KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

ISSN 1865-682X

10/2013

Jahrgang 6

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

Abhörsicher – sicher abgehört?

Eigentlich müssten wir Herrn Snowden sehr dankbar sein. Durch seine Enthüllungen hat sich nämlich das Bewusstsein dafür, was ein vertrauliches Datum ist, gravierend gewandelt.

Dass abgehört wird, hat wohl niemand bezweifelt. Bei begründetem Verdacht ist das wohl auch sinnvoll. Damit strebt die gefühlte Wahrscheinlichkeit dafür, dass man selbst ausspioniert wird, gegen Null und steigt eigentlich nur dann proportional zum Unfug, wenn man welchen macht. Irgendwie ist das wie bei einem Lottogewinn – nur umgekehrt: Das ich gewinne ist unwahrscheinlich, aber das irgendeiner gewinnt ist doch recht sicher.

Durch die kleine Indiskretion des Herrn Snowden ist allerdings bekannt geworden, dass die Wahrscheinlichkeit, völlig grundlos ausspioniert zu werden, inzwischen weit über Null liegt. Auch das wäre an und für sich gesehen noch kein Problem, wenn nicht die Möglichkeiten einer maschinellen Auswertung gegeben wären. Und auch das wäre völlig unproblematisch, wenn die auswertende Software nicht Fehler machen würde.

Geht man einmal davon aus, dass etwa 75 % der Deutschen das Internet nutzen [1] und gelegentlich eine E-Mail schreiben. Wenn nur 1 % dieser E-Mails, weil sie etwa über ein fremdes Land geroutet werden, mitgelesen und ausgewertet werden, dann sind das etwa 600 000 E-Mails. Wenn nur jede 1000ste ein verfängliches Wort enthält (wie vielleicht eine ungünstig getrennte Eis-Bombe) und die Auswertung mit 99 %iger Sicherheit sagen kann, ob eine E-Mail bösartig ist, dann steht die Polizei vielleicht immer noch bei 6 Personen überraschend vor der Tür.

Das ist so natürlich nicht korrekt, aber vielleicht vermittelt das ein Gefühl dafür, welche Gefahren ein grundsätzliches Abhören mit sich bringt. Täglich werden in Deutschland über eine Milliarde E-Mails verschickt [2] und wenn auch nur 1 % davon systematisch ausgewertet wird …

Dabei geht es übrigens nicht nur um Straftaten. In einer Firma wird schon seit Jahren erfolgreich ein Produkt verkauft, das auf Technologie Xy basiert. Würde nun bekannt, dass sich diese Firma mit der Technologie Z auseinandersetzt, könnte das fatale Folgen für die Verkaufszahlen haben, wenn dies – zurecht oder auch nicht – als Auslauf der aktuellen Produktlinie interpretiert würde.

Wir als Software-Entwickler sind deshalb angehalten, ungeachtet irgendwelcher gesetzlicher Vorschriften darauf hinzuweisen, dass (fast) alle Daten schützenswert sind und deshalb auch geschützt werden müssen. Und wir müssen nicht nur sagen wie das geht, sondern es auch nach bestem Wissen und Gewissen umsetzen. Deshalb sollten wir uns auch ständig zumindest soweit nebenbei damit auseinandersetzen, damit wir es auch ja nicht verpassen, wenn ein Sicherheitsstandard obsolet wird.

Da dieser Text das böse Wort „Snowden“ enthält, werde ich aus Sicherheitsgründen nicht mit meinem Namen unterzeichnen. Schließlich möchte ich nicht bei meiner nächsten Einreise in die USA festgehalten werden. Oh, jetzt enthält es auch noch die bösen Wörter „USA“ und „Einreise“ …

Herausgeber
(Name der Redaktion bekannt)

Referenzen

[1] Statista *Anteil der Internetnutzer in Deutschland von 2001 bis 2013* http://de.statista.com/statistik/daten/studie/13070/umfrage/entwicklung-der-internetnutzung-in-deutschland-seit-2001/

[2] iBusiness *Anzahl der täglich empfangenen E-Mails in Deutschland 2010* http://www.ibusiness.de/charts/ct/220114sb.html

Beitragsinformation

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

Beiträge

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

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

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

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

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

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

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

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

Leserbriefe

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

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

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

Werbung ist Information

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

- Stellenanzeigen
- Seminaranzeigen
- Produktinformation und -werbung

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

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

Abonnement

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

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

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

Copyright

Impressum

KaffeeKlatsch Jahrgang 6, Nummer 10, Oktober 2013
ISSN 1865-682X
Bookware – eine Initiative der
MATHEMA Verwaltungs- und Service-Gesellschaft mbH

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

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

Inhalt

Artikel

Kolumnen

Moderne Gesichter (Teil 2)

von William Siakam

Die Java EE 7, die im Juni diesen Jahres released wurde, enthält die Java Server Faces-Spezifikation in der Version 2.2. Im Frühjahr wurden zwei bedeutsame Neuerungen in einem Artikel vorgestellt. Nun möchten wir uns weitere Features ansehen, die diese neue Version mit sich bringt.

Gruppenzwang

von Michael Wiedeking

Zusammenhängende Einsergruppen zu zählen, hat sich im letzten Vergnügen als nicht all zu schwierig herausgestellt. Gelegentlich interessieren einen aber nur zusammenhängende Gruppen mit einer bestimmten Mindestgröße.

Ausgetrickst

von Michael Wiedeking

Als Alan Turing 1950 vorschlug einen Test zu entwickeln, um die Intelligenz eines Computers mit der eines Menschen zu vergleichen, konnte er nicht ahnen, dass schon 2013 die Maschine dabei besser abschneiden würde.

Moderne Gesichter (Teil 2)

Weitere neue Features in JSF 2.2

von William Siakam

Die Java EE 7, die im Juni diesen Jahres released wurde, enthält die Java Server Faces-Spezifikation in der Version 2.2. Im Frühjahr wurden zwei bedeutsame Neuerungen in einem Artikel [1] vorgestellt. Nun möchten wir uns weitere Features ansehen, die diese neue Version mit sich bringt.

HTTP-Get-Unterstützung

Ein HTTP-*Get* ist eine Methode um einen HTTP-*Request* an den Server zu schicken, indem die Request-Parameter als Wertepaare in der URL verschickt werden. Als *Response* bekommt der HTTP-Client eine HTTP-*Location* zurück.

Seit der Version 2.0 ist die Unterstützung der HTTP-*Get*-Methode durch neue *Tags* wie <*h:button*>, <*h:link*> und <*f:viewParam*> implementiert, was die Navigation zwischen Seiten, auch ohne dass ein *Managed Bean* einbezogen wird, möglich macht. Hierzu ein kleines Beispiel der Verwendung des <*h:button*> in der früheren Version.

```
<!-- Beispiel h:button  - seite.xhtml - ->
<h:button value="startseite" />

<!-- HTML output von seite.xhtml - ->
<input type="button"
   onclick="window.locatoin.href=
   '/aktuelleSeite.xhtml;return false' "
   value="startseite" />
```

Die Möglichkeit einen HTTP-*Get* von einer JSF-Seite aufzurufen, wurde seit JSF 2.2 so ergänzt, dass eine Aktion eines *Managed Bean* aufgerufen werden kann. Die Navigation in diesem Fall erfolgt ausschließlich mittels HTTP-*Redirect*.

Beispiel:

```
<!-- Beispiel viewAction – seite.xhtml - ->
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
  <f:metadata>
      <f:viewParam name="id"
        value="#{product.id}"/>
      <f:viewAction action= "#{product.showPriceInfo}"/>
  </f:metadata>
  <h:head>
    <title>JSF 2.2 - viewAction</title>
  </h:head>
  <h:body>
      <h:panelGrid columns="2">
        <h:outputText value="Name:"/>
        <h:outputText  value= "#{product.name}"/>
        <h:outputText value="Price:"/>
        <h:outputText  value= "#{product.price"/>
      </h:panelGrid>
  </h:body>
</html>
```

AJAX-Unterstützung

Wer eine performante, moderne und benutzerfreundliche Web-Anwendung schreiben möchte, kommt heutzutage an AJAX nicht vorbei. Die AJAX-Unterstützung, die bis JSF 2.0 nur durch *Rich-Client*-Bibliotheken wie RichFaces [2] oder PrimeFaces [3] möglich war, konnte ihre Geburt ab der 2.0-Version von JSF feiern. Der neue *Tag* <*f:ajax*> wurde eingeführt um JSF-Seiten AJAX-fähig zu machen und diese Lücke zu schließen. Hier ein Beispiel der früheren AJAX-Unterstützung:

```
<!-- Beispiel aus früher Version – seite.xhtml -->
<h:form>
  <h:commandButton value="Click">
  <f:ajax render="message"
    listener="#{bean.doSomething}" />
  </h:commandButton>
  <h:outputText id="message"
    value="#{bean.result}" />
</h:form>
```

In JSF 2.2 wurde diese Funktionalität um einen Mechanismus erweitert, der die AJAX-Requests in einer Warteschlange aufnimmt und diese am Server in der abgeschickten Reihenfolge abarbeitet. Zusätzlich wird

dem Entwickler eine besondere Flexibilität eingeräumt, welche erlaubt die Zeit zwischen zwei AJAX-Requests zu bestimmen.

```
<h:form>
  <h:inputText value="#{bean.customer}">
    <f:ajax event="onblur" render="list"
    delay="100" />
  </h:inputText>
  <h:panelGroup id="list" var="c">
    <ui:repeat value="#{bean.customers}">
    #{c.name}
    </ui:repeat>
  </h:panelGroup>
</h:form>
```

File Upload

Eine weitere sehr nützliche Neuerung, die sehr lange auf sich warten lies, ist das Vereinfachen des *Fileupload*-Mechanismus. Bis dato war das Hochladen von Dateien, was zur Grundfunktionalität eines entwicklungsunterstützenden Web-Frameworks gehören sollte, nur entweder manuell oder mit Hilfe eines JSF-Frameworks möglich. Das lang ersehnte Feature wurde endlich in dieser Version umgesetzt.

```
<!--File Upload auf JSF Seite – seite.xhtml-->

<h:form id="form" enctype="multipart/form-data">
  <h:inputFile id="file"
    value="#{bean.bescheinigung}"/>
  <h:commandButton value="Datei hochladen"
    action="#{bean.upload}"/>
</h:form>

//Managed Bean – Bean.java
import javax.servlet.http.Part;

public class Bean {

  private Part bescheinigung;
  private String inhalt;
  public void upload() {
    try {
      inhalt = new Scanner(
        bescheinigung.getInputStream())
    .useDelimiter("\\A").next();
    } catch (IOException e) {
     //Fehlerbehandlung
    }
  }

  public Bescheinigung getBescheinigung() {
    return bescheinigung;
  }

  public void setBescheinigung(
    Bescheinigung bescheinigung) {
    this.bescheinigung = bescheinigung;
  }
}
```

Ergänzend kann man einen *Validator* schreiben, der beispielsweise die Größe und den Inhalt der hochgeladenen Datei prüft und validiert.

Stateless Views

Hier geht es nicht darum, dass *View*-Instanzen im Server gepoolt oder als *Singleton* verwendet werden, wie es bei EJBs der Fall ist. Es bedeutet, dass der Zustand des Clients nicht server-seitig festgehalten wird. Der JSF-*Statemanager* braucht nicht die Werte des Komponentenbaums in die *Backing Beans* zu übertragen, was deutlich weniger CPU-Auslastung auf dem Server bedeutet. Das macht z. B. Sinn, wenn man einige Seiten der Anwendung verwenden möchte, ohne dass man sich als Benutzer einloggen muss. Ein Nachteil ist, dass der Komponentenbaum jedes Mal neu zum Client geschickt wird, was bei vielen Clients mehr Bandbreite beansprucht. Um dies zu realisieren reicht die Angabe des Attributes *transient* im *View*-Tag aus.

```
<f:view transient="true">

  <h:form>
    </h:form>

</f:view>
```

Window ID

Der Request unterschiedlicher Tabs eines Windows-Fensters konnte bis dato nicht server-seitig auseinander gehalten werden. Das Konzept von Cookies löst das Problem der Identifikation des Clients auf dem Server. Jedoch pro Browser und nicht pro Tab eines Browsers. Mit JSF 2.2 gibt es eine Lösung diesen Unterschied serverseitig festzustellen.

Dieses Feature wird durch einen Eintrag in *web.xml* eingeschaltet.

```
<context-param>
  <param-name>
  javax.faces.CLIENT_WINDOW_MODE
  </param-name>
    <param-value>url</param-value>
</context-param>
```

Zwei Werte sind bei dem Attribut <*param-value*> vorgesehen. Neben dem Default-Wert *none*, der nichts bewirkt, wird mit dem Wert *url* die Windows-ID, genauso wie bei der Session-ID, in der URL mitgeschickt, was dem Wert eines von JSF generierten versteckten Feldes entspricht. Die URL sieht dann beispielsweise so aus:

http://mathema.de/test?jfwid=78FzYGhGhYF7%b1

Möchte man diesen Mechanismus für eine gegebene JSF-Seite außer Kraft setzen, genügt es, wenn das Attribut *disableClientWindow* mit dem Feld *false* versehen wird. Dieses Attribut wird von zwei JSF-Komponenten unterstützt: <*h:link*> und <*h:button*>.

Fazit

Obwohl die neuen Features in JSF 2.2 keine revolutionären Vorsprünge im Vergleich zur vorherigen Version darstellen, wurden interessante Schritte eingegangen, um die JSF-Anwendungen ein Stück performanter und benutzerfreundlicher zu gestalten. Man hätte zwar bei drei Jahren Pause mehr erwartet, z. B. dass das CDI-Konzept mehr Terrain in der Spezifikation gewinnt, indem *Converters* und *Validatoren* als *Injection-Targets* verwendet werden. Letztendlich bleibt man bei der Entwicklung einer JSF-Anwendung so gut wie immer auf externe Hilfe von Frameworks angewiesen, die ein deutlich attraktiveres Komponenten-Portfolio aufweisen.

Referenzen

[1] Siakam, William *Moderne Gesichter*, http://www.bookware.de/kaffeeklatsch/archiv/KaffeeKlatsch-2013-01.pdf

[2] JBoss *Richfaces*, http://www.jboss.org/richfaces

[3] PrimeFaces *Ultimate JSF Component Suite*, http://primefaces.org

Kurzbiografie

William Siakam arbeitet als Software-Entwickler und -Consultant für die MATHEMA Software GmbH. Er interessiert sich besonders für die Java Standard und Enterprise Edition (Java SE und Java EE). Dabei liegt sein Schwerpunkt auf dem Design und der Anwendung von Web-Frameworks.

Des Programmierers kleine Vergnügen

Gruppenzwang

von Michael Wiedeking

Zusammenhängende Einsergruppen zu zählen, hat sich im letzten Vergnügen als nicht all zu schwierig herausgestellt.

Gelegentlich interessieren einen aber nur zusammenhängende Gruppen mit einer bestimmten Mindestgröße.

Mit dem letzten Vergnügen steht uns eine Funktion *groupCount* zur Verfügung, mit der die Anzahl der zusammenhängenden Gruppen gesetzter Bits gezählt werden kann. Dabei zählt ein einzelnes gesetztes Bit natürlich auch als eine Gruppe.

Sollen etwa nur die Gruppen mit mindestens zwei zusammenhängenden Bits gezählt werden, könnte man die existierende Funktion wiederverwenden, wenn man es nur schaffen würde sämtliche Bitgruppen um ein Bit zu vermindern. Dann nämlich würden einzelne Bits verschwinden und damit verblieben nur noch Gruppen, die vorher mindestens zwei Bits enthalten haben.

Und wie eliminiert man einzelne Bits? Indem man das Bit-Muster um eine Position (beispielsweise nach rechts) verschiebt und dann durch ein Bit-weises *Und* mit dem Original verknüpft. Das funktioniert deswegen, weil sich neben jeder Gruppe immer ein nicht gesetztes Bit befindet. So wird durch die *Und*-Verknüpfung immer das am linken Rand einer Gruppe befindliche Bit gelöscht.

```
1.11..1.111.11.1 x
.1.11..1.111.11. x >>> 1
...1.....11..1.. x & (x >>> 1)
```

Das stellt auch (bei einem Verschieben nach rechts) für die beiden Außenränder kein Problem dar, da ein einzelnes Bit einfach herausgeschoben und eine Null nachgeschoben wird und auch damit den erwünschten Effekt erzielt.

Damit erhält man die gesuchte Funktion für Gruppen mit mindestens zwei gesetzten Bits.

```
int groupGreaterOneCount(int x) {
  return groupCount(x & (x >>> 1));
}
```

Braucht man nun die Anzahl der Gruppen mit mindestens drei gesetzten Bits, so kann man das analog machen, indem man die Gruppen um ein Element verkleinert und es anschließend auf den bereits bekannten Fall zurückführt.

```
int groupGreaterTwoCount(int x) {
  return groupGreaterOneCount(x & (x >>> 1));
}

int groupGreaterThreeCount(int x) {
  return groupGreaterTwoCount(x & (x >>> 1));
}
```

Das kann man nun nach Belieben soweit sinnvoll fortsetzen. Irgendwann kommt natürlich der Punkt, dass die Anzahl der auszuführenden Schritte weit über denen der „einfachen“ Schleife liegt. Wie immer gilt es hier also abzuwägen, was denn die schnellste Lösung für das Problem ist.

PS: Wem die Funktion für mehr als zwei zusammenhängende Bits zu langsam ist, kann diese noch um zwei Instruktionen kürzen. In der Implementierung

```
int groupCount(int m) {
  return (
    Integer.bitCount(m ^ (m >>> 1)) + (m & 1)
  ) / 2;
}
```

musste ja die Korrektur … + (*m* & 1) vorgenommen werden, damit (beim Verschieben nach rechts) auch eine Bit-Gruppe mit nur einem Bit am rechten äußeren Rand erkannt wird. Darauf kann verzichtet werden, wenn man etwa im Fall von Dreiergruppen nicht zweimal nach rechts, sondern einmal nach rechts und einmal nach links verschiebt.

```
int groupGreaterTwoCount(int x) {
  return groupCount((x & (x >>> 1)) & (x << 1);
}
```

Damit ist gewährleistet, dass sich kein einzelnes Bit mehr an den äußeren Rändern befindet, womit diese Korrektur überflüssig wird.

Deutsch für Informatiker

Über Larts, Ingrid und das Heisen

Das bemerkenswerteste an Sprache ist sicherlich ihre Anpassungsfähigkeit. Diesbezüglich leistet ganz besonders die Jugend ihren Beitrag. Und weil es immer wieder eine neue Jugend mit einem eigenem, neuen Wortschatz gibt, hat der Duden jetzt Abhilfe geschaffen.

Ein *Lart* ist ein Gegenstand, mit dem ein Systemverwalter widerspenstige, nervige oder – auch wiederholt – auffällig gewordene Benutzer „auf den Pfad der Tugend" zurückführen kann. Oftmals ist es ein ausreichend großes Stück Holz, mit dem man den Benutzern Schläge androht, sollten sie sich daneben benehmen oder das System beschädigen. Davon abgeleitet beschreibt das *Larten* die mit einem *Lart* durchgeführte Tätigkeit, also die Bestrafung eines Anwenders für ein (vermeidbares) Fehlverhalten durch (s)einen Systemverwalter.

So heißt es beispielsweise in dem Szenensprachenwiki [1], das der Dudenverlag als Möglichkeit geschaffen hat, um neue Wörter zu sammeln und zu dokumentieren. Dort gibt es nicht nur Wörter die mit Computer und Technik zu tun haben; die bis zum Schreiben dieser Kolumne enthaltenen 2729 Wörter decken alle Bereiche des täglichen Lebens ab. In vierzehn Kategorien von Lifestyle & Wohnen, über Partnerschaft & Freundschaft, bis hin zu Gesellschaft & Politik und Feierabend & Nachtleben ist praktisch alles vertreten.

Für uns ist natürlich der Bereich rund um den Computer am interessantesten, weil wir auf diese Weise erfahren können, was den Anwender wirklich bewegt – so sehr, dass er dafür ein eigenes, meist deutlich präziseres Wort ins Leben rufen muss. Das Szenewörterbuch beschreibt deshalb auch normalere Begriffe wie etwa *Add-on* und *Ajax* oder *Zero-Day-Exploit* und *Zombie*.

So ist etwa *Ingrid* ein Kommentar, den der Autor zu seinem eigenen Beitrag macht. *Die Ingrid machen* ist die dazugehörige Tätigkeit. In dem Kommentar gibt es sogar eine etymologische Dokumentation, so dass ich sogar erfahre, warum es eine (real existierende) Ingrid ist, die namensgebend herhalten muss, und nicht etwa Yvonne oder Chantal. Apropos Chantal: Der *Chantalismus* ist die weibliche Form des *Kevinismus*, der die Unfähigkeit von Eltern beschreibt, ihren Kindern sozialverträgliche Namen zu geben.

Das *Heisen* schließlich ist das – für mein Dafürhalten sehr tragische – Ereignis, wenn eine Internet-Seite „durch die Nennung im großen News-Portal *heise.de* so oft aufgerufen wird, dass sie zusammenbricht". Ein Kommentar weist hier sogar darauf hin, dass es sich beim *Heisen* um das deutschsprachige Pendant zum *Slashdotting* handelt, bei dem ein Zusammenbruch durch die Erwähnung auf *slashdot.org* verursacht wird. Darüber hinaus erklärt der Kommentator, dass es sich dabei um eine nicht bösartige Form einer *Denial-of-Service*-Attacke handelt.

Dieses Sprachen-Wiki ist durchaus einen Besuch wert. Wer ein bestimmtes Wort nicht findet, kann es dort recht einfach eingeben, wobei die Kombination mit der Suche verhindert, dass ein Wort mehrfach definiert wird. Dabei haben viele der Beschreibungen einen so hohen Unterhaltungswert, dass sie auch Douglas Adams' legendärem *The Meaning of Liff* [2] entstammen könnten.

Referenzen

[1] Duden *Duden Neues Wörterbuch der Szenesprachen,* http://szenesprachenwiki.de/

[2] Adams, Douglas; Lloyd, John *The Meaning of Liff,* Pan Books, London, 1983 siehe auch: http://folk.uio.no/alied/TMoL.html

Weiterführende Literatur

- Adams, Douglas; Lloyd, John *The deeper Meaning of Liff,* Pan Books, London, 1990
- Böttcher, Sven *Der tiefere Sinn des Labenz – Das Wörterbuch der bisher unbenannten Gegenstände und Gefühle,* Rogner & Bernhard, Frankfurt/M., 1999
- Labenz *Freut euch des Labenz! Die Website der bisher unbenannten Gegenstände und Gefühle,* http://labenz.texttheater.net/index.php

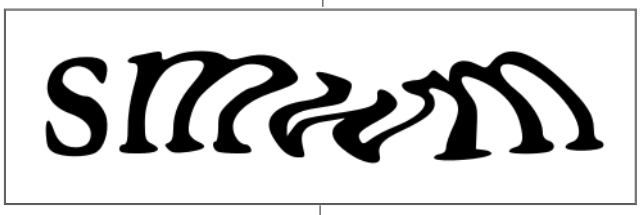
Kaffeesatz

Ausgetrickst

von Michael Wiedeking

Als Alan Turing 1950 vorschlug einen Test zu entwickeln, um die Intelligenz eines Computers mit der eines Menschen zu vergleichen, konnte er nicht ahnen, dass schon 2013 die Maschine dabei besser abschneiden würde.

Zugegeben, so ganz stellt sich die Sache nicht dar. Bei Turings ursprünglichem Test [1] sitzt eine Testperson mit Tastatur und Bildschirm ohne weiteren Hör- und Sichtkontakt einem Menschen und einer Maschine „gegenüber“ und soll über fünfminütiges Fragen und Antworten herausbekommen, ob er den Menschen von der Maschine unterscheiden kann. Können Mensch und Maschine die Testperson davon überzeugen, Mensch zu sein, dann hat die Maschine den Turing-Test bestanden und gilt als zum Menschen gleichwertig intelligent. Turing prognostizierte, dass schon im Jahr 2000 ein durchschnittlicher Tester nur noch in sieben von zehn Fällen Mensch und Maschine erfolgreich unterscheiden kann.



Turing wusste natürlich damals noch nichts vom Internet und dem Problem, dass sich Maschinen als Menschen ausgeben würden, um Blog-Einträge zu machen, E-Mails zu verschicken und sonstigen Schindluder zu treiben. Und damit man sich die unliebsamen Besucher nicht erst nach fünf Minuten vom Halse schaffen konnte, ersann man den vereinfachten *Completely Automated Public Turing test to tell Computers and Humans Apart* [2]. Ein CAPTCHA soll also möglichst schnell und zuverlässig Bots die Möglichkeit nehmen, sich als Mensch auszugeben.

Jetzt schreiben wir das Jahr 2013 und seit einigen Tagen gilt die Maschine als durchschnittlich cleverer als der Mensch, denn um so komplizierter diese CAPTCHAS werden, um so schwieriger tut sich der Mensch damit. Einer kalifornischen Firma will es nun gelungen sein mit virtuellen Neuronen sämtliche verfügbaren CAPTCHAS mit bisher nicht erreichter Zuverlässigkeit „knacken“ zu können. So wollen sie Googles reCAPTCHA in über 90% der Fälle erfolgreich überlistet und noch bessere Ergebnisse bei Yahoo, Paypal und Captcha.com erzielt haben [3].

Die Firma möchte mit ihren Ansätzen allerdings weniger Missbrauch betreiben, als herauszufinden, wie man auch noch andere Turing-Tests bestehen kann. Dafür gibt es dann genügend andere Firmen, die mit nicht ganz so guten Quoten, aber dafür mit erschwinglichen Preisen locken. So kann man also Software erwerben, mit deren Hilfe man etwa seine Werbebotschaften quasi überall unterbringen kann, wenn sie nur durch CAPTCHAS geschützt sind.

Wem das nicht zuverlässig genug ist, der kann sich bei Dienstleistern erkundigen, die das Schlagen von CAPTCHAS nicht mit einer Maschine, sondern mit einer Batterie von Menschen erledigen. Das Ergebnis ist dann aber vergleichbar: Der Zugang wird gewährt – nur statistisch gesehen nicht ganz so erfolgreich wie eben mit einer Maschine. Dafür aber „korrekt“, da ja wirklich Menschen vor dem Rechner sitzen.

Referenzen

[1] Wikipedia *Turing-Test*
http://de.wikipedia.org/wiki/Turing-Test

[2] Wikipedia *CAPTCHA*
http://de.wikipedia.org/wiki/Captcha

[3] MacGregor, Campbell *Software beats CAPTCHA, the web's 'are you human?' test*
http://www.newscientist.com/article/dn24476-software-beats-captcha-the-webs-are-you-human-test.html

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

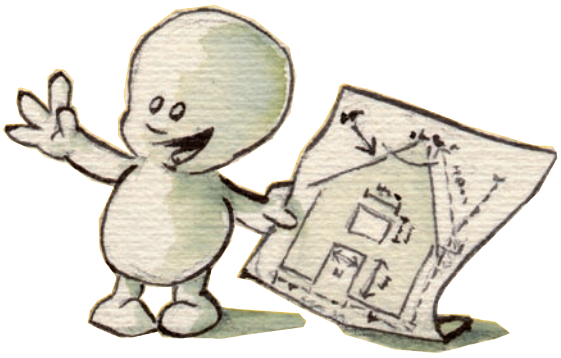
Deutschland, Schweiz, Österreich

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

- **Spring Framework**
  JavaEE ganz ohne EJB
  12. – 14. November 2013, 1.315,– € (zzgl. 19 % MwSt.)
- **Mobile Anwendungen für das Apple iPhone**
  25. – 27. November 2013, 1.180,- € (zzgl. 19 % MwSt.)
- **Sicherheitskonzepte unter Java**
  Mechanismen für Datenschutz und Datensicherheit in Netzwerken
  28. – 29. November 2013, 925,– € (zzgl. 19 % MwSt.)
- **JEE Anwendungsentwicklung**
  Entwicklung moderner, skalierbarer Anwendungen mit der Java Enterprise Edition (JEE)
  16. – 20. Dezember 2013, 2.095,– € (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

Software-Entwickler (m/w)
Software-Architekt (m/w)

Arbeiten Sie gerne selbstständig, motiviert und im Team? Haben Sie gesunden Ehrgeiz und Lust, Verantwortung zu übernehmen?

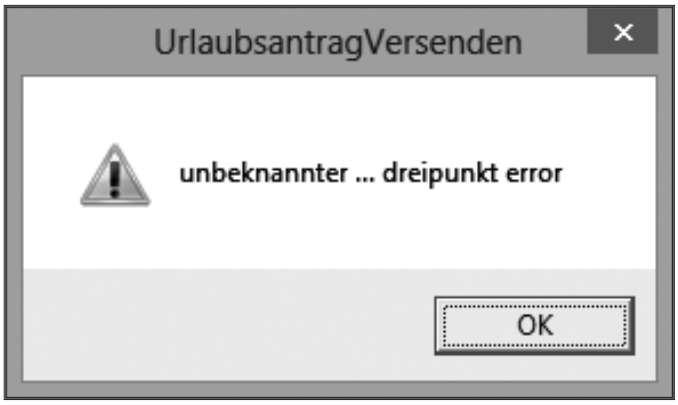
Wir bieten Ihnen erstklassigen Wissensaustausch, ein tolles Umfeld, spannende Projekte in den unterschiedlichsten Branchen und Bereichen sowie herausfordernde Produktentwicklung.

Wenn Sie ihr Know-how gerne auch als Trainer oder Coach weitergeben möchten, Sie über Berufserfahrung mit verteilten Systemen verfügen und Ihnen Komponenten- und Objektorientierung im .Net- oder JEE-Umfeld vertraut sind, dann lernen Sie uns doch kennen.

Wir freuen uns auf Ihre Bewerbung!

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

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 November.

Herbstcampus

Wissenstransfer par excellence

1. – 4. September 2014

in Nürnberg