AGILE BACKEND CLOUD EVOLUTION FRONTEND JAVA JAVAFX .NET QA/TEST SAP UI/UX WEB

Saxonia Systems
So geht Software.

JUNIT 5

NEUE OPTIONEN ZUM TESTEN IN JAVA

So geht Software.

sogehtsoftware.de

Was ist JUnit?

WAS IST JUNIT?

Saxonia Systems
So geht Software.

Testframework für Java

Fokus: Unittests

Preisfrage: Warum Unittests?

WARUM UNITTESTS?

Saxonia Systems
So geht Software.

- Funktioniert das, was ich entwickelt habe?
- Gute Codestruktur und testbare Einheiten (vor allem mit TDD)
- Anforderungsdokumentation
- Sicherheitsnetz für Änderungen

→ für mich und **für die anderen**

Saxonia Systems
So geht Software.

JUnit 5

JUNIT 5

Warum gibt es JUnit 5?

Saxonia Systems
So geht Software.

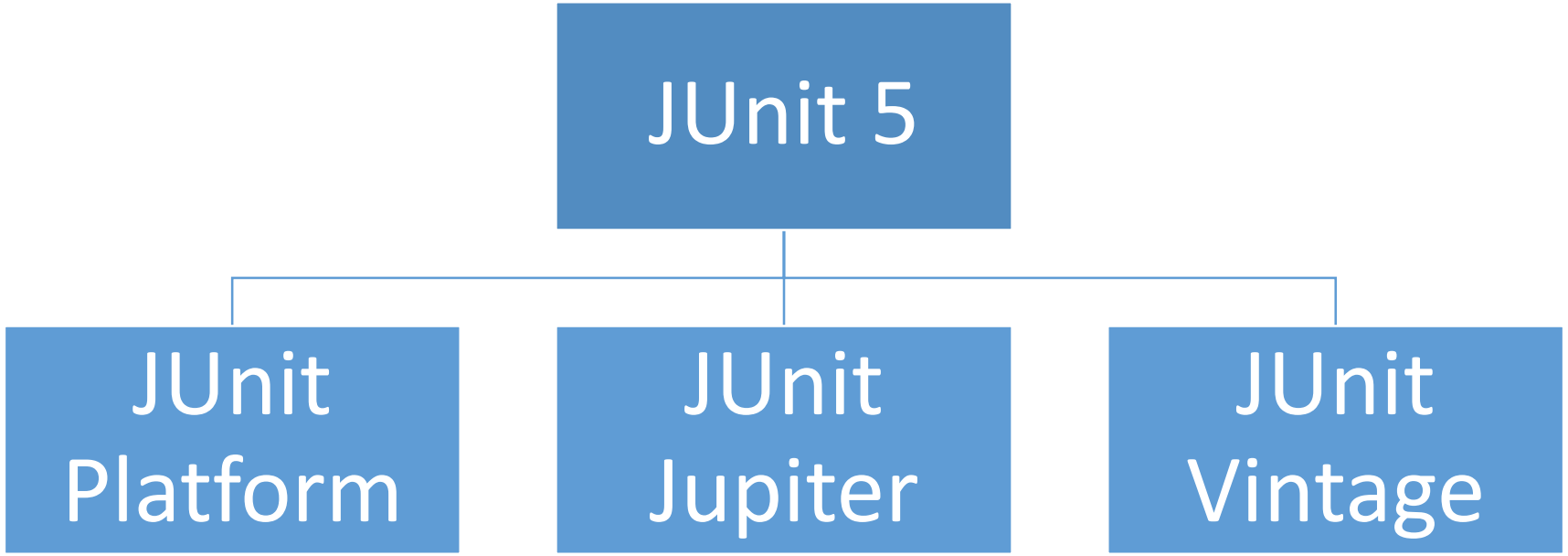
- Bisher JUnit 4
  - Geringe Modularität
  - Begrenzte Erweiterbarkeit
  - Java 8, Lambdas?

→ Crowdfunding Campaign

JUNIT 5

Wie ist JUnit 5 aufgebaut?

Saxonia Systems
So geht Software.

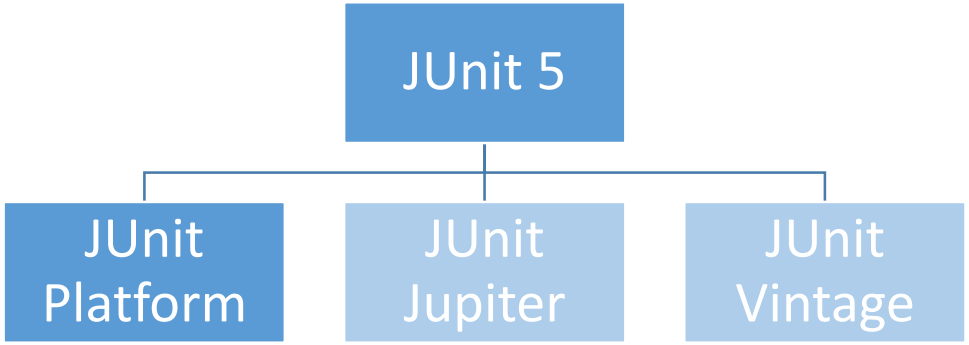


JUNIT 5

JUnit Platform

Saxonia Systems
So geht Software.

- Launcher API
  - Tests finden, filtern, ausführen
  - Test Engines finden
- Test Engine API
  - Test-Framework
  - Programmiermodell
  - Beispiel:
    - Jupiter, Vintage (JUnit 4)
- Built-in Launchers
  - Console Runner
  - Maven Plugin
  - Gradle Plugin
  - JUnit 4 Runner

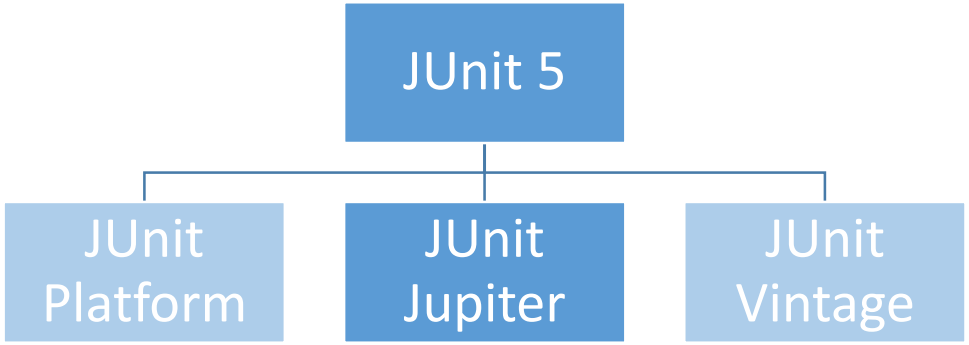


JUNIT 5

JUnit Jupiter

Saxonia Systems
So geht Software.

- JUnit 5 Test Engine
- JUnit 5 API
  - Annotationen
  - Assertions
- Extension Model

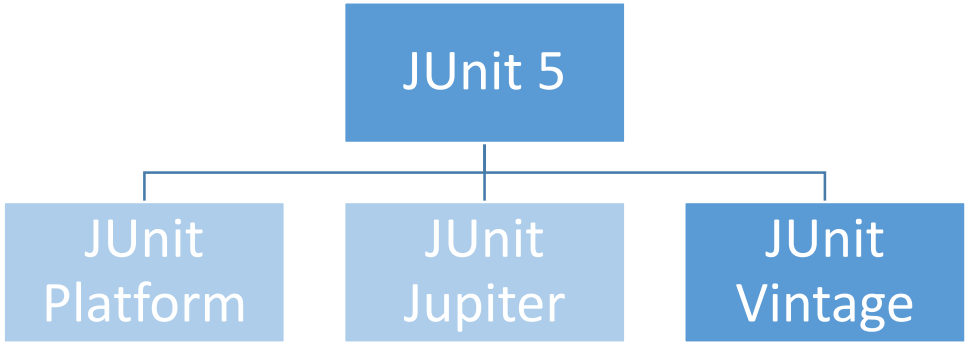


JUNIT 5

JUnit Vintage

Saxonia Systems
So geht Software.

- Test Engine für JUnit 3 und JUnit 4 Tests



Saxonia Systems
So geht Software.

Von JUnit 4 zu JUnit 5

VON JUNIT 4 ZU JUNIT 5

Saxonia Systems
So geht Software.

- Maven/Gradle
  - org.junit.platform:junit-platform-runner
  - org.junit.jupiter:junit-jupiter-api
  - org.junit.jupiter:junit-jupiter-engine

```
import org.junit.jupiter.api.Test;
import org.junit.platform.runner.JUnitPlatform;

@RunWith(JUnitPlatform.class)
public class TodoAppTest {

    @Test
    public void myTest() {
    }

}
```

Assertions

ASSERTIONS

Saxonia Systems
So geht Software.

org.junit.jupiter.api.Assertions

- Von JUnit 4 bekannte Assertions

```
assertEquals(9, 4 + 5);
assertTrue(myBoolean);
assertNull(result);
```

- Lambda-Erweiterungen

```
assertNull(result, () -> "result should be null");
```

- Grouped Assertions

```
assertAll("dim",
  () -> { assertEquals(200, dim.getHeight()); },
  () -> { assertEquals(300, dim.getWidth()); }
);
```

- Exception Assertions

```
MyException e = expectThrows(MyException.class,
  () -> { doSomething(); }
);
assertEquals("oh oh!", e.getMessage());
```

Test-Annotationen

TEST-ANNOTATIONEN

Saxonia Systems
So geht Software.

```
public MyTest {

   @BeforeAll
   static void setupClass() { … }

   @BeforeEach
   void setup() { … }

   @Test
   void someTest() { … }

   @AfterEach
   void tearDown() { … }

   @AfterAll
   static void tearDownClass() { … }

}
```

Assumptions

ASSUMPTIONS

Saxonia Systems
So geht Software.

org.junit.jupiter.api.Assumptions

- Tests nur unter bestimmten Bedingungen ausführen
  - z. B. nur im Buildserver, nur unter Windows etc.
- Tests werden abgebrochen, aber kein Testfehler

```
@Test
public void windowsOnly() {
   assumeTrue(System.getenv("OS").startsWith("Windows"));

   // test code here
}


@Test
public void buildServerOnly() {
   assumingThat("CI".equals(System.getenv("ENV")),
      () -> {
         // test code here
      }
   );
}
```

TESTS DEAKTIVIEREN

Saxonia Systems
So geht Software.

- Tests per Annotation deaktivieren
  - analog `@Ignore` in JUnit 4
- Tests werden übersprungen

```
@Disabled
public class DisabledTests() {

   void disabledTest() {
       // test code here
   }
}
```

```
@Test
@Disabled("needs a review")
void disabledTest() {
   // test code here
}
```

Tags

TAGS

Saxonia Systems
So geht Software.

- Tests markieren/gruppieren
  - ähnlich wie `@Category` in JUnit 4
- Eigene Tags sind möglich
- Können vom Launcher als Filter verwendet werden

```
@Tag("performance")
public void performanceTest() {
   // test code here
}

@Target({ ElementType.TYPE, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@Tag("performance")
public @interface PerformanceTest {
}

// Gradle plugin configuration
junitPlatform {
    platformVersion 1.0
    engines {
        include 'junit-jupiter'    }
    tags {
        include 'performance'
        // exclude 'fast', 'smoke'
    }
}
```

Tests strukturieren

TESTS STRUKTURIEREN

...und sprechend benennen

Saxonia Systems
So geht Software.

```
@DisplayName("TodoApp")
public class TodoAppTest {

    @Test
    @DisplayName("Should be empty when initialized")
    void init() { … }

    @Nested
    @DisplayName("Removing todos")
    class RemoveTodo {

        @Test
        @DisplayName("Todo list should be empty after deleting last todo")
        void removeLastTodo() { … }

        @Test
        @DisplayName("Should throw exception when deleting unknown todo")
        void removeUnknownTodo() { … }
    }
}
```

- de.sbley.junit5.TodoAppTest [Runner: JUnit 4] (0,026 s)
  - JUnit Jupiter (0,026 s)
    - TodoApp (0,026 s)
      - Should be empty when initialized (0,008 s)
      - Removing todos (0,018 s)
        - Should throw exception when deleting unknown todo (0,003 s)
        - Todo list should be empty after deleting last todo (0,015 s)

Saxonia Systems
So geht Software.

Interface Default-Methoden

INTERFACE DEFAULT-METHODEN

Saxonia Systems
So geht Software.

```
public interface EqualsContract<T> {

   T createValue();
   T createEqualValue();

   @Test
   default void reflexive() {
      T value = createValue();
      assertEquals(value, value);
   }

   @Test
   default void symmetric() {
      T value = createValue();
      T equalValue = createEqualValue();
      assertAll(() -> { assertEquals(value, equalValue); },
                () -> { assertEquals(equalValue, value); }
      );
   }
}
```

```
public class TodoTest implements EqualsContract<Todo> {

   public Todo createValue() { … }
   public Todo createEqualValue() { … }
}
```

Zusammenfassung

ZUSAMMENFASSUNG

Saxonia Systems
So geht Software.

- Komplette Neuentwicklung für Unittests in Java
- Java 8
- Modularer Aufbau
- Kernkonzepte
  - Assertions
  - Assumptions
  - BeforeEach, AfterEach, BeforeAll, AfterAll
  - Tags, @Disabled
  - @Nested, @DisplayName

ZUSAMMENFASSUNG

Saxonia Systems
So geht Software.

- Außerdem
  - Interface Default-Methoden
  - Extension Model
  - Integration mit Assertion-Frameworks
    - z. B. AssertJ, Hamcrest
  - Dependency Injection
  - Dynamische Tests

AUSBLICK

Tool-Unterstützung

Saxonia Systems
So geht Software.

- IDEs
  - Eclipse: Implementierung begonnen (4.7 Oxygene)
  - IntelliJ: IDEA 2016.2 kann JUnit 5 ☺
  - NetBeans: ?
- Build-Tools
  - Maven: prototypische Surefire-Erweiterung von den JUnit-Entwicklern
  - Gradle: Plugin von den JUnit-Entwicklern
- Buildserver
  - Jenkins: ?

AUSBLICK

Release

Saxonia Systems
So geht Software.

- Milestone M3 (Stand: Januar 2017)
- Release vermutlich im Frühjahr 2017
- https://github.com/junit-team/junit5

Vielen Dank

```
assertAll("Stefan Bley",
  () -> {
     assertEquals("https://www.twitter.com/sbley", onTwitter());
  },
  () -> {
     assertEquals("https://www.github.com/sbley", onGithub());
  }
);
```