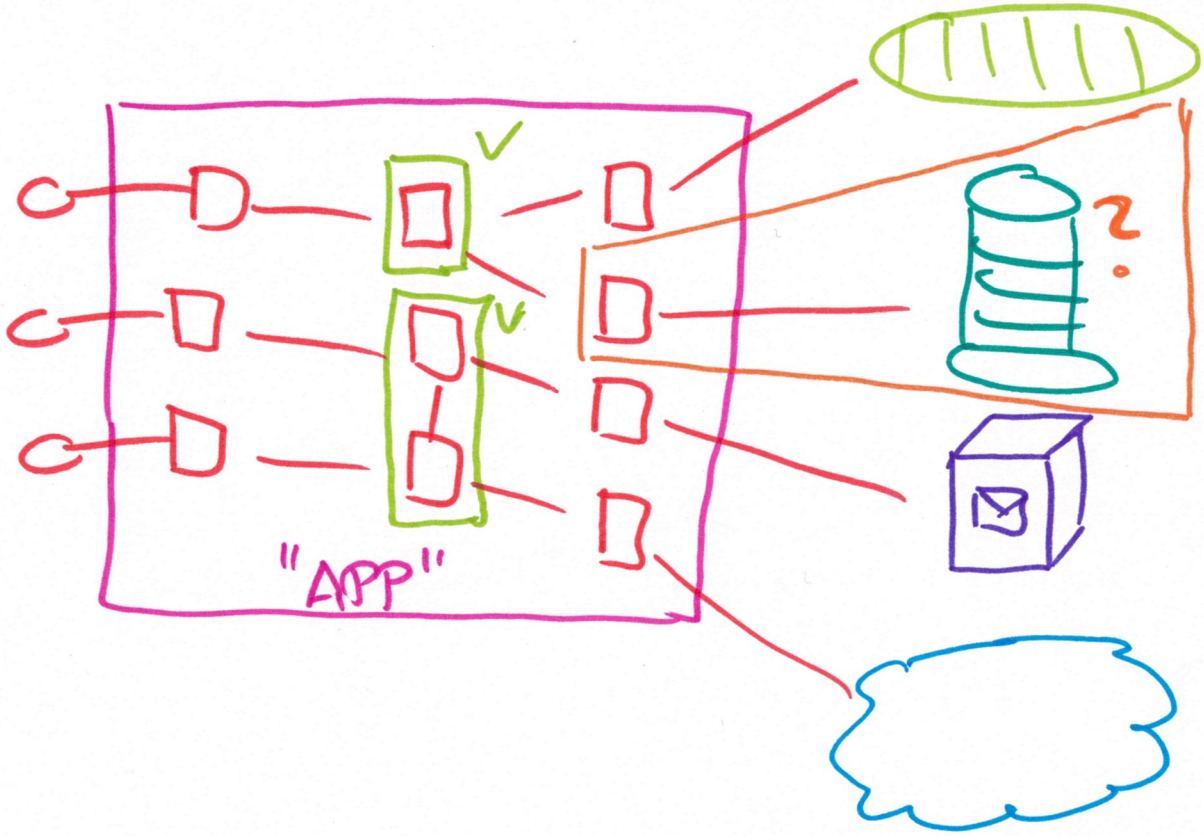
A test that will pass or fail based on the correctness of another system.

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


"APP"

Datenbanken

Datenbanken

- Embedded Datenbanken
  - H2, Derby
  - Nicht Produktionsnah
  - Nicht alles testbar
- Standalone Datenbanken
  - Abhängigkeit zur bestimmten Infrastruktur
- Shared Datenbanken
  - Abhängigkeiten zwischen Entwickler
  - Hohe false negative Fehlerrate


TESTCONTAINERS

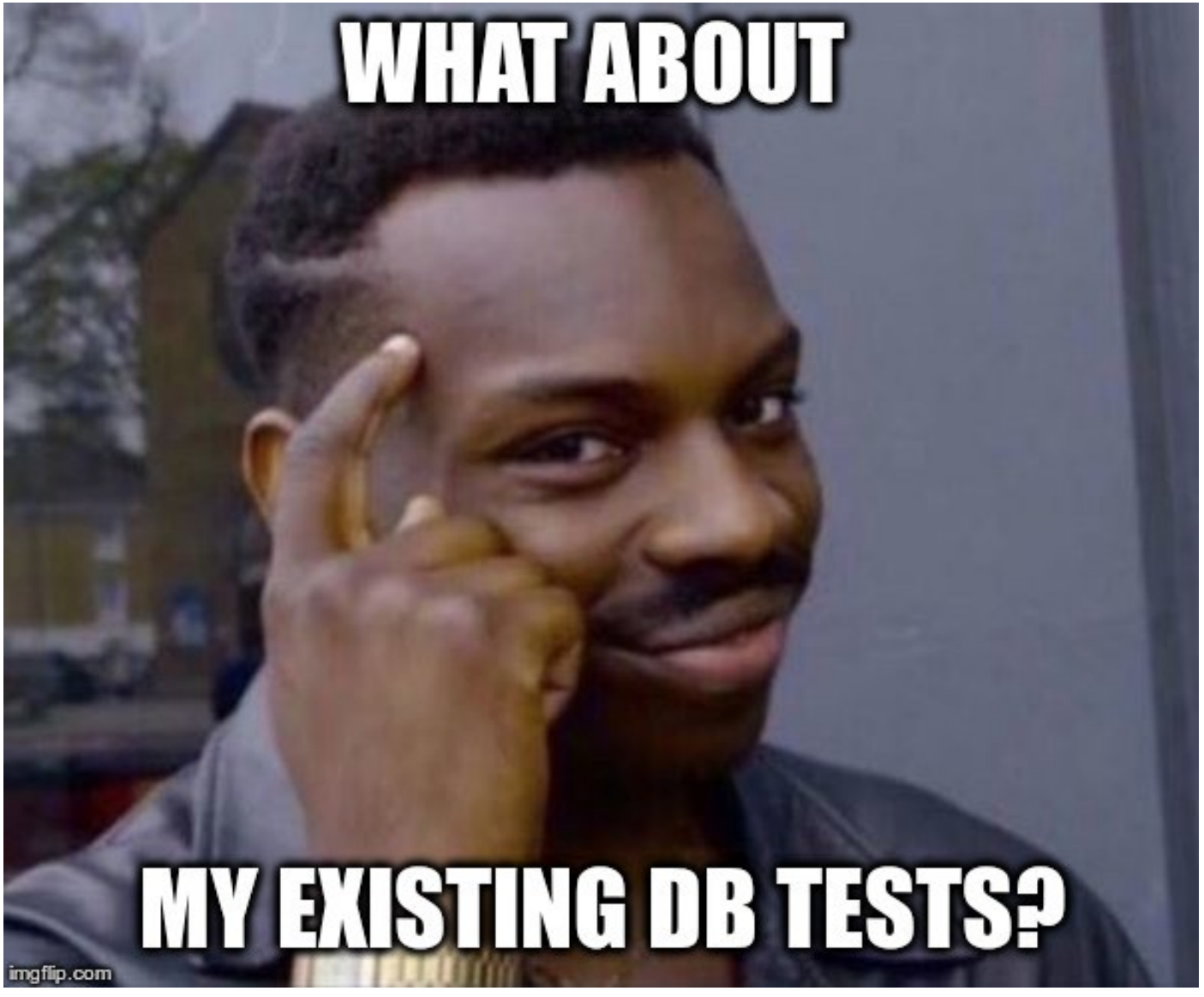
```java
@Testcontainers
class PersonRepositoryJUnit5Test {

    @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);
    }
}
```

```java
@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();
    }
}
```

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running db.migration.DbMigrationITest
INFO  - ertyClientProviderStrategy - Found docker client settings from environment
INFO  - ckerClientProviderStrategy - 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>,registryEma

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?
imgflip.com

Container Via JDBC URL

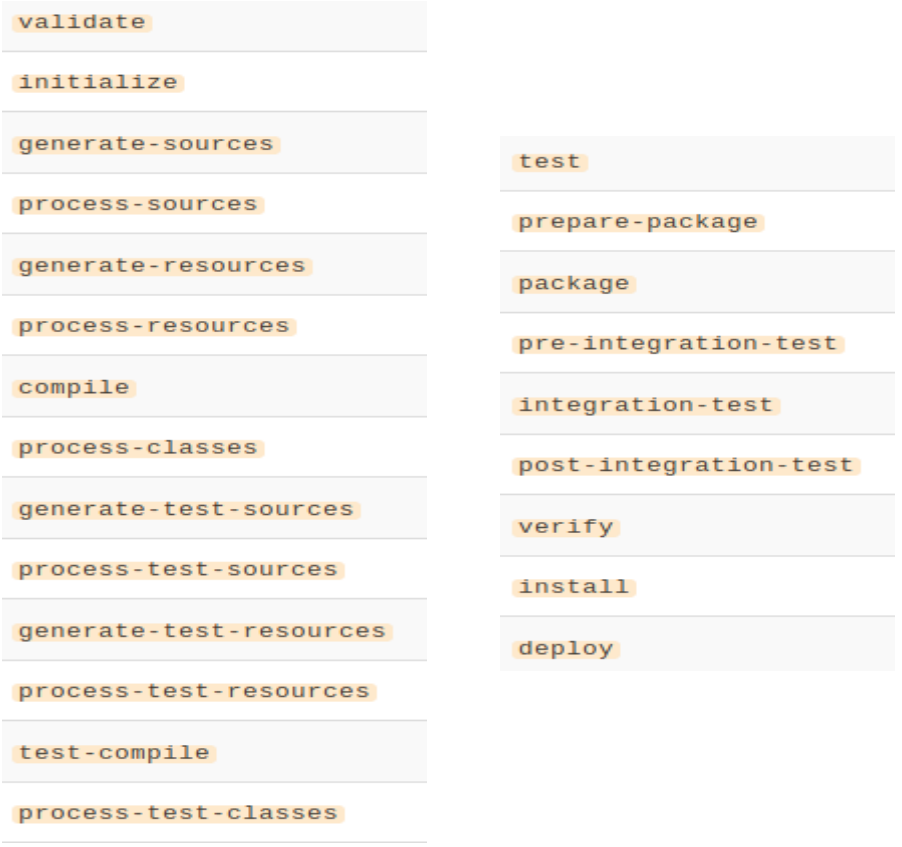
```
public class PersonRepositoryJdbcUrlTestContainerTest {
    private PersonRepository repositoryUnderTest;
    private Flyway flyway;

    @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
Testcontainers

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

Concurrency
Awaitility

Parametrisierte
Tests
JUnit5, Spock

Testdaten
generatedata.com
JavaFaker
ObjectMother, TestDataBuilder

Tests, die nur unter bestimmten
Bedingungen laufen
JUnit5 ConditionsTest

Fragen?

mail@sandra-parsick.de
@SandraParsick

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/

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/