KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

ISSN 1865-682X

08/2015

Jahrgang 8

KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

Sie können die elektronische Form des KaffeeKlatschs
monatlich, kostenlos und unverbindlich
durch eine E-Mail an

abo@bookware.de

abonnieren.

Ihre E-Mail-Adresse wird ausschließlich für den Versand
des KaffeeKlatschs verwendet.

Editorial

Abgerundet

Auch wenn wir als Software-Entwickler nur am Rande damit zu tun haben, so spielt die Darreichungsform unserer Produkte oft eine große Rolle. Dabei ist es nicht ganz unproblematisch, dass sich eine Unzahl von Patenten diesem Thema widmet und nicht intuitiv erkennbar ist, was man ungefragt machen darf und was nicht.

So hat etwa Apple für Mobiltelefone ein Patent eingereicht, dessen schützenswerte Eigenschaften ein Display, ein Lautsprecher, ein Home-Knopf und runde Ecken sind. Ein iPhone eben. Wie anders soll denn bitte etwas aussehen, was man sich vielleicht auch in die Hosentasche stecken soll und mit dem man telefonieren (sprich: hören) will? Zugegeben, der Home-Knopf muss nicht vorne sein, aber es bietet sich schon an, wenn man den sehen kann.

Ein solch banales Patent kann übrigens Milliarden kosten. So hat sich ja Apple diesbezüglich mit Samsung gestritten und bisher auch weitestgehend recht bekommen. Und Samsung ist eben zu diesen milliardenschweren Strafen verurteilt worden. Da hat es auch nichts genutzt, Ausschnitte vom Film 2001: *Odyssee im Weltraum* aus dem Jahr 1969 zu zeigen, in dem schon ein vergleichbares Tablet zu sehen ist.

Dieses Patent ist jetzt übrigens gekippt worden. Allerdings nicht, weil man dessen Banalität eingesehen hätte, sondern wegen eines Formfehlers. Womit Apples Klage haltlos werden könnte. Aber wie das dann wirklich ausgeht, bleibt abzuwarten.

Derartige Schwierigkeiten lauern aber nicht nur bei Gerätschaften sondern auch bei Prozessen. An dieser Stelle sei nur das 1-Click-Patent von Amazon erwähnt. Die hatten sich patentieren lassen, dass man mit einem Mausklick ein Produkt kaufen kann, ohne das Problem eines Warenkorbs zu haben. Was in den USA mehr oder weniger patentierbar war, hatte in Europa zum Glück keine Chance, zeigt aber, wie schwierig es sein kann eine Idee international zu vermarkten.

Sich als Software-Entwickler auf den Standpunkt zu stellen, dass man damit ja nichts zu tun hat, ist leider nicht ganz richtig. Es gibt einige Patente zu Bit-Frickeleien. Wer etwa verzweigungsfrei den Absolutwert einer ganzen Zahl bestimmen will, muss Obacht walten lassen, ob er nicht ein von Sun eingereichtes Patent verletzt.

Allerdings stellt sich auch hier die Frage, ob diese überhaupt haltbar sind. Das Problem ist nur, dass es dem Nutzungswilligen auferlegt ist, dass ein solches Patent nicht erteilt hätte werden dürfen. Und zunächst ist es eine lästige Produktivitätsbremse.

Am einfachsten ist es natürlich, den umgekehrten Weg zu gehen: Einfach erst einmal machen, und dann abwarten, ob sich überhaupt jemand meldet. Hat man Geld verdient, ist es auch deutlich einfacher, Gebühren zu entrichten; was viel besser ist, als Geld zu bezahlen, das man möglicherweise gar nicht erwirtschaftet.

Ihr Michael Wiedeking
Herausgeber

Beitragsinformation

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

Beiträge

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

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

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

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

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

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

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

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

Leserbriefe

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

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

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

Werbung ist Information

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

- Stellenanzeigen
- Seminaranzeigen
- Produktinformation und -werbung

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

Der Werbeplatz kann bei Frau Natalia Wilhelm via E-Mail an anzeigen@bookware.de oder telefonisch unter 09131/8903-90 gebucht werden.

Abonnement

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

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

Auf Wunsch schicken wir Ihnen auch ein gedrucktes Exemplar. Da es sich dabei um einzelne Exemplare handelt, erkundigen Sie sich bitte wegen der Preise und Versandkosten bei Natalia Wilhelm via E-Mail unter natalia.wilhelm@bookware.de oder telefonisch unter 09131/8903-90.

Copyright

Das Copyright des KaffeeKlatschs liegt vollständig bei der Bookware. Wir gestatten die Übernahme des KaffeeKlatschs in Datenbestände, wenn sie ausschließlich privaten Zwecken dienen. Das auszugsweise Kopieren und Archivieren zu gewerblichen Zwecken ohne unsere schriftliche Genehmigung ist nicht gestattet.

Sie dürfen jedoch die unveränderte PDF-Datei gelegentlich und unentgeltlich zu Bildungs- und Forschungszwecken an Interessenten verschicken. Sollten diese allerdings ein dauerhaftes Interesse am KaffeeKlatsch haben, so möchten wir diese herzlich dazu einladen, das Magazin direkt von uns zu beziehen. Ein regelmäßiger Versand soll nur über uns erfolgen.

Bei entsprechenden Fragen wenden Sie sich bitte per E-Mail an copyright@bookware.de.

Impressum

KaffeeKlatsch Jahrgang 8, Nummer 8, August 2015
ISSN 1865-682X
Bookware – eine Initiative der
MATHEMA Verwaltungs- und Service-Gesellschaft mbH

Henkestraße 91, 91052 Erlangen
Telefon: 0 91 31 / 89 03-90
Telefax: 0 91 31 / 89 03-99
E-Mail: redaktion@bookware.de
Internet: www.bookware.de

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

Inhalt

Editorial 3

Beitragsinfo 4

Inhalt 5

User Groups 21

Werbung 23

Das Allerletzte 27

Artikel

Scaling Scrum

Können wir Scrum skalieren und dabei noch Scrum „by the book“ folgen? 7

(All)gemeine Trickserei

Zustände mit Java Generics 11

Do it yourself

Eigene JPA Mappings mit *Attribute Convertern* 14

Kolumnen

Die spinnen, die Römer!

Des Programmierers kleine Vergnügen 17

Es kommt alles wieder

Kaffeesatz 19

Scaling Scrum

Können wir Scrum skalieren und dabei noch Scrum „by the book“ folgen? 7

von Tim Bourguignon

Eigentlich ist die Scrum-Methodologie schnell erklärt. So lässt sich in wenigen Minuten eine ausreichend verständliche Einführung in Scrum geben. Schwieriger ist es da schon, Scrum einzuführen und konsequent umzusetzen. Muss Scrum dann zusätzlich noch skaliert werden, stellt sich die Frage, ob das überhaupt möglich ist, ohne dabei die Treue zu Scrum zu opfern.

(All)gemeine Trickserei

Zustände mit Java Generics 11

von Róbert Bräutigam

Java Generics sind bei weitem nicht so mächtig wie in Scala oder wie in einer „richtigen“ funktionalen Sprache wie Haskell. Nichtsdestoweniger kann man sehr interessante – wenn auch vielleicht weniger praktische – Lösungen damit programmieren.

Do it yourself

Eigene JPA Mappings mit *Attribute Convertern* ... 14

von Werner Eberling

Das letzte Update des JPA Standards wartet mit einer Reihe interessanter neuer Features auf, die in der Entwicklergemeinde bisher leider weitestgehend unbeachtet geblieben sind. Um dies zu ändern soll hier eine dieser Neuerungen, der *Attribute Converter,* mit seinen Anwendungsmöglichkeiten, aber auch seinen Grenzen, genauer vorgestellt werden.

Herbstcampus

Wissenstransfer par excellence

31. August – 3. September 2015
in Nürnberg

Scaling Scrum

Können wir Scrum skalieren und dabei noch Scrum „by the book“ folgen?

von Tim Bourguignon

Eigentlich ist die Scrum-Methodologie schnell erklärt. So lässt sich in wenigen Minuten eine ausreichend verständliche Einführung in Scrum geben. Schwieriger ist es da schon, Scrum einzuführen und konsequent umzusetzen. Muss Scrum dann zusätzlich noch skaliert werden, stellt sich die Frage, ob das überhaupt möglich ist, ohne dabei die Treue zu Scrum zu opfern.

Wo wart ihr denn am 13. Februar 2001? Es war ein Dienstag. Im Juni 2001 war ich mit Abitur dran; oder eher mit dem „Bakkalaureat“ wie es in Frankreich heißt. Es ist schon fast die Hälfte meines Lebens her. Wo war ich genau? Das kann ich nur ahnen: erst im Gymnasium, Dienstag nachmittags hatten wir 2 Stunden Mathematik Unterricht die uns wie 2 oder 3 Ewigkeiten vorkamen. Mit Programmierung hatte ich damals wenig zu tun, höchstens ein bisschen BASIC auf meinem TI-92 Taschenrechner und HTML/Javascript als Web-Master für das Französische Comic-Buch Universal War 1 [1]. An diesem Tag, in Snowbird, einem US-amerikanischen Skigebiet in Utah, waren 17 Männer versammelt, um über ihre Erfahrungen zu quatschen. Was dabei raus kam wissen wir mittlerweile alle: das „Manifest für Agile Software-Entwicklung“, auch das „Agile Manifest“ [2] genannt.

Die Scrum-Methodologie ist im Grunde genommen ziemlich einfach. Chris Sims [3] schafft es in 13 Minuten eine tiefe und doch verständliche Einführung in Scrum zu geben. Die 3 Säulen von Scrum sind Transparenz, Introspektion und Anpassung. Ohne Transparenz können wir keine Introspektion machen. Ohne Introspektion können wir uns nicht an die Wünsche des Kunden anpassen – was das Ziel von Scrum ist. Wenn ich aber Scrum in 3 Konzepte zusammenfassen sollte, würde ich Introspektion und Anpassung in eines zusammenführen und dafür „Technische Exzellenz“ dazu nehmen.

Bei diesem Treffen in Snowbird waren Jeff Sutherland [4] und Ken Schwaber [5], die vermeintlichen Väter von Scrum [6], dabei. Auch Kent Beck [7], der Erfinder des *Extreme Programmings* (XP) [8]. Diese beiden Methoden sind Heutzutage zwei der bekanntesten, mit denen wir Entwickler zu tun haben. Wie J. B. Rainsberger [9] es in seinem Vortrag „7 minutes, 26 seconds, and the Fundamental Theorem of Agile Software Development“ [10] während der Øredev Konferenz [11] in 2013 hervorragend dargestellt hat: „Scrum kann ohne XP nicht funktionieren“, diese beide Methoden gehen Hand in Hand. Scrum fördert Technische Exzellenz.

Scaling Scrum

Die drei Bausteine „Technische Exzellenz“, „Transparenz“ und „Introspektion und Anpassung“ sind als Beschreibung für Scrum also gar nicht so schlecht. Jedoch auf der bereits genannten Seite – *AgileLearningLabs* [12] – wird Scrum auch so beschrieben:

> *„Scrum is a lightweight framework designed to help small, close-knit teams of people develop complex products“.*

Ist damit Scrum für kleine Mannschaften, für Hippies, die keine „richtigen Projekte bauen“, am besten geeignet? Ist es wirklich so?

Der Scrum-Methodologie „by the book“ zu folgen ist mit kleineren Teams „nicht ohne“, jedoch relativ einfach. Die Abhängigkeiten sind überschaubar. Die Kommunikation wird nicht von unnötigen Organisationshürden verhindert. Die Teams sind meistens schon ziemlich selbstorganisiert. Sobald man aber über mehrere Teams, größere Organisation und Agilität außerhalb der Scrum-Teams, also in den Firmen selbst redet, fängt es zu haken. Scaling Scrum ist eine Herausforderung. Daher die Frage: gibt es sowas wie skaliertes „Scrum by the book“ oder ist es ein Oxymoron?

Wer kennt die „Scrum-But“-Antipattern[1]?

> *„Wir machen Scrum, aber da es keine Vollzeit-Rollen sind, spielt unser CTO die PO- und Scrum-Master-Rollen“ oder „Wir machen Scrum, aber da wir – in unserer Firma – etwas besonderes sind, können wir uns diese Daily Scrum Meetings jeden Tag sparen und machen es stattdessen nur zwei mal die Woche“!*

[1] Scrum, aber…

Diese „Scrum-But"-Varianten sehen wir in allen Farben, Größen und Arten von Firmen, und es sind klare Zeichen für ein Missdeutung von Scrum. Die Scrum-Methodologie ist viel einfacher in kleineren Firmen einzusetzen; größere Firmen haben öfter starrere Prozesse, die zuerst invalidiert oder geändert werden müssen – und das geht oft nicht „ohne Blut fließen" zu lassen.

Am besten wäre es, wenn wir für diese Firmen eine schlüsselfertige Lösung hätten. Ein Framework, das in Konferenzen richtig sichtbar wäre, das viel Unterstützung von größeren Firmen hätte und ein paar sehr bekannte Namen als Sponsoren. Somit ließe sich ein solches Framework einfach beim Management verkaufen. Das wäre aber eine eierlegende Wollmilchsau. Viele kluge Köpfe haben es probiert und mehrere Frameworks sind so entstanden.

Das „Scaled Agile Framework" (SAFe® sic.) [13] von Dean Leffingwell [14], „Scrum at Scale" [15] von Jeff Sutherland, „Agility Path" [16] von Ken Schwaber oder etwas konservativer das von IBM und Scott Ambler [17] erfundene „Agility Scaling Model" (ASM) [18], auch „Disciplined Agile Delivery" (DAD) [19] genannt. Kleiner Exkurs, allein das Wort „Disciplined" verursacht zuerst einmal bei mir eine Abschreckreaktion... Heißt dies etwa, dass Scrum sonst anarchisch ist?

Alle diese Frameworks versuchen, das oben genannte Problem zu lösen. Alle haben mehr oder weniger gute Argumente, mischen sich mehr oder weniger in die Organisation ein und bleiben dem Scrum Geist mehr oder weniger treu. Ein anderes Framework allerdings hat bei mir in den letzten Jahren einen wirklich guten Eindruck hinterlassen: LeSS [20].

LeSS

Beim Dubioser-Namens-Contest ist LeSS hoch dabei. LeSS steht nämlich für „Large Scale Scrum" und wird hoch mit „More with LeSS" via die Domain www.less.works vermarktet. Ziemlich nah dran am „Bullshit Bingo", oder? Dafür hat das Framework bei mir ziemlich lang Respektminuspunkte bekommen. Aber je mehr ich mich mit Scrum-Skalierung beschäftigte desto mehr musste ich mich mit LeSS auseinander setzen. Kurz gesagt, LeSS taugt und kann nicht (mehr) ignoriert werden.

LeSS wurde schon 2005 von Craig Larman [21] und Bas Vodde [22] entworfen. Von den oben präsentierten Frameworks und Praktiken ist es dasjenige, das am nähsten an den Scrum-Vorschriften bleibt; die Agile Philosophie bleibt, trotz Skalierung. Wenn man im Enterprise-Umfeld, beim Skalieren von Scrum versucht den Scrum-Regeln zu folgen, landet man ziemlich schnell auf einem solchen Konstrukt wie LeSS.

Warum sollte man über allem versuchen den „Geist" von Scrum – trotz Skalierung – beizubehalten? Scrum fördert das Infragestellen und Revidieren von Entscheidungen und immer wieder die Analyse der Arbeitsweise. Während der Skalierung ist es viel zu einfach Regeln zu setzen und sich „blind" daran zu halten; daher stammt der Wunsch, nah an Scrum zu bleiben, um diese Fallgrube zu vermeiden. Ist LeSS also „Skaliertes Scrum by the book"?

LeSS gibt es in zwei Geschmacksrichtungen: „LeSS" und „LeSS Huge". Die beiden unterscheiden sich hauptsächlich durch die Anzahl an Teams die beteiligt sind: weniger und mehr als 8 Teams. Überhalb dieser Grenze werden zusätzliche Strukturen gebraucht, Zerteilung der Anwendung in behandelbare Teile und Delegation von Aufgaben.

Large Scale Scrum ist Scrum

Alle Scrum-Artefakte werden behalten: ein „Produkt-Backlog", eine „Definition of Done", ein „Potentially Shippable Increment" für alle Teams gleichzeitig (getaktete Sprints von 2 bis 4 Wochen), ein (übergeordneter) „Produkt Owner", nur „Cross-Funktionale Teams". Zusätzlich werden „Sprint Backlogs" und „Sprint-Ziele" pro Team hinzugefügt.

Die Sprint-Termine werden nur leicht angepasst. Es sollte keine Isolierung der Aufgaben entstehen. Daher müssen „Sprint Planung I" und „Backlog Refinement Meetings" für alle Teams mit dem Produkt-Backlog (und nicht ein Team-Backlog) passieren. Es ist leicht eine Abkürzung nehmen zu wollen und den Teams Teile der Software zuzuordnen. Dies hätte den Vorteil dass ein Team sich nicht mehr die gesamte Software anschauen müsste, sondern „nur" sein Spezialgebiet. Allerdings führt dieses Vorgehen zu einer Blockierung der Aufgaben: „Nur Team-X kann Aufgabe X erledigen", was aus mittelfristiger Sicht ein großer Nachteil ist.

Die Koordination ist auch ein Schlüsselelement von LeSS. „Continuous Integration" muss vorhanden sein und das Build-System muss Verantwortlichkeit des Scrum-Teams sein. Der Code muss wie ein internes Open-Source-Projekt behandelt werden, also ohne Ownership. Eine „Community of Practices" sorgt dafür, dass gemeinsame Elemente der Architektur, User Experience etc. Teamübergreifend gelöst werden.

In LeSS-Huge werden Bereiche der Applikation getrennt und unter der Verantwortlichkeit eines Bereich-POs gestellt. Jeder Bereich wird via LeSS behandelt, alle Sprints der LeSS-Huge sind zusätzlich getaktet.

LeSS Organisation

Organisatorisch führt LeSS zu wenigen aber wichtigen Änderungen. Ein Scrum-Team im LeSS-Umfeld ist ein Cross-funktionales Scrum-Team; inklusive z. B. Architekten und Testern. Das heißt, dass traditionelle Karrierepfade von LeSS ziemlich direkt beeinflusst werden.

Traditionell findet ein Ping-Pong statt zwischen Fach- und R&D-Abteilungen, zwischen dem Produktmanager und der Projektleitung. Wegen dieser Interessenskonflikte verschwindet die traditionelle Rolle der Programm-/Projektleiter und wird lieber auf mehrere Köpfe verteilt. An der Spitze des Projekts findet man den Produktmanager, der verantwortlich für die Entwicklung des Produkts ist. Eine neue Rolle „Head of Product Group" taucht aber auf. Diese Person ist der disziplinarische Vorgesetzte der gesamten Projektmannschaft. Seine höhere Position ist sehr wichtig, um dabei zu helfen, organisatorische Probleme zu beseitigen.

LeSS-Erfahrungen

Wie gesagt, wenn man dem Agilen Manifesto und den Scrum-Richtlinien folgt, landet man automatisch bei einem solchen Konstrukt wie LeSS. Schon zweimal habe ich persönlich Scrum skaliert (bis zu 6 Scrum-Teams), und beide Male dieses Muster wiedererkannt. Bei mir waren natürlich ein paar Lösungen anders; z. B. hatten die Teams einen fachlichen Fokus. In der ersten Skalierung waren die Teams Cross-funktional und konnten tatsächlich fachliche Stories von Anfang bis Ende lösen, allerdings in einem bestimmten Bereich der Anwendung: COM Kommunikation oder ImageProcessing. In der zweiten Skalierung sind die Teams ebenso Cross-funktional, allerdings bekommen sie Themen für eine Handvoll Sprints zugewiesen: Team-X kümmert sich 4 bis 5 Sprints lang um Thema Y, danach irgendein anderes Thema, und so weiter. Ich sehe schwer, wie sich 8 Scrum Teams ohne Spezialisierung auf alle Themen vorbereiten können, ohne die Zeiträume von unseren Refinements- und Planungs-Meetings zu sprengen. Da muss ich selber noch forschen.

Fazit

Scrum zu skalieren ist aktuell ein heißes Thema. Jedoch bevor man dies überhaupt angreift, sollte man sich zuerst die Alternative anschauen: „klein aber fein"!

Wenn es allerdings nicht mehr reichen sollte, ist LeSS ein sehr guter Ansatz, der treu an den Vorschriften des Agilen Manifests bleibt.

Skalieren? Ja, aber dabei nicht Eure Agilität verlieren!

Referenzen

[1] Universal War 1, http://uw1.bajram.com/US-series.html

[2] Beck, K. et al. *Manifesto for Agile Software Development,* http://agilemanifesto.org

[3] Sims, Chris *Scrum: A Breathtakingly Brief and Agile Introduction* http://www.agilelearninglabs.com/resources/scrum-introduction

[4] Wikipedia *Jeff Sutherland*, https://en.wikipedia.org/wiki/Jeff_Sutherland

[5] Schwaber, Ken *Ken Schwaber's Blog: Telling It Like It Is* https://kenschwaber.wordpress.com/

[6] Wikipedia *Scrum*, https://de.wikipedia.org/wiki/Scrum

[7] Wikipedia *Kent Beck*, https://en.wikipedia.org/wiki/Kent_Beck

[8] Wikipedia *Extreme Programming,* https://de.wikipedia.org/wiki/Extreme_programming

[9] Rainsberger, J. B. *jbrains*, http://www.jbrains.ca

[10] Rainsberger, J. B. *7 minutes, 26 seconds, and the Fundamental Theorem of Agile Software Development,* https://vimeo.com/79106557

[11] Jayway *Øredev*, http://oredev.org

[12] AgileLearningLabs, http://www.agilelearninglabs.com

[13] Leffingwell et al. *Scaled Agile Framework (SAFe®)* http://scaledagileframework.com

[14] Leffingwell, Dean *Dean Leffingwell,* http://deanleffingwell.com

[15] Scrum Inc. *Scrum at Scale*, http://www.scruminc.com/scrum_at_scale_part_i

[16] Scrum.org *Agility Path*, https://www.scrum.org/Portals/0/Documents/Agility-Path/Agility-Path_Executive-Summary.pdf

[17] Ambysoft *Scott W. Ambler's Home Page* http://www.ambysoft.com/scottAmbler.html

[18] Ambler, Scott *Agility@Scale: Strategies for Scaling Agile Software Development – Introduction to the Agile Scaling Model,* https://www.ibm.com/developerworks/community/blogs/ambler/entry/agile_scaling_model?lang=en

[19] Disciplined Agile Delivery *An agile process decision framework for the enterprise,* http://www.disciplinedagiledelivery.com

[20] The LeSS Company B.V. *Scaling Scrum, Scaled Agile – Large Scale Scrum (LeSS)*, http://less.works

[21] Larman, Craig *Craig Larman* http://www.craiglarman.com/wiki/index.php?title=Main_Page

[22] Vodde, Bas *The Odd-e Blog*, http://blog.odd-e.com/basvodde

Weiterführende Literatur

- Bodamer, Rodney *Comparing Scaled Agile Framework (SAFe®) and Disciplined Agile Delivery (DAD)* http://de.slideshare.net/RodneyBodamer1/scaled-agile-framework-and-disciplined-agile-delivery-comparisons
- Stevens, Peter *Scaling Scrum: SAFe®, DAD, or LeSS?* http://www.scrum-breakfast.com/2013/10/scaling-scrum-safe-dad-or-less.html
- Horvath, Kristof *Scaling Agile in Large Enterprises: LeSS, DAD or SAFe®?* http://intland.com/blog/agile/safe/scaling-agile-in-large-enterprises-less-dad-or-safe
- Larman, Craig; Vodde, Bas *Scaling Agile Development, Large and Multisite Product Development with Large-Scale Scrum*, http://static1.1.sqspcdn.com/static/f/702523/22609354/1367558447003/201305-Larman.pdf

Kurzbiografie

Timothée Bourguignon ist als Senior Consultant für MATHEMA Software GmbH tätig. Sein Spezialgebiet ist die Anwendungsprogrammierung mit dem .NET-Framework. Daneben beschäftigt er sich mit den Neuerungen der .NET-Welt und deren Communities. Als Agilist, Scrum Master und Scrum Coach liegt sein Schwerpunkt auf agilen Methoden, guter Teamarbeit und hoher Arbeitsqualität. Hierbei setzt er insbesondere auf die Philosophie des Software-Craftsmanships.

Herbstcampus

Wissenstransfer par excellence

31. August – 3. September 2015 in Nürnberg

Der **Herbstcampus** ist *die* Konferenz für Software-Entwickler und -Architekten mit den Technologieschwerpunkten .NET und Java.

Der **Herbstcampus** bietet ein umfassendes und hochwertiges Vortragsprogramm mit namhaften Referenten, das den Teilnehmern wichtiges Know-how vermittelt und über sämtliche aktuellen Entwicklungen informiert.

Club der toten Teams
Eine Geschichte über Motivation
Udo Wiegärtner

Erfolgsversprechend
Erfolgreich im Scrum-Team mit VS Team Foundation Server
Neno Loje

Love it, change it, lead it!
Wie Sie vom Kollegen zur Führungskraft werden
Sabine Zehnder

Docker Dir einen
Virtualisierung mit Docker
Simon Sudler

Nummer 5 lebt
Aktuelle Java-Features unter Android nutzen
Thomas Künneth

Das Paarungsverhalten qualitätsbewusster Software-Entwickler
Pair-Programming-Coaching im Großprojekt
Thomas Much

Das Aquarium lässt grüßen
Die Oracle Glassfish-Strategie und wie es mit Java EE 8 weitergeht
Wolfgang Weigend

Container verschiffen
Continuous Delivery mit Docker – revisited
Dr. Halil-Cem Gürsoy

Die Flux Application Architecture
Facebooks Ansatz für Client-side Web Applications
Sven Kölpin

und vieles mehr…

www.herbstcampus.de

(All)gemeine Trickserei

Zustände mit Java Generics

von Róbert Bräutigam

Java Generics sind bei weitem nicht so mächtig wie in Scala oder wie in einer „richtigen“ funktionalen Sprache wie Haskell. Nichtsdestoweniger kann man sehr interessante – wenn auch vielleicht weniger praktische – Lösungen damit programmieren.

Problemstellung: Validierte Objekte

Wie kann man zwischen beliebigen validierten und nicht validierten Objekten unterscheiden, wenn man annimmt, dass die Validierung hohe Kosten verursacht oder nicht mehrmals ausgeführt werden kann.

Hat man beispielsweise die folgenden zwei Methoden

```
public void prepare(Job job);
public void execute(Job job);
```

muss man möglicherweise eine vorgegebene Reihenfolge beachten: man muss zuerst die *prepare*-Methode aufrufen, dann den Job validieren, und erst dann *execute* aufrufen. Die *execute*-Methode darf man also nur nach einer Validierung nutzen.

Man könnte dieses Problem so angehen:

```
public interface Validateable {
  boolean isValidated();
  void validate();
}
public class Job implements Validateable {
  ...
}
public void execute(Job job) {
  if (!job.isValidated()) {
    throw new IllegalStateException(
      "job is not validated"
    );
  }
  ...
}
```

Dies ist aber eine Prüfung zur Laufzeit und kann damit fehlerhaften Code nicht schon zur Übersetzungszeit verhindern. Zusätzlich muss man die Prüfung mit *isValidated* jetzt überall da einbauen, wo man ein validiertes Job-Objekt braucht, und das könnte ja potentiell in mehreren dutzend Methoden sein. Wie könnte man eine Lösung bauen, bei der schon der Compiler dafür sorgt, dass alle Jobs validiert sind, die in *execute* ankommen?

Unveränderbare Objekte

Die Grundidee ist, dass Objekte „unveränderbar“ sind, damit man ihnen eindeutig einen Zustand statisch zuordnen kann. „Unveränderbar“ heißt aber nicht, dass es keine Setter-Methoden gibt; es bedeutet lediglich, dass alle Methoden, die den internen Zustand des Objekts verändern, ein neues Objekt mit den angeforderten Änderungen zurückliefern müssen.

```
public class Job {
  public static Job create() {
    // Ähnlich wie beim „Builder Pattern“
    return ...
  }
  ...
  public Job setName(String name) {
    Job newJob = ...
    return newJob;
  }
}
```

Da der Job sich selbst validieren muss, gibt es natürlich auch eine *validate*-Methode:

```
public Job validate() {
  ...
}
```

Damit kann man das ganze so verwenden:

```
Job job = Job.create().setName("Rechnen");
prepare(job);
job.validate();
execute(job);
```

Das Problem hierbei ist, dass noch immer nicht garantiert werden kann, dass die *validate*-Methode aufgerufen wird.

Zustand als Generic Parameter

Was wäre aber, wenn ein validiertes Objekt einen anderen Typ haben würde als ein nicht validiertes? Anstatt eine *isValidated*-Methode zu haben kann die *Validateable*-Klasse auch so aussehen:

```
public interface Validateable<B extends Bool> {
  Validateable<True> validate();
}
```

Der Typ-Parameter B entscheidet hierbei, ob das Objekt validiert ist oder nicht.

```
public interface Bool {
}
public interface True extends Bool {
}
public interface False extends Bool {
}
```

Natürlich muss dann der Job auch diesen Parameter mitziehen:

```
public class Job<B extends Bool>
  implements Validateable<B>
{
  @Override
  public Job<True> validate() {
    ...
  }

  public Job<False> setName(String name) {
    ...
  }

  public Job<B> setUnimportantFlag() {
    ...
  }
}
```

Job ist also so definiert, dass es garantiert *Job<True>* nach der Validierung zurückliefert, aber alle anderen Methoden, die den validierten Zustand potentiell zerstören, explizit *Job<False>* liefern. Es kann zusätzlich noch Methoden geben die entweder nichts ändern oder die Validierung zumindest nicht beeinflussen; diese liefern das Objekt dann so zurück, wie es vorher war, entweder validiert, oder nicht.

Die Methoden zur Nutzung implementiert man dann so:

```
public void prepare(Job<False> job);
public void execute(Job<True> job);
```

Die Nutzung wäre dann einfach:

```
prepare(Job
  .create()
  .setName("Apfel")
); // Ok
execute(Job
  .create()
  .setName("Apfel")
); // Kompiliert nicht!
execute(Job
  .create()
  .setName("Apfel")
  .validate()
); // Ok
execute(Job
  .create()
  .setName("Apfel")
  .validate()
  .setName("Birne")
); // Kompiliert nicht!
```

Problemstellung: Konfiguration

Denselben Trick kann man auch verwenden, um beliebige statische Zustände darzustellen. Ein Beispiel dafür ist ein Konfigurationsobjekt *JobConf*, das potentiell mehrmals verwendet wird, aber für manche Verwendungen zusätzliche Einstellungen erfordert.

```
public class JobConf {
  JobConf setName(String name) { … }
  JobConf setId(String id) { … }
  JobConf setHost(String host) { … }
  JobConf setRunnable(Runnable job) { … }
}
public void register(JobConf config);
public void execute(JobConf config);
```

Die *register*-Methode darf man nur aufrufen wenn entweder die Id oder der Name und der Host gesetzt sind; und *execute* darf man nur aufrufen, wenn das *Runnable* angegeben ist. Wie kann man das nun darstellen?

Komplexe Zustände

Um alle Anforderungen zu erfüllen muss man hier mehrere Zustandsparameter definieren:

```
public class JobConf<
  I extends Bool,
  H extends Bool,
  J extends Bool
>
```

Das *I* ist True, wenn entweder die Id oder der Name angegeben ist; *H* steht für den Host und *J* für den Job. Die Methoden müssen dann so definiert sein, dass alle nur denjenigen Parameter ändern, den sie tatsächlich beinflussen:

```
public JobConf<True, H, J> setName(String name) {
    ...
}
public JobConf<True, H, J> setId(String id) {
    ...
}
public JobConf<I, True, J> setHost(String host) {
    ...
}
public JobConf<I, H, True> setRunnable(Runnable job) {
    ...
}
```

Die Methoden die dieses Objekt nutzen, können dann so implementiert werden:

```
public void register(JobConf<True, True, ?> config);
public void execute(JobConf<?, ?, True> config);
```

Dann kann man sie so nutzten:

```
register(JobConf
    .create()
    .setName("Apfel")
    .setHost("localhost")
); // Ok
register(JobConf
    .create()
    .setHost("localhost")
); // Kompiliert nicht
register(JobConf
    .create()
    .setRunnable(job)
); // Kompiliert nicht
execute(JobConf
    .create()
    .setRunnable(job)
); // Ok
```

Fazit

Java Generics können also viel mehr als nur Listen von konkreten Typen zu erstellen. Obwohl die parametrisierten Typen zur Laufzeit nicht zur Verfügung stehen und das System selbst vielleicht nicht so mächtig ist wie in anderen Sprachen, kann es doch einige Tricks mitmachen. Man muss nur aufpassen, dass man es dabei nicht zu weit treibt.

Kurzbiographie

Robert Bräutigam ist als Senior-Consultant für die MATHEMA Software GmbH tätig. Er ist seit 1999 als Entwickler und Architekt im Java Enterprise-Umfeld beschäftigt und interessiert sich für leichtgewichtige Lösungen in Technologien sowie im Projektmanagement.

Herbstcampus

Wissenstransfer par excellence

Der Herbstcampus möchte sich ein bisschen von den üblichen Konferenzen abheben und deshalb konkrete Hilfe für Software-Entwickler, Architekten und Projektleiter bieten.

Dazu sollen die in Frage kommenden Themen möglichst in verschiedenen Vorträgen besetzt werden: als Einführung, Erfahrungsbericht oder problemlösender Vortrag. Darüber hinaus können Tutorien die Einführung oder die Vertiefung in ein Thema ermöglichen.

Haben Sie ein passendes Thema oder Interesse, einen Vortrag zu halten? Dann fragen Sie einfach bei info@bookware.de nach den Beitragsinformationen oder lesen Sie diese unter www.herbstcampus.de nach.

Do it yourself

Eigene JPA Mappings mit *Attribute Convertern*

von Werner Eberling

Das letzte Update des JPA Standards wartet mit einer Reihe interessanter neuer Features auf, die in der Entwicklergemeinde bisher leider weitestgehend unbeachtet geblieben sind. Um dies zu ändern soll hier eine dieser Neuerungen, der *Attribute Converter*, mit seinen Anwendungsmöglichkeiten, aber auch seinen Grenzen, genauer vorgestellt werden.

Obwohl die Versionsnummer des letzten Update der Java Persistence API (JPA) eher auf ein Pflegerelease hinweist, verstecken sich eine Reihe interessanter neuer Features in der Version 2.1. Neben der Vervollständigung der *Query Language* bzw. der *Criteria Buider API* und der Möglichkeit zum Aufruf von *Stored Procedures*, ergeben sich speziell im Bereich des Mappings neuen Möglichkeiten, die einen genaueren Blick Wert sind.

Eigene Typ-Mappings

Lange war die Einflussnahme auf die konkrete Abbildung von Attributen auf die Spalten der Zieldatenbank relativ eingeschränkt. Für die gängigen Mappings gab es von Anfang an die passenden Annotationen; sobald die grüne Wiese aber verlassen wurde und JPA Entitäten auf Basis von „kreativen" Datenbanklayouts erzeugt werden mussten, waren oft ziemliche Klimmzüge notwendig. Zwar ist mit Funktionalitäten wie *Embeddables* oder händischer Konvertierung unter Zuhilfenahme von *EntityListenern* fast jede Situation meisterbar, oftmals wären die klassischen *UserTypes*, wie sie z.B. aus Hibernate bekannt sind [1], eine deutlich elegantere Alternative, auf die man in der JPA bisher aber leider schmerzvoll verzichten musste, es sei denn man nutzte herstellerspezifisch Erweiterungen.

AttributeConverter

Mit den in JPA 2.1 neu eingeführten *Attribute Convertern* ist es jetzt auch innerhalb des JEE-Standards möglich, eigene Mappings zu realisieren, wenn auch nur für einfache Attribute. Alles was dafür nötig ist, ist die zu konvertierenden Attribute mit *@javax.persistence.Convert* zu markieren und die Realisierung des Interfaces *javax.persistence.AttributeConverter* [2]. Dabei müssen zwei Methoden implementiert werden. Eine Methode zur Umwandlung des Objektattributs in den Typ der Datenbankspalte und eine Methode für die Rückumwandlung.

```
public interface AttributeConverter<X,Y>{
  Y convertToDatabaseColumn(X attr);
  X convertToEntityAttribute(Y col );
}
```

Listing 1: Interface des Attribute Converters

Einsatzmöglichkeiten

Was ist nun mit diesem, auf den ersten Blick recht unscheinbaren, Interface möglich? Die einfachste Anwendung ist die Nutzung zur Kodierung von Entitätsattributen vor der Übergabe an die Datenbank bzw. zur Dekodierung im umgekehrten Falle. Hierbei können sowohl nicht unterstützte Sonderzeichen mit einem

passenden Encoding ersetzt werden oder jede beliebige Art von Verschlüsselung zum Einsatz kommen.

```
public class StringInvertierer
  implements AttributeConverter<String, String>
{
  @Override
  public String convertToDatabaseColumn(
    String attr
  ){
    return reverse(attr);
  }
  @Override
  public String convertToEntityAttribute(String col) {
    return reverse(col);
  }
  private String reverse(String in) {
    if (in == null) {
      return null;
    }
    return new StringBuilder(in).reverse().toString();
  }
}
```

Listing 2: Invertierte Speicherung von String Attributen

Um den Konvertierer zu nutzen, müssen nun nur noch die entsprechenden Attribute innerhalb der Entität passend annotiert werden.

```
@Entity
public class Kunde{
  @Convert(converter=StringInvertierer.class)
  private String vorname;
  @Convert(converter=StringInvertierer.class)
  private String nachname;
  //...
}
```

Listing 3: Annotierung der Entitätsattribute

Die Konvertierung erfolgt nun sowohl beim Laden und Speichern der Entität, sowie auch im Rahmen von JPQL Abfragen bzw. beim Aufbau von Abfragen durch die Criteria Builder API.

Ersatz für Standardannotationen

Eine spezielle Art der angesprochenen Konvertierung ist die Abbildung von Java Enumerationen mit Hilfe von Konvertern. Im Gegensatz zur Annotation *@Enumerated* stehen hier mehr Möglichkeiten zur Abbildung zur Verfügung. Ist eine Abbildung per Name oder Ordinal nicht möglich, kann das Mapping der Datenbankwerte per Konverter erfolgen. Dies ist beispielsweise hilfreich, wenn ein Statuscode nicht fortlaufend vergeben wird, sondern nur mit speziellen Zahlenwerten arbeitet:

-1: Status unbekannt
10: Auftrag angenommen
15: Auftrag geprüft
20: Abarbeitung begonnen
22: Abarbeitung pausiert
25: Abarbeitung abgebrochen
30: Auftrag abgearbeitet

In diesem Fall bietet es sich an, der Java Enumeration ein eigenes Attribut int *statusCode* zu spendieren, und die Enumeration auf dessen Basis mit Hilfe eines Attribute Converters in die Datenbank zu mappen:

```
public enum AuftragStatus {
  UNBEKANNT(-1), ANGENOMMEN(10), //...
  private final int statusCode;
  AuftragStatus(int statusCode){
    this.statusCode = statusCode;
  }
}
```

Um nun nicht jedes Attribut vom Typ *AuftragStatus* explizit annotieren zu müssen, kann der Converter automatisch auf alle Attribute dieses Typs angewendet werden. Hierzu muss die implementierende Klasse mit der Annotation *@Converter(autoApply=true)* versehen werden. Ist dies geschehen, wirkt der Converter bei allen passenden Attributen von Entitäten innerhalb der *PersistenceUnit*. Natürlich können auch einzelne Attribute wieder von diesem Automatismus ausgenommen werden. Hierzu müssen diese lediglich mit *@Convert(disableConversion=true)* gekennzeichnet werden.

```
@Converter(autoApply=true)
public class StatusConverter
  implements AttributeConverter<
    AuftragStatus,
    Integer
  >
{
  // ...
}
```

Listing 4: Automatische Anwendung des Converters

Aggregation von Attributen

Als letzte Anwendungsmöglichkeit soll nun noch die Aggregation von Attributen einer Entität betrachtet werden. Besteht beispielsweise eine Kundenreferenz aus fachlicher Sicht aus einer Kunden- und einer Vertriebsbereichsnummer, so kann diese in der Objektwelt mit zwei Attributen realisiert werden, auch wenn sie in der Datenbank in lediglich einem Feld (separiert durch ein Trennzeichen) gespeichert wird.

```
@Entity
public class Kunde{
   private Kundenreferenz kdref;
   //...
}

public class Kundenreferenz {
   private int kundennummer;
   private int vertriebsbereich;
   //..
}

@Converter(autoApply=true)
public class KundenreferenzConverter
   implements AttributeConverter<
      Kundenreferenz,
      String
   >
{
   @Override
   public String convertToDatabaseColumn(
      Kundenreferenz kdref
   ) {
      return kdref.getKundennummer() + ":" +
         kdref.getVertriebsbereich();
   }
   @Override
   public Kundenreferenz convertToEntityAttribute(
      String kdrefString
   ) {
      String[] kdrefParts = kdrefString.split(":");
      Kundenreferenz kdref =
         new Kundenreferenz(
            Integer.valueOf(kdrefParts[0]),
            Integer.valueOf(kdrefParts[1])
         );
      return kdref;
   }
}
```

Der auf den ersten Blick recht elegante Ansatz kommt aber mit einem Nachteil im Gepäck daher, der gleichzeitig den Schwachpunkt der *Attribute Converter* aufzeigt. Da in der Datenbank nur ein Feld existiert, können die einzelnen Teilattribute der Kundenreferenz nicht als Pfadausdrücke im Rahmen eines JPQL Queries verwendet werden. Der Ansatz schränkt also, im Gegensatz zur Konvertierung einzelner einfacher Attribute, die Suchbarkeit der Entität empfindlich ein und kann nicht als echter Ersatz für *@Embeddables* gesehen werden. Er ist vielmehr als möglicher Ansatz zum Umgang mit bestehenden Datenstrukturen zu verstehen.

Die Grenzen

Per Definition können *Attribute Converter* nur für einfache Attribute in Entitäten, *Embeddables* oder *Mapped-Superclasses* verwendet werden. Zwar lassen sich, wie oben beschrieben, auch mehrere Attribute auf eine Tabellenspalte abbilden, Ziel ist aber auch hier eine einzelne Spalte, was zu den beschriebenen Einschränkungen bei Abfragen führt.

Sobald sich ein selbstdefiniertes Mapping über mehrere Tabellenspalten erstreckt, kommt dieses Konstrukt an seine Grenzen. Hier können dann nur noch *Embeddables* (ggf. im Zusammenspiel mit *Attribute Convertern*) helfen. Alternativ bleibt natürlich weiterhin der Griff zu provider-spezifischen *Custom-Types*, wodurch aber die Portabilität empfindlich eingeschränkt wird.

Resümee

Attribute Converter bieten eine elegante Möglichkeit Einfluss auf einfache Mappings zu nehmen. Dadurch werden komplizierte Konstrukte aus nicht sichtbaren Datenbankfeldern und *EntityListenern* zur nachträglichen Konvertierung in vielen Fällen überflüssig. Besonders spannende ist in diesem Zusammenhang die damit auftauchende Alternative zu den Mapping-Annotationen *@Enumerated* und *@Temporal*.

Auf den echten *UserType* mit seinen Möglichkeiten auch komplexere Datenbankabbildungen zu realisieren, muss in JPA weiterhin verzichtet werden. Die Entscheidung, ob dies nun gut oder schlecht ist, bleibt dem geneigten Leser selbst überlassen.

Nichtsdestoweniger machen *AttributeConverter* den Umgang mit JPA-Mappings sicher leichter und den daraus resultierenden Code eleganter und lesbarer. Also: Selbst ausprobieren und den eigenen Mapping-Code endlich dahin umziehen, wo er hingehört: in den *Attribute Converter*.

Referenzen

[1] Hibernate *Custom types in Hibernate*
http://docs.jboss.org/hibernate/orm/4.3/manual/en-US/html/ch06.html#types-custom

[2] Javadoc *Attribute Converter*
https://docs.oracle.com/javaee/7/api/javax/persistence/AttributeConverter.html

Kurzbiografie

Werner Eberling ist Principal Consultant und Technical Lead bei der MATHEMA Software GmbH in Erlangen. Seit 1999 beschäftigt er sich mit verteilten Systemen, sein Fokus liegt dabei auf CORBA und der JEE. Neben seiner Projekttätigkeit hält er Technologie-Trainings und ist als Sprecher auf nationalen und internationalen Konferenzen anzutreffen. Sein aktuelles Steckenpferd ist die Anwendungsentwicklung für mobile Endgeräte bzw. deren Anbindung an Enterprise Systeme. Er ist u.a. Autor des Buches „Enterprise JavaBeans 3.1 Das EJB-Praxisbuch für Ein- und Umsteiger“, erschienen im Hanser Verlag.

Des Programmierers kleine Vergnügen

Die spinnen, die Römer!

Auch wenn Obelix mit seinem berühmten Ausspruch eher die Taten der in Gallien stationierten, römischen Legionäre gemeint haben mochte, so lässt sich seine Aussage problemlos auch auf die römischen Zahlen übertragen. Zum Glück lässt sich eine Zahl trotzdem relativ leicht in sein römisches Pendant umwandeln.

Wer rechnen muss, der ist mit unserem Stellenwertsystem schon gut bedient. Insbesondere mit der Null als Platzhalter oder als Anzahl für nicht vorhandene Elemente lassen sich souverän so viele Probleme derart lösen, dass sich alles, was mit Rechnen zu tun hat, sehr schön mechanisieren lässt. So grenzt es eigentlich an ein Wunder, dass es trotz der Bemühungen von Leonardo Fibonacci, der schon 1202 in seinem Buch *Liber abaci* die Vorzüge der indischen Zahlen zu loben wusste, noch weitere 300 Jahre mit Rechentabellen gearbeitet werden musste. Erst Adam Reis gelang es dann endlich Anfang des 16. Jahrhunderts durch seine strukturierten Rechenmethoden die römischen Zahlen aus dem praktischen Leben zu verbannen.

Für uns spielen die römischen Zahlen nur noch aus dekorativen Gründen eine Rolle. So finden sie sich etwa als Seitenzahlen, bevor es in einem Buch richtig zur Sache geht, oder eingemeißelt in Gebäuden, um exponiert deren Alter zu verraten.

Bei den römischen Zahlen stehen lateinische Buchstaben für Zahlenwerte. Diese werden der Größe nach derart aufgelistet, dass höchsten vier gleiche Buchstaben hintereinander stehen. So stehen die Buchstaben M, D, C, L, X, V und I für die Zahlenwerte 1000, 500, 100, 50, 10, 5 respektive 1. Letztes Jahr (2014) hätten wir demnach MMXIIII geschrieben.

Das wäre ja ganz einfach, wenn man sich nicht irgendwann die Subtraktionsregel überlegt hätte, damit nur noch höchsten drei gleiche Zeichen hintereinander stehen. Mit deren Hilfe lässt sich etwa für VIIII = 5 + 4 kürzer IX = 10 – 1 schreiben. Die Subtraktionsregel lässt I vor V oder X, X vor L oder C und schließlich C vor D und M zu. Das für die deutschen Fußballer so erfolgreicher Jahr 1974 lässt sich also statt MDCCCCLXXIIII deutlich kürzer, aber nicht unbedingt übersichtlicher, als MCMLXXIV schreiben.

Wie konvertiert man nun eine Zahl in sein römisches Äquivalent? Im Internet kursieren einige abenteuerliche, aber nicht weniger funktionstüchtige Lösungen. Aber im Grunde ist alles ganz einfach, wenn man nur die Zahlen bis 10 betrachtet.

Ist die Zahl kleiner oder gleich 3, dann kann die additive Methode verwendet werden. Für die 4 wird die Subtraktionsmethode verwendet. Die 5 steht allein da und die Zahlen bis einschließlich 9 werden vergleichbar zu den Zahlen bis ausschließlich 5 behandelt. Und nicht zuletzt passiert bei der 0 nichts.

```
void toRoman(StringBuilder s, int x) {
  if (x <= 3) {
    for (int i = x; i > 0; i--) {
      s.append("I");
    }
  } else if (x == 4) {
    s.append("IV");
  } else if (x < 9) {
    s.append("V");
    for (int i = x – 5; i > 0; i--) {
      s.append("I");
    }
  } else if (x == 9) {
    s.append("IX");
  }
}
```

Interessant bei diesem Ansatz ist, dass diese Regeln für X, V und I gleichermaßen für C, L und X gelten. Und für M, D und C. Wären doch nur die konstanten Zeichenketten Variablen. Aber das lässt sich ja problemlos einrichten. Der Einfachheit halber sollen die Variablen so wie die Zeichenketten X, V und I heißen. Besser wären natürlich Z, F und E für Zehner, Fünfer bzw. Einer, aber so benutzen wir eben die römischen Bezeichnungen.

```
void toRoman(
  StringBuilder s, int x, char X, char V, char I
) {
```

```
    if (x <= 3) {
      for (int i = x; i > 0; i--) {
        s.append(I);
      }
    } else if (x == 4) {
      s.append(I);
      s.append(V);
    } else if (x < 9) {
      s.append(V);
      for (int i = x – 5; i > 0; i--) {
        s.append(I);
      }
    } else if (x == 9) {
      s.append(I);
      s.append(X);
    }
  }
```

Um nun eine Zahl *x* (mit *x* < 10) in eine römische umzuwandeln, kann man die Routine wie folgt aufrufen:

```
StringBuilder s = new StringBuilder();
toRoman(s, x, 'X', 'V', 'I');
```

Die zu konvertierenden Zahlen sollen natürlich auch größer sein können. In diesem Fall müssen aber die Stellen aus dem Dezimalsystem auf die Zahlen 0 bis 9 und den dazugehörigen passenden Buchstaben abgebildet werden. Dabei muss mit der höchstwertigsten Ziffer begonnen werden.

```
public static String toRoman(int x) {
  StringBuilder s = new StringBuilder();
  while (x >= 1000) {
    s.append('M');
    x -= 1000;
  }
  toRoman(s, (x / 100) % 10, 'M', 'D', 'C');
  toRoman(s, (x / 10) % 10, 'C', 'L', 'X');
  toRoman(s, x % 10, 'X', 'V', 'I');
  return s.toString();
}
```

Bis auf winzige Unterschiede scheint hier alles doppelt vorzukommen. Leider lässt sich das auch nicht vermeiden, aber zumindest lassen sich die maximal zu durchlaufenden Tests minimieren.

```
if (x < 5) {
  if (x == 4) {
    s.append(I);
    s.append(V);
  } else {
    for (int i = x; i > 0; i--) {
      s.append(I);
    }
  }
} else {
  if (x == 9) {
    s.append(I);
    s.append(X);
  } else {
    s.append(V);
    for (int i = x – 5; i > 0; i--) {
      s.append(I);
    }
  }
}
```

Der Vollständigkeit halber sei noch erwähnt, dass es auch für 5 000 (ↁ), 10 000 (ↂ), 50 000 und 100 000 entsprechende Buchstaben oder Buchstabenkombinationen gibt. Diese haben übrigens auch – neben einer alternativen Darstellung für 1000 (ↀ) und einer umgedrehten 100 (Ↄ) – Einzug in den Unicode gehalten. Allerdings lassen sich 50 000 und 100 000 in Ermangelung eines geeigneten Zeichensatzes hier nicht darstellen.

Mit den Werten für 5 000 und 10 000 lässt sich obige Routine aber leicht verallgemeinern, was insbesondere im Zusammenhang mit Jahreszahlen sehr nützlich ist, die ja neuerdings den Bereich über 2000 benötigen. Dazu muss nur in der Schleife die Grenze und das dazugehörige Zeichen getauscht und die folgende, zusätzliche Zeile eingefügt werden.

```
while (x >= 10000) {
  s.append('ↂ');
  x -= 10000;
}
toRoman(s, (x / 1000) % 10, 'ↂ', 'ↁ', 'M');
```

Aus 9999 wird dann etwa MↂCMXCIX. Auf jeden Fall lassen sich damit alle Zahlen bis 39 999 = ↂↂↂMↂCMXCIX korrekt darstellen. Wer darüber hinaus tatsächlich noch größere Zahlen braucht, der kann einfach anstatt M die Kombination CIↃ benutzen und damit auch das Zeichen ↂ durch CCIↃↃ ersetzen [1]. Mit diesem Verfahren lässt sich dann auch eine Millionen als CCCCIↃↃↃↃ darstellen. Aber diese großen Zahlen sind dann doch etwas – na ja – umfangreich.

Referenzen

[1] Wikipedia *Römische Zahlschrift*
https://de.wikipedia.org/wiki/R%C3%B6mische_Zahlschrift

Kurzbiographie

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

Kaffeesatz

Es kommt alles wieder

von Lisa-Marie Wehlmann-Wiedeking

Obwohl ich eine ungeklärte Abneigung gegen Schwarzweißfilme habe, habe ich mir vor kurzem einen Film aus den 30er Jahren angesehen. Eine der Figuren fragte, wer ein Telefon hätte und es waren wirklich wenige, die sich daraufhin meldeten.

Als *Digital Native*, in die digitale Welt hineingewachsen, ist das für mich natürlich kaum vorstellbar. Obwohl ich mich an die Knochentelefone meiner Eltern erinnern kann. Diese orange-leuchtenden Displays und die Tasten, die man tatsächlich noch drücken musste. Meine ersten beiden Handys waren auch so, und da ich jetzt erst beim dritten bin, liege ich nicht gerade in der Norm. Trotzdem habe ich mir eine Beobachtung erlaubt.

Erst im Jahr 2013 wurden mehr Smartphones als herkömmliche Mobiltelefone verkauft. Seltsam, dass sich trotzdem kaum mehr jemand, der ebenfalls in einer dieser Internet-Generationen ist, vorstellen könnte, ein normales Handy zu besitzen. Ich habe schon Grundschüler mit Smartphones gesehen und welche ohne, die sich aufgeregt haben, dass sie keine Fotos machen können.

Es gibt tatsächlich auch eine eigene Kategorie von Kinder-Apps, und Kleinkinder, die sich nur mit dem iPad beschäftigen können. Ich war geschockt, dass der kleine Sohn von Bekannten nicht anders zu beruhigen war als mit dem Tablet. Selbst zu meiner Zeit hat man noch Bücher angeschaut, Bauklötze aufgebaut und umgeschmissen und es war verpönt, wenn Kinder zu viel Fernsehen schauen. Allerdings sagen mir manche Leute heute noch, dass ich zu viel fernsehe.

Natürlich kann jetzt jeder *Digital Immigrant*, der die digitale Welt ja erst als Erwachsener kennengelernt hat, sagen, dass es zu seiner Zeit höchstens drei Fernsehprogramme gegeben hat und überhaupt nur ein Festnetz. (Ich finde es wahnsinnig komisch, dass es Abends überhaupt keine Fernsehübertragung mehr gab.) So gesehen sind die Kinder früher auch ohne Fernsehen klar gekommen und da es neu war, sind die Vorbehalte noch in der älteren Generation verankert. Es ändern sich die Verhaltensweisen aber immer mit dem Angebot, man sehe sich das Tablet-Baby an, wo ich Vorbehalte habe.

Ich kenne natürlich Filme, in denen gerade die ersten Mobiltelefone vorkommen. Riesig, mit Antenne. Wahrscheinlich in etwa so robust wie heutzutage außer einem Stein nichts mehr. Damals, als die Mobiltelefone nicht mehr im Kofferraum eines Autos transportiert werden mussten, sondern endlich wirklich mobil wurden, soll heißen, man konnte sie tatsächlich ans Ohr heben, begann die Faszination des Mobiltelefonierens. Mit dem Smartphone ist es nicht viel anders, eine Faszination des mobilen Internets. Das mobile Verknüpftsein mit anderen scheint die Menschen zu interessieren, in verschiedenen Zeiten nur durch unterschiedliche Mittel.

Allerdings stelle ich mit Erschrecken fest, dass es eine Art rückläufigen Trend zu geben scheint. War es früher noch ein Wettbewerb, immer kleinere Handys herzustellen (keiner wollte so einen Knochen umhertragen), werden sie jetzt wieder riesig. Ich habe vergleichsweise lange Finger, aber ich kann deshalb trotzdem nicht mit dem Daumen alle Stellen des Displays erreichen (alle, die fernsehen, wissen welche Sorte Smartphone ich habe) und meines ist schon älter. Die sogenannten *Phablets*, diese Telefone und Tablets in einem, waren eine Zeit lang in Mode, jetzt sind die Smartphones einfach groß.

Ich persönlich sehe fast keinen Mehrwert in Tablets und finde ein Telefon, das nicht in die Hosentasche (oder in eine Handtasche) passt, wirklich einfach nur unpraktisch. Da fehlt mir der mobile Teil von Mobiltelefon; auf smart kann ich bei zu groß gerne verzichten. Wenn ich jetzt also Menschen mit solchen Riesentelefonen sehe, frage ich mich, ob sie vielleicht besser tippen können, wenn die Tastatur größer ist, besser sehen, weil mehr Bildschirm da ist. Wer weiß.

Auch in Mode (oder schon wieder nicht mehr) sind Telefonhörer zum Smartphone, mit Ringelschnur und allem. Einfach in die Kopfhörerbuchse des Smartphones einstecken und das Vintage-Gefühl verspüren. Fehlt nur noch, dass sie wieder Tasten einführen. Schließlich bringt der demographische Wandel zunehmend ein Publikum, das nicht so leichtfertig tippen und sehen kann. Allerdings sind die Trends so kurzlebig, dass ich lieber abwarte, was sich als wirklich nützlich herausstellt. Oder, was ästhetisch ist. Ich möchte nämlich für das Geld, was sie für ihre seltsamen Erfindungen verlangen, auch etwas Schönes haben, wenn es dann gegebenenfalls schon nicht in die Tasche passt.

Falls die Tasten wiederkommen, empfehle ich als Inspiration alte Filme. Schwarz-Weiß sind die Telefone ja meist schon (über Handy-Hüllen-Trends könnte man ein Buch füllen, also seien sie hier unberücksichtigt), also warum nicht zu den Wurzeln zurückkehren? Obwohl, damals waren die Telefone vielleicht sogar bunt. Kann man ja nie wissen bei Schwarzweißfilmen. Ich muss dazu mal einen Zeitzeugen befragen. Man hat mir absurde Geschichten erzählt, dass man sehen konnte, dass die Heldin ein rotes Kleid trägt. An einem Grauton; na klar.

Die Abneigung kommt übrigens wahrscheinlich daher, dass ich neben den bunten Büchern, die ich mir angeschaut habe als Kind, auch bunte Spielsachen hatte und im Fernsehen nur bunte Trickfilme anschauen wollte. Echte Menschen waren mir damals im Fernsehen suspekt, umso mehr also komische graue Menschen. Auch heute schaue ich mir nicht unbedingt Schwarzweiß an. Aber ich versuche diese Abneigung langsam zu überwinden, es kann nämlich sein, dass ich Pech habe: Es kommt schließlich alles wieder.

Kurzbiographie

Lisa-Marie Wehlmann-Wiedeking hat gerade die elfte Klasse hinter sich. Sie kann zwar Schwarz-Weiß-Filme nicht leiden, versucht aber in diesen Sommerferien ihr diesbezügliches Defizit aufzuholen.

Herbstcampus

Wissenstransfer par excellence

User Groups

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

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

Java User Groups

Deutschland

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Schweiz

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

.NET User Groups

Deutschland

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

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

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

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

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

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

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

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

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

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

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

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

Österreich

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

Software Craftsmanship Communities

Deutschland, Schweiz, Österreich

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

Metropolregion Nürnberg
JUG

Die Java User Group
Metropolregion Nürnberg
trifft sich regelmäßig einmal im Monat.

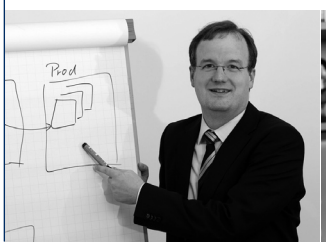
Thema und Ort werden über
www.jug-n.de
bekannt gegeben.

Weitere Informationen
finden Sie unter:
www.jug-n.de

MATHEMA

meet the experts of enterprise infrastructure

- **HTML5, CSS3 und JavaScript**
  21. – 24. Sept. 2015, 1.650,– € (zzgl. 19 % MwSt.)
- **Entwicklung mobiler Anwendungen mit iOS**
  5. – 7. Oktober 2015, 1.250,– € (zzgl. 19 % MwSt.)
- **Fortgeschrittenes Programmieren mit Java**
  Ausgewählte Pakete der Java Standard Edition,
  19. – 21. Oktober 2015, 1.350,– € (zzgl. 19 % MwSt.)
- **Anwendungsentwicklung mit der Java Enterprise Edition**
  30. Nov. – 4. Dez. 2015, 2.150,– € (zzgl. 19 % MwSt.)
- **AngularJS**
  7. – 8. Dezember 2015, 950,– € (zzgl. 19 % MwSt.)
- Unser Trainingsprogramm für 2016 finden Sie ab September auf unserer Homepage.



Lesen bildet. **Training macht fit.**

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

MATHEMA

„Die Herausforderung, jeden Tag etwas Neues zu lernen, habe ich gesucht und bei MATHEMA gefunden."

Tim Bourguignon, Senior Consultant

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

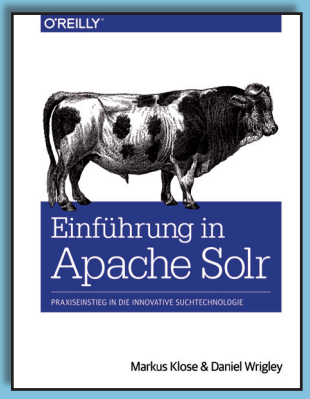
join the experts of enterprise infrastructure

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

Beautiful Code

ISBN 978-3-95561-514-7, 49,90 €

ISBN 978-3-95561-540-6, 39,90 €



ISBN 978-3-95561-421-8, 39,90 €

ISBN: 978-3-95561-141-5, 49,90 €

ISBN 978-3-95561-740-0, 39,90 €

ISBN 978-3-89721-448-4, 49,90 €

ISBN 978-3-95561-986-2, 49,90 €

ISBN 978-3-95561-596-3, 49,90 €

ISBN 978-3-95561-115-6, 39,90 €

ISBN 978-3-95875-037-1, 24,90 €

ISBN 978-3-95561-812-4, 39,90 €

ISBN 978-3-89721-567-2, 24,90 €

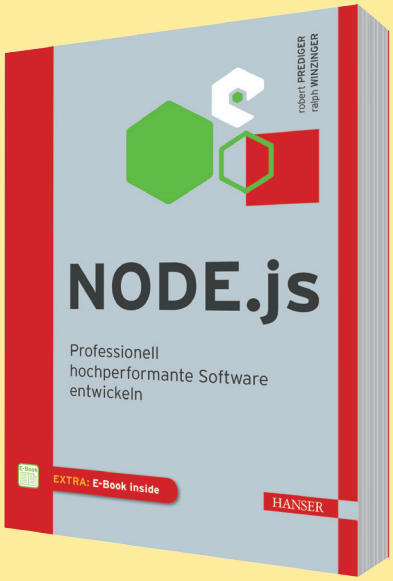
(Fast alle) O'Reilly-Bücher gibt es auch als E-Book: 20% günstiger & 100% DRM-frei!

O'REILLY®

www.oreilly.de

HANSER

Entwickeln, Programmieren und Analysieren

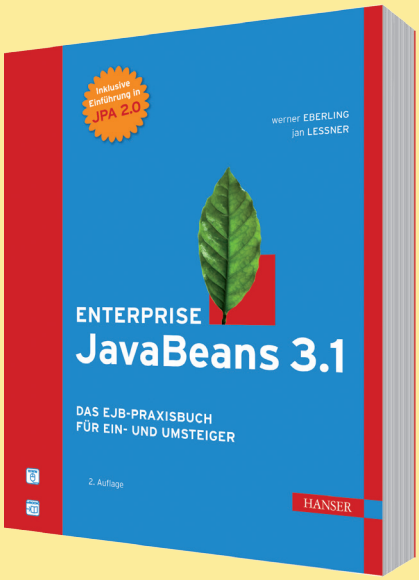


Robert Prediger, Ralph Winzinger

Node.js

Professionell hochperformante Software entwickeln

ISBN 978-3-446-43722-7



Werner Eberling, Jan Leßner

Enterprise JavaBeans 3.1

Das EJB-Praxisbuch für Ein- und Umsteiger

2., aktualisierte und erweiterte Auflage.

ISBN 978-3-446-42259-9

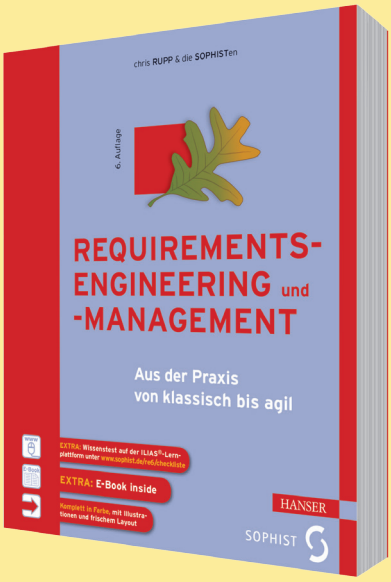
Stefan Zörner

Softwarearchitekturen dokumentieren und kommunizieren

Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten

2., aktualisierte und erweiterte Auflage.

ISBN 978-3-446-44348-8



Chris Rupp, die SOPHISTen

Requirements-Engineering und -Management

Aus der Praxis von klassisch bis agil

6., aktualisierte und erweiterte Auflage.

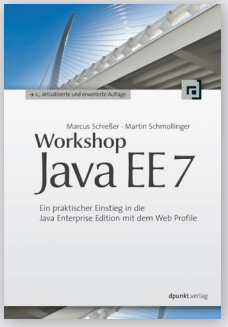
ISBN 978-3-446-43893-4

Abscannen und mehr entdecken!

www.hanser-fachbuch.de/computer

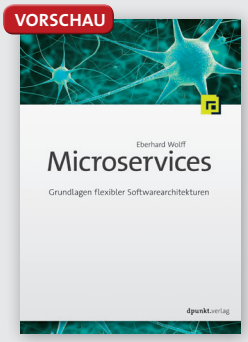
Softwareentwicklung

3. Auflage
2015 · 330 Seiten · € 37,90 (D)
ISBN 978-3-86490-120-1

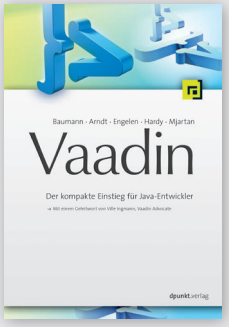


2. Auflage
2015 · 408 Seiten · € 34,90 (D)
ISBN 978-3-86490-195-9

2015 · 252 Seiten · € 32,90 (D)
ISBN 978-3-86490-256-7



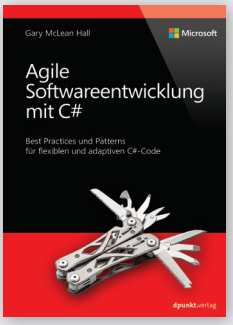
Nov. 2015 · ca. 300 Seiten · ca. € 32,90 (D)
ISBN 978-3-86490-313-7



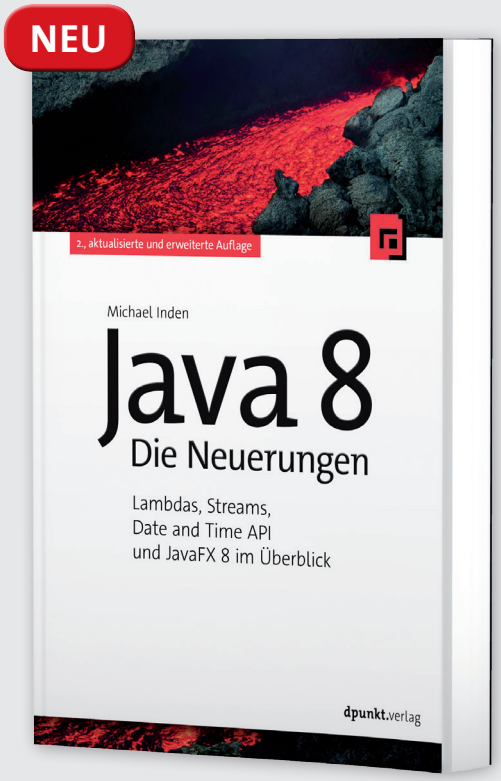
2015 · 280 Seiten · € 34,90 (D)
ISBN 978-3-86490-206-2

Programmierung

2015 · 296 Seiten · € 34,90 (D)
ISBN 978-3-86490-169-0

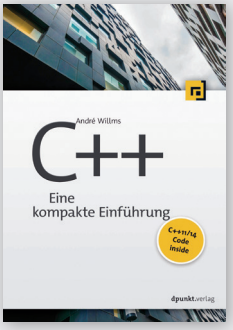


2015 · 462 Seiten · € 39,90 (D)
ISBN 978-3-86490-285-7
(Microsoft Press)



2. Auflage
2015 · 320 Seiten · € 24,90 (D)
ISBN 978-3-86490-290-1

4. Auflage
2014 · 318 Seiten · € 29,90 (D)
ISBN 978-3-86490-227-7

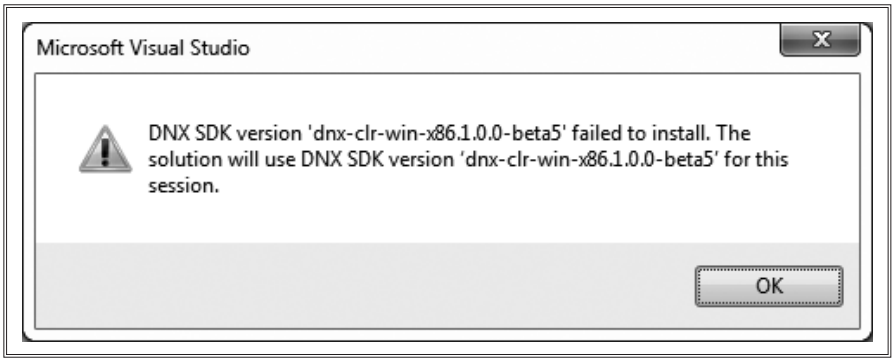


2015 · 428 Seiten · € 22,90 (D)
ISBN 978-3-86490-229-1

plus+ Buch + E-Book: www.dpunkt.de/plus

dpunkt.verlag GmbH · Wieblinger Weg 17 · D-69123 Heidelberg
fon: 0 62 21 / 14 83 40 · fax: 0 62 21 / 14 83 99 · e-mail: bestellung@dpunkt.de

Das Allerletzte



Dies ist kein Scherz!
Diese Fehlermeldung wurde tatsächlich in der
freien Wildbahn angetroffen.

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

Der nächste KaffeeKlatsch erscheint im September.

Herbstcampus

Wissenstransfer par excellence

31. August – 3. September 2015
in Nürnberg

veranstaltet von:

Bookware

KaffeeKlatsch

unterstützt durch:

MATHEMA

HANSER

dpunkt.verlag

O'REILLY®

Rheinwerk

open source PRESS

NIK Nürnberger Initiative für die Kommunikationswirtschaft

ASQF Arbeitskreis Software-Qualität und -Fortbildung e.V.