NEUES VON JUNIT 5

FROM REVOLUTION TO CONTINUOUS EVOLUTION

KARLSRUHER ENTWICKLERTAG

MARC PHILIPP

Software Engineer bei Gradle

JUnit Committer seit 2012
Team Lead seit 2016

Twitter: @marcphilipp
Web: marcphilipp.de

KARLSRUHER ENTWICKLERTAG

SHOW OF HANDS

KARLSRUHER ENTWICKLERTAG

JUNIT 5 RELEASES

5.0 – 10. September 2017

5.1 – 18. Februar 2018

5.2 – 29. April 2018

5.3 – 11. September 2018

5.4 – 7. Februar 2019

5.5 – Juni 2019

KARLSRUHER ENTWICKLERTAG

AGENDA

1. Wie schreibt man Tests und Extensions mit JUnit 5?
2. Was ist die JUnit Platform und wozu ist sie gut?
3. Was kommt noch und die fängt man mit JUnit 5 an?

JUNIT JUPITER

DAS NEUE TESTING FRAMEWORK FÜR JAVA

Image: NASA

5

KARLSRUHER ENTWICKLERTAG

JUPITER? 🤔

Is writing tests rocket science now?

Nein, “Jupiter” ist einfach ein neuer Name zur besseren Unterscheidung der verschiedenen Teile von JUnit 5.

… und es ist der fünfte Planet von der Sonne aus gezählt.

BASICS (DEMO)

https://github.com/marcphilipp/junit5-demo/tree/20190603-etka

KARLSRUHER ENTWICKLERTAG

BASICS (RECAP)

- `@Test` ist jetzt in `org.junit.jupiter.api`
- `@Disabled` anstatt `@Ignore`
- `@BeforeAll`, `@BeforeEach`, `@AfterEach`, `@AfterAll` haben neue Namen
- `Assertions` sehen ähnlich aus – zusätzlich `assertThrows`, `assertAll`
- Eigene `@DisplayNames` statt Camel Case
- `@TestInstance(PER_METHOD` oder `PER_CLASS)`
- `@Tag` anstatt `@Category`

DISPLAY NAME GENERATORS 5.4

```
@DisplayNameGeneration(ReplaceUnderscores.class)
class A_year_is_not_supported {

    @Test
    void if_it_is_zero() {/*...*/}

    @ParameterizedTest
    @ValueSource(ints = { -1, -4 })
    void if_it_is_negative(int year) {/*...*/}
}
```

KARLSRUHER ENTWICKLERTAG

TEST METHOD ORDERING 5.4

- `@TestMethodOrder(Random.class)` hilft sicherzustellen, dass Tests nicht Reihenfolge-abhängig sind
- `@TestMethodOrder(Alphanumeric.class)` und `@TestMethodOrder(OrderAnnotation.class)` für Integrationstests
- Erweiterbar: `MethodOrderer` implementieren

KOTLIN SUPPORT 5.1

```
import org.junit.jupiter.api.*

class KotlinAssertionsDemo {
    @Test
    fun `expected exception testing`() {
        val exception = assertThrows<ArithmeticException> {
            Calculator().divide(1, 0)
        }
        assertEquals("/ by zero", exception.message)
    }
    @Test
    fun `grouped assertions`() {
        assertAll("Person properties",
            { assertEquals("Jane", person.firstName) },
            { assertEquals("Doe", person.lastName) }
        )
    }
}
```

WEITERE TESTARTEN (DEMO)

https://github.com/marcphilipp/junit5-demo/tree/20190603-etka

KARLSRUHER ENTWICKLERTAG

WEITERE TESTARTEN (RECAP)

- `@ParameterizedTest` mit versch. `@Source`-Annotationen
  - `@ValueSource`, `@EnumSource`, `@CsvSource`, `@CsvFileSource`, `@MethodSource`, `@NullSource` 5.4, `@EmptySource` 5.4, `@ArgumentsSource(MyProvider.class)`, `@YourCustomSource`
- `@RepeatedTest` für “flaky” Tests
- `@TestFactory` um *dynamisch* Tests zu produzieren

PARALLEL EXECUTION 5.3 (DEMO)

https://github.com/marcphilipp/junit5-demo/tree/20190603-etka

KARLSRUHER ENTWICKLERTAG

PARALLEL EXECUTION 5.3 (RECAP)

- Tests laufen standardmäßig sequenziell
- Parallele Ausführung lässt sich per Configuration Parameter aktivieren
- `@Execution(SAME_THREAD` oder `CONCURRENT)`
- `@ResourceLock` zur deklarativen Synchronisation

KARLSRUHER ENTWICKLERTAG

EXTENSIONS (DEMO)

https://github.com/marcphilipp/junit5-demo/tree/20190603-etka

KARLSRUHER ENTWICKLERTAG

EXTENSIONS (RECAP)

- Registrierung (beliebig viele gleichzeitig):
  - Deklarativ: `@ExtendWith` an Klassen oder Methoden
  - Programmatisch: `@RegisterExtension` an Feldern 5.1
  - Global: per `ServiceLoader` (s. User Guide)
- Implementierung:
  - `Extension` Marker Interface
  - *1* Extension – *n* Extension Points/Interfaces

KARLSRUHER ENTWICKLERTAG

COMPOSED ANNOTATIONS

Jupiter-Annotation können als Meta-Annotation verwendet werden, um eigene Annotationen zu definieren.

```
@Retention(RUNTIME)
@Target(METHOD)
@ExtendWith(DisabledOnWeekdaysExtension.class)
@Tag("example")
public @interface DisabledOnWeekdays {
    DayOfWeek[] value();
}
```

KARLSRUHER ENTWICKLERTAG

EXTENSION POINTS

- Lifecycle: `BeforeAllCallback`, `BeforeEachCallback`, `BeforeTestExecutionCallback`, `TestExecutionExceptionHandler`, `LifecycleMethodExecutionExceptionHandler` *5.5*, `AfterTestExecutionCallback`, `AfterEachCallback`, `AfterAllCallback`, `InvocationInterceptor` *5.5*
- Andere: `ExecutionCondition`, `ParameterResolver`, `TestInstanceFactory` *5.3*, `TestInstancePostProcessor`, `TestWatcher` *5.4*, `TestTemplateInvocationContextProvider`

TEMPORÄRE VERZEICHNISSE 5.4

```java
import org.junit.jupiter.api.io.TempDir;

@Test
void writeAndReadFile(@TempDir Path tempDir) throws Exception {
    Path testFile = tempDir.resolve("test.txt");

    Files.write(testFile, asList("foo", "bar"));

    List<String> actualLines = Files.readAllLines(testFile);
    assertIterableEquals(asList("foo", "bar"), actualLines);
}
```

KARLSRUHER ENTWICKLERTAG

BEDINGTE AUSFÜHRUNG 5.1

- `@EnabledOnOs` / `@DisabledOnOs({LINUX, MAC, …})`
- `@EnabledOnJre` / `@DisabledOnJre({JAVA_11, …})`
- `@Enabled` / `DisabledIfSystemProperty(named = "someKey", matches = "someValue")`
- `@Enabled` / `DisabledIfEnvironmentVariable(named = "SOME_KEY", matches = "SOME_VALUE")`

DEKLARATIVE TIMEOUTS 5.5

```
@BeforeEach
@Timeout(10)
void setUp() {
    // schlägt fehl, falls Ausführung länger als 10 Sekunden dauert
}
@Test
@Timeout(value = 500, unit = MILLISECONDS)
void someTest() {
    // schlägt fehl, falls Ausführung länger als 500 Millisekunden dauer
}
```

KARLSRUHER ENTWICKLERTAG

THIRD-PARTY EXTENSIONS

JUnit Pioneer, Spring, Mockito, Testcontainers, Docker, Wiremock, JPA, Selenium/WebDriver, DbUnit, Kafka, Jersey, GreenMail, S3Mock, Citrus Framework, XWiki, …

https://github.com/junit-team/junit5/wiki/Third-party-Extensions

KARLSRUHER ENTWICKLERTAG

AGENDA

1. Wie schreibt man Tests und Extensions mit JUnit 5? ✅
2. Was ist die JUnit Platform und wozu ist sie gut?
3. Was kommt noch und die fängt man mit JUnit 5 an?

KARLSRUHER ENTWICKLERTAG

JUNIT PLATFORM

PLATTFORM ZUM TESTEN AUF DER JVM

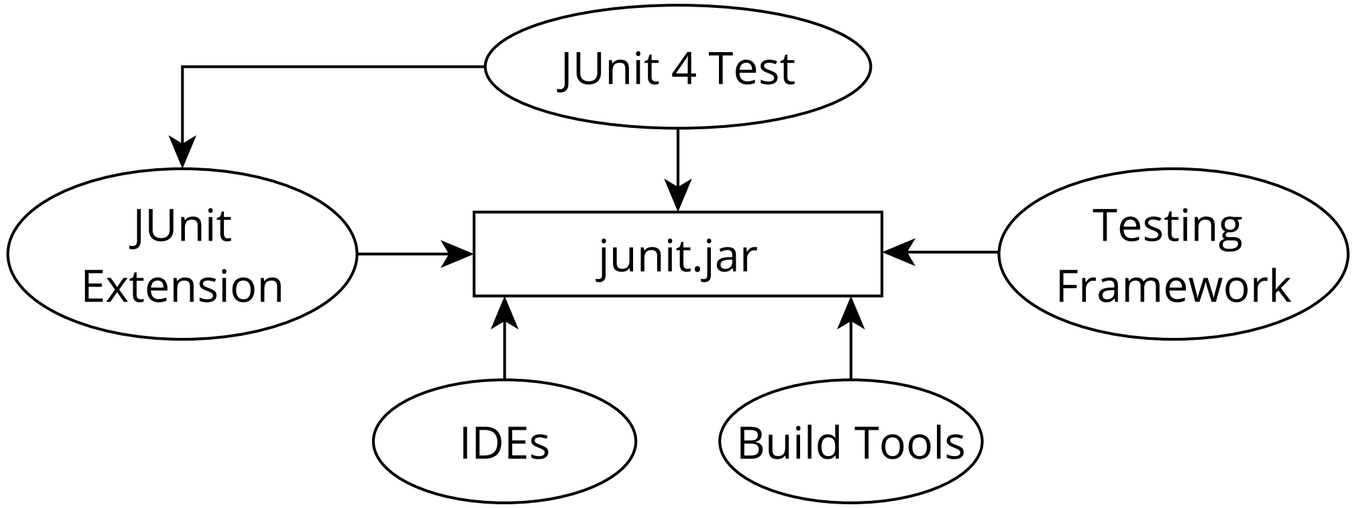
Image: NASA

5

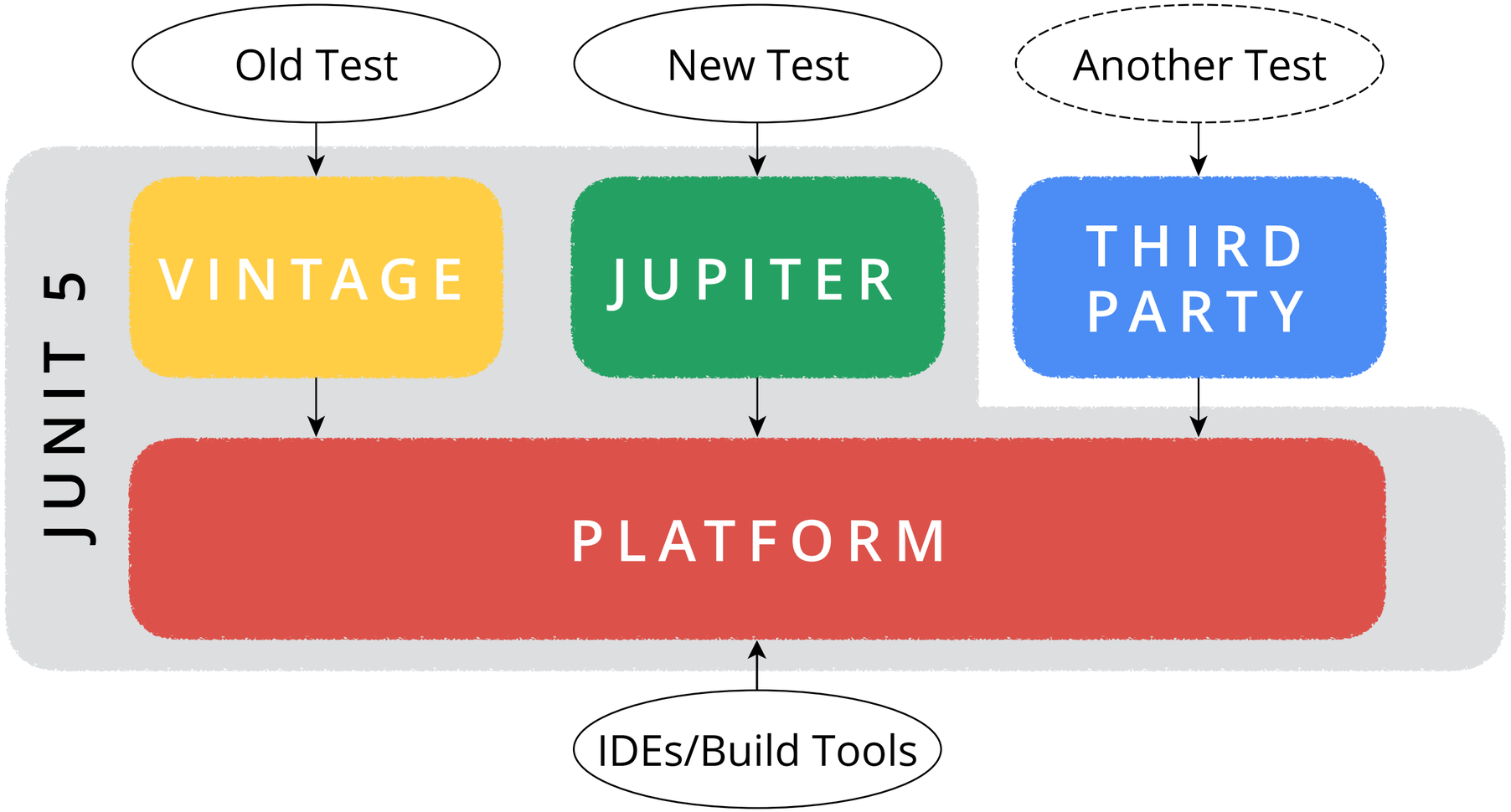
KARLSRUHER ENTWICKLERTAG

JUNIT ALS PLATTFORM?

- JUnit war schon immer eine Plattform
  - für IDEs und Build Tools
  - für andere Testing Frameworks
- Enge Kopplung (interne APIs, Reflection, Serialization)



Wenn JUnit eine Plattform ist, dann sollten wir die Architektur darauf auslegen!


Old Test
New Test
Another Test
JUNIT 5
VINTAGE
JUPITER
THIRD PARTY
PLATFORM
IDEs/Build Tools

JUNIT 5

=

JUPITER + VINTAGE + PLATFORM

KARLSRUHER ENTWICKLERTAG

THIRD-PARTY ENGINES

Specsy, Spek, KotlinTest, Cucumber, Drools, jqwik, Brahms, Mainrunner, …

https://github.com/junit-team/junit5/wiki/Third-party-Extensions

KARLSRUHER ENTWICKLERTAG

MEHRERE TEST ENGINES (DEMO)

https://github.com/marcphilipp/junit5-platform-demo

KARLSRUHER ENTWICKLERTAG

MEHRERE TEST ENGINES (RECAP)

- Mehrere Test Engines können in einem Testlauf verwendet werden
- Unterscheidung zwischen Abhängigkeiten in `testImplementation` und `testRuntimeOnly`
- Erlaubt eine schrittweise Migration von einer Test Engine zur anderen (z.B. von Vintage nach Jupiter)

KOMPATIBILITÄT / MIGRATION

- Vintage Engine führt JUnit 3/4 Tests auf der Plattform aus
- `@Category(UI.class)` entspricht `com.acme.UI`-Tag
- Teilweise Unterstützung von JUnit 4 `Rules`
- Unterstützung von `@Ignore` 5.4
- IDEs können Testklassen in die Jupiter API konvertieren
- Migrationswerkzeug aus der Community:
  https://github.com/junit-pioneer/convert-junit4-to-junit5

BUILD TOOLS

- Gradle (≥ 4.6), Ant (≥ 1.10.3) und Maven Surefire (≥ 2.22.0) unterstützen die JUnit Platform
- Mit dem `ConsoleLauncher` kann man Tests über die Kommandozeile ausführen und in andere Build Tools (z.B. Bazel) integrieren

IDES

- Sehr gute Unterstützung
  - IntelliJ IDEA (≥ 2016.2)
  - Eclipse (≥ 4.7.1a)
  - Visual Studio Code (Java Test Runner ≥ 0.4.0)
  - Netbeans (≥ 10.0)
- Für andere IDEs gibt es `@RunWith(JUnitPlatform)`

KARLSRUHER ENTWICKLERTAG

TAG EXPRESSIONS 5.1

Erlauben präzise anzugeben, welche Tests basierend auf Tags ausgeführt werden sollen:

```
test {
    useJUnitPlatform {
        includeTags("(smoke & feature-a) | (!smoke & feature-b)")
    }
}
```

KARLSRUHER ENTWICKLERTAG

UNTERSTÜTZUNG FÜR MODULE 5.1

Alle Tests in einem Modul ausführen:

```
$ java -jar junit-platform-console-standalone-1.4.0.jar \
       --select-module com.acme.foo
```

Den Modulpfad scannen:

```
$ java -jar junit-platform-console-standalone-1.4.0.jar \
       --scan-modules
```

KARLSRUHER ENTWICKLERTAG

AGENDA

1. Wie schreibt man Tests und Extensions mit JUnit 5? ✅
2. Was ist die JUnit Platform und wozu ist sie gut? ✅
3. Was kommt noch und die fängt man mit JUnit 5 an?

KARLSRUHER ENTWICKLERTAG

ROADMAP UND RESSOURCEN

Image: NASA

ROADMAP

- Wiederverwendbare Discovery für Test Engines **5.5 M1**
- Testausführung in nutzerdefinierten Threads **5.5 RC1**
- Deklarative/globale Timeouts **5.5 RC1**
- Reporting-Format, das neue Features unterstützt (z.B. Tags, Display Names, Report Entries) **5.6 M1**
- Deklarative Test Suiten
- Parametrisierte Testklassen
- *Eure Ideen?*

LOSLEGEN? LOSLEGEN!

- User Guide:
  http://junit.org/junit5/docs/current/user-guide/
- Beispielprojekte für Ant, Bazel, Gradle und Maven:
  https://github.com/junit-team/junit5-samples
- Javadoc:
  http://junit.org/junit5/docs/current/api/

KARLSRUHER ENTWICKLERTAG

GESUCHT: FEEDBACK!

- StackOverflow:
  http://stackoverflow.com/questions/tagged/junit5
- Code & Issues:
  https://github.com/junit-team/junit5/
- Chat mit dem Team:
  https://gitter.im/junit-team/junit5
- Twitter:
  https://twitter.com/junitteam

KARLSRUHER ENTWICKLERTAG

UNTERSTÜTZT DAS JUNIT-TEAM!

https://junit.org/sponsoring

BEISPIEL-CODE

- Jupiter:
  https://github.com/marcphilipp/junit5-demo/tree/20190603-etka
- Platform:
  https://github.com/marcphilipp/junit5-platform-demo

KARLSRUHER ENTWICKLERTAG

FRAGEN?

@marcphilipp / @junitteam auf Twitter

KARLSRUHER ENTWICKLERTAG

VIELEN DANK!

KARLSRUHER
ENTWICKLERTAG