KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

ISSN 1865-682X

02/2014

Jahrgang 7

Editorial

Der Joel Test

Manager sind immer darum bemüht herauszufinden wie gut eigentlich die eigene Software-Entwicklungsabteilung ist. Da gibt es die verschiedensten Methoden, die leider auch beliebig aufwändig sind. Es gibt aber auch einen ganz einfachen und schnellen Test, der auch sofort zu überzeugenden Ergebnissen führt: den Joel Test.

Vor über dreizehn Jahren beschrieb der bekannte Blogger Joel Spolsky einen einfachen Test, wie man in nur zwölf Schritten feststellen kann, wie gut ein Software-Team ist [1]. Der wohlverdient (von ihm selbst) nach ihm benannte Joel Test besteht aus nur 12 Fragen, die sich in der Regel sofort mit einem *Ja* oder *Nein* beantworten lassen. Hier eine sehr freizügige Übersetzung des dazugehörigen Fragenkatalogs:

1. Benutzen Sie eine Versionskontrolle?
2. Können Sie Ihre Software in einem Schritt zusammen bauen?
3. Bauen Sie Ihre Software täglich zusammen?
4. Haben Sie eine Fehlerdatenbank?
5. Beheben Sie Fehler bevor Sie Neues entwickeln?
6. Haben Sie einen aktuellen Zeitplan?
7. Haben Sie eine Spezifikation?
8. Haben die Entwickler eine ruhige Arbeitsumgebung?
9. Benutzen Sie die besten Tools, die man kaufen kann?
10. Haben Sie Tester?
11. Müssen Kandidaten bei Ihnen im Einstellungsgespräch Code schreiben?
12. Führen Sie Benutzerfreundlichkeitstests auch im Flur durch?

Die Fragen sind eigentlich selbsterklärend. Allenfalls Frage 12 bedarf einer Erläuterung. Für den Flur-Test (engl. *hallway usability testing*) gehen Sie einfach hinaus auf den Gang, greifen sich die nächstbeste Person, die vorbeikommt, und lassen diese einen Blick darauf werfen, was Sie gerade entwickeln. Übrigens können Sie, wenn Sie das mit nur 5 Personen machen, 95 % der Probleme ausmachen.

Der Test, auch wenn schon relativ alt, gibt einen sehr schönen Anhaltspunkt, wie es um Ihre Software-Entwicklung steht. Binnen drei Minuten können Sie herausfinden, wie gut Sie wirklich sind, ohne dass Sie irgendein kompliziertes Verfahren anwenden oder gar einen Auditor kommen lassen müssen.

Die Auswertung ist gleichermaßen einfach wie aussagekräftig: Geben Sie sich (oder Ihrem Team) einfach für jedes *Ja* einen Punkt. Haben Sie dabei die volle Punktzahl 12 erreicht, so ist das ausgezeichnet. Haben Sie nur 11 Punkte erzielt, so kann man das gerade noch als gut durchgehen lassen. Aber bei 10 und weniger Punkten haben Sie im Prinzip schon „verloren“. Herr Spolsky glaubte damals, dass die meisten Software entwickelnden Firmen nur eine Punktzahl von 2 oder 3 erreichen und das nur Unternehmen wie Microsoft bei diesem Test immer mit der besten Punktzahl abschneiden. Aber in den letzten 10 Jahren dürfte sich die Durchschnittspunktzahl deutlich verbessert haben.

Der Blog-Eintrag zum Joel Test ist immer noch lesenswert, und wer ihn noch nicht kennt, sollte bei Gelegenheit einmal hineinschauen. Dort wird auch die Motivation jeder einzelnen Frage erläutert. An dieser Stelle wollte ich Ihnen nur die Möglichkeit geben, auf die Schnelle überprüfen zu können, ob Sie unter optimalen Bedingungen arbeiten.

Also, wenn Sie weniger als 11 Punkte bekommen haben, sollten Sie einmal mit den Projektverantwortlichen darüber reden, dass Sie vermutlich mit verhältnismäßig wenig Aufwand deutlich mehr erreichen könnten.

In diesem Sinne wünsche ich Ihnen ein produktives Schaffen.

Ihr Michael Wiedeking
Herausgeber

[1] Spolsky, Joel *Joel on Software – The Joel Test: 12 Steps to Better Code*, 9. August 2010, http://www.joelonsoftware.com/articles/fog0000000043.html

Beitragsinformation

Der KaffeeKlatsch dient Entwicklern, Architekten, Projektleitern und Entscheidern als Kommunikationsplattform. Er soll neben dem Know-how-Transfer von Technologien (insbesondere Java und .NET) auch auf einfache Weise die Publikation von Projekt- und Erfahrungsberichten ermöglichen.

Beiträge

Um einen Beitrag im KaffeeKlatsch veröffentlichen zu können, müssen Sie prüfen, ob Ihr Beitrag den folgenden Mindestanforderungen genügt:

- Ist das Thema von Interesse für Entwickler, Architekten, Projektleiter oder Entscheider, speziell wenn sich diese mit der Java- oder .NET-Technologie beschäftigen?
- Ist der Artikel für diese Zielgruppe bei der Arbeit mit Java oder .NET relevant oder hilfreich?
- Genügt die Arbeit den üblichen professionellen Standards für Artikel in Bezug auf Sprache und Erscheinungsbild?

Wenn Sie uns einen solchen Artikel, um ihn in diesem Medium zu veröffentlichen, zukommen lassen, dann übertragen Sie Bookware unwiderruflich das nicht exklusive, weltweit geltende Recht

- diesen Artikel bei Annahme durch die Redaktion im KaffeeKlatsch zu veröffentlichen
- diesen Artikel nach Belieben in elektronischer oder gedruckter Form zu verbreiten
- diesen Artikel in der Bookware-Bibliothek zu veröffentlichen
- den Nutzern zu erlauben diesen Artikel für nicht-kommerzielle Zwecke, insbesondere für Weiterbildung und Forschung, zu kopieren und zu verteilen.

Wir möchten deshalb keine Artikel veröffentlichen, die bereits in anderen Print- oder Online-Medien veröffentlicht worden sind.

Selbstverständlich bleibt das Copyright auch bei Ihnen und Bookware wird jede Anfrage für eine kommerzielle Nutzung direkt an Sie weiterleiten.

Die Beiträge sollten in elektronischer Form via E-Mail an redaktion@bookware.de geschickt werden.

Auf Wunsch stellen wir dem Autor seinen Artikel als unveränderlichen PDF-Nachdruck in der kanonischen KaffeeKlatsch-Form zur Verfügung, für den er ein unwiderrufliches, nicht-exklusives Nutzungsrecht erhält.

Leserbriefe

Leserbriefe werden nur dann akzeptiert, wenn sie mit vollständigem Namen, Anschrift und E-Mail-Adresse versehen sind. Die Redaktion behält sich vor, Leserbriefe – auch gekürzt – zu veröffentlichen, wenn dem nicht explizit widersprochen wurde.

Sobald ein Leserbrief (oder auch Artikel) als direkte Kritik zu einem bereits veröffentlichten Beitrag aufgefasst werden kann, behält sich die Redaktion vor, die Veröffentlichung jener Beiträge zu verzögern, so dass der Kritisierte die Möglichkeit hat, auf die Kritik in der selben Ausgabe zu reagieren.

Leserbriefe schicken Sie bitte an leserbrief@bookware.de. Für Fragen und Wünsche zu Nachdrucken, Kopien von Berichten oder Referenzen wenden Sie sich bitte direkt an die Autoren.

Werbung ist Information

Firmen haben die Möglichkeit Werbung im KaffeeKlatsch unterzubringen. Der Werbeteil ist in drei Teile gegliedert:

- Stellenanzeigen
- Seminaranzeigen
- Produktinformation und -werbung

Die Werbeflächen werden als Vielfaches von Sechsteln und Vierteln einer DIN-A4-Seite zur Verfügung gestellt.

Der Werbeplatz kann bei Frau Natalia Wilhelm via E-Mail an anzeigen@bookware.de oder telefonisch unter 09131/8903-16 gebucht werden.

Abonnement

Der KaffeeKlatsch erscheint zur Zeit monatlich. Die jeweils aktuelle Version wird nur via E-Mail als PDF-Dokument versandt. Sie können den KaffeeKlatsch via E-Mail an abo@bookware.de oder über das Internet unter www.bookware.de/abo bestellen. Selbstverständlich können Sie das Abo jederzeit und ohne Angabe von Gründen sowohl via E-Mail als auch übers Internet kündigen.

Ältere Versionen können einfach über das Internet als Download unter www.bookware.de/archiv bezogen werden.

Auf Wunsch schicken wir Ihnen auch ein gedrucktes Exemplar. Da es sich dabei um einzelne Exemplare handelt, erkundigen Sie sich bitte wegen der Preise und Versandkosten bei Natalia Wilhelm via E-Mail unter natalia.wilhelm@bookware.de oder telefonisch unter 09131/8903-16.

Copyright

Impressum

KaffeeKlatsch Jahrgang 7, Nummer 2, Februar 2014
ISSN 1865-682X
Bookware – eine Initiative der
MATHEMA Verwaltungs- und Service-Gesellschaft mbH

Henkestraße 91, 91052 Erlangen
Telefon: 0 91 31 / 89 03-0
Telefax: 0 91 31 / 89 03-55
E-Mail: redaktion@bookware.de
Internet: www.bookware.de

Herausgeber/Redakteur: Michael Wiedeking
Anzeigen: Natalia Wilhelm
Grafik: Nicole Delong-Buchanan

Inhalt

Artikel

Kolumnen

SQL vs. NoSQL – Es kann nur einen geben?

von Marc Schwering

Datenbanken haben bereits eine über 40-jährige Geschichte hinter sich. NoSQL-Technologie dagegen ist innerhalb dieser Entwicklung noch eine recht junge Erscheinung – allerdings eine, die bereits große Veränderungen mit sich gebracht hat und auch in Zukunft nach sich ziehen wird. Aber wie wird es weitergehen? Heißt es nun SQL vs. NoSQL ? Standard vs. Newcomer? Oder wird die Zukunft zeigen, dass beide Modelle sich ergänzen?

Arquillian

von Saschas Gross

Testen macht Spaß. Testen ist blöd. Testen bringt mir nichts. Ich schreibe die Tests immer zuerst. Ich brauche keine Tests, ich mache keine Fehler. Für Tests hab ich keine Zeit. Für Tests werde ich nicht bezahlt. Meine Software ist zu kompliziert um in vertretbarem Aufwand Tests zu schreiben. Tests helfen die Software stabil und wartbar zu halten.

Geteiltes Vergnügen

von Michael Wiedeking

Im letzten Vergnügen wurde bereits eine Lösung für eine vorzeichenlose Division auf Basis einer vorzeichenbehafteten gezeigt. Allerdings wurde dabei frecherweise auf das nächst größere Maschinenwort zugegriffen, was beim größten Maschinenwort ja leider nicht mehr möglich ist. Wenn man sich aber ein paar Gedanken macht, dann schafft man es auch mit vorzeichenbehafteten Werten gleicher Größe die Division vorzeichenlos durchzuführen.

SQL vs. NoSQL – Es kann nur einen geben?

NoSQL-Datenbanken auf dem Vormarsch

von Marc Schwering

Datenbanken haben bereits eine über 40-jährige Geschichte hinter sich. NoSQL - Technologie dagegen ist innerhalb dieser Entwicklung noch eine recht junge Erscheinung – allerdings eine, die bereits große Veränderungen mit sich gebracht hat und auch in Zukunft nach sich ziehen wird. Aber wie wird es weitergehen? Heißt es nun SQL vs. NoSQL ? Standard vs. Newcomer? Oder wird die Zukunft zeigen, dass beide Modelle sich ergänzen?

Der lange Weg zu NoSQL

Die *Structured Query Language* (SQL)-Technologie wurde in den 1970er Jahren entwickelt und ist somit etwa 40 Jahre alt. NoSQL ist dagegen erst seit etwa 2010 definiert und als Lösung verfügbar. In dieser Zeit standen Google oder Facebook vor der Herausforderung, rasant wachsende Datenberge zu verwalten: 13 Millionen Anfragen pro Sekunde oder das Importieren von 100 GB Indexdaten jeden Tag. Doch 2010 war es für die vorherrschenden relationalen Datenbanken unmöglich, solche Datenmengen zu verarbeiten – ihr Maximum war bei einer halben Million Transaktionen pro Sekunde erreicht. Facebook schloss behelfsweise verschiedene Datenbanken zusammen – doch auch diese reichten noch immer nicht aus, um der Daten Herr zu werden. Eine andere Lösung musste Abhilfe schaffen – NoSQL. Und so kam es, dass sich NoSQL-Datenbanken schnell als Nischenlösung etablierten, da nur sie *Big Data* hinreichend bewältigen können.

Doch die Datenmengen, die 2010 eher außergewöhnlich waren und NoSQL zu einem Randphänomen machten, wurden in vielen Unternehmen und Anwendungen bald zur Normalität. Befeuert durch weitere Trends wie *Cloud Computing* und *Mobility* wuchsen die Anforderungen an die Verarbeitung und Skalierbarkeit von Daten noch stärker. Langsam befreiten sich NoSQL-Datenbanken wie *Cassandra* und *MongoDB* aus ihrem Nischendasein. Diese Entwicklung konnte vor 45 Jahren wohl noch niemand vorhersehen, als IBM 1969 das erste intelligente System für die Verwaltung von hierarchischen und Transaktions-Datenbanken, IMS, vorstellte, später gefolgt von dem kommerziellen Datenbankenverwaltungssystem *Integrated Data Store* (IDS). Es dauerte dann noch ein gutes Jahrzehnt bis schließlich die erste SQL-Technologie entwickelt wurde – und erstaunliche Möglichkeiten bot.

SQL stößt an seine Grenzen

Von Anfang an wurden Datenbanktechnologien auf einem *Query*- und Schema-Design entwickelt. Sie waren zwar sehr leistungsfähig, konnten jedoch nur genau für die geplanten Anwendungen entwickelt werden und ließen keine späteren Veränderungen mehr zu. SQL schaffte hier eine Neuentwicklung mit der Trennung der beiden Designs. So spielt am Anfang der Entwicklung nur das Schema eine Rolle. Wegen solcher Möglichkeiten der ersten SQL-Technologie sind relationale Datenbanken auch über 30 Jahre nach ihrer Vorstellung 1983 noch immer das vorherrschende Modell auf dem Markt. Da sie aber inzwischen an ihre Grenzen stoßen, können sie nicht mehr die alleinige Lösung für alle Probleme sein.

Doch wie genau machen sich diese Grenzen bemerkbar? Die Basis der funktionalen Logik von dokumentorientierten Datenbanken ist die Beziehung zwischen spezifischen Reihen. Die SQL-Datenbank erreicht ihre Leistungsgrenze, wenn viele Beziehungen über mehrere Reihen geschaffen werden müssen. Dies führt dazu, dass Entwickler auf einmal viel Zeit und Ressourcen für das Datenbankschema aufwenden müssen. Außerdem engt das feste relationale Schema die Funktionalität von SQL-Datenbanken ein.

In Zeiten von *Big Data* übersteigen die Anforderungen der steigenden Datenmengen also oft die Leistungsfähigkeit von SQL-Technologien. Probleme bereitet SQL aber nicht nur die Menge der Daten, sondern auch ihre verschiedenen Typen bzw. polymorphe Schemata. Große Datenberge sind nicht einheitlich und Datenbanken

müssen gleichzeitig strukturierte, unstrukturierte, semistrukturierte oder polymorphe Daten verarbeiten. Hier zeigt die NoSQL-Technologie ihre Stärke und ihr Potential, sich als neuer Standard durchzusetzen.

Eine Nasenlänge vorn

NoSQL-Datenbanken erleichtern durch ihre Dynamik die Entwicklung von Anwendungen, da diese auch zu einem späteren Zeitpunkt im Entwicklungsprozess verändert werden können. So können Entwickler im Vergleich zu einem relationalen Schema, bei dem sie schon von Anfang an alle Schritte vorhersehen müssen, Zeit sparen. Das verringert den Koordinationsaufwand und ermöglicht es, Daten schneller zu verarbeiten. Die horizontale Skalierbarkeit macht außerdem Methoden wie *JOINs* und Transaktionen, die in relationalen Datenbanken vorkommen, bei NoSQL unnötig. Und schließlich bieten dokumentorientierte NoSQL-Datenbanken mit besserer lokaler Datenspeicherung, *In-Memory Caching* und *In-Place-Updates* auch noch eine hervorragende Leistung.

David gegen Goliath?

Natürlich soll das nicht heißen, dass relationale Datenbanken in jedem Punkt nachteilig sind. Tatsächlich besitzen sie einige Stärken wie zum Beispiel eine hohe Funktionalität. NoSQL-Datenbanken sind nicht für jede Anwendung die einzige oder beste Wahl. Aber sie bieten etwa 80 Prozent dieser Funktionalitäten – ohne *JOINs* oder komplexe Transaktionen, dafür mit höherer Leistung und verbesserter Skalierbarkeit. Ihre Stärken zeigen sie bei *Cloud Computing* und zeitgemäßer Datenanalyse. Und bei solchen Anwendungsfällen wird die Nutzung von NoSQL-Technologie zunehmen, da sie besser als relationale Datenbanken auf diese Trends antworten kann. Es ist also nicht so, dass es nur einen Gewinner geben kann – beide Technologien haben ihre Existenzberechtigung. Doch langfristig wird es zu einem Wechsel mit NoSQL als neuem Standard an der Spitze kommen.

Kurzbiografie

Marc Schwering (marc.schwering@mongodb.com) ist Solutions Architect bei MongoDB Inc. mit langjährigem Hintergrund im Bereich der Web-Technologien. Er beschäftigt sich mit agiler Software-Entwicklung und deren Umsetzung im (online)Einzelhandel und Themen wie Industrie 4.0 / Internet der Dinge.

Arquillian

Testen auf dem Server – Eine Einführung

von Saschas Gross

> Testen macht Spaß.
> Testen ist blöd.
> Testen bringt mir nichts. Ich schreibe die Tests immer zuerst.
> Ich brauche keine Tests, ich mache keine Fehler.
> Für Tests hab ich keine Zeit. Für Tests werde ich nicht bezahlt. Meine Software ist zu kompliziert um in vertretbarem Aufwand Tests zu schreiben.Tests helfen die Software stabil und wartbar zu halten.

Tests zu schreiben kann Spaß machen, muss es aber nicht. Umso einfacher das zu erstellende System ist, desto einfacher ist es, dafür Tests zu schreiben. Algorithmen, bei denen eine Abbildung von A nach B erfolgt, sind relativ einfach zu testen.

```
@Test
public void toUpper() {
   Assert.assertEquals(
      "HALLO WELT!", "Hallo Welt!".toUpperCase()
   );
}
```

Kommen zu dem entwickelten System aber externe Systeme hinzu, beispielsweise eine Datenbank oder ein Webservice, die angebunden werden müssen, wird es schon aufwendiger und unspaßiger. Die externen Systeme müssen in den Test eingebunden oder emuliert/gemockt werden, um die üblichen CRUD-Operationen *Create*, *Read*, *Update* und *Delete* zu verifizieren. Das Dilemma dabei ist, dass dafür ziemlich viel Code geschrieben werden muss und spezielle Frameworks für Tests, wie z. B. *DBUnit* oder *Mockito*, eingebunden werden müssen, um sich die Arbeit zu erleichtern. Dagegen ist auch gar nichts zu sagen, außer dass es aufwendig ist und das entsprechende Wissen vorhanden sein muss.

Ist die Zielumgebung der Anwendung auch noch im *Java-EE*-Umfeld angesiedelt, d. h. wir haben *EJB*, Transaktionen, *JPA* und die sonst noch zur Verfügung stehenden Features der Java EE, dann wird es immer aufwendiger die Umgebung für den Test zu emulieren.

Da stellt sich die Frage, ob es nicht einfacher ist, den Test in einer Umgebung laufen zu lassen, in der die externen Systeme vorhanden sind und nicht nachgebildet werden müssen und auch Java-EE-Features, so wie später im Produktionssystem, zur Verfügung stehen.

Arquillian bietet die Möglichkeit die Tests im Applikations-Server, wie z. B. *JBoss AS/Wildfly* oder *Glassfish* auszuführen, also in der Umgebung in der auch der Produktiv-Code ausgeführt wird.

Das Gute an *Arquillian* ist, dass man für den Einsatz nicht wirklich viel wissen muss und die Tests dann wieder ganz einfach werden, und auch das Transaktionsverhalten ohne Aufwand getestet werden kann.

Von Vorteil ist, wenn die Anwendung mit *Maven* oder *Gradle* gebaut wird, damit man sich nicht um die ganzen Abhängigkeiten zu Bibliotheken kümmern muss. Im Folgenden wird Maven für den *Build* verwendet.

```
@RunWith(Arquillian.class)
public class ArquillianSimpleStringTest {

   @Deployment
   public static JavaArchive createDeployment() {
      return ShrinkWrap.create(JavaArchive.class);
   }

   @Test
   public void toUpper() {
      Assert.assertEquals(
         "HALLO WELT!", "Hallo Welt!".toUpperCase()
      );
   }
}
```

Der Test *toUpper()* wird jetzt nicht mehr lokal ausgeführt sondern in einem Applikations-Server, im Folgenden ein *JBoss AS 7.1.1.* Damit der Test im Applikations-Server läuft, muss man ein paar Einstellungen vornehmen.
Zuerst die offensichtlichsten Änderungen im Test-Code:

1. Den Test markieren, damit er von *Arquillian* ausgeführt wird.
   @RunWith(Arquillian.class)
2. Erzeugung eines deploybaren Artefakts (*jar*,*war*,...). In der Testklasse muss mindestens eine statische Methode mit *@Deployment* annotiert sein, die ein Archiv zurückliefert, in dem der zu testende Code enthalten ist, nicht die Tests selbst. Da der Test *toUpper() java.lang.String* testet, also keine eigene Klasse, wird hier ein leeres Archiv zurückgegeben.

Damit der Test-Code auch auf einem Server ausgeführt werden kann, wird dieser natürlich auch benötigt und *Arquillian* muss mitgeteilt werden, wo sich dieser befindet.

3. Hierfür gibt es die Konfigurationsdatei *arquillian.xml*, die im Verzeichnis *src/test/resources/* liegen muss.

```
<arquillian xmlns="http://jboss.org/schema/arquillian"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://jboss.org/schema/
    arquillian/arquillian_1_0.xsd"
>
  <container qualifier="jbossas-managed" default="true">
    <configuration>
      <property name="jbossHome">arquillian/jboss/
        jboss-as-7.1.1.Final</property>
    </configuration>
  </container>
</arquillian>
```

Mit der Property *jbossHome* wird der Pfad zum Server angegeben. Mit *<container qualifier="jbossas-managed">* wird *Arquillian* mitgeteilt, dass *Arquillian* den Server managen darf, d. h. hoch- und herunterfahren zum Testbeginn und -ende. Es gibt je nach Server zusätzlich noch die Möglichkeit von *Embedded* und *Remote*. Bei *Remote* hat *Arquillian* dann nicht mehr die Kontrolle über den Server und darf nur noch *deployen* und *undeployen*. Die Konfiguration in *arquillian.xml* ändert sich dann ebenfalls. Unterschiedliche Anbindungen diverser Server finden sich im *Reference Guide* [2] unter *Container adapters* [3].

Den Server, hier den JBoss AS 7.1.1, kann man unter *JBoss Application Server Downloads* [4] manuell herunterladen und eben mit dem *Buildtool*. Bei Maven hätte man in der *pom.xml* einen Abschnitt im */project/build/plugins*

```
<plugin>
  <artifactId>maven-dependency-plugin</artifactId>
  <executions>
    <execution>
      <id>unpack</id>
      <phase>process-test-classes</phase>
      <goals>
        <goal>unpack</goal>
      </goals>
      <configuration>
        <artifactItems>
          <artifactItem>
            <groupId>org.jboss.as</groupId>
            <artifactId>jboss-as-dist</artifactId>
            <version>7.1.1.Final</version>
            <type>zip</type>
            <overWrite>false</overWrite>
            <outputDirectory>arquillian/jboss</
              outputDirectory>
          </artifactItem>
        </artifactItems>
      </configuration>
    </execution>
  </executions>
</plugin>
```

oder in einem Profil, das nur aktiviert wird, wenn der Server noch nicht heruntergeladen und ausgepackt wurde.

```
<profile>
  <id>arquillianUnpackJBoss</id>
  <activation>
    <file>
      <missing>arquillian/jboss</missing>
    </file>
  </activation>
  ...
</profile>
```

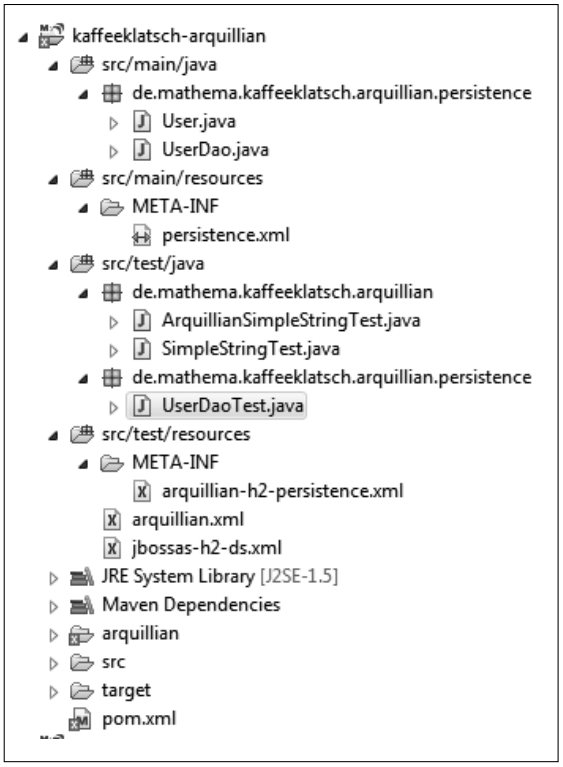
Arquillian und Java EE

Nachdem die Grundlagen von *Arquillian* gelegt sind, widmen wir uns jetzt einer winzigen Java-EE-Anwendung mit einer Entität *User*, die durch ein *UserDao* gespeichert, gelöscht und gefunden werden kann. Für den *UserDao* erstellen wir einen Test.

Die Entität *User* hat drei Attribute *id*, *name* und *mail*. Der *UserDao* hat die Operationen *save(), findByName(name), deleteAll()* und *findAll()*. Für *findByName()* und *findAll()* sind Tests in der Klasse *UserDaoTest* implementiert.

```
@Entity
public class User {

  @Id
  @GeneratedValue
  private Long id;
  private String name;
  private String mail;

  public User() {
    super();
  }

  public User(String name, String mail) {
    super();
    this.name = name;
    this.mail = mail;
  }
  …

}
@Stateless
public class UserDao {
  @PersistenceContext(name = "kaffeeklatsch")
```

```
    private EntityManager entityManager;

    public User save(User user) {
        entityManager.persist(user);
        return user;
    }

    public List<User> findAll() {
        TypedQuery<User> query =
            entityManager.createQuery(
                "from User u", User.class
            );
        return query.getResultList();
    }

    public List<User> findByName(String name) {
        TypedQuery<User> query = entityManager.createQuery(
            "from User u where u.name = :name", User.class
        );
        query.setParameter("name", name);
        return query.getResultList();
    }

    public void deleteAll() {
        Query query = entityManager.createQuery(
            "delete from User u"
        );
        query.executeUpdate();
    }
}

@RunWith(Arquillian.class)
public class UserDaoTest {

    @Deployment
    public static WebArchive createDeployment() {
    return ShrinkWrap
        .create(WebArchive.class)
        .addClass(UserDao.class)
        .addClass(User.class)
        .addAsWebInfResource(
            EmptyAsset.INSTANCE, "beans.xml"
        )
        .addAsResource(
            "META-INF/arquillian-h2-persistence.xml",
            "META-INF/persistence.xml"
        )
        .addAsWebInfResource(
            "jbossas-h2-ds.xml", "jbossas-ds.xml"
        );
    }

    @Inject
    private UserDao userDao;
    @Before
    public void before() {
        userDao.save(new User(
            "sascha", "sascha.gross@mathema.de"
        ));
        userDao.save(new User(
            "ursula", "ursula@trash-mail.com"
        ));
        userDao.save(new User(
            "max", "max@trash-mail.com"
        ));
    }

    @After
    public void after() {
        userDao.deleteAll();
    }

    @Test
    public void findAll() {
        List<User> users = userDao.findAll();
        Assert.assertNotNull(users);
        Assert.assertEquals(3, users.size());
    }

    @Test
    public void findByName() {
        List<User> users = userDao.findByName("sascha");
        Assert.assertNotNull(users);
        Assert.assertEquals(1, users.size());
        Assert.assertEquals("sascha", users.get(0).getName());
        Assert.assertEquals(
            "sascha.gross@mathema.de", users.get(0).getMail()
        );
    }

    @Test
    public void findByNameNone() {
        List<User> users = userDao.findByName("mister x");
        Assert.assertNotNull(users);
        Assert.assertEquals(0, users.size());
    }
}
```



Ab Java EE 6.0 reicht es ein WAR zu deployen, in dem auch EJBs enthalten sind. Genau das geschieht im *createDeployment()* in *UserDaoTest*. Zusätzlich zu den Klassen werden noch die Dateien *persistence.xml, beans.xml* (wegen *CDI*) und *jbossas-ds.xml* (legt im JBoss AS automatisch beim Deployment eine *DataSource* an, die in der *persistence.xml* mit JPA verwendet wird) hinzugefügt.

In *UserDaoTest* sind zusätzlich die Methoden *before()* und *after()* enthalten, welche Testdaten vor Beginn der Tests anlegen und danach auch wieder wegräumen.

Soviel als Einführung in die grundlegende Funktionalität von *Arquillian*. Für die Tests wurden keine *Mocks* benötigt. Würde man lokal testen, müsste der *EntityManager* gemockt werden, womit aber die Abfragen zur Datenbank nicht mit getestet würden und der Test somit an und für sich recht fraglich wäre. Alternativ könnte man auch lokal eine Datenbank anbinden. Müsste dann aber in der *persistence.xml* wieder die Entitäten angeben, was mit *Arquillian* nicht notwendig ist und der *EntityManager* müsste auch durch irgendeine Magie in *UserDao* injekted werden. Bei *Arquillian* ist es die Magie der Java EE.

Arquillian erleichtert das Testen in einem Java-EE-Projekt ungemein, da man sich wieder auf die fachlichen Tests beschränken kann und die technischen Aspekte in den Hintergrund rücken.

Die Sourcen zu diesem Artikel können unter [7] heruntergeladen werden.

Referenzen

[1] Arquillian *home*
http://arquillian.org

[2] Arquillian *Reference Guide*
https://docs.jboss.org/author/display/ARQ/Reference+Guide

[3] Arquillian *Container adapters*
https://docs.jboss.org/author/display/ARQ/Container+adapters

[4] JBoss *Application Server Downloads*
http://www.jboss.org/jbossas/downloads

[5] DbUnit *Sourceforge.net*
http://dbunit.sourceforge.net

[6] mockito
https://code.google.com/p/mockito

[7] Sourcecode
www.bookware.de/kaffeeklatsch/KK_Sourcecode_2014-02-002.zip

Kurzbiographie

Sascha Gross ist als Software-Entwickler, Consultant und Trainer für die MATHEMA Software GmbH tätig. Seine Spezialgebiete sind die Oberflächen- und Web-Programmierung mit Java. Insbesondere beschäftigt er sich stets mit den Neuerungen der Java-Welt (z. B. JSF) und gibt sein Wissen gerne als Referent auf verschiedenen Konferenzen weiter. Außerdem besitzt er Expertenwissen über XML, das er bereits als Buchautor veröffentlicht hat. Neben seiner Projekttätigkeit hält er Technologie-Trainings für MATHEMA und unterstützt Projekte beim Technologieumstieg auf JSF.

Des Programmierers kleine Vergnügen

Geteiltes Vergnügen

von Michael Wiedeking

Im letzten Vergnügen wurde bereits eine Lösung für eine vorzeichenlose Division auf Basis einer vorzeichenbehafteten gezeigt. Allerdings wurde dabei frecherweise auf das nächst größere Maschinenwort zugegriffen, was beim größten Maschinenwort ja leider nicht mehr möglich ist. Wenn man sich aber ein paar Gedanken macht, dann schafft man es auch mit vorzeichenbehafteten Werten gleicher Größe die Division vorzeichenlos durchzuführen.

Das Problem mit der vorzeichenlosen Division p/q tritt ja nur dann auf, wenn der Zähler p oder Nenner q (oder beide) das Vorzeichen-Bit gesetzt haben, also bei vorzeichenbehafteter Division als negative Zahl interpretiert werden. Diese Fälle muss man deshalb gesondert betrachten.

Ist beim Divisor q das Vorzeichen-Bit gesetzt, also $q < 0$, dann ist diese Zahl, wenn man sie als vorzeichenlose Zahl interpretiert, sehr groß. Jetzt gibt es zwei Möglichkeiten: entweder ist a) der Dividend p auch „negativ" oder *b*) der Dividend ist „positiv". Im Fall b) ist q sicher größer als p und bei dem typischen Rundungsverhalten der ganzzahligen Division muss das Ergebnis 0 sein.

Im Fall a) ist das nicht so leicht entscheidbar, also muss geprüft werden, ob p größer oder kleiner als q ist, wobei beide als vorzeichenlos interpretiert werden müssen. Ist p dann kleiner als q, so muss das Ergebnis wieder 0 sein. Ist p größer als q kann das Ergebnis nur 1 sein. Das wissen wir deshalb so genau, weil dann bei beiden das Vorzeichen-Bit gesetzt und damit der größtmögliche Wert $2^n - 1$ und der kleinstmögliche 2^{n-1} ist; $2^n/2^{n-1}$ wäre gleich 2, aber bei einem Zähler von $2^n - 1$ reicht es eben nur bis zur 1.

```
if (q < 0) {
  if (uLess(p, q)) {
    return 0;
  } else {
    return 1;
  }
}
```

Die *uLess*-Methode zum vorzeichenlosen Vergleich, ob $p < q$ ist, wurde auch schon im letzten Vergnügen [1] beschrieben und steht uns damit auch zur Verfügung.

Jetzt bleibt nur noch der unangenehme Fall, dass auch der Divisor p „negativ" sein kann. Natürlich könnte man auch hier wieder eine Fallunterscheidung machen, aber es würde doch viel einfacher gehen, wenn sich nur irgendwie das Vorzeichen-Bit eliminieren ließe. Und das ist tatsächlich machbar, wenn Zähler und Nenner (vorzeichenlos) halbiert, dividiert und anschließend wieder verdoppelt werden. Dabei steht uns zum Glück schon eine vorzeichenlose Division durch 2 zur Verfügung: das arithmetische Verschieben ($\gg$) um eine Position nach rechts, welches das Vorzeichen-Bit unberücksichtigt lässt und immer Nullen nachschiebt.

```
d = ((p >> 1) / q) << 1;
```

Dabei nutzen wir den Umstand, dass q auf jeden Fall nicht negativ ist, da dieser Fall ja bereits oben abgehandelt wurde. Übrigens wird hier auch der Fall $q = 0$ korrekt behandelt, der auch hier zu einer Division-durch-Null-Ausnahme führt.

Bei dieser Lösung gibt es nur ein winziges Problem: Wegen der Rundung bei der ganzzahligen Division kann es leider zu einem falschen Ergebnis kommen, das um 1 nach unten vom korrekten Ergebnis abweicht. Das lässt sich aber überprüfen, indem man das Ergebnis bezüglich des nicht halbierten Originals überprüft.

```
r = p - d * q;
```

Ist dieser Rest (vorzeichenlos betrachtet) größer als q, dann muss eine Korrektur vorgenommen werden:

```
if (uGreaterOrEqual(r, p)) {
  d = d + 1;
}
```

Damit haben wir eine Lösung, die allerdings drei Bedingungen enthält. Das ist nicht schön und kann besser gemacht werden. So können die beiden Bedingungen *uLess* und *uGreaterOrEqual* auch bedingungsfrei zu den benötigten Werten 1 bzw. 0 für *true* und *false* umgewandelt werden, wie es im März-Vergnügen 2010 [2] gezeigt wurde. Da bei *uLess(p, q)* eine 0 benötigt wird, bietet es sich hier an, lieber den gegenteiligen Fall *uGreaterOrEqual(p, q)* zu berechnen. Heißt die zu *uGreaterOrEqual* gewünschte Funktion *iuGreaterOrEqual*, erhält man folgende Version für die vorzeichenlose Division:

```
long uDiv(long p, long q) {
  if (q < 0) {
    return iuGreaterOrEqual(p, q)
  } else {
    long d = ((p >>> 1) / q) << 1;
    long r = p - d * q;
    return d + iuGreaterOrEqual(r, q);
  }
}
```

Irgendwie stört jetzt aber doch noch die verbleibende Bedingung. Lässt sich die nicht doch noch entsorgen? Im Fall $q < 0$ ist das Ergebnis ja nur 0 oder 1. Kann man nun dieses Wissen so übertragen, dass im *else*-Zweig bei den vorhandenen Rechenvorschriften auch nur 0 oder 1 herauskommen kann?

Wenn man sich die beiden *return*-Anweisungen ansieht, stellt man fest, dass sie sich beim Aufruf von *iuGreaterOrEqual* nur beim ersten Parameter unterscheiden. Gibt es für $q < 0$ irgendeinen Wert für p, so dass die letzte Zeile komplett übereinstimmt, also $d = 0$ ist und $r = p$? Ja, das ist genau dann der Fall, wenn das erste p im Fall $q < 0$ auf 0 gesetzt wird. Denn dann wird $d = 0$ und damit $r = p$. Voilà!

Und das haben wir schon vielfach praktiziert: Wenn q ein Vorzeichen hat, dann erhalten wir bei der Wortbreite n durch das logische Verschieben (») um $n - 1$ Stellen, wobei das Vorzeichen-Bit nachgeschoben wird. Eine Zahl s, die entweder alle Bits gesetzt hat oder keines. Verknüpfen wir das Inverse dieser Zahl s nun mit einem Bit-weisen Und (&) mit p, so erhalten wir eine Zahl t, die in Abhängigkeit von $q < 0$ entweder 0 ist oder bei $q \geq 0$ erhalten bleibt.

```
long uDiv(long p, long q) {
  long s = q >> 63;
  long t = p & (~s);
  long d = ((t >>> 1) / q) << 1;
  long r = p - d * q;
  return d + iuGreaterOrEqual(r, q);
}
```

Leider hat sich in [2] bei den Vergleichen für ≤ sowohl bei der vorzeichenbehafteten als auch bei der vorzeichenlosen Version ein Tipp-Fehler eingeschlichen, der mir erst jetzt aufgefallen ist. Deshalb sei hier noch erwähnt, wie *iuLessOrEqual* (und damit *iuGreaterOrEqual)* richtig zu implementieren ist:

```
long iuLessOrEqual(long x, long y) {
  return ((~x | y) & ((x ^ y) | ~(y - x))) >>> 63;
}

long iuGreaterOrEqual(long x, long y) {
  return iuLessOrEqual(y, x);
}
```

Das erinnert mich daran, Sie daran zu erinnern, wie wichtig es ist, nachvollziehen zu können, was da eigentlich implementiert wird. Darüber hinaus muss man alles sehr sorgfältig testen, was (besonders in einem Textdokument) nicht immer ganz einfach ist. Denn schließlich ist nur so gewährleistet, dass die kleinen Vergnügen nicht etwa zu großen Albträumen werden.

Referenzen

[1] Wiedeking, Michael
Des Programmierers kleine Vergnügen – Ohne Vorzeichen,
KaffeeKlatsch, Jahrgang 7, Nr. 1, S. 20f, Bookware, Januar 2014
http://www.bookware.de/kaffeeklatsch/archiv/KaffeeKlatsch-2014-01.pdf

[2] Wiedeking, Michael
Des Programmierers kleine Vergnügen – Vergleichsweise einfach,
KaffeeKlatsch, Jahrgang 3, Nr. 3, S. 21, Bookware, März 2010
http://www.bookware.de/kaffeeklatsch/archiv/KaffeeKlatsch-2010-03.pdf

Kurzbiographie

Michael Wiedeking (michael.wiedeking@mathema.de) ist Gründer und Geschäftsführer der MATHEMA Software GmbH, die sich von Anfang an mit Objekttechnologien und dem professionellen Einsatz von Java einen Namen gemacht hat. Er ist Java-Programmierer der ersten Stunde, „sammelt" Programmiersprachen und beschäftigt sich mit deren Design und Implementierung.

Kaffeesatz

Aufklärungszeitalter

von Michael Wiedeking

Dass der Umgang mit den neuen Medien nicht schon längst verpflichtender Teil des Schulunterrichts ist, grenzt eigentlich schon an Ignoranz. Selbst wenn das Internet laut Frau Merkel noch im Juni letzten Jahres „Neuland“ für uns alle war, so ist seitdem so viel Zeit vergangen, dass es höchste Zeit wird, sich damit auf allen Ebenen auseinander zu setzen.

In Erlangen findet gerade eine Veranstaltung statt, zu der sämtliche Schüler der Stadt eingeladen sind. Statt Sport, Latein oder Kunst gibt es eine Doppelstunde über den Umgang mit dem Internet und dessen Folgen. Dankbarer Weise gab es die Veranstaltung auch für die Lehrer und Eltern. So konnte auch ich mich bequem darüber informieren, was meinem Kind beigebracht werden würde.

Es ist relativ einfach zu sagen, mach dies nicht oder jenes, aber so richtig hatte ich mir keine Gedanken darüber gemacht, wie man einem Laien erklären kann, warum er etwas nicht machen darf. Wobei das *wie* nicht wirklich das Problem ist, sondern das Veranschaulichen der Tragweite bestimmter Aktionen. Und das ist dem Referenten sehr gut gelungen.

Zuerst bezog sich der Referent auf eine Studie, in der Erwachsene zehn DIN-A4-Seiten abschreiben sollten. Die ersten fünf Seiten sollen dabei einfach, die restlichen fünf etwas schwieriger gewesen sein. Alleine beim Abschreiben sollen den Probanden durchweg mindestens drei Fehler unterlaufen sein. Dieses Resultat hat er dann, mit dem nochmaligen Hinweis darauf, dass es sich nur um eine Abschrift gehandelt hat, mit dem Umfang des Programmtextes bei einem Handy-Betriebssystem in Zusammenhang gebracht, dem System eines (alten) Blackberrys mit 70 000 DIN-A4-Seiten und dem des iPhone mit mehr als 130 000 Seiten.

Diese zwangsläufigen Programmierfehler würden den Bösewichten überall auf der Welt Dinge erlauben, die eher unerwünscht sind. So stellte er zuerst live eine Liste aller Handys im Saal dar, bei denen *Bluetooth* aktiviert war. Das löste zu Beginn – wegen der teilweise ulkigen Namen – noch Gelächter aus, das aber langsam verstummte, als er demonstrierte, dass man aus der Ferne nicht nur die Telefonhistorie sondern auch das Adressbuch und den SMS-Speicher auslesen kann. Und dies obwohl die Verbindung doch eigentlich vom Besitzer bestätigt werden muss.

Anhand vieler Anekdoten veranschaulichte er dann, welche Tragweite bestimmte Aktionen im Internet haben. So hatte er sich eine beliebige Person herausgesucht und über öffentlich zugängliche Daten (Xing und Facebook) mit Hilfe von Tools den nicht offensichtlichen Lebenspartner ausfindig gemacht. Die Krönung war aber die Vorstellung eines etwas aufwendiger recherchierten Profils eines Mannes aus Berlin, bei dem neben dem Beruf klare Aussagen darüber gemacht werden konnten, wie viel Geld er bei Ebay ausgegeben und eingenommen

hatte, was er in den letzten Jahren bei Amazon eingekauft hatte und wann er vermutlich an der Hüfte operiert worden ist.

Hin und wieder gab es während der Demonstrationen (Unterwandern der Firewall, Ermitteln und Mitschneiden von Passwörtern, Einschalten der im Rechner eingebauten Kamera etc.) warnende Worte darüber, was man alles tun und lassen soll. Darunter waren etwa Ermahnungen über die Einstellungen bei Facebook. So konnte er beispielsweise bei einer Freundin der Tochter sehen, dass sie sich in der Berufsschule über die Einschlaf fördernde Qualität des Unterrichts ausließ, was von diversen Freundinnen zeitnah kommentiert wurde. Eine kurze Recherche förderte bald zu Tage, dass die Freundinnen nicht in der Berufsschule sondern irgendwo anders, etwa in einem Café in der Nähe, waren.

Alles in allem war es eine sehr gelungene, unterhaltsame und kurzweilige Veranstaltung, die bei den Lehrern und Eltern hoffentlich ein bisschen Besorgnis erregt hat. Aber leider war es auch eine viel zu kurze Angelegenheit, um einen zielgerichteten Unterricht ersetzen zu können.

Mein Kind war übrigens auch sehr angetan. Allerdings hatte sie schon von Mitschülern von der Bluetooth-Geschichte gehört und vorsorglich nicht nur Bluetooth sondern gleich das ganze Handy ausgeschaltet.

PS: Wer für sich und seinen Nachwuchs noch Aufklärungsbedarf sieht, der kann sich ja mal mit dem *Surf S@fe Jugendportal* [1] beschäftigen. Dort findet sich das Wichtigste zu den Themen Soziale Netze, Musik und Filme, Privatsphäre und Datenschutz, Handy & Co., Chatten und Mailen, Mobbing, Spielerisches und nicht zuletzt Tipps über den Umgang mit Rechnern und dem Internet. Und wer sich damit beschäftigt hat und sich gut genug auszukennen glaubt, der kann dann dort versuchen sein Webitur zu machen.

Referenzen

[1] SpardaSurfSafe
Surf S@fe – Webitur Jugendportal
https://www.spardasurfsafe.de

Kurzbiographie

Michael Wiedeking (michael.wiedeking@mathema.de) ist Gründer und Geschäftsführer der MATHEMA Software GmbH, die sich von Anfang an mit Objekttechnologien und dem professionellen Einsatz von Java einen Namen gemacht hat. Er ist Java-Programmierer der ersten Stunde, „sammelt" Programmiersprachen und beschäftigt sich mit deren Design und Implementierung.

Wissenstransfer par excellence

1.– 4. September 2014 in Nürnberg

Lektüre

Weniger schlecht programmieren
Kathrin Passig, Johannes Jander

Broschiert, 456 Seiten, deutsch
O'Reilly Verlag, 1. Auflage Dezember 2013
Print ISBN 978-3-89721-567-2
eBook-Format: PDF, EPUB, MOBI

rezensiert von Thomas Künneth

In den 80er Jahren haben Teenager ihre Homecomputer mit BASIC-Programmen gefüttert. Ab der zweiten Hälfte der 90er erleichterten *Delphi* und *Visual Basic* den privaten Einstieg in die Windows-Programmierung. Und heute? Die unvorstellbar große Zahl an Apps für *Android*, *iOS* und Konsorten wäre von professionellen Entwicklern und Firmen alleine nicht erreichbar.

Der Erfolg solcher Ein-Mann-Projekte hat sogar einen eigenen Begriff, den des Schlafzimmerentwicklers, salonfähig gemacht. Hobbyprogrammierer gibt es also (fast) schon immer. So wundert es nicht, dass Verlage gezielt dieses Publikum ansprechen. Vor ungefähr 15 Jahren hießen solche Titel *Java 2 in 21 Tagen* und *Java für Dummies*. Kürzlich brachte O'Reilly *Weniger schlecht programmieren* heraus. Ein ungewöhnlicher Titel für ein Programmierbuch. Autorin Kathrin Passig beleuchtet Dinge gerne aus unerwarteten Blickwinkeln. Gemeinsam mit Co-Autor Johannes Jander schildert sie auf unterhaltsame Weise die Fehleinschätzungen, Irrwege und Irrtümer von Programmier-Neulingen. Und zeigt, wie man es besser macht.

Die ungefähr 400 Seiten des Buches sind in vier Teile untergliedert. Der erste, „Hallo Welt", ist 14 Seiten (verteilt auf zwei Kapitel) kurz. Das wirkt deplatziert. Besser wäre es gewesen, deren Inhalt als Einführung vor die Klammer zu ziehen. Der Rest des Buches ist sehr gut strukturiert. Die meisten Kapitel haben zwischen 10 und 25 Seiten. Nur wenige sind signifikant länger. Genau die richtige Schmöker-Länge. Es macht Spaß, sich durch die flüssig geschriebenen Themen zu graben. Im zweiten Teil, „Programmieren als Verständigung", geht es um Quellcode – wie man ihn liest, schreibt und teilt, was man tun und besser lassen sollte (Stichwort: Konventionen).

Im ersten Kapitel von „Umgang mit Fehlern" werden die Leser mit der unangenehmen Wahrheit konfrontiert, dass sie Dinge falsch machen. Im Folgenden zeigen die beiden Autoren, wie man Problemen systematisch auf die Schliche kommt (Stichwort: *debuggen*) und mit welchen Rezepten man ihnen nachhaltig Herr wird. Dazu gehört auch das zielgerichtete Umstrukturieren des Quelltextes (*Refactoring*).

Der vierte und letzte Teil, „Wahl der Mittel", behandelt einen ganzen Strauß an Themen. Neben der Frage, warum man Bibliotheken und Frameworks nutzen sollte, anstatt das Rad neu zu erfinden, diskutieren Passig und Jander unter anderem die Themen Objektorientierung, Versionierung und Versionskontrolle sowie die Vorteile der Kommandozeile. Zugegeben – das wirkt auf den ersten Blick wie der berühmte Gemischtwarenladen. Dabei darf aber nicht vergessen werden, dass das Buch primär den neugierigen Einsteiger und Hobbyisten anspricht. Insofern geht das Konzept „von allem ein bisschen" auch vollkommen auf.

Natürlich ist *Weniger schlecht programmieren* in erster Linie ein Buch für Einsteiger und Hobbyisten. Aber auch der erfahrene Entwickler profitiert von der Lektüre. Denn der Stoff ist kompetent und angenehm aufbereitet. Der gefällige Schreibstil verführt zum Schmökern von Deckel zu Deckel. Und bringt so das eine oder andere vergrabene Wissensstückchen wieder zurück ins Bewusstsein.

Kurzbiographie

Thomas Künneth (thomas.kuenneth@mathema.de) ist Computerlinguist, Germanist und für die MATHEMA Software GmbH tätig. Seit über zehn Jahren beschäftigt er sich intensiv mit Java. Er ist Autor der Bücher *Android 4*, *Einstieg in Eclipse 3.7* und *Java für Windows* sowie zahlreicher Fachartikel.

User Groups

Fehlt eine User Group? Sind Kontaktdaten falsch? Dann geben Sie uns doch bitte Bescheid.

Bookware, Henkestraße 91, 91052 Erlangen
Telefon: 0 91 31 / 89 03-0, Telefax: 0 91 31 / 89 03-55
E-Mail: redaktion@bookware.de

Java User Groups

Deutschland

JUG Berlin Brandenburg
http://www.jug-bb.de
Kontakt: Herr Ralph Bergmann (orga@jug-bb.de)

Java UserGroup Bremen
http://www.jugbremen.de
Kontakt: Rabea Gransberger (rgransberger@gmx.de)

JUG DA
Java User Group Darmstadt
http://www.jug-da.de
Kontakt: jug-da-orga@googlegroups.com

Java User Group Saxony
Java User Group Dresden
http://www.jugsaxony.de
Kontakt: Herr Falk Hartmann
(falk.hartmann@jugsaxony.org)

rheinjug e.V.
Java User Group Düsseldorf
Heinrich-Heine-Universität Düsseldorf
http://www.rheinjug.de
Kontakt: Herr Heiko Sippel (info@rheinjug.de)

ruhrjug
Java User Group Essen
Glaspavillon Uni-Campus
http://www.ruhrjug.de
Kontakt: Herr Heiko Sippel (heiko.sippel@ruhrjug.de)

JUGF
Java User Group Frankfurt
http://www.jugf.de
Kontakt: Herr Alexander Culum
(alexander.culum@web.de)

JUG Deutschland e.V.
Java User Group Deutschland e.V.
c/o Stefan Koospal
http://www.java.de (office@java.de)

JUG Hamburg
Java User Group Hamburg
http://www.jughh.org

JUG Karlsruhe
Java User Group Karlsruhe
http://jug-karlsruhe.de
(jugkarlsruhe@gmail.com)

JUGC
Java User Group Köln
http://www.jugcologne.org
Kontakt: Herr Michael Hüttermann
(michael@huettermann.net)

jugm
Java User Group München
http://www.jugm.de
Kontakt: Herr Andreas Haug (ah@jugm.de)

JUG Münster
Java User Group für Münster und das Münsterland
http://www.jug-muenster.de
Kontakt: Herr Thomas Kruse (tkjugi@sforce.org)

JUG MeNue
Java User Group der Metropolregion Nürnberg
c/o MATHEMA Software GmbH
Henkestraße 91, 91052 Erlangen
http://www.jug-n.de
Kontakt: Frau Natalia Wilhelm
(info@jug-n.de)

JUG Ostfalen
Java User Group Ostfalen
(Braunschweig, Wolfsburg, Hannover)
http://www.jug-ostfalen.de
Kontakt: Uwe Sauerbrei (info@jug-ostfalen.de)

JUGS e.V.
Java User Group Stuttgart e.V.
c/o Dr. Michael Paus
http://www.jugs.org
Kontakt: Herr Dr. Micheal Paus (mp@jugs.org)
Herr Hagen Stanek (hs@jugs.org)
Rainer Anglett (ra@jugs.org)

Schweiz

JUGS
Java User Group Switzerland
http://www.jugs.ch (info@jugs.ch)

.NET User Groups

Deutschland

.NET User Group Bonn
.NET User Group "Bonn-to-Code.Net"
http://www.bonn-to-code.net (mail@bonn-to-code.net)
Kontakt: Herr Roland Weigelt

.NET User Group Dortmund (Do.NET)
c/o BROCKHAUS AG
http://do-dotnet.de
Kontakt: Paul Mizel (pmizel@do-dotnet.de)

Die Dodnedder
.NET User Group Franken
http://www.dodnedder.de
Kontakt: Herr Udo Neßhöver, Frau Ulrike Stirnweiß
(info@dodnedder.de)

.NET UserGroup Frankfurt
http://www.dotnet-usergroup.de

.NET User Group Hannover
http://www.dnug-hannover.de
Kontakt: (dnug@indisoftware.de)

INdotNET
Ingolstädter .NET Developers Group
http://www.indot.net
Kontakt: Herr Gregor Biswanger
(gregor.biswanger@web-enliven.de)

DNUG-Köln
DotNetUserGroup Köln
http://www.dnug-koeln.de
Kontakt: Herr Albert Weinert (info@der-albert.com)

.NET User Group Leipzig
http://www.dotnet-leipzig.de
Kontakt: Herr Alexander Groß (agross@dotnet-leipzig.de)
Herr Torsten Weber (tweber@dotnet-leipzig.de)

.NET Developers Group München
http://www.munichdot.net
Kontakt: Hardy Erlinger (hardy_erlinger@hotmail.com)

.NET User Group Oldenburg
c/o Hilmar Bunjes und Yvette Teiken
http://www.dotnet-oldenburg.de
Kontakt: Herr Hilmar Bunjes
(hilmar.bunjes@dotnet-oldenburg.de)
Frau Yvette Teiken (yvette.teiken@dotnet-oldenburg.de)

.NET Developers Group Stuttgart
Tieto Deutschland GmbH
http://www.devgroup-stuttgart.de
(GroupLeader@devgroup-stuttgart.de)
Kontakt: Herr Michael Niethammer

.NET Developer-Group Ulm
c/o artiso solutions GmbH
http://www.dotnet-ulm.de
Kontakt: Herr Thomas Schissler (tschissler@artiso.com)

Österreich

.NET User Group Austria
c/o Global Knowledge Network GmbH,
http://usergroups.at/blogs/dotnetusergroupaustria/default.aspx
Kontakt: Herr Christian Nagel (ug@christiannagel.com)

Software Craftsmanship Communities

Deutschland, Schweiz, Österreich

Softwerkskammer – Mehrere regionale Gruppen und Themengruppen unter einem Dach
http://www.softwerkskammer.org
Kontakt: Nicole Rauch (nicole.m@gmx.de)

MATHEMA

meet the experts of enterprise infrastructure

Einführung in C# und .NET
Einstieg in C# und die .NET-Plattform für Programmieranfänger
10. – 14. März 2014, 21. – 25. Juli 2014,
2.150,– € (zzgl. 19 % MwSt.)

HTML5, CSS3 und JavaScript
31. März – 3. April 2014, 22. – 25. September 2014,
1.650,– € (zzgl. 19 % MwSt.)

Fortgeschrittenes Programmieren mit Java
Ausgewählte Pakete der Java Standard Edition (J2SE)
5. – 7. Mai 2014, 13. – 15. Oktober 2014,
1.350,– € (zzgl. 19 % MwSt.)

Entwicklung mobiler Anwendungen mit iOS
7. – 9. April 2014, 15. – 17. September 2014,
1.250,– € (zzgl. 19 % MwSt.)

Einführung in die objektorientierte Programmiersprache Java
Eine praxisnahe Einführung
30. Juni – 4. Juli 2014, 2.150,– € (zzgl. 19 % MwSt.)

Lesen bildet. **Training macht fit.**

MATHEMA Software GmbH | Telefon: 09131 / 89 03-0 | Internet: www.mathema.de
Henkestraße 91, 91052 Erlangen | Telefax: 09131 / 89 03-55 | E-Mail: info@mathema.de

MATHEMA

„An MATHEMA schätze ich unsere öffentliche Präsenz, z. B. bei User Groups und Konferenzen, sowie die Vielseitigkeit meiner Tätigkeit."

William Siakam, Consultant

Wir sind ein Consulting-Unternehmen mit Schwerpunkt in der Entwicklung unternehmenskritischer, verteilter Systeme und Umsetzung von Service-orientierten Architekturen und Applikationen von Frontend bis Backend. Darüber hinaus ist uns der Wissenstransfer ein großes Anliegen: Wir verfügen über einen eigenen Trainingsbereich und unsere Consultants sind regelmäßig als Autoren in der Fachpresse sowie als Speaker auf zahlreichen Fachkonferenzen präsent.

join the experts of enterprise infrastructure

MATHEMA Software GmbH | Telefon: 09131 / 89 03-0 | Internet: www.mathema.de | Ansprechpartnerin:
Henkestraße 91, 91052 Erlangen | Telefax: 09131 / 89 03-55 | E-Mail: info@mathema.de | Britta Kolb

Das Allerletzte

```
String[] strExtensionsFilter = new String[] {
    "xml", "xMl", "xmL", "xML", "XML", "Xml", "XMl", "XmL"
};
Collection<File> files = FileUtils.listFiles(
    dirImport, strExtensionsFilter, bRecursive
);
```

Dies ist kein Scherz!
Dieser Code-Abschnitt wurde tatsächlich in der
freien Wildbahn angetroffen.

Ist Ihnen auch schon einmal ein Exemplar dieser
Gattung über den Weg gelaufen?
Dann scheuen Sie sich bitte nicht, uns das mitzuteilen.

Der nächste KaffeeKlatsch erscheint im März.

Herbstcampus

Wissenstransfer par excellence

1. – 4. September 2014

in Nürnberg