Vorwort

JUnit ist ein »ganz alter Hut« und **die** Standardbibliothek zum Schreiben automatisierter Tests in Java. Ich gehe davon aus, dass die meisten Java-Programmierer bei ihrer täglichen Arbeit JUnit-Tests schreiben. Was also, werden Sie sich vielleicht fragen, soll dieses Buch über eine Bibliothek, die Sie wahrscheinlich tagtäglich selbst einsetzen?

Ich benutze JUnit seit inzwischen über 10 Jahren und dachte deshalb noch vor einiger Zeit, dass ich mich ganz gut mit dieser Bibliothek auskenne. Trotzdem habe ich in den letzten zwei Jahren – sehr zu meiner eigenen Überraschung – noch viel Neues über JUnit dazugelernt. Und genau das ist der Grund, warum ich dieses Buch geschrieben habe und denke, dass auch Sie es lesen sollten. Oder wissen Sie bereits, was es mit dem `Theories`-Runner auf sich hat? Oder wie Sie testübergreifende Aspekte elegant in eigene `TestRule`-Klassen auslagern können? Oder vielleicht wie Sie einzelne Testmethoden mittels der `@Category`-Annotation verschiedenen Testgruppen zuordnen und anschließend nur Tests einer bestimmten Gruppe ausführen können?

All dies sind recht neue und leider auch recht unbekannte Features von JUnit. Der Grund hierfür ist, dass die letzte Major-Version von JUnit – die Version 4.0 – bereits im Jahr 2006 veröffentlicht wurde und dass über die vielen Verbesserungen und neuen Möglichkeiten, die in den letzten Jahren im Laufe mehrerer Minor-Versionen zu JUnit hinzugekommen sind, nur sehr wenig berichtet wurde. Nach der Lektüre dieses Buches werden Sie einen guten Überblick über alle Features haben, die JUnit 4.11 bietet, und Sie werden wissen, wann Sie diese sinnvoll einsetzen können.

Neben diesen Features gibt es noch einige weitere Open-Source-Bibliotheken – zum Beispiel Mockito und FEST Fluent Assertions –, die das Schreiben von JUnit-Tests erleichtern und die ich deshalb sehr schätze und Ihnen mit diesem Buch nahebringen möchte.

Da die Kenntnis der JUnit-API allein nicht ausreicht, um gut verständliche, leicht wartbare, stabile und schnelle Tests zu schreiben, soll dieses Buch gleichzeitig auch ein Ratgeber dafür sein. Hierzu finden Sie in allen Kapiteln mehrere einfache Regeln, die ich auch immer durch Beispiele veranschauliche. Oft stammen diese Beispiele aus Tests bekannter Open-Source-Projekte.

Apropos Beispiele: Sie werden fast auf jeder Seite etwas Quellcode finden, da ich denke, dass Quellcode das beste Kommunikationsmedium ist, wenn es ums Programmieren geht. Dieses Buch ist kein akademisches Werk, sondern ein Buch für Praktiker.

Und da sich JUnit nicht nur zum Schreiben von Unit-Tests eignet, gebe ich Ihnen auch Tipps für das Schreiben von Integrations-, Frontend-, Performance-, Stress- und Architekturtests. Gerade die Architekturtests (mit denen Sie automatisch überprüfen können, ob sich Ihr gesamter Quellcode bei voranschreitender Entwicklung immer noch an Ihre Architekturvorgaben hält) sind ein Thema, das mir besonders am Herzen liegt, da es recht unbekannt ist.

Abgerundet wird das Buch durch Hinweise und Tipps, wie Sie JUnit effektiv zusammen mit den bekannten Java-IDEs Eclipse und IntelliJ IDEA sowie zusammen mit den Build-Tools Ant und Maven einsetzen können.

Auch wenn das Schreiben dieses Buches – übrigens mein erstes Buch – anstrengender und zeitraubender war, als ich anfänglich dachte, bin ich doch sehr stolz auf das Ergebnis. Ich hoffe, Sie haben viel Spaß beim Lesen und können viel Neues lernen. Und falls Sie vielleicht irgendwelche Anmerkungen haben, freue ich mich natürlich über jede Art von Feedback, am besten einfach per E-Mail an: mail@michaeltamm.de

Michael Tamm
Berlin, im August 2013

Vorkenntnisse

Dieses Buch ist für alle Java-Programmierer bestimmt, egal ob Sie noch nie etwas von JUnit gehört oder schon Erfahrung im Schreiben von Tests mit JUnit haben. Selbst wenn Sie schon sehr viele Tests mit JUnit programmiert haben, bin ich überzeugt davon, dass Sie trotzdem noch eine Menge lernen können, wenn Sie dieses Buch lesen.

Neben der Tatsache, dass Sie grundlegende Kenntnisse der Java-Programmierung haben sollten, gehe ich davon aus, dass Sie wissen, was eine Jar-Datei ist, und dass Sie Jar-Dateien von Open-Source-Projekten downloaden und zum Klassenpfad eines Java-Projekts hinzufügen können.

Falls Sie Ant + Ivy, Maven oder Gradle für automatische Builds benutzen, sollten Sie in der Lage sein, eine neue Abhängigkeit zu Ihrem Projekt hinzuzufügen (oder zumindest den Kollegen kennen, der Ihnen dabei helfen kann). Wann immer ich eine Open-Source-Bibliothek näher vorstelle, zeige ich Ihnen, welche Zeilen zu einer *pom.xml*-Datei hinzugefügt werden müssen, um die Bibliothek in ein Maven-Projekt einzubinden. Die dabei angegebenen Maven-Koordinaten (*groupId*, *artifactId* und *version*) können Sie natürlich ebenfalls für Ivy oder Gradle benutzen.