ENTAGO

# JUnit Migration von Version 4 auf Version 5

*Die fünfte Version von JUnit wurde bereits Ende 2017 veröffentlicht. Viele Software-Projekte, vor allem im Enterprise-Umfeld, setzen noch auf die vierte Version, die Migration steht noch aus. Dieser Artikel geht auf einige interessante Neuerungen in Version 5 ein und erklärt darüber hinaus eine schrittweise Migration von JUnit 4 auf 5.*

JUnit ist eines der weit verbreiteten Frameworks für das Implementieren von Tests in Java.

In Version 5 bringt es einige hilfreiche und interessante Neuerungen:

- Die neue Version erlaubt mehrere «Testrunner» auf einmal zu verwenden. In Version 4 war nur ein Runner (beispielsweise SpringJUnit4ClassRunner) möglich.
- JUnit 5 baut auf Java 8 auf. Somit können die Java 8 Features wie z.B. Lambdas nativ verwendet werden.
- @DisplayName, damit kann ein lesbarer Name definiert werden, welcher bei Ausführung auf der Konsole und in der IDE ausgegeben wird.
- Die «Assertions» lassen sich gruppieren. Normalerweise läuft der JUnit-Test, bis eine «Assertion» fehlschlägt. Durch die Gruppierung werden alle Assertions ausgeführt und angezeigt, welche nicht funktioniert haben.
- Parametrisierte Tests: Testmethoden können in JUnit 5 Parameter entgegennehmen, dies funktioniert wie eine Art Dependency-Injection. Für den Parameter kann eine Liste von Eingangswerten definiert werden, der Test wird dann für jeden Wert ausgeführt.

Darüber hinaus besteht JUnit 5 aus drei Paketen: JUnit Platform, Jupiter und Vintage. Abbildung 1 zeigt den Aufbau von JUnit 5. JUnit Platform ist der zugrunde liegende Baustein für das Framework.

Dieser wird durch Maven, die IDE und Build Tools aufgerufen. Das Modul «Jupiter» beinhaltet die API, welche für das Schreiben von JUnit 5 Test-Cases verwendet wird. Z.B. ist «@DisplayName» ein Teil von Jupiter. Zu guter Letzt wird das Modul «Vintage» für das Durchführen von JUnit 4 Tests benötigt.

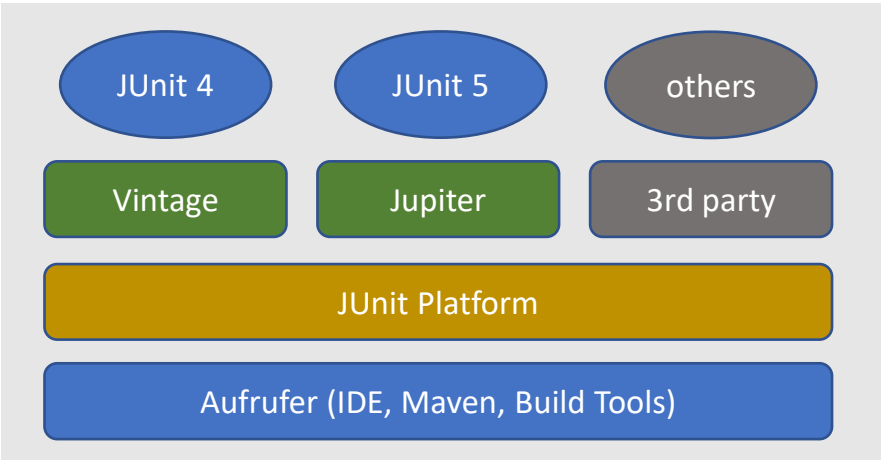


*Abbildung 1: Architektur JUnit 5*

ENTAGO

## Migration

Falls bis jetzt JUnit 4 in der Applikation verwendet wurde, kann im ersten Schritt für die Migration die Abhängigkeit ausgetauscht werden. Das folgende Listing zeigt das Beispiel an «maven».

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13</version>
    <scope>test</scope>
</dependency>
```

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.6.2</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.junit.vintage</groupId>
    <artifactId>junit-vintage-engine</artifactId>
    <version>5.6.2</version>
    <scope>test</scope>
</dependency>
```

Nach Austausch der Abhängigkeiten lässt sich das Projekt wie zuvor kompilieren. Durch Verwendung des Moduls «Vintage» ist ein Parallelbetrieb von JUnit4 und JUnit5 möglich.

Für das «Refactoring» von bestehenden JUnit 4 Tests zur JUnit5 API müssen folgende Schritte beachtet werden:

## Test-Annotation

Bis jetzt konnten Parameter in der @Test-Annotation verwendet werden, um beispielsweise auf Exceptions im Testfall zu reagieren, oder um ein Timeout einzurichten. Diese Parameter gibt es nicht mehr, stattdessen wird «assertThrows» beziehungsweise «assertTimeout» aus der JUnit 5 Assertions API verwendet. Das folgende Listing zeigt die Änderung von JUnit 4 zu JUnit 5.

```
@Test(expected = Exception.class)
public void throwSomeException() throws Exception {
   // Aufruf
}
```

```
@Test
public void throwSomeException() throws Exception {
  Assertions.assertThrows(Exception.class, () -> {
    // Aufruf
  });
}
```

## Weitere Annotationen

Anschliessend muss jede JUnit 4 Test Klasse zu JUnit 5 migriert werden. Der Name einiger Annotationen wurde geändert. In folgender Tabelle ist ersichtlich, welche JUnit 4 Annotation mit welcher JUnit 5 Annotation ersetzt werden muss.

| JUnit 4 | JUnit 5 |
|---|---|
| @Before | @BeforeEach |
| @After | @AfterEach |
| @BeforeClass | @BeforeAll |
| @AfterClass | @AfterAll |
| @Ignore | @Disabled |

## ExtendWith-Annotation

In JUnit 4 wurde «@RunWith» verwendet, um einen bestimmten Kontext der Applikation im Test zu verwenden. Mit Spring ist ein Beispiel dafür @RunWith(SpringJunit4ClassRunner.class). In JUnit 5 kann das durch @ExtendWith(SpringExtension.class) ersetzt werden. Des Weiteren müssen alle Instanzen von @Rule und @ClassRule ebenfalls durch die entsprechende @ExtendWith Annotation ersetzt werden. Die jeweils zu verwendende Klasse (wie bei Spring) findet sich in der entsprechenden Dokumentation.

## Checkstyle Prüfung

Um während der Migration automatisiert zu testen, wo noch JUnit 4 verwendet wird, empfiehlt sich folgende Checkstyle Violation:

```
<module name="IllegalImport">
    <property name="illegalPkgs"
              value="org.junit"/>
    <message key="import.illegal"
              value="Usage of Junit 4"/>
</module>
```

Entago AG
Buckhauserstrasse 34
CH-8048 Zurich

info@entago.ch

www.entago.ch