KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

ISSN 1865-682X

03/2014

Jahrgang 7

Editorial

Call for P…

Wenn man im akademischen Umfeld einen Vortrag halten will, dann schreibt man in der Regel einen Artikel und reicht diesen als Vorschlag ein. Dieses *Paper* wird dann vom Programmkomitee unter die Lupe genommen und – wenn für gut befunden – ins Programm aufgenommen. Dann haben die Autorinnen und Autoren die Gelegenheit auf der Konferenz das Papier in meist 20 Minuten vorzustellen.

Bei den eher praktisch orientierten Konferenzen für unsere Branche liegt die Länge der Vorträge meist irgendwo um die 45 Minuten. Die Grundlage für die Auswahl ist in der Regel das *Abstract*, eine kurze Zusammenfassung von dem, was in dem Vortrag geboten werden soll. Das ist nicht immer ganz einfach, weil zwar in der Kürze die Würze liegt, aber leider zu wenig über die Darreichungsform aussagt.

Beim Herbstcampus haben die Vorträge eine Länge von 70 Minuten. Das ist verhältnismäßig lang und bedeutet leider, insbesondere weil uns auch die Pausen sehr wichtig sind, dass wir vor dem Abendprogramm nur sechs Vorträge schaffen. Aber die 70 Minuten erlauben uns eine (neue) Technologie nicht nur vorzustellen sondern auch ein wenig zu vertiefen, über Erfahrungen oder Probleme zu sprechen und eine Bewertung vorzunehmen – sozusagen konkret zu werden.

Den kurzen Abstracts, die mit oft 300 bis 700 Zeichen natürlich deutlich kürzer als die mehreren Seiten sind, die von einem Paper erwartet werden, gelingt es natürlich nicht immer, zu beschreiben, was genau in dem Vortrag erzählt wird. Aber wenn es ihnen gelingt, in mehr als 80 % der Fälle den richtigen Eindruck zu vermitteln, dann sind wir schon sehr zufrieden.

Damit die bekannte Abkürzung *CfP* für den *Call for Papers* auch dann benutzt werden kann, wenn man nur ein Abstract haben möchte, will man auch schon einmal zu einem *Proposal* aufrufen. Wie man das nun auch nennen mag, mit unserem Beitragsaufruf möchten wir alle, die etwas zu erzählen haben, dazu animieren, eine ausreichend kurze Beschreibung einzureichen, die nicht nur dem auswählenden Komitee, sondern auch dem potenziellen Besucher des Vortrags eine Entscheidung ermöglicht, ob sich eine Aufnahme ins Programm bzw. ein Besuch lohnen könnte.

Dabei ist uns alles Interessante recht: eine bekannte oder neue Technologie, ein etabliertes oder neues Vorgehen, Bibliotheken, Frameworks und alles andere, was uns bei der Arbeit als Software-Entwickler helfen kann, also auch vielleicht etwas über Kommunikation, Dokumentation, Präsentation und Benehmen. Wir nehmen übrigens auch „halbe“ Vorträge mit 35 Minuten, falls jemand etwas zu erzählen hat, es aber nicht unnötig in die Länge strecken möchte. Darüber hinaus haben wir auch ganztägige Tutorien, die natürlich einen völlig anderen Blick auf ein Thema werfen können.

So möchten wir also alle, die ihr Wissen teilen möchten, herzlich dazu einladen, auf dem Herbstcampus einen Vortrag zu halten. Und wenn Sie sich nicht sicher sind, ob ein Thema für den Herbstcampus geeignet ist, reichen Sie doch einfach ein Abstract bis zum 5. Mai bei uns ein.

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 09131/8903-16 gebucht werden.

Abonnement

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

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

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

Copyright

Impressum

KaffeeKlatsch Jahrgang 7, Nummer 3, März 2014
ISSN 1865-682X
Bookware – eine Initiative der
MATHEMA Verwaltungs- und Service-Gesellschaft mbH

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

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

Inhalt

Artikel

Top 3 Fehler in echtem C#-Code

Ricos Philosophie heißt „Der gerade Weg führt zum Erfolg“ und durchdringt das Design von *C#* und dem *.NET-Framework*: der einfache Weg soll stets zur korrekten Lösung führen. Nun hat noch niemand eine universelle Sprache erfunden, mit der man keine Fehler machen könnte; auch das tatsächliche Design von C# und .NET hält noch immer einige Fallstricke bereit.

Kolumnen

Terzett zu zweit

Unbezahlbar

Top 3 Fehler in echtem C#-Code

von Eric Lippert

Im starken Gegensatz zum anstrengenden Weg zum Erfolg, der für eine Gipfelersteigung oder den Weg durch eine Wüste typisch ist, wollen wir es unseren Anwendern ganz einfach machen, bewährten Prinzipien zu folgen – indem Sie unsere Plattform und Frameworks nutzen.

Rico Mariani, Architekt von .NET, 2003

Ricos Philosophie heißt „Der gerade Weg führt zum Erfolg" und durchdringt das Design von *C#* und dem *.NET-Framework*: der einfache Weg soll stets zur korrekten Lösung führen. Nun hat noch niemand eine universelle Sprache erfunden, mit der man keine Fehler machen könnte; auch das tatsächliche Design von C# und .NET hält noch immer einige Fallstricke bereit.

Wenn man mit echtem Code arbeitet, sieht man die gleichen Fehler wieder und wieder auftauchen. Im Folgenden drei meiner Favoriten: Zuerst die Schwächen von *class libraries* im .NET-Framework, dann ein Problem mit dem Sprachen-Design und zuletzt ein überraschend häufiges, aber mangelhaftes Coding-Muster.

Platz 3: Fehlerhafte Zufallszahlen

```
public static int RollSixSidedDie()
{
  var random = new Random();
  return random.Next(1, 6);
}
```

Bevor Sie weiterlesen – Sehen Sie die Fehler in dieser einfachen Methode?

Dieser Fehler wird häufig auf *StackOverflow.com* gemeldet. Wenn man den Code in einer Schleife aufruft, wird normalerweise immer die gleiche Zahl ausgegeben – nicht gerade besonders zufällig. Der Pseudo-Zufallszahlengenerator, den der Standard-Konstruktor erstellt, wird von einem Zeitstempel gefüttert, der lediglich auf eine Millisekunde genau ist. Da Computer in einer Millisekunde aber Millionen von Rechenoperationen durchführen können, wird mit ziemlich hoher Wahrscheinlichkeit bei einer engen Schleife jede neue Instanz von *Random* mit dem gleichen *Input* gefüttert.

Der zweite Fehler ist, dass die zurückgegebene pseudo-zufällige Zahl „größer oder gleich dem *minValue* und kleiner als der *maxValue*" ist, wie es in der Dokumentation heißt. Dieser Code produziert also eine Zahl zwischen eins und fünf!

Eine korrektere Art, diesen Code zu schreiben, wäre:

```
static Random random = new Random();
public static int RollSixSidedDie()
{
  return random.Next(1, 7);
}
```

Doch auch diese Schreibweise bringt Probleme mit sich, da die *Random*-Klasse nicht *thread*-sicher ist – obwohl man von einer solch simplen statischen Methode *Thread*-Sicherheit erwarten könnte.

Diese Klasse lässt ihren Benutzer ins Messer laufen – denn der natürlichste Weg, Code zu schreiben, ist hier der falsche. Die Klasse hätte so entworfen werden sollen, dass sie das, was der Nutzer am meisten benötigen wird – eine Sequenz pseudo-zufälliger Zahlen in einer bestimmten Größenordnung – ohne diese Stolperfallen liefert.

Platz 2: Fehlerhafter Umgang mit Closures

Dieses Problem wurde in den C#-Versionen 3 und 4 am häufigsten gemeldet als „Ich habe einen *Bug* im *Compiler* gefunden". Der Code ist hier stark vereinfacht, um das

Problem deutlicher zu zeigen – im echten Code tritt er mit einiger Häufigkeit auf:

```
var multipliers = new List<Func<int, int>>();
var multiplicands = new[] {10, 20, 30};
foreach(int m in multiplicands)
{
  multipliers.Add( x => x * m );
}
var timesTen = multipliers[0];
Console.WriteLine(timesTen(50));
```

Man denkt, es würde 500 ausgegeben, in C# 3 und 4 gibt es jedoch 1500 aus. Sehen Sie warum?

Der *Lambda*-Ausdruck `x => x * m` bedeutet „multipliziere *x* mit dem aktuellen Wert von *m*", und nicht „multipliziere *x* mit dem Wert, den *m* hatte, als der Delegat erstellt wurde". Ein Informatiker würde sagen: *Lambda*-Ausdrücke werden über einer Variable geschlossen, nicht über einem Wert.

Dieses Problem wurde dem C#-Compiler-Team so häufig gemeldet, dass das Sprachen-Design mit C# 5 geändert wurde, um es zu verhindern. *Lambdas* schließen noch immer über Variablen, aber die *foreach*-Schleife wurde so angepasst, dass sie bei jedem Durchlauf eine neue Variable erstellt. Das entsprechende Problem bei regulären Schleifen besteht noch immer.

In diesem Fall hat jedes einzelne Feature der Sprache für sich seinen eigenen Sinn, aber die Kombination der Features bereitet dem Nutzer Schwierigkeiten.

Platz 1: Fehlerhafte Null-Referenz-Prüfung

Dies ist ein wirklich häufiger Fehler. Die C#-Analyse-*Engine* von Coverity findet einen davon in 4.000 – 5.000 analysierten echten Code-Zeilen. Das Beispiel ist ebenfalls stark vereinfacht, um den Fehler zu verdeutlichen:

```
public Animal FindAnimal(
  string name, List<Filter> filters, bool onlyFish
)
{
  // Performance optimization for the common case of
  // looking for a fish with no filters:
  if (onlyFish && (filters == null || filters.Count == 0))
  {
    return fish[name];
  }
  Animal result = animals[name];
  if (onlyFish && !(result is Fish))
  {
    return null;
  }
  foreach(var filter in filters)
  {
    if (!filter(result))
    {
      return null;
    }
  }
  return result;
}
```

Sehen Sie den Fehler?

Die Abfrage, ob *filters* null ist, legt nahe, dass der Autor dieses Codes erlaubt, dass das Argument null ist. Dies hätte einen Programmabsturz zur Folge, würde die *foreach*-Schleife mit diesem Wert ausgeführt. Der Verfasser muss eine von drei Sachen im Sinn gehabt haben:

1. *filters* darf dann null sein, wenn *onlyFish* wahr ist, sonst jedoch nicht
2. *filters* darf immer null sein
3. *filters* darf niemals null sein

Im ersten Fall hat sich der Entwickler selbst ein Bein gestellt. Die Methode sollte wahrscheinlich in zwei Methoden aufgespalten werden, jede mit einem klareren Kontrakt. Wenn die zweite Aussage wahr sein soll, so hat der Code einen *Bug*, der zum Absturz führt. Sollte die dritte Aussage die Absicht gewesen sein, dann führt die null-Abfrage den Leser in die Irre; der Code sollte umgeschrieben werden und mit einer Abfrage beginnen, die eine *ArgumentNullException* wirft. Egal, welcher der drei Fälle zutrifft – der Code hat ernsthafte Mängel.

Die gezeigten Fehler sind nur drei Beispiele aus Dutzenden Fehlermustern, die immer wieder in echten C#-Codebasen auftauchen. Bei jedem Fehler frage ich mich: „Lässt sich ein Code-Analysator schreiben, der diesen *Bug* automatisch findet?" In einer idealen Welt würden Sprache und Framework den Entwickler von Anfang an davon abhalten, defekte Software zu schreiben – aber Universalsprachen sind nicht perfekt. Zum Glück können wir für viele Fälle Analysen schreiben, die häufig auftretende Fehler erkennen und den Entwickler darauf aufmerksam machen, bevor der Fehler an den Kunden ausgeliefert wird.

Kurzbiografie

Eric Lippert entwickelt C#-Analyseprogramme bei Coverity. Er war 16 Jahre lang bei Microsoft Chefentwickler im C#-Compilerteam und Mitglied der C#-Sprachentwickler. Er hat bereits mehrere Programmierbücher verfasst und twittert unter @ericlippert; seinen Blog über sagenhafte Coding-Abenteuer findet man unter ericlippert.com. Fragen zu Code und Code-Problemen beantwortet er auf StackOverflow.com.

Des Programmierers kleine Vergnügen

Terzett zu zweit

von Michael Wiedeking

Gelegentlich hat man es mit Zahlen zu tun, die nicht ganz dem entsprechen, was man eigentlich braucht. Wenn das an einer Stelle der Fall ist, die man sehr oft durchläuft, stellt sich natürlich die Frage der Effizienz und wie man die Zahlen in das gewünschte Format bekommt.

Im siebten Vergnügen [1] wurde nach einer Lösung für eine Vergleichsfunktion gesucht, um herauszufinden, ob eine Zahl x kleiner, gleich oder größer einer anderen Zahl y ist. Die souveräne Lösung, die wunschgemäß die Werte –1, 0 und 1 annimmt

$$[x > y] - [x < y]$$

verlangte aber, dass es eine hier mit [.] markierte Möglichkeit gibt, die Ergebnisse von Bedingungen in Zahlen umzuwandeln. Die *C*-Programmierer etwa haben keine Probleme damit, denn jede Bedingung wird automatisch – wie im Beispiel angenommen – in die Werte 1 für *true* und 0 für *false* umgewandelt. Und in einem anderen Vergnügen [2] wurde dann gezeigt, wie man diese Werte auch in anderen Sprachen bedingungsfrei bestimmen kann.

Was ist nun, wenn man bereits eine optimale *compare*-Funktion *cmp* hat, die wie oben angenommen bei einem Vergleich –1, 0 und 1 liefert, aber man für die einzelnen Vergleiche die Werte 0 und 1 benötigt?

	$\text{cmp}(x, y)$	$[x < y]$	$[x \le y]$	$[x = y]$	$[x \ne y]$	$[x > y]$	$[x \ge y]$
$x < y$	–1	1	1	0	1	0	0
$x = y$	0	0	1	1	0	0	1
$x > y$	+1	0	0	0	1	1	1

Um etwa $x < y$ in eine Zahl umzuwandeln, muss also eine Funktion f gefunden werden, für die $f(-1)$ zu einer 1 wird und $f(0)$ und $f(1)$ auf 0 abgebildet werden. Was ist denn an der –1 so besonders? Naja, sie hat im Gegensatz zu 0 und 1 ein Vorzeichen. Extrahiert man dieses nun (unter der Annahme, dass man es mit 32-Bit-Wörtern zu tun hat), so erhält man das gewünschte Ergebnis.

$$[x < y] = \text{cmp}(x, y) >>> 31$$

Das ist schön, denn man bekommt das gewünschte Ergebnis in nur einer zusätzlichen Instruktion. Viel höher dürfte dann der Aufwand für $x > y$ auch nicht sein, denn das ist ja $y < x$. Leider ist es nicht immer möglich, den Anwender dazu zu zwingen, doch einfach die Argumente umzudrehen, aber für uns ist das nicht schwieriger, wenngleich uns das eine zusätzliche Instruktion kostet: Wir können nämlich erreichen, dass wir –1 statt 1 geliefert bekommen, indem wir das Vorzeichen wechseln.

$$[x > y] = -\text{cmp}(x, y) >>> 31$$

Der Fall $x \le y$ mutet jetzt schon etwas schwieriger an. Gibt es denn keine Möglichkeit, dass das Ergebnis der *Compare*-Funktion so verschoben werden kann, dass –1 sein Vorzeichen behält, 0 eins bekommt und 1 auf jeden Fall vorzeichenlos bleibt? Ja, die gibt's, wenn man einfach 1 abzieht. Damit erhält man statt –1, 0 und 1 die Werte –2, –1 und 0, was genau den Zweck erfüllt.

$$[x \le y] = (\text{cmp}(x, y) - 1) >>> 31$$

Für den Fall $x \ge y$ müsste es doch wie schon im Fall $x > y$ genügen, einfach das Vorzeichen zu wechseln.

$$[x \ge y] = (-\text{cmp}(x, y) - 1) >>> 31$$

Bleiben nur noch die Fälle für Gleichheit und Ungleichheit. Der Fall für die Ungleichheit scheint etwas einfacher, weil er doch die Fälle –1 und +1 in eine 1 umwandeln muss. Und das macht man – wenn das eine optimale Funktion ist – am einfachsten über die Absolutfunktion.

$$[x \ne y] = \text{abs}(\text{cmp}(x, y))$$

Steht die Funktion aber nicht als Maschinenbefehl zur Verfügung, so scheint die etwa im Vergnügen [3] hergeleitete Funktion *abs* doch etwas zu aufwändig zu sein. Aus der bereits oben erwähnten vergnüglichen Lektüre

von [2] haben wir gelernt, dass für $x \neq 0$ bei $x \mid -x$ das Vorzeichen-Bit gesetzt ist und könnten uns dementsprechend bedienen. Allerdings geht es doch noch etwas einfacher, weil wir nicht allgemein wissen müssen, ob eine Zahl von 0 verschieden ist, sondern ob sie –1 oder 1 ist. Und was ist beiden Zahlen gemeinsam? Im Gegensatz zur 0 ist das niederwertigste Bit gesetzt.

$$[x \neq y] = \text{cmp}(x, y) \,\&\, 1$$

Und das gilt doch sicherlich auch für den gegenteiligen Fall $x = y$. Allerdings reicht hier nicht wie bisher ein Negieren, da – 0 leider immer noch 0 ist. Hier müssen die Bits also invertiert werden.

$$[x = y] = \sim\text{cmp}(x, y) \,\&\, 1$$

Ein Maximum von zwei zusätzlichen Instruktionen ist sicherlich akzeptabel. Deshalb stellt sich die Frage, ob der Fall $x \geq y$ nicht noch verbessert und um eine Instruktion vermindert werden kann. Im Fall $x = y$ mussten wir die Bits invertieren und das können wir auch im Falle $x \geq y$ statt der Negation machen. Dafür gibt es sogar eine formale Begründung, denn es gilt $-v = \sim v + 1$. Setzt man das ein, so erhält man:

$$\begin{aligned}[x \geq y] &= (-\text{cmp}(x, y) - 1) >>> 31 \\ &= (\sim\text{cmp}(x, y) + 1 - 1) >>> 31 \\ &= \sim\text{cmp}(x, y) >>> 31\end{aligned}$$

Ach, es ist doch immer wieder vergnüglich, so schöne Lösungen für kleine Probleme finden zu können.

Referenzen

[1] Wiedeking, Michael
Des Programmierers kleine Vergnügen – Komparativ für Programmierer,
KaffeeKlatsch, Jahrgang 1, Nr. 7, S. 14, Bookware, Juli 2008
http://www.bookware.de/kaffeeklatsch/archiv/KaffeeKlatsch-2008-07.pdf

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

[2] Hahn, Jörn
Des Programmierers kleine Vergnügen – Absolut ohne Sprünge
KaffeeKlatsch, Jahrgang 1, Nr. 5, S. 24, Bookware, Mai 2008
http://www.bookware.de/kaffeeklatsch/archiv/KaffeeKlatsch-2008-05.pdf

Kurzbiographie

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

Kaffeesatz

Unbezahlbar

von Michael Wiedeking

Die Idee ist eigentlich gar nicht so dumm: Ein Gerät auf dem man alle wichtigen Funktionen vereint, so dass man diese immer und überall verfügbar hat. Die Idee ist auch schon recht alt, wenn man etwa an die jetzt fast zwanzig Jahre alte JavaCard-Idee denkt. Was dem Smartphone bei Kommunikation und Spielen gelungen ist, will sich aber nicht auch für die ernsteren Dinge des Lebens etablieren – das Bezahlen.

Warum nur, titelte es neulich in einem Artikel, will die Welt nicht mit dem Handy bezahlen? Das Smartphone hat sich doch bei der Jugend als das Prestige-Gerät schlechthin entwickelt und konnte sich selbst bei den älteren Generationen etablieren. Die neuen Modelle sind auch schon alle mit irgendwelchen Nahfunk-Standards ausgerüstet, so dass ein Auflegen des Geräts an der Kasse zum Bezahlen völlig ausreichend wäre. Aber was an Flughäfen schon gang und gäbe ist, will sich im Laden um die Ecke nicht etablieren.

Ich für meinen Teil sehne mich immer noch nach meinem Nokia 6310i zurück, mit dem man noch wirklich gut telefonieren konnte. Und der Akku hielt bei meinem Telefonverhalten über eine Woche, derweil es mein jetziges Smartphone keine zwei Tage schafft. Ich will nicht undankbar sein, denn eine alternative Zugverbindung herauszusuchen oder mal eben etwas nachzuschlagen ist deutlich einfacher geworden – wenn denn eine Internet-Verbindung besteht.

Aber möchte ich wirklich mit meinem Handy bezahlen? Ehrlich gesagt bin ich ganz froh, dass meine Scheckkarte nicht ans Internet angeschlossen ist und nur dann mit irgendwas kommuniziert, wenn ich das Teil in den entsprechenden Kartenleser stecke. Nicht dass ich wüsste, was sich die Karte mit dem Gerät zu erzählen hat, aber ich weiß, dass es eine verhältnismäßig einseitige Unterhaltung ist, die es dem Gerät praktisch unmöglich macht, meine Karte zu manipulieren.

Bei meinem Smartphone weiß ich ehrlich gesagt nicht, wer da wann auf was zugreift. Auch wenn sich mein App-Haushalt in Grenzen hält, sind doch einige „nützliche“ Apps darauf enthalten, die alle in regelmäßigen Abständen auf meinen Standort, mein Adressbuch und das Internet zugreifen wollen. Und weil diese Zugriffe schon nicht für mich nachvollziehbar sind, möchte ich sicher nicht, dass jene auch auf mein Konto zugreifen können.

Mal ganz davon abgesehen, dass der Bezahlvorgang durch ein in der Regel eindeutig identifizierbares Gerät erfolgen würde. Sicher, dass ist auch bei meiner Scheckkarte der Fall, aber die ist nicht so datenhungrig und merkt sich, wo ich überall eingekauft habe. Und im Moment vertraue ich meiner Bank doch eher als beispielsweise Google und Apple.

Warum aber die Jugend nicht mit dem Smartphone bezahlen will, ist nicht ganz so offensichtlich. Aber vielleicht gibt es einen ganz simplen Grund dafür, dass sie sich nicht für ein Bezahlsystem mit dem Handy interessiert: Sie haben gar kein Geld mehr, das sie ausgeben könnten – das haben Sie schon für das all zu teure Smartphone verprasst.

Lektüre

Architektur- und Entwurfsmuster der Softwaretechnik
Mit lauffähigen Beispielen in Java
von Joachim Goll und Manfred Dausmann
Gebundene Ausgabe, 300 Seiten, deutsch
Springer Vieweg, Juli 2013
ISBN: 3834824313
ISBN-13: 978-3834824318
E-Book ISBN 978-3-8348-2432-5
rezensiert von Ilker Yümsek

Software ist „lebendig", weil das Umfeld, in dem die Software eingesetzt wird auch lebendig ist und damit eine Eigendynamik hat. Aufgrund dessen ist das Entwickeln von Software vor allem eine Kunst. Es ist die Kunst, aus den Gegebenheiten und gestellten Anforderungen eine Lösung so zu realisieren, dass sie auch in der Lage ist, sich den Änderungen in ihrem Umfeld zu stellen und entsprechend (möglichst aufwandsarm) anzupassen. Ein weiterer Aspekt dieser Kunst ist außerdem, die richtige Balance zwischen „idealistischem Vorgehen" und „Kosteneffizienz" zu finden. Durch diese Grundlage bleibt das Thema „Architektur und Entwurfsmuster" in der Software-Entwicklung immer interessant und beitragsreich.

Das Buch „Architektur- und Entwurfsmuster der Softwaretechnik" von Joachim Goll und Manfred Dausmann ist eines der neu erschienenen Bücher auf diesem Gebiet. Das Buch kommt mit einer guten Druckqualität und mit lauffähigen Beispielen in der Programmiersprache Java. Als Zielgruppe werden dabei folgende Leser angesprochen:

- Studierende der Informatik und der ingenieurwissenschaftlichen Disziplinen
- berufliche Umsteiger und Entwickler in der Praxis

Das Buch ist in 5 Kapitel aufgeteilt, in denen folgende Themen behandelt werden:

- Prinzipien für den objektorientierten Entwurf
- Software-Architekturen
- Muster beim Software-Entwurf
- objektorientierte Entwurfsmuster
- Architekturmuster

Im Kapitel 1 werden die objektorientierten Entwurfsprinzipien (*Seperation of Concerns, Single Responsibility etc.*) behandelt.

Im Kapitel 2 wird auf den Begriff „Software-Architektur" eingegangen. Neben den Merkmalen einer Software-Architektur werden dabei auch die Begriffe Referenzarchitekturen, Architektur- und Entwurfsmuster erläutert. Weitere praxisrelevante Themen wie „die anzugehenden Aufgaben bei der Konzeption eines Systems" ergänzen diesen Inhalt.

Im nächsten Kapitel geht es um den Einsatz von Entwurfsmustern. Die Gründe für den Einsatz der Entwurfsmuster sowie die kritischen Aspekte des musterbasierten

Vorgehens sind hauptsächliche Themen des Kapitels. Außerdem wird in diesem Kapitel verständlich beschrieben, aus welchen Bestandteilen sich die Beschreibung eines Entwurfsmusters zusammensetzt. Die Entwurfsmuster werden im Kapitel 4 nach diesem Schema behandelt. In diesem Schema ist besonders die Erwähnung der ähnlichen Muster nützlich. Darüber kann man bei der Suche nach einem geeigneten Muster für ein Problem gezielt weitersuchen, falls ein Entwurfsmuster die Bedürfnisse nicht komplett erfüllt.

Die behandelten Entwurfsmuster sind thematisch kategorisiert. Am Anfang des Kapitels wird ein guter Überblick über die behandelten Entwurfsmuster gegeben und es gibt auch eine für Neulinge speziell zusammengestellte Liste (von Mustern), damit sie einfacher in die Thematik einsteigen können. In allen Musterbeschreibungen etc. wird auf weitere Quellen verwiesen (online sowie Literatur), wo man Detailinfos zum jeweiligen Thema finden kann.

Im Kapitel 5 werden Architekturmuster nach einem ähnlichen Schema wie bei den Entwurfsmustern vorgestellt. Besonders gut gelungen ist dabei die Erläuterung der Zwei- und Drei-Schichten-Architektur und des MVC-Patterns. Außerdem werden bei der Erläuterung des Musters „*Service-Oriented Architecture*“ beide Standards *JAX-WS* (Java API for XML Web Services) und *JAX-RS* (Java API for RESTful Web Services) verständlich beschrieben.

In jedem Kapitel gibt es eine Zusammenfassung und Aufgaben über das behandelte Thema. Die Code-Beispiele für das Buch sind bewusst einfach gehalten und unabhängig von bestimmten Frameworks sowie Werkzeugen (Spring, JUnit usw.). Dadurch sind sie für Einsteiger besser verständlich.

Die Betrachtung der Entwurfsmuster aus Sicht der objektorientierten Prinzipien ist das wesentliche Merkmal dieses Buches. Auch wenn eine dieser Prinzipien (Liskovsches Substitutionsprinzip) zu oft erwähnt wird – und damit das Lesen erschwert – ist der Ansatz richtig. Die behandelten Prinzipien im ersten Kapitel sind für das Verstehen der weiteren Inhalte im Buch essentiell. Deswegen empfehle ich dem Leser sich Zeit für das erste Kapitel zu nehmen und sich mit den Themen und Beispielen darin unbedingt auseinanderzusetzen.

Zusammenfassend ist das Buch empfehlenswert für Studenten und berufliche Umsteiger, die sich mit dem Programmieren schon beschäftigt haben.

User Groups

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

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

Java User Groups

Deutschland

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Schweiz

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

.NET User Groups

Deutschland

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

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

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

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

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

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

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

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

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

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

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

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

Österreich

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

Software Craftsmanship Communities

Deutschland, Schweiz, Österreich

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

MATHEMA

meet the experts of enterprise infrastructure

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

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

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

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

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

Lesen bildet. **Training macht fit.**

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

MATHEMA

„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
Henkestraße 91, 91052 Erlangen | Telefax: 09131 / 89 03-55 | E-Mail: info@mathema.de | Ansprechpartnerin: Britta Kolb

.NET DAY FRANKEN

Die fränkische Community-Konferenz

10. Mai 2014

Seit 2010 findet der .NET Day Franken, die Community-Konferenz Frankens, in Nürnberg statt. Er bietet alles, was das Entwicklerherz höher schlagen lässt.

Am 10. Mai 2014 dreht sich wieder einen Tag lang alles um moderne und professionelle Software-Entwicklung. Neben den angebotenen Vorträgen stehen vor allem die Diskussion und das Networking im Mittelpunkt. Hierfür bietet das NH Nürnberg-City wieder eine angenehme und zentrale Location.

Auch zum Jubiläum in diesem Jahr konnten wir wieder erfahrene Redner mit interessanten Schwerpunkten gewinnen. Wie in den vergangenen Jahren werden neben den aktuellen Themen in der Software-Entwicklung auch innovative Verfahren und Anwendungen vertreten sein. Zu den technischen Inhalten gehörten unter anderem Cloud Computing mit Windows Azure, App-Entwicklung für das Windows Phone 8, HTML5, CSS 3 und vieles mehr. Aber auch die technologieübergreifenden Themen kommen mit Vorträgen zu Kanban und CQRS nicht zu kurz.

Damit dürfte die Veranstaltung nicht nur für Entwickler, sondern auch für Produktmanager und Verantwortliche aus Software-Unternehmen interessant sein.

Weitere Informationen, das komplette Programm und das Anmeldeformular finden Sie unter www.dotnet-day-franken.de. Sichern Sie sich noch schnell Ihre Teilnahme, bevor die Veranstaltung ausgebucht ist.

Unter anderem:

Beyond Mobile Device Management Lars Keller & Frank Solinske

System und Umwelt Ilker Cetinkaya & Dennis Wagner

SQL Server Data Tools – die IDE für Datenbankentwickler Constantin Klein

Kanban – Agile 2.0? Thomas Schissler

Xamarin 2.0 mit C# Timur Zanagar

www.dotnet-day-franken.de Community Conference

Das Allerletzte

```
Private Sub Befehl99_DblClick(Cancel As Integer)
Dim Passwort As String
  Passwort = Input Box("Passwort eingeben:", "Passwortabfrage")
  If Passwort = "depp" Then DoCmd.OpenForm "Admin"
End Sub
```

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

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

Der nächste KaffeeKlatsch erscheint im April.

Herbstcampus

Wissenstransfer par excellence

1. – 4. September 2014

in Nürnberg