# Junit Basics

## **Anleitung: Entwickle ersten JUnit Test “Code first”**

Gegeben: SUT: **ch03.x01firstJUnitTest.Money**

Test: **ch03.x01firstJUnitTest.MoneyTest**

## **Junit Assertions**

Folgende Liste von Junit-Assertions aus der JUnit-JavaDoc möge Ihnen als Referenz dienen:

<http://junit.sourceforge.net/javadoc/org/junit/Assert.html>

## **Anleitung/Übung: Das Verhalten von Asserts**

Folgende Fragen soll durch Ausprobieren (z.B. mit Erweitern eines Tests und anschließendem Debuggen des Tests) beantwortet werden:

1. Werden *alle* assertXXX-Aufrufe einer @Test-Methode ausgeführt, selbst wenn eine fehlschlägt? - Nein
2. Falls eine Test-Methode fehlschlägt, werden dann die restlichen Test-Methoden dieser Klasse noch ausgeführt? - Ja

Siehe **ch03.x01firstJUnitTest.lösung1.MoneyTest**

## **Übung**

Schreibe MoneyTest.java um, sodass assertTrue() verwendet wird. Vergleiche die Lesbarkeit von Fehlermeldungen in Eclipse durch Schreiben eines fehlschlagenden Tests

**Lösung:**

**ch03.x01firstJUnitTest.lösung2.MoneyTest**

=> schlechter lesbar (nur boolean)

## **Demo: Tests ausschließen mit JUnit @Ignore und @Assume**

Mit @Ignore schließt man einen Test aus. Mit den statischen Methoden aus org.junit.Assume.\* formuliert man eine Precondition unter der ein Test läuft.

Tests: **ch03.x02JUnitExcludeTests.IgnoreTest** und **ch03.x02JUnitExcludeTests.AssumeTest**

## **Unit Tests mit Maven: Das Surefire Plugin**

Aus: <http://maven.apache.org/surefire/maven-surefire-plugin/examples/inclusion-exclusion.html> :

|  |
| --- |
| By default, the Surefire Plugin will automatically include all test classes with the following wildcard patterns:   * "\*\*/Test\*.java" - includes all of its subdirectories and all Java filenames that start with "Test". * "\*\*/\*Test.java" - includes all of its subdirectories and all Java filenames that end with "Test". * "\*\*/\*TestCase.java" - includes all of its subdirectories and all Java filenames that end with "TestCase".   If the test classes do not follow any of these naming conventions, then configure Surefire Plugin and specify the tests you want to include. |

### **Übung: Ausschließen von Tests nach Beispielen aus der Maven-Doku zum Surefire Plugin**

1. Probieren Sie, alle bisherigen Unit-Testfälle durch eine Einstellung in der pom.xml auszuschließen.
2. Schließen Sie nur den letzten Test ein. Erweitern Sie dafür diesen test um gut sichtbare Ausgaben oder erstellen Sie eine Kopie, benennen Sie sie um und erweitern diesen dann um gut sichtbare Ausgaben z.B. als Datei ch03.x01firstJUnitTest.lösung3.MoneyTest1
3. Was passiert, wenn in der pom.xml auf eine Testklasse verwiesen wird, die nicht existiert?

## **Integration Tests mit Maven**

Aus: <https://maven.apache.org/surefire/maven-failsafe-plugin/>

|  |
| --- |
| The Failsafe Plugin is designed to run integration tests while the Surefire Plugin is designed to run unit tests. The name (failsafe) was chosen both because it is a synonym of surefire and because it implies that when it fails, it does so in a safe way.  The Maven lifecycle has four phases for running integration tests:   * pre-integration-test for setting up the integration test environment. * integration-test for running the integration tests. * post-integration-test for tearing down the integration test environment. * verify for checking the results of the integration tests.   If you use the Surefire Plugin for running tests, then when you have a test failure, the build will stop at the integration-test phase and your integration test environment will not have been torn down correctly.  The Failsafe Plugin is used during the integration-test and verify phases of the build lifecycle to execute the integration tests of an application. The Failsafe Plugin will not fail the build during the integration-test phase, thus enabling the post-integration-testphase to execute.  NOTE: when running integration tests, you should invoke Maven with the (shorter to type too)  mvn verify  rather than trying to invoke the integration-test phase directly, as otherwise the post-integration-test phase will not be executed. |

## **Anleitung: Integration Tests mit Maven einrichten und Unit-Tests und Integration Tests separat ausführen können**

1. Das Failsafe-Plugin für das Projekt in der plugins-Section der pom.xml aktivieren, wie unter <https://maven.apache.org/surefire/maven-failsafe-plugin/usage.html> beschrieben:

|  |
| --- |
| <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-failsafe-plugin</artifactId>  <version>2.16</version>  <executions>  <execution>  <goals>  <goal>integration-test</goal>  <goal>verify</goal>  </goals>  </execution>  </executions>  </plugin> |

1. Einen Test schreiben, der der Namenskonvention unter <https://maven.apache.org/surefire/maven-failsafe-plugin/examples/inclusion-exclusion.html> folgt:

|  |
| --- |
| By default, the Failsafe Plugin will automatically include all test classes with the following wildcard patterns:   * "\*\*/IT\*.java" - includes all of its subdirectories and all Java filenames that start with "IT". * "\*\*/\*IT.java" - includes all of its subdirectories and all Java filenames that end with "IT". * "\*\*/\*ITCase.java" - includes all of its subdirectories and all Java filenames that end with "ITCase". |

Klasse **ch03.x03IntegrationTestsLösung.SomeDatabaseIntegrationTestIT:**

|  |
| --- |
| **public** **class** SomeDatabaseIntegrationTestIT {  @Test  **public** **void** testSelect(){  System.***err***.println("########### NOW RUNNING: SomeDatabaseIntegrationTestIT.testSelect()###########");  }  } |

1. Benutze Flags in den Plugins Surefire und Failsafe, um Unit Tests rsp. Integration Tests auszuschalten  
   Für Failsafe:  
   [https://maven.apache.org/surefire/maven-failsafe-plugin/verify-mojo.html#skipITs](https://maven.apache.org/surefire/maven-failsafe-plugin/verify-mojo.html" \l "skipITs) :“ Set this to 'true' to skip running integration tests, but still compile them. Its use is NOT RECOMMENDED, but quite convenient on occasion.“  
     
   [https://maven.apache.org/surefire/maven-failsafe-plugin/verify-mojo.html#skipTests](https://maven.apache.org/surefire/maven-failsafe-plugin/verify-mojo.html" \l "skipTests) :“ Set this to 'true' to skip running tests, but still compile them. Its use is NOT RECOMMENDED, but quite convenient on occasion.“

Für Surefire:  
<https://maven.apache.org/surefire/maven-surefire-plugin/examples/skipping-tests.html> :“To skip running the tests for a particular project, set the **skipTests** property to **true**.”

Dort speziell die Idee unter

|  |
| --- |
| Skipping by default  If you want to skip tests by default but want the ability to re-enable tests from the command line, you need to go via a properties section in the pom:  (…)  This will allow you to run with tests disabled by default and to run them with this command:  mvn install -DskipTests=false |

Dies Idee aufgegriffen und sowohl auf Surefire wie auch auf Failsafe angewendet:

|  |
| --- |
| <properties>  **<skipTests>false</skipTests>**  **<skipITs>${skipTests}</skipITs>**  **<skipUTs>${skipTests}</skipUTs>**  </properties>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-plugin</artifactId>  <version>2.19.1</version>  <configuration>  **<skipTests>${skipUTs}</skipTests>**  </configuration>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-failsafe-plugin</artifactId>  <version>2.16</version>  <executions>  <execution>  <goals>  <goal>integration-test</goal>  <goal>verify</goal>  </goals>  </execution>  </executions>  <configuration>  <skipTests>${skipTests}</skipTests>  <skipITs>${skipITs}</skipITs>  </configuration>  </plugin>  </plugins>  </build> |

1. Zur leichteren Überprüfung aktivieren Sie in der pom.xml nur einen Unit Test, der eine gut sichtbare Ausgabe erzeugt (gemäß der Lösung von 3.6 b ).
2. Zur Erinnerung: Die Maven Lifecycle-Phasen des default-Lifecycles sind nach <https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html> :

|  |
| --- |
| For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the [Lifecycle Reference](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html" \l "Lifecycle_Reference)):   * validate - validate the project is correct and all necessary information is available * compile - compile the source code of the project * test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed * package - take the compiled code and package it in its distributable format, such as a JAR. * integration-test - process and deploy the package if necessary into an environment where integration tests can be run * verify - run any checks to verify the package is valid and meets quality criteria * install - install the package into the local repository, for use as a dependency in other projects locally * deploy - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.   These lifecycle phases (plus the other lifecycle phases not shown here) are executed sequentially to complete the default lifecycle.  (…) To do all those, you only need to call the last build phase to be executed, in this case, deploy:  mvn deploy  That is because if you call a build phase, it will execute not only that build phase, but also every build phase prior to the called build phase. |

1. Somit ergeben sich folgende Möglichkeiten für den Aufruf:  
   **mvn verify** - Führt Unit Tests und Integration Tests aus  
   **mvn verify -DskipITs** – Führt nur Unit Test aus, skipt Integration Tests

**mvn verify -DskipUTs** – Führt nur Integration Tests aus, skipt Unit Tests

**mvn verify -DskipTests** – Führt weder Unit Tests noch Integration Tests aus

(credit <https://gist.github.com/delor/5916802> )

1. Natürlich kann man diese Flags auch in Maven-Profilen verwenden, so z.B:

|  |
| --- |
| <profiles>  <profile>  <id>integration</id>  <properties>  <skipUTs>true</skipUTs>  <skipITs>false</skipITs>  </properties>  </profile>  </profiles> |

Aufruf mit

**mvn -P integration verify**

oder

**mvn verify -P integration**

1. Nun die Integration Tests parallel laufen lassen.
   1. Erschaffen Sie dazu einen zweiten Integration Test und sorgen Sie dafür, dass dieser, wie auch der erste, sichtbar lange dauern:

|  |
| --- |
| **public** **class** SomeDatabaseIntegrationTestIT {  @Test  **public** **void** testSelect(){  **try** {Thread.*sleep*(5000);} **catch** (InterruptedException e) {}  System.***err***.println("########### NOW RUNNING: SomeDatabaseIntegrationTestIT.testSelect()###########");  }    @Test  **public** **void** testLongRunningSelect(){  **try** {Thread.*sleep*(5000);} **catch** (InterruptedException e) {}  System.***err***.println("########### NOW RUNNING: SomeLongRunningDatabaseIntegrationTestIT.testLongRunningSelect()###########");  }  } |

## **Anleitung: Konfigurieren Sie das Failsafe-Plugin für die parallele Abarbeitung von Integration Tests**

## Unter <https://maven.apache.org/surefire/maven-failsafe-plugin/examples/junit.html> heißt es:

|  |
| --- |
| Running Tests in Parallel From JUnit 4.7 onwards you can run your tests in parallel. To do this, you must set the parallel parameter, and may change the threadCount or useUnlimitedThreads attribute.  **(…)**  If your tests specify any value for the parallel attribute and your project uses JUnit 4.7+, your request will be routed to the concurrent JUnit provider, which uses the JUnit JUnitCore test runner.  This is particularly useful for slow tests that can have high concurrency. |

In unserer pom.xml angewendet:

|  |
| --- |
| <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-failsafe-plugin</artifactId>  <version>2.16</version>  <configuration>  **<parallel>methods</parallel>**  **<threadCount>10</threadCount>**  </configuration>  (…)  </plugin> |

Lassen Sie **mvn verify** einmal ohne und einmal mit dieser configuration laufen und beobachten Sie den Unterschied (auch in der Ausgabe der abgelaufenen Zeit).