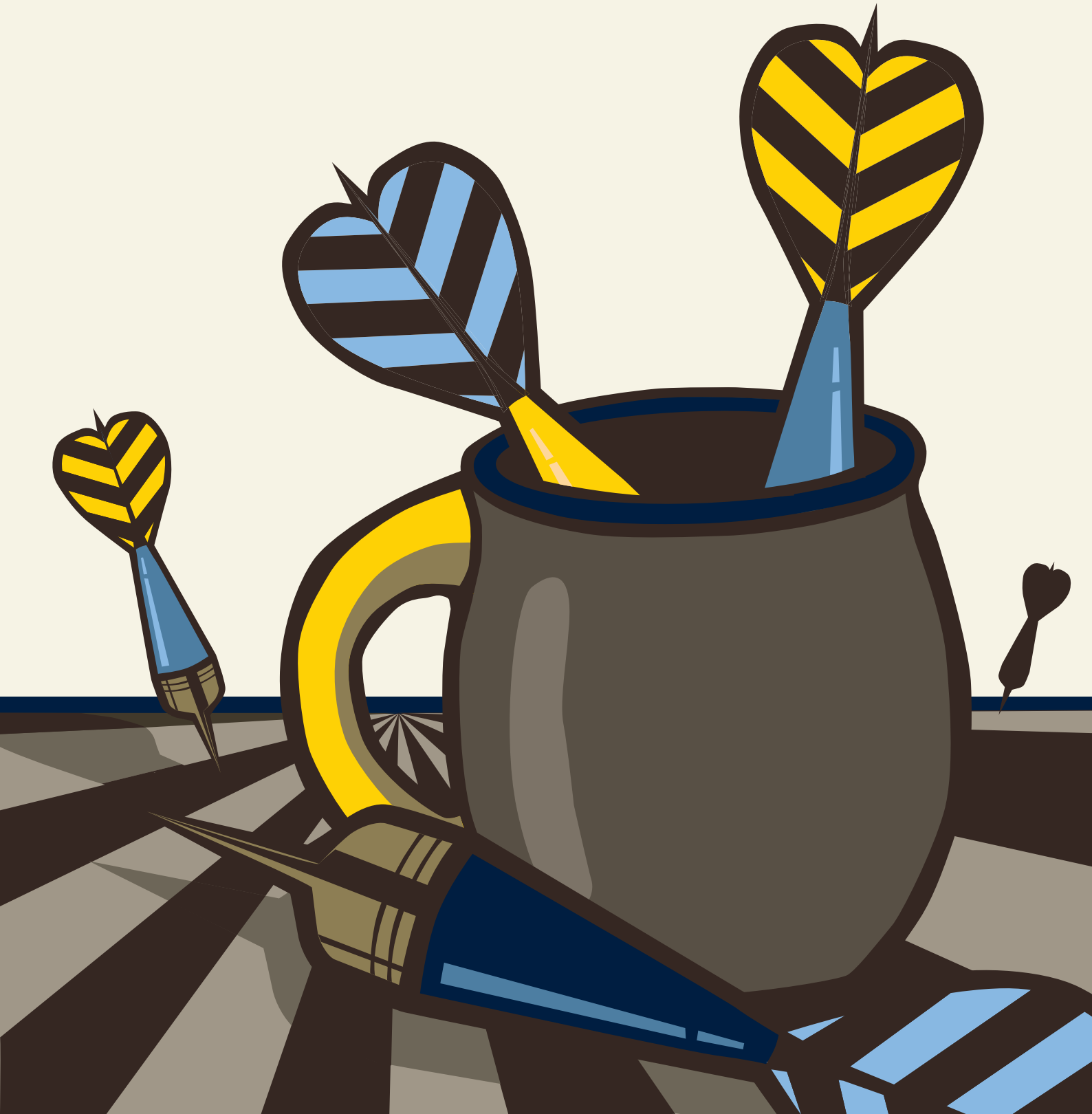
KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

ISSN 1865-682X

02/2017

Jahrgang 9



KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

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

abo@bookware.de

abonnieren.

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

Editorial

In eigener Sache

Ende des vergangenen Jahres haben wir unter unseren Abonnenten eine Umfrage durchgeführt, um zu erfahren wie der KaffeeKlatsch von Ihnen, liebe Leser, wahrgenommen wird. Hier fassen wir einige Ergebnisse zusammen, die Sie sicherlich auch interessieren dürften.

Teilgenommen haben 106 Personen, wovon 74 die Umfrage abgeschlossen haben. Die Befragten stufen die generelle Aufmachung (84 %), aber auch im Einzelnen die Titelseiten (88 %) sowie die Artikelseiten (74 %) als gut bis sehr gut ein. Besonders erfreulich ist für uns die inhaltliche Bewertung. Ein Großteil findet die Artikel anspruchvoll (69 %) und leserfreundlich (78 %). Auch die Länge der Artikel sehen 77 % der Befragten als genau richtig an. Dass es dennoch Verbesserungsvorschläge in Bezug auf Themenvielfalt, Layout und Format gab, werden wir in künftigen Ausgaben berücksichtigen.

Obwohl der KaffeeKlatsch insgesamt bereits von 84 % der Befragten mit gut bzw. sehr gut bewertet, ist das für uns der Ansporn, ihn weiter auf diesem Niveau zu halten und punktuell zu optimieren.

Kommen wir aber nun zu den Inhalten dieser Ausgabe. Dieses Mal geht es unter anderem um Excel und objektorientierte Entwicklung. Hristiyan Pehlivanov zeigt in „*Excel – die populärste funktionale Sprache der Welt*“ (Seite 9), inwiefern Excel und funktionale Programmierung zusammenhängen und geht dabei auf Konzepte ein, die auf die objektorientierte Entwicklung übertragbar sind. Wie Sie „*Dart spielen*“ (Seite 6) können, erklärt Ihnen Thomas Künneth. In seinem Beitrag gibt er einen kurzen Ein- und Überblick über die klassenbasierte, vollständig objektorientierte Allzweckprogrammiersprache Dart. Florian Hurlbrink beschreibt in „*Testdatengenerierung mit Java-faker*“ (Seite 12), wie sich mit dem Gem „*faker*“ Testdaten erzeugen lassen und welche Besonderheiten es dabei gibt.

Zum Abschluss haben wir in der Kolumne „*Das Allerletzte*“ wieder etwas zum Schmunzeln für Sie.

Ich wünsche Ihnen viel Spaß mit unserer Lektüre.

Ihr Oliver Klosa
Chefredakteur

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 Herrn Oliver Klosa via E-Mail an anzeigen@bookware.de oder telefonisch unter 09131/8903-0 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 Oliver Klosa via E-Mail unter redaktion@bookware.de oder telefonisch unter 09131/8903-0.

Copyright

Impressum

KaffeeKlatsch Jahrgang 10, Nummer 2, Februar 2017
ISSN 1865-682X
Bookware – eine Initiative der
MATHEMA Software GmbH

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

Herausgeber/Redakteur: Dr. Oliver Klosa
Anzeigen: Dr. Oliver Klosa
Grafik: Nicole Delong-Buchanan

Inhalt

Artikel

Dart spielen

von Thomas Künneth

Dart ist eine klassenbasierte, vollständig objektorientierte Allzweckprogrammiersprache mit Einfachvererbung und Mixins. Die Entwicklung begann 2010 in einem kleinen Team rund um Lars Bak, Gilad Bracha und Kasper Lund bei Google in Aarhus. Dort wurde die Sprache auch etwa ein Jahr später im Rahmen der GOTO erstmals öffentlich präsentiert. Ende 2013 lag die stabile Version 1.0 vor. Seit 2014 ist Dart ECMA-standardisiert[1].

Excel – die populärste funktionale Sprache der Welt

von Hristiyan Pehlivanov

Software-Entwickler machen sich gerne über Excel lustig. Falls aber das Tool so schlecht ist, warum ist es so erfolgreich? Wenn ganze Unternehmen ihr Geschäft mit Excel organisieren können, dann muss diese Software auch ein paar Tricks drauf haben. Der Autor ist der Meinung, dass Excel ein sehr gutes Beispiel für funktionale Programmierung ist und dass auch Entwickler etwas daraus lernen können. Dieser Artikel stellt Konzepte aus Excel vor, die direkt auf die objektorientierte Entwicklung übertragbar sind.

Testdatengenerierung mit Java-faker

von Florian Hurlbrink

Wer mit Ruby on Rails entwickelt, kennt das Gem „*faker*“. Mit diesem Gem lassen sich verschiedene Testdaten erzeugen wie Vor- und Zunamen, Post-, E-Mail-, IP-Adressen, Datumsangaben und vieles mehr. Diese Daten kann man abgestimmt auf ein Land bzw. Sprache erstellen.

Dart spielen

Ein neugieriger Blick auf die Programmiersprache Dart

von Thomas Künneth

Dart ist eine klassenbasierte, vollständig objektorientierte Allzweck-programmiersprache mit Einfachvererbung und Mixins. Die Entwicklung begann 2010 in einem kleinen Team rund um Lars Bak, Gilad Bracha und Kasper Lund bei Google in Aarhus. Dort wurde die Sprache auch etwa ein Jahr später im Rahmen der GOTO erstmals öffentlich präsentiert. Ende 2013 lag die stabile Version 1.0 vor. Seit 2014 ist Dart ECMA-standardisiert[1].

Ursprünglich wollte Google die Sprache als gleichberechtigte Alternative zu JavaScript im Browser etablieren. Die hierzu erforderliche Laufzeitumgebung Dart VM ist auch heute noch in der Dartium genannten Variante von Chrome enthalten. Nach einem im Frühjahr 2015 bekannt gegebenen [2] Strategiewechsel steht aber die Übersetzung in JavaScript – und damit die enge Annäherung an ein riesiges Ökosystem – im Zentrum. Damit tritt Dart in Konkurrenz zu anderen transpilierenden Sprachen wie TypeScript und CoffeeScript.

Die Sprache kann mit dem Online-Editor DartPad [3] ohne Tool-Installationen ausprobiert werden. Wer tiefer einsteigen möchte, sollte das Dart SDK [4] herunterladen. Eine Reihe von Editoren kennen die Dart-Syntax und -Formatierungsrichtlinien oder bieten entsprechende Plug-ins an. Dies gilt auch für die unter Java-Entwicklern sehr beliebte Entwicklungsumgebung IntelliJ IDEA von JetBrains. Dart orientiert sich in vielerlei Hinsicht an C-artigen Sprachen. C-/C++-, C#- und Java-Entwicklern wird deshalb vieles bekannt und vertraut vorkommen. Dennoch ist Dart kein alter Wein im neuen Schlauch. Die Sprache geht viele Herausforderungen pragmatischer an als die genannten Vorbilder. Deshalb ist Dart-Quelltext sehr schlank und doch intuitiv verständlich.

Nichts als Objekte

Die Sprache ist klassenbasiert und vollständig objektorientiert. Es wird also nicht zwischen primitiven und Referenztypen unterschieden. Vielmehr ist alles in Dart zur Laufzeit ein Objekt und jedes Objekt ist die Instanz einer Klasse. Die Wurzel der Klassenhierarchie ist *Object*. Von alldem ist im Quelltext aber nicht unbedingt etwas zu sehen. Dart möchte dem Entwickler viele Freiheiten lassen und ihm möglichst selten im Weg stehen. Deshalb sind eine Reihe von Konstrukten (beispielsweise die explizite Deklaration einer Klasse) optional. Auch das Zusammenfassen mehrerer Funktionen, Klassen und Variablendeklarationen zu einer Quelltextdatei (die übrigens auf *.dart* endet) wird akzeptiert. Natürlich liegt es nicht im Interesse der Dart-Entwickler, das Entstehen von riesigen, nicht wartbaren Code-Monstern zu fördern. Die Sprache enthält deshalb effiziente Mechanismen, um Quelltext zu modularisieren. Dart-Programme bestehen aus Bibliotheken (*Libraries*). Diese wiederum können in Teile (*parts*) zerlegt werden. Der von einer Bibliothek exportierte, also von außen sichtbare Namensraum, lässt sich beliebig steuern. Bibliotheken werden üblicherweise zu Paketen geschnürt. Dart beinhaltet den Dependency- und Asset-Manager *pub*. Er sorgt gegebenenfalls für das Herunterladen von Packages. Das folgende Listing

zeigt eine sehr einfache Bibliothek – ein ausführbares Programm. Solche Libraries werden *script* genannt.

```
main() {
  print("Hallo, Welt");
}
```

Listing: hallo.dart

Neben DartPad können Sie auch das Dart SDK verwenden, um das Beispiel auszuprobieren:

```
...\dart-sdk\bin\dart.exe hallo.dart
```

Vor der Ausführung ist keine Übersetzung nötig. Die Laufzeitumgebung Dart VM arbeitet direkt auf dem Quellcode. Um dem Programm über die Kommandozeile Argumente übergeben zu können, wird die leere Parameterliste von main() um eine typisierte Liste erweitert:

```
main(List<String> args) {
  var i = 0;
  for (var arg in args) {
    print("${i += 1}: $arg");
  }
}
```

Listing: argumente.dart

Dart ist optional typisiert. Deshalb wäre auch *main*(*var args*) und sogar *main*(*args*) erlaubt. Die Definition der Variable *i* wie im Listing zu sehen sagt nicht explizit, welcher Datentyp erwartet wird. Dies ergibt sich implizit aus der Zuweisung eines Werts. Dieser legt fest, welche Operationen mit ihm möglich sind. Fans von Typangaben können aber auch schreiben: *int i = 1*; Und schon fühlen sich Java-oder C#-Entwickler wieder wohl. Dart nennt die Angabe eines Typs übrigens *Typannotation*. Mit diesem Begriff soll ausgedrückt werden, dass der Typ keine Auswirkungen auf die Semantik des Programms zur Laufzeit hat. *var i = 10*; und *int i = 10*; haben also dieselbe Bedeutung. Bloße Zierde sind *Typannotationen* freilich nicht. Sie helfen Compiler und Tools beim Aufdecken von Inkonsistenzen und potenziellen Fehlern. Wem dies nicht ausreicht, kann seit einiger Zeit den sogenannten *Strong Mode* aktivieren. Mit ihm wird Dart fast zu einer stark typisierten Sprache.

Datentypen

Praktisch ist die automatische Auswertung von Ausdrücken innerhalb von Zeichenketten (*string interpolation*). Einfachen Variablen (*Bezeichnern*) stellt man ein Dollarzeichen voran, komplexere Ausdrücke (wie zum Beispiel Methodenaufrufe) werden zusätzlich in geschweifte Klammern gesetzt.

Verglichen mit anderen Sprachen beinhaltet Dart weniger eingebaute Datentypen. Im Folgenden werden ein paar davon kurz vorgestellt. Zahlen sind entweder *int* oder *double*. Die gemeinsame Oberklasse *num* ist abstrakt, kann also nicht direkt instanziiert werden. Ganze Zahlen können in Dart eigentlich beliebig groß oder klein werden. Soll das Programm allerdings in JavaScript übersetzt werden, dürfen die Werte derzeit -2^{53} nicht unter- und 2^{53} nicht überschreiten. Dies ist der Darstellung von Zahlen in JavaScript geschuldet. Das folgende Programm demonstriert das daraus entstehende Problem:

```
import "dart:math";

main(List<String> args) {
  var i = pow(2, 53);
  print(i);
  print(i + 1);
  i *= -1;
  print(i);
  print(i - 1);
}
```

Listing: integerdemo.dart

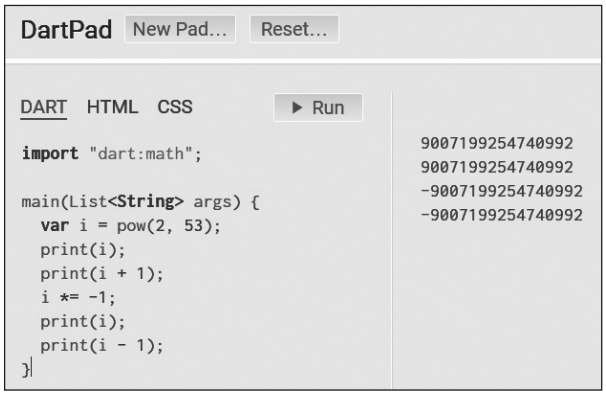


Abbildung 1

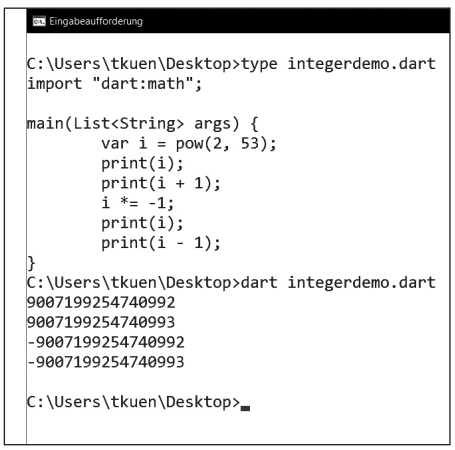


Abbildung 2

Bei der Ausführung durch die Dart VM werden korrekte Werte ausgegeben. Innerhalb von DartPad erscheint allerdings fälschlicherweise viermal dieselbe Zahl (natürlich abgesehen von den Vorzeichen). *Doubles* sind wie in JavaScript 64-bit Fließkommazahlen. Sie folgen dem IEEE-754 Standard. Wahrheitswerte werden mit bool-Objekten abgebildet:

```
...
var b1 = true;
bool b2 = false;
print("${b1 == b2}");
...
```

Listing: Wahrheitswerte

Es gibt immer genau zwei Instanzen dieses Typs. Diese entsprechen den Literalen *true* und *false*. Anders als beispielsweise in JavaScript gibt es in Dart deshalb keinen Zweifel darüber, welche Werte oder Ausdrücke *true* entsprechen. Nur die *bool*-Instanz *true* ist wahr.
An Collection-artigen Datentypen kennt Dart unter anderem Listen und Maps. Erstere entsprechen als angeordnete Mengen den aus anderen Programmiersprachen bekannten Arrays:

```
...
var list = [2, 5, 3];
for (var i = 0; i < list.length; i++) {
   print(list[i]);
}
...
```

Listing: Listen

Ist Ihnen aufgefallen, dass die Elemente der Liste *list* zwischen eckigen Klammern angegeben werden? Die in vielen anderen Sprachen gebräuchlichen geschweiften Klammern werden für Maps verwendet. Dazu gleich mehr. Wie in neueren Java-Versionen sind komplexe Operationen auf Listen möglich. Das folgende Codefragment verdoppelt den Wert jedes Listenelements.

```
...
print(list.map((i) => i * 2).toList());
...
```

Listing: Zahlen verdoppeln

Das folgende Beispiel erzeugt eine Map mit zwei Einträgen. Diese haben die Schlüssel 001 und 002. Deren Werte sind wiederum Maps, die die Schlüssel *isbn* und *titel* enthalten. Das Iterieren über die Schlüssel geschieht mit *for* (*var key* in *books.keys*) { ... }.

```
var books = {
  "001" : {
    "isbn": "978-3-8362-4200-4",
    "titel" : "Android 7"
  },
  "002" : {
    "isbn": "978-3-8362-2958-6",
    "titel" : "Einstieg in Eclipse"
  }
};
```

Listing: Eine Map definieren

Fazit

Vieles in Dart ist für Kenner C-artiger Sprachen vertraut. Dadurch fällt der Einstieg sehr leicht. Diese Tatsache mag aber dazu verleiten, alte Gewohnheiten ungefragt zu übernehmen. Was dann entsteht, ist zwar sauberer Code, lässt das Potenziell der Sprache aber ungenutzt. Deshalb sei an dieser Stelle angeraten, tiefer in die Philosophie von Dart einzutauchen. Einen guten Einstieg bietet eine Serie von Kurzclips, die auf YouTube zur Verfügung steht. [5] Noch mehr Futter bietet das Buch *The Dart Programming Language* von Gilad Bracha. [6]

Referenzen

[1] ecma international *Standard ECMA-408*
https://www.ecma-international.org/publications/standards/Ecma-408.htm

[2] Dart *News & Updates*
http://news.dartlang.org/2015/03/dart-for-entire-web.html

[3] Dart *Dartpad*
https://dartpad.dartlang.org

[4] Dart *Tools*
https://www.dartlang.org/tools/sdk

[5] YouTube *dart tips*
https://www.youtube.com/playlist?list=PLq7yml7rGSYbz7EHWjWjWCSnd92-Pa43K

[6] Gilad Bracha *The Dart Programming Language*
Addison Wesley; 1. Auflage 201; ISBN-13: 978-0321927705

Kurzbiografie

Thomas Künneth ist als Senior Consultant bei MATHEMA Software GmbH tätig. Seit fast 20 Jahren beschäftigt er sich intensiv mit Java. Er ist Autor der Bücher *Android 7*, *Einstieg in Eclipse* und *Java für Windows*, sowie zahlreicher Fachartikel.

Excel – die populärste funktionale Sprache der Welt

Was ein OO-Entwickler aus Excel lernen kann

von Hristiyan Pehlivanov

Software-Entwickler machen sich gerne über Excel lustig. Falls aber das Tool so schlecht ist, warum ist es so erfolgreich? Wenn ganze Unternehmen ihr Geschäft mit Excel organisieren können, dann muss diese Software auch ein paar Tricks drauf haben. Der Autor ist der Meinung, dass Excel ein sehr gutes Beispiel für funktionale Programmierung ist und dass auch Entwickler etwas daraus lernen können. Dieser Artikel stellt Konzepte aus Excel vor, die direkt auf die objektorientierte Entwicklung übertragbar sind.

Der ewige Feind

Unter vielen Entwicklern herrscht eine persönliche Abneigung gegen Excel. Beispielsweise hat ein Kollege neulich einen Zufallsgenerator mit Excel implementiert, der Übungsteilnehmer paarweise gruppiert hat. Dann kam die Anmerkung „*Du bist sehr gut mit Excel. Man merkt, dass du keinen Code mehr schreibst*.“, gefolgt von Lachen. Der lustige Kollege hat aber übersehen, dass dieser Zufallsgenerator nur ein paar Funktionsaufrufe erfordert hat. Wie viele Zeilen Code hätte die Java-Implementierung gehabt?

Der schlechte Ruf von Excel ist eher darauf zurückzuführen, dass das Tool in der Praxis häufig missbraucht wird. Datenbanken, Prozessmanagement-Tools und alles Mögliche wird als Spreadsheet umgesetzt. An dieser Stelle muss aber die IT-Branche auch einen Teil der Verantwortung dafür übernehmen. In der Tat halten die meisten Software-Projekte ihre geplanten Features, Budget und/oder Zeit nicht ein. Dann bleibt Excel eine sehr kosteneffiziente Lösung für viele Probleme, auch wenn diese Lösung nicht besonders robust ist. Es ist sogar faszinierend, was alles mit Excel in der Praxis überhaupt funktionieren kann.

Falls man die Beispiele für Excel-Missbrauch zur Seite legt, hat das Tool manche hervorragende Konzepte, die wir gleich näher betrachten werden. Zuerst aber die Frage: Stimmt der Titel dieses Artikels?

Excel ist eine Programmiersprache?

Excel ist nicht dafür gedacht, Software damit zu entwickeln. Nichtsdestotrotz kann es Daten manipulieren und *If-Else*-Entscheidungen treffen. Deswegen ist es möglich, damit eine Turing-Maschine zu bauen. Genau das haben Felienne Hermans und ihre Freunde demonstriert [1]. Wenn man mit Excel eine Turing-Maschine bauen kann, dann lässt sich auch jeder Algorithmus implementieren.

Was machen wir eigentlich als Software-Entwickler? Meistens bekommen wir Daten, verarbeiten sie mit Hilfe von Geschäftsregeln und liefern ein Ergebnis zurück. In Excel passiert genau dasselbe, aber nur mit primitiven Datentypen anstatt mit komplexen Objekte.

Wären Spreadsheets eine Programmiersprache, ließe sich leichter begründen, warum Excel eine funktionale Sprache ist. Spreadsheets bestehen aus Formeln, die Eingabewerte bekommen und ein Ergebnis berechnen. Sie

können allerdings andere Zellen nicht überschreiben. Aus diesem Grund kann man sie als Funktionen ohne Seiteneffekte bzw. Excel als reinfunktionale Programmiersprache betrachten.

Wie kann es aber sein, dass so viele Leute mit Excel arbeiten, wenn Software-Entwicklung als kompliziert gilt? Irgendwie schafft es Excel, eine Universallösung zu sein, während es einfach bleibt. Aus Sicht des Autors besteht ein wichtiger Grund dafür in den funktionalen Konzepten, die Excel verwendet. Sie helfen dem Benutzer, viele Fallen aus der Software-Entwicklung zu vermeiden.

Data In, Data Out

Die Excel-Formeln sind Funktionen, die Eingabewerte bekommen und ein Ergebnis liefern. Was sie aber auf keinen Fall machen, ist die Außenwelt zu verändern. Sie können weder ihre Eingabeparameter noch das Ergebnis einer anderen Formel überschreiben. Sie sind nur dafür zuständig, mit den eingegebenen Parametern einen neuen Wert zu berechnen. In der funktionalen Programmierung nennt man dieses Prinzip Wirkungsfreiheit oder „*Data In, Data Out*".

Ein Vorteil von „*Data In, Data Out*" ist, dass sich solche Funktionen relativ leicht testen lassen. Sie sind nur von ihren Eingabeparametern abhängig. Da sie auf die Außenwelt nicht zugreifen, muss man keinen externen Zustand des Systems vorbereiten und danach überprüfen. Es reicht vollkommen aus, wenn das richtige Ergebnis berechnet wird. Das macht es auch viel wahrscheinlicher, dass das komplette System funktioniert, wenn die Unit-Tests grün sind. Bei der imperativen Programmierung mit variablem Zustand, sollte man alle möglichen Ausführungspfade mit Tests abdecken, um ein ähnliches Sicherheitsniveau zu erreichen. Sonst könnte eine bestimmte Sequenz von Funktionsaufrufen zu einem Bug führen.

Außerdem sind die „*Data In, Data Out*"-Funktionen meistens leichter zu verstehen. Damit ist hier gemeint, dass bei reinfunktionalen Sprachen die Reihenfolge der Funktionsaufrufe keine Rolle spielt. Das Ergebnis einer Funktion hängt nicht von einem externen Zustand ab, der unter bestimmten Umständen von einer anderen Funktion falsch überschrieben wird. Ein falsches Ergebnis ist mit den gegebenen Eingabeparametern immer falsch, was Fehler immer reproduzierbar macht. Deshalb kann man sich bei der Analyse nur auf die Eingaben und die betroffene Funktion konzentrieren. Und wenn man schon wirkungsfreie Funktionen hat, kann man gleich weiter gehen und ohne Zustand arbeiten.

Immutability

Der Wert einer Excel-Zelle wird einmal berechnet und danach nicht mehr verändert. Dass Zuweisungen und veränderlicher Zustand gefährlich sind, klingt seltsam. Genau das lernt man als erstes bei vielen Programmiersprachen. Stellen Sie sich aber vor, dass Sie sich keine Sorgen machen müssen, wer und wann eine Variable überschreibt. In der Praxis findet man häufig Felder von Klassen, die an mehreren Stellen überschrieben werden. Das wird zu einem Problem, wenn die Klasse relativ komplex wird und der Entwickler Schwierigkeiten hat, sich alle Zugriffe zu merken. Dann wird jede Änderung zu einer mühsamen Analyse, was man mit der Variable machen darf und was nicht.

Wachsame Leser könnten gleich anmerken, dass eine so komplexe Klasse die SOLID-, Clean Code- etc. Prinzipien verletzt. Sie werden meistens Recht haben. Man kann aber die Frage auch anders stellen. Braucht man unbedingt diese Felder? Oder sind sie nur dafür da, um sich Zwischenergebnisse einfacher zu merken? Falls die Klasse ein Ergebnis berechnen und liefern soll, aber dieses Ergebnis für den nächsten Client-Aufruf nicht benötigt wird, kann diese Klasse auf den Zustand komplett verzichten. Anstatt dass Methoden über lokale Felder Ergebnisse austauschen, kann man den Rückgabewert einer Methode gleich als Eingabe für die nächste Methode verwenden.

Spätestens wenn man das selbst ausprobiert, kommt aber der Realitätscheck. Leider bieten objektorientierte Programmiersprachen wie Java sehr schlechte Unterstützung für Immutability an. Wegen des hohen Aufwands empfiehlt es sich auch nicht, alle Java-Klassen im System als Immutable-Klassen zu implementieren. Lohnt es sich dann, diese Konzepte in der Praxis zu verwenden?

Der Mönch, der seine Objekte verkaufte

Wenn funktionale Programmiersprachen so viele Vorteile hätten, warum programmieren wir alle nicht nur funktional? Wie John Hughs meint, klingt das Ganze wie ein mittelalterlicher Mönch, der auf alle Vergnügen im Leben verzichtet, um tugendhaft zu sein [2]. Er nimmt diverse Unannehmlichkeiten in der Gegenwart in Kauf, um in der Zukunft den größten Preis zu bekommen.

Funktionale Programmierung ist keine Universallösung. Sie stellt ein weiteres Werkzeug dar und man soll das passende Werkzeug für den jeweiligen Auftrag nehmen.

Der pragmatische Entwickler

Die objektorientierte sowie die funktionale Programmierung haben ihre Vor- und Nachteile. Nichts spricht aber

dagegen, dass man sie kombiniert. Man kann weiter objektorientiert entwickeln und die funktionalen Konzepte nur bei bestimmten Problemen einsetzen. Wenn man diese Probleme entdeckt hat, muss man die Prinzipien nicht um jeden Preis einhalten. Stattdessen empfiehlt es sich, dass man nach wirkungsfreien Funktionen strebt und dabei das Aufwand-Nutzen-Verhältnis im Auge behält. Es kommt auf den konkreten Fall an.

Zum Beispiel eignen sich grafische Oberflächen oder Datenbankinteraktionen nicht für funktionale Programmierung. Das Ziel dieser Komponenten ist, dass sie Zustand behalten. Im Gegensatz dazu wäre die Business-Logik des Systems ein besserer Startpunkt. Fast jedes Softwaresystem hat eine Kernkomponente, die das Geschäftsmodel enthält. Bei dieser Komponente kann man die Vorteile der funktionalen Programmierung bestens ausnutzen. „*Data In, Data Out*"-Funktionen lassen sich einfacher und sicherer mit Unit-Tests abdecken und Änderungen im Code haben viel weniger Risiko von Nebeneffekten. Außerdem wird diese komplizierte Komponente des Systems verständlicher, weil man die Reihenfolge der Aufrufe ignorieren kann. Im Endeffekt kommt bei bestimmten Eingabedaten immer das gleiche Ergebnis zurück.

Performance

Die Performance ist das Problem, das vielleicht am häufigsten angesprochen wird, wenn man über funktionale Programmierung diskutiert. Es stimmt zwar, dass die Anzahl der Funktionsaufrufe und der erzeugten Objekte zunimmt. Erfahrungsgemäß leidet aber die Performance meistens an Aufrufen von externen Systemen sowie an großen Datenbankoperationen. Der Fall, dass die Datenmengen zu groß sind oder der Algorithmus zu aufwändig ist, findet man in der Applikationsentwicklung eher selten. Nichtsdestotrotz sind frühzeitige Performance-Optimierungen in solchen Fällen auch nicht immer bedingungslos anzuwenden. Überlegen Sie sich, wann Sie zuletzt ein Performance-Problem mit einem Algorithmus gehabt haben?

Parallele Programmierung

Einer der größten Vorteile der funktionalen Programmierung ist bei der parallelen Programmierung zu sehen. Der besteht darin, dass wirkungsfreie Funktionen implizit parallel ausführbar sind, ohne dass der Entwickler sie synchronisieren muss. Da sie keinen gemeinsamen Zustand haben, spielt es keine Rolle, wie viele Threads parallel eine Funktion aufrufen. Außerdem kann der Compiler die Aufrufe selbst optimieren.

Mit der zunehmenden Popularität von Cloud Computing, Computerclustern und CPUs mit mehreren Kernen wird die parallele Programmierung in Zukunft immer mehr an Bedeutung gewinnen. Stellen Sie sich vor, dass Sie keinen Mehraufwand für Synchronisation hätten, aber Ihre Software implizit parallel ausführbar wäre und beliebig skalieren könnte. Ein Traum!

Fazit

Die funktionale Programmierung wird sehr oft zugunsten von der objektorientierten Programmierung vernachlässigt. Es mag zwar ein gutes Werkzeug sein, es kann aber nicht zu jedem Auftrag passen. Es gibt Probleme, die besser funktional lösbar sind. Der Erfolg von Excel in bestimmten Bereichen ist ein Beispiel dafür. „Data In, Data Out"-Funktionen und Immutable-Werte sind bei komplexer Verarbeitung von Daten sehr gut anwendbar und können viele potenzielle Fehlerquellen umgehen. Dass die Software damit auch threadsicher ist, wird mit der Zeit immer wichtiger sein.

,

Referenzen

[1] Felienne Hermans Blog, http://www.felienne.com/archives/2974

[2] Hughes, John, *Research Topics in Functional Programming ed. D. Turner*, Addison-Wesley, 1990, pp 17–42. , https://www.cs.kent.ac.uk/people/staff/dat/miranda/whyfp90.pdf

Hristiyan Pehlivanov ist als Entwickler und Consultant bei der MATHEMA Software GmbH im Java Umfeld tätig. Er ist ein pragmatischer Entwickler, der immer auf der Suche nach neuen Ideen ist. Neben Java interessiert er sich für alle Technologien und Prozesse, die die Software-Entwicklung besser machen.

Testdatengenerierung mit Java-faker

Java-faker generiert sinnvolle Fantasiedaten für den Softwaretest.

von Florian Hurlbrink

Wer mit Ruby on Rails entwickelt, kennt das Gem „*faker*“. Mit diesem Gem lassen sich verschiedene Testdaten erzeugen wie Vor- und Zunamen, Post-, E-Mail-, IP-Adressen, Datumsangaben und vieles mehr. Diese Daten kann man abgestimmt auf ein Land bzw. Sprache erstellen.

In einer Testumgebung machen diese Daten durchaus Sinn, um nicht auf reale Daten zugreifen bzw. um nicht eigene Daten entwerfen zu müssen. So wird unnötiger Arbeitsaufwand eingespart und es wird verhindert, dass diese Daten eine wenig angenehme generische Form wie „*Erika Mustermann, Musterstraße 99, 99999 Musterstadt*“ haben. Stattdessen werden die Daten in einer angebrachten Struktur dargestellt.

Die erzeugten Daten können dazu benutzt werden, eine Testdatenbank zu befüllen. Mit Java-*faker* existiert eine Variante dieses Gems auch für Java-Entwickler. Java-*faker* kann mit Hilfe von Maven eingebunden werden:

```
<dependency>
   <groupId>com.github.javafaker</groupId>
   <artifactId>javafaker</artifactId>
   <version>0.12</version>
</dependency>
```

Alternativ ist auch eine Integration mit Gradle möglich:

```
repositories {
   mavenCentral()
}
dependencies {
   testCompile group: 'com.github.javafaker', name:
   'javafaker', version: '0.12'
}
```

Anhand einer kleinen Beispielanwendung wird die Benutzung von Java-*faker* erklärt. Die Anwendung hat eine Klasse für Personen, die wiederum Attribute für Vorname, Nachname sowie Straße mit Hausnummer enthält und die per Setter modifiziert werden können. Außerdem umfasst die Klasse eine *toString()*-Methode:

```
package localhost.canens;

public final class Person {
   private String firstName;
   private String lastName;
   private String streetAddress;

   public void setFirstName(final String firstName) {
      this.firstName = firstName;
   }

   public void setLastName(final String lastName) {
      this.lastName = lastName;
   }

   public void setStreetAddress(final String streetAddress)
{
      this.streetAddress = streetAddress;
   }

   public String toString() {
      final StringBuilder stringBuilder;

      stringBuilder = new StringBuilder();

      stringBuilder.append(this.firstName);
      stringBuilder.append(" ");

      stringBuilder.append(this.lastName);
      stringBuilder.append(": ");

      stringBuilder.append(this.streetAddress);

      return stringBuilder.toString();
   }
}
```

Die Klasse *PersonManager* implementiert eine sehr rudimentäre Datenbank für eine Personenverwaltung. Man kann Personen hinzufügen, sich die Personen anzeigen lassen und die Datenbank zurücksetzen. Da die Datenbank flüchtig ist, sollte für die Beispielanwendung diese sehr minimale Implementierung ausreichen.

```
package localhost.canens;

import java.util.ArrayList;

public final class PersonManager {
   private ArrayList<Person> persons = new ArrayList<>();

   public void addPerson(final Person person) {
      this.persons.add(person);
   }
```

```
    public void reset() {
        persons.clear();
    }

    public void showPersons() {
        persons.forEach(person -> System.out.println(person));
    }
}
```

Die *Main*-Klasse benutzt die Klasse *PersonManager*.

```
package localhost.canens;

import java.util.Locale;

import com.github.javafaker.Faker;

public final class Main {
    // Privater Konstruktor, weil die Klasse nur statisch
    benutzt werden soll.
    private Main() {
    }

    private static final PersonManager
        PERSON_MANAGER = new PersonManager();

    private static void seedTestData() {
        // Neuen Faker erzeugen. Faker soll Testdaten in
        deutscher Sprache
        // erzeugen.
        final Faker faker = new Faker(new Locale("de"));

        Person person;

        // Datenbank wird zurückgesetzt.
        Main.PERSON_MANAGER.reset();

        for (int i = 0; i < 10; i++) {
            person = new Person();

            // Vorname wird per Faker generiert.
            person.setFirstName(faker.name().firstName());

            // Zuname wird per Faker generiert.
            person.setLastName(faker.name().lastName());

            // Straße mit Hausnummer wird per Faker generiert.
            person.setStreetAddress(
                faker.address().streetAddress()
            );

            // Person wird der Datenbank hinzugefügt.
            Main.PERSON_MANAGER.addPerson(person);
        }
    }

    public static void main(final String[] args) {
        // Datenbank wird mit Testdaten befüllt.
        Main.seedTestData();
        // Datenbank wird ausgelesen.
        Main.PERSON_MANAGER.showPersons();
    }
}
```

Die Klasse enthält eine gleichnamige Methode *main()*. Diese ruft zunächst eine Methode *seedTestData()* der Klasse auf, die die Datenbank mit Testdaten befüllt. Danach wird die Methode *showPersons()* der Datenbank aufgerufen, mit der die Daten angezeigt werden. Die Methode *seedTestData()* benutzt den Java-*faker*, um die Testdaten zu generieren. Zunächst wird ein neuer *faker* mit dem Befehl

```
final Faker faker = new Faker(new Locale("de"));
```

erzeugt, wobei ein deutschsprachiges Locale übergeben wird. Es sind auch Dialekte möglich, beispielsweise *de-AT* für den österreichischen Dialekt. Über Befehle wie

```
faker.name().firstName();
```

werden die eigentlichen Testdaten generiert. Mit *name()* wird z. B. angegeben, dass der Generator für die Kategorie „Namen“ ausgewählt werden soll. Mit *firstName()* wird hingegen bestimmt, dass als konkretes Testdatum ein Vorname generiert werden soll. Bei Ausgabe der Anwendung zeigt sich, welche Testdaten generiert wurden:

```
Gino Büngener: Pastorskamp 209
Enya Konig: Saarlauterner Str. 30c
Ina Pflieger: Carl-Leverkus-Str. 39c
Lene Gierisch: Schumannstr. 07
Lara Kozakiewicz: Dhünnstr. 05c
Dana Lohmann: Monheimer Str. 73
Chris Schellenbeck: Franz-Kail-Str. 3
Phil Anggreny: Havelstr. 749
Conner Breitenstein: Damaschkestr. 388
Ron Huls: An den Irlen 746
```

Die Daten werden dabei zufällig erzeugt und unterscheiden sich somit bei jedem Aufruf der Anwendung. Java-*faker* kann Testdaten für mehr als 30 Sprachen erzeugen. Des Weiteren gibt es Generatoren für knapp 30 Kategorien, zum Teil auch zu kuriosen Themengebieten wie Chuck-Norris-Fakten.

Die Webseite *http://dius.github.io/java-faker* bietet ausführliche Informationen.

Kurzbiografie

Florian Hurlbrink ist als Junior Java Entwickler bei der MATHEMA Software GmbH tätig. Er interessiert sich seit mehreren Jahren für die Sprache Java. Außerdem beschäftigt er sich mit Ruby on Rails.

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)

JUG Deutschland e.V.
Java User Group Deutschland e.V.
c/o Stefan Koospal
http://www.java.de (office@java.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 Görlitz
Java User Group Görlitz
http://www.jug-gr.de
kontakt@jug-gr.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: (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 Friedrichshafen
http://www.dotnet-fn.de
Kontakt: Tobias Allweier
(info@dotnet-fn.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)

MATHEMA

meet the experts of enterprise infrastructure

- **Anwendungsentwicklung mit der Java Enterprise Edition**
  27. März 2017, 11. Sept. 2017, 2.150,– € (zzgl. 19 % MwSt.)
- **HTML5, CSS3 und JavaScript**
  26. Juni 2017, 1.650,– € (zzgl. 19 % MwSt.)
- **Einführung in die objektorientierte Programmiersprache Java – Eine praxisnahe Einführung**
  10. Juli 2017, 2.150,– € (zzgl. 19 % MwSt.)
- **Entwicklung mobiler Anwendungen mit iOS**
  17. Juli 2017, 1.250,– € (zzgl. 19 % MwSt.)
- **Scrum Basics**
  950,– € (zzgl. 19 % MwSt.)
- **Scrum im Großen** – Agiles Organisationsdesign nach LeSS, 950,– € (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."

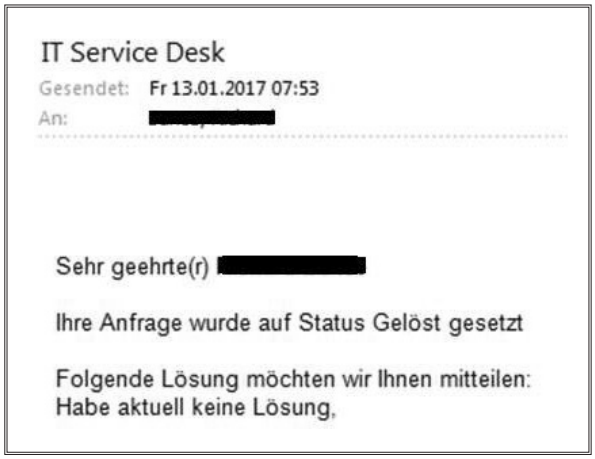
Tim Bourguignon, Senior Consultant

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

Das Allerletzte



IT Service Desk
Gesendet: Fr 13.01.2017 07:53
An: [redacted]

Sehr geehrte(r) [redacted]

Ihre Anfrage wurde auf Status Gelöst gesetzt

Folgende Lösung möchten wir Ihnen mitteilen:
Habe aktuell keine Lösung,

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

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

Der nächste KaffeeKlatsch erscheint im März.

Herbstcampus

Wissenstransfer par excellence

5.– 7. September 2017
in Nürnberg