Das iJUG Magazin

Java aktuell

Sommer 2011

Java aktuell

Magazin der Java-Community

- Neu: GlassFish 3.1
- Single Sourcing in Java
- Nützliche Java-Tools
- Entwickeln mit Scala

Kommt es zum Bruch?

Community

ORACLE

Erfahrungen, Ideen und Lösungen für Java-Entwickler

iJUG Verbund

www.ijug.eu D: 4,90 EUR A: 5,60 EUR CH: 9,80 CHF Benelux: 5,80 EUR ISSN 2191-6977 03/2011

Wolfgang Taschner
Chefredakteur Java aktuell

Mehr aufeinander zugehen!

Vor einiger Zeit sah ich wieder einmal einen meiner Lieblingsfilme im Fernsehen: Alexis Sorbas. Zwei Männer – wie sie unterschiedlicher nicht sein könnten – treffen aufeinander, arbeiten miteinander, lernen im Lauf der Ereignisse viel voneinander und finden schließlich zueinander. Unvergesslich die Szene, wie die beiden am Ende zusammen im Sand den Sirtaki tanzen.

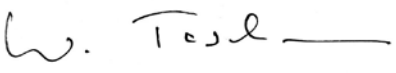
Ähnliches würde ich mir auch von Oracle und der Java-Community wünschen. Momentan steht auf der einen Seite die Community mit vielen kreativen Ideen – auf der anderen Seite das erfolgreiche Unternehmen Oracle. Nach dem ersten Beschnuppern ist die gegenseitige Anerkennung sehr gering. Teile der Community möchten am liebsten ein neues Java auf den Weg bringen, Oracle hält sich mit seinen Plänen bedeckt. Die Signale auf beiden Seiten stehen auf Abwehr.

Die Aussichten auf Erfolg bei einem neuen Java-Fork scheinen gering. Die Community würde sich aufsplitten, außerdem sind viele rechtliche Fragen ungeklärt, die das Projekt auf Jahre belasten könnten.

Oracle hat die JavaOne, einstige Pilgerstätte der Java-Community, zum Anhängsel der OpenWorld degradiert. Dort, wo früher Java-Entwickler aus der ganzen Welt ihre Ideen und Visionen ausgetauscht haben, findet man heute nur noch genervte Teilnehmer auf der Suche nach ihren weit auseinanderliegenden und schwer auffindbaren Vortragsräumen. Oracle trug auf der letzten JavaOne gebetsmühlenartig seit Langem bekannte Roadmaps vor, ohne ins Detail zu gehen. Wo waren die Oracle-Vertreter? Die eigens für sie reservierten Plätze in den vordersten Reihen blieben während der Keynote leer.

Wenn es so weitergeht, stehen am Ende beide Seiten mit leeren Händen da: Dem Java-Fork fehlt die Unterstützung durch die Industrie, Oracle bekommt keine kreativen Ideen aus der Community.

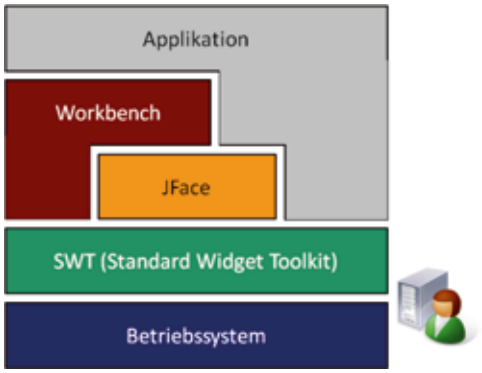
Der Weg zur Annäherung führt über gegenseitiges Verständnis: Ein erfolgreiches Unternehmen wie Oracle setzt Geschäftsideen um, sonst droht das gleiche Schicksal wie bei Sun. Eine kreative Community braucht eine Spielwiese wie die JavaOne, sonst verkümmert sie. Die Symbiose wäre ein Glücksfall für die Zukunft von Java.



Inhalt

„Any place – any time“: Java spielt bei deren Entwicklung eine tragende Rolle, Seite 12

GlassFish 3.1: Oracle präsentiert den ersten hochverfügbaren Java-EE-6-Server, Seite 15



Single Sourcing in Java: Desktop-Anwendung und Web-Applikation aus einer Quelle, Seite 28

„Die langfristige Java-Strategie von Oracle ist mir nicht transparent genug ...“

Usergroups bieten vielfältige Möglichkeiten zum Erfahrungsaustausch und zur Wissensvermittlung unter den Java-Entwicklern. Sie sind aber auch ein wichtiges Sprachrohr in der Community und gegenüber Oracle. Wolfgang Taschner, Chefredakteur von Java aktuell, sprach darüber mit Dr. Michael Paus, dem Vorsitzenden der Java User Group Stuttgart.

Wie bist du zur Java User Group Stuttgart gekommen?

Dr. Michael Paus: Ich bin bereits Ende der 1990er Jahre auf die User Group aufmerksam geworden. Zuvor war ich aus beruflichen Gründen Mitglied in einer Next-Step-User-Group und habe nach der Übernahme von Next Step durch Apple ein neues Betätigungsfeld gesucht. Wenn mir damals jemand gesagt hätte, wo Java heute stehen würde, hätte ich das nicht geglaubt.

Zur Person: Dr. Michael Paus

Dr. Michael Paus ist promovierter Luft- und Raumfahrtingenieur mit dem Schwerpunkt Flugführungssysteme. In diesem Bereich ist er auch seit vielen Jahren als selbständiger Berater, Architekt und Softwareentwickler für namhafte Firmen im Luftfahrtsektor tätig. Sein Tätigkeitsbereich umfasst dabei sowohl die Entwicklung von Embedded-Systemen in Ada als auch von Desktop-Systemen, z.B. für die Missionsplanung, in Java. Neben seiner beruflichen Tätigkeit ist er seit vielen Jahren Vorsitzender der Java User Group Stuttgart.

Wie hat sich deine Aktivität bei der Java User Group Stuttgart entwickelt?

Dr. Michael Paus: Zu Beginn war ich ganz normales Mitglied und habe die Veranstaltungen besucht. Im Laufe der Zeit half ich bei der Organisation mit und bin dann im September 2002 in den Vorstand gewählt worden.

Wie ist die Java User Group Stuttgart organisiert?

Dr. Michael Paus: Neben dem Vorstand gibt es noch ein umfangreiches Board mit rund zehn Aktiven, die alle gemeinsam die Vereinsarbeit und die Veranstaltungen organisieren. Die Arbeit ist ehrenamtlich und erfolgt bei allen Aktiven neben dem eigentlichen Beruf. Die Java User Group Stuttgart zählt heute rund 400 Mitglieder. Wichtigstes Event ist das Java Forum mit knapp 1.200 Teilnehmern.

Was zeichnet die Java User Group Stuttgart aus?

Dr. Michael Paus: Zum einen haben wir über die letzten 15 Jahre eine große Kontinuität – das Java Forum findet demnächst zum 14. Mal statt. Zum anderen sind auch unsere Abendveranstaltungen für ihre hohe Qualität bekannt. Es gibt wenige Anwendergruppen im Java-Umfeld, die so lange konstant aktiv sind und auch noch weiter wachsen.

Wie viele Abendveranstaltungen gibt es pro Jahr?

Dr. Michael Paus: Wir organisieren diese immer dann, wenn ein interessantes Thema anliegt. In der Regel kommen wir jährlich auf etwa 15 Veranstaltungen, die jeweils von bis zu 100 Teilnehmern besucht werden.

Was motiviert dich besonders, als Vorstand die Java User Group Stuttgart zu führen?

Dr. Michael Paus: Diese Frage stelle ich mir auch immer wieder. Auf der einen Seite ist es auch nach all den vielen Jahren meine Begeisterung für Java. Eine Usergroup ist natürlich die ideale Stelle, um sich über alle Weiterentwicklungen auf dem Laufenden zu halten. Insofern bin ich ein wenig

stolz darauf, als Vorstand die Geschicke des Vereins beeinflussen zu können. Darüber hinaus bin ich in der glücklichen Lage, dass mich ein großes Team an Aktiven in der Vereinsarbeit unterstützt.

Was bedeutet Java für dich?

Dr. Michael Paus: Für mich steht weniger die Sprache im Vordergrund, mich begeistert vielmehr die Java-Plattform. Meine frühere Arbeitswelt war geprägt von den unterschiedlichsten Infrastrukturen und Sprachen. Es existierte keine Vereinheitlichung. Java hingegen bietet mir eine einheitliche Plattform, auf der alle wesentlichen Teile vorhanden sind.

Welchen Stellenwert besitzt die Java-Community für dich?

Dr. Michael Paus: Die Community ist sehr wichtig, um diesen Plattform-Gedanken weiterzutragen und auszubauen. Von daher fände ich es nicht gut, wenn die Java-Entwicklung in der Hand einer einzigen Person, Firma oder Institution läge. Die Java-Plattform lebt für mich von der Vielfalt, beispielsweise wenn ich an die alternativen Sprachen denke, die sich in diesem Umfeld entwickeln.

Was hast du bei der Übernahme von Sun durch Oracle empfunden?

Dr. Michael Paus: Es hatte sich ja schon zuvor über einen längeren Zeitraum angedeutet, dass mit Sun etwas passieren würde. Heute sehe ich die Sache mit einem lachenden und einem weinenden Auge. Ich finde es gut, dass Oracle ganz offensichtlich wesentlich zielgerichteter vorgeht und viele Dinge, die bei Sun eine Art Dornröschenschlaf hatten, jetzt entweder abschafft oder engagiert weitertreibt. Weniger gut gefällt mir die Kommunikation beziehungsweise Zusammenarbeit von Oracle mit der Community. Hier bestehen noch erhebliche Informationslücken. Insbesondere die langfristige Java-Strategie von Oracle ist mir nicht transparent genug.

Wie sollte sich Java weiterentwickeln?

Dr. Michael Paus: Ich hoffe, dass Java noch eine große Zukunft vor sich hat. Das hängt sicherlich von den Entwicklungen der nächsten Jahre ab. Das bedeutet nicht, dass Java einfach vom Markt verschwinden könnte, aber die Weichen müssen richtig gestellt sein. Dabei spielt auch das Verhältnis zwischen Google und Oracle in Bezug auf Android eine Rolle. Ich persönlich habe keine Berührungsängste mit Android, auch wenn es kein reines Java ist. Hier würde ich eine weitere Harmonisierung begrüßen.

Wo siehst du die Vorteile von Android?

Dr. Michael Paus: Das klassische Java ist zwar eine Integrationsplattform, aber immer als Aufsatz auf einem anderen Betriebssystem. Android hingegen stellt eine geschlossene Umgebung dar, die von der Architektur her wesentlich moderner und umfassender ist. Von daher ist Android für mich vielleicht das „bessere" Java.

Wie sollte Oracle deiner Meinung nach mit Java umgehen?

Dr. Michael Paus: Ich denke, dass zur Weiterentwicklung von Java ein starker Unternehmenspartner notwendig ist, der für Kontinuität und ausreichend Ressourcen sorgt. Dazu wäre Oracle in der richtigen Position. Aber Oracle müsste sich wesentlich mehr gegenüber der Community öffnen. Außerdem sollte Oracle auch mit anderen Partnern wie beispielsweise Google mehr ins Gespräch kommen, um eine Harmonisierung zwischen dem Standard-Java und Android herbeizuführen. Momentan erleben wir eine große Vielfalt an Systemen, vergleichbar mit der Situation damals bei den Betriebssystemen für PCs. Ich hoffe, dass bei dem System, das sich herauskristallisieren wird, Java eine führende Rolle spielt.

Wie sollte sich die Community gegenüber Oracle verhalten?

Dr. Michael Paus: Die Antwort ist nicht ganz einfach, weil man es hier mit den unterschiedlichsten Interessenlagen zu tun hat. Wichtig ist, den Einsatz von Java auf den unterschiedlichsten Plattformen zu kommunizieren, denn die Weiterentwicklung von Java ist nur gemeinsam möglich. Die Bestrebungen, einen zweiten Fork von Java zu eröffnen, wäre für mich das letzte Mittel, wenn mit Oracle absolut keine Kommunikation mehr möglich ist. Vor einer solchen Entscheidung sollte die Community noch etwas Geduld mit Oracle haben aber gleichzeitig den Druck erhöhen, um konkrete Entscheidungen zu erreichen und somit für einen Fortschritt zu sorgen.

Wie schätzt du Java FX ein?

Dr. Michael Paus: Ich halte Java FX für einen guten Ansatz. Aber auch hier ist mir nicht ganz klar, wohin die Reise geht. Die Konkurrenz auf diesem Gebiet ist sehr groß und Oracle ist relativ spät dran. Da ich die Technologie für sehr interessant halte, wünsche ich mir, dass sich möglichst bald etwas bewegt. Eine Portierung auf die Android-Plattform würde es ermöglichen, gleichzeitig für den klassischen PC und für Android-Systems Software entwickeln zu können.

Sollte sich ein junger Neueinsteiger noch mit Java beschäftigen?

Dr. Michael Paus: In jedem Fall! Java ist eine gute Basis für die nächsten Jahre.

Kontakt:
Dr. Michael Paus
mp@jugs.org

Das Java-Tagebuch

Andreas Badelt, Leiter SIG Java, DOAG Deutsche ORACLE-Anwendergruppe e.V.

In der letzten Ausgabe der Java aktuell wurde das Java-Tagebuch gestartet, um in einer turbulenten Zeit einen Überblick über die wichtigsten Geschehnisse rund um Java zu geben – in komprimierter Form und chronologisch geordnet. Wichtige Releases und Presse-Erklärungen, Ereignisse im Java Community Process (JCP) und, soweit möglich, Diskussionen in den User Groups und in der nicht organisierten Community werden hier betrachtet. Der erste Teil des Tagebuchs endete im Dezember 2010, der vorliegende zweite Teil knüpft direkt daran an.

11. Januar 2011

Streit um Hudson

Der Streit zwischen Oracle und einem großen Teil der Hudson-„Entwicklergemeinde" eskaliert. Die Diskussion um die optimale Infrastruktur für Quellcode, Mailing-Listen etc. des Build-Servers „Hudson" hat eine lange Vorgeschichte: Bedingt durch häufige Ausfälle der java.net-Systeme wurde immer wieder der zumindest teilweise Umzug zu anderen Plattformen diskutiert. Nach erneuten Ausfällen, insbesondere auch im Zusammenhang mit missglückter Kommunikation rund um die Migration von java.net auf neue Infrastruktur (Kenai) durch Oracle, will die Community nun endgültig handeln. Hudson-Erfinder Kawaguchi schlägt vor, das Projekt auf GitHub zu migrieren. Ted Farrell, Oracle Senior Vice President Tools and Middleware, versucht sie davon zu überzeugen, dass Hudson unbedingt auf der java.net-Plattform bleiben soll. Oracle könne einen Fork nicht verhindern, werde Hudson aber auf java.net weiterentwickeln. Dabei erwähnt er auch, dass die Markenrechte für Hudson bei Oracle liegen, das heißt, der Fork müsste unter einem anderen Namen weiterlaufen. Das kommt bei der Community gar nicht gut an – zumal sich wenige Tage später herausstellt, dass Markenrechte für Hudson kurz vorher überhaupt erst beantragt wurden.

Nach langen Verhandlungen mit Oracle schlägt Lead-Entwickler Andrew Bayer nun vor, Hudson in „Jenkins" umzubenennen, verbunden mit der Migration auf eine von Oracle unabhängige Infrastruktur und der Einrichtung eines Interims-Governance-Boards. Hauptgrund ist, dass Oracle keine umfassenden Garantien geben möchte, von den Markenrechten niemals Gebrauch zu machen beziehungsweise sie an eine unabhängige Instanz abzutreten. Von einem Fork möchte Bayer aber nicht sprechen, eher von einer Umbenennung, da Hudson-Erfinder Kohsuke Kawaguchi mitgeht. Klingt gut, bleibt aber eine semantische Finte, auch wenn offenbar der Kern der Community geschlossen dahintersteht. Über den Vorschlag soll nun die Community abstimmen.

13. Januar 2011

Visage („the language formerly known as JavaFX Script") auf Android

2010 hat Steven Chin das Visage-Projekt gegründet, um JavaFX Script nach Oracles Strategiewechsel hin zu einer reinen Java API weiterzuentwickeln. Jetzt berichtet er in seinem Blog [http://javafx.steveonjava.com], dass er daran arbeitet, Visage-Applikationen auf Android laufen zu lassen, und zeigt ein erstes, gelungenes Beispiel. Was für Visage möglich ist, sollte auch für JavaFX selbst möglich sein. Ein Hoffnungsschimmer für geplagte Java-Mobile-Entwickler?

Java SE 7 ist „Feature Complete"

Ganz knapp hinter dem Zeitplan, aber nach dem jahrelangen Warten auf den Nachfolger für Version 6 kommt es darauf auch nicht mehr an: Oracles Java-Produktstratege Henrik Ståhl verkündet in seinem Blog den Status „Feature Complete". Damit sollte auch die Developer Preview bald verfügbar sein. Kleine Randnotiz: In einem Blog-Kommentar nach der Lauffähigkeit von SE 7 auf Android gefragt, schreibt Ståhl, dass dies aktuell nicht geplant sei.

24. Januar 2011

Brasilianische User Group SouJava für JCP nominiert

Oracle hat die brasilianische User Group SouJava, vertreten durch ihren ehemaligen Vorsitzenden Bruno Souza, für einen der vakanten Sitze im Java SE/EE Executive Committee des JCP nominiert. Ende 2010 war Oracle mit dem ersten Vorschlag, der weitgehend unbekannten Partnerfirma Holigic, auf viel Skepsis gestoßen und letztlich gescheitert. Generell wird ja argwöhnisch beobachtet, wie Oracle weiter mit dem JCP verfahren wird. Aber eine User Group beziehungsweise ihren Vertreter einzubinden, ist definitiv ein gutes Signal. Die Entscheidung wird in der Community auch entsprechend positiv aufgenommen – einer erfolgreichen Wahl wird hier wohl nichts im Wege stehen.

28. Januar 2011

Das OpenJDK soll eine Satzung erhalten

Mark Reinhold, Java-Chef-Architekt bei Oracle, schreibt in seinem Blog, dass er bereits seit November 2010 in einem über-

greifenden Team an einer Satzung für das OpenJDK-Projekt arbeitet.

Mit dabei sind Vertreter von IBM und der Eclipse Foundation und – Was für eine Überraschung: Doug Lea. Er war erst im Oktober im Streit um die Zukunft des Java Community Process nicht mehr zur Wiederwahl in dessen Executive Committee angetreten. In einer Erklärung dazu hatte er geschrieben, dass der JCP für ihn keine Zukunft habe, dass das OpenJDK aber zumindest auf den zweiten Blick eine Alternative sei: „At this point, a Linux-style model for collaboratively developed common source is likely to be more effective in meeting upcoming challenges than is the JCP." Auf den zweiten Blick ist seine Beteiligung also gar nicht mehr so überraschend.

Grundlage der Satzung (Community Bylaws) sind laut Reinhold die Entwürfe des „OpenJDK Interim Governance Board", das sich in den Jahren 2007 bis 2009 unter Sun bereits einmal an dieser Aufgabe versucht hatte.

Ein erster Entwurf soll bald veröffentlicht werden. Die schriftlichen Regeln sollen, so Reinhold, das langfristige Wohlergehen und Wachstum der OpenJDK Community sicherstellen und seine Mitglieder befähigen und ermutigen, offen, transparent und leistungsorientiert („meritocratic") zu agieren. Klingt gut und nach dem Gegenteil dessen, was Oracle vielfach vorgeworfen wird, wie etwa in der Hudson/Jenkins-Diskussion.

Die Reaktionen in der Community sind teilweise positiv, es werden aber auch viele kritische Fragen gestellt. Eine der deutlichsten: Wenn alles offen und transparent sein soll, warum wird dann erst mal mehrere Monate in einer kleinen Gruppe gearbeitet, ohne die Community einzubinden oder auch nur zu informieren?

29. Januar 2011

Die Entscheidung ist gefallen: Hudson wird Jenkins

Die Hudson-Gemeinde hat sich mit überwältigender Mehrheit (214 zu 14 Stimmen) für die Umbenennung von Hudson zu Jenkins entschieden. Damit ist der Fork perfekt.

30. Januar/1. Februar 2011

Oracle will Hudson vorantreiben

Oracle verkündet, dass es Hudson mit einem dedizierten Entwicklerteam weiter vorantreiben wird. Unterstützung bekommt es dabei von Sonatype, einem Unternehmen, das schon einiges in die Entwicklung des CI-Servers investiert hat und dessen Gründer van Zyl in die Diskussionen um den drohenden Fork involviert war. Van Zyls Standpunkt war schon vorher recht klar, jetzt bekennt Sonatype sich offiziell zur Hudson-Variante. Diese wird auf java.net zurückziehen – wobei Oracle-Mitarbeiterin Susan Duncan betont, dass dies nicht endgültig sein muss („We can also discuss moving this back to github once we get things settled").

31. Januar 2011

Java-DoS-Bug sorgt für Hektik bei Java-Anwendern

Anfang Januar hat Rick Regan im Blog „exploringbinary.com" über einen Fehler in PHP berichtet, der bei der Konvertierung des Strings „2.2250738585072012e-308" in ein Double die Anwendung in eine Endlosschleife schickt. Am 31. Januar 2011 ergänzt Konstantin Preißer in einem Kommentar dazu, dass dies auch für die Java Virtual Machine gilt. Den Defekt habe er bereits drei Wochen vorher an Oracle gemeldet – aber bislang nur eine Eingangsbestätigung erhalten.

Die Nachricht verbreitet sich in Windeseile und es entwickelt sich eine lebhafte Diskussion im Blog. Schon einen Tag später ist klar: Erstens ist das Problem seit 2001 bekannt und wurde von Sun als „low priority" eingestuft – daher nach fast zehn Jahren noch kein Fix! Zweitens ist das Bedrohungspotenzial, insbesondere für Web-Applikationen, alles andere als „low".

Selbst Applikationen, die keine Nutzer-Eingaben akzeptieren und in Doubles umwandeln, können gefährdet sein: HTTP sieht bei einigen Header Values die Möglichkeit der Priorisierung in einer Liste anhand eines „quality factors" vor (zum Beispiel „Accept-Charset: iso-8859-5, unicode-1-1; q=0.8"). Bei einigen Servlet-Containern reicht es, hier die besagte Zahl zu verwenden, um einen Thread „abzuschießen". Das ist wohl der Moment, in dem viele Administratoren nervös werden.

2. Februar 2011

And the winner is … Java Desktop!!

In einer aktuellen Umfrage auf java.net votierte eine große Zahl der Teilnehmer für Verbesserungen im Bereich Desktop. Die Umfrage lautete: "Which area of Java/JVM technology most desperately needs serious attention in 2011?" Hier die Top 3:

- Java on the desktop: 31 Prozent
- JDK (Java 7, Java 8): 25 Prozent
- Java on mobile platforms: 16 Prozent

Java EE war als Vierter mit 7 Prozent schon weit abgeschlagen. Über die Gründe wird nun natürlich allerseits spekuliert. Sind es ganz einfach die größeren „Schmerzen", die die Entwickler aus diesem Bereich an die Wahlurne getrieben haben?

Oder ist Java auf dem Desktop (und auch Java für mobile Geräte) doch nicht so mausetot, wie z.B. von John Rhymer (http://blogs.forrester.com/john_r_rymer/11-01-23-the_future_of_java) behauptet wird?

Wenn die Umfrage auch nur halbwegs repräsentativ ist, ist das eine Steilvorlage für Oracle und JavaFX! Nutzt sie! Und vergesst bitte auch nicht „mobile" mit immerhin 16 Prozent – auch dafür war JavaFX ja mal vorgesehen. Und noch ein Wunsch hinterher: Die Sache mit der JavaFX-Runtime-Lizenz sollte auch noch geregelt werden. Gibt es dafür eigentlich schon ein Twitter hashtag? #ceterumCenseo oder so?

3. Februar 2011

Draft der OpenJDK-Satzung veröffentlicht

Mark Reinhold veröffentlicht einen Entwurf der OpenJDK-Satzung unter http://cr.openjdk.java.net/~mr/bylaws/draft-openjdk-bylaws-07.html. Gleichzeitig gibt er in seinem Blog die Zusammensetzung des „Initial Governing Board" bekannt

(wohlgemerkt: Initial, nicht Interim). Dieses besteht im Prinzip aus den Mitgliedern des Teams, das an dem Entwurf gearbeitet hat.

Das Governing Board wird nach der Satzung aus einem Vorsitzenden, einem Stellvertreter sowie einem „OpenJDK Lead" bestehen. Der Stellvertreter wird von IBM ernannt, die beiden anderen von Oracle. Daneben sind zwei freie Mitglieder („at-large members") vorgesehen, die von den OpenJDK-Mitgliedern für jeweils ein Jahr gewählt werden.

Vier Wochen lang soll der Entwurf nun über eine OpenJDK-Mailingliste diskutiert und überarbeitet werden, danach soll eine Abstimmung erfolgen. Die ersten kritischen Fragen lassen nicht lange auf sich warten. Unter anderem geht es um rechtliche Fragen: Ein „Contributor" muss einen Vertrag mit Oracle abschließen, das eigentlich ja nur eines der Mitglieder ist, wenn auch ein „primus inter pares". Auch die Zusammensetzung des Boards wird kritisch gesehen, zum einen die Übermacht von Oracle und IBM, zum anderen das Fehlen von RedHat, das im Gegensatz zu IBM viel in das OpenJDK investiert hat.

8. Februar 2011

Java-DoS Bug – die Erlösung

Nachdem der DoS-Bug seit der „Wiederentdeckung" am 31. Januar 2011 viel Aufmerksamkeit erhalten hat, geht es recht schnell: Am 8. Februar 2011 veröffentlicht Oracle ein „Java Critical Patch Update", das den Fehler behebt. Zumindest vereinzelte Attacken auf Web-Applikationen werden bereits bekannt. Bevor die Lage schlimmer wird, werden wohl an den darauffolgenden Tagen viele Betriebsabteilungen damit beschäftigt sein, das Patch auszurollen.

Das Rennen gegen die PHP-Entwickler hat Oracle zwar verloren – dort wurde von der Existenz des Bug am 30. Dezember 2010 berichtet und am 3. Januar 2011 der Fix eingespielt. Für einen großen Konzern wie Oracle ist die Reaktionsgeschwindigkeit jedoch sicher nicht schlecht, wenn man die zehn Jahre nicht mitrechnet, die der Bug-Report erst bei Sun, dann bei Oracle im Keller schlummerte.

14. Februar 2011

Hudson zieht auf GitHub um, wird geforkt, zieht auf java.net um, zieht auf GitHub um

Der Titel mag ein wenig verwirrend klingen, aber genau so ist es. Was Susan Duncan direkt nach dem Fork als Option in Aussicht gestellt hatte, wird nun von den verbleibenden Hudson-Entwicklern beschlossen. Das Projekt wird wieder auf GitHub umziehen. Auf java.net verbleibt ein Mirror. Das hat auf den ersten Blick etwas Ironisches, da der Streit, der zum Fork führte, sich an der Infrastruktur entzündet hatte. Aber in diesem Punkt wäre man sich wohl einig geworden – am Ende waren es die bei Oracle liegenden Markenrechte, die einem großen Teil der Community nicht behagten.

22. Februar 2011

Java SE 7 Developer Preview verfügbar

Jetzt ist auch die Developer Preview für die Java SE 7, die im Sommer herausgebracht werden soll, veröffentlicht worden. Damit ist nun die Community aufgefordert, das neue Java intensiv zu testen und Feedback zu geben.

23. Februar 2011

NetBeans 7.0 Beta 2 ist da!

Und nun ist auch NetBeans 7, Beta 2 verfügbar. Nichts spektakulär Neues im Vergleich zur Beta 1, aber Unterstützung für die Versionskontrolle Git (Preview) und viele Performance- und Qualitätsverbesserungen.

28. Februar 2011

GlassFish 3.1 ist freigegeben – Hochverfügbarkeit inklusive

Die neue Version 3.1 des Open Source GlassFish Application Servers ist freigegeben. Insbesondere für Planer und Betreiber größerer Applikationen werden viele Wünsche erfüllt. Unter anderem arbeiten die Server im Cluster (müsste es hier nicht eigentlich „Schwarm" heißen?), sind zentral – auch über SSH – administrierbar und bieten eine Menge von Features für Failover, Hochverfügbarkeit und Load Balancing. Damit hat Oracle das im Frühjahr 2010 in der Roadmap gegebene Versprechen eingelöst, die Hochverfügbarkeits-Features nachzuziehen, die in GlassFish 2 noch vorhanden waren, in 3.0 aber fehlten.

java.net-Migration auf neue Infrastruktur abgeschlossen

Seit mehreren Monaten wurden die Seite java.net und die unzähligen enthaltenen Projekte von CollabNet auf die Kenai-Infrastruktur migriert. Dabei wurde wohl auch alter Ballast entfernt (Projekte mussten sich aktiv für die Migration anmelden). Jetzt ist die Migration komplett, CollabNet ist „read-only" und wird wohl bald abgeschaltet.

8. März 2011

JSR für Java EE 7 registriert

Java EE 6 kommt gerade richtig in Schwung, jetzt wird der Specification Request für den Nachfolger beim JCP registriert. Damit wird die Lücke zwischen Version 6 und 7 bei Java EE (hoffentlich) nicht so groß wie bei Java SE. Das Leitmotiv für Release 7 wird gleich zu Beginn des Requests klargestellt: „The main theme for this release is the Cloud."

Bestandteil des Schirm- oder besser Mantel-JSRs („Umbrella" JSR 342) sollen mehrere aktualisierte APIs sein: Servlets 3.1, Expression Language 3.0, JMS 2.0 und JSF 2.2; die jeweiligen JSRs wurden gleichzeitig zum Umbrella registriert (JSRs 340, 341, 343 und 344). Außerdem sollen die JSRs „Concurrency Utilities for Java EE" und „JCache" neu aufgenommen werden. Beide sind noch in ihrer Anfangsphase und seit Jahren inaktiv – sie müssten erst einmal wiederbelebt werden. Zwei weitere APIs werden darüber hinaus als optionale Kandidaten genannt: Java Web Sockets API und Java JSON API.

Jetzt wird erst einmal bis zum 14. März 2011 innerhalb des Executive Committee diskutiert und abgestimmt, sowie Bewerbungen für das Spezifikations-Team von Mitgliedern des JCP entgegengenommen (das sind noch nicht einmal 1500 Perso-

nen, Firmen und Institute – obwohl die Mitgliedschaft prinzipiell jedem offen steht und für persönliche Mitglieder auch nichts kostet).

14. März 2011

Java EE 7 bekommt 13 Mal ein „Ja"

Der „JSR Review Ballot" für Java EE 7 ist abgeschlossen, alle 13 Mitglieder des Executive Committes haben mit „Ja" gestimmt – sowohl für den „Umbrella"-JSR als auch für die anderen neuen Einzel-Spezifikationen (JMS, JSF etc.). Specification Leads für den Mantel-JSR sind Roberto Chinnici und Bill Shannon von Oracle. Daneben sitzen in der Expert Group mit Adam Bien und Werner Keil zwei deutsche Java-Kapazitäten. Die Expert Groups für die anderen JSRs sind noch dünn besetzt und können sicher ein paar Freiwillige vertragen. Wir wünschen viel Erfolg!

28. März 2011

James Gosling geht zu Google

Java-Erfinder und Sun-Urgestein James Gosling fängt nach einem Jahr Schaffenspause heute bei Google an, verkündet ein Eintrag in seinem Blog. Da er nach der schnellen Trennung von Oracle im Rahmen der Sun-Übernahme und dem einige Zeit später begonnenen Android-Streit bereits Position für Google bezogen hatte, ist der Schritt vielleicht nicht mehr ganz so überraschend. Aber nach eigener Auskunft weiß er selbst noch nicht genau, was seine Aufgabe sein wird. Hoffen wir, dass er dort sein schöpferisches Potenzial zugunsten von Java ausleben kann und sich nicht doch wieder mit juristischen Planspielen auseinandersetzen muss.

Kontakt:
Andreas Badelt
Andreas.badelt@doag.org

Andreas Badelt ist Berater und Softwareentwickler / -Architekt bei der CGI Information Systems and Management Consultants (Deutschland) GmbH. Sein Schwerpunkt ist die Realisierung von großen eCommerce-Systemen, insbesondere für die Telekommunikationsbranche. Daneben organisiert er seit 2001 die Special Interest Group (SIG) Development bzw. SIG Java der DOAG.

Die Göttinger Source Talk Tage von 30. August bis 1. September 2011 sind seit Jahren der Technologie-Kongress im IT-Bereich mitten in Deutschland. Die Teilnehmer erwartet eine dreitägige Mischung aus intereressanten, einmaligen Spezialtagungen sowie Tracks und Trainings zu Basis-Technologien wie Java, Web-Anwendungen, Datenbanken und Betriebssystemen.

Source Talk Tage

Das Programm 2011

- Strömungen berechnen – OpenFOAM voranbringen,
  Dr. Gerd Rapin, Volkswagen
  OpenFOAM (Open Field Operation and Manipulation) ist ein in C++ geschriebenes numerisches freies Simulationsoftware-Paket für kontinuumsmechanische Probleme. Das Hauptaugenmerk liegt dabei auf dem Lösen von Strömungsproblemen. Im Rahmen des Source Talk findet ein Arbeitsgruppentreffen statt.
- Mathematik erlernen – Mathematik & eLearning umsetzen,
  Prof. Dr. Christian von Bothmer, Mathematisches Institut Göttingen
  Hier kommen die Aktiven an der Schnittstelle zwischen Technik und Didaktik zusammen, um das Thema „eLearning" in der Mathematik voranzubringen.
- Systeme verwalten – Netzwerke beherrschen,
  Dr. Georg Pagendarm, DNW
  Die moderne Systemwelt ist heterogen und vernetzt. Welche Tools helfen, die Übersicht zu behalten und Netzwerke zu beherrschen?
- Java trinken – objektorientiert programmieren,
  Frank Schwichtenberg, FIZ Karlsruhe
  Java ist aus der modernen IT-Welt nicht wegzudenken. Neben der Sprache an sich hat sich die Plattform in vielen Einsatzgebieten bewährt. Entwickler, die den Umgang mit Java als Sprache und als Plattform beherrschen sind gesucht. Profis lassen sich hier in die Karten schauen.
- Auftritt verbessern – Webserver optimieren,
  Dr. Robert Zorres, wer-kennt-wen.de
  Webtechnologien sind Zukunftstechnologien. Wer zu spät kommt, den bestraft das Leben, und wer sich nicht informiert, verliert den Anschluss.
- Weiterbildung organisieren – Stud.IP entwickeln,
  Marco Bohnsack, data-quest
  Bei dem webbasierten Ausbildungsmanagement sind die Open-Source-Lösungen besonders beliebt, da hier die Anpassung an eigene Anforderungen kostengünstig realisiert werden kann. Gleichzeitig steht professioneller Support zu angemessenen Preisen zur Verfügung. Die zweitägige Stud.IP-Jahrestagung bringt deutschsprachige Anwender und Entwickler aus Unternehmen und Hochschulen zum Erfahrungsaustausch zusammen.
- Jobs anbieten – Mitarbeiter finden
  Ausgewählte Firmen stellen sich vor und zeigen Zukunftsperspekitven in ihrem Unternehmen auf.
- Mobil bleiben – Apps erstellen,
  Dr. Jens Trapp, Google
  Eine der wichtigsten Entwicklungen im IT-Bereich finden in der mobilen Welt statt. Es ist Zeit sich mit den Grundlagen und aktuellen Trends zu beschäftigen.
- Weiterbildungs- und Seminarmanagement über Intranet und Internet
  Das Training im Umfang einer Stud.IP-Adminschulung gibt einen Überblick über die administrativen Möglichkeiten beim Einsatz der Open-Source-Management-Plattform Stud.IP.
- Projekt entwickeln – Datenbank auswählen
  Das Training soll helfen bei der Auswahl der Datenbank für ein Projekt die richtigen Entscheidungen zu treffen.

Infos unter http://www.sourcetalk.de/2011

Kommt es zum Bruch?

iJUG e.V.

Aus der deutschen Java-Community wurde die Idee an den iJUG e.V. herangetragen, aus dem Apache Harmony-Projekt und einem Fork des OpenJDK eine eigenständige Sprache zu entwickeln, die weitestgehend unabhängig von Oracle und dem „Original-Java" ist.

Für die weitere Diskussion wird zwischen der reinen Sprache Java (festgelegt in der Language Specification) und der Java Plattform (JVM, APIs) unterschieden. Auf der JVM lassen sich auch andere Sprachen verwenden, die außerhalb der Kontrolle des JCPs frei definiert werden können, allerdings innerhalb der technischen Grenzen, die die verfügbaren JVMs setzen. In dieser Diskussion geht es um die Plattform, insbesondere um die Java Platform Standard Edition.

Der diskutierte Vorschlag hätte in jedem Fall ein Aufsplitten des „Java-Ökosystems" zur Folge. Ein Teil würde sich dem Aufbau einer neuen Plattform widmen, ähnlich der Java-Plattform, aber prinzipiell unabhängig und stärker Open-Source-getrieben. Der andere Teil bliebe bei der ursprünglichen Plattform, die demgegenüber stärker kommerziell getrieben wäre, insbesondere von den großen Konzernen wie Oracle und IBM. Beide Teile würden nicht mehr die volle Unterstützung des heutigen großen Ökosystems erfahren. Und mit Sicherheit wäre es ein großer Umbruch, der viele Mitglieder auch dazu bewegen könnte, sich komplett neu zu zu orientieren. Das Risiko besteht, dass am Ende alle verlieren.

Die Schwierigkeiten, die ein solches Unterfangen mit sich bringt, sollten ebenfalls nicht unterschätzt werden. Reicht ein Ersatz für Java SE (entsprechend der Basis Harmony und OpenJDK-Fork) aus? Sollte nicht auch Java EE berücksichtigt werden, das gerade eine Renaissance erlebt? Darüber hinaus ist das Risiko einer juristischen Auseinandersetzung nicht zu unterschätzen, da hier viele Beteiligte viel zu verlieren haben. Das könnte das Innovationspotenzial beider Seiten deutlich beeinträchtigen, auch hier wäre das „Scheidungsopfer" Java wieder der Leidtragende.

Nach der Sun-Übernahme ist viel Bewegung in das Java-Ökosystem gekommen. Es häufen sich sowohl die negativen als auch die positiven Schlagzeilen. Negative, weil klar geworden ist, dass Oracle nicht nur ungeliebte Strategien von Sun fortsetzen wird, sondern Java noch stärker dominieren und ihm eine konsequentere kommerzielle Ausrichtung geben wird – was in weiten Teilen der Community nicht gut ankommt. Positive, weil endlich wieder Bewegung in die technische Weiterentwicklung gekommen ist und es zumindest für die wichtigsten Bereiche wieder eine klare Roadmap – und nicht nur Ideen – gibt. Inzwischen hat Oracle auch gezeigt, dass es die Roadmap ernst nimmt und die angekündigten Innovationen zügig vorantreibt. Bei allem Streit um einzelne Teile und Projekte sollte die offensichtlich immer noch vorhandene große Energie im Java-Ökosystem möglichst gebündelt und damit zum Vorteil aller genutzt werden. Das gilt für alle Beteiligten. Die Spaltung einzelner Projekte wie Hudson lässt sich sicher verschmerzen. Aber auch Oracle kann kein Interesse daran haben, dass das Java-Ökosystem sich insgesamt aufspaltet. Einige Signale lassen hoffen, dass diese Botschaft langsam dort ankommt.

Ein Argument, das pro Java Fork vorgebracht wird, ist die Möglichkeit der „Reduzierung von Altlasten". Mit anderen Worten, ein Fork soll dafür genutzt werden, Teile von Java zu entfernen, die nur aus Kompatibilitätsgründen noch im Sprachumfang enthalten sind, aber die weitere Entwicklung eher behindern. Dieses Argument ist nicht ganz von der Hand zu weisen. Eine Bereinigung des Sprachumfangs ist im Rahmen eines Forks natürlich viel einfacher – Kompatibilität wäre sowieso nur zweitrangig. Aber auch so sollte Java von alten Zöpfen befreit werden. Einen ersten zaghaften Ansatz gab es in Java EE 6. Das Thema sollte auf jeden Fall für neue JSRs immer wieder auf den Tisch kommen. Es wird nicht einfach sein, aber mit vereinten Kräften kann es klappen.

Die reine Drohung kann Oracle vielleicht dazu bringen, stärker auf die Open-Source-Community zuzugehen, einen unverzichtbaren Teil des Ökosystems. Aber ihre Umsetzung sollte die „ultima ratio" sein, da sie für alle Beteiligten negative Konsequenzen hätte. Die im Interessenverbund der Java User Groups e.V. organisierten Gruppen wünschen sich daher grundsätzlich eine gemeinsame Zukunft mit Oracle.

Beispiele wie Linux zeigen, dass ein Ökosystem mit einer zentralen Figur funktionieren kann – solange alle letztlich profitieren. Klar ist dann aber, dass die Rahmenbedingungen für alle Beteiligten stimmen müssen – auch und besonders für das OpenJDK. Hier muss Oracle sich zur Community bekennen. Es rechnet wohl niemand mehr damit, dass das TCK ohne Restriktionen freigegeben wird. Aber beim Thema OpenJDK „Governance" und „Oracle Contributor Agreement" (OCA) muss Oracle zeigen, dass es die Einbindung der Community ernst nimmt: Das „Governing Board" des OpenJDK darf kein Abnick-Gremium werden. Und es muss klargestellt werden, dass das OpenJDK nicht in ein „Open Core"-Modell driften kann, das am Ende alle Beteiligten von Oracle abhängig macht. Indem es alle wesentlichen Funktionen anbietet und nicht von Oracle durch Add-ons unter einer anderen (nicht Open-Source-) Lizenz ausgehöhlt wird; am besten aber durch eine Änderung des bzw. Ergänzung zum OCA, die dies sicherstellt.

„Any place – any time“

Der Wunsch, jederzeit und überall internetfähig und erreichbar zu sein, lässt mobile Endgeräte boomen. Java spielt bei deren Entwicklung eine tragende Rolle. Rolf Scheuch, Chief Strategy Officer (CSO) der OPITZ CONSULTING GmbH, zeigt die Hintergründe dieses Phänomens auf.

Wie ist der momentane Hype zu den mobilen Lösungen zu erklären?
Zwar haben die meisten Menschen heute einen Laptop mit Breitband-Internet-Anschluss beziehungsweise einen WLAN- oder UMTS-Internetzugang. Diese Geräte sind jedoch noch zu groß, zu umständlich und beim Booten zu langwierig, um allzeit einsatzbereit zu sein. So sind Handys und – stark zunehmend – Smartphones eine Alternative, insbesondere da diese Geräte zunehmend leistungsfähiger werden. Apple hat mit dem iPhone einen regelrechten Hype angestoßen; das mobile Internet ist zudem schneller und günstiger geworden. Ganz neue Möglichkeiten ergeben sich über die Nutzung der eingebauten Sensoren, etwa Google Maps Integration, Location Based Services oder Augmented Reality. Auch Video-Telefonie beginnt sich durchzusetzen. Die Innovationsgeschwindigkeit ist hier enorm.

Welche Bereiche sind davon am meisten betroffen?
Einen sehr hohen Anteil unter den Apps stellen die Spiele dar, gefolgt von Informationssystemen, die eher im persönlichen Umfeld interessant sind. Meine Lieblingsapps: Kicker Online und 1. FC Köln. Diese Informationssysteme nutzen in einem verstärkten Maße die eingebauten Funktionen des mobilen Endgerätes. Zum Beispiel kennt das System über GPS meinen Standort und kann mir über eine personalisierte Lokalisation Informationen zur Verfügung stellen. Mittels der eingebauten Kamera kann ich einen Barcode erfassen, umwandeln und über das System einen Preisvergleich durchführen. Unternehmen beginnen, mobile Lösungen als Möglichkeit zu sehen, um Prozesse zu beschleunigen, externe Partner noch leichter in Vorgänge einzubinden und so die eigenen Prozesskosten zu senken. Viele Business-to-Consumer-Angebote nutzen wir bereits für Flugbuchungen via Handy, Nachverfolgung von Paketsendungen, Terminauskünfte zu Lieferungen und Öffnungszeiten. Von Unternehmensseite sind hier weitere kreative Angebote gefragt. Ziel ist die Erhöhung der Kundenbindung.

Welche Sprachen bieten sich bei der Entwicklung mobiler Lösungen an?
Fast jede Plattform hat ihre eigene Programmiersprache oder eine recht proprietäre Entwicklungsumgebung. Die gängigsten Ansätze sind Apple mit Objective-c auf iOS und Microsoft mit Silverlight auf Windows Phone 7. Glaubt man den Markt-Analysen, so macht das Open-Source-Betriebssystem Android zurzeit bei den Wachstumszahlen den weitesten Sprung nach vorn. Den größten Marktanteil halten aber nach wie vor Symbian bei den Feature Phones und RIM (BlackBerry) bei den Smartphones. Der Abstand zu den Verfolgern wird allerdings geringer. Google, einer der Key-Player, hat die Java-basierte Android-Plattform initiiert, steht voll dahinter und entwickelt diese maßgeblich weiter. Nahezu alle Business-relevanten mobilen Anwendungen benötigen irgendeine Art von Backend zur Erfüllung der angebotenen Dienste. Hier ist Java extrem weit verbreitet und etabliert.

Warum ist Java eine gute Wahl für die Entwicklung mobiler Lösungen?
Die Programmiersprache kann hier ihre Stärken vollständig ausspielen, etwa bei der Bereitstellung neuer Endpunkte für bestehende Komponenten oder bei der Implementierung von Adapterschichten. Java bietet eine ausgesprochen gute Interoperabilität und genau das macht die Sprache in diesem Umfeld so beliebt. Zudem kommt Java als Programmiersprache auch im Client der Android-Plattform vor. JavaME, zwar sehr weit verbreitet auf Millionen von „Feature Phones“, hat sich hingegen nie durchsetzen können und wird dies in Zukunft vermutlich auch nicht schaffen. Java-Anwendungsentwicklung ist ebenfalls auf der BlackBerry-Plattform möglich und wird zudem von Oracle in Form von ADF Mobile für diverse Endgeräte unterstützt. Das von Sun initiierte JavaFX Mobile wird heute eher totgesagt.

Einen Quantensprung für die mobile Entwicklung stellt das neue HTML5 dar. Hierunter versteht man den aktuellen Stand von HTML mit diversen Erweiterungen, etwa für den Zugriff auf spezifische Gerätefeatures wie Kamera, Mikrofone, Kompasse, Bewegungssensoren oder GPS-Funktionalität inklusive der Verwendung von JavaScript und CSS. Auf diesem Wege sind bereits heute plattformübergreifende Lösungen möglich.

Java ist eine sehr gute Wahl zur Implementierung der nötigen Backend-Funktionalitäten. Im Bereich der mobilen Clients sieht dies anders aus: Android beispielsweise verwendet zwar Java als Programmiersprache sowie ein an das Java-Runtime-Environment angelehntes Software-Development-Kit, dieses ist allerdings nicht Byte-Code-kompatibel. Hieraus wird deutlich, dass es in der Tat technische Grenzen gibt, die den Einsatz von Java auf mobilen Endgeräten erschweren. Es gibt viele Ansätze, die versuchen, Java auf die mobilen Clients zu bringen. Unbekannt ist, wie sich dies in Zukunft entwickeln wird. Android zeigt massive Wachstumsraten. Es wird spannend sein zu sehen, wie sich die Verteilung der Plattformen in den kommenden zwei bis drei Jahren einpendeln wird.

Wie entwickelt man mobile Lösungen?

Die möglichen Architekturen in der Entwicklung von Mobile Apps sind extrem unterschiedlich – entsprechend gibt es auch unterschiedliche Ansätze, die mobilen Lösungen auf das Handy zu bringen. Eine Möglichkeit besteht darin, auf dem Endgerät einen Browser zu betreiben. Die mobile App ist somit eine für das Endgerät optimierte Web-Anwendung. Diesen Ansatz mögen Entwickler natürlich am liebsten, denn die Kontrolle bleibt in der Zentrale, Download beziehungsweise Aktualisierung von Software auf dem Endgerät sind nicht nötig und der vielleicht wichtigste Grund: Es bedarf keiner neuen Kenntnisse, da es sich um alte Internet-Technologie handelt, eben nur für kleinere Bildschirme. Deshalb sind die Entwicklungskosten bei diesem Ansatz auch geringer als beim Development von nativen Anwendungen. Diese werden direkt auf der jeweiligen Zielplattform erstellt.

Da viele Dinge hier mehrfach entwickelt werden müssen, steigen die Kosten. Diesem Problem nehmen sich heute bereits viele Zwischenlösungen an, etwa interpretierende Laufzeit-Umgebungen oder Cross-Compiler sowie MDA-Ansätze, die Anwendungen aus einer einheitlichen Code-Base für diverse Ziel-Plattformen wie PhoneGap oder Titanium erzeugen.

Ob man sich nun generell für webbasierte Lösungen oder eher für Native Apps entscheidet, hängt – wie immer – von den konkreten Anforderungen ab. Aktuell ist nicht vorhersehbar, welcher Trend sich durchsetzen wird.

Das App-Konzept wächst zurzeit stark, beispielsweise aufgrund der Verankerung im Mac-Betriebssystem oder über diverse neue Marktplätze für Apps wie Amazon. Webbasierte Lösungen haben ihre Vorteile bei normalen Oberflächen, die schnell entwickelt werden können, da hier weder ein Deployment noch die Auswahl einer Distributionsplattform nötig sind.

Fragen für die Zukunft sind: „Sollen die Lösungen verkauft oder nur angeboten werden?“ oder „Sind Subscriptions-Modelle nötig?“ Android ist hier generell offener als Apple, weshalb auch dort die Wachstumsraten hoch sind. Treiber sind neue Anwendungsfälle, die heute so noch gar nicht denkbar sind. Zum Beispiel könnte bei einem Autounfall eine Versicherungs-App auf dem Handy den folgenden Ablauf ermöglichen: GPS aktivieren und die Unfallstelle anzeigen, dann den Unfall mit der Kamera aufnehmen und per Sprach-Memo den Hergang beschreiben. Anschließend erfolgt die Weiterleitung der Daten über die Mobile App direkt an den Versicherer. Wie man so eine Anwendung in einer weitestgehend generischen Form für die gängigsten Smartphones entwickelt, hängt von den unterschiedlichen Optionen für den individuellen Fall ab, um sie für den Kunden passend umzusetzen.

Wo liegen die Unterschiede zwischen Oracle und Google?

Das lässt sich ganz gut an einem aktuellen Rechtsstreit verdeutlichen. Oracle hat Google wegen der Nutzung seines geistigen Eigentums für die Entwicklung von Android verklagt. So wie ich das sehe – ich bin allerdings kein Rechtsexperte – möchte Oracle mit dieser Klage erreichen, dass alle Kopien der entsprechenden Java-Projekte von Google beschlagnahmt und zerstört werden.

Auf der Suche nach dem Beweggrund von Oracle finden sich weitere Klagen gegen Android-Vermarkter. So hat Apple HTC, einen der wichtigsten Promoter von Android, wegen der Verletzung von Patentrechten verklagt. Zieht sich für Android damit die Schlinge zu? Aus meiner Sicht

Zur Person: Rolf Scheuch

Rolf Scheuch (Jahrgang 1957) begann seine Laufbahn bei der Bull AG im Bereich „Transaktionssysteme Großrechner“, nachdem er sein Mathematik-Studium an der Universität zu Köln mit dem Diplom abgeschlossen hatte. Nach der Gründung von OPITZ CONSULTING war er zunächst als Entwicklungsleiter tätig. Seit 2000 verantwortet er das Business Development und Marketing der Unternehmensgruppe.

Er hat von 1998 bis 2008 die Interessen der Oracle-Anwender im Vorstand der DOAG Deutsche ORACLE-Anwendergruppe e.V. vertreten. Heute ist er zudem als Management-Coach tätig und unter anderem im SOA-Umfeld als Autor diverser Bücher und Publikationen bekannt.

geht es den klagenden Unternehmen vor allem um Kontrolle. Google und andere verdienen ihr Geld mit Anzeigen oder Handy-Verkäufen und sind nicht so auf den Verkauf von Software-Lizenzen angewiesen wie etwa Oracle. Je geringer die Kosten und je höher die Verbreitung von Android sind, desto besser für diese Unternehmen und wahrscheinlich auch für den Markt.

Oracle ist dagegen noch nicht im Open-Source-Land angekommen und kann daher noch nicht so schnell auf den

Android-Zug aufspringen und von diesem profitieren. Stattdessen befürchtet der Datenbank-Hersteller Einbußen im Lizenzgeschäft. Die entscheidende Frage im Markt für mobile Lösungen lautet „Wer hat die Kontrolle über Java auf dem Endgerät?"

Was muss Oracle tun, um die Bedürfnisse der Entwickler zu erfüllen?

Oracle bietet, soweit ich weiß, derzeit nur eine Entwicklungsumgebung an, die mobile Variante des Application Development Framework (ADF). Die mobile ADF-Umgebung soll die Entwicklung nativer Java-Anwendungen für unterschiedliche mobile Betriebssysteme ermöglichen. Die Architektur besteht aus einem Browser zum Anzeigen der Apps, einer kleinen Datenbank zur lokalen Speicherung von Daten und einer Java-Runtime für Geräte, auf denen Java nicht installiert ist. Oracle nutzt das Framework „JavaServer Faces" für die Oberfläche und hat vorkonfigurierte Komponenten für die Daten-Synchronisation mit einem Server im Backend. Im Augenblick gibt es nur Lösungen für veraltete Windows Mobile 5/6 und die BlackBerry-Geräte. iOS, Android oder Windows Phone 7 werden aktuell nicht unterstützt. Dieser Ansatz sollte seitens Oracle gründlich überdacht werden.

Zwar besitzen die BlackBerry-Geräte, für die ADF Mobile begonnen wurde, immer noch einen sehr hohen Marktanteil unter den Business-Anwendern. Aus meiner Sicht begibt Oracle sich derzeit aber in eine Sackgasse, wenn es viele moderne Plattformen nicht bedient. Im Markt und auch in der Java-Community entstehen nämlich bereits deutlich flexiblere und robustere Architekturen mit den entsprechenden Frameworks. Insbesondere können Native Apps mit kompletter Kontrolle über die Endgeräte erstellt werden. Da steht Oracle noch einige Arbeit bevor.

Welche Erwartungen haben Sie an die Java-Community?

Ich würde mich freuen, wenn sich die deutsche Java-Community diesem Thema stärker widmen und mehr Gelegenheiten schaffen würde, bei denen sich Entwickler treffen und austauschen können. Wir können noch viel voneinander lernen. Die Technologie ist noch frisch – und die ersten Best- und Worst-Practices sind gerade im Entstehen. Java-Entwickler werden vermutlich wieder ein paar weitere Sprachen für spezielle Einsatzszenarien lernen müssen. Ganz im Sinne von Bruce Tate, der in seinem „Beyond Java" forderte, sich pro Monat eine weitere Sprache anzusehen. Das Wichtigste ist also, Raum für Diskussion und Informationsaustausch zu schaffen. Das wird allen Beteiligten helfen und die Marktdurchdringung insgesamt fördern. Vielleicht kann der iJUG hier ja unterstützen …

Pressemeldung des iJUG

Auch in diesem Jahr wird Oracle die JavaOne wieder parallel zur OpenWorld in San Francisco veranstalten. Die im Interessenverbund der Java User Groups e.V. (iJUG) zusammengeschlossenen deutschen Java-Anwender befürchten, dass die JavaOne dadurch zunehmend mehr an Bedeutung verliert.

Java-Anwender wünschen sich neue Impulse von der JavaOne für die Community

Die Dominanz und Bedeutung der JavaOne ist spürbar gesunken. Insbesondere in Europa gibt es mittlerweile alternativ dazu mehrere große Java-Konferenzen. Zudem waren die Teilnehmer der letztjährigen JavaOne mit ihrem Tagungsort sehr unzufrieden. Neben den unzureichenden Räumen sowie der intransparenten Konferenz-Organisation bemängeln die Teilnehmer auch den fehlenden Community-Spirit. „Die Teilnehmer erwarten im Rahmen der JavaOne beispielsweise die früher vorhandene Community-Ecke mit Mini-Talks oder den freien Gedankenaustausch in den Birds-of-a-feather-Sessions", so Michael Hüttermann von der Java User Group Köln.

Michael Hüttermann war selbst als Referent auf der letzten JavaOne vor Ort vertreten und meint dazu: „Oracle leitet den JCP sowie wichtige JSRs. Genau von diesen Leuten will die Community wissen, wohin die Reise geht beziehungsweise die Richtung auch mit beeinflussen. Die JavaOne wäre eine ideale Gelegenheit dafür."

Fried Saacke, Vorstandsvorsitzender des iJUG, fasst zusammen: „Um der JavaOne wieder den Spirit früherer Jahre zu verleihen, muss Oracle die Bedürfnisse der Community einbeziehen. Ansonsten werden andere Java-Veranstaltungen der JavaOne den Rang als führende Java-Konferenz ablaufen."

GlassFish 3.1

Markus Eisele, msg systems ag

Das ehemalige Flaggschiff von Sun, der Java-EE-Applikationsserver, erhielt bereits mit Erscheinen der neuen Java-EE-6-Spezifikation eine „3" als Versionsnummer. Nach der Übernahme durch Oracle wurde viel über die Zukunft des GlassFish spekuliert. Die Befürchtungen galten vordringlich der Tatsache, dass GlassFish keine Cluster-Unterstützung bekommen würde. Die in Kürze erscheinende Version 3.1 markiert nun einen weiteren Meilenstein auf der von Oracle Anfang letzten Jahres vorgestellten Roadmap für den in der Community bekannten und beliebten GlassFish-Server.

Noch Anfang März 2010 war sich die Community einig: Oracle wird den GlassFish zur Referenz-Implementierung degradieren. Er bleibt ein Kinderspielzeug und wird sich auch aufgrund der fehlenden Cluster-Unterstützung nicht für den produktiven Einsatz eignen. Was die noch unter Sun entstandene Vorgängerversion 2.1 bei vielen Kunden zu einer echten Alternative für kommerzielle Produkte wie beispielsweise den WebLogic-Server machte, fehlte der 3.0-Version bis heute. Mit dem Release 3.1 macht Oracle den Spekulationen um eine unsichere Zukunft des beliebten Open-Source-Servers ein vorläufiges Ende und erfüllt eines der Versprechen der Anfang 2010 vorgestellten Produkt-Roadmap. Zudem sind viele neue Funktionen hinzugekommen.

Das Projekt „3.1" wurde am 19. April 2010 gestartet. Nach rund zehn Monaten, acht Milestones, insgesamt 43 „promoted builds" sowie Tausenden von Bug-Reports und Fixes ist ein modularer Java-EE-6-GlassFish-3-Kernel entstanden, der Clustering, zentrale Administration und Hochverfügbarkeit liefert. Die Version 3.1 verbindet die Vorteile von GlassFish 3.0 und GlassFish 2.x und reichert diese mit vielen neuen Funktionen an. Sie basiert, wie ihr Vorgänger, auf dem H2K-Mikrokernel entsprechend dem OSGi-Standard und stellt somit eine leicht erweiterbare, flexible Plattform dar.

Clustering und Hochverfügbarkeit

Im direkten Vergleich mit der clusterfähigen Vorgängerversion 2.1 wurde nur unter der Haube etwas weiterentwickelt. Eine der nennenswerten Änderungen ist der Wegfall des sogenannten „Node-Agents". War dieser in der 2.1 noch die zentrale Schaltstelle für die Cluster-Kommunikation, geschieht dies bei 3.1 auf zweierlei Arten: Die Synchronisation von Server-Konfigurationen und Anwendungen wird direkt per SSH-Zugriff vom Domain Admin Server (DAS) auf die entfernten Maschinen vorgenommen. Handelt es sich um lokale Instanzen, greifen alle auf das gleiche Filesystem zu. Die im Cluster anfallenden Benachrichtigungen sind über den Group Management Service (GSM) verwaltet. Diese auf dem Shoal-Framework basierende Komponente benachrichtigt den DAS über den Zustand einzelner Instanzen und verantwortet darüber hinaus auch andere Cluster-Funktionalitäten (wie in-memory session replication, transaction services und timer services). Trotz dieser neuen Funktionalität bleibt der DAS die zentrale Schwachstelle im Cluster. Er ist nach wie vor nicht ausfallsicher und andere Knoten können seine Aufgaben nicht übernehmen.

Die Cluster-Funktionen im GlassFish erstrecken sich auf drei Bereiche: Lastverteilung auf verschiedene Knoten und Instanzen erfolgt mit dem Apache „mod_jk". Hochverfügbarkeit für Session-Daten wird per „in-memory"-Session-Replikation erreicht. Der Java Message Service (JMS) wird via Connection-Pooling und Failover sowie MQ-Clustering hochverfügbar. Alle Cluster-Funktionen verwenden als Basis RMI-IIOP-Last-Verteilung und Failover der Namensdienste (JNDI). Im direkten Vergleich mit GlassFish 2.1.1 stieg die Cluster-Performance um 34 Prozent [1]. Dazu gehört auch die maximal mögliche Anzahl von Instanzen im Cluster.

Produktivität der Entwickler

Grundlage für eine gesteigerte Entwickler-Produktivität ist nach wie vor die Java-EE-6-Spezifikation. Im Gegensatz zur Vorgängerversion ist diese um einiges leichtgewichtiger und einfacher zu implementieren. Aber auch der GlassFish als Laufzeit-Umgebung hat mit der neuen Version nachgelegt und bietet Verbesserungen für Entwickler. Startup- und Deployment-Zeiten sind durchschnittlich um 30 Prozent geringer [2]. Dies gelingt wohl vordergründig durch das dynamische Laden der Module im H2K-Kernel. Bei der Gesamtzahl von 262 einzelnen Modulen ist diese Geschwindigkeitssteigerung durchaus eine Leistung. Zudem ist die Version 3.1 in der Lage, den HttpSessionState nach erneutem Deployment wiederherzustellen. Diese bei Entwicklern komplexer Anwendungen beliebte Funktionalität erreicht nun auch Statefull-Session und Timer-Beans. Das vereinfacht das Beheben komplexer Fehlersituationen.

Auch bei der Integration mit den Entwicklungs- und Build-Umgebungen hat sich einiges geändert. Der Embedded-GlassFish hat eine neue API bekommen und auch die zugehörigen Maven-Plug-ins wurden überarbeitet. Damit wird automatisiertes Testen im Container einfacher. Als Entwicklungsumgebungen sind Eclipse und NetBeans aus erster Hand unterstützt. Zudem wird das noch in der Beta-Phase befindliche NetBeans 7.0 eine umfassende Unterstützung bekommen. Aber auch die Eclipse-Server-Adapter wurden entsprechend erweitert. Eine noch stärkere Integration gelingt nur noch mit dem Oracle Enterprise Pack for Eclipse (OEPE).

Management und Monitoring

Der Bereich „Management und Monitoring“ ist nun auf den kompletten Cluster ausgeweitet. Neben der bereits angesprochenen zentralisierten Administration gehört auch die erweiterte Skalierbarkeit des DAS dazu. Eine komplette RESTful-API für die Admin-Console ermöglicht die komplette Fernsteuerung von Installationen. Die französische Firma Seril hat ein Versionierungssystem für Deployments gestiftet, das nun ebenfalls verfügbar ist. Mithilfe des Application-Versionings ist es möglich, Deployments mit Versionsnummern zu versehen und diese gezielt zu administrieren. Ebenfalls neu ist die Möglichkeit, sogenannte „application scoped resources“ zu definieren. Dazu stehen eigene Deployment-Deskriptoren bereit, in denen man die Konfiguration benötigter Server-Ressourcen hinterlegt (JDBC, JMS, Connector, JavaMail, Custom, JNDI etc.). Die „glassfish-resources.xml“ genannten Deskriptoren ermöglichen das Anlegen von modulbezogenen Ressourcen und liegen daher entweder im WEB-INF- oder META-INF-Verzeichnis der jeweiligen Module. Sie werden im Server beim Deployment registriert und im Falle eines Undeployments auch wieder entfernt.

Auch das JDBC-Monitoring wurde erweitert. Neben der Statement-Leak-Erkennung lassen sich jetzt auch SQL-Queries umfangreicher verfolgen. Zudem kann man die Connection-Pool-Nutzung einer Anwendung überwachen, was bisher nur auf der Ebene der serverweiten Connection-Pools möglich war. Zur Überwachung sind jetzt auch eigene Validation-Templates für Connection-Pools möglich. Auch im Security-Bereich sind neue Funktionen hinzugekommen. Unix-basierte Betriebssysteme können jetzt die sogenannte „Pluggable Authentication Module (PAM) Realm“ verwenden. Damit werden unter Unix angelegte Benutzer direkt auf Anwendungen im GlassFish zugelassen. Für Anwendungen mit erweiterten Authentisierungsanforderungen steht die Certificate-Realm bereit, um zusätzliche Funktionen in zertifikatsbasierte Login-Module einzubauen, die sich in ein auf Client-Zertifikaten basiertes Szenario einfügen lassen. Darüber hinaus sind eine Vielzahl von kleineren Security-Verbesserungen eingeflossen [3].

Die Diskussion um Sinn und Zweck der Koexistenz beider Applikationsserver (GlassFish und WebLogic) im Hause Oracle hat mit dieser GlassFish-Version eine neue Richtung erhalten. Erstmalig ist dieser in der Lage, definierte Deployment-Deskriptoren des WebLogic zu lesen. Damit ist es erstmals möglich, Anwendungen für WebLogic auch auf GlassFish zu entwickeln – und zwar bereits mit den spezifischen Konfigurations-Informationen der eigentlichen Zielplattform. Aktuell ist allerdings nur die „weblogic.xml“ unterstützt, weitere sollen folgen. Auch auf Seiten des WebLogic-Servers sollen mittelfristig die GlassFish-spezifischen Deskriptoren unterstützt werden.

Technologie-Updates

Abseits der bisherigen Neuerungen sind auch etliche Java-EE-6-Technologien verbessert. JSF wurde von 2.0 auf 2.1 angehoben. Die im GlassFish enthaltene Referenz-Implementierung von Contexts- und Dependency-Injection-Weld ist auf die Versionsnummer 1.1 aktualisiert. Die Referenz-Implementierung für JPA 2.0 Eclipselink ist jetzt in Version 2.1 enthalten und auch die OSGi-Runtime des GlassFish (Felix) aus dem Hause Apache wurde auf eine neuere Version (3.0) gebracht. Nicht zuletzt sind auch Grizzly und Jersey auf neue Versionen angehoben worden, die eine Menge an Bugfixes enthalten. Vor allem das Grizzly-Update bringt den lang ersehnten WebSocket-Support auch in GlassFish. Einziger Wehrmutstropfen ist, dass sich das WebSocket-Protokoll schneller weiterentwickelt als es neue GlassFish-Versionen gibt. Daher ist die „-76“-Version des Protokolls schon mit Veröffentlichung von GlassFish veraltet. Dank der Modulari-

sierung ist ein Austausch aber vergleichsweise einfach möglich.

Kommerzielle Erweiterungen

Neben der Open-Source-Version gibt es auch eine kommerzielle Variante von Oracle, die neben dem Support noch weitere Add-ons enthält. Diese stehen nicht unter der Open-Source-Lizenz und sind somit nur für zahlende Kunden verfügbar. Dazu gehört beispielsweise das GlassFish-Server-Control (früher GlassFish Enterprise Manager). Es enthält Funktionen für Backup & Recovery vom DAS. Hiermit können komplette Domains in Archive überführt und im Katastrophenfall wieder zurückgespielt werden. Der Performance-Tuner analysiert die zugrundeliegende Infrastruktur und optimiert die Laufzeit-Einstellungen des GlassFish im Hinblick auf Durchsatz und Skalierbarkeit. Der Monitoring-Scripting-Client bietet eine JavaScript-Schnittstelle zur Laufzeit und ermöglicht das einfache, programmatische Auslesen von Laufzeit-Informationen. Gleichfalls nur zahlenden Kunden vorbehalten bleiben die JMS-Optimierungen zur Kommunikation mit WebLogic und WebSphere-Servern.

Für Solaris-basierte Installationen in Kombination mit dem JDK 7 steht eine Integration der GlassFish-Metriken für DTrace zur Verfügung. Bereits von aktuellen WebLogic-Versionen bekannt ist der ActiveCache. Die Zusammenarbeit von Java-EE-Server und Coherence ist zukünftig auch für GlassFish verfügbar, um Session-Informationen im Coherence-Cache zu speichern. Die dafür notwendige Coherence-Version 3.7 ist allerdings noch nicht verfügbar, wird aber noch im ersten Halbjahr dieses Jahres erwartet. Die Integration mit dem Oracle Access Manager (OAM) dürfte vornehmlich für umfangreiche Oracle-Infrastrukturen interessant sein. Die bisher beschriebenen Funktionen sind bereits im kommerziellen Produkt enthalten. Der darüber hinaus noch erhältliche LoadBalancer und die entsprechenden Plug-ins kommen mit einem separaten Installer und ermöglichen es, den Oracle-Web-Server als Reverse-Proxy für Failover-Szenarien zu verwenden.

Upgrade-Pfade

Von alten Versionen umzusteigen ist vergleichsweise einfach. Alle 2.x Installationen lassen sich mithilfe des Upgrade-Tools aktualisieren. Dabei werden die neue Version 3.1 parallel installiert und einzelne Domains über das Werkzeug aktualisiert. Hier unterstützt das Upgrade-Tool für Domains ab Sun Java System Application Server 9.1 Update 2 bis einschließlich Oracle GlassFish 3.0.1. Einfacher geht es mit dem Update Center. Alle 3.x-Versionen des GlassFish lassen sich direkt über die eingebaute Aktualisierungsfunktion updaten. Für Java-EE-6-basierte Anwendungen ist kaum mit Migrationsaufwand zu rechnen. Auch wenn einige Updates der Implementierungen durchaus Auswirkungen auf die Entwicklung haben können.

Fazit

Was versprochen wurde, konnte gehalten werden. Oracle präsentiert mit dem GlassFish 3.1 den ersten hochverfügbaren Java-EE-6-Server und lässt damit erneut viele Mitbewerber hinter sich. Kommerzielle Server sind erst für das letzte Quartal dieses Jahres zu erwarten. Bei den Open-Source-Mitbewerbern sieht es nicht viel besser aus. Wer mit der aktuellen Java-EE-Version im großen Stil produktiv gehen will, hat aktuell keine Alternative. Dennoch bleiben viele Baustellen offen. Die Zusammenarbeit mit den Nicht-Oracle-Lieferanten von Referenz-Implementierungen bietet beliebig Zündstoff – allen voran ist Weld ein Zankapfel. Für dringend notwendige Verbesserungen in der nächsten Weld-Version kam das Release 3.1 zu früh. Hier müssen alle Beteiligten zukünftig besser zusammenarbeiten, um die Kritik aus der Community zu entkräften.

Die ersten Arbeiten an einer gemeinsamen Plattform mit dem WebLogic-Server wurden für Anwender vergleichsweise geräuschlos abgeschlossen. Dies äußert sich offensichtlich in den deutlich besseren Leistungsdaten im Cluster. Ersten gemeinsamen Deployment-Deskriptoren werden weitere folgen. Es bleibt zu spekulieren, wie die fertige Implementierung der gemeinsamen Basis ausgeprägt sein wird. Im besten Fall besteht ein WebLogic-Server zukünftig aus den GlassFish-Modulen und erweiterten Add-ons. Ein Indikator hierfür können die ebenfalls hinzugekommenen Service Provider Interfaces (SPI) sein, die eine Erweiterung von GlassFish ermöglichen. Der Premier Support für den GlassFish 3.1 endet im März 2016; extended Support kann bis Ende März 2019 bezogen werden.

Weitere Informationen

[1] http://weblogs.java.net/blog/sdo/archive/2011/03/01/whats-new-glassfish-v31-performance

[2] http://www.youtube.com/watch?v=O9TCWZ-nIgo

[3] http://blogs.sun.com/gfsecurity/entry/what_s_new_in_glassfish

[4] http://terrencebarr.wordpress.com/2011/03/07/glassfish-webinar-series-on-youtube/

Kontakt:
Markus Eisele
markus.eisele@msg-systems.com

Markus Eisele arbeitet im Bereich Software-Technologie im Center of Competence IT-Architecture der msg systems ag. Bei seiner täglichen Arbeit begleitet er Kunden und Projekte auf dem Weg durch die Tücken neuer Technologien und Produkte im Java-EE-Umfeld.

Das Cargo-Kult-IT-Problem

Oliver Szymanski, Freiberufler, Source-Knights.com

Nach dem zweiten Weltkrieg haben sich inbesondere im Bereich des Pazifischen Ozeans Gebräuche entwickelt, die durch Beobachtung entstanden. Ähnliches gab es auch zur Zeit der Kolonialherrschaften. Man könnte es als „Copy & Paste" von Verhaltensweisen interpretieren. Die dadurch entstandene Kultur, basierend auf Riten, bezeichnet man als „Cargo-Kult". Es ist es wert, dies in Hinsicht auf Software-Entwicklung näher zu betrachten.

Zahlreiche Inseln im Pazifischen Ozean wurden von Einheiten der Alliierten im zweiten Weltkrieg als Stützpunkte genutzt. Die Soldaten lebten dort, aber natürlich gab es häufig auch ursprüngliche Bewohner. Viele dieser Bewohner lebten in sich von unseren Zivilisationsvorstellungen abweichenden Gemeinschaften. Sie hatten andere Glauben und Wertvorstellungen. Wie auch immer, Kulturen trafen aufeinander. Und beobachteten sich. Mancher Zyniker mag sagen, die Soldaten studierten die ursprünglichen Einwohner sicher weniger als umgekehrt, aber das müssen wir gar nicht weiter betrachten. Denn der Cargo-Kult bezieht sich auf die Beobachtungen und Rückschlüsse der ursprünglichen Bewohner.

Cargo

Das zahlreiche Material, dass die Truppen brauchten, wurde meist per Luftfracht gesendet und mit Fallschirmen abgeworfen. Das blieb natürlich nicht verborgen. Teilweise wurden die Soldaten und die Bewohner mit Material jeder Art schier überhäuft. An so etwas kann man sich gewöhnen – Nachschub im Überfluss, der vom Himmel fällt.

Cargo-Kult

Allerdings verließen die Truppen die Inseln wieder, wie uns die Geschichte lehrt. Damit blieb auch das Material aus. So, nun muss man versuchen, sich ein wenig in die Köpfe anderer Zivilisationen hineinzuversetzen. Diese beobachteten in den Zeiten, in denen noch Cargo vom Himmel fiel, fremde Menschen, die auf platten Ebenen der Inseln standen, seltsame Sachen am Kopf beziehungsweise Ohr trugen und wilde Zeichen machten. Der Mensch ist seit Kindheit auf Beobachtung und Nachahmung angewiesen. Es entwickelten sich tatsächlich Kulte, die begannen, Kopfhörer aus Holz zu schnitzen, Flugzeugmodelle zu bauen, Landebahnen-ähnliche Anlagen nachzubauen und Signalfeuer zu entzünden. Alles, damit endlich wieder Cargo fiel.

Copy & Paste

Letztlich ist dies nichts weiter als Copy & Paste von Verhaltensweisen in der Hoffnung, zu gleichen Ergebnissen zu kommen. Worauf soll das hinauslaufen? In der heutigen IT-Welt versuchen Entwickler auch, gegenseitig Lösungen „abzugucken" und auf ihre eigenen Probleme zu adaptieren. Seien es Entwurfsmuster oder das direkte Kopieren von Codezeilen, Skripten und Architekturentwürfen.

Ergebnis

Das Ergebnis kann zufriedenstellend sein. Aber die Erfahrung zeigt, dass ein Verständnis des Originalproblems vorliegen muss, um etwas erfolgreich zu adaptieren. Die fremden Vorbedingungen müssen einem klar sein und sollten mit den eigenen übereinstimmen.

Das Cargo-Kult-IT-Problem

Auch Firmen betreiben Cargo-Kult. Sie denken, sie könnten ihre Probleme so lösen, wie andere Firmen es tun. Das bezieht sich bereits im Abstrakten sowohl auf Requirements-Analysen als auch auf Software-Architekturen, Code und Prozesse. „Wenn agile Entwicklung bei denen funktioniert hat, muss es auch bei uns genauso funktionieren", hört man manchmal – und dann ahmt man nach, was man gesehen hat. Aber alles lässt sich nur mit Erfolg kopieren, wenn die Vorbedingungen passen und man es dann auch noch richtig kopiert. Ein Kopfhörer aus Holz arbeitet nun einmal anders als das beobachtete Original. Beobachten ist gut und wichtig. Zu sehen, wie andere Probleme gelöst haben, ist unerlässlich für die eigene Lösungssuche. Doch am wertvollsten ist das Verstehen. Es schadet nicht, jemanden einzuladen und über dessen Lösung zu reden, bevor man versucht diese nachzubauen.

Kontakt:
Oliver Szymanski
oliver.szymanski@source-knights.com

Oliver Szymanski (Dipl. Inform., Univ.) ist als Software-Architekt / Entwickler, Berater und Trainer in den Bereichen Java, .NET und Mobile-Development tätig. Parallel dazu arbeitet er als Schriftsteller (siehe auch Seite 53)

PDF-Dokumente mit iText

Gunther Petzsch, Saxonia Systems AG

Immer häufiger wird die Generierung von PDF-Dokumenten im Funktionsumfang von Software-Produkten verlangt. Dafür gibt es am Markt eine Vielzahl von Lösungen. Dieser Artikel stellt die PDF-Generierung mittels Java und der Bibliothek „iText" vor.

Die Open-Source-Bibliothek „iText" (AGPL-Lizenz) wurde für das Erstellen, Lesen und Verändern von PDF-Dokumenten entwickelt. Die erste Version erschien im Jahr 2000. Seitdem wurde das Produkt von der Firma 1T3XT BVBA (Gent, Belgien) kontinuierlich verbessert und erweitert. iText wurde mit dem Ziel entwickelt, die beste und einfachste Open-Source-PDF-Engine auf dem Markt zu sein.

Der Funktionsumfang von iText lässt sich in drei Gruppen gliedern: Generieren, Lesen und Bearbeiten von Dokumenten, wobei der Umfang beim Lesen auf das Extrahieren von Daten beschränkt ist (siehe Abbildung 1). Die Bibliothek steht aktuell in der Version 5.0.6 (Stand 14. Februar 2011) zum Download auf der Internetseite www.itextpdf.com bereit.

Die nächsten Abschnitte zeigen, wie iText für unterschiedliche Anforderungen wie einfache Texte, Bilder, Logos und Barcodes eingesetzt werden kann. Alle Beispiele sind mit der Produktversion 5.0.6 entstanden.

Einfaches PDF-Dokument

Zunächst wird ein einfaches PDF-Dokument erzeugt. Dieses soll DIN-A4-Format besitzen sowie eine Überschrift und ein paar Zeilen Text beinhalten. Dies lässt sich mittels iText in fünf Schritten umsetzen (siehe Abbildung 2):

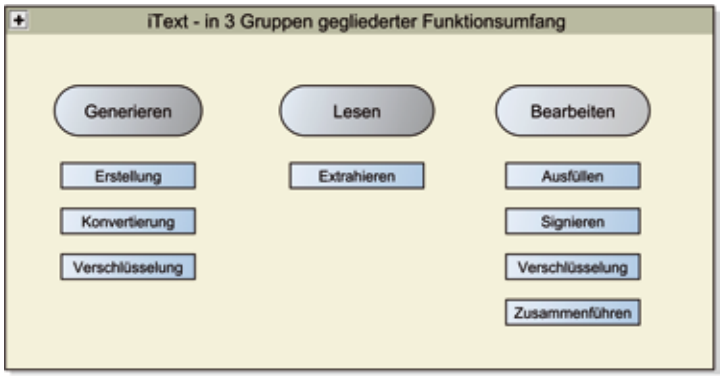


Abbildung 1: Funktionsumfang von iText und PDF-Dokumenten

```
package article.itext;

import com.itextpdf.text.Document;
import com.itextpdf.text.Font;
import com.itextpdf.text.FontFactory;
import com.itextpdf.text.PageSize;
import com.itextpdf.text.Paragraph;
import com.itextpdf.text.Phrase;
import com.itextpdf.text.pdf.PdfWriter;
import java.io.FileOutputStream;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * Simple example for creating a pdf document with iText
 * @author GPetzsch
 */
public class HelloWorld {

    /**
     * main method
     * @param args
     */
    public static void main(String[] args){
        HelloWorld.createPDF();
    }


    private static void createPDF(){
        /* step 1*/
        Document doc = new Document(PageSize.A4);
        try{
            /*step 2*/
            PdfWriter.getInstance(doc,
                    new FileOutputStream(«c:\\Temp\\HelloWorld.pdf»));
            /*step 3*/
            doc.open();
            /*step 4*/
            doc.add(new Paragraph(
                    «1. Einfaches Beispiel f,r Java aktuell»,
                            FontFactory.getFont(«Arial», 16F, Font.BOLD)));

            doc.add(new Phrase(«Hier ist unser erstes PDF-Dokument.»));
            /*step 5*/
            doc.close();
        }catch (Exception ex){
            Logger.getLogger(HelloWorld.class.getName()).log
            (Level.SEVERE, null,ex);
        }
    }
}
```

Abbildung 2: Beispiel HelloWorld.java

```
/**
 * simple example for a font
 */
private static final Font BOLD_UNDERLINED =
        new Font(Font.FontFamily.COURIER, 12, Font.ITALIC);



/**
 * simple template for JAVA aktuell
 * @param text as String
 * @return Phrase with formatted text (style italic)
 */
private static Phrase templateStringAsItalic(String text){
    Phrase director = new Phrase();
    director.add(new Chunk(text, BOLD_UNDERLINED));
    return director;
}
```

Abbildung 3: Beispiel für ein Template

Einfacher Barcode für Java aktuell

```
                        /**
                         * simple barcode example
                         * @return Barcode object
                         */
                        private static Barcode createBarcode(){
                            BarcodeEAN barcodeEAN = new BarcodeEAN();
                            barcodeEAN.setCodeType(BarcodeEAN.EAN13);
                            barcodeEAN.setCode(«4191978304903»);
                            return barcodeEAN;
                        }
import com.itextpdf.text.pdf.Barcode;
import com.itextpdf.text.pdf.BarcodeEAN;
import com.itextpdf.text.pdf.PdfWriter;


    doc.add(new Paragraph(
                «Einfacher Barcode f¸r Java aktuell»,
                    FontFactory.getFont(«Arial», 16F, Font.BOLD)));

    Image img = BarcodeTest.createBarcode().createImageWithBarcode(
                writer.getDirectContent(), BaseColor.BLACK, BaseColor.BLACK);

    doc.add(img);
```

Abbildung 4: Barcode und Quellcodeausschnitte

```
Image img = Image.getInstance(
                Toolkit.getDefaultToolkit().createImage(«c:\\Temp\\logo.gif»), null);
```

Abbildung 5: Bestehende Grafiken in PDF-Dokumenten einbinden

- Ein neues Dokumenten-Objekt vom Typ „com.itextpdf.text.document“ erzeugen
- Ein neues Writer-Objekt vom Typ „com.itext.text.pdf.pdfwriter“ erzeugen
- Die Methode „open()“ auf dem Dokumenten-Objekt aufrufen
- Mittels Methode „add()“ Text oder andere Objekte dem Dokumenten-Objekt hinzufügen
- Die Methode „close()“ auf dem Dokumenten-Objekt aufrufen

Templates

Möchte man nun Daten aus unterschiedlichen Datenquellen – wie zum Beispiel aus einer Datenbank – für die Dokumenten-Erstellung benutzen, wird lediglich ein gefülltes Resultset mit Daten benötigt. Diese Daten können dann mittels eines Templates aufbereitet und einem Dokument hinzugefügt werden (siehe Abbildung 3). Somit ist es möglich, viele Objekte mit unterschiedlichsten Konfigurationen, bezogen auf Schriftart, Größe etc. in ein Dokument einzubinden.

Barcodes

iText kann auch Barcodes generieren und in die zu erstellenden Dokumente aufnehmen. So entstehen PDF-Dokumente, wie man sie etwa aus dem Kino kennt. Dort werden die Barcodes zur Autorisierung bei Online-Kinokarten eingesetzt. Erzeugt wird ein Barcode mittels der gleichnamigen Klasse, indem man den gewünschten Typ und Wert setzt und in das Dokument einfügt. Im Beispiel (siehe Abbildung 4) wird ein Barcode vom Typ EAN13 generiert und einem Dokument hinzugefügt.

Bilder in PDF-Dokumenten

Bilder mittels iText in PDF-Dokumente einzubinden ist ebenso einfach wie das Erzeugen von Barcodes. Bei bestehenden Bilder oder Grafiken ruft man die entsprechende „getInstance()“-Methode von der „com.itextpdf.text.Image“-Klasse auf. Dabei entsteht ein neues Bildobjekt (siehe Abbildung 5), das dann dem Dokument hinzugefügt wird.

Das Erzeugen eigener Grafiken ist dagegen mit etwas mehr Aufwand verbunden. Zum einen müssen die genauen Koordinaten bestimmt werden und zum anderen ist etwas mehr Codierung notwendig. Auch hier bietet iText eine Lösungsmöglichkeit. Die Klasse „PDFContentByte“ fügt positionierte Grafiken sehr gut in ein Dokument ein. Über die Methoden „lineTo()“ und „moveTo()“ werden Punkte an die jeweiligen Positionen gezeichnet. Die Methode „saveState()“ speichert das Objekt in dem Dokument und die Methode „eoFillStoke()“ füllt die Regionen mit Farbe. Diese Farbe muss vorher am Objekt mithilfe der Methode „setColorFill()“ gesetzt sein (siehe Abbildung 6). Als Alternative für dieses Vorgehen bietet iText auch die Möglichkeit, dass Eigenentwicklungen Grafiken direkt in einem vorgegebenen Java-Graphics-Kontext erstellen können, welche sich somit in ein Dokument einbinden lassen.

Fazit

iText ist ein mächtiges Werkzeug rund um das Erzeugen, Lesen und Bearbeiten von PDF-Dokumenten. Der Funktionsumfang für das Erzeugen beinhaltet zahlreiche Einsatzgebiete, die auch die Liste

```
PdfContentByte canvas = writer.getDirectContent();
DrawStarTest.createStar(canvas,80,670,60,20);


/**
 * creates a simple star
 * @param canvas as PdfContentByte
 * @param x position as float
 * @param y position as float
 * @param radius as float
 * @param gutter as float
 */
private static void createStar(PdfContentByte canvas,
      float x, float y, float radius, float gutter){


   canvas.saveState();
   canvas.setColorStroke(new GrayColor(0.2f));
   canvas.setColorFill(new CMYKColor(13,255,255,2));
   canvas.moveTo(x + 10, y);
   canvas.lineTo(x + 80, y + radius);
   canvas.lineTo(x, y + radius);
   canvas.lineTo(x + 70, y);
   canvas.lineTo(x + (radius - gutter), y + 90);
   canvas.closePath();
   canvas.fill();
   canvas.eoFillStroke();
   canvas.restoreState();
}
```

Abbildung 6: Gezeichnete Grafik mittels iText (Ergebnis und Quellcode-Ausschnitt)

der Referenzprojekte und Firmen auf der Internetseite von iText widerspiegelt. Hervorzuheben ist auch der ständige Weiterentwicklungsprozess. Das Produkt befindet sich seit dem Jahr 2000 auf dem Markt und wird kontinuierlich verbessert. Das Verwenden von festen Formatvorgaben im Quellcode kann aus der Sicht des Autors aber zu Problemen führen. Damit ist es kaum möglich, Formate, Schriftarten oder Ähnliches zur Laufzeit zu ändern. Ist dieser Punkt aber eine Anforderung an die zu entwickelte Anwendung, ist dies beim Design entsprechend zu beachten. Danach sind auch das Ändern von Formaten, Schriftarten etc. zur Laufzeit bei der Verwendung von iText kein Problem mehr und lässt sich leicht realisieren. Dies ist jedoch aufwändiger gegenüber anderen Produkten wie beispielsweise Jasper Reports und sollte beim Einsatz von iText berücksichtigt werden.

Kontakt:
Gunther Petzsch
Gunther.petzsch@saxsys.de

Gunther Petzsch ist Sun-zertifizierter Java-Entwickler. Er lebt und arbeitet in Dresden. Dort studierte er auch erfolgreich Wirtschaftsinformatik an der Hochschule für Technik und Wirtschaft Dresden. Danach arbeitete er unter anderen für das Bundeskriminalamt in Wiesbaden. Aktuell ist Gunther Petzsch als Senior Consultant für die Saxonia Systems AG tätig.

JUGS
java user group stuttgart

Java Forum Stuttgart

Die Java User Group Stuttgart e.V. veranstaltet am 7. Juli 2011 im Kultur- & Kongresszentrum Liederhalle (KKL) in Stuttgart wieder das Java Forum Stuttgart. Es werden wie im Vorjahr rund 1.200 Teilnehmer erwartet. Geplant sind 42 Vorträge in sechs parallelen Tracks. Die Vorträge sind eingeteilt in Non-Sponsored Talks und Sponsored Talks und auch als solche im Vortragsprogramm gekennzeichnet. Zudem werden rund 30 Aussteller vor Ort sein. Es stehen noch Flächen zur Verfügung.

An der Community-Wand stehen sowohl offene White-Boards als auch BoF-Boards (Birds-of-a-Feather). Abends gibt es die Gelegenheit, sich bei verschiedenen BoF-Sessions mit Gleichgesinnten zu treffen, um über ein bestimmtes Thema zu diskutieren und sich auszutauschen. Darüber hinaus wird es wieder eine Jobbörse/Karriereecke für die Besucher geben.

Der Frühbucherpreis (bis einschließlich 11. Mai 2011) liegt bei 129 Euro, anschließend gilt der Normalpreis von 150 Euro.

Workshop „Java für Entscheider"

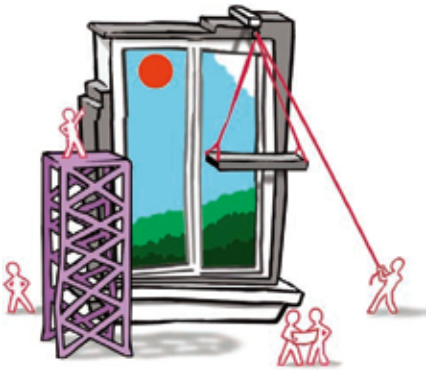
Die eintägige Überblicksveranstaltung am Vortag (6. Juli 2011) zeigt Begrifflichkeiten und wichtige Technologien aus der seit Jahren in der Industrie etablierten Plattform „Java". Ausgehend von strategischen Gesichtspunkten wie Bedeutung und Verbreitung reicht der Blick über das Client-seitige Java (Java SE) und die wesentlichen Entwicklungswerkzeuge bis zum Server-seitigen Java (Java EE). Dort stehen dann die Bedeutung von Java als Integrationsplattform und die verschiedenen Technologien im Mittelpunkt. Weiterhin wird noch der Einsatz von Java in den immer wichtiger werdenden mobilen Lösungen (Android, iOS) gestreift. Abschließend kommen noch das Ausrollen von Java-Lösungen und das sehr interessante Eclipse als Rich-Client zur Sprache, um dann den Bogen von der Software-Entwicklung hin zum Betrieb zu schlagen.

Experten-Forum-Stuttgart

Am 8. Juli 2011 findet wieder im Anschluss an das Java Forum Stuttgart ein Experten-Forum-Stuttgart statt. Auf dem Programm stehen zwölf halbtägige Workshops in sechs parallelen Tracks. Die Workshops in kleinen Gruppen mit maximal 25 Teilnehmern ermöglichen einen intensiven Austausch zwischen Trainer und Zuhörern.

Anmeldung und weitere Informationen:
www.java-forum-stuttgart.de

Google macht Eclipse ein Geschenk – WindowBuilder Pro



Jürgen Thierack

Im August 2010 kaufte Google die Firma Instantiations [1], die Tools für Entwickler produziert, darunter auch solche für die Java-Entwicklung mit Eclipse. Was den Entwickler früher einige tausend Dollar kostete, stellt Google frei zur Verfügung. Der Artikel stellt „WindowBuilder Pro" vor, einen GUI-Designer aus dem Gesamtpaket, von dem es heißt, es sei fünf Millionen Dollar wert.

Instantiations [1] existiert weiterhin als Spezialfirma für Smalltalk-Tools. Google hat die Java-bezogene Software ins eigene Portfolio übernommen [2]. Zwei der erworbenen Projekte wurden der Eclipse Foundation übergeben [3], wobei Google sogar weiterhin personelle Unterstützung gewährt. Beide Projekte sollen schon im nächsten Eclipse-Release-Train „Indigo" im Juni 2011 mitfahren und sind zuvor schon als Proposals angelegt.

- „WindowBuilder Pro" als WindowBuilder [4]
- „CodePro Profiler" als Runtime Analysis Tools (RAT) [5]

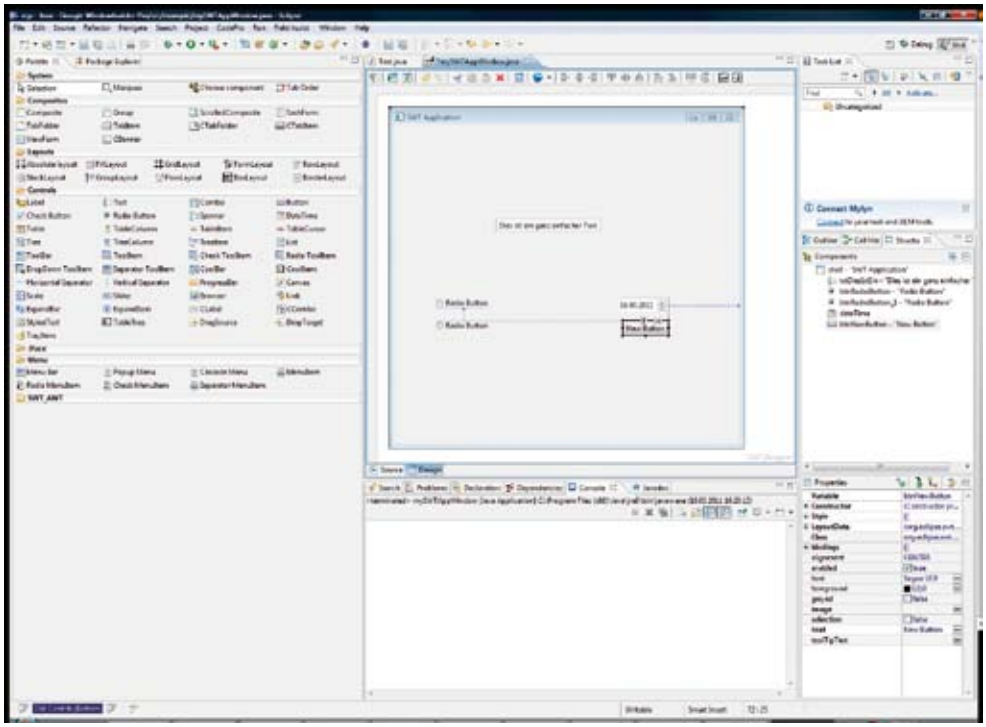


Abbildung 1: „WindowBuilder Pro" in der Eclipse Workbench

Während der „CodePro Profiler" im Vorfeld der Übergabe an Eclipse nicht verfügbar ist, sind „WindowBuilder Pro" und die andere übernommene Software direkt bei Google abzuholen. Es handelt sich um „WindowTester Pro" zur automatischen Erstellung von GUI-Tests, den „GWT Designer" (ein Java GUI Designer zur Erstellung von GWT GUI Applikationen (Ajax), der in den Google Web Toolkit übernommen wurde) und eine bemerkenswerte Tool-Sammlung unter dem Namen „CodePro AnalytiX". Die Sammlung umfasst sowohl eine visuelle Abhängigkeitsanalyse, als auch die schnelle Erstellung von JUnit-Tests sowie verschiedene Analysemethoden für Code.

WindowBuilder Pro

Das hat Eclipse noch gefehlt: Ein GUI-Designer – jedenfalls ein offizieller, denn anders als die NetBeans-IDE besitzt Eclipse keinen in der Grundausstattung. Es gibt allerdings eine Vielzahl von Lösungen, die von Drittanbietern kostenlos oder gegen Bezahlung angeboten werden. Eine (nicht ganz aktuelle) Übersicht findet sich bei [6]. In Abbildung 1 ähneln alle GUI-Designer einander: Von einer Palette der zur Verfügung stehenden Widgets (Controls) werden die benötigten auf das bearbeitete Fenster gezogen. Die besseren GUI-Designer, wie der hier besprochene, blenden simultan Hilfslinien ein.

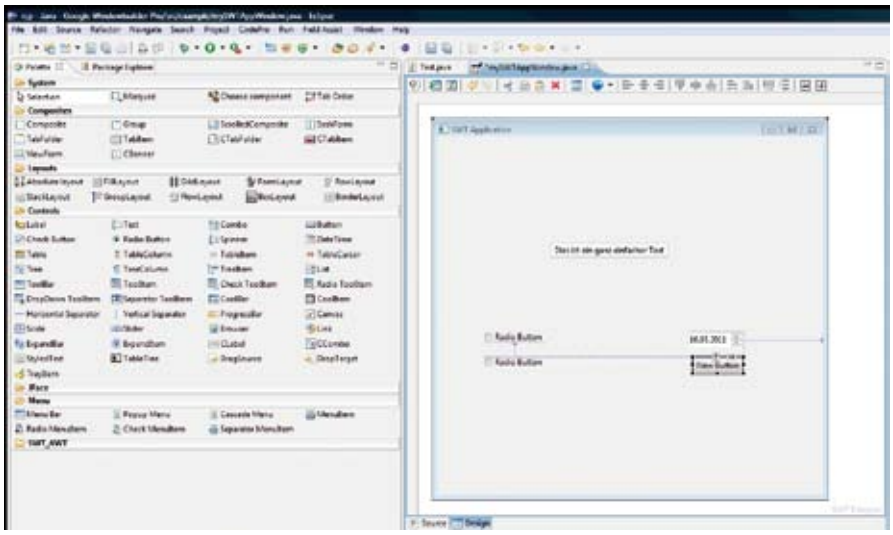


Abbildung 1: „WindowBuilder Pro" in der Eclipse Workbench

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TestRGB extends JApplet implements ActionListener
{
	/*
	 * RGBChooserComponent ist eine Komponente, welche mit drei Slidern die
	 *Auswahl eines Rot/Grün/Blau-Wertes ermöglicht.
	 * Der Code befindet sich in einer anderen Java-Datei.
	 */
	RGBChooserComponent rgb;

	public void init()
	{

		rgb = new RGBChooserComponent();
		getContentPane().add(rgb, BorderLayout.CENTER);

		/*
		 * Ein weiteres GUI-Element kommt hinzu.
		 */
		JButton button = new JButton(«Zufallsfarbe setzen»);

		button.addActionListener(this);
		getContentPane().add(button, BorderLayout.SOUTH);
	}

	public void actionPerformed(ActionEvent evt)
	{
		int r = (int) (Math.random() * 256) + 1;
		int g = (int) (Math.random() * 256) + 1;
		int b = (int) (Math.random() * 256) + 1;
		rgb.setColor(new Color(r, g, b));
	}

}
```

Listing 1

Bei Eclipse hat man hochfliegende Pläne mit dem WindowBuilder: „The scope of the project is to provide the definitive Eclipse extensible framework for creating GUI design tools for any language or UI framework. The current framework provides solid support for Java and XML based UI frameworks and exemplary tools for creating Swing and SWT UIs. Extending the framework to support additional languages (C++, JavaScript, PHP etc.) and UI toolkits in the future is highly desirable." [4]

Hier erfahren wir, dass nicht nur Eclipse-SWT-Oberflächen, sondern auch Swing und „XML based UI frameworks" [7] unterstützt werden. Doch der eigentliche Grund für die im Zitat ausgedrückte Begeisterung ist: Der WindowBuilder versteht Quellcode, kann also Quellcode parsen, eine Fähigkeit, die künftig auch bei anderen Sprachen als Java genutzt werden könnte.

Jede grafische Java-Oberfläche kann via Quellcode in den WindowBuilder eingelesen werden (in 90 bis 95 Prozent der Fälle ohne weitere Bemühungen) und erscheint da, als ob sie dort auch entworfen wäre. Das macht Installation [8] und Einarbeitung so lohnend, denn man kann auch seinen existierenden Bestand an Projekten damit bearbeiten.

Nach der Installation erscheint ein weiterer Editor für Java-Files, der die Datei im Text- und im Design-Modus darstellen kann. Jede Änderung in einem der beiden Modi ist dem anderen Modus sofort bekannt. Es gibt kein Repository und auch keine Trennung des Quellcodes in geschützte Bereiche und „Injection Points", wie man es von manch anderen GUI-Buildern kennt.

Ein kleines Java-Applet, das den WindowBuilder zuvor noch nie gesehen hat, zeigt das Reverse Engineering. Es besteht aus zwei Java-Files, dem gelisteten und einem mit der Slider- und Anzeigekomponente („rgb"), (siehe Listing 1).

Abbildung 2 zeigt, wie man den RGB-Wert über die drei Slider festlegen oder per Button einen Zufallswert erzeugen kann. In Abbildung 3 ist das obere Listing im WindowBuilder geöffnet. Obgleich nur der Button im File definiert ist, wird auch der GUI-Teil der zweiten Java-Datei für die Komponente „rgb" richtig dargestellt. GUI-Elemente, Slider, Label und die Anzeigefläche lassen sich auch visuell in der Komponenten-Struktur darstellen (siehe Abbildung 4).

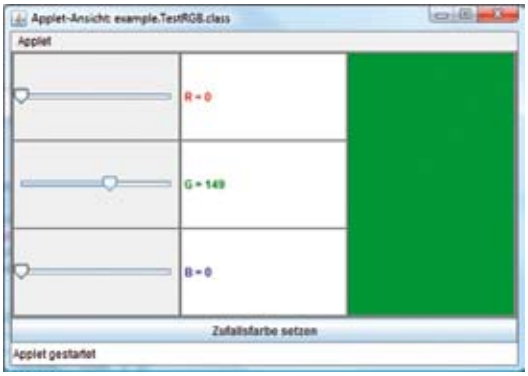


Abbildung 2: Ein Applet zur Auswahl eines Rot/Grün/Blau-Werts

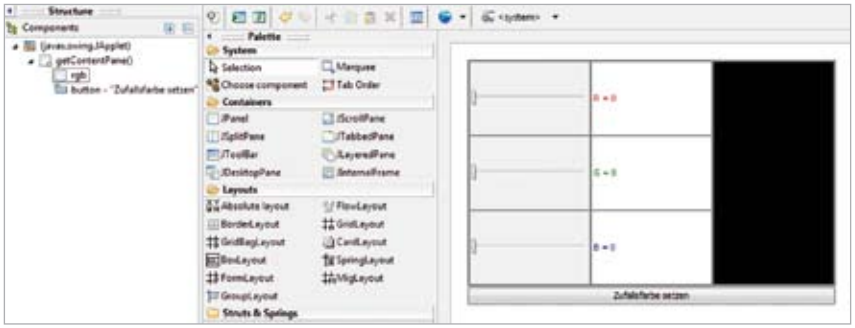


Abbildung 3: Das Applet im WindowBuilder (Ausschnitt)

Abbildung 4: Die Komponente „rgb" im WindowBuilder (Ausschnitt)

Weitere Informationen

[1] http://www.instantiations.com
[2] http://code.google.com/intl/de/javadevtools/index.html
[3] http://www.heise.de/developer/meldung/Googles-WindowBuilder-und-CodePro-Profiler-werden-Eclipse-Projekte-1153131.htm
[4] http://www.eclipse.org/proposals/tools.windowbuilder
[5] http://www.eclipse.org/proposals/tools.rat
[6] Peter Friese, Schöner entwickeln, Vergleichstest aktueller SWT GUI Builder, Eclipse Magazin, 1.1. 2006
[7] http://wiki.eclipse.org/E4/XWT
[8] http://code.google.com/intl/de-DE/javadevtools/wbpro/index.html

Jürgen Thierack ist Diplomchemiker, wechselte aber in die Software-Branche, wo er freiberuflich unterwegs ist. Seit fast 10 Jahren liegt sein inhaltlicher Schwerpunkt bei Fragen der Finanzmathematik, darunter der Preisfindung für Optionen und Optionsscheine. Aktuelle Thematik: Trendfolgesysteme. Mit Java arbeitet er seit 2001.

Kontakt:
Jürgen Thierack
thierack@igfm-muenchen.de

Scala – Polyglott-Programmierung mit Java

Dario Giancola, Dr. Christine Müller und Dr. Normen Müller, BSgroup Technology Innovation AG

Scala vereint objektorientierte und funktionale Programmier-Paradigmen und bietet dabei neue Wege, komplexe Anwendungen mit hoher Qualität stabil und performant zu entwickeln. Scala wird zunehmend interessanter für die Java-Gemeinschaft aufgrund der vollen Kompatibilität zu Java und der somit leichten Integration in bestehende Java-Anwendungen. Dieser Artikel zeigt, wie einfach Scala verwendet werden kann, um ausgewählte Komponenten in bestehenden Java-EE-Lösungen neu zu realisieren und zu erweitern.

Scala ist eine funktionale und objektorientierte Programmiersprache, die unter anderem auf der Java Virtual Machine (JVM) ausgeführt werden kann. Dabei schöpft Scala die Möglichkeiten der JVM besser aus als Java und läuft somit performanter. Allein die Verwendung des Scala-Compilers stellt schon einen Effizienzgewinn dar. Ein Grundsatz der Sprache ist das Streben nach Kürze und Prägnanz, ohne dadurch die Lesbarkeit und Verständlichkeit von Programmen zu beeinträchtigen. Im Gegensatz zu Java sind die Sprachkonstrukte nicht auf spezielle Anwendungsfälle zugeschnitten, sondern als sogenannte „library abstractions" (Bibliotheksfunktionen) modelliert und somit leicht erweiterbar und anpassbar. Dadurch wird Scala zu einer „**sca**lable **la**nguage".

Ein zentrales Argument für Scala ist die Möglichkeit, dynamisch kontrolliert in einer statisch typisierten Sprache zu programmieren. Scala bietet Programmierern volle Flexibilität, etwa durch „pattern matching" (Mustererkennung) und andere funktionale Konstrukte wie „higher-order functions" (Funktionen höherer Ordnung), vermeidet dabei jedoch einen Überbau durch Typ-Inferenz. Scala fühlt sich somit wie eine dynamisch kontrollierte Sprache an, in der die Vorteile von statischer Typisierung – frühzeitige Fehlererkennung, sichere Refaktorisierung, starke Abstraktionsmöglichkeiten, inhärente Dokumentation und automatisierte Effizienzoptimierung – besser zur Geltung kommen.

Scala ist voll kompatibel zu Java und subsumiert somit Vorteile wie große Akzeptanz, weite Verbreitung und eine Vielzahl bestehender Bibliotheken. Umgekehrt kann Java direkt die Vorteile von Scala nutzen: starke Typsicherheit, einfache Erweiterbarkeit und unkomplizierte Nebenläufigkeit.

Dieser Artikel zeigt, wie Scala eine bestehende Java-Enterprise-Lösung erweitert. Dabei werden auch die Vorteile einer Re-Implementierung ausgewählter Java-Komponenten in Scala aufgezeigt: Neben der Steigerung von Qualität, Performanz und Stabilität verhilft eine Integration mit Scala auch zu einer gesteigerten Übersichtlichkeit und somit besseren Wartbarkeit, Erweiterbarkeit und Lesbarkeit.

Erweiterung einer Java-EE-Anwendung

Für komplexe und interaktive Client-Server-Anwendungen hat sich Java EE in weiten Teilen der Geschäftslogik als Standard etabliert und ist für viele Firmen eine wichtige Plattform für die Integration bestehender und neuer Dienstleistungen. Die folgenden Anwendungsbeispiele erläutern die Integration von Scala 2.8.1 in eine Java-EE-5-Umgebung. Zu Illustrationszwecken sind die Beispiele einfach gehalten und die

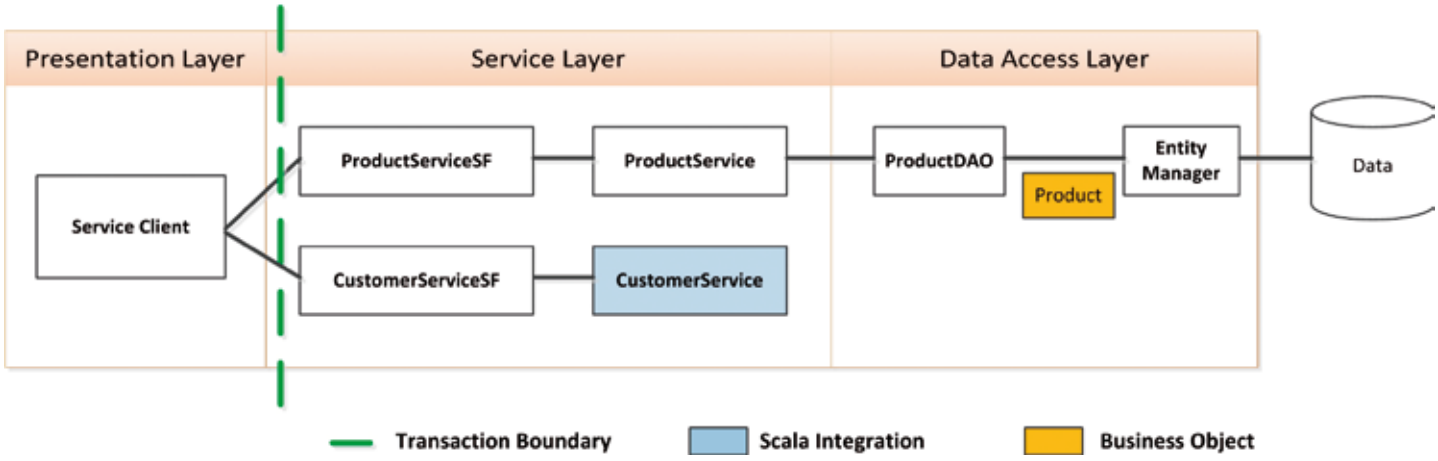


Abbildung 1: Erweiterung eines Services mit Scala

Integrationsaspekte in den Vordergrund gestellt.

Abbildung 1 zeigt eine typische Java-EE-Architektur. Die zur Verfügung gestellten Services werden durch ein Java Interface (ProductService) und einen Scala Trait (CustomerService) beschrieben.

```
import javax.ejb._

@Stateless @LocalBean
class CustomerServiceImpl extends CustomerService

trait CustomerService {
  def checkUser(fName: String, lName: String) = {
    case class Customer(fName: String, lName: String)
    Customer(fName, lName) match {
      case Customer("Dario","Giancola") => "Engineer"
           case Customer(_,"Müller")      => "Architect"
           case Customer(_, _)            => "Employee"
    }
  }
}
```

Listing 1: CustomerServices.scala

Listing 1 veranschaulicht die Realisierung des CustomerService-Traits durch die „CustomerServiceImpl"-Session-Bean. Ein Trait ist so ähnlich wie ein Java-Interface. Im Gegensatz zu Interfaces können Traits jedoch Methoden-Implementierungen enthalten und ermöglichen somit eine bessere Modularisierung. Durch einen intelligenten Linearisierungsmechanismus kann man mit Traits – unter Vermeidung des Diamanten-Problems – Mehrfachvererbung realisieren.

Zur Definition der „CustomerServiceImpl"-Klasse lassen sich alle Java-EE-5-Annotationen verwenden. So kann eine Scala-Klasse beispielsweise durch die Annotation „@Stateless" als Session-Bean gekennzeichnet sein. Der einzige Unterschied zu einer Java-Bean besteht in der Definition einer lokalen Bean: Hierfür muss man in Scala die Annotation „@LocalBean" (anstelle „@Local") verwenden. Die Annotationen werden analog zu Java mittels „import"-Klauseln eingebunden. Der einzige Unterschied ist, dass Scalas „on-demand import"-Klausel mit einem Unterstrich (_) gekennzeichnet ist statt mit einem Stern (*).

Während in Java öffentliche Klassen in eine gleichnamige Datei platziert werden müssen, können in Scala Dateien beliebig benannt werden sowie ein oder mehrere Artefakte enthalten. Scala stellt somit neben dem „package"-Konstrukt eine weitere Gruppierungsmöglichkeit zur Verfügung, mit der semantische Abhängigkeiten fein-granular, leicht ersichtlich und explizit modelliert werden können – ein entscheidender Schritt zur Erleichterung von Wartbarkeit und Erweiterbarkeit. Zur Veranschaulichung und aufgrund der „extends"-Abhängigkeit der „CustomerServiceImpl"-Bean zu ihrer Schnittstellenbeschreibung „CustomerService" sind beide Artefakte in der „CustomerServices.scala"-Datei definiert.

Methoden- und Funktionsdefinitionen beginnen in Scala mit „def". Dem Funktionsnamen, beispielsweise „checkUser" in Listing 1, folgt eingeschlossen in runden Klammern eine Komma-getrennte Parameterliste. Jeder Parameter muss mit einem Typ annotiert sein, wobei der Parameter-Name und die Typ-Annotation durch einen Doppelpunkt voneinander getrennt sind (fName: String). Nach der schließenden Klammer der Parameterliste folgt die Annotation des Ergebnistyps der Funktion selbst. Der Scala-Compiler verlangt die Annotation des Ergebnistyps jedoch lediglich in der Definition von rekursiven Funktionen. Im Fall von „checkUser" kann diese Annotation ausgelassen werden. Der entsprechende Typ wird vom Compiler eingesetzt. Nach dem Ergebnistyp (beziehungsweise der Parameterliste) folgen ein Gleichheitszeichen und zwei geschweifte Klammern, die den Rumpf der Funktion einschließen. Das Gleichheitszeichen ist ein Hinweis auf den funktionalen Anteil von Scala: Eine Funktion ist ein Ausdruck, welcher zu einem Wert evaluiert. Somit ist in Scala eine explizite „return"-Anweisung optional. Wenn eine Funktion aus einer einzigen Anweisung besteht, kann man die geschweiften Klammern weglassen.

Im Rumpf verwendet „checkUser" Mustererkennung. Scala bietet einen generischen Pattern-Matching-Mechanismus, der in seiner einfachsten Form ähnlich einer „switch"-Anweisung in Java ist. Im Gegensatz zu dem sehr rudimentären Ansatz in Java können in Scala beliebige Ausdrücke verwendet werden. Scalas Pattern-Matching-Ansatz ist vor allem bei der Mustererkennung auf algebraischen Datentypen – dargestellt durch „case classes" – wertvoll.

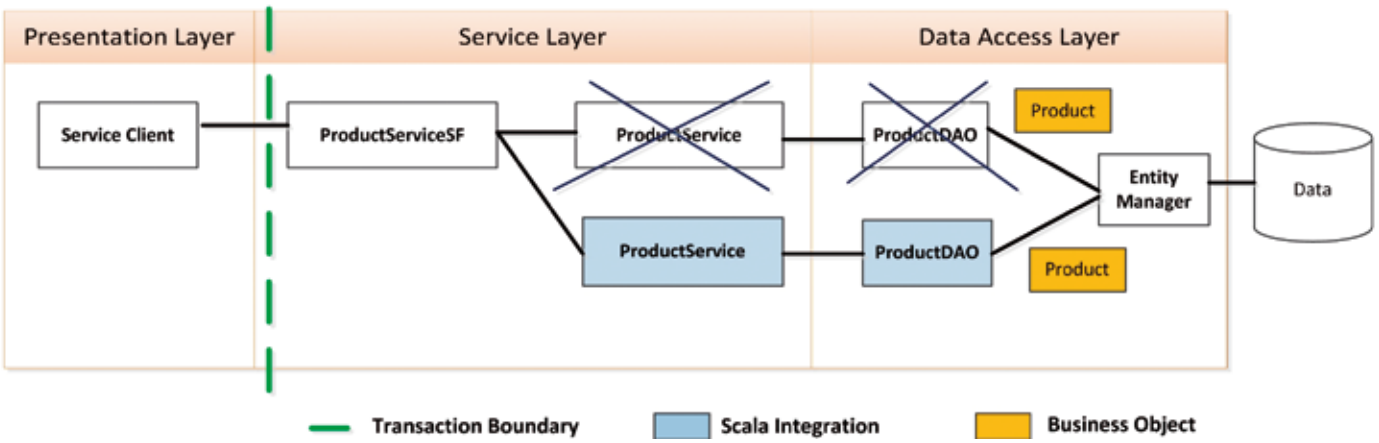


Abbildung 2: Re-Implementierung eines bestehenden Java-Services in Scala

Anhand des „CustomerServiceImpl"-Beispiels sind einige Vorteile von Scala verdeutlicht: Performanz und Lesbarkeit durch einen mächtigen Pattern-Matching-Mechanismus, neue Modularisierungsmöglichkeiten mittels Traits, Übersichtlichkeit durch das Zusammenfügen mehrerer struktureller Scala-Komponenten (Klassen, Traits etc.) in einer Datei und Kompaktheit durch Typ-Inferenz. Insbesondere Letzteres ermöglicht das Gefühl, in einer dynamisch kontrollierten Sprache mit statischer Typsicherheit zu programmieren und die Vorteile von Skriptsprachen (weniger Schreibarbeit) zu nutzen.

Re-Implementierung von Java-Komponenten

Um weitere funktionale Konzepte von Scala und deren Vorteile aufzuzeigen, wird im Folgenden die Re-Implementation der „ProductServiceImpl"-Enterprise-Bean erläutert (siehe Abbildung 2 und Listing 2).

In der Realisierung der„ProductService"-Bean (siehe Listing 2) wird ein Vorteil von Scala hinsichtlich „import"-Anweisungen deutlich: Diese können überall stehen. Man kann somit explizit aufzeigen, ab welcher Stelle im Code die importierten Artefakte benötigt werden.

Durch die Unifizierung des objektorientierten und funktionalen Paradigmas existieren in Scala zwei Arten von Platzhaltern: veränderliche und unveränderliche. Variablen sind mit „var" gekennzeichnet und Konstanten mit „val" (analog zu „final" in Java). Variablen können mit einem „default"-Platzhalter ('_') initialisiert sein („null" für Referenztypen, „false" für Boolean und die entsprechende Version von „0" für numerische Werte).

Eine weitere Besonderheit von Scala hinsichtlich Funktionsdefinitionen ist die optionale Deklaration von Parameterlisten. Die Funktionen „getPrices" und „getMaxPrice" werden zum Beispiel schlicht durch ihren Namen aufgerufen, das heißt, man ruft parameterlose Methoden wie Felder auf. Dieser einheitliche Zugriff auf Felder und parameterlose Methoden bietet dem Entwickler eine enorme Flexibilität in der Erstellung von Klassen-Definitionen. Der Unterschied zwischen Feldern und parameterlosen Methoden liegt in der jeweiligen Definition: Die rechte Seite eines Feldes wird in der Initialisierung des entsprechenden Objekts (nur einmal) ausgewertet. Die rechte Seite einer parameterlosen Methode hingegen wird bei jedem Aufruf der Methode ausgewertet.

```
implicit object ProductOrd extends Ordering[Product] {
  def compare(x: Product, y: Product) = x.getPrice.compareTo(y.getPrice)
}

@Stateless @LocalBean
class ProductServiceImpl extends ProductService {
  protected def ordered(pList: Seq[Product]) = pList.sorted
  protected def mostExpensive(pList: Seq[Product]) = pList.max
}

trait ProductService {
  @EJB
  private var productDAO:ProductDAO = _

  import scala.collection.JavaConversions._

  def getPrices = ordered(productDAO.getActiveProducts())
  def getMaxPrice = mostExpensive(productDAO.getActiveProducts())

  protected def ordered(pList: Seq[Product]): Seq[Product]
  protected def mostExpensive(pList: Seq[Product]): Product
}
```

Listing 2: ProductServices.scala

In der Erstellung von Klassen und dem damit verbundenem Design der Zugriffsrichtlinien bietet Scala analog zu Java die Einschränkungen „private" und „protected". Im Wesentlichen behandelt Scala diese Zugriffseinschränkungen wie Java. Es gibt einige wichtige Unterschiede, die jedoch außerhalb des Umfangs dieses Artikels liegen. Scalas Zugriffseinschränkung „default" ist „public".

Ein Aspekt, in dem sich Scala klar von Java abhebt – insbesondere hinsichtlich des objektorientierten Paradigmas – ist, dass Scala-Klassen keine statischen Felder oder Methoden enthalten können. Stattdessen besitzt Scala sogenannte „Singleton-Objekte". Eine Singleton-Definition entspricht im Wesentlichen einer Klassen-Definition – anstelle des Schlüsselworts „class" wird allerdings „object" verwendet. Ein „object" definiert ausschließlich ein einzelnes, eindeutiges Objekt. Es ist nicht möglich, mit „new" weitere Objekte gleicher Struktur zu erstellen. Somit fehlen einer „object"-Definition die Konstruktor-Parameter.

Die in Listing 2 verwendeten Scala-Methoden „sorted" und „max" sowie das Objekt „ProductOrd" demonstrieren eine weitere Besonderheit von Scala: implizite Parameter. Hierfür stellt Scala das Schlüsselwort „implicit" zur Verfügung, das am Anfang einer Parameter-Liste verwendet wird und alle darauffolgenden Parameter als „implizit" kennzeichnet. Die folgenden (vereinfachten) Scala-Methoden haben jeweils einen impliziten Parameter:

```
def sort(implicit ord: Ordering[T]): List[T]
def max(implicit comp: Ordering[T]):T
def sum(l: List[T])(implicit adder:Adder[T]):T
```

Diese Beispiele zeigen, dass es möglich ist, normale und implizite Parameter zu kombinieren. Allerdings kann jeweils nur eine implizite Parameterliste pro Methode oder Konstruktor definiert werden. Diese muss immer zuletzt aufgeführt sein. Der Grundgedanke hinter „implicit" ist, dass Argumente für Methoden explizit ausgelassen werden können und implizit durch den Scala-Compiler inferiert werden. Im gezeigten Beispiel inferiert der Scala-Compiler für die Methoden „sort" und „max" das als implizit deklarierte Objekt „Product-

Ord". Diese Eigenschaft vereinfacht das Design vieler Konstrukte und erhöht deren Flexibilität. So lässt sich beispielsweise die Repräsentation der Ordnung auf Product-Instanzen einfach austauschen, ohne darauf aufbauende Strukturen explizit anpassen zu müssen.

Als Letztes sind für das„ProductService"-Beispiel implizite Konvertierungen zu nennen. Auch hierfür wird das Schlüsselwort „implicit" verwendet. Zur Erläuterung stelle man sich einen Ausdruck „e" vom Typ „t" (e: t) vor, der jedoch als Typ „s" erwartet wird. Typ „t" ist nicht konform zu „s". Bevor der Scala-Compiler einen Compile-Fehler ausgibt, versucht er – als letzten Ausweg – eine implizite Konvertierung (i(e)) anzuwenden, die den Typ „e" Typ-konform zu „s" überführt. In Listing 2 wird für„productDAO.getActiveProducts()"eine solche implizite Konvertierung angewandt, die„java.util.List" in„scala.collection.Seq" überführt. Diese Konvertierung ist implizit durch das „scala.collection.JavaConversions"-Objekt sichergestellt, das eine Ansammlung von diversen Konvertierungsmethoden zwischen Java und Scala zur Verfügung stellt.

Anhand der Beispiele wurden weitere Unterschiede zwischen Scala und Java aufgezeigt: die „import"-Klauseln, die Schlüsselwörter „var" und „val", die Initialisierung von Objekten, optionale Parameterlisten oder auch die Notation für Singleton-Objekte. Dennoch verläuft die Integration von Scala-Komponenten in eine bestehende Anwendung nahtlos und einfach. Ein entscheidendes Werkzeug hierbei sind die verschiedenen Konvertierungsmethoden, die mit der Scala-Bibliothek bereitgestellt sind.

Fazit

Scala ist eine generell einsetzbare Programmiersprache mit dem Augenmerk, gängige Programmiermuster prägnant, elegant und typsicher modellieren zu können. Durch die Vereinigung objektorientierter und funktionaler Eigenschaften steigert Scala die Produktivität von Entwicklern. Einen entscheidenden Sachverhalt für den Erfolg von Scala ist die volle Interoperabilität mit Java: Java-Entwickler sind nicht gezwungen, bestehenden Java-Code komplett in Scala umzuschreiben.

Der Umstieg zu Scala ist leicht und iterativ möglich, weil bestehende OO-Kenntnisse weiterhin anwendbar sind. Konstrukte wie Pattern-Matching, Traits und implizite Konvertierungen beschleunigen den Arbeitsprozess. Weitere funktionale Konzepte ergänzen die Java-Welt und ermöglichen es, das volle Potenzial von Scala auszuschöpfen. Scala überzeugt somit nicht nur als primäre Sprache. Auch als zusätzliches Werkzeug kann man Scala im Zusammenspiel mit Java EE erfolgreich einsetzen.

Kontakt:
Dario Giancola
dario.giancola@bsgroup.ch
Christine Müller
christine.mueller@bsgroup.ch
Normen Müller
normen.mueller@bsgroup.ch

Dr. Christine Müller ist Software Ingenieur bei der Technology Innovation AG. Sie verfügt über weitreichende theoretische und praktische Erfahrung mit objektorientierten sowie funktionalen Sprachen.

Dario Giancola ist Senior Software Engineer bei der BSgroup Technology Innovation AG. Er ist seit mehreren Jahren im Java-EE-Umfeld tätig. Seine derzeitigen Schwerpunkte liegen auf Java-EE- und Spring-Anwendungen sowie Rich-Internet-Applikationen mit Adobe Flex. Neben dem Projektgeschäft gilt sein besonderes Interesse Scala.

Dr. Normen Müller ist Senior Software Ingenieur bei der Technology Innovation AG. Er verfügt über mehrjährige Projekt- und Programmiererfahrung mit Scala. Bei Technology Innovation AG ist er verantwortlich für Scala-Schulungen sowie den verstärkten Einsatz der Sprache in verschiedenen Projekten.

Vorschau

Java aktuell – Herbst 2011

Die nächste Ausgabe der Java aktuell erscheint am 1. September 2011. Bitte schreiben Sie uns an redaktion@ijug.eu, was Ihnen an dieser Zeitschrift gefallen hat und welche Inhalte Sie sich für kommende Ausgaben wünschen.

Falls Sie selbst einen Artikel in der nächsten Java aktuell veröffentlichen möchten, schicken Sie bitte vorab Ihren Themenvorschlag an redaktion@ijug.eu, Redaktionsschluss ist am 8. Juli 2011.

Single Sourcing in Java: Desktop-Anwendung und Web-Applikation aus einer Quelle

Björn Christoph Fischer & Oliver Zandner, Triestram & Partner GmbH

Rich Internet Applications wie etwa Google Maps sind aus dem Internet nicht mehr wegzudenken. Sie zeichnen sich durch ihre hohe Interaktivität aus und sind überall dort verfügbar, wo ein Internet-Zugang und ein Browser vorhanden sind. Nutzer erwarten diese Merkmale deshalb auch zunehmend von ihren Business-Anwendungen. Diese sind zwar komfortabel zu bedienen, aber nicht mobil, denn sie sind an das Endgerät gebunden, auf dem sie installiert sind.

Wie lassen sich die beiden genannten Anforderungen nach hohem Bedienkomfort und nach Mobilität zusammenbringen? Eine Möglichkeit wäre, zwei separate Programme zu entwickeln – eine Desktop- und eine Web-Applikation. Das ist jedoch wirtschaftlich nicht sinnvoll, da etwa doppelter Aufwand und damit doppelte Kosten anfallen. Eine Lösung bietet das Single Sourcing mit Eclipse RCP und RAP.

Ausgangslage im Projekt

Die oben genannten Anforderungen stammen aus der Produkt-Entwicklung: Das Unternehmen der Autoren entwickelt seit 1991 lisa.lims – ein Labor-Informations- und Managementsystem. Diese basiert auf der Oracle-Datenbank sowie Oracle Forms und Reports. Forms läuft seit der Version 6*i* im Browser, und zwar als Java-Applet. Das ist jedoch an Voraussetzungen gebunden:

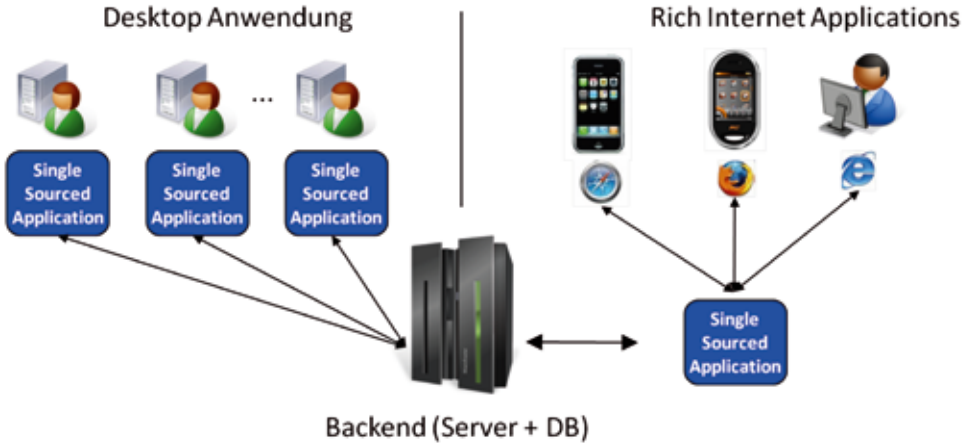


Abbildung 2: Single Sourcing – Ziel

Es ist das Java-Plug-in für den Browser erforderlich und es müssen die Oracle-spezifischen Java-Bibliotheken vorhanden sein. Damit ist die Mobilität des Benutzers eingeschränkt.

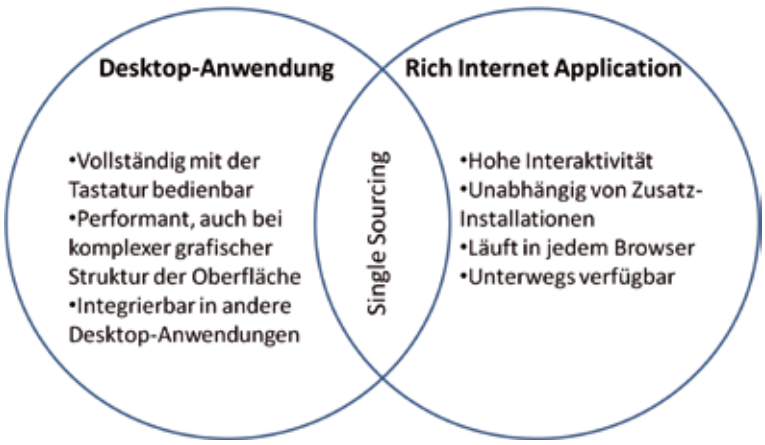


Abbildung 1: Single Sourcing – Funktionale Sicht

Die Erwartungen neuer Nutzer an lisa.lims sind geprägt durch die eingangs beschriebene Situation: Bestimmte Gruppen von Anwendern sind mobil außerhalb ihres Büros tätig; sie erfassen Daten auf mobilen Endgeräten und werten sie auch dort aus. Andere Anwender haben einen festen Arbeitsplatz und erwarten vor allem hohen Bedienkomfort – zum Beispiel derart, dass die Anwendung komplett über Funktionstasten bzw. Tastatur-Kürzel bedienbar ist. Um beide Anforderungen wirtschaftlich und effizient erfüllen zu können, sollten sie aus einer Code-Basis bedient werden. Daher basiert lisa.lims mit der Version 10 auf dem Single-Sourcing-Ansatz.

Single Sourcing

Das Ziel von Single Sourcing ist es, die klassische Desktop-Anwendung und die Rich Internet Application aus derselben Code-Basis zu bedienen, um Kosten zu senken

und die Wartbarkeit der Applikation zu erhöhen (siehe Abbildung 2).

Single Sourcing mit Eclipse

Als Entwicklungs- und Laufzeitumgebung für das genannte Projekt wurde Eclipse gewählt. Eclipse stellt zur Entwicklung von grafischen Benutzeroberflächen für Desktop-Anwendungen die sogenannte „Rich Client Platform" (RCP) zur Verfügung. Die grafische Benutzeroberfläche der Anwendung wird aus Komponenten des Standard-Widget-Toolkits (SWT) zusammengesetzt – eine Bibliothek mit Oberflächen-Elementen wie Button, TextField etc.

Bemerkenswert hierbei ist, dass die einzelnen Komponenten mit den grafischen Bord-Mitteln des jeweiligen Betriebssystems dargestellt werden, auf dem die Anwendung läuft. Die strenge Abgrenzung von Betriebssystem und SWT gegenüber den höheren Schichten der Anwendung ermöglicht, Betriebssystem und SWT durch entsprechende Web-Schichten zu ersetzen und somit Single Sourcing zu betreiben (siehe Abbildung 3).

Von der Eclipse Rich Client Platform zur Rich Ajax Platform

Mittels der sogenannten „Rich Ajax Platform" (RAP) wird aus der RCP-Desktop-Anwendung eine Rich Internet Application: Im Modell (siehe Abbildung 3) werden die unteren beiden Schichten ausgetauscht – und zwar das SWT gegen das Rap Widget Toolkit (RWT) und das Betriebssystem gegen einen Servlet-Container, da es sich ja um eine Web-Anwendung handelt (siehe Abbildung 4).

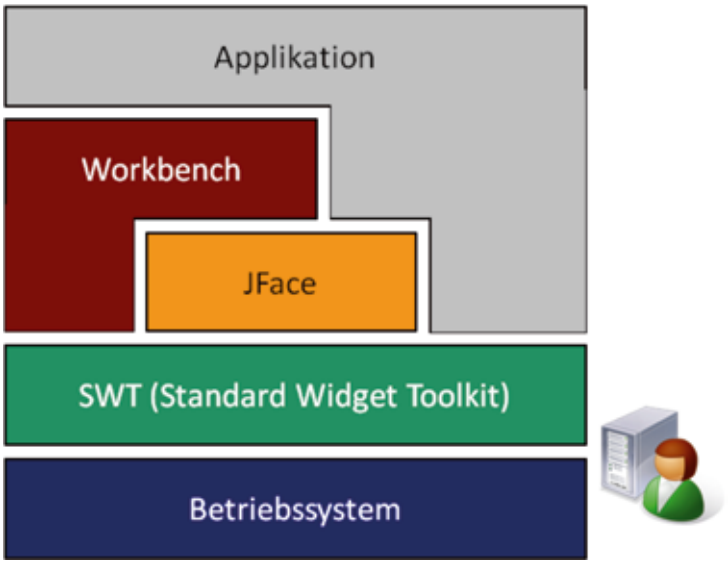


Abbildung 3: Eclipse Rich Client Platform (RCP) und Standard Widget Toolkit (SWT)

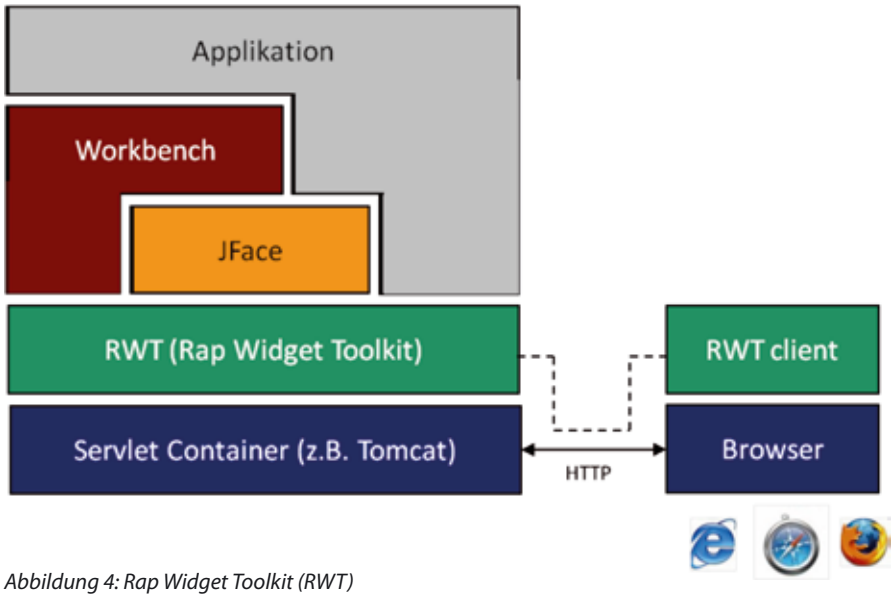


Abbildung 4: Rap Widget Toolkit (RWT)

RWT ist ebenfalls eine Komponenten-Bibliothek zum Erstellen grafischer Benutzeroberflächen – allerdings auf Basis von Servlets, JavaScript und einer Ajax-Kommunikation zwischen Browser und Anwendung. RWT hat zwei wichtige Besonderheiten:

1. Seine Schnittstellen (APIs) sind kompatibel mit denen von SWT. Das bedeutet: Der Quellcode einer SWT-basierten Applikation muss nicht geändert werden, wenn SWT gegen RWT ausgetauscht wird. Diese Eigenschaft ermöglicht das Single-Sourcing-Verfahren. Desktop-Anwendung und Rich Internet Application basieren auf demselben Quellcode, nutzen jedoch unterschiedliche grafische Bibliotheken (RWT vs. SWT), deren Schnittstellen identisch sind.
2. RWT basiert auf der Ajax-Technologie. Das Besondere daran ist: Wenn der Benutzer beispielsweise auf einen Button klickt, können als Reaktion auf dieses Ereignis ein oder mehrere Teile der Web-Seite im Browser aktualisiert werden, und nicht nur die komplette Seite. Dies macht einen großen Teil der Interaktivität und des Bedienkomforts von Rich Internet Applications aus. Eine solche Internet-Seite erreicht annähernd den Bedienkomfort einer Desktop-Anwendung.

Die Grenzen des Single Sourcing

Bei einer Java-Desktop-Applikation arbeitet jeder Nutzer mit seiner eigenen Instanz des Programms, die im Speicher seines lokalen Rechners läuft. Im Gegensatz dazu läuft bei einer Web-Anwendung lediglich genau eine Instanz der Anwendung im Speicher des zentralen Web-Servers. Das

bedeutet, dass sich alle Anwender der Web-Anwendung diese eine Programm-Instanz teilen (Multiuser-Umfeld). Daraus resultieren folgende Grenzen des Single Sourcing:

- *Singletons (Einzelstücke)*
  Ein Singleton ist eine Klasse, von der in einer Anwendung genau eine Instanz existiert. Wird dieses Entwurfsmuster in der Desktop-Anwendung genutzt, muss geprüft werden, ob das auch für die Web-Anwendung adäquat ist. Nicht adäquat wäre es zum Beispiel, wenn sich in der Web-Anwendung alle Benutzer dasselbe Singleton-Warenkorb-Objekt teilen.
- *Maus-Tracking*
  Maus-Tracking bedeutet, dass die Anwendung auf jede Bewegung reagiert, die der Benutzer mit der Maus vollzieht. So entsteht hohe Interaktivität in einer Desktop-Anwendung. Bei einer browserbasierten Anwendung würde jedoch die Übermittlung jeder Mausbewegung an den Server zu einer immensen Netzwerklast und vermutlich schnell zur Auslastung der Verbindung führen. Deshalb besitzt die Rich Ajax Platform keinerlei Möglichkeit, auf Mausbewegungen zu reagieren. Jedoch könnte eine Anwendung das Maus-Tracking clientseitig (im Browser per JavaScript) behandeln, was jedoch das Single Sourcing erschwert. Davon nicht betroffen sind Maus-Klick-Events. Diese verarbeitet das RAP und sie sind deshalb in der Anwendung möglich.
- *Desktop-Integration (etwa der Dialog „Datei öffnen / speichern")*
  In den meisten Betriebssystemen existiert ein Dialog zur Auswahl von Dateien (zum Öffnen beziehungsweise Speichern). Im Web ist jedoch nicht allgemein festlegbar, ob die Datei auf der Client- oder auf der Server-Seite geöffnet oder gespeichert werden soll beziehungsweise ob dem Benutzer diese Möglichkeit aus Sicherheitsgründen überhaupt zur Verfügung stehen darf. Deshalb verzichtet die Rich Ajax Platform auf einen solchen Dialog, der im Desktop-Bereich zur Standardausstattung gehört. Es gibt jedoch Lösungen (etwa das sogenannte „Upload-Widget"), um konkrete Anforderungen umzusetzen. Diese erfordern jedoch beim Single Sourcing besondere Aufmerksamkeit.
- *Key Bindings*
  Als „Key Binding" bezeichnet man die Verknüpfung einer Programmaktion mit einer bestimmten Tastenkombination. Drückt der Benutzer die definierte Tastenkombination, wird die entsprechend gebundene Aktion ausgelöst. Im Gegensatz zur Desktop-Welt ist es mit JavaScript nicht möglich, sämtliche Tastenkombinationen für die Anwendung zu verwenden, da diese möglicherweise vom Browser belegt sind (beispielsweise F5 zur Aktualisierung der Webseite oder F1 zum Aufruf der Browser-Hilfe). Diese Einschränkungen sind beim Single Sourcing zu berücksichtigen.

Wiederverwendungsrate im Projekt

In dem beschriebenen Projekt ging es unter anderem darum, auf Basis des Single-Sourcing-Verfahrens ein Framework zu schaffen, um die Standard-Software lisa.lims in Richtung Java und Open Source zu modernisieren. Dazu wurde eine Code-Basis von derzeit rund 24.000 Java-Code-Zeilen erstellt. Diese verteilen sich wie folgt:

- 22.750 im Bereich „Common"
- 460 im Bereich „RCP"
- 820 im Bereich „RAP"

Damit ergibt sich eine Wiederverwendungsrate von ansehnlichen 95 Prozent (siehe Abbildung 5).

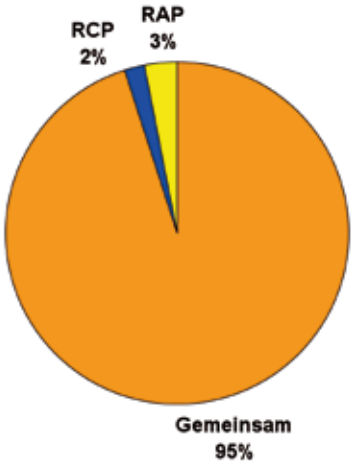


Abbildung 5: Wiederverwendungsrate

Fazit

Die Erfahrungen der Autoren sind die folgenden:

- Single Sourcing ist praktikabel. Auch bei einer umfangreichen und komplexen Anwendung wie lisa.lims.
- Single Sourcing ist wirtschaftlich. Denn bei einer zweigleisigen Entwicklung von Desktop-Anwendung und Rich Internet Application reduziert es den Entwicklungsaufwand erheblich – dank einer gemeinsamen Code-Basis von rund 95 Prozent.
- Single Sourcing ist schlank beziehungsweise leichtgewichtig. Die gewählten Basis-Technologien wie RWT und SWT basieren allesamt auf der Java Standard Edition (JSE). Damit ist als Laufzeitumgebung ein Servlet-Container ausreichend. Das macht die Anwendung unabhängig von den Hersteller-Spezifika der am Markt zurzeit verfügbaren Application Server, die für eine Java-Enterprise-Architektur erforderlich wären.
- Single Sourcing ist offen. Weitere Java-Bibliotheken lassen sich einfach integrieren, beispielsweise JFreeChart zur interaktiven Visualisierung von Messkurven. Ebenso einfach ist die Anbindung von Datenbanken.
- Die Wahl von Eclipse RCP / RAP fiel im Rahmen der Migration der Standard-Software lisa.lims in Richtung Java. Dieser Entschluss legte die Verwendung von OSGi auch für die Serverseite nahe, um eine Durchgängigkeit der Basistechnologien zu gewährleisten. Zudem sind wichtige Bibliotheken ebenfalls in Form von OSGi-Bundles verfügbar – so etwa der OR-Mapper EclipseLink für den Datenbankzugriff sowie das Spring Framework für das Remoting. Durch diese Technologien ist es möglich, eine leichtgewichtige Server-Komponente zu entwickeln, die alle für den Umstieg von Oracle Forms notwendigen Funktionalitäten bietet, wie zum Beispiel die Wiederverwendung von PL/SQL-Packages in der Datenbank, pessimistisches Sperren von Datensätzen und die dedizierte

Benutzer-Anmeldung mit eigener Session an der Datenbank (siehe „Smart Migration: schrittweise Umstellung von Oracle- auf Java-Open-Source-Technologien“ von Thomas Haskes und Oliver Zandner in der DOAG *News* 02/2011).

Kontakt:
Björn Christoph Fischer
b.fischer@t-p.com
Oliver Zandner
o.zandner@t-p.com

Oliver Zandner (unten) ist bei Triestram & Partner zuständig für Marketing und Vertrieb von Individual-Lösungen und -Projekten auf Basis der Plattformen Java, Oracle und .Net. Ein aktueller Schwerpunkt ist das Marketing für das „rapid.Java development framework“.

Björn Christoph Fischer ist Consultant bei Triestram & Partner (t&p). Er ist einer der Architekten des „rapid. Java development framework“ von t&p – ein Java-Framework für das Rapid Application Development und die schrittweise Migration von Oracle-Forms-Anwendungen nach Java. Schwerpunktmäßig befasst er sich dabei mit Eclipse-Technologien.

Orientierungsfragen rund um Oracle

Till Brügelmann, Director Customer Care Center, ORACLE Deutschland B.V. & Co. KG

Durch die Übernahme von Sun Microsystems wird Java nun von Oracle weitergeführt. Java hat sich weltweit als systemunabhängige Plattform für Web-basierte Applikationen bewährt. Zum Nutzen seiner zahlreichen Anwender und Entwickler wird Java auch von Oracle weiter verbessert und fortgeführt werden. Nachfolgend einige wichtige Hinweise von Oracle.

Deutschsprachige Informationen zu Java stehen unter www.oracle.de, wobei die englischsprachigen Informationen auf www.oracle.com umfangreicher und teilweise aktueller sind. Für Java-Entwickler ist zudem das Oracle Technology Network (www.oracle.com/technetwork/java/index.html) mit allen technischen Details relevant.

Auf www.java.com/de kann die Software für die verschiedenen Betriebssysteme kostenlos heruntergeladen werden. Dort befindet sich auch die damit verbundene Lizenzvereinbarung. Auch die grundsätzliche Frage „Was ist Java?“ wird hier beantwortet. Auf nahezu alle technischen Fragen stehen im Hilfe-Bereich die gesuchten Antworten und die Support-Optionen. Oracle bittet um Verständnis, dass sich auch alle darüber hinausgehenden Anfragen auf diese Support-Optionen beschränken.

Für Feedback oder Input an das Java-Development gibt es unter http://bugreport.sun.com/bugreport/contact.jsp?language=de eine Möglichkeit. Für einen interaktiven Austausch mit anderen Java-Benutzern empfiehlt sich das Oracle Discussion Forum unter http://forums.oracle.com/forums/category.jspa?categoryID=285. Die Java User Group Deutschland e.V. (JUG) bietet unter www.java.de weitere Informationen an, die von Interesse sein können. Von Oracle unabhängige Unternehmen wie LivePerson haben sich auf den kostenpflichtigen Support von Java spezialisiert, deren Angebot man ebenfalls nutzen kann.

Wer kommerzieller Nutzer einen Wartungsvertrag für Java mit Oracle abgeschlossen hat und über eine gültige CSI-Nummer verfügt, dem steht neben dem Web-Tool My Oracle Support (https://support.oracle.com) auch die Oracle Support Hotline unter der Nummer 0180 2000 170 (für 0,06 € pro Anruf aus dem Festnetz oder maximal 0,42 €/Minute aus dem Mobilfunknetz) zur Verfügung.

Hinweis für PC-Benutzer mit Microsoft Windows: Zahlreiche Webseiten und Programme benötigen Java für ihr einwandfreies Funktionieren. Daher sind viele PCs bereits beim Kauf mit vorinstalliertem Java ausgeliefert. Alle Java-Funktionen inklusive der automatischen Update-Funktion können in der Systemsteuerung bedient werden. Wer Java deinstallieren möchte, kann dies über die in der Systemsteuerung vorgesehene Funktion erledigen. Für Fragen hierzu gibt es die integrierte Windows Hilfe-Funktion oder die Infos von Microsoft unter http://windows.microsoft.com/de-DE/windows/help.

Kontakt:
Hallo Oracle
http://www.oracle.com/de/corporate/contact/index.html

Till Brügelmann ist seit 17 Jahren bei Oracle. Er ist Gründer beziehungsweise Pionier von Oracle Direct in Deutschland und war dann deutschlandweit Team-Leiter zur gezielten Gewinnung von neuen Lösungs-Partnern (IVSs). Im Jahr 2000 erfolgte die Gründung und Leitung des deutschen Customer Care Centers „Hallo Oracle“, er ist verantwortlich für Kundenprogramme und alle Themen rund um die Kunden(un)zufriedenheit.

Concurrency – Durchsatz steigern

Dr. Stefan Koch, ORDIX AG

Der Kunde fordert einen Durchsatz von zwanzig Nachrichten pro Sekunde, Ihr Algorithmus schafft gerade mal zwei. Zehn parallel laufende Algorithmen können vielleicht helfen. Keine einfache Aufgabe. Doch Java hat mit dem Paket „java.util.concurrent" wichtige Hilfsmittel an Bord. Der Artikel stellt ThreadPool, BlockingQueue und atomare Klassen exemplarisch vor.

Der Motor der Datenverarbeitung ist der Prozessor. Dieser wird durch Lese- oder Schreibprozesse von der Arbeit abgehalten. Parallele Verarbeitungsprozesse nutzen diese Arbeitslücken. Während eine Verarbeitung auf Daten wartet, kann die zweite weiterrechnen.

Bei dem gewöhnlichen Dreikampf Eingabe, Verarbeitung und Ausgabe liegt genug Potenzial zur Steigerung des Durchsatzes vor. Ein Gefühl dafür gibt folgende grobe Faustformel: Ein Zugriff auf den Datenbank-Server bringt mindestens 5 ms Wartezeit mit sich. In 1 ms können etwa 1000 Zeilen Quellcode ausgeführt werden. Hinzu kommt: Bei Prozessoren mit mehreren Kernen sowie Rechnern mit mehreren Prozessoren herrscht häufig ein Überangebot an CPU-Kapazität.

Spezielle Bibliotheken nutzen

Java verfügt über die Grundbausteine der Parallelisierung. Die Parallelverarbeitung bringt jedoch Herausforderungen mit sich [1]; parallele Zugriffe auf dasselbe Objekt führen zu unerwünschten Zuständen. Diese lassen sich durch Synchronisierung vermeiden – das aber kann Dead-Locks nach sich ziehen. Das Paket „java.util.concurrent" liefert fertige Komponenten, die helfen, solche Probleme zu vermeiden. Einige dieser Komponenten werden im Folgenden vorgestellt.

Arbeit zentral verwalten: ThreadPool

Der ThreadPool übernimmt wesentliche Aufgaben der Thread-Verwaltung:

- Threads ausführen
- Anzahl parallel laufender Threads überwachen
- Threads in eine Warteschlange übernehmen
- Threads nach Gebrauch wiederverwerten
- Geordnetes Herunterfahren des Pools

Damit stellt der ThreadPool eine Lösung für wiederkehrende Aufgaben der parallelen Programmierung zur Verfügung. Listing 1 zeigt Programm-Code zur Nutzung eines ThreadPools. Im Beispiel wird dieser durch die Executors-Methode „newFixedThreadPool" erzeugt. Die Ausführung eines Programmteils wird durch den Aufruf der Methode „execute" und die Übergabe eines Runnable-Objekts angestoßen. Dabei startet die Ausführung erst dann, wenn ein Thread frei ist. Ansonsten kommt der Auftrag zunächst in eine Warteschlange.

Damit beim Ende der Anwendung keine Aufträge im Executor-Service verloren gehen, wird die Methode „shutdown" aufgerufen. Neue Aufträge werden nicht entgegengenommen, Aufträge in der Warteschlange abgearbeitet und der Service beendet.

Producer an Consumer vermitteln: BlockingQueue

Das Muster „Producer and Consumer" beschreibt einen Lösungsansatz in der Architektur zur Strukturierung paralleler Prozesse: der Producer-Algorithmus erzeugt Request-Objekte. Der Consumer-Algorithmus führt aufgrund dieser Request-Objekte eine Aktion aus. Ein Request-Objekt kann beispielsweise eine Bestellung sein. Durch Einsatz dieses Architekturmusters werden Producer und Consumer voneinander entkoppelt. Der gemeinsame Zugriff auf Objekte wird vermieden. Die Leistungsfähigkeit des Consumers ist unabhängig vom Producer skalierbar.

Eine Queue (java.util.Queue) ist zur Implementierung des Architekturmusters gut geeignet (siehe Abbildung 1). Der Producer stellt den Request in die Queue ein. Die Methode „put" stellt sicher, dass kein Request verloren geht. Ist die Queue voll, so wird so lange gewartet, bis sie den Request wieder aufnehmen kann. Der Consumer holt mit „take" den Request aus der Queue. Ist diese leer, so wird auf einen Request gewartet.

Für die eingesetzte „BlockingQueue" [3] gibt es Zugriffsmethoden, die entweder auf die Antwort der „BlockingQueue" warten oder ohne Ergebnis zurückkehren.

Daten sammeln: Concurrent Map

In manchen Fällen lässt sich der Zugriff auf gemeinsame Objekte nicht vermeiden. Nicht selten kommen Collections zum Einsatz, um Daten aus unterschiedlichen Threads zu sammeln und zur Verfügung zu stellen. In diesem Fall ist die Collection das Objekt, das durch parallele Threads verändert wird.

Synchronisierte Collections (etwa mit „SynchronizedMap") werden mit der Helper-Klasse „java.util.Collections" erstellt.

Durch die Synchronisierung der Zugriffsmethoden ist die Konsistenz der Collection sichergestellt. Vorsicht ist beim Einsatz von Iteratoren geboten. Hier muss der Entwickler selbst für den synchronisierten Zugriff sorgen.

Die „java.util.concurrent.ConcurrentMap“ liefert weitere synchronisierte Methoden, die in der Konkurrenz-Situation behilflich sein können:

- „Remove(key, value)“ sichert zu, dass ein Element aus der Map nur dann gelöscht wird, wenn „key“ und „value“ mit den Angaben übereinstimmen.
- „Replace(key, oldValue, newValue)“ ersetzt das durch „key“ und „oldValue“ spezifizierte Element durch „key“ und „newValue“.
- „PutIfAbsent(key, value)“ fügt nur dann das durch „key“ und „value“ spezifizierte Element ein, wenn es zu „key“ noch keinen anderen Wert gibt.

Die Ausführung dieser Methoden erfolgt atomar, vergleichbar mit einer Datenbank-Transaktion.

Arbeiten ohne Lock: Atomic Variables

Durch die Synchronisation von Methoden können unerwünschte parallele Zugriffe verhindert werden – Deadlocks oder Performance-Einbußen können aber die Folge sein [1]. Als eine Alternative zum Sperren kommt im Datenbank-Umfeld das Optimistic Locking zum Einsatz. Ein Datensatz wird gelesen und verarbeitet. Beim Speichern wird überprüft, ob der Datensatz noch dieselbe Version hat. Hat sich der Datensatz in der Zwischenzeit verändert, so ist ein Konkurrent schneller gewesen – die Verarbeitung muss erneut erfolgen.

Das Optimistic Locking verzichtet vollständig auf eine Sperre während der Verarbeitung. „Optimistic“ ist dieses Verfahren, weil angenommen wird, dass ein konkurrierender Zugriff nur in Ausnahmefällen stattfindet. Deadlocks können nicht vorkommen.

In Java ist eine Art Optimistic Locking für konkurrierende Prozesse mithilfe des Pakets „java.util.concurrent.atomic“ möglich. Dazu gibt es die Methode „compareAndSet“. Sie verändert das Objekt nur dann, wenn es noch den erwarteten Wert hat. Das entspricht im Datenbank-Bereich dem Statement „update“ mit Versionsvergleich: „where version=myVersion“. Die Methode „compareAndSet“ ist zwar atomar implementiert. Der Entwickler muss aber den Fall berücksichtigen, dass der Wert des Objekts zwischenzeitlich verändert wurde.

Neben „compareAndSet“ werden in der Klasse „AtomicInteger“ eine Reihe atomarer Methoden angeboten: „addAndGet“ fügt eine Zahl hinzu und liefert unmittelbar das Ergebnis zurück. Die Methoden „decrement-„ und „incrementAndGet“ helfen beispielsweise beim Runter- oder Raufzählen.

Neben den Typen „Boolean“, „Integer“ und „Long“ wird für allgemeine Klassen die „AtomicReference“ angeboten. Diese Klasse verwaltet ein Objekt beliebigen Typs. Neben der Methode „compareAndSet“ wird die atomare Methode „getAndSet“ angeboten.

Fazit

Dieser Artikel zeigt exemplarisch, wie das Paket „java.util.concurrent“ für die Implementierung paralleler Ausführungen hilfreiche Klassen zur Verfügung stellt. Die Verwendung dieser Komponenten spart Entwicklungszeit und reduziert die Risiken der Entwicklung erheblich. Betont werden muss jedoch, dass der Einsatz dieser Klassen nicht die Lösung aller Probleme bedeutet. Die Implementierung paralleler Algorithmen stellt immer eine Herausforderung dar.

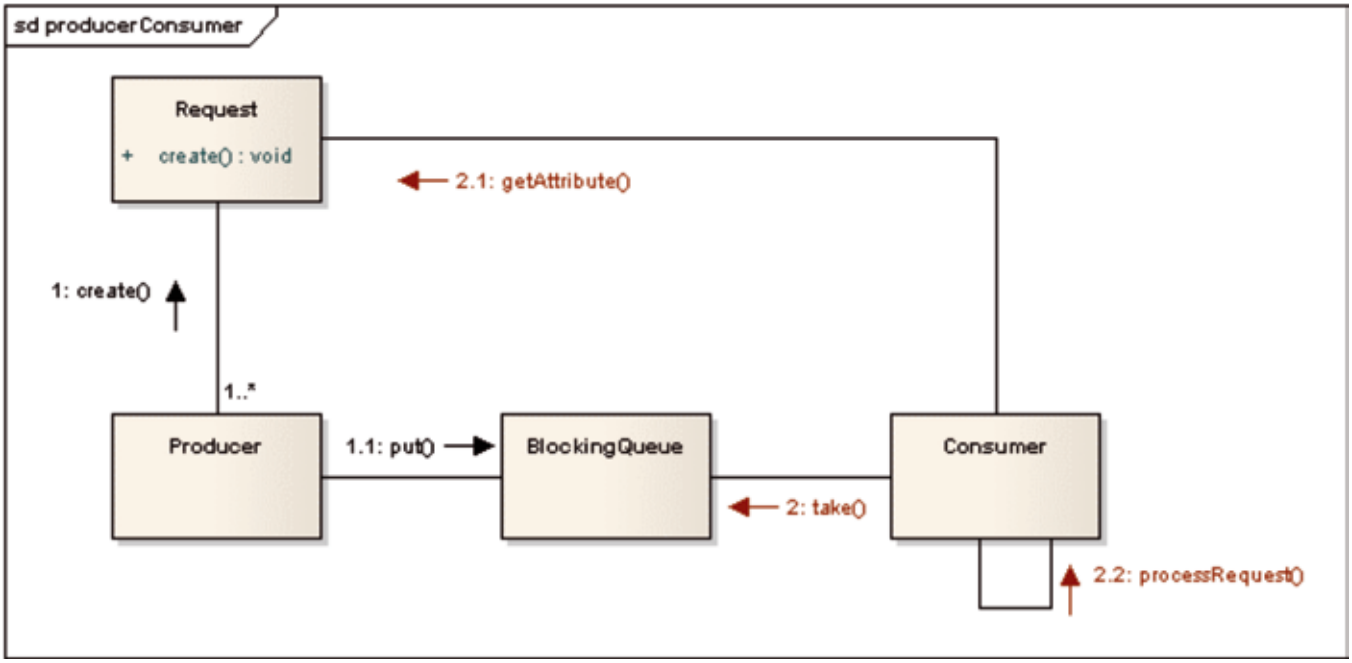


Abbildung 1: Kommunikationsdiagramm zum Producer-Consumer-Pattern

```
package de.ordix.threadPool;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
public class ThreadPool01 {
	public static class Zaehler implements Runnable {
		private long maxNumber;
		private String name;
		public Zaehler(long maxNumber, String name) {
			super();
			this.maxNumber = maxNumber;
			this.name = name;
		}
		public void run() {
			for(long i=0; i< maxNumber; i++);
			System.out.println(name + " " + maxNumber);
		}
	};
	public static void main(String[] args) {
		ExecutorService threadPool = Executors.newFixedThreadPool(10);
		for(int i=0; i < 1000; i++) {
			threadPool.execute(new Zaehler(i * 1000,"zaehler" + i));
		}
		threadPool.shutdown();
	}
}
```

Listing 1: Anwendungsbeispiel für einen ThreadPool

Weiterführende Links

[1] Kapitel „Concurrency" im Java-Tutorial: http://download.oracle.com/javase/tutorial/essential/concurrency/index.html

[2] Java theory and practice – Going atomic: http://www.ibm.com/developerworks/java/library/j-jtp11234/

[3] Beispielimplementierung des Producer-Consumer-Patterns: http://www.ordix.de/ORDIXNews/verzeichnis_12011.html

Kontakt:
Dr. Stefan Koch
info@ordix.de

Dr. Stefan Koch, Java-Spezialist der ORDIX AG, schreibt seit Jahren Artikel über Neuigkeiten und Best-Practice-Möglichkeiten in der Java-Entwicklung für das firmeneigene IT-Magazin „ORDIX News".

Des Entwicklers kleine nützliche Helfer ...

Daniel van Ross, Java User Group Deutschland e.V.

Die tägliche Arbeit eines Software-Entwicklers beinhaltet neben den herausfordernden Aufgaben, die dem Fortschreiten des Projekts dienen, auch immer wiederkehrende, eher lästige Tätigkeiten wie das Aufräumen alten Codes und das Suchen nach Fehlern.

Wer selbst aktiv in der Entwicklung tätig ist, weiß, wie viel Zeit eines Arbeitstages man mit Warten verbringt. Je nachdem woran man gerade arbeitet, muss man einen kompletten Zyklus bestehend aus Build-Prozess, Deployment und Initialisierung der Software abwarten, bis man testen kann, ob die gerade gemachten Änderungen das gewünschte Ergebnis erzielen. Unter Umständen gehört auch noch ein Restart des Application Servers zu diesem Zyklus, wodurch sich die Wartezeit weiter erhöht.

Dieser Artikel stellt eine Reihe nützlicher Werkzeuge kurz vor, um die tägliche Arbeit des Entwicklers zu beschleunigen. Die vorgestellten Programme bieten dabei Unterstützung auf ganz unterschiedliche Art und Weise; die einen zeigen potenziell fehlerhaften Code direkt während des Entwickelns an, die anderen automatisieren Teile des Release-Zyklus' und wieder andere helfen, die Code-Qualität zu verbessern, wodurch ebenfalls Fehler im Code vermieden werden sollen.

Die meisten vorgestellten Projekte richten sich primär an Java-Entwickler, viele lassen sich aber auch für andere Sprachen nutzen, und für einige gibt es sicherlich auch ebenbürtige Alternativen für andere Plattformen. Die Liste der vorgestellten Helfer erhebt natürlich keinen Anspruch auf Vollständigkeit, denn einige der vorgestellten Projekte sind noch recht jung, während andere, die vor einem Jahr noch Teil dieses Artikels gewesen wären, nicht mehr ausgebaut werden, was gerade bei einer sich noch weiterentwickelnden Sprache wie Java ausschlaggebend für die Nützlichkeit der Hilfsmittel ist.

Entwicklungsumgebungen

Die erste große Erleichterung, die in den letzten Jahren in den Alltag des Java-Entwicklers Einzug gehalten hat und die viele der jüngeren Leser für selbstverständlich halten, sind die IDEs. Eclipse, NetBeans, IntelliJ IDEA & Co. haben das Schreiben von Java-Code erheblich erleichtert, sodass es heute schwer vorstellbar ist, dass noch vor zehn Jahren die gleiche Arbeit mit mehr oder weniger komfortablen Editoren erledigt wurde. Dabei sind die von den IDEs integrierten GUI-Editoren, Compiler, Debugger oder auch Profiler nur ein kleiner Teil. Das eigentliche Schreiben des Quellcodes wird durch Code Completion, Code Generation und Refactoring-Funktionalitäten erleichtert.

Auf dem Markt finden sich sowohl kommerzielle Produkte als auch sehr fortschrittliche Open-Source-Projekte. Der Funktionsumfang der aktuellen Versionen ist durchaus in weiten Teilen miteinander vergleichbar. Wer für Oracle-Produkte entwickelt, wird eher zu JDeveloper [1] tendieren; wer seinem Projekt eine SWT GUI verpassen möchte oder muss, wird sich sicherlich eher für die Eclipse IDE [2] als für NetBeans [3] entscheiden.

Welche IDE ein Entwickler einsetzt, hängt in vielen Fällen aber nicht nur von den gebotenen Features ab: Äußere Faktoren wie „Welche IDE wird von den Kollegen oder Mitentwicklern benutzt?" und „Wel-

che IDE finde ich persönlich in der Handhabung am intuitivsten?“ spielen bei der Wahl oft eine genauso große Rolle.

Die beiden bedeutendsten Open-Source-Vertreter unter den Java-IDEs sind die bereits genannten Eclipse und NetBeans. Beide haben bereits seit einigen Jahren ihren festen Platz in der Entwicklergemeinde gefunden und bieten im Grunde alles, was man braucht. NetBeans unterstützt dabei von Hause aus eine breitere Palette an Programmiersprachen wie das inzwischen obsolete JavaFX Script, während sich die Eclipse-Anhänger an den unzähligen 3rd-Party-Plug-ins für nahezu jeden Zweck erfreuen können. Einige dieser Plug-ins werden im Verlauf dieses Artikels noch genauer beleuchtet.

Darüber hinaus ist natürlich IntelliJ IDEA [4] aus dem Hause JetBrains zu erwähnen. Das kommerzielle Produkt ist erst kürzlich in der Version 10 erschienen und in einer kostenlosen Community-Edition verfügbar, die jedoch Java EE nicht unterstützt. Die Ultimate Edition deckt den vollen Umfang der Java-Entwicklung ab, der Preis ist vom Einsatzzweck abhängig. Zur Entwicklung von Open-Source-Projekten ist eine Ultimate-Edition ebenfalls kostenlos verfügbar.

Die neue Art der Entwicklungsumgebungen ist browserbasiert. Der Trend, immer mehr Anwendungen auf den Server zu verlagern und diese über den Browser zu bedienen, macht auch vor IDEs nicht halt. Der Funktionsumfang wird wohl auch in naher Zukunft nicht annähernd den der schwergewichtigen Desktop-Anwendungen erreichen, aber mit dem AppInventor[5] für Android Apps zeigt Google, was bereits heute machbar ist und wohin die Reise geht.

Versionierungssysteme

Ein ebenfalls nicht mehr wegzudenkendes Werkzeug sind Versionierungssysteme, egal ob zentrale Repositories wie CVS [6] und Subversion [7] oder dezentrale wie GIT [8] oder Mercurial [9]. Natürlich gibt es auch hier neben den genannten kostenfreien Systemen kommerzielle Varianten auf dem Markt. Ähnlich wie bei den IDEs gilt auch hier: Die Entscheidung für oder gegen ein bestimmtes Versionierungssystem hängt von den persönlichen Vorlieben und dem im Projekt/Arbeitsumfeld verwendeten System ab.

Build Management

Zu den schon nicht mehr ganz so selbstverständlichen Werkzeugen gehören die Build-Management-Tools. Diese bestimmen den Build-Prozess vom Sourcecode hin zur auslieferbaren Software. Die prominentesten Vertreter sind hier Ant [10] und erst seit wenigen Jahren auch Maven [11]. Bei beiden handelt es sich um Apache-Projekte, sie haben sich als die dominierenden Werkzeuge für diese Aufgabe durchgesetzt. Ant ist zwar die ältere Variante, aber flexibler. Ant stellt ein Framework bereit, in dem sogenannte „Tasks“ ausführbar sind. Aus den Tasks wie „Kompilieren“, „Kopieren“ oder „Jars erstellen“ baut der Entwickler sich seinen Build-Prozess. Maven hingegen hat den Build-Prozess quasi vorgegeben, ist aber auch durch Ant-Tasks erweiterbar. Viele der im Folgenden vorgestellten Tools liefern einen eigenen Ant-Task mit und lassen sich somit in den Build-Prozess integrieren.

Code-Analyse

Die bisher erwähnten Werkzeuge dienen primär der Arbeitserleichterung der Entwickler, die folgenden beschäftigen sich mehr mit dem Quellcode selbst und sollen helfen, Fehler zu finden beziehungsweise zu vermeiden und die Code-Qualität zu verbessern. Sie untersuchen den Quellcode und prüfen unter anderem den Code-Stil. Sie zeigen, ob Standards und Coding-Konventionen eingehalten wurden, finden leere Code-Blöcke oder Statements und analysieren die Länge von Methoden und Klassen. Einige prüfen den Code anhand von Metriken und berechnen die Komplexität einzelner Methoden, bestimmen Change-Risk-Anti-Patterns und analysieren Abhängigkeiten. Wieder andere suchen nach Mustern in Quell- oder Bytecode, von denen bekannt ist, dass sie vermehrt zu Fehlern führen können.

Wie der Name bereits sagt, erlaubt CheckStyle [12] die Prüfung des Code-Stils und hilft Coding-Richtlinien einzuhalten. Das fängt bei fehlenden Kommentaren an und geht über Warnungen bei Überschreitung einer maximalen Anzahl Boole'scher Operationen in einem Ausdruck bis hin zu Komplexitätsprüfungen. CheckStyle liefert sowohl einen Ant-Task als auch ein Kommandozeilen-Werkzeug, mit dessen Hilfe man Berichte über den analysierten Code erstellen kann. Es gibt zudem CheckStyle-Plug-ins für alle gängigen IDEs, diese zeigen Warnungen und/oder gefundene Fehler direkt im Kontext des Codes an. CheckStyle steht kostenlos unter der GNU Lesser General Public License (LGPL) zur Verfügung.

CRAP4J [13] analysiert Quellcode und bestimmt die Komplexität einzelner Methoden. Je komplexer eine Methode, desto größer ist die Wahrscheinlichkeit, bei notwendigen Änderungen Fehler einzubauen. Mit Unit-Tests kann man der Fehlerwahrscheinlichkeit entgegenwirken, daher bezieht CRAP4J die Testabdeckung in die Berechnung des CRAP-Faktors mit ein. Die Ergebnisse werden in einem Be-

richt in HTML-Form zusammengefasst und erlauben dem Entwickler, schnell die größten potenziellen Fehlerquellen zu finden. CRAP4J ist als Eclipse-Plug-in oder als Ant-Task verfügbar und kann kostenlos von der Projekt-Homepage heruntergeladen werden.

Zu den bekanntesten Tools dieser Gattung zählt FindBugs [14]. Es basiert auf Bug-Patterns und untersucht den Bytecode danach. Die Patterns wurden entwickelt, um komplexe Sprach-Features, häufig missverstandene API-Methoden oder die falsche Verwendung Boole'scher Operatoren aufzuspüren. Die gefundenen potenziellen Bugs werden in Kategorien eingeteilt und so dem Entwickler präsentiert. Die Zahl der geprüften Patterns wächst stetig seit dem ersten Erscheinen von FindBugs, sodass inzwischen nicht nur wirklich potenzielle Fehler gefunden werden, sondern auch auf ineffizienten Code oder mögliche Sicherheitslücken hingewiesen wird. FindBugs beinhaltet neben dem Kommandozeilen-Interface, dem Ant-Task und dem Eclipse-Plug-in auch eine eigene GUI und wird unter der LGPL entwickelt.

Ähnlich arbeitet auch PMD [15] . Es basiert ebenfalls auf Patterns, inspiziert aber den Java-Quellcode und nicht den Bytecode. Neben möglichen Fehlern versucht PMD, auch ungenutzten oder ineffizienten Code zu finden, und warnt auch vor kopierten Code-Passagen. Neben dem obligatorischen CLI und dem Ant Task kommt auch PMD mit einer eigenen GUI. Zudem ist eine Integration in nahezu alle gängigen IDEs und Java-Editoren möglich.

Der UCDetector (= Unnecessary Code Detector) [16] ist ein Eclipse-Plug-in, das hilft, ungenutzten (alten) Code zu finden. Eclipse selbst weist den User bereits auf ungenutzte „private"-Methoden hin, kann dies aber nicht für „public"- oder „protected"-Methoden. Diese Lücke versucht UCDetector zu schließen.

Ein noch relativ wenig verbreitetes Hilfsmittel, das Google erst kürzlich kostenlos zur Verfügung gestellt hat, ist CodePro AnalytiX [17]. Das Eclipse-Plug-in weist Entwickler auf mögliche Fehler hin und visualisiert die Komplexität des Codes und die Abhängigkeiten von Packages untereinander. Es bietet aber auch darüber hinausgehende Features wie Test Generation und Code Coverage Analyse. CodePro AnalytiX steht zusammen mit weiteren Tools, die Google mit der Übernahme von Instantiations übernommen hat, ebenfalls kostenlos zur Verfügung.

Continuous-Integration-Tools

Eine andere Klasse von Werkzeugen sind die sogenannten „Continuous-Integration-Tools" wie Bamboo, Cruise Control oder das Open-Source-Projekt Hudson [18] beziehungsweise Jenkins [19], das in den letzten Wochen aufgrund von Streitigkeiten mit Oracle um die Namensrechte öfter Thema in der Fachpresse war. Continuous-Integration-Werkzeuge automatisieren den Build-Test-Deploy-Zyklus und helfen, schneller auf Fehler im Code und fehlgeschlagene Builds zu reagieren. Richtig eingesetzt motivieren diese Werkzeuge alle Entwickler im Team dazu, ihren Code besser zu prüfen, bevor sie ihn ins Repository einchecken, da ein fehlgeschlagener Build oder Test sofort für alle sichtbar ist.

Code-Review-Tools

Meist nur in größeren Teams anzutreffen sind Code-Review-Tools wie das kommerzielle Crucible [20] oder die Open-Source-Tools ReviewBoard [21], JCR [22] oder CodeStriker [23]. Code Reviews an sich gehören seit je her zur Software-Entwicklung. In der einfachsten Form hat man Absprachen mit Kollegen, gegenseitig nochmal schnell den Code des anderen zu überfliegen, bevor er ins Version Control System eingecheckt wird. Code Review Tools helfen diesen Prozess zu formalisieren, indem sie sicherstellen, dass der Verantwortliche den Code freigeben muss, bevor er endgültig im Repository landet. Ob und in welchem Ausmaß der Einsatz sinnvoll ist, liegt oft nicht in der Entscheidung des einzelnen Entwicklers. Hier sind es häufig Vorgesetzte oder firmeninterne Richtlinien, die angeben, wie so ein Prozess auszusehen hat.

Fazit

Neben den genannten Tools wird jeder Software-Entwickler täglich mit einer Reihe weiterer Werkzeuge konfrontiert, die seine Arbeitsweise beeinflussen. Diese reichen von Bugtrackern über Wikis zur Dokumentation bis hin zu Bookmark-Sammlungen. Die Anzahl der Werkzeuge, die die tägliche Arbeit eines Entwicklers erleichtern sollen, wächst stetig und es gibt beinahe so viele, wie es Aufgaben zu lösen gibt. Ständig werden neue Tools entwickelt, andere verschwinden vom Markt und viele warten nur darauf, entdeckt zu werden.

Weitere Informationen

[1] JDeveloper: http://www.oracle.com/technetwork/developer-tools/jdev
[2] Eclipse IDE: http://www.eclipse.org
[3] NetBeans: http://www.netbeans.org
[4] IntelliJ IDEA: http://www.jetbrains.com/idea
[5] AppInventor: http://appinventor.googlelabs.com/about
[6] CVS: http://www.cvshome.org
[7] Subversion: http://subversion.apache.org
[8] GIT: http://git-scm.com
[9] Mercurial: http://mercurial.selenic.com
[10] Ant: http://ant.apache.org
[11] Maven: http://maven.apache.org
[12] CheckStyle: http://checkstyle.sourceforge.net
[13] CRAP4J: http://www.crap4j.org
[14] FindBugs: http://findbugs.sourceforge.net
[15] PMD: http://pmd.sourceforge.net
[16] UCDetector: http://www.ucdetector.org
[17] CodePro AnalytiX: http://code.google.com/javadevtools
[18] Hudson: http://hudson-ci.org
[19] Jenkins: http://jenkins-ci.org
[20] Crucible: http://www.atlassian.com/software/crucible
[21] ReviewBoard: http://www.reviewboard.org
[22] JCR: http://jcodereview.sourceforge.net
[23] CodeStriker: http://codestriker.sourceforge.net

Kontakt:
Daniel van Ross
daniel@vanross.de

Daniel van Ross ist Software-Entwickler bei der NeptuneLabs GmbH in Lemgo. Sein Arbeitsschwerpunkt ist die Entwicklung serverseitiger Java-Anwendungen. Daneben versorgt er die Webseite der Java User Group Deutschland e.V. mit aktuellen News.

Schematron: XML mit Präzision

Tobias Israel, buschmais GbR

Die Realisierung technischer Schnittstellen auf Basis von XML hat sich einen festen Platz erobert. Mithilfe aktueller Werkzeuge und Frameworks sind hervorragende Möglichkeiten geschaffen, Geschäftspartnern die Anbindung an die eigene IT-Infrastruktur zu ermöglichen.

Die automatisierte Abwicklung von Geschäftsprozessen erfolgt nicht selten auf der Basis vertraglicher Vereinbarungen. Die Messlatte für die Qualität der betroffenen Schnittstellen liegt damit entsprechend hoch. Im Folgenden soll der Frage nachgegangen werden, wie Integritätsbedingungen einer XML-Schnittstelle zielführend erfasst und überprüft werden können.

Im allgemein akzeptierten Kontext der Webservice-Technologien bilden die Web Services Description Language (WSDL), das Simple Object Access Protocol (SOAP) sowie das XML-Schema die Eckpfeiler einer jeden XML-Schnittstelle. Bei WSDL und SOAP weisen ein entsprechendes Verständnis der Spezifikationen und die Kenntnis angrenzender Standards des „WS-*"-Universums den Weg hin zum sauber gearbeiteten Korsett der Schnittstelle. Der eigentliche Knackpunkt liegt jedoch im Entwurf der auszutauschenden XML-Nachrichten. Mit ihnen wird die fachlich motivierte Anbindung an die eigene Prozesslandschaft realisiert.

Ein häufig auftretender Stolperstein ist dabei oft schon der Wiederstreit zwischen Flexibilität und strukturell verankerter Präzision. Die Frage, ob wegen jedes neu über die Lieferkette abgewickelten Geschäftsfalles auch die Schnittstelle angepasst werden soll, lässt sich eben nicht so einfach beantworten. Egal wie man es dreht und wendet, die Beschreibung der Datenstrukturen in XML-Schema wird in jedem Fall eine Restmenge an gewollten oder ungewollten Grauzonen zurücklassen.

Eine häufig ins Feld geführte Waffe hiergegen ist die Produktion einiger Kilogramm beschriebenen Papiers. Mit ihm soll den externen Schnittstellen-Partnern die Integritätsbedingungen der Schnittstelle nahegebracht werden. Die unterschiedlichen Standpunkte, Sprach- und Prozesswelten sorgen jedoch nicht selten für ein breit gefächertes Spektrum an Missverständnissen. Abhilfe kann hier die Flankierung der grammatikbasierten XML-Schema-Beschreibungen mit einer kontextabhängigen Validierung schaffen. Möglichst nah an der technischen Schnittstellen-Beschreibung kann so das umfangreiche Regelwerk der Integritätsbedingungen erfasst und später auch überprüft werden.

Gestatten: Schematron

Die Sprache Schematron bietet die Möglichkeit, XML-Dokumente regelbasiert zu validieren. Basis eines Schematrons ist – anders als bei XML-Schema – keine formale Grammatik. Die Beschreibung beruht vielmehr auf der Definition von Regeln, in denen Bezug auf einen bestimmten Kontext innerhalb des XML-Baums genommen wird.

Die Notation einer Schematron-Beschreibung beruht – wie auch bei XML-Schema – auf der XML (siehe Listing 1). Weiterhin ist eine Abfragesprache zur Ermittlung der Kontexte und zur Formu-

```
<s:schema xmlns:s="http://purl.oclc.org/dsdl/schematron">
  <s:title>Issue management service interface V1.00.0
        integrity constraints</s:title>
  <s:ns prefix="t"
      uri="http://example.com/IssueMngtService/1.0"/>
  <s:pattern>
    <s:rule context= "/t:CreateIssueRequest/t:ServiceLevel">
      <s:assert test="(t:ServiceLevel='EXPRESS') and
                (count(t:Issue) = 1)">
        Only one issue per request is permitted for
        service level EXPRESS.
      </s:assert>
    </s:rule>
    <s:rule context=
            "/t:CreateIssueRequest/t:Issue/t:Detail">
      <s:report test="contains(t:Key, 'SysERR05') and
                not(contains(t:Key, 'SysERR05-1'))">
        To avoid further enquiries about your issue,
        please provide sub code 'SysERR05-1' for main error
        code 'SysERR05'.
      </s:report>
    </s:rule>
    </s:pattern>
</s:schema>
```

Listing 1: IssueManagementServiceConstraints.sch

lierung der Zusicherungen und Tests erforderlich. In der Regel kommt an dieser Stelle XPath zum Einsatz. In Abhängigkeit von der eingesetzten Schematron-Implementierung kann jedoch auch auf die Mächtigkeit von XPath 2.0 oder XQuery zurückgegriffen werden.

Jede Schematron-Beschreibung wird mit dem Wurzelelement <schema> eingeleitet. Durch Angabe von <title> kann man optional eine verbale Beschreibung ergänzen. Die in den XPath-Ausdrücken verwendeten Namespaces und deren Präfixe sind zunächst mithilfe von <ns>-Elementen zu definieren. Nun folgen die innerhalb von <pattern>-Elementen gruppierten Regeln <rule>. Das Context-Attribut jedes Rule-Elements definiert den Bezugspunkt für die eingebetteten Zusicherungen. Diese zu formulieren ist Aufgabe der beiden Elemente <assert> und <report>.

Ein Assert beschreibt eine Zusicherung, die innerhalb des zuvor bestimmten Kontextes durch ein XML-Dokument erfüllt sein muss. Validiert der XPath-Ausdruck zu „false“, wird die angegebene Fehlermeldung ausgegeben. Bei Report-Elementen ist hingegen der Erfolg der gemachten Zusicherung für die Ausgabe des Textes entscheidend. Die Spezifikation schreibt ihnen daher auch einen informellen Charakter zu.

Asserts und Reports ermöglichen über die Prüfung einer technisch korrekten Umsetzung der XML-Datenformate hinaus auch einen Bezug zu den dahinter liegenden, fachlich motivierten Aspekten. Einzige Voraussetzung: Die Fehlermeldungen und Hinweise sind fachlich korrekt formuliert. Ganz nebenbei sorgt dieser Umstand dafür, dass komplexe Details auch schon in der Analysephase in verständlicher Form vorliegen.

Schematron-Prüfung

Das Anwenden eines Schematrons auf konkrete XML-Dokumente erfordert eine entsprechende Implementierung dieses Standards. Gleich welchen technologischen Kontext das eigene Projekt in diesem Punkt vorgibt, die Einstiegshürde ist gering. Die Verarbeitung wird durch die meisten Frameworks als eine Reihe von Transformationsschritten durchgeführt. Die Umsetzung mit einer neutralen Technologie wie XSLT liegt nicht nur auf der Hand, sondern wurde mit der Referenzimplementierung auch genauso umgesetzt.

```
<?xml version="1.0" encoding="UTF-8"?>
<t:CreateIssueRequest
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
  xmlns:t='http://example.com/IssueMngtService/1.0'
  xsi:schemaLocation='http://example.com/IssueMngtService/
                      1.0 IssueManagementService.xsd'>
  <t:CustomerID>20000000300340 1</t:CustomerID>
  <t:ServiceLevel>EXPRESS</t:ServiceLevel>
  <t:Issue>
    <t:Detail>
      <t:Key>SysERR01</t:Key>
      <t:Value>12AXU3B</t:Value>
    </t:Detail>
    </t:Issue>
    <t:Issue>
       <t:Detail>
          <t:Key>SysERR02</t:Key>
          <t:Value>34A45H8E</t:Value>
       </t:Detail>
    </t:Issue>
</t:CreateIssueRequest>
```

Listing 2: IssueManagementService.xml

Abbildung 1 stellt den Ablauf der Prüfung dar. Die erstellte Schematron-Beschreibung der abzusichernden Integritätsbedingungen wird in der ersten Phase mithilfe generischer XSLT-Skripte der Schematron-Implementierung transformiert. Das Ergebnis dieser Transformation ist ein weiteres XSLT-Skript. Dieses wird zur Überprüfung und Erstellung eines Testreports in Phase 2 auf die zu validierenden XML-Daten angewendet. Das Ergebnis der Prüfung ist eine XML-Datei im SVRL-Format. Sie bildet die Ausgangsbasis für eine mögliche dritte Phase, in der sie entsprechend bestehender Reporting-Anforderungen in ein besser lesbares Format überführt wird. SVRL ist als Standard-Ausgabeformat für ISO-Schematron fester Bestandteil der Spezifikation.

Abhängig vom Funktionsumfang der Implementierung ist jedoch die Möglich-

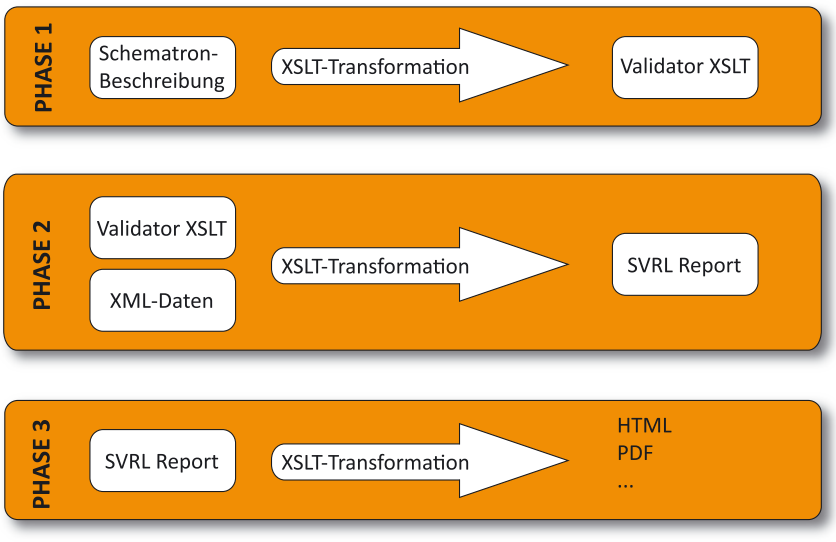


Abbildung 1: Ablauf der Schematron-Prüfung

keit gegeben, in Phase 2 das Prüfergebnis bereits in einem anderen Format ausgeben zu lassen. Wendet man das Schematron aus Listing 1 auf die XML-Datei aus Listing 2 an, entspricht das in SVRL formulierte Ergebnis dem aus Listing 3. Neben den bei der Prüfung angewandten Regeln finden wir auch eine fehlgeschlagene Zusicherung und die dazugehörende verbale Beschreibung.

Fazit

Obwohl dieser Artikel nur einen Teil der Mächtigkeit von Schematron beleuchtet, kann die vorgestellte Teilmenge schon in vielen Situationen ein hilfreicher Begleiter bei der Erstellung und Umsetzung XML-basierter Schnittstellen sein. Der zunächst entstehende Mehraufwand bei Erstellung und Pflege des zusätzlichen Dokuments ist sicher nicht zu leugnen. Er wird sich in den allermeisten Projekten jedoch sehr schnell bezahlt machen – gerade wenn es darum geht, eigene Systeme für eine Vielzahl unterschiedlicher Partner zu öffnen.

Quellen

http://xml.ascc.net/schematron/
http://www.schematron.com/
http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

Kontakt:
Tobias Israel
tobias.israel@buschmais.com


```
<?xml version="1.0" encoding="utf-8" ?>
<fileset date="2010/11/08">
   <file name="IssueManagementService.xml">
      <svrl:schematron-output
            xmlns:iso="http://purl.oclc.org/dsdl/schematron"
            xmlns:sch="http://www.ascc.net/xml/schematron"
            xmlns:svrl="http://purl.oclc.org/dsdl/svrl"
            xmlns:schold="http://www.ascc.net/xml/schematron"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:t="http://example.com/IssueMngtService/1.0"
            title="Issue management service interface V1.00.0
                   integrity constraints"
            schemaVersion="">
         <svrl:ns-prefix-in-attribute-values
               uri="http://example.com/IssueMngtService/1.0"
               prefix="t"/>
         <svrl:active-pattern/>
         <svrl:fired-rule
            context="/t:CreateIssueRequest/t:ServiceLevel"/>
         <svrl:failed-assert
      test="(t:ServiceLevel='EXPRESS') and
          (count(t:Issue) = 1)"
      location="/*[local-name()='CreateIssueRequest'
        and namespace-uri()='http://example.com/
        IssueMngtService/1.0']/*[local-name()=
        'ServiceLevel' and namespace-uri()=
        'http://example.com/IssueMngtService/1.0']">
        <svrl:text>
          Only one issue per request is permitted for
          service level EXPRESS.
        </svrl:text>
      </svrl:failed-assert>
      <svrl:fired-rule context="/t:CreateIssueRequest/
                    t:Issue/t:Detail"/>
      <svrl:fired-rule context="/t:CreateIssueRequest/
                    t:Issue/t:Detail"/>
   </svrl:schematron-output>
  </file>
</fileset>
```

Listing 3: result.xml

Schematron 1.0	Erste Version entwickelt am Academia Sinica Computing Centre, Taiwan
Schematron 1.3 Schematron 1.5 Schematron 1.6	Weiterentwicklungen der Spezifikation; Umsetzung in zahlreichen Implementierungen für unterschiedliche Plattformen
ISO Schematron	Aufbauend auf Version 1.6 offizieller ISO/IEC-Standard (19757-3); enthält das standardisierte SVRL-Ausgabenformat

Tabelle 1: Versionsübersicht Schematron

Tobias Israel arbeitet als IT-Berater auf dem Gebiet der Java-Enterprise-Technologien. Ein Schwerpunkt seiner Arbeit liegt auf der Erstellung und Umsetzung von Integrationsstrategien für verteilte Unternehmensanwendungen. Er ist Mitinhaber der buschmais GbR – einem Beratungshaus mit Sitz in Dresden.

Initiierung und Motivation von JSRs durch Open-Source-Systeme

Bernd Müller, Hochschule für angewandte Wissenschaften

Der Artikel „Die Geschichte der Bohne" in Java aktuell Q4/2010 beschreibt die historische Entwicklung von Java inklusive verschiedener Meilensteine, die an SE- und EE-Versionen geknüpft sind. Die Entwicklung von Java ist in bestimmten Teilgebieten von Java-EE stark durch Open-Source-Systeme geprägt. In diesem Zusammenhang wird die praktizierte und durchaus als positiv zu sehende gegenseitige Verwendung von Open-Source-Systemen beispielhaft an den populären Applications-Umfeld-Servern GlassFish und JBoss-AS dargestellt.

Im Mai 2006 gab Sun bekannt, dass das Java Development Kit (JDK) zukünftig unter einer GPL-Lizenz stehen werde. Im November 2006 erschienen zunächst der Compiler und die JVM als Open-Source. Die Klassenbibliothek wurde schrittweise auf Open-Source umgestellt, da sie proprietären Code enthielt, für den Sun nicht die notwendigen Rechte besaß. Mittlerweile stellt das Projekt OpenJDK [1] ein vollständiges Open-Source JDK bereit, sodass auch Linux-Distributionen wie Fedora, die ausschließlich freie Software enthalten (wollen), es verwenden können.

Diese Öffnung des Java JDK ist allgemein bekannt. Weniger bekannt ist der Umstand, dass Open-Source-Frameworks Java-Spezifikationen richtungsweisend und maßgeblich befruchtet haben. Wir belegen dies beispielhaft an JPA, CDI und Bean Validation.

Java Persistence API 1.0 (JSR 220)

Die zur Java 2 Enterprise Edition gehörenden Spezifikationen, insbesondere die Spezifikationen EJB 2.0 und 2.1, stießen auf große Kritik innerhalb der Entwicklergemeinde (zu komplex, zu umfangreich, nicht redundanzfrei etc.) und führten zur Entwicklung von Alternativen. Eine davon war Hibernate [2] im Bereich der Persistenz. Die Spezifikation EJB 3.0 (JSR 220) innerhalb der Java-Plattform Enterprise Edition 5 enthält als Unterspezifikation das Java Persistence API 1.0, das viele Hibernate-Konzepte übernommen hat. Der Hibernate-Code (siehe Listing 1) ändert sich in JPA (siehe Listing 2). Die Ähnlichkeiten (Ersetzung von Session durch EntityManager) sind offensichtlich. Gavin King, der geistige Vater von Hibernate, war Mitglied der Expertengruppe des JSR 220.

„Contexts and Dependency Injection for the Java EE platform" (JSR 299) und „Dependency Injection for Java" (JSR 330)

Das allgemein Martin Fowler zugeschriebene Konzept der Dependency Injection (DI) wurde in Java EE 5 erstmalig in einer Java-Spezifikation verwendet, um Container-spezifische Objekte und EJBs zu injizieren. JBoss-Seam [3] erweiterte die dortige Verwendung von DI auf beliebige Objekte, insbesondere deren Verwendung in der JSF-basierten GUI-Schicht, und verallgemeinerte das Prinzip des DI. Durch den Erfolg motiviert, initiierte JBoss den JSR 299, der nach zweimaliger Umbenennung schließlich den Namen „Contexts and Dependency Injection for the Java EE platform (CDI)" erhielt. Er führt die durch Seam begonnenen Innovationen fort und erweiterte sie beispielsweise um Typsicherheit. CDI ist in Java EE 6 enthalten. Gavin King war als maßgeblicher Kopf hinter Seam auch Leiter des Spezifikationsgremiums für CDI. Die Referenzimplementierung von CDI wird unter dem Code-Namen „Weld" als Seam-Teilprojekt von Seam-Entwicklern als Open-Source realisiert.

Kurz vor der Fertigstellung des JSR 299 haben Google und SpringSource einen weiterer Vorschlag zur Spezifikation von DI in Java eingereicht. Google und SpringSource bieten mit Google Guice und Spring zwei weitverbreitete DI-Frameworks auf Open-Source-Basis an. Der Vorschlag von „Dependency Injection for Java" als JSR 330 führte zu etlichen Unruhen im Java-Bereich, da eine Konkurrenz zum JSR 299 gesehen wurde. Mittlerweile ist die Spezifikation jedoch verabschiedet und Basis des JSR 299. Bob Lee von Google und Rod Johnson von SpringSource waren die Leiter des Spezifikationsgremiums.

Bean Validation (JSR 303)

Hibernate Validator [4], ein Hibernate-Teilprojekt, realisiert die Validierung von Objekt-Properties. Die Validierung erfolgt durch die Annotation der Properties wie mit @Min, @Max, @NotNull, @NotEmpty, @Future und @Past, deren Namen für sich sprechen. In geschichteten Anwendungen erfolgt häufig eine Validierung auf der GUI- aber auch auf der Persistenz-Ebene, was dem DRY-Prinzip widerspricht. Das bereits erwähnte JBoss Seam zeigte, wie mithilfe des Hibernate Validators das DRY-Prinzip

```
SessionFactory factory =
new Configuration().configure().buildSessionFactory();
Session session = factory.openSession();
Transaction tx = session.beginTransaction();
session.save(<ein Entity>);
tx.commit();
```

Listing 1

```
EntityManagerFactory factory =
Persistence.createEntityManagerFactory(„<name>“);
EntityManager em = factory.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
em.persist(<ein Entity>);
tx.commit();
```

Listing 2

respektiert und die auf der Persistenz-Schicht erfolgte Annotation von Properties mit Validierungs-Constraints bis zur Validierung von Eingaben in der Oberfläche hochgehoben werden kann.

Java EE 6 enthält mit Bean Validation eine Spezifikation, die Konzepte von Hibernate Validator aufnimmt und in Java-EE-6-Containern sowohl für die Oberfläche (JSF) als auch die Persistenz (JPA) verwendet werden kann. Bean Validation ist als JSR-303 spezifiziert und Hibernate Validator ist die Referenzimplementierung. Emmanuel Bernard, ein JBoss-Mitarbeiter und maßgeblicher Kopf von Hibernate Validator, war Leiter des Spezifikationsgremiums.

Firmenübergreifende Verwendung von Open-Source-Implementierungen

Nach den drei genannten Beispielen der Motivation von Java Specification Requests durch Open-Source-Systeme wollen wir den Artikel mit einer kleinen Analyse der beiden populären Application-Server GlassFish und JBoss-AS beschließen. Beide Container stehen unter Open-Source-Lizenzen und werden maßgeblich von Sun- beziehungsweise JBoss-Mitarbeitern entwickelt. Erwähnenswert ist hier, dass Teilprojekte jeweils ausgetauscht werden. JBoss verwendet in seinem Container die von Sun realisierte Referenzimplementierung von JSF (JSR 314), während Sun im eigenen Container die von JBoss realisierten Referenz-Implementierungen von CDI (JSR 299) und Bean Validation (JSR 303) benutzt. Neben dieser gegenseitigen Verwendung bedienen sich beide Container noch weiterer Open-Source-Systeme wie von Apache, Eclipse und Codehaus. Eine vollständigere Übersicht findet man in [5].

Fazit

Wir haben an drei Beispielen gezeigt, dass populäre Open-Source-Systeme die Entwicklung von Java im EE-Umfeld stark beeinflussen. Dies ist als äußerst positiv zu betrachten, da sich die Popularität der Open-Source-Systeme auf die Java-Plattform überträgt und viele Entwickler ihr bestehendes Know-how weiterverwenden können.

Weitere Informationen

[1] http://www.openjdk.org
[2] http://www.hibernate.org
[3] http://www.seamframework.org
[4] http://www.hibernate.org/subprojects/validator.html
[5] Bernd Müller. Java und Open-Source – Ein Geben und Nehmen. Proc. Informatik 2010, Klaus-Peter Fähnrich, Bogdan Franczyk (Hrsg.), Service Science – Neue Perspektiven für die Informatik, Band 2, 2010, Leipzig.

Kontakt:
Bernd Müller
bernd.mueller@ostfalia.de

Bernd Müller ist seit März 2005 Professor für Software-Technik an der Fakultät Informatik der Hochschule Braunschweig/Wolfenbüttel, nachdem er sieben Jahre Professor für Wirtschaftsinformatik an der Hochschule Harz war. Praktische Erfahrung hat er zuvor im Wissenschaftlichen Zentrum der IBM in Heidelberg sowie bei HDI Informationssysteme in Hannover gesammelt. Er ist Autor mehrerer Bücher zu den Themen „JSF“, „JPA“ und „Seam“.

Mehr Rapid Java mit XDEV 3

Gerald Kammerer, freier Redakteur

Wer Geschäftsanwendungen bislang mit 4GL-Tools wie Access, FoxPro, Oracle Forms oder Visual Basic entwickelt hat, wird mit XDEV 3 umgehend auf Java umsteigen können. In den letzten Monaten gab es erneut einen gewaltigen Entwicklungsschub.

In der letzten Ausgabe haben wir die freie Java-Entwicklungsumgebung XDEV 3 vorgestellt. Diese Umgebung ermöglicht Rapid Application Development (RAD) auch für Java. Datengetriebene Geschäftsanwendungen lassen sich damit genauso schnell und einfach entwickeln wie mit Oracle Forms und Microsoft Access. Mit XDEV 3 ist das Ergebnis jedoch immer Java. Die Entwicklungsumgebung richtet sich damit an Entwickler, Software-Architekten, Business-Manager und Fachkräfte in IT- und Fachabteilungen, die sich sowohl um die Geschäfts- als auch um die Entwicklungsseite kümmern und von proprietären Datenbank-Tools wie Access, FoxPro, Oracle Forms, Powerbuilder oder Filemaker ohne viel Lernaufwand auf Java umsteigen möchten.

Bislang kamen RAD-Ansätze für Java kaum über Baukasten-Niveau hinaus. Das ermöglicht zwar einen schnellen Einstieg – individuelle Anpassbarkeit, Erweiterbarkeit und Skalierbarkeit sind dabei jedoch meist nur begrenzt möglich. XDEV 3 ist dagegen die erste IDE für Java, die Rapid Application Development mit konventioneller Java-Programmierung vereint und während der Entwicklung einen fließenden Übergang ermöglicht. Eingriffe in den generierten Code sind an jeder Stelle des Programms möglich. Durch die zahlreichen RAD-Features lassen sich die Entwicklungs-

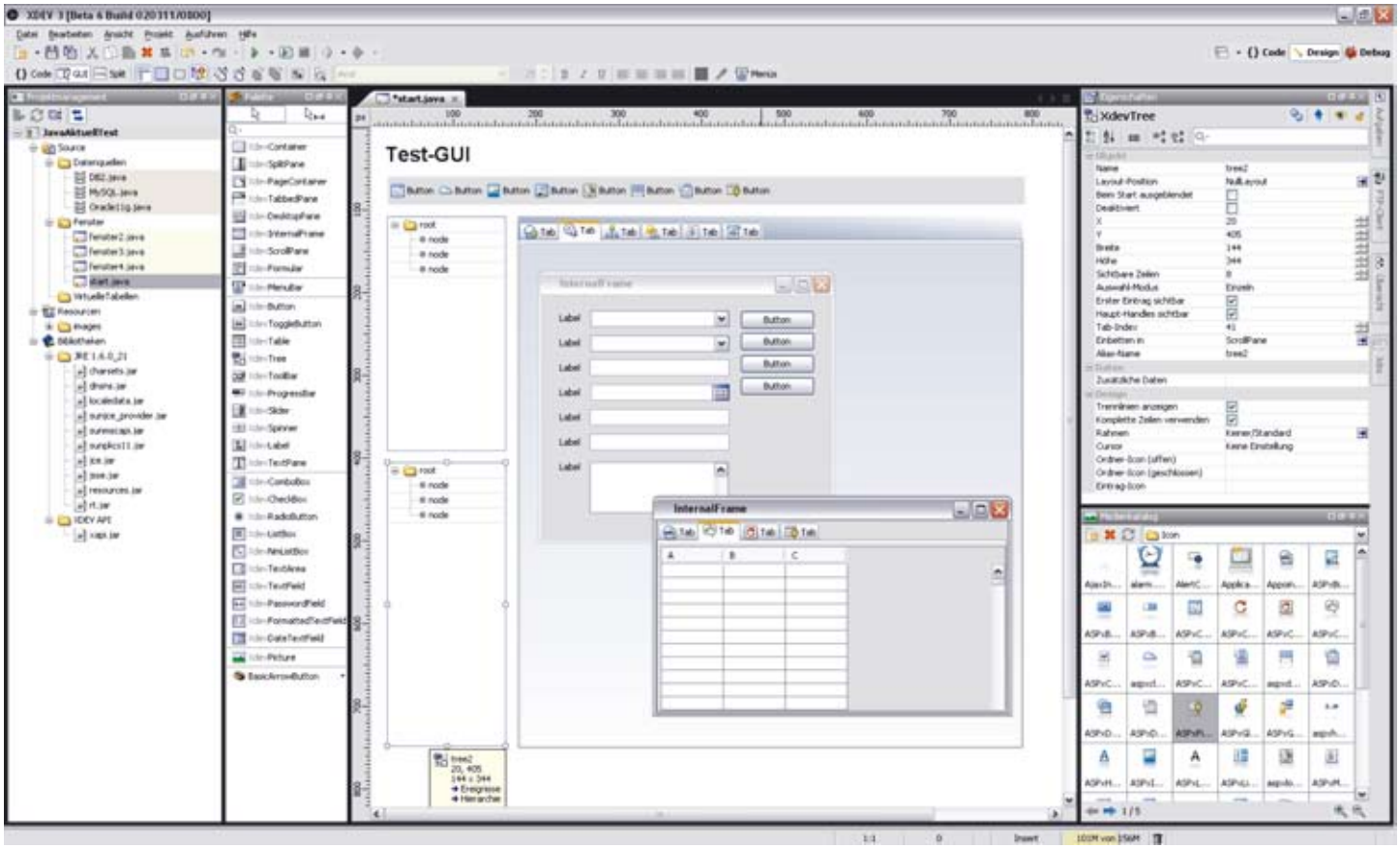


Abbildung 1: Swing GUI-Builder funktioniert wie ein Grafikprogramm

zeiten im Vergleich zur klassischen Java-Programmierung drastisch reduzieren. Vor allem bei GUI-Entwicklung sind zwischen 60 bis 90 Prozent kürzere Entwicklungszeiten möglich. Unweigerlich drängt sich der Vergleich mit Microsoft Visual Studio LightSwitch auf, das die Nachfolge von Access und FoxPro antreten soll. Somit existiert nun auch für Java eine vergleichbare und mehr als gleichwertige Lösung.

Der Artikel konzentriert sich auf eine neue Version, deren Entwicklung weiter fortgeschritten ist und die kurz vor dem Final Release steht. Seit der letzten Ausgabe hat sich hier eine Menge getan und vor allem die XDEV Component Suite verspricht überwältigende neue Features für die Oberflächen-Entwicklung mit Java Swing. Gleichzeitig erspart die Suite dem Entwickler eine Menge an Codierarbeit und Entwicklungszeit. Im Lieferumfang enthalten sind ein Richtext-Editor, mehrere Datagrids mit konfigurierbaren Sortier- und Filterfunktionen, ein vollständiger Kalender, ein GUI-Docking-Framework, eine GUI-Persistence-API, ein Format-Assistent für Tabellen sowie zahlreiche wichtige Funktionen wie Autovervollständigung und Quickfinder. Ein Databinding zwischen grafischer Oberfläche und Datenschicht ist standardmäßig mitgeliefert.

Formulare und Master-Detail-Filter generieren lassen

Zu den wichtigsten neuen Standardfeatures zählen vor allem die neuen Funktionen des GUI-Builders wie Formular-Generator, Master-Detail-Ansichten, das neue Konzept zur Umsetzung mehrsprachiger Oberflächen sowie die überarbeitete JavaBean-Schnittstelle für die Einbindung eigener und externer GUI-Komponenten. Das automatische Generieren von Formularen war bereits in der Vorgängerversion möglich. Neu ist, dass nun auch Speicher- und Suchfunktion mit generiert werden. Beim Speichern wird dabei automatisch unterschieden, ob ein Datensatz neu angelegt oder editiert wurde und somit per Insert- beziehungsweise Update-Statement in der Datenbank persistiert werden muss. Die Persistierung kann auch mittels OR-Mapping erfolgen. Damit ist der Einsatz sämtlicher Technologien möglich, die für Java verfügbar sind.

Master-Detail-Ansicht ist eine häufig verwendete Methode zur Filterung größerer Datenmengen. Solche Konstruktionen lassen sich nun schnell und einfach durch Verknüpfen zweier Ausgabe-Komponenten umsetzen. So kann etwa eine Combo-Box, die Automobile-Hersteller enthält, mit einer Tabelle verknüpft werden, welche die entsprechenden Fahrzeugmodelle dazu auflistet. Die Verknüpfung erfolgt per Mausklick im GUI-Builder. Anhand der in einem ER-Diagramm definierten Relationen kann die IDE die gesamte Geschäftslogik samt Datenbankabfragen dazu generieren. Auch deutlich komplexere Konstruktionen sind sehr leicht umsetzbar.

Mehrsprachige Oberflächen

Mehrsprachige Oberflächen lassen sind mit Textvariablen realisieren, die man an

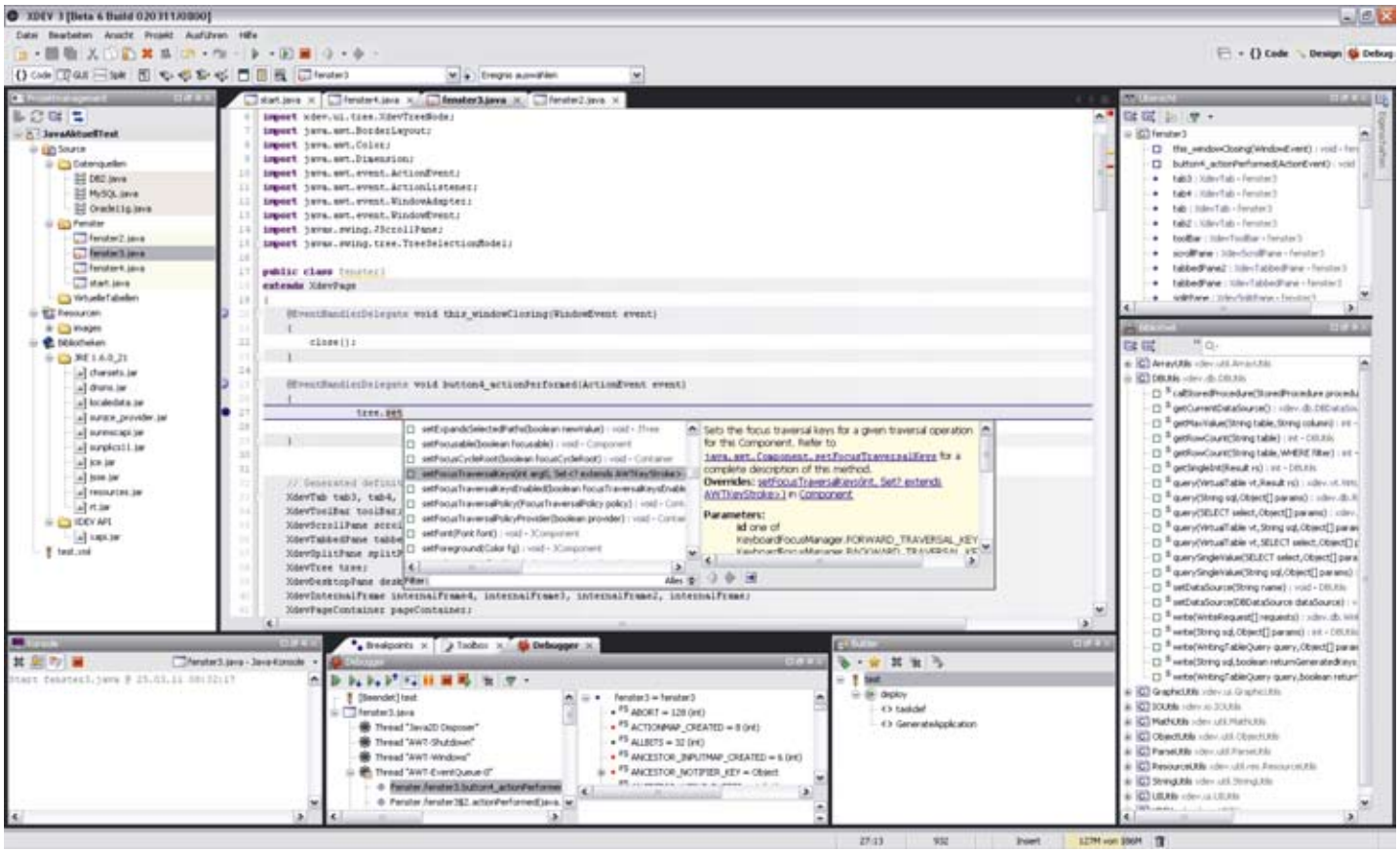


Abbildung 2: Unter der Haube wird Javacode generiert. Der Editor bietet alle wichtigen Profi-Features.

jeder Stelle der Oberfläche verwenden kann, beispielsweise für die Beschriftung von Formularen und Tabellenköpfen oder für Erklärungen und Hinweistexte. Die Vorgehensweise ist einfach. Auf der Oberfläche verwendet man nur noch Textvariablen, die in geschweiften Klammern und mit „$"-Zeichen angegeben werden, z.B. „{$save}". Die tatsächlichen Texte werden in Sprachdateien ausgelagert. Dabei handelt es sich um gewöhnliche Textdateien, die als „project_de.properties" benannt und als Ressource über das Projektmanagement eingebunden werden. Je nach Sprache muss das entsprechende ISO-Länderkürzel „de" entsprechend ersetzt werden. Die Oberflächentexte selbst werden in Form von Variable-Wert-Paaren zeilenweise in die Sprachdatei geschrieben, etwa „button4 = Speichern". Zur Laufzeit ermittelt die Applikation über die System-Einstellungen des Anwenderrechners die eingestellte Sprache, liest anhand des Länderkürzels die dazugehörige Sprachdatei aus und mappt die darin enthaltenen Texte mithilfe der Textvariablen auf die Oberfläche. Auch das Umschalten der Sprache ist zur Laufzeit möglich.

Der Entwickler kann jetzt eigene GUI-Komponenten schreiben und externe GUI-Beans nahtlos einbinden, sodass sich die Komponentenpalette des GUI-Builders nahezu grenzenlos erweitern lässt. Dafür stellt die IDE eine komfortable JavaBean-Integration zur Verfügung. Die verwendeten GUI-Beans werden direkt im GUI-Builder ausgeführt, wobei eine Instanziierung nur dann vorgenommen werden kann, wenn die Bean nicht auf externe Ressourcen angewiesen ist, die bei der Entwicklung gegebenenfalls nicht zur Verfügung stehen.

Eigene Komponenten bislang nur etwas für Profis

Wer von Tools wie Oracle Forms auf Java umsteigt, wird zunächst viele komfortable GUI-Funktionen vermissen, denn Java Swing bietet nur einen recht dürftigen Umfang an Standardkomponenten. JTable, JTree, JTabbedPane und JSplitPane sind dabei schon die Highlights. Die Entwickler von Swing sahen für eine größere Auswahl offenbar keine Notwendigkeit, denn durch seine MVC-Architektur (Model-View-Control) ist Swing wie keine andere Grafik-API zuvor bestens für individuelle Erweiterungen ausgelegt, sodass sich in Swing nahezu jedes denkbare GUI-Feature umsetzen lässt. Das Problem dabei ist allerdings, dass die Komponenten-Programmierung ohne umfassendes Expertenwissen in Swing praktisch nicht möglich ist. Die Entwicklung einer TreeTable, welche die Daten nicht nur wie eine herkömmliche Tabelle auflistet, sondern diese zudem wie einen Tree gruppieren kann, ist selbst für erfahrene Java-Entwickler eine echte Herausforderung. Ganz abgesehen von dem enormen Zeitaufwand, der dafür erforderlich ist. XDEV 3 bietet mit seinen von Swing abgeleiteten Komponenten (Xdev-

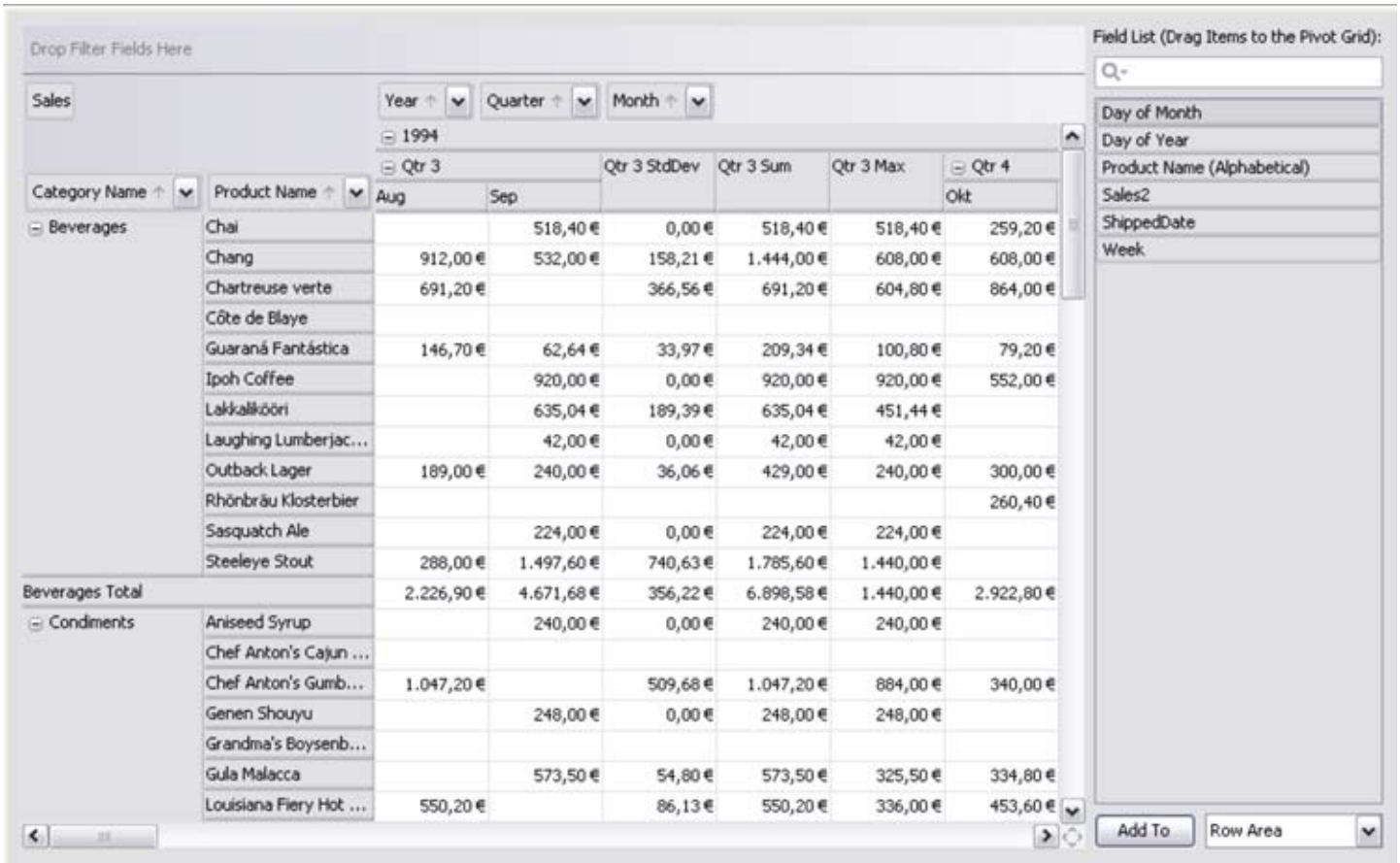


Abbildung 3: PivotGrid-Komponente ermöglicht die Abbildung Excel-ähnlicher Anwendungsfälle

Table, XdevTree etc.) zwar standardmäßig schon viele nützliche Erweiterungen (Databinding, Lokalisierung, Validierung etc.), doch in vielen Geschäftsanwendungen wird mittlerweile deutlich mehr erwartet, als Swing standardmäßig bieten kann. Um diese Anforderungen gemäß RAD-Philosophie auch schnell und einfach erfüllen zu können, bietet die XDEV Software Corp. jetzt eine Component Suite mit vielen leistungsfähigen Zusatz-Komponenten. Diese lassen sich per Installation in den GUI-Builder aufnehmen und somit wie Standard-Komponenten per Drag&Drop in jede Oberfläche einbinden, sodass auch Java-Einsteiger damit problemlos zurechtkommen.

Für die tabellarische Ausgabe von Datensätzen bietet die Component Suite eine SortableTable, mit der sich nach mehreren Spalten sortieren lässt, sowie Group- und TreeTable, mit denen Datensätze gruppiert werden können. Die GroupTable erlaubt dem Anwender sogar, Eingabefelder und Comboboxen in die Table zu ziehen, um spaltenbezogene Filter selber zu konfigurieren. Auch komplexe Pivot-Tabellen sind geplant.

Bei allen Grids lassen sich über ein Popup-Menü einzelne Spalten gezielt ein- und ausblenden. Ergänzend dazu kann im Databinding ein Cell-Style-Renderer implementiert werden. Damit lassen sich bei der Datenausgabe Rahmen, Schriftart, -größe und -farbe sowie Formatierungen und Hintergrundfarben individuell setzen. Die verschiedenen Datagrids lassen sich nicht nur autark verwenden, sondern auch per Mausklick mit anderen Komponenten verknüpfen, sodass diese interagieren. Durch die Verknüpfung mit einem Formular werden selektierte Datensätze automatisch in das Formular übertragen, sodass diese editiert werden können. Jedes Formular kann sowohl als Eingabemaske als auch als Suchformular verwendet werden. Mit der PageNavigationBar lassen sich ein Paging realisieren und das Abfrageergebnis datensatzweise durchblättern. Die dazu notwendige Programmlogik wird vollständig vom GUI-Builder generiert.

Unkompliziertes Databinding

Swing-Entwickler müssen gewöhnlich einen großen Aufwand betreiben, um Daten

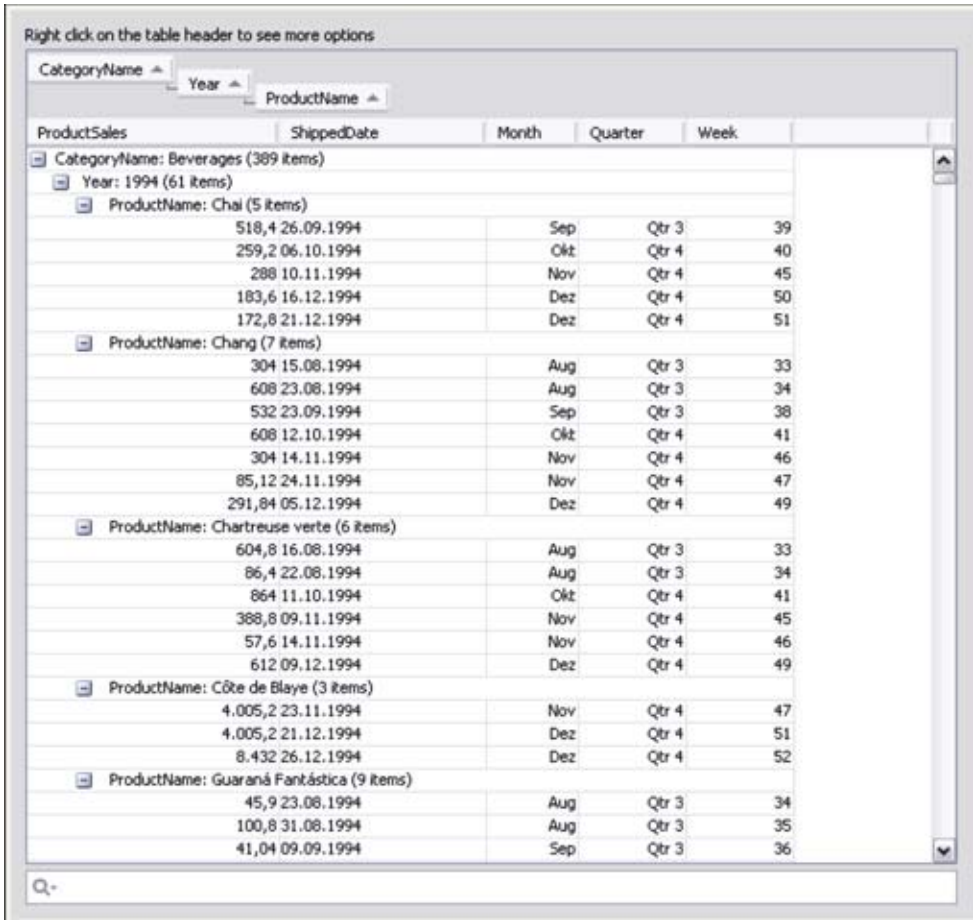


Abbildung 4: GroupTable-Komponente erlaubt individuelle Filter-Konfigurationen per Drag&Drop

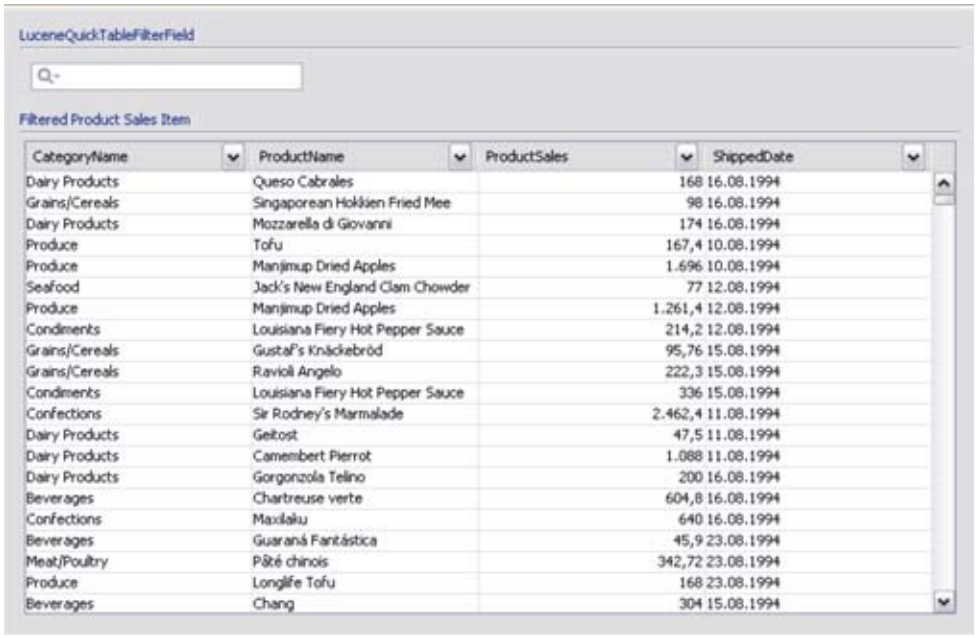


Abbildung 5: Quickfinder und Autovervollständigung lassen sich mit Grids verknüpfen

aus der Datenbank auf eine Oberfläche zu bekommen. Zuerst werden die Daten aus dem Abfrageergebnis (Resultset) ausgelesen. Anschließend muss für jede GUI-Komponente ein spezielles Daten-Modell erzeugt werden, beispielsweise ein Table- oder TreeModel. Für formatierte Ausgaben wird zudem ein Renderer benötigt.

Das quelloffene Application-Framework, das den Kern des RAD-Konzepts von XDEV 3 bildet, stellt alternativ sogenannte „virtuelle Tabellen" zur Verfügung, womit der ganze Arbeitsprozess radikal vereinfacht wird. Anstatt mit unterschiedlichen Daten-Modellen hantieren zu müssen, kann man jeder GUI-Komponente ein und dieselbe virtuelle Tabelle (Instanz) zuweisen. Die Zuweisung kann sowohl im GUI-Builder per Drag&Drop erfolgen als auch programmatisch durch den Aufruf der Methode „setModel (virtuelle Tabelle)". Das eigentliche Daten-Modell wird im Hintergrund automatisch erzeugt. Virtuelle Tabellen bilden damit eine Abstraktionsschicht zwischen GUI- und Datenschicht. Passend dazu werden auch Resultsets automatisch ausgelesen und praktischerweise gleich als virtuelle Tabelle zurückgegeben. Um ein Abfrageergebnis in einer Table anzuzeigen, muss der Entwickler somit keine Zeile Code mehr selber schreiben. Auch den Komponenten der XDEV Component Suite, für deren Verwendung man zum Teil sehr komplexe Daten-Modelle und Renderer erzeugen müsste, etwa für die TreeTable oder Kalender-Komponente, lassen sich die Daten bequem per virtuelle Tabelle übergeben. Aber nicht nur für Komponenten, sondern auch für Charts, Reports und Datenexporte kann eine virtuelle Tabelle die Datenquelle sein. Mit etwas Programmiererfahrung in Swing kann man auch eigene Adapter schreiben, um beliebigen GUI-Beans und Datenstrukturen unkomplizierte virtuelle Tabellen als Datenquelle übergeben zu können.

Für die Persistierung der Daten stellt die virtuelle Tabelle Methoden zur Verfügung, die bereits standardmäßig Transaktionssicherheit bieten. Alternativ dazu kann man diese Aufgabe jedoch auch einem Enterprise-Framework wie Hibernate, Spring oder Enterprise JavaBeans überlassen, was vor allem bei größeren Projekten viele weitere Vorteile haben kann. SQL-Profis dagegen möchten die entsprechende Logik möglicherweise in die Datenbank auslagern. Grundsätzlich sind alle drei Varianten möglich.

Flexibel konfigurierbare Oberflächen

Als besonders benutzerfreundlich gelten Applikationen, die mit Docking-Funktionen auf der GUI glänzen können, denn damit können Anwender ihre Oberfläche völlig individuell selber zusammenstellen. Mithilfe intelligenter Andock-Funktionen lassen sich damit einzelne Programm-

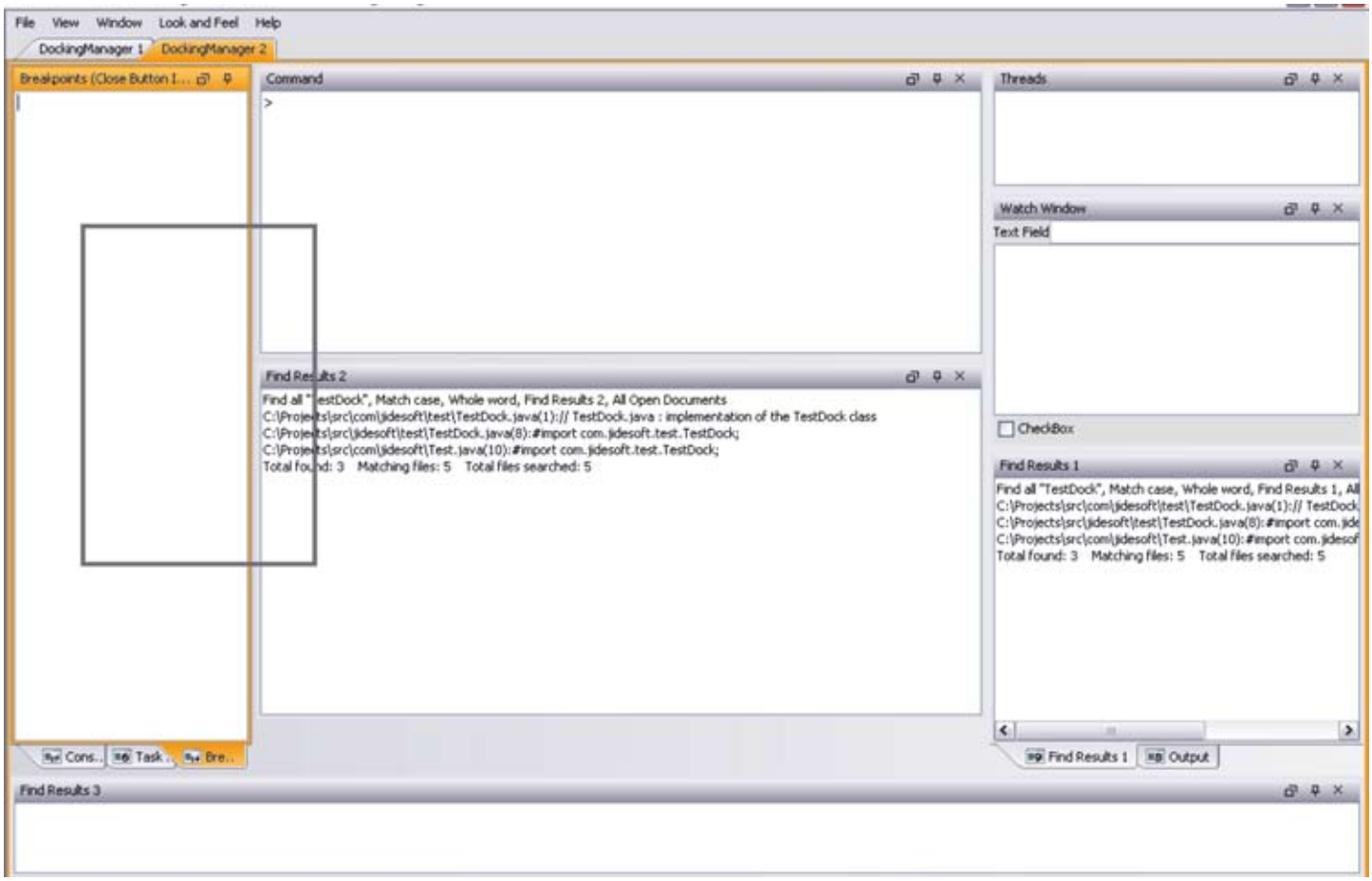


Abbildung 6: Docking-Framework ermöglicht Anwendern die freie Aufteilung einer Oberfläche und Speicherung der Einstellungen

fenster sowie Toolbars unabhängig voneinander auf dem Bildschirm verschieben und an einer beliebigen Position wieder einrasten, wobei sich das komplette Layout automatisch neu organisiert. Dieses Feature würde man sicher gerne jeder Anwendung spendieren, doch mit konventioneller Layout-Manager-Programmierung allein ist das nicht umsetzbar. Man braucht dazu ein sogenanntes „Docking-Framework". Die entsprechende Funktionalität selber zu implementieren käme einer Herkules-Aufgabe gleich. Hier ist das auch nicht nötig, denn die Component Suite liefert auch ein vollständiges Docking-Framework mit.

Die Einbindung im GUI-Builder ist recht einfach, denn dort konstruierte Fenster können nicht nur als gewöhnliches Programmfenster, sondern auch als Docking-Window angelegt werden. Das komplette Docking-Framework ist bereits im GUI-Builder voll funktionsfähig, sodass sich die Startkonfiguration im Handumdrehen erstellen lässt.

Eine wichtige Anforderung an eine moderne Geschäftsanwendung, die mit dem Einsatz eines Docking-Frameworks noch verstärkt wird, ist die Möglichkeit, sämtliche auf der Oberfläche vorgenommenen Nutzereinstellungen abspeichern zu können, unter anderem veränderte Spaltenbreite in Tabellen, Splitter-Positionen, ein- oder aufgeklappte Trees, selektierte Registerreiter und ToggleButtons, Position und Größe von Internal Frames sowie Fenster in einem Docking-Framework. Da dies keine triviale Aufgabe für den Anwender darstellt, bietet XDEV 3 auch dafür die komplette Funktionalität und bringt Standard-Implementierungen für eine dateibasierte XML-Serialisierung und die Java-Preferences-API mit. Bei weitergehenden Anforderungen lässt sich die Speicherstrategie auch beliebig vom Java-Entwickler implementieren.

Vollständige Kalender-Komponenten

Das Highlight ist eine vollständige Kalender-Komponente mit einer sehr ansprechenden Oberfläche und verschiedenen Ansichten für Tag, Woche, Monat und Jahr. Die Ansicht lässt sich modular aufbauen und via Properties entsprechend einfach konfigurieren, während die Kalenderdaten am einfachsten mit virtuellen Tabellen übergeben werden können. Auch das gleichzeitige Abbilden mehrerer Kalender ist möglich. Sämtliche Benutzerfunktionen wie das Anlegen, Verlängern, Verkürzen und Verschieben von Terminen mit der Maus werden automatisch von der Controllerschicht erledigt, sodass man dafür keine Zeile Code mehr schreiben muss. Um Änderungen zu speichern, muss lediglich das Daten-Modell persistiert werden, in der Regel eine virtuelle Tabelle. Erfahrene Entwickler können natürlich über die umfangreiche API in alle Details der Komponente eingreifen.

Abbildung 7: Vollständige Kalender-Komponente lässt sich schnell und einfach einbinden

Auch ein vernünftiger Text-Editor ist in vielen Geschäftsanwendungen unverzichtbar. Die Suite bietet daher einen voll funktionsfähigen Richtext-Editor auf Grundlage von JTextEditorPane, den man nur noch in seine Oberfläche einbinden muss. Die Symbolleiste des Editors lässt sich individuell konfigurieren. So können Formatierungen wie „fett", „kursiv" oder „unterstrichen" zugelassen oder gezielt ausgeblendet werden.

Neben Komponenten stellt die Component Suite auch zahlreiche Einzelfunktionen zur Verfügung, die eine Menge Codierarbeit ersparen, unter anderem Autovervollständigung und Quickfinder. Per Decorator Pattern können diese Features sehr leicht anderen GUI-Komponenten zugewiesen werden. Die entsprechenden Einstellungen lassen sich im GUI-Builder über die Properties der jeweiligen Komponente vornehmen. Auch ein Daten-Export nach Microsoft Excel steht zur Verfügung, ohne den kaum noch eine Firmenanwendung auskommen dürfte.

In Bezug auf das Deployment hat der Entwickler mittlerweile die Wahl zwischen einer klassischen Fat-Client-, Client-Server- oder modernen Java-Webstart-Applikation, welche jeweils automatisch unter Windows, Linux und auf dem Mac lauffähig sind. Alternativ ist auch die Generierung einer Web-Anwendung (Rich-Internet-Appliation – RIA) möglich, die sich über das Internet aufrufen und im Web-Browser ausführen lässt. Weitere Anpassungen sind dafür nicht erforderlich. Die Build-Skripte auf Basis von Ant werden automatisch generiert.

Nach der Freigabe des Final Release soll rasch eine Versionsverwaltung auf der Grundlage von Subversion folgen. Auch das Deployment soll erweitert werden, sodass man in Kürze mit XDEV 3 auch Ajax-Applikationen auf Basis von Vaadin und Google Web-Toolkit generieren kann, welche dann in sämtlichen Browsern auch ohne Java–Plug-in lauffähig wären.

Fazit

Die Entwicklung von Geschäftsanwendungen mit Java-Editoren wie Eclipse, NetBeans und JDeveloper gilt als kompliziert, zeitintensiv und war bislang ohne langjährige Java-Erfahrung in der Praxis nicht möglich. XDEV 3 bietet auch für Java ein Rapid Application Development, welches die Anwendungsentwicklung mit Java radikal vereinfacht und dennoch professionellen Ansprüchen gerecht wird.

Durch die große Ähnlichkeit zu 4GL-Lösungen wie Microsoft Access und Oracle Forms eignet sich die Umgebung vor allem für die Entwicklung datengetriebener Business-Applikationen und ermöglicht jedem 4GL-Entwickler den sofortigen Umstieg auf Java.

Mit dem GUI-Builder lassen sich Java-Oberflächen so schnell und detailgenau umsetzen, dass sich damit auch umfangreiche Altsysteme in erstaunlich kurzer Zeit auf Java migrieren lassen, während Datenbank und Geschäftslogik, sofern in der Datenbank vorhanden, weitestgehend erhalten bleiben können. Mit der Component Suite stehen viele weitere GUI-Features zur Verfügung, die Java standardmäßig nicht bietet, unter anderem Kalender, Datagrids, Richtext-Editor sowie wichtige GUI-Funktionen wie Autovervollständigung, Quickfinder, Persistierung der User-Einstellungen auf der GUI sowie Excel-Output. Damit lassen sich die Entwicklungszeiten vor allem bei GUI-lastigen Projekten um 60 bis 90 Prozent reduzieren, während sich Entwicklungskosten und Projektrisiko deutlich verringern.

Kontakt:
Gerald Kammerer
info@redaktion-kammerer.de

Gerald Kammerer ist als freier Redakteur seit 2006 für verschiedene Computerfachzeitschriften mit Schwerpunkt Java- und AJAX-Entwicklung tätig (siehe auch Seite 62).

Impressum

Herausgeber:
Interessenverbund der Java User Groups e.V. (iJUG)
Tempelhofer Weg 64, 12347 Berlin
Tel.: 0700 11 36 24 38
www.ijug.eu

Verlag:
DOAG Dienstleistungen GmbH
Fried Saacke, Geschäftsführer
info@doag-dienstleistungen.de

Chefredakteur (VisdP):
Wolfgang Taschner,
redaktion@ijug.eu

Chefin von Dienst (CvD):
Carmen Al-Youssef,
office@ijug.eu

Titel, Gestaltung und Satz:
Claudia Wagner,
DOAG Dienstleistungen GmbH

Anzeigen:
CrossMarkeTeam, Ralf Rutkat,
Doris Budwill
redaktion@ijug.eu

Mediadaten und Preise:
http://www.ijug.eu/images/vorlagen/2011-ijug-mediadaten_java_aktuell.pdf

Druck:
adame Advertising and Media GmbH Berlin
www.adame.de

Java aktuell
Magazin der Java-Community

Quo vadis Java?

Wolfgang Weigend, ORACLE Deutschland B.V. & Co. KG

Java ist eine der bekanntesten Marken der IT und eine der am häufigsten bereitgestellten Technologien. Sie wird durch eine Vielzahl von Anwendungen und Services repräsentiert, die in der Java-Sprache geschrieben sind. Mit der Übernahme von Sun wird Oracle mit Innovationen und Investitionen in die Java-Technologie zum Vorteil der Java-Community und der Kunden fortfahren. Oracle hat bei der Unterstützung von Java – seit der Einführung im Jahr 1995 – eine führende Rolle eingenommen und wird mit hohem Engagement und Klarheit dazu beitragen, die Java-Technologie zusammen mit der Community entscheidend voranzubringen.

Die Zahlen und Fakten unterstreichen die hohe Verbreitung der Java-Technologie-Plattform mit weltweit mehr als neun Millionen registrierten Java-Entwicklern, 930 Millionen Java Runtime Environment (JRE) Downloads pro Jahr, mehr als einer Milliarde Java-Desktops, drei Milliarden mobilen Endgeräten mit Java und 1,4 Milliarden Java-Cards, die Jahr für Jahr hergestellt werden. Java führt die Liste der am häufigsten verwendeten Programmiersprachen an und ist in allen Schulen und Universitäten fester Bestandteil der Ausbildung. Darin liegt auch die Stärke von Java: Die Community vergrößert sich und tauscht sich global über die Kodierung aus, sodass ohne komplexe Einarbeitung der Programmiercode gelesen, nachvollzogen und verändert werden kann. Aus diesem Grund wird der freie Zugang zu aktuellen Informationen für die Weiterentwicklung der Java-Community uneingeschränkt zur Verfügung gestellt.

Die einheitliche Entwicklungsbasis bildet das offene und frei verfügbare OpenJDK als zentrale Grundlage für die Aktivitäten der Java Standard Edition 7 (Java SE 7) und der Java Standard Edition 8 (Java SE 8). Java ist der technologische Ausgangspunkt der meisten Hard- und Software-Hersteller und bildet auch die Basis für die Oracle Fusion Middleware mit den Fusion Applications. Dies verdeutlicht auch das Geschäftsmodell für die Java-Entwickler, das anhand der gelernten Programmiersprache und der frei zugänglichen Java-Technologie die von ihnen erstellte Programmierlogik in Form von Anwendungen und neuen Services in die kommerzielle Vermarktung bringt. Die Verwendung von Java in Open-Source-Projekten macht einen Großteil der IT-Landschaft aus, bietet doch der kommerzielle Einsatz des Java-Programmier-Codes die Möglichkeit einer Einnahmequelle für die Entwickler.

Bereits bei der Verwendung von OpenJDK ist der Entwickler integraler Bestandteil einer klar umrissenen Java-Strategie. Die neuen OpenJDK-Community-Richtlinien wurden in Zusammenarbeit mit John Duimovich (IBM), Jason Gartner (IBM), Mike Milinkovich (Eclipse), Prof. Doug Lea (State University NY Oswego) und Adam Messinger (Oracle) erstellt und führen die Arbeiten des OpenJDK-Interim-Governance-Boards weiter. Das Gremium hat Regeln aufgestellt, die den langfristigen Bestand und das Wachstum der OpenJDK-Community fördern und sicherstellen, dass die Mitglieder in klarer und offener Weise agieren und die administrative Governance nach dem Leistungsprinzip erfolgt. So wird ein hohes Qualitätsniveau für das OpenJDK erreicht. An dessen Weiterentwicklung sind neben Oracle große Hersteller wie IBM, Apple, HP und VMware beteiligt. Alle setzen auf die einheitliche Java-Plattform, die aus der Java-Sprache, der Java Virtual Machine (JVM) und den Java-APIs für unterschiedliche Funktionalitäts- und Hardware-Anforderungen wie Java Enterprise Edition (Java EE), Java Standard Edition (Java SE) und Java Micro Edition (Java ME) besteht (siehe Abbildung 1).

Java Virtual Machine und Java Development Kit

Durch die Akquisition von Sun Microsystems durch Oracle sind die beiden virtuellen Java-Maschinen (JVM) „HotSpot“ und „JRockit“ unter einem Dach. Hotspot JVM ist allgemein einsetzbar, parametrisierbar, qualitativ hochwertig und am Markt sehr stark verbreitet. JRockit JVM ist eine spezielle Server-JVM mit hoher Leistungsfähigkeit, die für den Serverbereich und die Oracle Fusion Middleware optimiert wurde. Mit JRockit verbessert sich das Laufzeitverhalten von Anwendungen, die effizienter mit den Ressourcen Speichergröße, Anzahl der Threads, Netzwerk sowie I/O-Aktivitäten umgehen und durch deterministische Garbage-Collection und minimale Pausenzeiten niedrige Antwortzeiten erreichen. Zudem werden alle Bestandteile der JRockit JVM (Code Generierung, Speicher-Management, Thread-Management, I/O, Reflection) optimiert. Die beiden JVMs JRockit und HotSpot werden in einem mehrjährigen Entwicklungsprozess zu einer einheitlichen JVM verschmolzen, die sich dann aus den besten Funktionsmerkmalen beider JVMs zusammensetzen wird. Die Arbeitsergebnisse fließen inkrementell dem OpenJDK-Projekt zu, wobei die JRockit-Funktionalität den bisher größten Code-Beitrag zum OpenJDK darstellt. Vorhandene JRockit-basierte Produkte wie JRockit Mission Control, JRockit Real Time und JRockit Virtual Edition bleiben separat und werden kommerziell lizenziert. Das Java Development Kit (JDK) und Java Runtime Environment (JRE) stehen weiterhin kostenfrei zur Verfügung und Oracle wird eine offene Java-Implementierung (OpenJDK) dauerhaft unterstützen, die reine Open-Source-Komponenten beinhaltet.

Mit dem zentralen Einstiegspunkt von OpenJDK gelangen die erreichten Ergebnisse in die Java SE 7 und in die Java SE 8. Diese Java-SE-Versionen bieten eine hö-

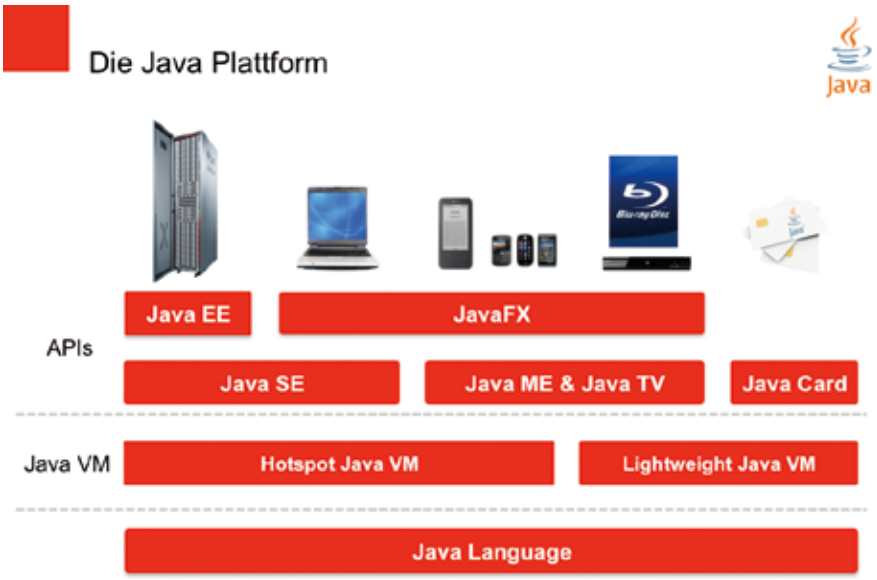


Abbildung 1: Die Java-Plattform

here Entwicklerproduktivität, bessere Ausnutzung von Multi-Core-Prozessoren sowie die Unterstützung großer Hauptspeicher und Hochgeschwindigkeitsnetze. Das JDK 7 ist funktional nahezu fertig. Der „Feature Complete"-Status ist erreicht und der Plan wurde auf openjdk.java.net veröffentlicht. Der Funktionsumfang vom JDK 7 lässt sich mit NetBeans 7 beta und GlassFish 3.1 testen. JDK 7 umfasst im Wesentlichen Sprachverbesserungen aus dem Projekt Coin, die Concurrency- und Collections-Updates und die Unterstützung für dynamisch typisierte Sprachen. Die Freigabe des JDK 7 ist für den 28. Juli 2011 geplant, JDK 8 soll Mitte 2012 zur Verfügung stehen. Die wichtigsten Inhalte des JDK 8 werden die Java-Plattform-Modularisierung und die Lambda-Expressions (Closures) sein. Vorab hatte sich das JCP-SE/EE-Executive-Committee in der Abstimmung mit teils deutlichen Mehrheiten für die jeweiligen Java Specification Requests entschieden: JSR 334 „Small Enhancements to the Java Programming Language", JSR 335 „Lambda Expressions for the Java Programming Language", JSR 336 „Java 7 SE Release Contents" und JSR 337 „Java 8 SE Release Contents".

Java Micro Edition

Oracle beginnt mit der Modernisierung der Java-Micro-Edition-Plattform (Java ME) und arbeitet mit der Java-Mobile-Community gemeinsam am Projekt Java ME.next, der evolutionären Weiterentwicklung der neuen Version von Java ME. Ziel ist es, die zugrundeliegende Sprachspezifikation zu aktualisieren und moderne Geräte-/Hardware-Funktionalität wie Near Field Communication, IP Multimedia Subsystem (IMS), Sensoren, Telefonie und Lokation durch neue Java-APIs besser zu unterstützen. Wie auf dem Java-Client werden sowohl native Java-Anwendungen als auch auf Web-Technologien basierende Anwendungen unterstützt, die programmatisch untereinander kommunizieren können.

JavaFX

Für die Entwicklung von Rich-Internet-Anwendungen (RIA) mit Unterstützung von Multimedia und modernen Hardware-GPUs (Graphics Processing Unit) wird JavaFX angeboten. Es besteht jetzt noch aus der JavaFX-Script-Sprache, den JavaFX-Script-APIs sowie den Runtime-Libraries und läuft auf der Java Virtual Machine. Für die neue JavaFX-Plattform kommt ein Sprachwechsel. Damit wird in JavaFX 2.0 JavaFX-Script nicht mehr fortgeführt. Die JavaFX-APIs werden künftig in Java implementiert. Dadurch stehen viele Vorteile der Java-Plattform wie Generics, Annotations und Multithreading unmittelbar auch für JavaFX zur Verfügung. Java-Programmierer können JavaFX nutzen, ohne eine weitere Programmiersprache lernen zu müssen und es wird leichter sein, JavaFX innerhalb von Swing zu benutzen. Andere Scripting-Sprachen wie JRuby, Groovy und Scala, die auf der JVM laufen, sind auch für JavaFX-Anwendungen einsetzbar und stellen vergleichbare Merkmale wie JavaFX-Script bereit.

Java Enterprise Edition 6

Java EE 6 bietet mit der Einführung von Java-EE-Profilen neue Funktionalität für Web-Anwendungen wie die Erweiterbarkeit/Plug-in-Fähigkeit, „Contexts and Dependency Injection for the Java EE Platform" (CDI), Managed Beans mit POJO-Modell, RESTful Web Services (JAX-RS), schichtenübergreifende Validierung (Bean Validation), erweiterte APIs mit EJB 3.1, JSF 2.0, JPA 2.0, Servlet 3.0 und verbesserte Nutzbarkeit durch Konventionen anstatt Konfigurationen (weniger XML) und Annotationenbasiertem Programmiermodell (decorate und inject). Damit wird Java EE 6 mit seinen Innovationen und neuen APIs leichtgewichtiger und flexibler und ist schneller von den Java-Entwicklern in Projekten einsetzbar. Der GlassFish-Anwendungsserver ist in der Java-EE-6-Referenz-Implementierung JEE-6-zertifiziert und der WebLogic-Server wird dies mit dem nächsten Major-Release ebenfalls sein (siehe Abbildung 2).

Java Enterprise Edition 7

An der nächsten Version der Java Platform Enterprise Edition 7 (Java EE 7) wird bereits gearbeitet (JSR 342). Alle Executive-Committee-Mitglieder für SE/EE haben dem JSR 342 zugestimmt, um die Inhalte von Java EE 7 zu entwickeln. Der Schwerpunkt des neuen Java EE 7 Release ist die Cloud. Die Java-EE-Plattform wird sich besser an die Anforderungen von Cloud-Umgebungen anpassen und einfacher mit Private und Public Clouds zusammenspielen. Java EE 7 wird die Funktionalität als Service unterstützen, um Mandantenfähigkeit und horizontale Skalierbarkeit (Elasticity) abzudecken. Java-EE-7-Anwendungen werden damit die Vorteile von Cloud-Umgebungen besser ausnutzen können. Im „Cloud Platform as a Service (PaaS)"-Modell können verschiedene Anwendungskomponenten mit unterschiedlichem Quality-of-Service und Sicherheitsbereichen getrennt voneinander betrieben werden, wie es bei Mandantenfähigkeit notwendig ist. Bisher hatte Java EE eine Container-basierte Umgebung

Java-EE-7-Plattform: JSRs & Technologie

Folgende Java Specification Requests und Technologien werden verwendet und noch angepasst:

- Concurrency Utilities for Java EE (JSR-236)
- JCache (JSR-107)
- Java Persistence API (JPA)
- Java API for RESTful Web Services (JAX-RS)
- JavaServer Faces (JSF)
- Servlets
- Enterprise JavaBeans (EJB)
- JavaServer Pages (JSP)
- Expression Language (EL)
- Java Messaging Service (JMS)
- Java API for XML-based Web Services (JAX-WS)
- Contexts and Dependency Injection for Java EE (CDI)
- Bean Validation
- Dependency Injection for the Java Platform (JSR-330)
- Common Annotations (JSR-250)
- Java Connector Architecture
- Java Web Sockets API (TBD)
- Java JSON API (TBD)

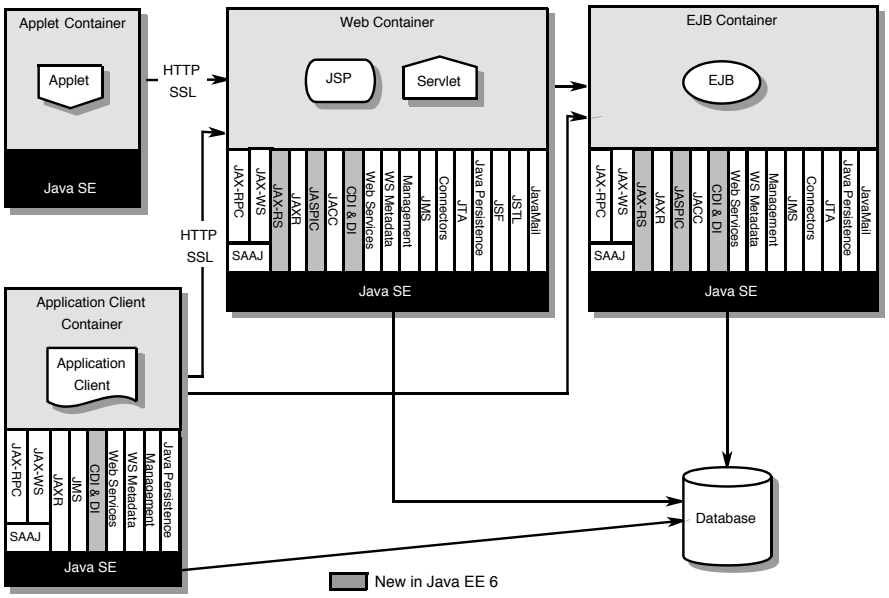


Abbildung 2: Java EE 6: Architektur-Diagramm

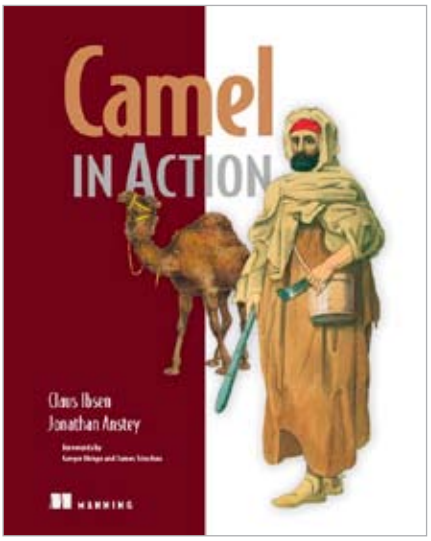
angeboten, die im Einzelbetrieb oder großen Cluster-Anwendungen den Zugriff auf das System oder externe Ressourcen steuerte, ohne das Programmiermodell ändern zu müssen. Hier wirkt der Cloud-Ansatz evolutionär, um einige inkrementelle Änderungen am existierenden Java-EE-Programmiermodell vorzunehmen. JSR 342 wird die Java-EE-Plattform-Architektur erweitern, um die Belange des PaaS-Betriebsmodells aufzunehmen. Dafür werden neue Rollen wie der PaaS-Administrator eingeführt und die Java EE Security-Architektur angepasst. Der JSR soll auch Voraussetzungen für Anwendungen bereitstellen, die PaaS-spezifische Funktionalität wie Mandantenfähigkeit nutzen möchten, sich selbst als Cloud-fähige Anwendungen ausweisen und sich an die definierten Regeln halten. Die verwendeten Plattformtechnologien müssen an die neuen Cloud-Modell-Anforderungen angepasst werden. Beispielsweise die Ressource-Manager-basierten APIs wie JPA, JDBC und JMS. Das Programmiermodell soll soweit verbessert werden, dass die meisten APIs verbindungslos agieren. Die in den Anwendungen verwendeten Modell-Ressourcen Data Sources und Message Queues werden für mehrere Anwendungen als „Shared" gekennzeichnet. Die Expert Groups steuern die Anforderungen für die einzelnen Technologien und bewerten sie, um deren konkrete Umsetzung sicherzustellen. Der JSR definiert einen Deskriptor für Anwendungsmetadaten, damit die Entwickler ihre Anwendungscharakteristik in der Paas-Umgebung beschreiben können. Diese Eigenschaften können wie folgt lauten: Mandantenfähigkeit, Ressource-Sharing, Quality-of-Service-Information, Anwendungsabhängigkeiten. Die Anwendungsmetadaten-Struktur wird erweiterbar sein und soll Standard-Metadaten-Attribute bereitstellen, die mit künftigen Java-EE-Plattform-Versionen erweitert werden.

Ausblick

- Die Java-EE-7-Plattform wird aktuelle Web-Standards wie HTML 5 und Web Sockets unterstützen und ein HTTP-Client-API mit JAX-RS 2.0 enthalten.
- Zur verbesserten Entwicklung mit Java EE 7 wird das überarbeitete JMS-2.0-API enthalten sein. Die Dependency-Injection-Struktur wird mit einem API zur Injector-Konfiguration erweitert, in DI 1.1 definiert und in CDI 1.1 unterstützt.
- Das Managed-Bean-Modell wird überarbeitet und verschiedene Inkonsistenzen, wie sie bei Managed Beans, EJB, Servlets, JSF, CDI und JAX-RS auftreten können, werden entfernt.
- Java EE 7 wird mit Java SE 7 aufgebaut und alle beteiligten Java-Spezifikationen werden analysiert und verbessert, um ihre Schnittstellen auf die kommenden Java-SE-8-Sprachänderungen auszurichten. Die Entwickler bekommen so den größtmöglichen Nutzen aus der neuen Funktionalität (siehe Kasten).
- Der Zeitplan für die Entwicklung der Java-EE-7-Plattform sieht ein Early Draft im dritten Quartal 2011 vor. Dem folgen ein Public Review im ersten Quartal 2012 und ein Final Release für JSR 342 im dritten Quartal 2012.

Kontakt:
Wolfgang Weigend
Wolfgang.Weigend@oracle.com

Wolfgang Weigend, Systemberater für die Oracle Fusion Middleware bei der Oracle Deutschland B.V. & Co. KG, ist zuständig für Java-Technologie und -Architektur mit strategischem Einsatz bei Großkunden. Er verfügt über langjährige Erfahrung in der Systemberatung und im Bereich objektorientierter Softwareentwicklung mit Java.



Camel in Action

gelesen von Kai Wähner

Der Datenaustausch zwischen Unternehmen oder einzelnen Bereichen wird immer intensiver. Die Zahl der Anwendungen, die integriert werden müssen, steigt stetig an. Die Schnittstellen nutzen diverse Technologien, Protokolle und Datenformate. Die Integration dieser Anwendungen soll aber trotzdem standardisiert modellierbar, effizient umsetzbar und automatisiert testbar sein.

Apache Camel ist ein Framework, welches diese Anforderungen erfüllt. Es implementiert die allseits bekannten Enterprise Integration Patterns (EIP) und bietet dadurch eine standardisierte, domänenspezifische Sprache, die für die Integration von Anwendungen eingesetzt wird. Neben Java können auch Spring XML, Scala oder Groovy für die Integration genutzt werden.

Das vor kurzem erschienene Buch „Camel in Action" erläutert die Konzepte und Einsatzmöglichkeiten von Camel auch anhand vieler Beispiele sehr ausführlich. Es kann sowohl für Einsteiger zum Erlernen des Frameworks als auch als Nachschlagewerk genutzt werden. Daher ist nicht weiter tragisch, dass keine Alternative zu diesem Buch verfügbar ist.

Die ersten zwei Kapitel ermöglichen einen einfachen Einstieg in das Thema. Sie geben einen Überblick über die grundlegenden Konzepte sowie das übliche „Hello-World"-Beispiel. Im Hauptteil des Buches sind die einzelnen Konzepte ausführlich erläutert, insbesondere Routing, Transformation, Fehlerbehandlung, Testbarkeit, die am häufigsten genutzten EIPs sowie die wichtigsten Komponenten für Schnittstellen (wie JMS, FTP, File, Mock). Danach wird beschrieben, wie Camel-Anwendungen mithilfe von Maven erstellt, in Web-Containern (wie Tomcat) oder Application Servern (wie JBoss AS) eingesetzt und mit JMX überwacht werden. Der Appendix enthält schließlich noch einige weitere interessante Themen, beispielsweise wie man Camel durch die Akka-Komponente sinnvoll mit der Programmiersprache Scala kombinieren kann.

Alle Kapitel enthalten gute und einfach verständliche Beispiele sowohl in Java DSL als auch in Spring XML. Der Quellcode inklusive den zugehörigen JUnit-Tests kann auf der Webseite des Buches heruntergeladen und direkt ausgeführt werden. Dadurch lassen sich schnell und einfach erste praktische Erfahrungen mit Camel sammeln.

Das Buch ist jedem Software-Entwickler zu empfehlen, der Systeme im JVM-Umfeld integrieren muss. Camel ist ideal geeignet, um zu erlernen, wie Enterprise Application Integration (EAI) effizient und wartbar umzusetzen ist. Nach dem Lesen des Buches sind alle wichtigen Konzepte bekannt und weitere Camel-Features können ohne große Probleme im Internet gesucht und in das eigene Projekt integriert werden. Ohne dieses Buch war es in der Vergangenheit deutlich schwerer und frustrierender, in Camel einzusteigen.

Positiv ist auch zu erwähnen, dass Forum-Support gegeben wird. Claus Ibsen, Autor des Buches und Camel-Committer, hilft bei Verständnisfragen und Problemen beim Ausführen der Code-Beispiele. Antworten werden fast immer innerhalb von 24 Stunden gegeben.

Für die nächste Auflage des Buches wären einige Beispiele mit kommerziellen Software-Produkten (wie WebSphere MQ) sowie ausführlichere Beispiele zum sinnvollen Einsatz von Groovy und Scala wünschenswert. Außerdem fallen Beispiele für manche Komponenten (wie zum Beispiel für XStream und JAXB) sehr kurz aus oder fehlen ganz. Trotz dieser kleinen Kritikpunkte ist dieses Buch jedem dringend zu empfehlen, der System-Integration effizient mit Camel umsetzen möchte.

Kontakt:
Kai Wähner
kai.waehner@mwea.de

Titel:	Camel in Action
Autoren:	Claus Ibsen, Jonathan Anstey, Hadrian Zbarcea
Verlag:	Manning, 2011
Umfang:	552 Seiten
Sprache:	Englisch
Preis:	36,80 €
ISBN-10:	1935182366

Kai Wähner ist als IT-Consultant bei „MaibornWolff et al" tätig. Seine Schwerpunkte liegen in den Bereichen JEE, EAI und SOA. Außerdem berichtet er auf „www.kai-waehner.de/blog" über seine Erfahrungen mit neuen Technologien, IT-Konferenzen und Zertifizierungen.

Die Integrationskraft von Java in der Praxis

Hans Niedermeier, PRO-Software GmbH

Als langjähriger Anwendungsentwickler mit dem Oracle-Designer-Developer schnuppert und arbeitet der Autor seit gut zwei Jahren in der Java-Welt. Was im Gegensatz zu den früher verwendeten Plattformen sehr positiv auffällt, ist die Integrationskraft von Java. Agierte man früher in relativ geschlossenen Technik- und Datenräumen, mit Austausch nur per Excel-CSV, Batch-Input in SAP, im besten Fall Database-Link in Oracle, kann man heute mit Java und den darin angebotenen Connectivity-Features über Datenbank und Systemgrenzen hinweg auf elegante Art integrierende Anwendungen schaffen.

Anhand eines Beispiels, das keineswegs fiktiv, sondern in den realen IT-Landschaften häufig zu finden ist, wird im Folgenden über Probleme, Lösungen, Techniken und Konzeptionen gesprochen:

- Ein System A bekommt Rechnungsdaten per Datenaustausch. Die erhaltenen Daten werden mittels einer Standardsoftware in eine dort integrierte Datenbank geschrieben, dort geprüft und weitergeleitet.
- Ein System B (in diesem Fall SAP) bekommt Daten aus A per Batch-Input zur Weiterverrechnung. Auch andere Prozesse in B wie Bestellungen korrespondieren mit importierten Daten aus A.
- Ein weiteres System C bezieht gefilterte Daten aus System B, um Einzelfall-bezogene Nachweise über Verbrauch und Kosten darstellen und verteilen zu können. System C seinerseits erschließt sich Daten von weiteren Systemen via Database-Link.

Die Sachbearbeiter in diesem Komplex müssen zwangsläufig in allen drei Systemen arbeiten und agieren, um die Geschäftsprozesse vollständig abwickeln zu können. Im Normalfall funktionieren die implementierten Automatismen zwischen den Systemen gut. Ein gewisser Rest, oftmals und zu Recht „Sumpf" genannt, erfordert aus verschiedensten Gründen Einzelfall-bezogene Recherchen. Den Sumpf trockenzulegen, also die Einzelfälle zu klären, entspricht einer Strafarbeit, auch wenn man drei Bildschirme nebeneinander zur Verfügung hat. Da die Systeme A und B Standardsoftware sind, ist eine Erweiterung zur Sumpfvermeidung nicht zu realisieren. System C ist eine geschlossen Oracle-Forms-Welt.

Lösungswege

Das Idealste wäre eine Software, die alle Funktionen der drei Systeme abdeckt. Da nur in den seltensten Fällen eine solche Software verfügbar ist, benötigt man eine individuelle Lösung. Es wird ein Budget besorgt und ein Projekt aufgesetzt, um eine übergeordnete System-Instanz, „ABC+" genannt, zu realisieren. Hauptziel ist, zur Klärung des Sumpfes möglichst nur eine Anwendung bedienen zu müssen. Wenn man die Geschäftsprozesse dann auch noch überwiegend aus ABC+ steuern könnte, wäre das ein Vorteil. Sind alle drei Systeme SOA-fähig, bietet sich ein weiterer Lösungsweg – aber nur eines der Systeme beherrscht dies theoretisch.

Java als Basis-Technologie für ABC+

Hohe Plattformunabhängigkeit und stabile Zugriffstechniken via JDBC auf alle gängigen Datenbank-Systeme sind hinlänglich bekannte Pluspunkte, die Java auszeichnen. Verbreitungsgrad und Größe der Fangemeinde unter den Software-Entwicklern versprechen eine langjährige Nutzungsdauer einer einmal entwickelten Anwendung. Weiterentwicklung scheint gesichert. Die Wahl fällt trotz möglicher Alternativen auf Java. Aus der Palette der Java-Workbenches ist die geeignete zu ermitteln. Da der Autor im konkreten Fall der Auswahl einer Java-Workbench frei von jeglichen Altlasten war, fiel die Entscheidung zugunsten XDEV2 aus – Grund war der RAD-Ansatz (Rapid-Applications-Umfeld-Development). Ohne übermächtigen Lernaufwand lassen sich Anwendungen erzeugen. Die Tatsache, dass RAD-Ansätze in vielen Fällen einschränkend wirken, trifft bei XDEV nicht zu. Mit XDEV3 entstehen pure Java-Anwendungen, die man 1:1 in andere Workbenches wie Eclipse übertragen kann. Noch zu bemerken ist, dass zum Zeitpunkt der Entscheidung für XDEV2 von ABC+ nichts bekannt war, jedoch ein Werkzeug geschaffen werden sollte, um schnell, agil und damit auch kostengünstig Systeme entwickeln zu können.

Trotz der schnell erfolgten Auswahl der XDEV2-Workbench hatte der Autor noch nichts gewonnen, noch keine einzige Anwendung geschaffen. Eine Strategie für zukünftige Projekte, deren schnelle und kostengünstige Fertigstellung wurde gesucht und schnell gefunden, angesichts in der Vergangenheit gemachter Erfahrungen im Projektgeschäft. Verfügbare Zeit und ein gewisses Maß an Überzeugung veranlassten den Autor, als Erstes ein allgemeines Basissystem für datengetriebene Anwendungen zu planen und zu realisieren. Ziel dieses Systems ist, den wesentlichen Teil von GUI-Frontends, also Komponenten wie Tree, Table, Formular, Buttons, Pivot-Blatt etc. – und diese in beliebigen Kombinationen mit fertigen Synchronisations-Mechanismen versehen, nur einmal mit Java-Code zu implementieren und anschließend in allen Anwendungen zu benutzen. Zum Zeitpunkt der Kompilierung und des Deploy-Vorgangs ist es nicht erforderlich, die Applikations-Daten der zukünftigen Anwendung zu kennen.

Zum Vergleich: In den Mainstream-Java -Workbenches muss man zum Zeitpunkt der GUI-Entwicklung die Datenbank-Objekte wie Table, View, Prozeduren etc. kennen, um mit den entsprechenden Assistenten daraus Code zu generieren. Für jede GUI-Komponente werden die Informationen der Datenbank zu Code, das heißt jede GUI-Komponente wird durch eine Code-Unit (hart codiert und kompiliert) repräsentiert. Hat man in einer Anwendung dreißig Master-Detail-Darstellungen – immer mit anderen Datenbank-Objekten –, dann gibt es auch dreißig Code-Units, in denen sich zumindest der Synchronisationsteil immer wiederholt und nur die aus den Datenbank-Objekten abgeleiteten Codeteile unterschiedlich sind. Weitere Informationen zu diesem datengetriebenen Konzept (DAP5) stehen unter www.dap5.de.

Einen plastischen Vergleich bieten die Lego-Baukästen. In DAP5 liegen vorgefertigte Baugruppen bereit, die man zu einem Gebäude zusammensetzen kann. Man wählt die Gebäude-Komponenten aus, um bestimmte Räume (für die Daten) zu bekommen, und es entsteht ein mehrgeschossiger Komplex – die Anwendung.

In den Code-orientierten Workbenches erzeugt man hingegen erst die Baugruppen, bevor man diese zu einem Gebäude zusammensetzt. Jedes Stockwerk, jeder Raum wird mehr oder weniger individuell gefertigt. Durch falsche oder unvollständige Angaben erzeugt man oft mangelhafte Baugruppen – ab in den Müll und neu erstellen. Ändert sich der Datenaufbau geringfügig, muss man die Baugruppe wieder neu produzieren und in das Gesamtgebäude einpassen.

DAP5 und das Repository

Die in DAP5 vorgefertigten GUI-Komponenten sind im Programm selbst schon vorhanden, haben einen eindeutigen Namen zur Adressierung, sonst jedoch nichts, was herkömmlicherweise aus Anwendungsdaten kommt. In ein Table-Objekt wird erst zur Laufzeit der Inhalt der zugeordneten virtuellen Tabelle (VT) gemappt – die VT wiederum wird vorher mit dem Resultset einer Abfrage gefüllt. Genauso werden Formulare, Trees und andere GUI-Komponenten über die VTs versorgt. Diese elegante Funktionalität ist möglich, weil XDEV entsprechende Methoden bezüglich der GUI-Komponenten, der VTs und deren Interaktion anbietet. Alle spezifischen Informationen zu einer kompletten Anwendung wie Selects, Beschriftungen, Breiten, Prozeduraufrufe und sonstige Eigenschaften von Attributen sind in einem (Oracle-basierten) Repository – REPOS gespeichert.

Im REPOS ist hinterlegt, aus welchen Komponenten (Bausteinen) eine Anwendung besteht. Die einzelnen Komponenten und deren Items sind wohlgeordnet mittels einer Baumstruktur im REPOS aufgebaut. Kommt eine Komponente zur Anzeige, erfolgt deren Aktivierung (Zeichnung); alle nötigen Komponenten und Item-Informationen werden aus dem REPOS geholt – diese beeinflussen die Zeichnung in der gewünschten Art. Die Anordnung der ausgewählten DAP5-Komponenten und deren Synchronisation untereinander bei Benutzeraktionen, Refresh etc. wird ausschließlich durch die Informationen aus dem REPOS gesteuert.

DAP5 und die Integrationskraft von Java

Die Applikationsdaten-neutrale Programmierung von GUI-Komponenten ist ein leichtes, weil in XDEV entsprechende Methoden bereitstehen. Zusätzlich verwendbare Methoden des JDK und anderer Java-Libraries öffnen einen Bündel an Möglichkeiten, fertige Funktionen und Fremdkomponenten einzubinden. Aktuell in DAP5 geschieht dies, indem mittels der Bibliothek „iText“ aus gespeicherten Texten, Skripten und Grafiken (Screenshots, Diagramme etc.) des REPOS PDF-Dokumente erzeugt werden.

In absehbarer Zeit wird „jFreeChart“ innerhalb DAP5 nutzbar sein. Genauso ist das „jExcelApi“ ein heißer Kandidat, wenn das Thema der Excel-Connectivity in DAP5 vervollständigt wird – CSV-Export und -Import sind schon implementiert. „jLog“ könnte DAP5-intern verwendet werden, um Logging (Codetracing) für Fehlersuchen und Dokumentationen dynamisch abrufbar zu machen. Die Einbindung von Java Beans in XDEV konnte mangels Zeit und Wissen darüber noch nicht getestet werden.

Dies ist nur eine kleine Liste von sinnvollen Ergänzungen, die man in Java-Programmen einsetzen kann. Sogar die Anbindung von SAP-Bausteinen, mittels Auswertung von Metadaten aus IDOC etc., wurde schon intern diskutiert. Auch hierfür gibt es Bibliotheken, die den Übergang aus Java-Programmen nach SAP und anderen SOA-fähigen Systemen erleichtern. Eine zusammenfassende Bewertung der beteiligten Schichten:

- Java bietet die sprachliche Basis und integrative Technik für jede zukünftige Anwendung – Offenheit ist gegeben
- XDEV2 (3) nutzt Java und stellt zusammen mit dem Framework einen hoch entwickelten Satz von RAD-Features zur Verfügung
- Das DAP5-REPOS ist ein datenbankbasiertes System zum Definieren einer Anwendung unter Nutzung der vorhandenen GUI-Komponenten in DAP5.exe
- DAP5 selbst arbeitet gesteuert aus den Applikationsbeschreibungen aus dem REPOS – nur der funktionale Kern ist codegetrieben, der Rest ist datengetrieben
- Für jene GUI-Anforderungen, die über DAP5-Bausteine nicht datengetrieben abgedeckt werden können, ist eine Erweiterung mit gezielt (klassisch) programmierten XDEV-Komponenten möglich
- Die Erweiterung kann isoliert erfolgen, aber auch hochintegriert, wenn die Infrastruktur des REPOS mit berücksichtigt/genutzt wird

Fazit

Eine einmalig erstellte Infrastruktur wird immer wieder genutzt. Agiles Vorgehen wird unterstützt, der Übergang vom Entwurf einer Anwendung – dem Prototypen – bis zur ersten Einsatzversion ist fließend. Da zum Zeitpunkt der Definition einer Anwendung deren Datenbank-Schemata per XDEV-Datasources offenliegen und angebunden sind, können die erforderlichen Definitionen weitgehend generisch durch DAP5-Fetaures erzeugt werden. Lediglich das Parameter-Mapping aus hierarchisch übergeordneten VTs in darunterliegende Statements muss sorgfältig manuell erfolgen, um das gewünschte Ergebnis schnell erreichen zu können.

```
#---  XDEVT1.ini
REPOS SERVER=mydap5server
REPOS PORT=1521
REPOS DATABASE=ORCL
REPOS USERNAME=DAP5
REPOS PASSWORD=ENCENC (DAP5- und APP-User Passworte encrypted)
REPOS CONFIRM=NO
#---
MSET1 SERVER=mysqlserverA\sqlserver
MSET1 PORT=1433
MSET1 DATABASE=account
MSET1 USERNAME=?
MSET1 PASSWORD=?
MSET1 CONFIRM=NO
#--
ORAS2 SERVER=mysapserverB
ORAS2 PORT=1521
ORAS2 DATABASE=sap2
ORAS2 USERNAME=?
ORAS2 PASSWORD=?
ORAS2 CONFIRM=NO
#---
ORAS3 SERVER=myoraserverC
ORAS3 PORT=1521
ORAS3 DATABASE=orcl
ORAS3 USERNAME=?
ORAS3 PASSWORD=?
ORAS3 CONFIRM=NO
#---
```

Listing 1

Lösungsansatz der Anwendung ABC+

Nachfolgend sind die wesentlichsten Steuerungselemente für die Datenanzeigen aus drei Systemen und deren Synchronisation dargestellt. Dabei handelt es sich um Standardmechanismen in DAP5; die Definition dieser Anwendung erfolgt 100 Prozent datengetrieben, es wird keine einzige Zeile spezieller GUI-Code benötigt.

In einer vollständig definierten ABCP1-Anwendung sind auch Buttons definiert. Unter diese Buttons kann man alle Aufrufe an die DB-Systeme (A, B, C) legen. Auch das Weiterverarbeiten von virtuellen Tabellen in Richtung Datenbanken kann auf generische Art erfolgen: VTs aus System A oder System B als Tabellen in das System C zu übernehmen, wird mit wenigen Zeilen ermöglicht. Wird das gemacht, so hat man aufbereitete Daten aus A und B zusätzlich in C gespeichert; dann steht dem Entwickler das gesamte Spektrum von SQL (Joins etc.) zur Verfügung, um mit SQL-Mitteln weitergehende Datenanalysen und Darstellungen definieren zu können.

Zuordnung zu den Datasources

Eine Ini-Datei für DAP5 erleichtert die Zuweisung der unterschiedlichsten Server für jede in DAP5 per XDEV erklärte Datenquelle (DS) (siehe Listing 1). Username und Password sind hier Dummy-Infos und werden im Verlauf aus REPOS geholt. MSET1 könnte in diesem Beispiel auch durch eine ODBC-DB ersetzt werden.

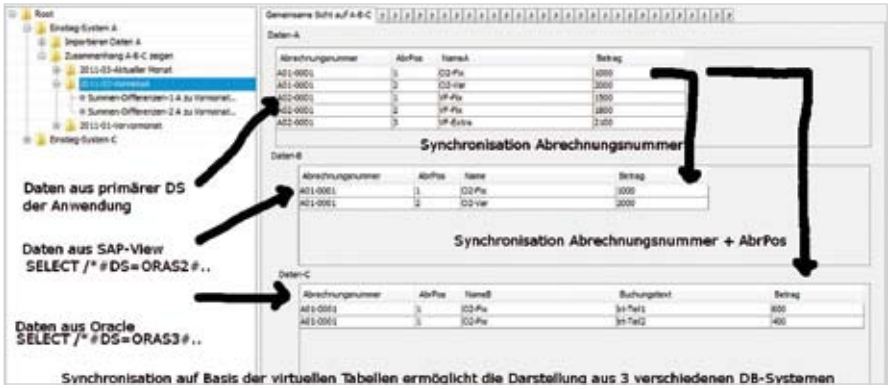


Abbildung 1: Zusammenwirken des DAP5-Treecontrols

Definition einer Anwendung im REPOS

Die REPOS-Tabelle„APP_APPLICATION" besitzt unter anderem folgende Inhalte:

ID	4711
NAME	ABCP1
XDEV_DATASOURCE	MSET1
DS_LIST_PLUS	ORAS2,ORAS3
APPGRP_ID	21

Die im REPOS angelegte Anwendung heißt „ABCP1", die primäre Datenquelle ist „MSET1". Als sekundäre Datenquellen werden „ORAS2" und „ORAS3" zugeordnet.

Userverwaltung und Single-Sign-On

Ein Konstrukt aus Tabellen im REPOS sorgt für diskrete Speicherung userbezogener Connects (Username, Password) an die Datenquellen:

- APP_GROUPS
- APP_XDEV_USER
- APPGRP_MEMBERS

Ein Pflegedialog ist in DAP5 enthalten.

Testdaten der Anwendung ABCP1

Abbildung 1 zeigt das Zusammenwirken des DAP5-Treecontrols (interner Objektname=NAVT) mit dem „TAB13" (Master-Detail, Detail). Die Synchronisation erfolgt über die in „A5" definierten Parameter (..cr_param) (siehe Wörter innerhalb der eckigen Klammern).

```
-- ******************************************************************************************
--
-- der folgende PL/SQL-Block beschreibt die Anwendung
--   diese wird per SQLPLUS  in das REPOS eingearbeitet
--   Aufruf :      start app_abcp1.dat ABCP1      ...   damit wird    alles unter ABCP1 laufen
BEGIN
--
dap_pac.dap_prc_set_app(‹&1›);
dap_pac.dap_prc_del_appx(‹&1›);
commit;
dap_pac.dap_prc_cr_constant( ‹NAVT›, null, ‹DA›, ‹Einstieg-System A›);
dap_pac.dap_prc_cr_constant( ‹NAVT›, ‹DA›, ‹DA_IMP›, ‹Importieren Daten A›);
 dap_pac.dap_prc_cr_constant( ‹NAVT›, ‹DA_IMP›, ‹DA_IMP1›, ‹Importieren Daten A ...›);
 dap_pac.dap_prc_cr_constant( ‹NAVT›, ‹DA_IMP›, ‹DA_IMP2›, ‹Kontrolle Daten A ...›);
--
dap_pac.dap_prc_cr_constant( ‹NAVT›, ‹DA›, ‹DA_ABC›, ‹Zusammenhang A-B-C zeigen›);
  dap_pac.dap_prc_cr_sql( ‹NAVT›, ‹DA_ABC›, ‹DA_ABC_MONATE›,0,
          ‹SELECT /*#MONATE*/                    ‹|| chr(10) ||
          ‹ to_char(sysdate,››RRRR-MM››)          ‹|| chr(10) ||
          ‹,››Aktueller Monat››                 ‹|| chr(10) ||
          ‹ FROM DUAL                            ‹|| chr(10) ||
          ‹UNION SELECT                          ‹|| chr(10) ||
          ‹ to_char(add_months(sysdate,-1),››RRRR-MM››)    ‹|| chr(10) ||
          ‹,››Vormonat››                          ‹|| chr(10) ||
          ‹ FROM DUAL                              ‹|| chr(10) ||
          ‹ order by 1 desc›);
  dap_pac.dap_prc_cr_displayitem (‹DA_ABC_MONATE›, 1);
  dap_pac.dap_prc_cr_displayitem (‹DA_ABC_MONATE›, 2);
  --
  -- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
  -- Daten-A  aus der primären Datenquelle
      dap_pac.dap_prc_cr_sql( ‹TABX›, ‹DA_ABC_MONATE›, ‹DA_ABC_MONATE_13X›,13,
          ‹SELECT /*#DATEN-A*/                      ‹|| chr(10) ||
...... Columnliste ....hier entfernt
          ‹  from DATEN_A x                         ‹|| chr(10) ||
          ‹ where x.MONAT    = ‹›<JJJJ_MM>››        ‹|| chr(10) ||
          ‹ order by 3,4 ‹);
--
      dap_pac.dap_prc_cr_param(‹DA_ABC_MONATE_13X›, ‹<JJJJ_MM>›, ‹NAVT›, -1, 1, null);
      dap_pac.dap_prc_cr_displayitem (‹DA_ABC_MONATE_13X›,-1,›Gemeinsame Sicht auf A-B-C›);
      dap_pac.dap_prc_cr_displayitem (‹DA_ABC_MONATE_13X›, 0,›Daten-A›);
  --   ... Zeilen entfernt
  -- BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
  --  Detail-B  wird über Monat und ABR_NUMMER unter A synchronisiert
  --
      dap_pac.dap_prc_cr_sql( ‹TABX›, ‹DA_ABC_MONATE_13X›, ‹DA_ABC_MONATE_13Y›,13,
          ‹SELECT /*#DS=ORAS2#DATEN-B*/             ‹|| chr(10) ||
...... Columnliste .... hier entfernt
          ‹  from V_DATEN_B x                      ‹|| chr(10) ||
          ‹ where x.MONAT      = ‹›<JJJJ_MM>››      ‹|| chr(10) ||
          ‹   and x.ABR_NUMMER = ‹›<ABR_NUMMER>››   ‹|| chr(10) ||
          ‹ order by 3,4 ‹);
--
      dap_pac.dap_prc_cr_param(‹DA_ABC_MONATE_13Y›, ‹<JJJJ_MM>›,    ‹NAVT›, -1, 1, null);
      dap_pac.dap_prc_cr_param(‹DA_ABC_MONATE_13Y›, ‹<ABR_NUMMER>›, ‹TABX›, 13, 3, ‹X›);
....... Beschriftung und Breiten der Spalten entfernt
  -- CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
  -- Daten-C  aus der 2. sekundären Datenquelle ... dem Oracle-system
  --  Detail-C  wird über Monat und ABR_NUMMER und ABR_POS unter A  synchronisiert
  --
```

```
    dap_pac.dap_prc_cr_sql( ‹TABX›, ‹DA_ABC_MONATE_13X›, ‹DA_ABC_MONATE_13Z›,13,
      ‹SELECT /*#DS=ORAS3#DATEN-C*/            ‹|| chr(10) ||
      ‹ x.ID                        TextID  ‹|| chr(10) ||
      ‹,››0››                       TextSCN ‹|| chr(10) ||
      ‹,x.ABR_NUMMER                TextF03 ‹|| chr(10) ||
      ‹,x.ABR_POS                   TextF04 ‹|| chr(10) ||
      ‹,x.NAME                      TextF05 ‹|| chr(10) ||
      ‹,x.BUCHUNGSTEXT              TextF06 ‹|| chr(10) ||
      ‹,x.BETRAG                    TextF07 ‹|| chr(10) ||
      ‹  from DATEN_C x                     ‹|| chr(10) ||
      ‹ where x.MONAT     = ‹›<JJJJ_MM>››   ‹|| chr(10) ||
      ‹   and x.ABR_NUMMER = ‹›<ABR_NUMMER>›› ‹|| chr(10) ||
      ‹   and x.ABR_POS    =  <ABR_POS>     ‹|| chr(10) ||
      ‹ order by 3,4,6 ‹);
    dap_pac.dap_prc_cr_param(‹DA_ABC_MONATE_13Z›, ‹<JJJJ_MM>›,   ‹NAVT›, -1, 1, null);
    dap_pac.dap_prc_cr_param(‹DA_ABC_MONATE_13Z›, ‹<ABR_NUMMER>›, ‹TABX›, 13, 3, ‹X›);
    dap_pac.dap_prc_cr_param(‹DA_ABC_MONATE_13Z›, ‹<ABR_POS>›,    ‹TABX›, 13, 4, ‹X›);
..... Beschriftung und Breiten der spalten entfernt
..... weitere Knoten siehe screenshot
-- --
END;
/
```

Listing 2

In Listing 2 sind wesentliche Teile der Skriptdatei, mit der die Anwendung ABCP1 definiert wurde, aufgezeigt. Die vollständigen Definitionen können unter www.dap5.de heruntergeladen werden.

Kontakt:
Hans Niedermeier
hn@dap5.de

Hans Niedermeier arbeitet seit Ende der 1980-er Jahre mit Oracle-Datenbanken (beginnend mit Version 4.1.4) sowie von Anfang an mit der Oracle Designer-Schiene und war einer der ersten Mitglieder der DOAG Deutsche ORACLE-Anwendergruppe e.V.
Seit zwei Jahren beschäftigt sich Hans Niedermeier intensiv mit der Rapid Java Entwicklungsumgebung XDEV 2, mit der er den Umstieg von Oracle Forms auf Java erfolgreich vollziehen und seinen eigenen RAD-Konzepte umsetzen konnte. Schwerpunkt ist dabei die Leistungsmerkmale von Anwendungen in der Datenbank zu definieren, um möglichst kompakten Code erstellen zu können.

iJUG Verbund

Die iJUG-Mitglieder stellen sich vor

Ziel des Interessenverbunds der Java User Groups e.V. (iJUG) ist die umfassende Vertretung der gemeinsamen Interessen der Java-Anwendergruppen sowie der Java-Anwender im deutschsprachigen Raum, insbesondere gegenüber Entwicklern, Herstellern, Vertriebsunternehmen sowie der Öffentlichkeit. Bei Wahrung der Eigenständigkeit der Mitglieder sollen insbesondere eine gemeinsame Öffentlichkeitsarbeit stattfinden und der Informations- und Erfahrungsaustausch sowie Netzwerkbildung gefördert werden. Eine weitere Aktivität ist die Herausgabe dieser Zeitschrift. Nachfolgend eine Übersicht aller Mitglieder im iJUG (http://www.ijug.eu).

JUG Deutschland

Java User Group Deutschland e.V.

Die Java User Group Deutschland e.V. (JUG) wurde 1998 von Java-Aktiven aus der Sun User Group (SUG) gegründet. In den Jahren von 1994 bis 1998 war die Gruppe schon innerhalb der SUG aktiv. Nachdem sich Java immer mehr auch außerhalb von sun microsystems entwickelte, erschien ein eigener Verein sinnvoll. Der Verein versteht sich als ein Forum für den Austausch von Erfahrungen und als Kontaktadresse für Java-interessierte im deutschsprachigen Raum. Er betreibt den Webserver java.de und veranstaltet zusammen mit der SUG die Source Talk Tage in Göttingen.

http://www.java.de

DOAG
Deutsche ORACLE-Anwendergruppe e.V.

DOAG Deutsche ORACLE Anwendergruppe e. V.

Die DOAG Deutsche ORACLE Anwendergruppe e. V. ist die einzige Interessenvertretung der Anwender von Oracle-Produkten in Deutschland. Sie ist gegenüber der Oracle Deutschland B.V. & Co. KG wirtschaftlich und rechtlich selbständig. Ziele der DOAG sind Informationsaustausch und Wissensvermittlung über Einsatz, Umgang und Erfahrungen mit den Produkten von Oracle sowie die Interessenvertretung der Anwender gegenüber dem Hersteller. Aufgrund der Sun-Übernahme durch Oracle hat die DOAG ihr Engagement im Bereich Java durch die Gründung einer Special Interest Group Java deutlich verstärkt.

http://www.doag.org

JUGS
java user group stuttgart

Java User Group Stuttgart e.V. (JUGS)

Kurz nach Erscheinen des JDK 1.1 Release wurde vor nunmehr über 11 Jahren im März 1997 die Java User Group Stuttgart (JUGS) als unabhängiger, gemeinnütziger Verein gegründet und hat heute über 400 Mitglieder. Ziel ist es, Java-Anwendern und Interessierten ein Forum zum Austausch von Informationen und Erfahrungen zu bieten und den Aufbau direkter Kontakte zu innovativen Firmen der Region zu ermöglichen. Dazu werden regelmäßig Vortrags- und Diskussionsabende, größere Events zu aktuellen Themen sowie seit 1998 das Java Forum Stuttgart veranstaltet.

http://www.jugs.de

Java User Group Köln

Die Java User Group Köln (JUGC) ist eine im weltweiten JUG-Verbund intensiv verknüpfte Gruppe von Java-Interessierten der gesamten Rheinschiene (bis Aachen, Bonn, Ruhrgebiet, insbesondere Köln). Neben dem Wissens- und Erfahrungsaustausch steht das Kennenlernen und der Spaß im Mittelpunkt. In der unabhängigen Gruppe tauschen sich seit 2001 sowohl Anfänger als auch gestandene Java-Profis aus.

http://www.jugcologne.eu

Java User Group München (JUGM)

Die Java User Group München (JUGM) besteht seit dem Jahr 2000 und hat zur Zeit über 700 Mitglieder. Initiiert wurde die JUGM von der Regionalgruppe München der Gesellschaft für Informatik e.V. Im Umfeld der JUGM sind Gruppen mit spezialisierter Ausrichtung entstanden:

- JBoss User Group München, http://www.jbug-munich.org
- Groovy, Grails & Griffon User Group München, http://ggg.mixxt.de
- Scala München, http://scala-southern-germany.mixxt.de/networks/groups/ScalaMuenchen/index

http://www.jugm.de

Java User Group Metropolregion Nürnberg

Die Java User Group Erlangen/Nürnberg wurde 2007 mit der Idee gegründet, die Java-Gemeinschaft der Region zu fördern. Man trifft sich regelmäßig zu interessanten Vorträgen und Themen des Java-Umfeldes. Mitgliedschaft bedeutet lediglich den Email-Newsletter zu abonnieren, damit man über die aktuellen Events informiert ist. Dank der Unterstützung zahlreicher in der Metropolregion Nürnberg ansässiger Unternehmen variieren die Orte der Treffen und fördern so den Kontakt und das Netzwerken in der Region. Vorstand Oliver Szymanski ist stolz darauf, gemeinsam mit den anderen JUGs und der DOAG die iJUG gegründet zu haben und damit die Java User Group Erlangen/Nürnberg in die neue starke Gemeinschaft führen zu dürfen. Damit können auch die Interessen erfolgversprechend vertreten werden.

http://www.source-knights.com

Java User Group Ostfalen

Im Januar 2010 war die Gründung der JUG Ostfalen. Erste Ziele wurden bei einem lockeren Stammtisch formuliert, dem inzwischen mehrere Vorträge folgten. Neben reinem Networking sollen Industrie und Forschung (hier gibt es überall Hochschulen) einander nähergebracht, Know How weitergegeben und der fachliche Austausch ermöglicht werden. Das ist aber noch lange nicht alles! Eine ganze Menge Ideen warten noch auf Umsetzung, aber davon mehr zu gegebener Zeit.

http://www.jug-ostfalen.de

Java User Group Saxony

Die Idee zur Gründung der Java User Group Saxony wurde im Dezember 2007 geboren und im Laufe des Frühjahrs 2008 von Torsten Rentsch und Falk Hartmann in die Tat umgesetzt. Schon kurz nach Einrichtung der Internetpräsenz stieß Kristian Rink als Nutzer der ersten Stunde zum Organisationsteam. Nach der Gründungsveranstaltung im April 2008 mit 43 Teilnehmern wächst die JUG Saxony stetig weiter. Man ist bemüht, qualitativ hochwertige Vorträge mindestens sechsmal jährlich zu organisieren. Eine formale Mitgliedschaft gibt es in der JUG nicht. Neben den Vorträgen steht der (Wissens-)austausch zwischen den Teilnehmern und das Knüpfen von Kontakten im Vordergrund. Auch die Zusammenarbeit mit wissenschaftlichen Einrichtungen ist wichtig.

http://www.jugsaxony.org

Sun User Group Deutschland e.V.

Die Sun User Group e.V. (SUG) wurde 1987 gegründet. In der Anfangszeit trat sun microsystems im großen Umfang als Förderer auf. Ende der 1990er Jahren verlor das Unternehmen das Interesse an User Groups. Trotzdem hat der Verein überlebt und mit den Source Talk Tagen in Göttingen weiterhin Service-Leistungen für seine Mitglieder erbracht und den Kontakt zum Unternehmen gehalten. Die Übernahme von sun microsystems durch Oracle stellt den Verein vor die Herausforderung, den Interessen Anwender von sun Technologie beim neuen Eigentümer Gehör zu verschaffen.

http://www.sugd.de

Swiss Oracle User Group (SOUG)

Die SOUG wurde 1987 als unabhängiger Verein gegründet und umfasst heute rund 600 Mitglieder. Als Community der Oracle-Spezialisten fördert sie den Informations- und Erfahrungsaustausch, vertritt die Interessen der Mitglieder auf nationaler sowie internationaler Ebene und bietet verschiedene Möglichkeiten zur Vernetzung mit anderen Personen. Die SOUG-R (SOUG Romandie) ist in der Westschweiz für die französisch sprechenden Mitglieder aktiv. Traditionell war die SOUG eher Datenbank- und Oracle-Tool-orientiert. Mit dem Aufkommen von Java gewann dieses Thema natürlich auch für Oracle-Spezialisten an Bedeutung. Heute hat Java einen festen Stammplatz an den Veranstaltungen und im Newsletter der SOUG.

http://www.soug.ch

Testen mit JUnit

Gerald Kammerer, freier Redakteur

Bei vielen Software-Projekten muss das Team mit weniger Zeit, Entwickler-Personal und Budget auskommen, als für die Umsetzung eigentlich notwendig wären. In der Praxis besteht in Sachen Budget und Personal-Ressourcen nur wenig Spielraum. Folglich entstehen zwangsweise Abstriche in Bezug auf Liefertermin, Funktionsumfang oder Qualität.

Für Projektverantwortliche hat meistens das pünktliche Erreichen gesetzter Milestones und die Einhaltung von Budgets oberste Priorität. Denn die Freigabe von Folge-Budgets ist oft an Entwicklungsstände und Termine gekoppelt, die erreicht beziehungsweise eingehalten werden müssen. Erhebliche Verzögerungen sowie Budget-Überschreitungen können schnell das gesamte Projekt gefährden. Hierbei spielt auch die persönliche Komponente eine wichtige Rolle, denn für Projektverantwortliche kann sich ein gescheitertes Projekt sehr negativ auf die eigene Karriere auswirken. Eine Reduzierung der erwarteten Features wiederum erzeugt meist heftigen Widerstand auf der Anwenderseite.

Der vermeintlich einfachste Kompromiss ist an dieser Stelle das „vorläufige" Opfern von Qualität. Hierbei besteht aus Sicht der meisten Beteiligten der größte Spielraum, um den erfahrungsgemäß niemand sonderlich streiten wird. Das Argument, Dokumentation und Tests auch nach erfolgreichem Projektabschluss problemlos nachholen zu können, wirkt beruhigend und zieht bei Verantwortlichen, Anwendern und Entwicklern gleichermaßen. Für Letztere sind Softwaretests und Dokumentation in klassischen Software-Projekten ohnehin vergleichsweise wenig spannend und bedeuten zudem erheblichen Mehraufwand, auf den man gern verzichtet, um die meist knappe Entwicklungszeit erst einmal für Planung und Implementierung nutzen zu können. Bereits während der Entwicklung mit der Vorahnung zu testen, dass sich viele Dinge sowieso wieder ändern können, wird den meisten Beteiligten ohnehin als nicht sinnvoll erscheinen, sodass man sich in Sachen Softwaretests hierbei schnell einig sein wird.

Nicht getesteter Code birgt große Risiken

Eine solche Entscheidung kann jedoch fatale Folgen nach sich ziehen. Programmfehler, die man zu spät erkennt und beseitigt, können sich fortsetzen und zu Folgefehlern und Fehlerverkettungen führen, insbesondere wenn der fehlerhafte Code von unterschiedlichen Entwicklern verwendet wird. Entsprechende Folgefehler sind später kaum mehr nachvollziehbar und lassen sich oft nur mit erheblichem Such- und Testaufwand aufspüren. Mit der Anzahl an Programmzeilen steigt zudem der notwendige Zeitaufwand für die Fehlersuche exponentiell an. Nicht getesteten Programmcode über einen längeren Zeitraum zu verwenden vergrößert demnach das Projektrisiko erheblich und die Folgen sind nicht absehbar. Nicht selten müssen im Nachhinein aufwändige Programmänderungen durchgeführt werden, was im schlechtesten Fall zur Unrentabilität des gesamten Projekts führen kann.

Vernünftiger und wesentlich effektiver ist es, ein Programm bereits während der Entstehung ständig zu testen. Das klingt aufwändiger, als es ist, denn Software-Tests lassen sich weitgehend automatisieren. Dafür gibt es das von Kent Beck und Erich Gamma geschriebene Java-Test-Framework JUnit. Das Framework steht aktuell in Version 4.8.2 als Open Source unter der Common Public License zur Verfügung und kann über www.junit.org bezogen werden. Es ermöglicht dem Entwickler, Programmcode schnell und effizient zu testen. Die meisten Java IDEs wie Eclipse, JBuilder und IntelliJ bieten bereits eine JUnit-Integration. Oracle soll dagegen bereits Vorbereitungen treffen, JUnit aus NetBeans zu entfernen, da man offenbar mit der von JUnit verwendeten CPL-Lizenz rechtliche Probleme sieht.

Anwendungen lassen sich testen, indem man den Programmcode mit einem Debugger analysiert und zum Beispiel Variablenwerte überprüft oder indem man entsprechende Informationen über die Konsole ausgibt. Zudem kann der Code auch von mehreren Personen kontrolliert werden (Code-Review). Diese Methoden der Code-Analyse haben jedoch den Nachteil, dass der Entwickler selbst die Prüfung der Anwendung vornehmen muss, was enorm zeitaufwändig ist und daher in der Praxis nur grob durchgeführt wird. Noch schwieriger ist der manuelle Test von zeitgleichen Abläufen in einer Applikation, welche in Java mit Threads gesteuert werden.

Automatisierte Tests verringern das Fehlerrisiko

JUnit bietet eine dynamische Methode zur Durchführung der Code-Analyse. Die Tests werden dabei direkt in Java als Testklassen geschrieben. JUnit-Tests laufen automatisiert ab und können beliebig oft wiederholt werden. Ein Test prüft jedoch selten die gesamte Anwendung, sondern

immer nur die Korrektheit eines kleinen Programmbausteins, etwa einer Klasse oder einer einzelnen Methode. Natürlich ist auch das Testen einer ganzen Abfolge einzelner Tests möglich. Ein JUnit-Test kann zwei Ergebnisse liefern: „Test erfolgreich" oder „Test fehlgeschlagen". Die jeweiligen Zustände werden in den gängigen IDEs mit einem grünen beziehungsweise roten Symbol angezeigt. Man spricht deshalb davon, dass ein Test grün oder rot ist.

Bei der Umsetzung eines Tests werden im Idealfall zuerst JUnit-Testklassen mit Testmethoden geschrieben – erst danach der zu testende Code, da man ansonsten wichtige Testszenarien übersehen könnte. Damit ist JUnit eine agile Methode und ein Bestandteil von Extreme Programming. In der Praxis werden JUnit-Tests aus Zeitgründen dennoch oft erst im Nachhinein geschrieben. Der zu testende Code wird erst freigegeben, wenn dieser sämtliche Tests bestanden hat. Später werden vor jeder Änderung des Codes die JUnit-Tests aufgerufen, um zu gewährleisten, dass der Code vor dem Eingriff fehlerfrei ist. Nach der Änderung werden die Tests erneut aufgerufen, wodurch sich Fehler bei der Erweiterung beweisen lassen. JUnit-Tests laufen in der Regel so schnell ab, dass man diese sogar bei jedem Speichervorgang ausführen kann. Durch diese testgetriebene Vorgehensweise wird sichergestellt, dass man ausschließlich getesteten Programmcode verwendet.

An dieser Stelle muss man sich jedoch darüber bewusst sein, dass dies nicht bedeutet, dass der geschriebene Code gänzlich frei von Programmfehlern ist. Mit JUnit lässt sich lediglich erreichen, dass der Code spezielle Testszenarien besteht. Somit ist ganz entscheidend, welche und wie viele Testszenarien man abbildet. Theoretisch kann der Testcode um ein Vielfaches umfangreicher sein als der zu testende Code. Der Entwickler muss also selber abwägen, welche Tests wirklich wichtig sind. Selbst wenn nur die wichtigsten Klassen und Methoden getestet werden und diese nur die allerwichtigsten Fälle abdecken, gibt die Verwendung von getestetem Code allen Projektbeteiligten deutlich größere Sicherheit. Dies wirkt sich auch psychologisch vorteilhaft aus, insbesondere bei sehr wichtigen Softwareprojekten, bei denen die beteiligten Personen unter großem Leistungs- und Erwartungsdruck stehen.

```
public class MathUtil
{
        /* Addiert a mit b und gibt Summe c zurück */
        public static double addition(double a , double b)
        {

                double c = a + b;
                return c;
        }
}
```

Listing 1

```
public class MathTest
{
        @Test
        public void addition()
        {
                double expected = 7.5;
                double actual = MathUtil.addition(5, 2.5);
                Assert.assertEquals(expected, actual);
        }
}
```

Listing 2

```
/* Dividiert a durch b und gibt Summe c zurück */
public static double division(double a , double b)throws ArithmeticException
{
        if(b == 0)
        {
                throw new ArithmeticException(«Divisor must not be zero.»);
        }
        double c = a / b;
        return c;
}
```

Listing 3

```
@Test(expected = ArithmeticException.class)
public void divisionException() throws ArithmeticException
{
        MathUtil.division(2, 0);
}
```

Listing 4

Nach dem Download muss JUnit lediglich zum Klassenpfad hinzugefügt werden. Mit JUnit 4 gibt es einige hilfreiche Neuerungen wie die Unterstützung von Annotations. Die Klassen befinden sich nun im Package „org.junit". Alle Testklassen müssen nun nicht mehr von einer bestimmten Klasse abgeleitet werden, sondern können auch einfach nur gewöhnliche Java-Klassen sein. Sobald eine Methode durch

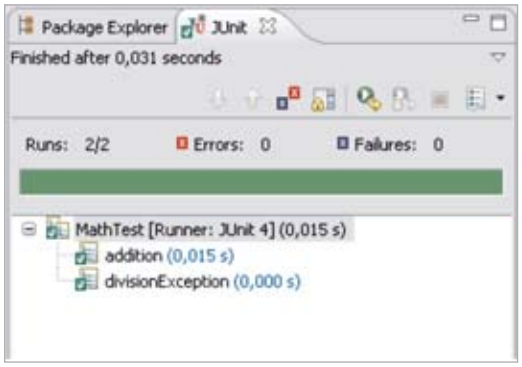


Abbildung 1: JUnit-Test in Eclipse – erfolgreich

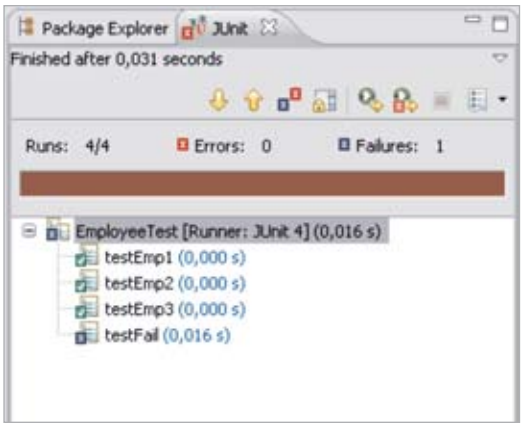


Abbildung 2: JUnit-Test – nicht erfolgreich

die Annotation „@Test" gekennzeichnet wird, ist diese eine Testmethode, die automatisch von JUnit aufgerufen wird. Der gesamte Testcode ist ein völlig unabhängiger Programmcode. Es bietet sich an, für die Testcases eigens einen Source-Folder anzulegen.

Das folgende Beispiel veranschaulicht, wie einfach JUnit funktioniert. Dazu werden zuerst JUnit-Tests für eine einfache Addition sowie für eine Division geschrieben. Zum Ausführen der Tests stellen die genannten IDEs Assistenten zur Verfügung. So lässt sich eine Testklasse in Eclipse beispielsweise bequem per Rechtsklick – „Run as" – „JUnit Test" starten (siehe Listing 1).

Die Klasse MathUtil besitzt die Methode „addition", die zwei Zahlen „a" und „b" miteinander addiert. Als Ergebnis wird „c" zurückgegeben. Um diese Methode zu testen, schreibt man eine entsprechende Testklasse „MathTest" mit einer Testmethode, welche die Addition ausführt und das Ergebnis mit einem erwarteten Wert vergleicht. Für die Einbindung von JUnit sind die Imports „junit.framework.Assert;" und „org.junit.Test;" nötig (siehe Listing 2).

Durch die Annotation „@Test" erkennt JUnit, dass es sich bei der nachfolgenden Methode „addidtion()" um einen Test handelt, und führt diesen automatisch aus. In der Testmethode wird der Wert „7.5" als erwartetes Ergebnis definiert. Anschließend wird die zu testende Methode aufgerufen und das Ergebnis in der Variable „actual" gespeichert. Mithilfe der häufig verwendeten JUnit-Methode „assertEquals()" wird dann das Ergebnis mit dem erwarteten Wert verglichen. Wenn beide Werte gleich sind, gilt der JUnit-Test als bestanden und ist grün.

In bestimmten Fällen wird von einer Methode kein Rückgabewert, sondern eine Exception erwartet. Ob diese korrekt geworfen wird, lässt sich mit JUnit ebenfalls testen. Dazu wird zuerst die zu testende Klasse „MathUtil" um eine zweite Methode „division()" erweitert (siehe Listing 3).

Anschließend erweitert man auch die Testklasse „MathTest" mit einer zweiten Testmethode, die prüft, ob beim Aufruf der Methode „division()" wie erwartet eine ArithmeticException geworfen wird, wenn „0" als Divisor übergeben wird (siehe Listing 4).

Der Test ist erfolgreich, wenn zur Laufzeit wie erwartet eine ArithmeticException geworfen wird. Ist dies nicht der Fall oder eine Exception eines anderen Typs wird geworfen, schlägt dieser Test fehl und ist rot. Neben „@Test" gibt es weitere wichtige Annotationen. „@Before" benennt Methoden, die vor jedem Testfall ausgeführt werden sollen, sodass diese nicht bei jedem Testfall einzeln codiert werden müssen (siehe Listing 5). „@After" kennzeichnet solche, die entsprechend nach jedem Testfall ausgeführt werden.

Auch für bestimmte Zustände, die für verschiedene Tests hergestellt werden müssen, stehen Annotationen zur Verfügung. „@BeforeClass" wird für jede Testklasse nur ein einziges Mal und zudem vor allen „@Before"-Methoden ausgeführt, während „@AfterClass" nach allen „@After"-Methoden ausgeführt wird.

Typischerweise werden in den mit „@Before" und „@BeforeClass" gekennzeichneten Methoden Datenbank-Verbindungen hergestellt, Ordner angelegt oder Testdaten bereitgehalten. Die mit „@After" und „@AfterClass" gekennzeichneten Methoden werden meist genutzt, um den Ursprungszustand wiederherzustellen. Mit der Annotation „@Ignore" kann man Tests

```
@Before
public void init()
{
m = new MathUtil(10, 5);
}
```

Listing 5

```
MathUtil m;

@BeforeClass
public static void beforeClass()
{
System.out.println(«Test - MathUtil Start»);
}


@Before
public void init()
{
m = new MathUtil();
}


@Test
public void additionTest1()
{
double expected = 15;
double actual = m.addition(10, 5);
Assert.assertEquals(expected, actual);
}


@Test
public void division()
{
double expected = 2;
double actual = m.division(10, 5);
Assert.assertEquals(expected, actual);
}


@AfterClass
public static void afterClass()
{
System.out.println(«Test - MathUtil Finished»);
}
```

Listing 6

kennzeichnen, die nicht ausgeführt werden, weil z.B. die Implementierung dafür noch fehlt. Um das folgende Beispiel einfach zu halten, ist die Ausgabe auf die Konsole reduziert (siehe Listung 6).

Ergänzende Tools

Die JUnit-Standard-Integrationen der jeweiligen Java-IDEs bieten meist nur einen rudimentären Funktionsumfang. Code-Coverage-Tools wie Clover, JTest, Serenity für Hudson oder ELCEmma stellen eine sehr hilfreiche Ergänzung für die Überprüfung und Auswertung von JUnit-Tests dar. Letzteres ist Open Source und als Plug-in für Eclipse unter www.eclemma.org verfügbar. Das Tool visualisiert die von einem JUnit-Test durchlaufenen sowie nicht durchlaufenen Code-Passagen, etwa bei "If-Else"-Verzweigungen. Auch die prozentuale Testabdeckung eines Programms wird sichtbar. Damit erhält der Entwickler eine hilfreiche Übersicht.

Die in JUnit implementierten Testcases sind in der Praxis weit mehr als nur Software-Tests. Die Testklassen zeigen auf, wie man die zu testende Klasse oder ein API richtig verwendet und geben Auskunft über die resultierenden Rückgabewerte der Methoden. Damit stellen JUnit-Tests gleichzeitig eine hilfreiche Dokumentation dar, was für die spätere Wartung und Weiterentwicklung einer Anwendung von großem Wert sein wird. Viele Entwickler sehen sich sogar lieber JUnit-Tests an als klassische Tutorials oder seitenweise Dokumentationen.

Effektivere Teamarbeit

Einem Kollegen mitzuteilen, dass sein Code fehlerhaft ist, kann oft schwierig sein, vor allem wenn das Entwicklerteam über mehrere Standorte verteilt ist oder wenn sprachliche Barrieren bestehen. Bei der Problembeschreibung in Prosa kann es erfahrungsgemäß leicht zu Missverständnissen kommen. Effektiver ist es, einen JUnit-Testcase zu schreiben und diesen für alle Beteiligten als nachvollziehbaren Beweis des Problems weiterzugeben. Selbst gegenüber der Variante, dem Kollegen ein formloses Projekt mit dem isolierten, fehlerhaften Code zu senden, hat ein JUnit-Test Vorteile. So muss kein Startpunkt gesucht werden und man muss sich vor allem nicht auf ein Ziel des Beispielcodes verständigen, der meist nicht alle Fälle abdeckt. Mit JUnit ist die Sache einfach. Sobald der JUnit-Test erfolgreich läuft, ist das Problem gelöst. Gleichzeitig erhält der Entwickler damit automatisch eine ständig wachsende Testabdeckung.

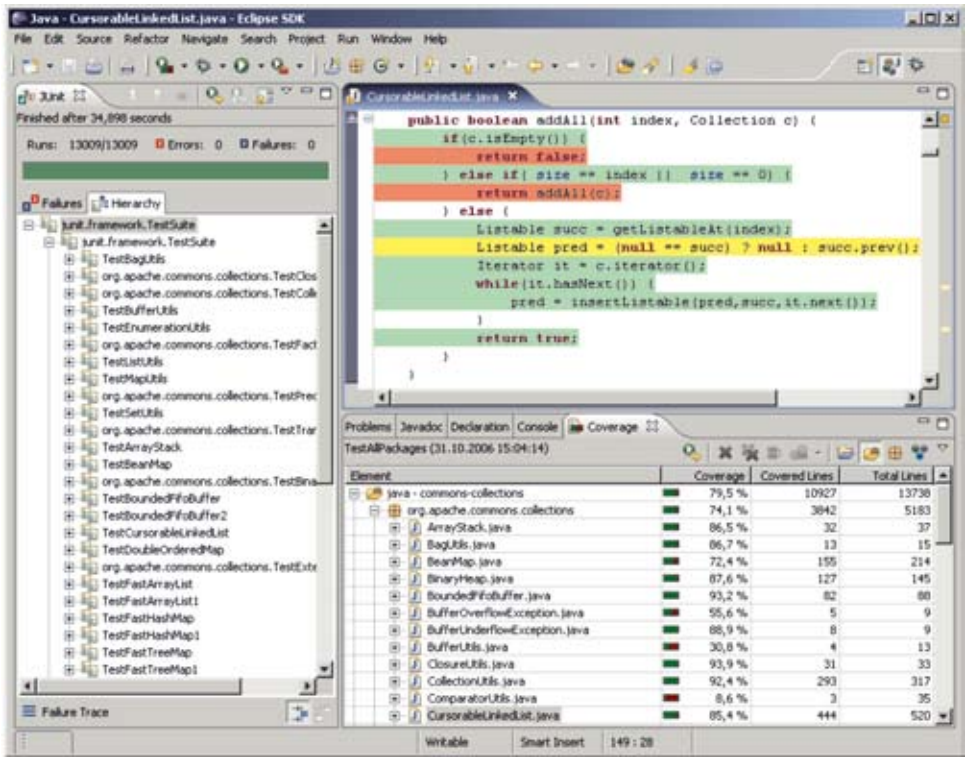


Abbildung 3: Code-Coverage-Tool ELCEmma für JUnit-Tests

Fazit

JUnit ist ein Java-Framework für den automatisierten Test von Java-Programmen. Alle Tests werden direkt in Java geschrieben und sind damit beliebig oft wiederholbar. Ein erfolgreicher JUnit-Test stellt sicher, dass eine Methode auch tatsächlich das erwartete Ergebnis oder eine bestimmte Exception liefert. Je mehr Testcases für eine Klasse oder Methode geschrieben werden, desto geringer wird das Fehlerrisiko. Nach der JUnit-Philosophie sollten die Testfälle sogar vor dem zu testenden Programmcode geschrieben werden, damit wichtige Testfälle nicht versehentlich vergessen werden. Aus Zeitgründen können in der Praxis oft nur für die wichtigsten Klassen und Methoden JUnit-Tests geschrieben werden und immer wieder wird man wichtige Testcases vergessen, sodass sich komplett fehlerfreie Anwendungen auch mit JUnit nicht erreichen lassen. Doch schon mit Tests für die wichtigsten Programmteile lässt sich bereits eine gewaltige Steigerung der Code-Qualität erreichen, was dem Entwickler zusätzliche Sicherheit gibt und vor allem bei wichtigen und besonders zeitkritischen Projekten psychologisch von großem Vorteil ist. Der Einsatz von JUnit hat sich bewährt und gilt mittlerweile als ein wichtiges Merkmal einer weitsichtigen Projektplanung und guten Qualitätssicherung.

Kontakt:
Gerald Kammerer
info@redaktion-kammerer.de

Gerald Kammerer ist als freier Redakteur seit 2006 für verschiedene Computerfachzeitschriften mit Schwerpunkt Java- und AJAX-Entwicklung tätig (siehe auch Seite 44).

Java aktuell – das Abo

4 Ausgaben für 18 Euro

Jetzt Abonnement sichern:

- O Java aktuell – das iJUG-Magazin Abo: vier Ausgaben zu 18 Euro im Jahr
- O Abonnement Newsletter: Java aktuell – der iJUG-Newsletter, kostenfrei

Für Oracle-Anwender und Interessierte gibt es das Java aktuell Abonnement auch mit zusätzlich sechs Ausgaben im Jahr der Fachzeitschrift DOAG News und zwei Ausgaben im Jahr Business News zusammen für 75 Euro. Weitere Informationen unter www.doag.org/shop/

Senden Sie das ausgefüllte Formular an:

Interessenverbund der Java User Groups e.V.
Tempelhofer Weg 64
12347 Berlin

oder faxen Sie es an:

0700 11 36 24 39

oder bestellen Sie online:

go.ijug.eu/go/abo

Anschrift:

Name, Vorname

Firma

Abteilung

Straße, Hausnummer

PLZ, Ort

ggf. Rechnungsanschrift:

Straße Hausnummer

PLZ, Ort

E-Mail

Telefonnummer

Die allgemeinen Geschäftsbedingungen* erkenne ich an, Datum, Unterschrift

*Allgemeine Geschäftsbedingungen:
Zum Preis von 18 Euro (inkl. MwSt.) pro Kalenderjahr erhalten Sie vier Ausgaben der Zeitschrift "Java aktuell - das iJUG-Magazin" direkt nach Erscheinen per Post zugeschickt. Die Abonnementgebühr wird jeweils im Januar für ein Jahr fällig. Sie erhalten eine entsprechende Rechung. Abonnementverträge, die während eines Jahres beginnen, werden mit 4,90 Euro (inkl. MwSt.) je volles Quartal berechnet. Das Abonnement verlängert sich automatisch um ein weiteres Jahr, wenn es nicht bis zum 31. Oktober eines Jahres schriftlich gekündigt wird. Die Wiederrufsfrist beträgt 14 Tage ab Vertragserklärung in Textform ohne Angabe von Gründen.

www.ijug.eu