KaffeeKlatsch

Das Magazin rund um Software-Entwicklung

ISSN 1865-682X

11/2013

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

Wurmkur

Vor 25 Jahren wurde der erste größere Internet-Wurm auf den Weg geschickt. Das ist für sich genommen schon bemerkenswert genug, aber interessant ist die Sache rückblickend eigentlich nur deswegen, weil sie bewusst gemacht hat, dass die Welt der Computer deutlich anfälliger ist, als es zunächst den Anschein hatte.

Spätestens seit dem Erscheinen des Films WARGAMES [1] im Jahr 1983 war bekannt, dass man mit Hilfe eines PCs und eines Modems beliebig viel Schaden anrichten kann. Natürlich hat Hollywood diese Art des Computer-Missbrauchs nicht erfunden. Aber wenn so etwas einmal in Hollywood angekommen ist, bedeutet das, dass dieses Thema in den Medien ausführlich und erfolgreich genug behandelt worden ist. So behandelt der Film auch sehr schön, wie ein Jugendlicher mit Hilfe seines Rechners und eines Modems nicht nur seine Schulnoten manipuliert sondern durch eine ungünstige Aneinanderkettung von Ereignissen fast den dritten Weltkrieg auslöst.

Das nur wenig später im Jahr 1986 der *Computer Fraud* and *Abuse Act* (als eine Erweiterung des *Comprehensive Crime Control Act* von 1984) [2] verabschiedet wurde, hat mit dem Film sicherlich nichts zu tun, demonstriert aber sehr schön, dass schon zu diesem Zeitpunkt den US-Gesetzgebern klar war, wie anfällig Computer für externe Attacken waren. Deshalb wurde durch das Gesetz nicht nur der unerlaubte Zugriff auf nicht-öffentliche und staatliche Computer unter Strafe gestellt, sondern auch alles, was den reibungslosen Ablauf eines Computers direkt oder indirekt in Frage stellen könnte – zumindest, was man bis zu jenem Zeitpunkt für vorstellbar hielt.

Der im November 1988 in die freie Wildbahn entlassene MORRIS-Wurm, der wie viele technischen Errungenschaften den Namen seines Erfinders erhalten hat, zeigte dann auf sehr beeindruckende Art und Weise, wie anfällig vernetzte Computer tatsächlich sind. Der damals 23-jährige ROBERT TAPPAN MORRIS [3] entließ seinen Wurm, wohl wissend, was er leisten würde. Was er allerdings nicht erahnte, war die Schnelligkeit und Effizienz, mit der sich der Wurm verbreitete und so die vernetzten Rechner in noch nie da gewesener Weise derart effizient lahmlegte, dass selbst die normale Tagespresse Wind davon bekam.

Jetzt erst wurde klar, dass Computer zusammen mit einem Netzwerk unglaublich schnell beliebigen Schaden anrichten konnten. So dämmerte auch die Erkenntnis, dass in der Vernetzung ein bisher unerkanntes und ungeahntes kriminelles Potenzial schlummerte. Darüber hinaus zeigten die von MORRIS selbst angestrebten, gescheiterten Versuche, System-Administratoren darüber zu benachrichtigen, wie der Wurm auszuschalten ist, dass eine Infrastruktur fehlte, die diese Aufgabe hätte übernehmen können [4].

So ist es diesem Wurm zu verdanken, dass kurze Zeit später das erste *Computer Emergency Response Team Coordination Center* (CERT/CC) an der CARNEGIE-MELLON-Universität eingerichtet wurde [5]. Seitdem weiß man um die Notwendigkeit, dass solche Attacken stattfinden und wie man diese und Gegenmaßnahmen dokumentiert und kommuniziert. Und so muss man heutzutage nicht mehr wie damals unvorbereitet und ohnmächtig zusehen, wie die eigene Infrastruktur attackiert und lahmgelegt wird.

Also muss man dem Herrn ja eigentlich dankbar sein. Vielleicht ist er ja auch deshalb so glimpflich davon gekommen: 400 Stunden gemeinnützige Arbeit und 10 050 $ Strafe (zuzüglich der Gerichtskosten) – ein Gefängnisaufenthalt blieb ihm glücklicherweise erspart.

Ihr MICHAEL WIEDEKING
Herausgeber

Referenzen

[1] BADHAM, JOHN *WarGames* (1983)
http://www.imdb.com/title/tt0086567

[2] WIKIPEDIA *Computer Fraud and Abuse Act*
http://en.wikipedia.org/wiki/Computer_Fraud_and_Abuse_Act

[3] WIKIPEDIA *Robert Tappan Morris*
http://de.wikipedia.org/wiki/Robert_Tappan_Morris

[4] INTEL FREE PRESS *Interview mit Eugene Spafford: Lessons from the First Major Computer Virus*
http://www.intelfreepress.com/news/lessons-from-the-first-computer-virus-the-morris-worm/7223

[5] Borchers, Detlef *Vertreibung aus dem Paradies*
http://www.heise.de/ix/artikel/Vertreibung-aus-dem-Paradies-1981722.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 11, November 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

Editorial 3

Beitragsinfo 4

Inhalt 5

User Groups 17

Werbung 19

Das Allerletzte 20

Artikel

Vom Duke zum Droid

Lehren eines Android-Anfängers 6

Getaktete Passwörter 10

Kolumnen

Der allerletzte Wächter

Des Programmierers kleine Vergnügen 14

Reise nach Engadin

Deutsch für Informatiker 15

Suchmaschine sei Dank

Kaffeesatz 16

Vom Duke zum Droid

Lehren eines Android-Anfängers 6

von Andreas Heiduk

Wenn man für Android entwickeln möchte, gibt es im Netz genügend gute Tutorials, in denen die APIs und deren Verwendung sehr gut erklärt werden. Einem Umsteiger von JSF, Swing oder SWT wird aber leider nicht verraten, welche „höheren" Design-Prinzipien auf der neuen Plattform effizient sind und auf welche man nicht mehr bauen sollte. In diesem Artikel werden einige dieser „Fallen" an einer Beispiel-App erläutert.

Getaktete Passwörter

..... 10

von Wolfgang Wolf

Trotz neuer Entwicklungen, wie die Auswertung biometrischer Merkmale (Fingerabdruck, Handflächen- oder Irisscan), ist das Passwortverfahren gegenwärtig noch immer die gängigste Authentifizierungsmethode. Aus kryptoanalytischer Sicht hängt die Sicherheit eines Passwortes von dessen Länge und von der Anzahl verfügbarer Zeichen ab. Serverseitig kann das Passwort gesalzen werden, um so die Sicherheit zu erhöhen. Aber auch client-seitig gibt es noch bisher ungenutzte (getaktete) Spielräume zur Verbesserung der Sicherheit.

Der allerletzte Wächter

Des Programmierers kleine Vergnügen 14

von Michael Wiedeking

Wenn der Bedarf einmal besteht, kann eine Instruktion mehr oder weniger erfolgsentscheidend sein. Deshalb ist es gelegentlich sinnvoll auf Bedingungen zu verzichten, wenn man im Gegenzug sicherstellt, dass eine Andere sicher eintritt.

Vom Duke zum Droid

Lehren eines Android-Anfängers

von Andreas Heiduk

Wenn man für Android entwickeln möchte, gibt es im Netz genügend gute Tutorials, in denen die APIs und deren Verwendung sehr gut erklärt werden. Einem Umsteiger von JSF, Swing oder SWT wird aber leider nicht verraten, welche „höheren" Design-Prinzipien auf der neuen Plattform effizient sind und auf welche man nicht mehr bauen sollte. In diesem Artikel werden einige dieser „Fallen" an einer Beispiel-App erläutert.

Für den Herbstcampus 2013 in Nürnberg wurde eine App entwickelt, die das Vortragsprogramm auf Android verfügbar macht. Die App soll vor allem drei Fragen bequem beantworten:

- Welche Vorträge gibt es (heute) zum Thema Java (oder .NET, ...)?
- In welchen Vortrag soll ich als Nächstes gehen?
- Welche Informationen gibt es zu einem dieser Vorträge?

Dazu bietet die App zwei Ansichten an:

- Eine übersichtliche Listen-Ansicht, die entweder die Vorträge zu einem Thema (Bild 1) oder zu einer Uhrzeit (Bild 2) auflistet. Zwischen den Themen (bzw. Uhrzeiten) kann man *swipen* oder sie direkt über die *Action-Bar Tabs* auswählen.
- Durch Auswählen eines Vortrages in der Liste gelangt man in die Detail-Ansicht, in der die Vortragsbeschreibung angezeigt wird. Auch hier kann man durch *Swipen* oder *Action-Bar Tab* zwischen den Vorträgen eines Themas (Bild 3) bzw. der Uhrzeit (Bild 4) wechseln.

Die App navigiert also auf einem Datenmodell und bietet verschiedene Sichten auf dieses Modell an. Dieses Datenmodell liegt als Objektgraph vor, der beim Start der App aus einer XML-Datei eingelesen wird.

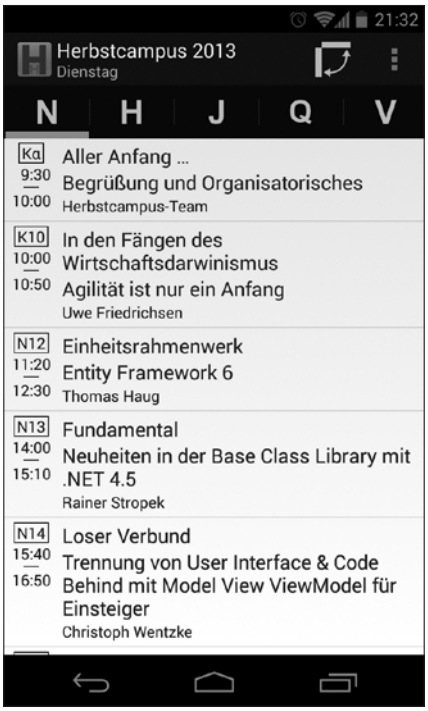

Abbildung 1

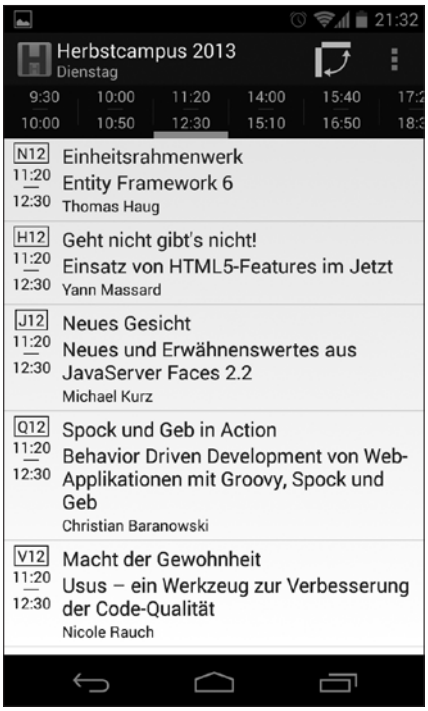

Abbildung 2

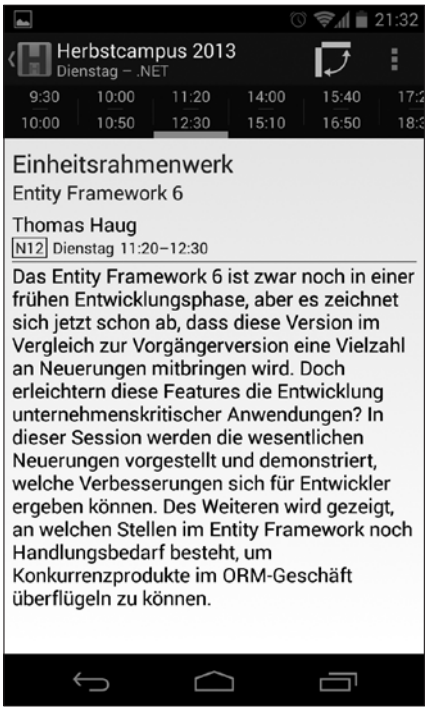

Abbildung 3

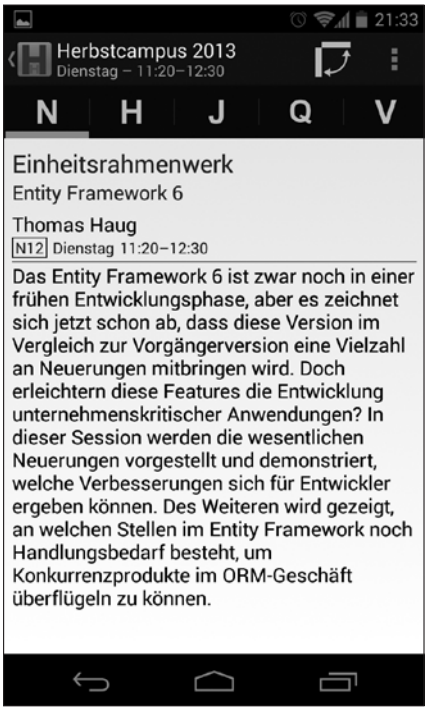

Abbildung 4

Objektreferenzen – Nein Danke!

Wählt man in der Liste einen Vortrag aus, so ruft die erste *Activity*, die Listen-Ansicht, die zweite *Activity* mit der Detail-Ansicht auf und übergibt dabei natürlich als Parameter, welcher Vortrag denn nun im Detail angezeigt werden soll.

Hier beginnt schon das erste Problem: Bei anderen UI-Technologien wie Swing, SWT oder auch bei Web-Anwendungen ist es kein Problem beliebige Daten zwischen den beiden Ansichten zu transferieren – seien es primitive Datentypen oder auch Referenzen auf Objekte in einem Objektgraph.

Bei Android jedoch ist die Übergabe von Objektreferenzen zwischen *Activities* stark eingeschränkt.

Eine *Activity* wird gestartet, indem man zuerst eine Instanz von *Intent* erzeugt und darin festlegt, wer diesen *Intent* behandeln soll. Anschließend kann mit diesem *Intent* eine neue *Activity* gestartet werden. Parameter in dem *Intent* kann man einmal als URI in *data* übergeben oder aber in *extra* über *Key/Value*-Paare. Als *Key* ist ausschließlich *String* erlaubt, als *Value* sind Folgende erlaubt.

- primitive Typen
- String
- Bundle
- Parcelable
- Serializable
- Arrays dieser Typen

Dabei ist ein *Bundle* wieder eine *Key/Value*-Struktur mit denselben Einschränkungen. *Parcelable* dient primär der Interprozesskommunikation (IPC) und erlaubt ebenfalls nur primitive Typen, *Strings*, *Bundles* usw. Zu *Serializable* komme ich gleich noch.

Es gibt also keine Möglichkeit, eine beliebige *Object*-Referenz zu übergeben.

Das ergibt sich implizit daraus, was Android überhaupt mit seinem *Intent/Activity*-Mechanismus erreichen will:

- *Activities* sollen nach Möglichkeit auch von anderen Apps heraus gestartet werden können. Objektreferenzen über Prozessgrenzen hinweg sind halt etwas schwierig zu implementieren.
- Gestartete *Activities* bilden einen Stapel, von dem (in der Regel) nur die oberste *Activity* sichtbar ist. Aus Speicherplatzgründen behält sich Android vor, nicht mehr sichtbare *Activities* und eventuell auch deren Prozesse zu beenden. Werden diese *Activities* wieder sichtbar, muss alles wiederhergestellt werden. Dazu benötigt Android natürlich wieder einen *Intent*, der im Zweifelsfall auch irgendwo persistiert werden muss. Auch hier stören nicht-persistierbare Objektreferenzen ungemein.

Damit geht es wieder zurück zu *Serializable*.

Technisch könnte die Listen-Ansicht der Detail-Ansicht im *Intent* ein serialisierbares Objekt übergeben, in dem jede Klasse des Datenmodells mit *Serializable* markiert wird.

Was aber passiert, wenn Android tatsächlich die Aktivität mangels Speicherplatz beendet und den *Intent* serialisiert? Damit würde das gesamte Datenmodell gleich mitserialisiert werden, da es genügend Querbeziehungen im Objektgraph gibt. Und nach dem Laden des *Intent* und der Deserialisierung würde das Datenmodell eventuell mehrfach im Speicher liegen: Einmal das Modell, das beim Start der Applikation geladen wird sowie ein weiteres Mal für jede wiederhergestellte Aktivität.

Wenn also Speicher knapp ist, dann ist dieses Verfahren nicht geeignet um sparsam damit umzugehen.

Eine gute Lösung ist, die relevanten Klassen des Datenmodells mit einer ID zu versehen, die auch von außen sichtbar ist. Letzteres ist wichtig, wenn die Detail-*Activity* auch von anderen Apps heraus aufgerufen werden soll. Denn von außen ist eine Vortrags-ID „42“ nicht ermittelbar. Das Kürzel „J12“ dagegen steht sogar im offiziellen Vortragsprogramm.

Die Verwendung des Vortragskürzels als Parameter hat aber zwei interessante Nebenwirkungen.

Erstens schlägt das reale Leben wieder mal das Modell: Im gedruckten Vortragsprogramm haben nicht alle Vorträge ein Kürzel. Die Begrüßung des ersten Tages und die Verabschiedung am letzten Tag bekommen daher, fachlich völlig unmotiviert, die Kürzel „Kα“ und „KΩ“. Ebenso benötigen alle Spalten und Zeilen im „Stundenplan“ IDs, denn die Detail-Ansicht muss wissen, über welche Einträge sie *swipen* muss. Hierfür eignen sich die Themenkürzel aus den Tab-Titeln bzw. die Startzeiten der Vorträge.

Zweitens erfordert die Übergabe von IDs letztendlich eine Service-Schicht, die sehr viele „findXByY“-Methoden enthält.

Man sieht, dass schon die kleine Notwendigkeit, Aufrufinformationen von einer *Activity* an die Nächste zu übergeben, stellt in Android einige „harte“ Anforderungen, die in anderen UI-Technologien so einfach nicht existieren.

Weitere Objektreferenzen

Bisher wurde die Übergabe von Parametern von einer *Activity* an die Nächste beleuchtet. Ein weiterer Punkt mit demselben Grundproblem ist „globaler Zustand“.

In beiden Ansichten kann man umschalten, ob die Tabs der *Action-Bar* eine Liste von Themen oder eine Liste von Uhrzeiten anzeigen (Bild 1 und Bild 2). Nun kann der Benutzer Folgendes tun.

- Er befindet sich in der Listen-Ansicht und sieht die Liste der Vorträge zum Thema .NET. („Was gibt's für .NET?“)
- Er wählt den Vortrag um 11:30 Uhr aus und gelangt so in die Detail-Ansicht. („Titel von N12 sieht interessant aus, was sagt die Zusammenfassung?“)
- Er schaltet von „*Swipe* zwischen Vorträgen zum Thema“ auf „*Swipe* zwischen Vorträgen zur selben Uhrzeit“ um. („Neeee, die Zusammenfassung von N12 klingt langweilig, was gibt's denn parallel dazu?“)
- Er *swipt* immer noch in der Detail-Ansicht, zum „Java“-Tab. („Ja, die Zusammenfassung für J12 klingt gut!“)
- Er schaltet nochmal um und
- drückt den *Back*-Button, um wieder in die Listen-Ansicht zu gelangen. („Was gibt es denn neben J12 noch so bei Java?“)

Das Problem: Ohne Vorkehrung landet der Benutzer an der Stelle in der Listen-Ansicht, von der aus er die Detail-Ansicht gestartet hat. In vielen Apps wird das auch der gewünschte Effekt sein. Aber da der Benutzer in der Detail-Ansicht von den „.NET“-Themen zu den „Java“-Themen wechselte, ist die Erwartung, jetzt auch in der Listen-Ansicht die Vorträge zum Thema „Java“ aufgelistet zu bekommen.

Also operieren die beiden *Activities* nicht nur auf demselben fachlichen Datenmodell, dem Vortragsprogramm, auch der Zustand der jeweiligen Ansicht muss abgeglichen werden. Dieser Zustand umfasst:

- ausgewählten Tag der Konferenz
- aktuell sichtbaren Tab
- Anzeige von „Thema“ oder „Uhrzeit“ in der *Action-Bar*

Bei anderen UI-Technologien würde auch hier im einfachsten Fall ein Datenobjekt mit diesen drei Werten definiert und zwischen allen interessierten Ansichten geteilt werden.

Und damit sind wir wieder beim Thema „keine Objektreferenzen zwischen *Activities*“.

Die einfachste Möglichkeit bei Android ist diesen Zustand ebenfalls als extra Parameter zu übergeben. Dieses Verfahren hat ja schon für die fachlichen Parameter funktioniert. Oder ist hier etwas anders?

Ja, es gibt einen großen Unterschied! Zuvor ging es um einen expliziten Übergang, bei dem Parameter übergeben werden. Im Gegensatz dazu muss der Zustand bei allen impliziten und expliziten Übergängen zwischen den *Activities* übertragen werden. Dazu gehören auch Übergänge, die Android selbst und nicht die App veranlasst. Eine nicht vollständige Liste:

- Übergang von Listen-Ansicht zur Detail-Ansicht durch Auswahl eines Vortrages
- Übergang von der Detail-Ansicht zur Listen-Ansicht durch den *Back*-Button
- Übergang von der Detail-Ansicht zur Listen-Ansicht durch den *Home*-Button
- Wiederherstellung einer *Activity* nach Speicherengpass

Abgesehen davon, dass man den Zustand in unterschiedlichen APIs sichern und wiederherstellen muss (*Intent* und *Bundle*), ist es außerdem schwer abschätzbar, ob man denn nun tatsächlich alle Übergänge in allen Varianten bedacht hat.

Die saubere Lösung ist, diesen gemeinsamen Zustand als *Service* anzubieten.

Ein *Service* hat in Android einen eigenen Lebenszyklus, der unabhängig von Benutzerinteraktionen und Sichtbarkeit von *Activities* sein kann. Um im vorliegenden Fall den *Service* zum Austausch des gemeinsamen Zustandes verwenden zu können, muss sich jede *Activity* beim Starten an den *Service* binden. Dies verhindert, dass der *Service* vom System beendet wird. Nachdem das *Binding* erfolgt ist, kann über eine eigene Schnittstelle auf den gemeinsamen Zustand zugegriffen werden – im einfachsten Fall über ein *Interface*, das die Objektreferenz auf das Zustandsobjekt zurück gibt.

Natürlich sollte man den *Service* zu geeigneten Zeitpunkten wieder freigeben, damit Android ihn und die App als Ganzes problemlos beenden kann. So können Speicher, CPU und Akku ein wenig geschont werden. Man muss aber dafür sorgen, dass bei Übergängen zwischen Listen- und Detail-Ansicht zu jeder Zeit mindestens ein *Binding* existiert.

Die Implementierung eines derartigen *Service* und allen notwendigen Codes ist nicht in zehn Zeilen erledigt. Aber dafür entfällt der Code bei der Verpacken-Übergeben-Auspacken-Lösung.

Außerdem gibt es beim *Service*-orientierten Ansatz einen kleinen Bonus: Das persistente Speichern des Zustandes wird einfacher.

Bei der ersten Lösung müsste im Prinzip bei jedem Wechsel der *Activity*, bei jedem *Swipen* und bei jedem Auswählen eines Tabs der Zustand gespeichert werden. Im vorliegenden Fall sind die zu speichernden Datenmengen nicht groß, aber das Speichern würde sehr häufig passieren und zudem die Benutzerinteraktionen leicht verlangsamen.

Der *Service* dagegen hat einen definierten Lebenszyklus und kann zudem nur beendet werden, wenn kein *Binding* durch eine *Activity* mehr existiert. Findet das Speichern des Zustandes also beim Beenden des *Service* statt, geschieht das sehr viel seltener.

Fazit

Die fehlende Möglichkeit Objektreferenzen zwischen den Komponenten einer App auszutauschen hat einige Konsequenzen. Darunter fallen rein interne Punkte wie die *Service*-Schicht mit den „findXByY"-Methoden, aber auch sichtbare und damit eventuell fachlich störende Dinge wie die künstlichen IDs „Kα" und „KΩ".

Es ist kein Problem, diese Punkte frühzeitig im Design zu berücksichtigen. Mit Erfahrung in der Android-Programmierung hat man sie sowieso auf dem Radar. Für die Umsteiger von anderen UI-Technologien gibt es den KaffeeKlatsch.

Kurzbiographie

Andreas Heiduk (andreas.heiduk@mathema.de) ist als Senior Consultant für die MATHEMA Software GmbH tätig. Seine Themenschwerpunkte umfassen die Java Standard Edition (JSE) und die Java Enterprise Edition (JEE). Daneben findet er alle Themen von hardware-naher Programmierung bis hin zu verteilten Anwendungen interessant.

Getaktete Passwörter

von Wolfgang Wolf

Trotz neuer Entwicklungen, wie die Auswertung biometrischer Merkmale (Fingerabdruck, Handflächen- oder Irisscan), ist das Passwortverfahren gegenwärtig noch immer die gängigste Authentifizierungsmethode. Aus kryptoanalytischer Sicht hängt die Sicherheit eines Passwortes von dessen Länge und von der Anzahl verfügbarer Zeichen ab. Server-seitig kann das Passwort gesalzen werden, um so die Sicherheit zu erhöhen. Aber auch client-seitig gibt es noch bisher ungenutzte (getaktete) Spielräume zur Verbesserung der Sicherheit.

Das Passwortverfahren

Das Passwortverfahren identifiziert den Benutzer anhand einer Benutzerkennung und eines persönlichen Passwortes. Das Verfahren basiert auf der Annahme, dass nur die berechtigte Person über das nötige Wissen (PIN oder Passwort) für den eigenen Zugang zum System verfügt. Die Authentizität des Benutzers bleibt daher nur gewahrt, wenn er dieses Wissen geheim hält und potenzielle Angreifer nicht in der Lage sind, an diese Informationen zu kommen [1].

Passwörter im Ziel der Angreifer

Bei einem Angriff auf das Passwort versucht der Angreifer das Passwort des Benutzers zu ermitteln. Sofern der Benutzer bei der Wahl seines Passwortes bestimmte Regeln einhält, bleibt dem Angreifer nur noch die sogenannte *Brute-Force*-Methode, also das systematische Probieren von möglichen Zeichenkombinationen. Dabei ist offensichtlich, dass die Anzahl der nötigen Versuche mit der Passwortlänge als auch mit dem Zeichenvorrat wächst. Mathematisch lässt sich die Anzahl der möglichen Passwörter mittels Formel $N = Z^k$ berechnen, wobei N für die Anzahl der Kombinationsmöglichkeiten steht, Z die Anzahl der verfügbaren Zeichen bedeutet und k die Passwortlänge angibt.

Aus dieser Überlegung leiten viele *Passwort Policies* ihre Empfehlungen für sichere Passwörter ab. Wenn das Passwort nur ausreichend lang und komplex ist, schafft kein Rechner dieser Welt den mathematisch möglichen Suchraum in einer realistischen Zeitspanne zu durchsuchen. Ein *Passwort-Cracker*-Programm müsste immerhin ca. $2{,}04 \times 10^{15}$ Kombinationen testen, um ein achtstelliges Passwort, das aus Buchstaben, Zahlen und zwanzig Sonderzeichen besteht, zu berechnen.

Dennoch ist die Aussicht des Angreifers, selbst bei diesen Bedingungen, nicht hoffnungslos. Die Leistung aktueller Hardware verbessert sich ständig. Für die Berechnung der Passwörter werden inzwischen sogar die sehr performanten Chips auf Grafikkarten eingesetzt [2].

Aber auch Software-seitig gibt es Entwicklungen, welche die Suche nach der Nadel im Heuhaufen beschleunigen. In sogenannten Regenbogentabellen (*Rainbow-Tables*) werden berechnete Passwörter, genau genommen die *Hash*-Werte der Passwörter vorgehalten. Das Suchen in diesen Tabellen geht wesentlich schneller als die Passwörter immer wieder aufs Neue zu berechnen. Begünstigt

wird die Suche in den Tabellen durch den Einsatz schneller Hardware, wie zum Beispiel *SSD*-Festplatten [3].

Schutz gegen diese Angriffe bieten vor allem lange und gute Passwörter. Gut im Sinne von zufälligen Zeichenkombinationen, die sich weder aus Wörterbüchern noch aus anderen logisch nachvollziehbaren Ausdrücken ableiten. Vor allem die Passwortlänge ist entscheidend für die Sicherheit. Im Artikel „Passwörter – fünf Mythen und fünf Versäumnisse“ haben die Autoren Fox und Schäfer bereits auf die Problematik der Sonderzeichen in Passwörtern hingewiesen. Der Artikel zeigt, dass ein zusätzliches Zeichen im Passwort mehr bringen kann als die Verwendung von schwer merkbaren und eingabeunfreundlichen Sonderzeichen [4].

Der aufmerksame und mathematisch geübte Leser hat es eh schon bemerkt: $N = 82^8$ (achtstelliges Passwort bestehend aus Buchstaben, Zahlen und zwanzig Sonderzeichen) ergibt einen erheblich kleineren Suchraum (Faktor 6,6) als $N = 62^9$ (neunstelliges Passwort ohne Sonderzeichen).

Gesalzene Passwörter

Passwörter werden in der Regel nicht auf dem Zugangssystem gespeichert. Vielmehr berechnet die Zugangssoftware aus dem Passwort einen sogenannten *Hash*-Wert und speichert diesen in einer Benutzerdatenbank. Dem *Hash*-Wert kann man die repräsentierten Daten nicht entnehmen, weil diese nur in eine Richtung funktionieren. Damit sind *Hash*-Funktionen zu vergleichen mit mathematischen Einwegfunktionen, komplexitäts-theoretisch leicht zu berechnen aber schwer umzukehren.

Sollte nun ein Angreifer an die gespeicherten *Hash-Werte* gelangen, so kann er daraus nicht ohne Weiteres die Passwörter ableiten. Aber mittels Wörterbücher, Regenbogentabellen und leistungsfähiger Hardware sind seine Chancen gar nicht so schlecht, doch noch die begehrten Geheimnisse zu knacken. Die Herstellerseite RainbowCrack wirbt mit Tabellen mit Wortlängen bis zu acht Zeichen (95 Zeichen auf der Standardtastatur), welche innerhalb von 5 bis 40 Minuten einen Treffer landen. Auch hier bestätigt sich: Je länger ein Passwort ist, desto schlechter die Karten des Angreifers, aber die Schwelle zur Sicherheit nimmt kontinuierlich zu [5].

Eine Methode, um *Hash*-Werte von schwachen (zu kurzen) Passwörtern sicherer abspeichern zu können, nennt sich *Salting* (zu Deutsch: salzen). Bei diesem Verfahren wird das Passwort des Benutzers mit weiteren (zufälligen) Werten erweitert. Aus der Kombination Passwort und Salz wird anschließend der *Hash*-Wert berechnet. Bei einer späteren Passwortprüfung wird die Operation mit dem eingegebenen Passwort erneut durchgeführt. Dazu benötigt die Funktion das eingebrachte Salz und natürlich auch die Information darüber, wie das Salz zur Erweiterung des Passwortes verwendet wurde.

Durch das Salzen erhöht sich der Aufwand für Attacken auf den *Hash*. Eine Wörterbuchattacke wird deshalb scheitern, weil das zufällig gesalzene Passwort in keinem Wörterbuch vorkommt. Auch der Aufwand für im Voraus berechnete Tabellen steigt um den Faktor zwei potenziert mit der Bitbreite des Salzes. Ein zusätzliches Byte Salz ist wesentlich effektiver als ein Zeichen aus dem Passwort, weil das Salz aus dem vollen *ASCII*-Zeichensatz (256 Zeichen) bestehen kann. Die Anzahl der Kombinationsmöglichkeiten für ein achtstelliges Passwort (Buchstaben, Zeichen und Sonderzeichen) erhöht sich mit zwei Bit Salz um Faktor 65.536 auf $3{,}34 \times 10^{20}$.

Das Salzen ist eine wirksame Methode zum Schutz abgespeicherter *Hashes*. Eines kann das Salzen jedoch nicht verhindern: Angreifer oder böswillige Software kann korrekte Anmeldeinformationen ebenso eingeben wie legitime Anwender. Die Sicherheit der Authentifizierung wird durch das *Salting* nicht verbessert, wenn der Angreifer ein gültiges Passwort erspähen konnte [6].

Psylock

Neben dem Passwortverfahren gibt es zur Authentifizierung auch weitere Verfahren, zum Beispiel die Biometrie mit Fingerabdruck, Handvenenerkennung, Iris- und Gesichtserkennung oder Verhaltenserkennung. In der Regel benötigen diese Verfahren zusätzliche Hardware, also Sensoren oder Lesegeräte welche die biometrischen Erkennungsmerkmale der geprüften Person erfassen.

Nicht so *Psylock*. Diese Authentifizierungsmethode identifiziert die Benutzer anhand ihres Tippverhaltens an der PC-Tastatur. Das Verfahren wurde von Prof. Dr. Dieter Bartmann am Lehrstuhl für Bankinformatik der Universität Regensburg entwickelt und wird inzwischen von der Psylock GmbH vertrieben.

Für die Erkennung des Benutzers werden verschiedene Merkmale wie Tippgeschwindigkeit, Tipprhythmus, Korrekturverhalten etc. analysiert und mit einem gespeicherten Muster verglichen. Das Tippprofil des Anwenders muss allerdings vorab als Vergleichsreferenz erfasst sein. Das ist bei *Psylock* vergleichsweise aufwendig und erfordert eine zwei- bis zehnminütige Trainingsphase, in der das System das Typverhalten des Anwenders analysiert. Dafür muss dieser mehrfach wiederholte Tippproben abgeben. Auch ein Auffrischen des Profils kann nach einiger Zeit erforderlich werden [7].

Psylock benötigt keine zusätzliche Hardware, weil das System eine reine Software-Lösung ist. Allerdings muss die Software da vorhanden sein, wo das Tippverhalten analysiert werden soll. Das ist bei client-seitig installierter Software sicherlich kein Problem. Bei der Authentifizierung im Web-Browser wird es schon schwieriger. In der Regel benötigt der Browser dafür ein zusätzliches *Plugin* (*Flash* usw.).

Nimmt man die bisher genannten Faktoren zusammen, ergibt sich folgendes Bild:

- Das Passwortverfahren freut sich hoher Akzeptanz beim Anwender und Systemanbieter, weil es einfach zu handhaben und sehr günstig in der Anwendung ist. Die Sicherheit des Verfahrens hängt sehr von der Passwortqualität ab. Server-seitig kann die Sicherheit durch das Salzen (künstliche Passwortverlängerung) verbessert werden.
- *Psylock* ist eine sehr sichere Authentifizierungsmethode, weil sie über einen effektiven *Replay*-Schutz verfügt. Allerdings ist *Psylock* vergleichsweise schwierig in der Anwendung.

Wie wäre aber ein Verfahren, welches diese Erkenntnisse zusammenfasst und sich die einzelnen Vorteile zu Nutze macht?

Wie das unsichere Passwort mit Tippverhalten gesalzen wird und so zum getakteten Passwort wird, beschreibt folgender Abschnitt.

Getaktete Passwörter

Um das Tippverhalten des Anwenders eindeutig zu identifizieren, analysiert *Psylock* fünfzehn verschiedene Tippmerkmale. Das ist der Grund für den aufwendigen Erkennungsalgorithmus und die lange Trainingsphase mit relativ langen Sätzen. Für die hier vorgestellten getakteten Passwörter ist nur das Erkennungsmerkmal Tipprhythmus relevant. Die Auswertung dieses Musters reduziert sich auf die Messung der Intervalle zwischen zwei Tastenschlägen. Die so ermittelten Daten werden mit einer Stützvektormethode analysiert und in kurze und lange Anschläge klassifiziert. Das sich daraus ergebende Bit-Muster dient als Salz zur client-seitigen Passworterweiterung.

Der Suchraum für den Angreifer ändert sich dadurch zu $N = Z^k \times (2^{k-1} - 2)$. In Beispielzahlen ausgedrückt, werden aus den $2{,}04 \times 10^{15}$ (achtstelliges Passwort aus Buchstaben, Zahlen und Sonderzeichen) nun $2{,}57 \times 10^{17}$ Kombinationsmöglichkeiten, was einer Verbesserung um Faktor 126 entspricht. Selbst durch den Verzicht auf die kritischen Sonderzeichen, verbleibt noch immer ein Gewinn von Faktor 13,4.

Das hier vorgestellte Verfahren erwartet vom Anwender, dass dieser bewusst bei der Passworteingabe einen individuellen Eingaberhythmus anwendet. Das kann er dadurch erreichen, dass er die einzelnen Passwort-Zeichen in kleinen Gruppen eingibt. Am besten funktioniert das, wenn der Anwender sein Passwort sozusagen singt!

Hier ein Beispiel: Wird das Passwort ikaAD764 in Zeichengruppen zusammengefasst, könnte das wie folgt aussehen: ika*A*D*764. Das Sternchen symbolisiert hier eine kleine Tipppause. Die Software analysiert die Pausen zwischen den einzelnen Tastenschlägen und erzeugt daraus das Takt-Bitmuster 0011100 und damit das gesalzene Passwort i0k0a1A1D1764. Hierbei stellt

Suchraum für Passwörter				
Normale Passwörter				
k	Berechnung	Z=B=52	Z=B+R+62	Z=B+R+S=82
7	$N = Z^k$	1,028E+12	3,521E+12	2,492E+13
8	$N = Z^k$	5,345E+13	2,183E+14	2,044E+15
9	$N = Z^k$	2,779E+15	1,353E+16	1,676E+17
10	$N = Z^k$	1,445E+17	8,392E+17	1,374E+19
Getaktete Passwörter				
7	$N = Z^k \times (2^{k-1} - 2)$	6,374E+13	2,183E+14	1,545E+15
8	$N = Z^k \times (2^{k-1} - 2)$	6,735E+15	2,751E+16	2,575E+17
9	$N = Z^k \times (2^{k-1} - 2)$	7,060E+17	3,438E+18	4,257E+19
10	$N = Z^k \times (2^{k-1} - 2)$	7,372E+19	4,280E+20	7,009E+21
N = Kombinationsmöglichkeiten, Z = Zeichenvorrat, *k* = Passwortlänge, B = Klein- und Großbuchstaben, R = Zahlen, S = Sonderzeichen				

die exemplarische Einstreuung des Salzes nur eine mögliche Variante dar. Das Zugangssystem kann beliebig in der Wahl der Taktzeichen als auch im Algorithmus der Einbindung konfiguriert werden. Es spricht auch nichts dagegen, dass sich diese Parameter bei jeder Authentifizierung ändern. Das erhöht zusätzlich die Sicherheit.

Für den Anwender passiert das alles transparent. Er muss lediglich darauf achten, dass er sein Passwort im richtigen Takt eingibt. Kleine Abweichungen werden von der Software kompensiert. Der Algorithmus zum Messen der Eingaben ist einfach und erfordert keine komplexen Berechnungen. Damit kann das Verfahren auch in beliebigen Browsern ohne Zusatzsoftware angewendet werden. Hier reicht eine kleine *JavaScript*-Funktion. Gleiches gilt für Zahlenschlösser, *Key-Pads* usw., also Eingabegeräte mit schwacher Hardware-Ausstattung [8].

Auch bei diesem Verfahren muss das Zugangssystem ein Zugangsprofil für den Anwender anlegen. Eine Trainingsphase ist nicht erforderlich. In mehrfachen Versuchen hat sich gezeigt, dass die einmalig wiederholte Passwort-Eingabe, die standardmäßig auch bei der Profilerfassung im Passwortverfahren angewendet wird, ausreicht. Eins unterscheidet sich jedoch grundlegend: Das Passwort kann nicht mehr per *Copy&Paste* eingefügt werden. Das erschwert einerseits die Angriffe mit sonstigen Schädlingen wie *Trojaner* oder *Keylogger*. Andererseits funktionieren gängige Passwortmanager, die Passwortfelder automatisch befüllen, auch nicht mehr.

Benutzer und Systemanbieter sollten darüber hinaus noch einige Besonderheiten beachten:

- Bei der Passworteingabe sind Korrekturen nicht möglich. Ein Vertippen erfordert die erneute vollständige Eingabe. Das kann der Systemanbieter berücksichtigen, indem er auf das Drücken der Rück-Taste reagiert und die ganze bisherige Eingabe löscht.
- Bei der Ersterfassung des Benutzerprofils oder bei einer Passwortänderung ist eine optische Rückmeldung sinnvoll. Das erleichtert die getaktete Eingabe. Bei der normalen Anmeldung sollte jedoch aus Sicherheitsgründen darauf verzichtet werden.

Fazit

Soziale Netzwerke, Anwendungen und Dienste aus der *Cloud* und die zunehmende Vernetzung in Unternehmen und im privaten Bereich erfordern immer öfter eine Authentifizierung der Benutzer. Dabei ist ein ausgewogener Kompromiss zwischen Sicherheit und Aufwand bei der Anmeldung erstrebenswert.

Das Passwortverfahren ist aus der heutigen IT-Landschaft (noch) nicht wegzudenken. Die zunehmenden Fähigkeiten von Angreifern und deren Hardware zeigen jedoch, dass die Sicherheit dieser Methode, zumindest so wie sie aktuell angewendet wird, abnimmt.

Für den Zugang zu kritischen Bereichen sind neue oder verbesserte Verfahren erforderlich. Die alternativen biometrischen Methoden erfordern allerdings meistens Zusatz-Hardware oder sind komplex in der Anwendung.

Der Einsatz getakteter Passwörter kann das Passwortverfahren spürbar verbessern. Der Zusatzaufwand für den Systemanbieter ist gering. Das Verfahren kann überall da eingesetzt werden, wo auch jetzt schon Passwörter, PINS oder Code-Schlösser verwendet werden. Das schließt auch öffentliche Plätze wie Internet-Kaffees ein.

Demo und Download

Unter http://www.ww-a.de/GetaktetePasswoerter.html können Sie die getakteten Passwörter online testen oder eine Demo-Windows-Anwendung herunterladen.

Referenzen

[1] Beutelspacher, Albrecht; Schwenk, Jörg; Wolfenstetter, Klaus-Dieter (2010) *Moderne Verfahren der Kryptographie*

[2] Arbeiter, Stefan; Deeg, Mathias *Bunte Rechenknechte. Grafikkarten beschleunigen Passwort-Cracker*, c't 6/2009, S. 204 – 206

[3] Heise-Online *Passwortknacker 100 mal schneller durch SSD* http://www.heise.de/security/meldung/Passwortknacker-100-mal-schneller-durch-SSD-949988.html

[4] Fox, Dirk; Schaefer, Frank *Passwörter – fünf Mythen und fünf Versäumnisse*, DuD 7/2009, S. 425-429

[5] RainbowCrack Project http://project-rainbowcrack.com/index.htm

[6] Schiffer, Mathias *Schwache Kennwörter sicher speichern* http://msdn.microsoft.com/de-de/library/bb979406.aspx

[7] Psylock GmbH http://www.information.com/?redir=frame&uid=www5295c81da3dfc3.71649649

[8] Wolf, Wolfgang *Passwort(un)sicherheit in Theorie und Praxis*. Bachelor-Thesis in Wirtschaftsinformatik (2011), FOM Nürnberg

Erschienen im *Podium der Wirtschaft*, Band 27,
ISBN 978-3-930616-77-0

Kurzbiografie

Wolfgang Wolf ist Wirtschaftsinformatiker und Geschäftsführer bei der WW-Anwendungsentwicklung. Er arbeitet als Software-Entwickler und Ausbilder für Fachinformatiker in der Dommel GmbH. In seiner Freizeit betreut er die eLearning-Plattform eEx-online.de.

Des Programmierers kleine Vergnügen

Der allerletzte Wächter

von Michael Wiedeking

Wenn der Bedarf einmal besteht, kann eine Instruktion mehr oder weniger erfolgsentscheidend sein. Deshalb ist es gelegentlich sinnvoll auf Bedingungen zu verzichten, wenn man im Gegenzug sicherstellt, dass eine Andere sicher eintritt.

Will man etwa wissen, ob in einem Array *a* ein ganz bestimmtes Element *e* enthalten ist, so kann man es einfach von vorne bis hinten durchsuchen. Dabei gibt es zwei Abbruchbedingungen: Entweder das gesuchte Element wurde gefunden oder das Ende des Arrays wurde erreicht.

```
for (int i = 0, n = a.length; i < n; i++) {
   if (a[i].equals(e)) {
      return i;
   }
}
return –1;
```

Vielen Entwicklern sieht dies viel zu langsam aus, weswegen sie diese Suche gelegentlich wie folgt umformulieren:

```
int i = 0;
int n = a.length;
while ((i < n) && !a[i].equals(e)) {
   i++;
}
if (i == n) {
   return –1;
}
return i;
```

Wie man es aber auch dreht und wendet, es bedarf immer dieser zwei Tests, um sicherzustellen, dass nur auf definierte Elemente des Arrays zugegriffen wird. Aber Bedingungen haftet der Ruf an, dass sie unbedingt zu vermeiden sind. Das ist natürlich so nicht ganz richtig, aber wenn man sich die eine oder andere Bedingung sparen kann, hat man sicher nichts falsch gemacht.

Wüsste man aber, dass in dem Array ganz sicher das gesuchte Element enthalten ist, so könnte man auf die Bereichsprüfung für das Array verzichten:

```
while (!a[i].equals(e)) {
   i++;
}
```

Damit wird die Performanz natürlich erheblich gesteigert. Allerdings kann man nicht immer damit rechnen, dass ein beliebiges Element auch tatsächlich enthalten ist. Außer natürlich, man hat es selbst hineingesteckt.

Dazu bietet sich die letzte Stelle in dem Array an – vorausgesetzt, dass man das Array beschreiben kann. Dann weiß man sicher, dass das Element zumindest an der letzten Stelle enthalten ist. Man setzt also zuerst das gesuchte Element an die letzte Stelle und durchsucht dann das Array. Ist das gefundene Element das letzte, so muss dann noch das echte letzte Element verglichen werden. Und natürlich darf man bei all dem nicht vergessen das letzte Element wieder zu restaurieren.

```
int last = a.length – 1;
T store = a[last];
a[last] = e;
int i = 0;
while (!a[i].equals(e)) {
   i++;
}
if (i == last) {
   if (!store.equals(e)) {
      i = –1;
   }
}
a[last] = store;
return i;
```

Ein solches *Sentinel* ist deswegen so nützlich, weil es keinen Sonderfall darstellt, sondern einen zuverlässigen Bestandteil, mit dem sich Sonderfälle vermeiden lassen. Diese Idee lässt sich auch auf andere Anwendungsfälle übertragen, die dadurch wesentlich einfacher werden.

Aber das ist eine andere Geschichte und soll ein andermal erzählt werden.

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

Reise nach Engadin

von Michael Wiedeking

Deutsch ist selbstverständlich meine Lieblingssprache. Aber wenn es um das Thema Internationalisierung und Lokalisierung geht, verliert Deutsch ganz klar an Punkten. Und das lässt sich relativ leicht begründen.

Aus Sicht der Informatik ist Sprache – und dabei ist es völlig egal welche – nur eine Sequenz von kodierten Zeichen, die hoffnungsvoll für den jeweiligen Benutzer die richtige Bedeutung haben. Landesspezifische Nachrichten sind im Wesentlichen darauf beschränkt Aufgaben oder Tätigkeiten zu beschreiben, also das, was in Menüs, auf Buttons und anderen Elementen der Oberfläche steht. Darin hat man ja schon seit Jahrzehnten Übung sammeln können und Alles in Allem funktioniert das ganz gut.

Die Bibliotheken für die Internationalisierung unterscheiden sich nicht wirklich und bieten vor Allem Unterstützung für die Formatierung von Zahlen, Beträgen, Zeiten, Datumsangaben und so weiter. Sollen aber auch beliebige Texte mit Parametern versorgt werden, sind die Möglichkeiten doch ein wenig begrenzt, tun sich doch speziell im Deutschen ungeheure Abgründe auf.

Stellt man sich beispielsweise vor, dass man einem Kunden internationalisiert zu seiner gebuchten Reise beglückwünschen will, so gestaltet sich das dann doch nicht ganz so einfach: „Wir freuen uns, dass Sie Ihren Urlaub in Italien verbringen wollen!“ Da die Wahl des Landes typischerweise beliebig ist, muss der Text etwa in der Form „Wir freuen uns, dass Sie Ihren Urlaub in {1} verbringen wollen!“ abgelegt werden, wobei „{1}“ durch einen Text-Parameter – hier das aktuelle Land – ersetzt wird. Das Land selbst muss natürlich auch internationalisiert werden. Also wird man problemlos nach Italien, Deutschland und Frankreich fahren können. Aber nach Mallorca kommt man so nicht.

Internationalisierungen werden also sehr oft Opfer einiger Besonderheiten. Wir sind zwar in Spanien, aber dann doch auf Mallorca. Erschwerend kommt hinzu, dass wir in die Schweiz, nach Österreich und ins Engadin fahren. Natürlich kann man einigen Einfluss auf die Texte nehmen, so dass diese Formen weitestgehend vermieden werden können, aber über kurz oder lang fällt Alles der schieren Masse zum Opfer. „Sie fahren mit dem Zug nach Italien“ hat schon drei Parameter, wenn man in das Land seiner Wahl fahren oder fliegen kann und dabei Verkehrsmittel wie Auto, Zug und Flugzeug zur Verfügung stehen.

Im Zusammenhang mit Zahlen gibt es auch viel Freude, weil die Pluralbildung ebenfalls ihre Überraschungen bereit hält: „keine Dateien“, „eine Datei“ und „viele Dateien“. Neben den Zahlwörtern, die man natürlich auch als Zahlen belassen kann, sind dann auch noch die Pluralformen praktisch unvorhersehbar: Atlas – Atlanten, Haus – Häuser, Laus – Läuse, Fenster – Fenster usw., wobei das leider die Semantik außer Acht lässt: Mutter – Mütter und Mutter – Muttern. Ohne dass hier die Wörter berücksichtigt worden wären, die nur im Singular (Glück) oder nur im Plural (Kosten) vorkommen.

Nicht zuletzt bereiten die zu internationalisierenden Tätigkeiten vielfältige Probleme. Das Geldausgeben, das im Englischen einfach als „You spend money“ formuliert werden kann, hat im Deutschen eine eher abenteuerliche, umschließende Form „Sie geben Geld aus“. Ja und dann gibt es noch unser Artikelproblem, das den Wörtern die exotischsten Genera beschert.

Eigentlich ist es recht interessant, wie weit man es mit ein bisschen Geschick schafft, in Benutzeroberflächen die Eigenheiten der deutschen Sprache weitestgehend zu umschiffen.

Kaffeesatz

Suchmaschine sei Dank

von Michael Wiedeking

Mit der Zeit gewöhnt man sich ja an ziemlich viel. Und oft verinnerlicht man diese Dinge so sehr, dass man sich gar nicht mehr richtig daran erinnern kann, wie es denn früher einmal gewesen ist.

Als ich das Editorial für diese Ausgabe geschrieben habe, war ich mir nicht mehr ganz sicher, ob mich mein Gedächtnis nicht auf hinterhältigste Art und Weise im Stich lässt. Eigentlich war es gar nicht so lange her, dass ich mir den John-Badham-Film *WarGames* [1] angesehen habe, und so meinte ich mich daran zu erinnern, dass der junge Matthew Broderick [2] alias David sich via Modem in seiner Schule einloggt und dann eine seiner Schulnoten verändert.

Dann war ich mir aber plötzlich nicht mehr ganz so sicher, befürchtete ich doch, dass ich das vielleicht mit dem drei Jahre älteren Broderick in *Ferris macht blau* [3] verwechseln könnte, der sich – Ferris Bueller mimend – via Modem in seiner Schule einloggt, um seine Fehltage zu korrigieren. Woraufhin er sich einen legendären freien Schultag leistet (mit der Bewertung 7,8/10), der ihm eine Verfolgung durch seinen Rektor einbringt und ihn quer durch Chicago führt.

Also musste ich mir *WarGames* sicherheitshalber noch einmal ansehen. Mit der Bewertung 7,0/10 immer noch eine lohnende Sache, denn trotz seines Alters ist der Film immer noch sehr sehenswert, wenngleich den Jüngeren unter uns das eine oder andere sehr befremdlich anmuten muss. Dazu gehört natürlich ganz besonders das Computer-Equipment selbst: der Terminal, das Modem, die Benutzeroberfläche und nicht zuletzt die Sprachausgabe (die einmal bei ihm Zuhause eingerichtet, plötzlich im ganzen Film überall zur Verfügung steht).

Was mich aber beim Ansehen des Films besonders beeindruckt hat, war die Suche nach dem Passwort. So probiert er, um vielleicht früher an Informationen über ein neu angekündigtes Computer-Spiel zu kommen, in der Gegend des Herstellers sämtliche Telefonnummern nach Modem-Zugängen ab, wird fündig und braucht nun eine Zugangsberechtigung. Auf Anraten von ihm bekannten, sehr sehenswerten Computer-Nerds versucht er dann eine „Hintertür" zu finden, um in das System zu kommen. Und hat dazu nur den einen Namen: Falken.

Und dann muss er erst einmal raus. Also Suchen war etwas, dass man vor dreißig Jahren nicht von Zuhause aus machen konnte. Er ging in die Bibliothek, um dort den Namen zu recherschieren. Da gab es in kleinen langen Schubladen etwa DIN A7-große, mit Schreibmaschine beschriebene Zettel, auf denen die Schlagwörter oder die Autoren alphabetisch sortiert zu finden waren. Die dort gezeigte Bibliothek verfügte sogar über einen Computer, der die Ergebnisse natürlich noch über einen Nadeldrucker ausgab, auf am Rand gelochtem Endlospapier.

Wenn man dann wusste, wo man das Gesuchte finden konnte, machte man sich auf zu den Regalen, um in der Zeitung zu blättern. Oder man ging in den Lesesaal an die Microfiche-Leser, diese optischen Lesegräte mit den großen „Monitoren", mit denen man DIN A6-große Filmblätter mit Mikroschrift lesen konnte. Und wenn die Bibliothek nicht über einen wie auch immer gearteten Kopierer verfügte, um sündhaft teure Kopien zu machen, dann musste man sich eben Notizen oder Abschriften machen. Sie wissen schon: von Hand, mit Stift und Papier.

Er ist übrigens fündig geworden und hat das Passwort korrekt erraten. Und dann spielt er ein bisschen herum, löst damit fast den dritten Weltkrieg aus, rettet dann aber die Welt und lernt, dass es bei Tic-Tac-Toe und Kriegen keine Gewinner gibt. Und das Mädchen bekommt er obendrein.

Ach, waren das noch Zeiten.

Referenzen

[1] Badham, John *WarGames* (1983)
http://www.imdb.com/title/tt0086567

[2] Wikipedia *Matthew Broderick*
http://de.wikipedia.org/wiki/Matthew_Broderick

[3] Hughes, John *Ferris Bueller's Day Off* (1986)
http://www.imdb.com/title/tt0091042

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

Einführung in die objektorientierte Programmiersprache Java

Eine praxisnahe Einführung
10. – 14. Februar 2014, 30. Juni – 4. Juli 2014,
2.150,– € (zzgl. 19 % MwSt.)

Einführung in C# und .NET

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

Entwicklung mobiler Anwendungen mit iOS

7. – 9. April 2014, 15. – 17. September 2014,
1.250,– € (zzgl. 19 % MwSt.)

HTML5, CSS3 und JavaScript

31. März – 3. April 2014, 22. – 25. September 2014,
1.650,– € (zzgl. 19 % MwSt.)

Fortgeschrittenes Programmieren mit Java

Ausgewählte Pakete der Java Standard Edition (J2SE)
5. – 7. Mai 2014, 13. – 15. Oktober 2014,
1.350,– € (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

„An MATHEMA schätze ich unsere öffentliche Präsenz, z. B. bei User Groups und Konferenzen, sowie die Vielseitigkeit meiner Tätigkeit."

William Siakam, Consultant

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

join the experts of enterprise infrastructure

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

Das Allerletzte

Beim Start der Android-Konsole kam es zu folgender Kommunikation (3. Zeile meine Eingabe)

```
Android Console: type 'help' for a list of commands
OK
help
KO: unknown command, try 'help'
```

Dies ist kein Scherz!
Diese Meldung 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 Dezember.

Herbstcampus

Wissenstransfer par excellence

1. – 4. September 2014

in Nürnberg