KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

ISSN 1865-682X

09/2012

Jahrgang 5

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

Über die Philosophie des Testens

Wenn man einen Blick auf die Curricula sowohl an Hoch- als auch Berufsschulen wirft, muss man enttäuscht feststellen, dass Testen nicht zu den wichtigsten Themen gehört. So hat man es in glücklichen Fällen zwar mit Berufsanfängern zu tun, die (mehr oder weniger gut) programmieren können, aber das Glück reicht meist nicht so weit, dass sie auch testen können.

Das soll nicht heißen, dass diese jungen Entwickler keine funktionierenden Programme schreiben können. Ganz im Gegenteil: Die begabten Entwickler erstellen guten (sogar oft lesbaren) Code, der mit (nicht zu) cleveren und innovativen Ideen durchsetzt und scheinbar fehlerfrei ist. Wozu also soll man diesen Code dann noch testen? Erschwerend scheint das Einzige, was alle über das Testen wissen, die Tatsache zu sein, dass sich durch Tests nur die Abwesenheit bestimmter Fehler belegen lässt, nicht aber die Korrektheit eines Programms.

Und hier liegt oft das eigentliche Problem: Wie bringt man jemandem – der offensichtlich alles richtig macht – bei, wie, was und warum man testet? Wenn jemand lang genug alleine oder in einer überschaubaren Gruppe Software entwickelt hat – wobei es dabei in der Regel vermessen ist von „Software" zu reden, denn selbst die meisten Studien- oder Diplomarbeiten sind eher kleinere Programme – wird sich in den seltensten Fällen die Notwendigkeit für strukturiertes Testen ergeben.

Ein funktionierendes Programm gibt den Entwicklern in ihrer Annahme recht, dass ihnen Testen nicht geholfen hätte. Und die im Rahmen der Ausbildung erstellten Arbeiten haben nicht selten die Eigenschaft nach dem Abschluss nie wieder angefasst zu werden – was dazu führt, dass auch noch die einzigen „Tests" wegfallen, die überhaupt gemacht würden: Die Bedienung des Programms durch den Benutzer.

Neulich tauchte in einer Diskussion über die Notwendigkeit des Testens die Frage auf, ob es nicht zu aufwändig wäre, durch Testen zu belegen, dass die Software keinen einzigen Fehler enthält. Schnell wurde klar, dass es schon genügen würde, vorweg nur die Fehler zu erkennen, die sonst der geplagte Benutzer zu spüren bekäme. Wenn man dann noch sicher stellte, dass der Benutzer nichts anderes als das eingeben kann, was schon getestet wurde, wäre aus Sicht des Benutzers die Software fehlerfrei – womit man schon extrem viel erreicht hätte.

Stellvertretend für den Benutzer steht natürlich die Anforderung mit den konkreten Anwendungsfällen. Eigentlich ist es sehr schade, dass mit diesen Anforderungen nicht gleich die Tests mitgeliefert werden. Damit wäre der Entwickler entlastet und müsste sich etwa nur auf das Schreiben der Unit-Tests konzentrieren. Bestandene Tests könnten dann auch zur Gewissheit beitragen, dass die Spezifikation richtig verstanden und korrekt umgesetzt wurde.

Was sich dann hinter den Kulissen in der Software abspielt, ist eigentlich egal. Allerdings verbleibt immer ein Restrisiko, das bestimmte Abläufe nicht getestet wurden oder eine Fehlbedienung nicht ausgeschlossen werden kann. Und ist es nicht die wahre Kunst des Testens, soviel geschickt wie möglich zu testen und damit das unvorhersehbare Risiko zu minimieren? Diese Risikominimierung ist umso wichtiger, umso fataler die Folgen für einen unvorhergesehenen Fehler sind – insbesondere wenn Menschenleben davon abhängen können.

Wie in so vielen Fällen kann aber auch nur dann sinnvoll getestet werden, wenn die Testziele bekannt sind. So ist es durchaus legitim – was eher zu unrecht einem großen amerikanischen Betriebssystemhersteller unterstellt wird – seine Benutzer als Tester zu missbrauchen. Aber natürlich nur dann, wenn Kundenzufriedenheit nicht zu den Qualitätsmerkmalen der zu erstellenden Software gehört.

Es bleibt aber zu befürchten, dass nur diejenigen wirklich ans Testen glauben, die die Folgen ungetesteter Software „am eigenen Leib" verspürt haben, emphatisch oder phantasievoll genug sind, auf diese Erfahrung verzichten zu können oder den Wunsch haben, einen guten Job zu machen.

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 Alexandra Specht via E-Mail unter alexandra.specht@bookware.de oder telefonisch unter 09131/8903-14.

Copyright

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

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

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

Impressum

KaffeeKlatsch Jahrgang 5, Nummer 9, September 2012
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 03-55
E-Mail: redaktion@bookware.de
Internet: www.bookware.de

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

Inhalt

Editorial 3
Beitragsinfo 4
Inhalt 5
User Groups 17
Werbung 19
Das Allerletzte 21

Artikel

Android-Applikationen von der Stange

Auslagerung wiederkehrender Code-Muster mit Hilfe der ADT-Templates 6

Die aktuelle Version des *Android Development Tools* (ADT) Plug-ins für Eclipse brachte neben tiefgreifenden Änderungen am Layout-Editor und einer verbesserten Unterstützung für Android XML-Dateien erstmals die Möglichkeit, bei der Erstellung von Projekten und *Activies* Code-Grundgerüste an Hand von *Templates* zu generieren. Dies erspart dem Entwickler einen gerade bei Android nicht geringen Anteil an Boilerplate-Code. Bisher existiert allerdings nur eine Handvoll Templates, die allesamt nicht verändert werden können. In der kommenden Version 21 des ADT-Plug-ins ist die Erstellung und Integration eigener Templates jedoch möglich, weshalb in diesem Artikel schon einmal ein Blick auf deren Struktur und Erstellung geworfen wird.

Kolumnen

Führende Nullen

Des Programmierers kleine Vergnügen 12

Herbstcampi, -campen, und -campoden

Deutsch für Informatiker 15

Wer hat's erfunden?

Kaffeesatz 16

Android-Applikationen von der Stange

Auslagerung wiederkehrender Code-Muster mit Hilfe der ADT-Templates

von Johannes Köstler

Die aktuelle Version des *Android Development Tools* (ADT) Plug-ins für Eclipse brachte neben tiefgreifenden Änderungen am Layout-Editor und einer verbesserten Unterstützung für Android XML-Dateien erstmals die Möglichkeit, bei der Erstellung von Projekten und *Activies* Code-Grundgerüste an Hand von *Templates* zu generieren. Dies erspart dem Entwickler einen gerade bei Android nicht geringen Anteil an Boilerplate-Code. Bisher existiert allerdings nur eine Handvoll Templates, die allesamt nicht verändert werden können. In der kommenden Version 21 des ADT-Plug-ins ist die Erstellung und Integration eigener Templates jedoch möglich, weshalb in diesem Artikel schon einmal ein Blick auf deren Struktur und Erstellung geworfen wird.

Nach einem Update auf die aktuelle Version des ADT-Plug-ins in Eclipse wird spätestens bei der Erstellung eines neuen Android-Projekts der neu hinzugekommene *Activity-Wizard* auffallen. Dieser verlangt auf der ersten Seite die Selektion eines Templates, bevor auf der Folgeseite die eigentlichen Eigenschaften der Activity abgefragt werden (vgl. Abbildung 1). Dazu gehören der Titel, der Klassenname und der Name der zugehörigen Layout-Datei. Zusätzlich muss die gewünschte Navigationsart ausgewählt werden. Kommt in dieser die Action Bar zum Einsatz kann eine Activity als Ziel für die „Nach-oben"-Navigationsfunktion der *Action Bar* definiert werden. Nachdem alle Angaben getätigt und der Wizard geschlossen wurde, erstellt Eclipse automatisch die Klassendatei sowie die von der Activity genutzten Ressourcen (wie Layouts, Menüs etc.) und fügt die Activity-Definition in das *Manifest* der Applikation ein.

Abbildung 1: Activity Template

Soweit eine hilfreiche Funktion, die dem Entwickler mancherlei Boilerplate-Code abnimmt, der gelegentlich ja auch nur aus bereits vorhandenen Source-Dateien zusammenkopiert wird. Bei den richtigen Templates kann es sich zumindest um Best-Practices handeln. Ein weiteres Plus ist die – gerne vergessene – Manifest-Integration. Auch die Zeit, die man sich für das Anlegen der zahlreichen Ressourcendateien spart, die jede Activity mit sich bringt, ist eine Überlegung Wert, den Template-Mechanismus zumindest einmal auszuprobieren. Richtig interessant wird es aber erst, wenn der Template-Mechanismus mit eigenen Templates angereichert werden kann, wie es im aktuellen Preview Release zum ADT-Plug-in 21 der Fall ist. Neben weiteren Templates – wie beispielsweise einem Template für eine *Login-Activity* oder häufig verwendeten Applikationskomponenten wie dem *Content Provider* beziehungsweise dem *Broadcast Receiver*, die im Preview Release enthalten sind – finden sich bereits jetzt erste Template-Sammlungen im Netz, darunter auch Templates für den Einsatz der *ActionBarSherlock* [1]. Grund genug sich im Folgenden näher mit dem neuen Template-Mechanismus zu befassen.

Template-Struktur

Grundlage für die Generierung des Codes bildet die *FreeMarker* Template-Engine [2], die auch in einer Vielzahl von Java-Frameworks zum Einsatz kommt. Bei der Verwendung eines Templates beim Erstellen eines Projekts beziehungsweise einer Activity werden mit Hilfe der *FreeMarker*-Bibliothek Teile der sich im Template-Verzeichnis befindlichen Quellcodedateien in das aktuelle Projektverzeichnis kopiert. Die ursprünglichen Quellcodedateien werden dazu mit der *FreeMarker Template Language* (FTL) durchsetzt. Auf diese Weise ist es möglich Variablen des *FreeMarker Data-Models* in den Quellcode einzufügen oder bestimmte Teile des Code mittels in *FTL-Tags* definierte Bedingungen explizit ein- oder auszuschließen. Zusätzlich gibt es die Möglichkeit mittels eigens definierter Direktiven und Methoden Texttransformationen durchzuführen. Im ADT-Plug-in finden sich eine Vielzahl solcher Methoden, die beispielsweise eine Umwandlung des Klassennamens in den entsprechenden Layout-Dateinamen bereitstellen. Dies ist notwendig da in Ressourcendateinamen bekanntermaßen keine Camel-Case-Schreibweise, wie bei Klassennamen üblich, verwendet werden kann.

Das Template-Verzeichnis befindet sich im SDK-Installationsverzeichnis unter *tools/templates*. Darin enthalten sind zwei Ordner (*projects* und *activities*) für Projekt- beziehungsweise Activity-Templates sowie ein weiterer Ordner (*other*) mit Vorlagen für andere Applikationsbestandteile wie UI-Komponenten oder Content Provider. Die Templates verfügen jeweils über einen eigenen Ordner, dessen Struktur in Abbildung 2 dargestellt ist. Dabei kennzeichnet die Dateiendung *.ftl* Dateien, die FTL Anweisungen enthalten.

CustomTemplate/

- template.xml
- recipe.xml.ftl
- globals.xml.ftl
- template.png
- template_foo.png
- template_bar.png
- root/
  - AndroidManifest.xml.ftl
  - res/
    - …
  - src/
    - app_package/
      - CustomActivity.java.ftl

Abbildung 2: Verzeichnisstruktur des Templates (vgl. [3])

Die Datei *template.xml* ist die zentrale Konfigurationsdatei des Templates. Neben der Festlegung von Metainformationen und dem Einbinden weiterer Konfigurationsdateien werden darin so genannte Parameter definiert, die einen Teil des Data-Models von FreeMarker ausmachen. Zusätzlich können in der Datei *globals.xml.ftl* dem Data-Model weitere globale Variablen hinzugefügt werden. Die eigentliche Ausführungslogik ist innerhalb der Datei *recipe.xml.ftl* spezifiziert. Darin finden sich neben einfachen FreeMarker-Direktiven drei mehr oder weniger komplexe Kopieroperationen – *copy*, *instatiate* und *merge*. Die *copy*-Anweisung kopiert eine Datei eins zu eins von einer Quelle zu einem Ziel. Die *instantiate*-Anweisung hingegen führt eine Transformation der Quelldatei in eine Zieldatei durch, bei der die Quelldatei von einem *FreeMarker*-Processor verarbeitet wird, während bei der *merge*-Anweisung die Inhalte der Quell- und Zieldatei vereinigt werden. Alle Quelldateien befinden sich unterhalb des Verzeichnisses *root*, welches wiederum die bekannte Verzeichnisstruktur eines Android-Projekts aufweist. Die einzige Auffälligkeit – abgesehen von den *FreeMarker*-Dateiendungen – stellt der Ordner *app-package* dar, welcher später beim Kopieren der Quelldateien durch den *Package*-Pfad ersetzt wird. Die ausführliche Spezifikation des Template-Formats ist unter [3] abrufbar.

Template-Erstellung

Die Template-Erstellung soll an einem kleinen Beispiel-Template vorgestellt werden. Dabei handelt es sich um eine einfache Activity, die mit Hilfe einer *WebView* eine Web-Seite anzeigt. In einer zweiten Variante verfügt die Activity noch über eine Adressleiste, mit deren Hilfe auch eine neue Web-Seite angesteuert werden kann. Bei beiden Varianten soll die Standard-URL, die beim Start der Activity geladen wird, über das Template definiert werden können. Sicherlich benötigt eine solch simple Activity nicht zwangsweise ein eigenes Template – als Beispiel eignet sie sich jedoch hervorragend, da mit ihr viele Anwendungsfälle abgedeckt werden können. Wie oben bereits angedeutet, funktioniert dieses Beispiel nur, wenn das *Preview Release* des ADT-Plug-ins installiert ist. Genauere Informationen zum Wechsel auf das *Preview Release* und ebenso zur Rückkehr auf das *Stable Release* finden sich unter [4].

Ausgangspunkt für das neue Template ist die Datei *template.xml*. Im *template*-Element werden zunächst einige Metainformationen definiert. Das *format*-Attribut gibt Aufschluss über die Version des *Template*-Formats. Hier ist bislang nur der Wert 3 zulässig. Das *revision*-

Attribut spezifiziert hingegen die Version des Templates. Darauf folgen der Name sowie eine Beschreibung der Vorlage. Die Werte der beiden Attribute werden später im Eclipse-Wizard angezeigt (vgl. Abbildung 1). Außerdem könnten an dieser Stelle die erforderlichen Android-Versionen angegeben werden, die das Template voraussetzt.

Die nachstehende Definition einer Template-Kategorie ist optional. Mögliche Werte sind Applications, Activities und UI Components. Diese werden sicherlich noch erweitert werden, da ein oben bereits erwähnter *Content Provider* nicht wirklich in eine der drei Kategorien passt. Auch die Kategorie *Applications* ist derzeitig noch nicht wirklich von Bedeutung, da die Erstellung von *Application*-Templates bisher noch nicht unterstützt wird. Generell ist geplant diese Funktionalität auch für benutzerdefinierte Templates zu öffnen – ob dies aber bereits im nächsten Release oder in einem darauf folgenden Release umgesetzt wird ist noch nicht bekannt.

Anschließend geht es an die Definition der Parameter, deren Werte später im *Template-Wizard* abgefragt werden (vgl. Abbildung 1). Dazu werden die Parameter der Reihe nach aus der XML-Datei ausgelesen und in das Formular gerendert. Dies ist auch der Grund dafür, dass neben den Parametertypen Zeichenkette, boolescher Wert und Aufzählung auch noch der etwas Fehl am Platz wirkende Typ *separator* existiert. Mit Hilfe eines solchen Parameters kann eine horizontale Trennlinie an die Stelle des Parameters in das Formular einfügt werden. Neben dem Typ besitzen alle Parameter eine ID, einen Anzeigenamen sowie einen Hilfetext. Optional sind ein Standardwert definierbar und der geforderte Wert mit Hilfe des *constraints*-Attributs näher spezifizierbar. Beim *suggest*-Attribut handelt es sich um einen dynamischen Standardwert, der gesetzt wird, wenn der Wert dieses Parameters noch mit dem Standardwert übereinstimmt und der Wert eines anderen Parameters geändert wird. Im unten stehenden Beispiel lassen sich so Klassenname und Titel der Activity sowie der Name der Layout-Datei synchronisieren. Die Werte der beiden obenstehenden *suggest*-Attribute verweisen auf die bereits vorgestellten *FreeMarker*-Methoden. In diesem Falle kümmern sich die Methoden um die korrekte Konvertierung der Klassennamen in Layout-Dateinamen und umgekehrt. Bei Parametern vom Typ Aufzählung müssen die zur Verfügung stehenden Werte in jeweils einem *option*-Element zur Verfügung gestellt werden. Jeder dieser Werte könnte mit dem *minApi*-Attribut eine andere Android-Version als Voraussetzung für die Verwendbarkeit des Templates definieren.

Im Beispiel-Template werden zunächst die bereits bekannten Activity-Eigenschaften Klassenname, Name der Layout-Datei und Titel der Activity definiert. Des Weiteren kann eine Activity als Launcher Activity markiert werden, damit diese beim Start der Applikation automatisch ausgeführt wird. Der boolesche Wert wird später als Checkbox im Formular des Eclipse-Wizards dargestellt. Um die semantische Trennung der Parameter auch im Formular widerzuspiegeln folgt die Definition einer horizontalen Trennlinie. Zuletzt werden die oben geforderten Eigenschaften des Templates, die Möglichkeit zur Angabe der Standard-URL sowie die Auswahl von Adressleistenvarianten, auf zwei weiteren Parametern abgebildet.

```
<?xml version="1.0"?>
<template format="3" revision="1"
  name="HelloWebView"
  description="Creates a new activity with a
    web view and optional controls.">

  <category value="Activities" />

  <parameter id="activityClass"
    name="Activity Name"
    type="string"
    constraints="class|nonempty"
    suggest="${layoutToActivity(layoutName)}"
    default="MainActivity"
    help="The name of the activity class to create" />

  <parameter id="layoutName"
    name="Layout Name"
    type="string"
    constraints="layout|unique|nonempty"
    suggest="${activityToLayout(activityClass)}"
    default="activity_main"
    help="The name of the layout to create for the activity" />

  <parameter id="activityTitle"
    name="Title"
    type="string"
    constraints="nonempty"
    default="MainActivity"
    suggest="${activityClass}"
    help="The name of the activity. For launcher activities,
      the application title." />

  <parameter id="isLauncher"
    name="Launcher Activity"
    type="boolean"
    default="false"
    help="If true, this activity will have a
      CATEGORY_LAUNCHER intent filter,
        making it visible in the launcher." />

  <parameter type="separator" />
```

```
<parameter
  id="initialUrl"
  name="Initial url"
  type="string"
  constraints="nonempty"
  default="http://www.bookware.de"
  help="The initial url that will be loaded at startup." />

<parameter
  id="addressBar"
  name="Adress bar"
  type="enum"
  default="none"
  help="The type of the address bar to use" >
    <option id="none" default="true">None</option>
    <option id="simple">Simple</option>
</parameter>

<thumbs>
  <thumb>template_webview_activity.png</thumb>
  <thumb addressBar=
    "none">template_webview_activity.png</thumb>
  <thumb addressBar=
  "simple">template_webview_activity_simple.png</thumb>
</thumbs>

<globals file="globals.xml.ftl" />
<execute file="recipe.xml.ftl" />

</template>
```

Auf die Parameterdefinition folgt die Spezifizierung der Vorschaubilder. Die Pfadangaben der Bilddateien sind dabei relativ zur Template-Datei. Mit Hilfe der Angabe von Parameter-IDs als Attribute innerhalb des *thumb*-Elements können unterschiedliche Bilder abhängig vom jeweils ausgewählten Wert definiert werden. Das zutreffende Bild wird im Activity-Wizard rechts neben dem Formular angezeigt (vgl. Abbildung 1).

Auch im Beispiel-Template wird eine Datei für die globalen Variablen angelegt, obgleich nur eine Variable benötigt wird. Jene definiert den Pfad zum Ausgabeverzeichnis der Activities. Dazu werden mittels der Methode *slashedPackageName* die im *Package*-Pfad der Applikation enthaltenen Punkte durch Schrägstriche ersetzt und das Ergebnis der Variablen zugewiesen.

```
<globals>
  <global id="srcOut"
  value="src/${slashedPackageName(packageName)}" />
</globals>
```

Bevor die Ausführungslogik in der Datei *recipes.xml.ftl* definiert werden kann, müssen erst noch die eigentlichen Code-Vorlagen erstellt werden. Den Anfang macht dabei das Layout der Activity *activity_webview.xml.ftl*. Innerhalb der FTL-Datei können alle Variablen des Data-Models verwendet werden, indem deren Name in geschweiften Klammern mit einem voranstehenden Dollarzeichen in die Datei eingefügt werden. So wird im *context*-Attribut, welches eingeführt wurde, um in der Layout-Datei den Kontext der dazugehörigen Activity nutzen zu können, der Klassenname der Activity eingefügt. Über die *if*-Tags der FTL werden die UI-Komponenten, die zusammen die Adressleiste der Activity bilden nur eingebunden, wenn es sich nicht um die Template-Variante ohne Adressleiste handelt. Da die Layout-Parameter der *WebView* abhängig von den Adressleistenkomponenten sind, werden diese ebenfalls an die Bedingung geknüpft.

```
<RelativeLayout xmlns:android=
  "http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".${activityClass}">
<#if addressBar != "none">
  <EditText
    android:id="@+id/addressBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/goButton"
    android:layout_alignParentLeft="true"
    android:layout_toRightOf="@id/goButton"
    android:ems="10"
    android:hint="@string/addressBarHint"
    android:inputType="textUri" />

  <Button
    android:id="@id/goButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:onClick="onGoClicked"
    android:text="@string/go" />

  <ImageView
    android:id="@+id/separator"
    android:layout_width="match_parent”
    android:layout_height="1dp"
    android:layout_below="@id/goButton"
    android:background="@drawable/separator"
    tools:ignore="ContentDescription" />
  </#if>
<WebView
  android:id="@+id/webContent"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
```

```
        <#if addressBar != "none">
          android:layout_alignParentLeft="true"
          android:layout_below="@id/separator"
        <#else>
          android:layout_alignParentLeft="true"
          android:layout_alignParentTop="true"
        </#if>>

        <requestFocus />
      </WebView>

    </RelativeLayout>
```

In der zugehörigen *Activity*-Klasse zeichnet sich ein ähnliches Bild. Alle Anweisungen die sich nur auf die Adressleiste beziehen, werden abhängig von der Bedingung in- oder exkludiert. Daneben werden wieder einige Platzhalter definiert, die später durch die Werte aus dem Data-Model ersetzt werden, namentlich der Paket-, Layout- und der Klassenname sowie die Standard-URL.

```
package ${packageName};

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;
<#if addressBar != "none">
  import android.view.View;
  import android.widget.Button;
  import android.widget.TextView;
</#if>

public class ${activityClass} extends Activity {

  private static final String initialUrl = "${initialUrl}";

<#if addressBar != "none">
  private TextView mAddressBar;
  @SuppressWarnings("unused")
  private Button mGoButton;
</#if>

  private WebView mWebView;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.${layoutName});

<#if addressBar != "none">
  mAddressBar =
    (TextView) findViewById(R.id.addressBar);
  mGoButton =
    (Button) findViewById(R.id.goButton);
</#if>
  mWebView =
    (WebView) findViewById(R.id.webContent);
    mWebView.setWebViewClient(new WebViewClient());
    mWebView.getSettings()
      .setLoadWithOverviewMode(true);
    mWebView.getSettings().setUseWideViewPort(true);

<#if addressBar != "none">
  mAddressBar.setText(initialUrl);
</#if>
  mWebView.loadUrl(initialUrl);
  }

<#if addressBar != "none">
  public void onGoClicked(View view) {
    mWebView.loadUrl(mAddressBar.getText().toString());
  }
</#if>

}
```

In der Manifest-Datei findet sich eine kleine Besonderheit bei dem *Label* der Activity. Dieses wird beim Anlegen eines neuen Projektes nämlich mit dem Applikationsnamen anstelle des Titels der Activity belegt. Das erscheint erst logisch, wenn man weiß dass das entsprechende Formularfeld im Projekt-Wizard unterdrückt wird. Außerdem wird der Activity der Intent-Filter zum Laden der Activity beim Applikationsstart hinzugefügt, falls die Activity die Launcher-Eigenschaft aufweist.

```
<manifest xmlns:android=
  "http://schemas.android.com/apk/res/android" >
  <application>
  <activity android:name=".${activityClass}"
    <#if isNewProject>
      android:label="@string/app_name"
    <#else>
      android:label=
        "@string/title_${activityToLayout(activityClass)}"
    </#if>
    <#if isLauncher>
      <intent-filter>
        <action android:name=
          "android.intent.action.MAIN" />
        <category android:name=
          "android.intent.category.LAUNCHER" />
      </intent-filter>
    </#if>
    </activity>
  </application>
  <uses-permission android:name=
    "android.permission.INTERNET"/>
</manifest>
```

Auch die *String*-Ressourcendatei benötigt kleinere Anpassungen. Zuerst wird der Titel der Activity angegeben, falls es sich nicht um ein neues Projekt handelt und im Anschluss die Beschriftungen der Adressleiste definiert, soweit diese erforderlich sind.

```
<resources>
  <#if !isNewProject>
    <string name=
      "title_${activityToLayout(activityClass)}"
      >${activityTitle}</string>
  </#if>
  <#if addressBar != "none">
    <string name="addressBarHint">http://www.</string>
    <string name="go">Go!</string>
  </#if>
</resources>
```

Nachdem die Code-Vorlagen vorbereitet sind, folgt die Definition der Ausführungslogik. Wie bereits angesprochen stehen drei unterschiedliche Kopieroperationen zur Verfügung. Die *from*- und *to*-Attribute geben dabei die Quell- beziehungsweise Zieldateien an, wobei die Angabe des *to*-Attributs optional ist. Im Beispiel-Template werden zuerst die benötigten Bilder abhängig vom Vorhandensein der Adressleiste kopiert. Anschließend werden die *Manifest*-Datei und die *String*-Deklarationen mit den entsprechenden Dateien im vorliegenden Projekt zusammengeführt, woraufhin die erforderliche *Activity*-Klasse und die entsprechende Layout-Datei erstellt werden. Abschließend wird mit der *open*-Anweisung die soeben erstellte Layout-Datei in Eclipse geöffnet.

```
<?xml version="1.0"?>
<recipe>
  <#if addressBar != "none">
    <copy from="res/drawable-hdpi/separator.9.png" />
    <copy from="res/drawable-ldpi/separator.9.png" />
    <copy from="res/drawable-mdpi/separator.9.png" />
    <copy from="res/drawable-xhdpi/separator.9.png" />
  </#if>

  <merge from="AndroidManifest.xml.ftl" />
  <merge from="res/values/strings.xml.ftl" />

  <instantiate
    from="res/layout/activity_webview.xml.ftl"
    to="res/layout/${layoutName}.xml" />

  <instantiate
    from="src/app_package/WebViewActivity.java.ftl"
    to="${srcOut}/${activityClass}.java" />

  <open file="res/layout/${layoutName}.xml" />
</recipe>
```

Damit sind alle notwendigen Arbeiten abgeschlossen. Das Template kann nun mit dem Template-Development-Wizard (*New → Android → Template Development Wizard*) getestet werden (siehe Abbildung 3). Wenn das Template seine Aufgabe zuverlässig erfüllt kann es in das Templates-Verzeichnis verschoben werden, damit es auch in den anderen Android-Wizards zur Verfügung steht.

Abbildung 3: HelloWebView Template

Fazit

Sicherlich handelt es sich bei den Android Templates nicht um das neue Killer-Feature der Android-Plattform. Dennoch denke ich, dass die meisten Beteiligten vom Einsatz der Templates profitieren können. Entwickler bekommen mit dem Template-Mechanismus eine große Auswahl an Best-Practices an die Hand gegeben und ersparen sich eine Vielzahl von Code-Zeilen. Entwicklerteams können eigene Template-Repositories erstellen, in denen Sie häufig wiederkehrende Anwendungskomponenten pflegen und in unterschiedlichen Projekten wiederverwenden können. Analog dazu könnten in größeren Firmen auch Templates an die Corporate Identity angepasst werden, um die Einhaltung dieser Vorgaben in den einzelnen Geschäftsbereichen zu vereinfachen. Die Open-Source-Community kann mit dem Template-Mechanismus Lösungen zu Standardproblemen schneller verbreiten. Und das Android-Entwicklerteam hat sich damit einen effizienten Mechanismus geschaffen, mit dem es die Entwickler zur Einhaltung der Entwicklungsvorgaben und zur Adaption neuer Konzepte anhalten kann.

Referenzen

[1] Gilfelt, Jeff *Android ADT Templates*
https://github.com/jgilfelt/android-adt-templates

[2] FreeMarker project *Overview*
http://freemarker.sourceforge.net/

[3] The Android Open Source Project *ADT Template Format*
https://android.googlesource.com/
platform/sdk/+/refs/heads/master/templates/docs/index.html
in besserer Darstellung auch unter
https://dl.dropbox.com/u/231329/android/templatedocs/index.html

[4] Android Tools Project Site *Preview Channel*
http://tools.android.com/preview-channel

Des Programmierers kleine Vergnügen

Führende Nullen

von Michael Wiedeking

Nein, es geht nicht um Führungskräfte. Es geht darum, herauszufinden, wie viele der höchstwertigsten Bits in einem Wort Null sind. Diese Funktion ist sehr nützlich, wurde schon in dem einen oder anderen Vergnügen benutzt und soll endlich effizient implementiert werden.

Spätestens seit der Septemberausgabe in 2009 [1] wird im kleinen Vergnügen eine Funktion benötigt, die die Anzahl der führenden, höchstwertigsten Bits liefert, die Null sind. Aber bisher wurde an dieser Stelle die Frage nicht beantwortet, wie man diese denn effizient zählen kann.

Wie üblich gibt es einen naiven, effektiven Ansatz, bei dem einfach die führenden Nullen Stück für Stück abgezählt werden. Dazu wird nur das höchstwertigste Bit geprüft und im Erfolgsfall, wenn es Null ist, die nächsten Bits nachgeschoben (wobei « die Shift-Operation ist):

```
int numberOfLeadingZeros(int x) {
  if (x == 0) {
    return 32;
  }
  int n = 0;
  while ((x & 0x80000000) == 0) {
    n++;
    x = x « 1;
  }
  return n;
}
```

Das funktioniert sehr wohl, ist aber erfahrungsgemäß nicht akzeptabel, weil im schlechtesten Fall die Schleife 31 Mal durchlaufen wird. Es gibt eigentlich auch keinen Grund, immer nur ein einziges Bit zu testen – warum also sollen nicht einfach mehrere Bits auf einmal geprüft werden?

Zuerst zu prüfen, ob *x* denn Null ist, ist der einfachste Fall, weil er keine weiteren Schritte erfordert. Ist *x* ungleich Null, dann muss mindestens ein Bit gesetzt sein, und so gilt es die ursprüngliche Frage zu beantworten, wo sich dieses oder das erste der gesetzten Bits befindet.

Gute Erfahrungen lassen sich fast immer machen, wenn man seine Probleme einfach halbiert. Da es gilt, das höchstwertigste gesetzte Bit zu finden, bietet es sich an, zunächst die höchstwertige Hälfte zu prüfen.

```
if ((x & 0xFFFF0000) == 0) { … }
```

Jetzt gibt es zwei Möglichkeiten: Entweder ist darin kein Bit gesetzt, dann hat man es mit 16 führenden Nullen zu tun; Oder darin ist ein gesetztes Bit enthalten, dann gilt es, dieses zu ausfindig zu machen. Betrachtet man spaßeshalber den letzten Fall, so interessiert zunächst wieder nur ob sich das gesetzte Bit in dem ersten oder zweiten Viertel, also in der ersten oder in der zweiten Hälfte dieses Halbworts, befindet.

```
if ((x & 0xFF000000) == 0) { … }
```

Unter der Annahme, dass auch hier wieder ein Bit in der ersten Hälfte gefunden wurde, wird dieses Vorgehen dann solange durch Halbierung fortgesetzt, bis nur noch ein einzelnes Bit getestet werden muss.

Jetzt stellt sich natürlich die Frage, ob es eine Möglichkeit gibt, diese „halbierten“ Tests auch dann verwenden zu können, wenn kein Bit in der jeweiligen höchst-

wertigeren Hälfte gefunden wurde. Wenn sich diese Möglichkeit nämlich nicht findet, dann gibt es einen Bedingungsbaum, der zwar nur eine Tiefe von fünf hat, es müssen aber trotzdem 32 verschiedene Fälle mit unterschiedlichsten Masken unterschieden werden.

Hier gibt es tatsächlich eine schöne Möglichkeit, man muss nur dafür sorgen, dass man sich immer nur um die führende Hälfte kümmern muss. Wurde ein Bit in der jeweils führenden Hälfte gefunden, so wird durch Halbierung nur deren führende Hälfte betrachtet. Wurde kein Bit in der führenden Hälfte gefunden, muss sich ein Bit in der zweiten Hälfte befinden. Schiebt man nun die zweite Hälfte auf die Position der ersten, so kann man den Algorithmus wie gehabt fortsetzten.

```
if ((x & 0xFFFF0000) == 0) {
  n += 16;
  x = x << 16;
}
```

Damit ergibt sich für die Ermittlung der führenden Nullen die folgende Implementierung mit einer binären Suche:

```
int numberOfLeadingZeros(int x) {
  if (x == 0) {
    return 32;
  }
  int n = 0;
  if ((x & 0xFFFF0000) == 0) {
    n += 16;
    x = x << 16;
  }
  if ((x & 0xFF000000) == 0) {
    n += 8;
    x = x << 8;
  }
  if ((x & 0xF0000000) == 0) {
    n += 4;
    x = x << 4;
  }
  if ((x & 0xC0000000) == 0) {
    n += 2;
    x = x << 2;
  }
  if ((x & 0 x x80000000) == 0) {
    n += 1;
  }
  return n;
}
```

Auf die letzte Bedingung kann dabei verzichtet werden, denn ob das höchstwertigste Bit gesetzt ist, lässt sich einfach in eine 1 umwandeln. Indem man das Vorzeichen an die niederwertigste Stelle schiebt erhält man eine Eins, wenn es gesetzt ist, und Null, wenn es nicht gesetzt ist.

Nur wenn das Bit nicht gesetzt ist, dann soll *n* inkrementiert werden. Also schreibt man anstatt der letzten Bedingung (wobei ⋙ der vorzeichenlose Shift-Operator ist, also ausnahmslos Nullen nachgeschoben werden):

```
n = n + 1 − (x >>> 31);
```

Initialisiert man *n* dann auch noch *n* mit Eins, dann kann auf an dieser Stelle sogar auf das Inkrement verzichtet werden.

Die obige Variante hat den Nachteil, dass man es mit großen Konstanten zu tun hat, die eventuell nicht in einem Maschinenbefehl untergebracht werden können und hier ggf. zusätzlich Zeit benötigt wird, um diese zu laden. Das Problem kann gelöst werden, indem die Bits anstatt nach links nach rechts verschoben werden:

```
int numberOfLeadingZeros(int x) {
  if (x == 0) {
    return 32;
  }
  int n = 1;
  if ((x >>> 16) == 0) {
    n += 16;
    x = x << 16;
  }
  if ((x >>> 24) == 0) {
    n += 8;
    x = x << 8;
  }
  if ((x >>> 28) == 0) {
    n += 4;
    x = x << 4;
  }
  if ((x >>> 30) == 0) {
    n += 2;
    x = x << 2;
  }
  return n − (x >>> 31);
}
```

Übrigens kann man anstatt hoch zu zählen auch runter zählen. In diesem Fall prüft man nicht, ob Bits nicht gesetzt sind, sondern ob sie gesetzt sind. Damit muss die Anzahl der führenden Bits zunächst mit 32 initialisiert werden. Ist nun in der oberen Hälfte ein Bit gesetzt, dann kann es höchstens noch 16 führende Nullen geben.

```
y = x >>> 16;
if (y != 0) { … }
```

Ist kein gesetztes Bit vorhanden, dann macht man einfach mit der niederwertigeren Hälfte weiter. Dabei muss

darauf achtgegeben werden, dass durch das Verschieben der alte Wert nur dann verändert wird, wenn ein gesetztes Bit gefunden wurde, denn nur dann muss ja weiterhin auf die obere Hälfte zugegriffen werden.

```
if (y != 0) {
  n -= 16;
  x = y;
}
```

Besondere Beachtung verdient dabei der abschließende Fall. In dem werden zwei Bit betrachtet, aber nur eines wird getestet. Es könnte ja sein, dass wirklich kein einziges Bit gesetzt ist. Ist das Bit gesetzt, so muss *n* zwangsläufig dekrementiert werden. Ist es aber nicht gesetzt, so kann das verbleibende Bit Eins sein, muss es aber nicht notwendigerweise. Durch das Verschieben verbleibt aber nur das letzte, ungetestete Bit in *x*. Das kann also dazu benutzt werden, um noch ein letztes Mal Eins von der Anzahl der Bits abzuziehen, oder eben keinen Abzug mehr zu machen.

```
return n – x;
```

Das Ergebnis sieht dann insgesamt also wie folgt aus:

```
int numberOfLeadingZeros(int x) {
  int n = 32;
  int y;
  y = x >>> 16;
  if (y != 0) {
    n -= 16;
    x = y;
  }
  y = x >>> 8;
  if (y != 0) {
    n -= 8;
    x = y;
  }
  y = x >>> 4;
  if (y != 0) {
    n -= 4;
    x = y;
  }
  y = x >>> 2;
  if (y != 0) {
    n -= 2;
    x = y;
  }
  y = x >>> 1;
  if (y != 0) {
    n -= 1;
    x = y;
  }
  return n – x;
}
```

Wie schon in der additiven Version kann auch hier die letzte Bedingung noch ein bisschen optimiert werden und gegen den folgenden Code ausgetauscht werden:

```
if (y != 0) {
  return n – 2;
}
```

Diese subtraktive Version sieht zwar vom Schema her aus wie die additive, hat aber einen unschätzbaren Vorteil: Alle Konstanten kommen innerhalb eines Blocks nur einmal vor. Wem nämlich diese Art der Aneinanderreihung zu viel Schreibarbeit macht, kann das nun in einer Schleife schreiben:

```
int numberOfLeadingZeros(int x) {
  int n = 32;
  int c = 16;
  do {
    int y = x >>> c;
    if (y != 0) {
      n -= c;
      x = y;
    }
    c = c >>> 1;
  } while (c != 0);
  return n – x;
}
```

Der Preis dafür ist allerdings eine zusätzliche Bedingung pro Halbierung. Das macht vielleicht einen etwas kontraproduktiven Eindruck, denn anstatt ohne Bedingungen dazustehen, sind es plötzlich doppelt so viele. Kann man diese Bedingungen denn nicht ganz wegoptimieren?

Ja, man kann. Aber wie genau das geht, das soll erst beim nächsten Mal gezeigt werden.

Referenzen

[1] Wiedeking, Michael *Des Programmierers kleine Vergnügen – Die allerletzte Null*, KaffeeKlatsch, Jahrgang 2, Nr. 9, Seiten 31 – 33, Bookware, September 2012 http://www.bookware.de/kaffeeklatsch/archiv/KaffeeKlatsch-2009-09.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.

Deutsch für Informatiker

Herbstcampi, -campen, und -campoden

von Michael Wiedeking

Gerade erst ist der Herbstcampus vorbei, da stellt sich auch schon die Frage, wie er sich denn in die Reihe der anderen einreiht. Denn der diesjährige Herbstcampus war der bestbewerteste Campus aller vorherigen … tja von was? Von den *Campussen*? Den *Campi*? Vielleicht den *Campen*? Oder gar den *Campoden*?

Viele Wörter der Deutschen Sprache haben ihren Ursprung im Lateinischen. Einige dieser Wörter haben die Endung *-us*. Das ist weiter nicht verwerflich, hätte uns deren Deklination nicht schon im Lateinunterricht gelegentlich Probleme bereitet. Denn *-us* war da schon nicht gleich *-us*.

Die auf *-us* endenden Wörter zeichnen sich dadurch aus, dass sie sich der „normalen" Bildung von Genitiv und Plural entziehen. Dem Genitiv würde normalerweise ein *-s* nachgestellt werden, aber diese Wörter haben ja bereits ein *-s* am Ende. Das kann man also einfach so lassen und beispielsweise von den Referenten des Herbstcampus reden.

So bleibt nur noch das Problem des korrekten Plurals. Im einfachsten Fall bleibt bei der Pluralbildung die Endung unverändert. Allerdings spricht man diese dann doch anders aus: *Lapsu:s* (statt *Lapsuss*), als ob da – anstelle des Doppelpunkts – noch ein weiteres *u* oder ein Dehnungs-*h* stehen würde.

In allen anderen Fällen, wenn es denn einen Plural überhaupt gibt, muss man zwischen diversen Endungen entscheiden: *-i* (*Boni*), *-en* (*Globen*), *-usse* (*Modusse*), *-era* (*Opera*), *-een* (*Kakteen*) oder *-oden* (*Oktopoden*). Leider gibt es hier keine brauchbare Regel, der man entnehmen könnte, welche der Endungen zu welchem Wort gehört. Die einzige Regel, die es diesbezüglich gibt, ist die, dass wenn die Endung *-usse* lautet, das *-s* immer verdoppelt wird.

der Abusus	Abusus
der Bonus	Boni, Bonusse
der Diskus	Disken, Diskusse
der Duktus	–
der Exitus	–
der Fötus	Föten, Fötusse
der Genus	Genera
der Globus	Globen, Globusse
der Humus	–
der Kasus	Kasus
der Kaktus	Kakteen
der Konus	Konen, Konusse
der Korpus	Korpusse
das Korpus	Korpora
der Krokus	Krokusse, Krokus
der Kubus	Kuben
der Lapsus	Lapsus
der Luxus	–
der Malus	Malus, Malusse
der Modus	Modi
der Nimbus	Nimbusse
der Numerus	Numeri
der Obolus	Obolus, Obolusse
der Oktopus	Oktopoden
der Opus	Opera
der Passus	Passus
der Primus	Primi, Primusse
der Sinus	Sinus, Sinusse
der Sozius	Sozii, Soziusse
der Status	Status
der Torus	Tori
der Turnus	Turnusse
der Usus	–

Der Duden [1] spricht von „*Campus der;* –, – <lat.-engl.-amerik.>: *Gesamtanlage einer Hochschule; Universitätsgelände*". Damit gibt der Duden als Plural *Campus* an. Aber das will nicht so richtig klingen, bei den vielen Herbstcampus, die ich schon besucht habe.[1]

Auch wenn der Herbstcampus dem Wort nach einen Ort darstellt, versteht er sich aber eher als Treffpunkt zum Erfahrungsaustausch. Nun gut, Treffpunkte sind auch Orte, aber im Herbstcampus schwingt ja auch die Zeit mit und eben der Know-how-Transfer. Und deswegen würde ich vorschlagen, Herbstcampusse zu sagen.

Also war der diesjährige Herbstcampus der bestbewerteste aller vergangenen Herbstcampusse. Und darüber haben wir uns sehr gefreut.

Referenzen

[1] *Duden Duden – Das Fremdwörterbuch [CD-ROM]*, 7. Auflage, Dudenverlag, Mannheim, 2001

[2] Stowasser, J. M. et al. *Der kleine Stowasser – Lateinisch-deutsches Schulwörterbuch*, Oldenbourg Schulbuchverlag, 2010

[1] Nicht hilfreicher ist der Eintrag im *Kleinen Stowasser* [2], *campus, i, m., der* – wenn ich mich recht erinnere – damit auch den Plural als *campi* angibt. Aber das ist alles so lange her…

Kaffeesatz

Wer hat's erfunden?

von Michael Wiedeking

Wenn alles so bleibt wie es im Moment ist, dann wird Samsung eine Milliarden Dollar an Apple überweisen, vielleicht noch ein paar hundert Millionen dazu und möglicherweise kein Galaxy mehr in den USA verkaufen [1]. Ich habe den Prozess nicht wirklich verfolgt, kann mich aber des Eindrucks nicht erwehren, dass da irgendetwas nicht mit rechten Dingen zugeht.

Bisher hatte ich immer geglaubt, dass ein Patent dafür da ist, auch „armen" Erfindern die Möglichkeit zu geben, mit einer Erfindung Geld zu verdienen. Darüber hinaus soll es dem Erfinder die Möglichkeit geben, Einfluss darauf zu nehmen, ob und zu welchen Konditionen die Erfindung reproduziert werden darf. Aber was bereits bekannt ist, darf meines Wissens nicht mehr patentiert werden.

So hat wohl auch Samsung in dem Prozess gegen Apple versucht, einen Ausschnitt aus dem Kinofilm *2001 – A Space Odyssey* [2], in dem Stanley Kubrik Regie führte, vorzulegen. Denn in dem wird ein *Tablet* gezeigt. Es sieht auch so aus wie *Tablets* eben auch heute aussehen: Sie sind relativ flach und haben einen Bildschirm von der Größe einer handlichen DIN-A5-Seite.

In dem gleichzeitig erschienen Buch von Arthur C. Clark, das auf dem Drehbuch von ihm und Stanley Kubrick basiert, wird auch die Funktionsweise genauer beschrieben. Es dient nicht nur zur Eingabe von Berichten und Memoranden – und ist damit offensichtlich interaktiv zu gebrauchen – sondern auch zum Lesen von Zeitungen. Interessant ist hierbei, dass man eine Nummer eintippen muss, um an den entsprechenden Artikel zu gelangen, anstatt wie heute üblich, auf das Icon zu tippen.

> *„When he tired of official reports and memoranda and minutes, he would plug in his foolscap-size newspad into the ship's information circuit and scan the latest reports from Earth. One by one he would conjure up the world's major electronic papers…Switching to the display unit's short-term memory, he would hold the front page while he quickly searched the headlines and noted the items that interested him. Each had its own two-digit reference; when he punched that, the postage-stamp-size rectangle would expand until it neatly filled the screen and he could read it with comfort." [3, aus 4]*

Unabhängig davon, was die Gründe für oben genannten Prozess sind, würde meiner Ansicht nach jedes *Tablet*, das diese Arbeitsweise bietet, nicht mehr patentiert werden können. Und auf etwas darufzudrücken, um an dessen Inhalt zu kommen, ist ja bereits mit dem WWW eingeführt worden. Wobei sich natürlich hier die Frage aufdrängt, ob sich Tim Berners-Lee das hat patentieren lassen. Statische „Hyperlinks" verwenden wir allerdings schon, so lange es Enzyklopedien gibt, und die gibt es zum Glück länger als Patente.

Tablets gibt es erst, seit die Technik das zulässt. Ein *Tablet* ist, wie der Name vermuten lässt, rechteckig. Die Ecken dieses Geräts sind mehr oder weniger abgerundet, damit man sich nicht verletzen kann. Das ein *Tablet* interaktiv ist, ergibt sich von ganz alleine, denn sonst wäre es nur eine bedruckte Glasscheibe, die auf die Dauer sehr langweilig würde. Was ich allerdings tatsächlich nicht verstehe, ist die Existenz des mechanischen Knopfs, den es auch auf der „Kopie" gibt. Allerdings haben auch diese unterschiedliche Formen. Aber ich befürchte, dass es um den nicht ging.

Tablets sind übrigens nicht die einzigen Dinge, die schon früher erfunden wurde. Beispielsweise werden in [4] noch zehn andere Erfindungen aufgelistet. Sollte also mal jemand einen Tricorder [5] aus dem Star-Trek-Universum bauen, so bin ich dafür, dass er auch diesen nicht mehr patentieren darf. Das ist aber auch nicht so wichtig, denn es wird dann genügend Möglichkeiten geben, die Technik, die sich in dem Tricorder befindet, zu patentieren; denn dafür sind doch die Patente da, oder?

Referenzen

[1] FTD *Apple weitet Klage gegen Samsung aus*, Financial Times Deutschland, 23.09.2012
http://www.ftd.de/it-medien/it-telekommunikation/:patentstreit-apple-weitet-klage-gegen-samsung-aus/70094542.html

[2] Kubrick, Stanley *2001: A Space Odyssey, 1968*
http://www.imdb.de/title/tt0062622/

[3] Clarke, Arthur C. *2001 A Space Odyssey*
New American Library, 1968, siehe auch
http://en.wikipedia.org/wiki/2001:_A_Space_Odyssey_%28novel%29

[4] Kessler, Sarah *11 Astounding Sci-Fi Predictions That Came True Meshable Tech*, 25. 09. 2010
http://mashable.com/2010/09/25/11-astounding-predictions/

[5] Wikipedia *Tricorder,* http://en.wikipedia.org/wiki/Tricorder

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 User Group Saxony
Java User Group Dresden
http://www.jugsaxouy.de
Kontakt: Herr Torsten Rentsch (torsten@jugsaxony.de)
Herr Falk Hartmann (falk@jugsaxony.de)
Herr Kristian Rink (kristian@jugsaxony.de)

rheinjug e.V.
Java User Group Düsseldorf
Heinrich-Heine-Universität Düsseldorf
Universitätsstr. 1, 40225 Düsseldorf
http://www.rheinjug.de
Kontakt: Herr Heiko Sippel (info@rheinjug.de)

ruhrjug
Java User Group Essen
Glaspavillon Uni-Campus
Universitätsstr. 12, 45127 Essen
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 asc-Dienstleistungs GmbH
Ehrengard-Schramm-Weg 11, 37085 Göttingen
http://www.java.de (office@java.de)

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

JUG Karlsruhe
Java User Group Karlsruhe
Universität Karlsruhe, Gebäude 50.34
Am Fasanengarten 4, 76131 Karlsruhe
http://jug-karlsruhe.de
jug-karlsruhe@gmail.com

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

jugm
Java User Group München
Jupiterweg 8, 85586 Poing
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 Alexandra Specht
(alexandra.specht@jug-n.de)

JUG Ostfalen
Java User Group Ostfalen
(Braunschweig, Wolfsburg, Hannover)
Siekstraße 4, 38444 Wolfsburg
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
Schönaicherstraße 3, 70597 Stuttgart
http://www.jugs.org
Kontakt: Herr Dr. Micheal Paus (mp@jugs.org)
Herr Hagen Stanek (hs@jugs.org)

Schweiz

JUGS
Java User Group Switzerland
Postfach 2322, 8033 Zürich
http://www.jugs.ch (info@jugs.ch)
Kontakt: Frau Ursula Burri

.Net User Groups

Deutschland

.NET User Group OWL
http://www.gedoplan.de/cms/gedoplan/ak/ms_net
c/o GEDOPLAN GmbH
Stieghorster Str. 60, 33605 Bielefeld

.Net User Group Bonn
.NET User Group "Bonn-to-Code.Net"
Langwartweg 101, 53129 Bonn
http://www.bonn-to-code.net (mail@bonn-to-code.net)
Kontakt: Herr Roland Weigelt

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

.net Usergroup Frankfurt
c/o Thomas Sohnrey, Agile IService
Mendelssohnstrasse 80, 60325 Frankfurt
http://www.dotnet-ug-frankfurt.de
Kontakt: Herr Thomas 'Teddy' Sohnrey
(thomas.sohnrey@gmx.de)

.NET DGH
.NET Developers Group Hannover
Landwehrstraße 85, 30519 Hannover
http://www.dotnet-hannover.de
Kontakt: Herr Friedhelm Drecktrah
(friedhelm@drecktrah.de)

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

DNUG-Köln
DotNetUserGroup Köln
Goldammerweg 325, 50829 Köln
http://www.dnug-koeln.de
Kontakt: Herr Albert Weinert (info@der-albert.com)

.Net User Group Leipzig
Brockhausstraße 26, 04229 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
Sachsenstr. 24, 26121 Oldenburg
http://www.dotnet-oldenburg.de
Kontakt: Herr Hilmar Bunjes
(hilmar.bunjes@dotnet-oldenburg.de)
Yvette Teiken (yvette.teiken@dotnet-oldenburg.de)

.NET User Group Paderborn
c/o Net at Work Netzwerksysteme GmbH,
Am Hoppenhof 32, 33104 Paderborn
http://www.dotnet-paderborn.de
(raacke@dotnet-paderborn.de)
Kontakt: Herr Mathias Raacke

.Net Developers Group Stuttgart
Tieto Deutschland GmbH
Mittlerer Pfad 2, 70499 Stuttgart
http://www.devgroup-stuttgart.de
(GroupLeader@devgroup-stuttgart.de)
Kontakt: Frau Catrin Busley

.net Developer-Group Ulm
c/o artiso solutions GmbH
Oberer Wiesenweg 25, 89134 Blaustein
http://www.dotnet-ulm.de
Kontakt: Herr Thomas Schissler (tschissler@artiso.com)

Österreich

.NET Usergroup Rheintal
c/o Computer Studio Kogoj
Josefgasse 11, 6800 Feldkirch
http://usergroups.at/blogs/dotnetusergrouprheintal/default.aspx
Kontakt: Herr Thomas Kogoj (thomas@kogoj.com)

.NET User Group Austria
c/o Global Knowledge Network GmbH,
Gutheil Schoder Gasse 7a, 1101 Wien
http://usergroups.at/blogs/dotnetusergroupaustria/default.aspx
Kontakt: Herr Christian Nagel (ug@christiannagel.com)

Software Craftsmanship Communities

Deutschland

Softwerkskammer – Mehrere regionale Gruppen unter einem Dach, http://www.softwerkskammer.de

Metropolregion Nürnberg
JUG

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

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

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

web & mobile DEVELOPER präsentiert:

Trends · Lösungen · Know-How

IPDC iPhone Developer Conference 2012

22.-23. Oktober 2012
DORMERO Hotel Stuttgart

Apple iOS • Betriebssysteme • Crossplatform • Android

IPDC buchen & MDC gratis besuchen für € 649,–* (statt € 799,–*) Ihr Anmeldecode: IPDC12kk

* alle Preise zzgl. MwSt.

Referenten (u.a.):

Heiko Behrens, Gründer und CEO, BeamApp

Tammo Freese, CEO, FlockOfBirds UG

Dr. Stephan Gillmeier, TeamLead Mobile, HolidayCheck

Felix Lamouroux, CEO, iosphere GmbH

Amin Negm-Awad, Buchautor und Consultant für Objective-C/Cocoa

iphone-developer-conference.de IPDConference

Medienpartner: KaffeeKlatsch

MATHEMA

meet the experts of enterprise infrastructure

Refactoring
Verbesserung von Design und Quellcode bestehender Software,
25. – 26. Oktober 2012, 925,– € (zzgl. 19 % MwSt.)

Persistenz mit Hibernate
OR-Mapping von Java-Objekten
29. – 31. Oktober 2012, 1.315,– € (zzgl. 19 % MwSt.)

Neues in Java 7
Die nächste Java Generation
5. – 6. November 2012, 835,– € (zzgl. 19 % MwSt.)

Web-Anwendungen mit JavaServer Faces (JSF)
Komponentenbasierte Web-Entwicklung mit JavaServer Faces (JSF)
21. – 23. Nov. 2012, 1.180,– € (zzgl. 19 % MwSt.)

Testkonzepte
Software-Tests zur kontinuierlichen Sicherung der Qualität
28. – 30. November 2012, 1.315,– € (zzgl. 19 % MwSt.)

Sicherheitskonzepte unter Java
Mechanismen für Datenschutz und Datensicherheit in Netzwerken
29. – 30. Nov. 2012,
925,– € (zzgl. 19 % MwSt.)

Lesen bildet. Training macht fit.

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

web & mobile DEVELOPER präsentiert:

Trends Lösungen Know-How

MDC Mobile Developer Conference 2012

22.-23. Oktober 2012
DORMERO Hotel Stuttgart

Android • Crossplatform • Betriebssysteme • Apple iOS

Referenten (u.a.):

Markus Junginger, Chief Android, greenrobot

Janusz Leidgens, Plattformmanager Android, Kupferwerk GmbH

Amin Negm-Awad, Buchautor und Consultant für Objective-C/Cocoa

David Tielke, Trainer, david-tielke.de

Lars Vogel, Android Developer, Vogella.com

MDC buchen & IPDC gratis besuchen für € 649,–* (statt € 799,–*) Ihr Anmeldecode: MDC12kk

* alle Preise zzgl. MwSt.

mobile-developer-conference.de MDConference

Medienpartner: KaffeeKlatsch

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

Das Allerletzte

v1.XX - Streichpreise in ROT
v1.XX - Streichpreise in ROT (diesmal wirklich die Streichpreise)
v1.XX - Streichpreise in ROT (diesmal wirklich rot)

Dies ist kein Scherz!
Diese Commit-Kommentare wurden 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 Oktober.

Herbstcampus

Wissenstransfer par excellence

September 2013

in Nürnberg