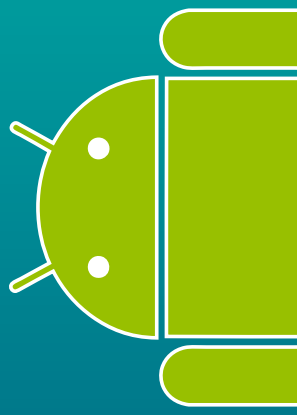
Nr.01 | Frühjahr 2012 | www.ijug.eu

Java aktuell

Das Magazin der Java-Community

Java aktuell

ANDROID in der Praxis



JavaOne 2011
Neuigkeiten und Trends

Oracle Public Cloud
Bereit für Wolke sieben

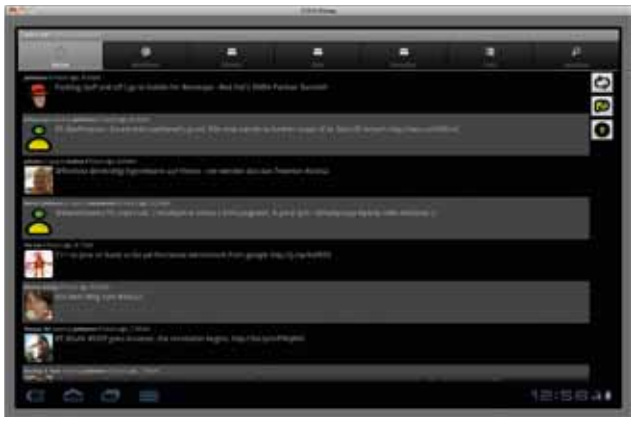
Adobe AIR
Anspruchsvolle Applikationen realisieren

D: 4,90 EUR A: 5,60 EUR CH: 9,80 CHF Benelux: 5,80 EUR ISSN 2191-6977

Sonderdruck

iJUG Verbund

Inhalt



Android-Apps fit für die Zukunft machen, Seite 12

Android-Apps fit für die Zukunft machen

Heiko W. Rupp, Red Hat

Im Februar 2011 hat Google die Version 3 „Honeycomb" speziell für Tablets freigegeben, gefolgt von Version 4 „Ice Cream Sandwich" im Oktober. Während die meisten Applikationen, die für frühere Versionen geschrieben wurden, ohne Probleme weiterlaufen, geben sie dem Benutzter auf den Geräten mit „Honeycomb" oder „Ice Cream Sandwich" ein angestaubtes Gefühl, da sie aktuelle Änderungen im User-Interface nicht reflektieren. Dieser Artikel zeigt, wie man mit relativ wenigen Änderungen eine Applikation, die für Android 2.x geschrieben wurde, auf Android 3+ heben kann.

Der Autor hatte unter Android 2.1 mit der Entwicklung seines Twitter-Clients „Zwitscher" begonnen (siehe https://github.com/pilhuhn/ZwitscherA) und war Feuer und Flamme, als das Xoom dann endlich erhältlich war. Man konnte zwar schon im Emulator testen, aber zum einen war dieser bei der hohen Auflösung eher gemächlich und zum anderen hat man einfach auch kein echtes Gefühl für das Verhalten der Anwendung. So erfolgt beispielsweise das Drücken eines Buttons im Emulator mit dem Mauszeiger, den man genau positionieren kann, während man auf echten Geräten die Icons nicht zu klein machen darf. Hier wurde dann schnell klar, dass eine Überarbeitung der Anwendung notwendig ist, während sie auf dem Telefon mit seinem beschränkten Bildschirmplatz soweit ganz gut aussieht (siehe Abbildungen 1 und 2).

Eine der deutlich sichtbarsten Änderungen in „Honeycomb" ist die ActionBar, die am oberen Rand des Bildschirms einen Platz für Namen und Icon der Applikation, einen möglichen Untertitel und eine Taste für den Aufruf des Menüs besitzt. Ist in der Leiste Platz, so kann Android auch Menü-Aktionen direkt in der ActionBar zugänglich machen.

Eine weitere deutliche Änderung, die sich aus dem obigen ergibt, ist, dass Anwendungen, die noch nicht auf Android 3 portiert wurden, auf Geräten ohne Hardware-Tasten (Tablets wie das Motorola-Xoom) am unteren Rand eine zusätzliche Soft-Taste für das Menü haben (siehe Abbildung 3).

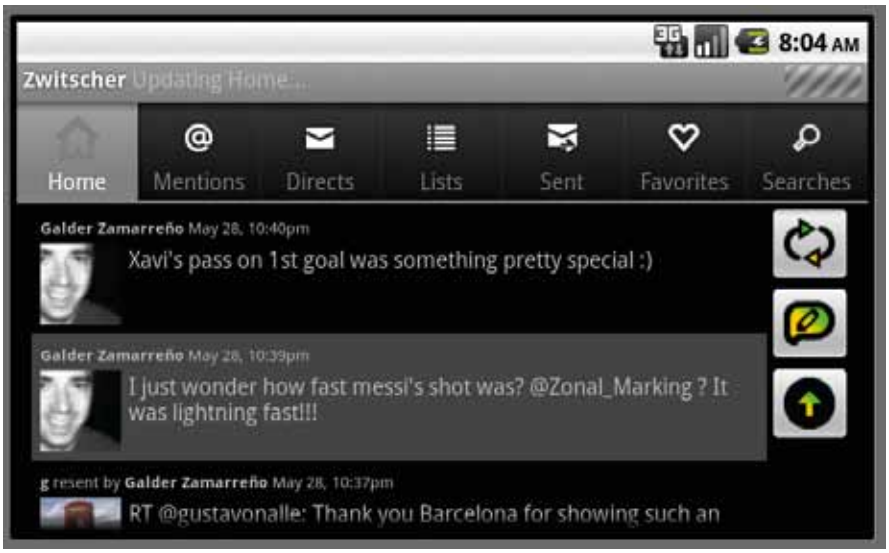


Abbildung 1: Die Anwendung für Android 2 auf dem Mobiltelefon

API-Abfrage

Um die Features von Android 3 oder später nutzen zu können, ist es zunächst notwendig, die SDK-Version im Attribut „targetSdkVersion" in der Datei „AndroidManifest.xml" auf „11" (Honeycomb 3.0) oder größer zu setzen (siehe Listing 1). Die verfügbaren SDK-Versionen stehen in der Tabelle 1 „Android-SDK-Versionen".

```
<manifest
        package="de.bsd.zwitscher">
 ...
    <uses-sdk android:minSdkVersion="8"
            android:targetSdkVersion="11"/>
</manifest>
```

Listing 1

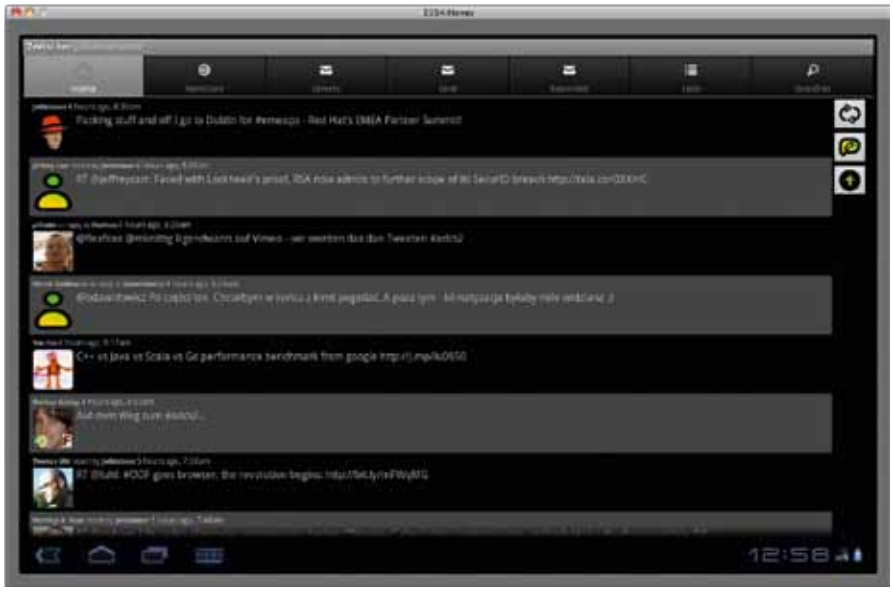


Abbildung 2: Die unveränderte Anwendung auf dem Tablet

Abbildung 3: Die Softkeys zeigen einen extra Knopf für das Menü, wenn die Anwendung nicht auf API-Level 11 oder höher läuft

Abbildung 4: Ausschnitt aus der Menü-Leiste mit Zurück-Indikator, Icon, Titel und Untertitel

Abbildung 5: Ausschnitt der ActionBar mit direkt zugänglichen Menü-Einträgen und dem Menü-Indikator auf „Honeycomb“

Die Änderung der Target-SDK-Version lässt nicht nur den zusätzlichen Taster für das Menü verschwinden, sondern schaltet auch das Handling der neuen Position des Menüs in der ActionBar frei - dazu nachfolgend mehr. Innerhalb des Codes kann die laufende SDK-Version über die Konstante „android.os.Build.VERSION.SDK_INT“ ermittelt werden, sodass Code ab Version 11 die neuen Features nutzen kann, während man für frühere Versionen den alten Code beibehält (siehe Listing 2).

```
if (android.os.Build.VERSION.SDK_INT< 11) {
   // alter Code für vor HC
} else {
  // Code für HC
}
```

Listing 2

Auf den älteren Geräten (ab der im Manifest angegebenen minSdkVersion) laden die Anwendungen wie gewohnt; der Code für „Honeycomb“ wird einfach nicht ausgeführt und gerät damit auch nicht in Probleme, wenn die entsprechenden Methoden nicht vorhanden sind. Mittlerweile ist es möglich, im Markt verschiedene Binarys für unterschiedliche SDK-Versionen einer Applikation einzustellen, sodass man nicht notwendigerweise dies alles in eine Anwendung einkodieren muss; dies war jedoch beim Start von Android 3 nicht möglich. Ein Split der Binary ist hier klarer im Kodieren, hat aber auch den Nachteil, dass man immer mehrere Versionen pflegen muss. Die beschriebene Code-Weiche kann auch genutzt werden, wenn man unterschiedliche Layout-Ressourcen für eine Aktivität je nach API-Level laden möchte.

Auswahl

Nun, da wir die alte Menü-Taste haben verschwinden lassen, ist es Zeit, ein Menü speziell für „Honeycomb“ und später anzulegen. Im Code müssen wir wieder die Android-Version checken, ein Menü aus der XML-Definition (siehe Listing 3) erzeugen und dieses der ActionBar hinzufügen.

Über „actionBar.setDisplayHomeAsUp Enabled(true)“ teilen wir dem System mit, dass der Benutzer durch Druck auf das Home-Icon auf den Schirm davor gelangen kann. Dies wird dann mit einem kleinen Winkel neben dem Programm-Icon angezeigt (Variante ab 3.1; in 3.0 gab es noch ein kleines Dreieck links oben in der Ecke, siehe Abbildung 4).

Abgefragt wird diese „Zurück-Taste“ über den Callback für Menüs:

```
public boolean
onOptionsItemSelected(MenuItem item)
{
 switch (item.getItemId()) {
    case android.R.id.home:
        ....
```

Die im obigen Code-Schnipsel angegebene Menü-Ressource erhält ein neues Attribut, das festlegt, ob der Menü-Eintrag in der ActionBar erscheinen soll oder nicht:

```
android:showAsAction="withText"
```

Mögliche Werte sind dabei:

- withText: es soll nicht nur das Icon angezeigt werden, sondern auch der Alternativtext
- ifRoom: Anzeige nur, falls Platz in der ActionBar ist
- always: immer in der ActionBar anzeigen
- never: soll nie in der ActionBar angezeigt werden

Die einzelnen Werte können dabei mithilfe des Pipe-Symbols kombiniert werden (z.B. ifRoom|withText). Innerhalb der ActionBar werden die Menü-Einträge von links nach

```
public boolean onCreateOptionsMenu(Menu menu) {
  if (android.os.Build.VERSION.SDK_INT>=11) {
          MenuInflater inflater = getMenuInflater();
          inflater.inflate(R.menu.user_detail_menu_honey,menu);

          ActionBar actionBar = this.getActionBar();
          actionBar.setDisplayHomeAsUpEnabled(true);
     return true;
  }
    return false;
}
```

Listing 3

```
<ProgressBar xmlns:android="http://schemas.android.com/apk/res/android"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/ProgressThingy"
    />
```

Listing 4

```
public MyActivity extends Activity {
   ProgressBar pg;

   public boolean onCreateOptionsMenu(Menu menu) {
       if ( Build.VERSION.SDK_INT>=11)) {
          MenuInflater inflater = getMenuInflater();
          inflater.inflate(R.menu.demo,menu);
          pg = (ProgressBar) menu.findItem(R.id.ProgressBar).getActionView();
     }
  }
```

Listing 5

```
    <style name="MyActionBar" parent="android:style/Widget.
Holo.ActionBar">
        <item name="android:background">#446644</item>
    </style>
```

Listing 6

```
    <style name="ZwitscherStyle" parent="android:style/Theme.Holo">
        <item name="android:actionBarStyle">@style/MyActionBar</item>
    </style>
```

Listing 7

rechts in der Reihenfolge, in der sie in der XML-Menü-Definition aufgelistet sind. Abbildung 5 zeigt einen Teil der ActionBar mit zwei direkt zugänglichen Menü-Einträgen und dem Indikator für das Menü.

Eigenen Titel generieren

Viele Anwendungen nutzen mit Android 2 die Titelleiste zur Anzeige von Fortschrittsbalken oder Detail-Informationen zur aktuellen Activity und legen dazu eine „Custom Title Bar" an. Sobald API 11 oder höher angewählt wurde, funktioniert dies nicht mehr und die Applikation crasht, da es in dem Sinne keine Titlebar mehr gibt, sondern eben die ActionBar. Hier muss man wieder die API-Weiche benutzen, um die Custom Title Bar nur für API-Level kleiner als 11 einzusetzen. Für Texte, die angezeigt werden sollen, ist ein Ersatz auch einfach via „ActionBar.setSubtitle(string)" erhältlich. Schwieriger wird es für den Fortschrittsbalken, der so erst mal nicht mehr in der ActionBar existiert. Dies lässt sich allerdings durch einen speziellen Menü-Eintrag lösen:

```
<menu >
 <item android:id="@+id/ProgressBar"
  android:actionLayout="@layout/pg"
  android:visible="true"
  android:showAsAction="always"
 />
```

Listing 4 zeigt die dazugehörige Layout-Ressource. Dies gibt dann statt eines Fortschrittbalkens einen Kreis. Lustig wird es, wenn man die Höhe und Breite unterschiedlich angibt.

Um die ProgressBar sichtbar zu machen, muss man ein wenig Klimmzüge machen und eine Variable anlegen, in die dann beim Anlegen des Menüs als Referenz auf die ProgressBar abgelegt wird (siehe Listing 5).

ActionBar mit Stil

Im gezeigten Fall entsprach die Standard-ActionBar nicht dem Geschmack des Autors und die existierenden, teilweise transparenten Icons sahen auch nicht gut aus, sodass ein wenig Styling notwendig war. Der erste Schritt besteht darin, einen eigenen Stil für die ActionBar festzulegen (in der Datei „res/values/style.xml"), der auf

der Standard-Definition aufbaut (siehe Listing 6).

Dieser Stil kann dann im nächsten Schritt als Teil eines eigenen Stils verwendet werden (siehe Listing 7).

Er kann dann in AndroidManifest.xml als Stil für die Applikation eingebunden werden (siehe Listing 8).

Netzwerken

In Android-Versionen vor „Gingerbread" konnte man bezüglich lang dauernder Aktionen im Prinzip tun und lassen, was man wollte, im Zweifelsfall auch bei den Netzwerk-Zugriffen. Im schlimmsten Fall bekam der Benutzer in einem Dialog zu sehen, dass die Anwendung nicht antwortet, auch als „Application Not Responding" (ANR) bekannt. Android 2.3 führte dann den StrictMode ein, um Aktionen bei solchen potenziell langsamen Operationen auszuführen. Dieser ist in erster Linie eine API, die dem Entwickler helfen soll, langsamen Code zu finden. „Honeycomb" geht hier einen Schritt weiter: Sobald die Anwendung versucht, im UI-Thread Netzwerk-Aufrufe auszuführen, wirft das System eine „NetworkOnMainThreadException". Glücklicherweise lassen sich diese Netzwerkaufrufe via „AsyncTasks" in den Hintergrund verlagern (siehe Listing 9).

Die wesentliche (und einzige benötigte) Methode dabei ist „doInBackground()", die automatisch in einem eigenen Thread ausgeführt wird. Dies ist also der Platz für lange laufende Aktionen und Netzwerk-Zugriffe. Bevor das System diese Methode aufruft, wird im UI-Thread der „onPreExecute()"-Callback aufgerufen, in dem man beispielsweise einen „Bitte-Warten"-Dialog öffnen oder – im Fall von „Honeycomb" und später – den Untertitel in der ActionBar setzen kann. Das Gegenstück ist „onPostExecute()", das nach „doInBackground()" aufgerufen wird und wieder im UI-Thread läuft, um dann den Dialog zu schließen und auch die Ergebnisse des Netzwerk-Zugriffs in der Benutzeroberfläche anzuzeigen.

Fazit

Mit den beschriebenen Änderungen läuft eine Anwendung nativ sowohl auf „Froyo" als auch auf „Honeycomb" und nutzt die jeweiligen Features aus (siehe Abbildung 6).

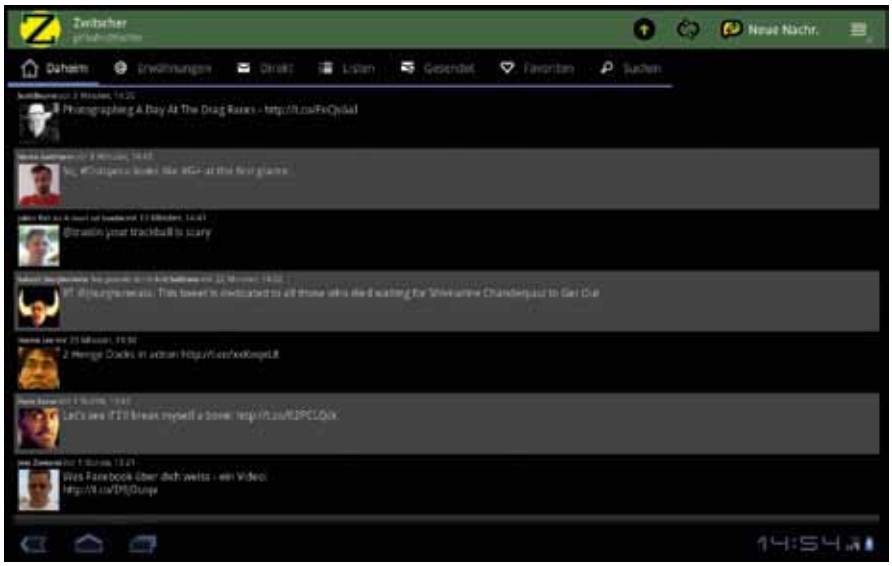


Abbildung 6: Die Anwendung mit den oben beschriebenen Änderungen auf Honeycomb 3.0

```
<manifest ... >
    <application android:icon="@drawable/icon"
                 android:label="@string/app_name"
                 android:theme="@style/ZwitscherStyle" >
```

Listing 8

```
public class MyTask extends AsyncTask<Void,Void,String> {
 Activity context;
 public MyTask(Context context) {
   this.context=context;
 }

 protected String doInBackground(Void... voids) {
    // Netzwerkzugriff
   String x = ....
   return x;
 }

 protected void onPreExecute() {
   context.getActionBar().setSubTitle(„Updating");
 }

 protected void onPostExecute(String message) {
   context.getActionBar().setSubTitle(message);
 }
```

Listing 9

Natürlich hört an dieser Stelle die Arbeit nicht auf. Wie man in Abbildung 6 sieht, wird der Platz auf dem Bildschirm durch diese Änderungen noch nicht optimal ausgenutzt. Hier kann man beispielsweise über Fragmente die Details für einen Tweet, die bei einem Telefon auf einer eigenen Seite gezeigt werden, rechts neben der Liste der Tweets anzeigen. Oder es könnten mehrere Timelines nebeneinander angezeigt werden. Auch die Icons sollte man idealerweise an die neue Bild-

schirmauflösung anpassen, um das Aussehen der Anwendung weiter zu optimieren. Der Zwitscher-Quellcode ist unter https://github.com/pilhuhn/ZwitscherA verfügbar, sodass sich die genannten Beispiele dort nachschauen lassen. Code-Beiträge sind ebenfalls willkommen.

Heiko W. Rupp
heiko.rupp@redhat.com

Heiko Rupp arbeitet bei Red Hat als Entwickler im Bereich System-Management und Monitoring und hier speziell am Jboss-Operations-Network und RHQ-Project.org. Rupp arbeitet und lebt mit seiner Familie in Stuttgart. Seine Hobbys beinhalten Klettern, Lesen und „Vor-dem-Computer-Hocken".

Version	Name	SDK
1.0	Base	1
1.1	Base	2
1.5	Cupcake	3
1.6	Donut	4
2.0	Eclair	5
2.0.1	Eclair	6
2.1	Eclair	7
2.2	Frozen Yoghurt („Froyo")	8
2.3	Gingerbread	9
2.3.3	Gingerbread	10
3.0	Honeycomb	11
3.1	Honeycomb	12
3.2	Honeycomb	13
4.0	Ice Cream Sandwich	14

Tabelle 1: Android-SDK-Versionen

Enterprise JavaBeans 3.1

gelesen von Jürgen Thierack

Mit ihrer geballter Sachkompetenz aus jahrelanger Arbeit mit Java-Komponenten muten die Autoren dem Leser sehr viel zu. Das Buch wendet sich an Leser, die wirklich alles von Grund auf wissen wollen, was es mit jenen standardisierten Komponenten („Beans") auf sich hat, in denen Unternehmen (Enterprises) ihre Geschäftslogik auf Web-Servern codieren. Die Autoren mühen sich um Auflockerung, indem sie öfter mal in den Plauderstil verfallen: „Wenn wir jeden Kaffeeküchen-Besuch mit den gleichen Worten beenden wollten, dann würden die wohl lauten: ‚Im Übrigen sind wir der Meinung, dass jede EJB-Anwendung Unit-getestet werden muss.'" (Seite 348).

Der Untertitel nennt Ein- und Umsteiger. Was ein Einsteiger ist, können wir uns sofort vorstellen. Aber von woher kommt der „Umsteiger"? Er kommt von Version 2 der EJB, könnte also auch „Aufsteiger" heißen. Ein solcher Umsteiger kann Kapitel 1 „Technische Grundlagen und historische Entwicklungen" überspringen und sich stattdessen am Anfang des zweiten Kapitels einen zusammenfassenden Überblick zu den Neuerungen in den Versionen 3.0 und 3.1 (samt Version 2.0 des JPAs) verschaffen.

Wie es sich für ein „Praxisbuch" gehört, werden viele Beispiele (Quellcode per Download) durchgesprochen. Als Plattform dient der kostenlose Open Source Jboss-Application-Server in der Version 6. Die Beispiele kann man manuell per Java-Konsole zum Laufen bringen oder das Build-Tool ANT verwenden. Mit Fragen einer Entwicklungsumgebung beschäftigt sich das Buch nicht, an nur einer Stelle findet die IDE-Eclipse eine kurze Erwähnung. Der Einsteiger kann mit einer EJB-Version von „Hello World" beginnen. Das komplexeste Beispiel ist besonders nett und originell: Das offizielle Begleitspiel „EinStein würfelt nicht!" der Wanderausstellung zum Einstein-Jahr 2005 ist als EJB realisiert. Sehr viele Aspekte der EJB-Entwicklung werden abgedeckt, so kommen alle Bean-Typen zum Einsatz.

Nachdem in Kapitel 11 das Gelernte praktisch erprobt wurde, folgt noch ein abschließendes Kapitel 12, in dem wiederum die Praxis ganz im Vordergrund steht und mit „Kochrezepte" betitelt ist.

Titel:	Enterprise JavaBeans 3.1
Autoren:	W. Eberling, J. Leßner
Verlag:	Hanser München
Umfang:	364 Seiten
Preis:	39,90 €, eBook (downloadbar) inklusive
ISBN:	978-3-446-42259-9

Jürgen Thierack ist in der Software-Branche freiberuflich unterwegs. Seit 10 Jahren liegt sein Schwerpunkt bei Fragen der Finanzmathematik, darunter der Preisfindung für Optionen und Optionsscheine. Aktuelle Thematik: Trendfolgesysteme.

www.ijug.eu

JETZT ABO BESTELLEN

Sichern Sie sich 4 Ausgaben für 18 EUR

Für Oracle-Anwender und Interessierte gibt es das Java aktuell Abonnement auch mit zusätzlich sechs Ausgaben im Jahr der Fachzeitschrift *DOAG News* und vier Ausgaben im Jahr *Business News* zusammen für 75 EUR. Weitere Informationen unter www.doag.org/shop/

FAXEN SIE DAS AUSGEFÜLLTE FORMULAR AN

0700 11 36 24 39

ODER BESTELLEN SIE ONLINE

go.ijug.eu/go/abo

Interessenverbund der Java User Groups e.V.
Tempelhofer Weg 64
12347 Berlin

Java aktuell

+++ AUSFÜLLEN +++ AUSSCHNEIDEN +++ ABSCHICKEN +++ AUSFÜLLEN +++ AUSSCHNEIDEN +++ ABSCHICKEN +++ AUSFÜLLEN

☐ **Ja**, ich bestelle das Abo Java aktuell – das IJUG-Magazin: 4 Ausgaben zu 18 EUR/Jahr

☐ **Ja**, ich bestelle den kostenfreien Newsletter: Java aktuell – der iJUG-Newsletter

ANSCHRIFT

Name, Vorname

Firma

Abteilung

Straße, Hausnummer

PLZ, Ort

GGF. RECHNUNGSANSCHRIFT

Straße, Hausnummer

PLZ, Ort

E-Mail

Telefonnummer

X

Die allgemeinen Geschäftsbedingungen* erkenne ich an, Datum, Unterschrift

*Allgemeine Geschäftsbedingungen:

Zum Preis von 18 Euro (inkl. MwSt.) pro Kalenderjahr erhalten Sie vier Ausgaben der Zeitschrift "Java aktuell - das iJUG-Magazin" direkt nach Erscheinen per Post zugeschickt. Die Abonnementgebühr wird jeweils im Januar für ein Jahr fällig. Sie erhalten eine entsprechende Rechung. Abonnementverträge, die während eines Jahres beginnen, werden mit 4,90 Euro (inkl. MwSt.) je volles Quartal berechnet. Das Abonnement verlängert sich automatisch um ein weiteres Jahr, wenn es nicht bis zum 31. Oktober eines Jahres schriftlich gekündigt wird. Die Wiederrufsfrist beträgt 14 Tage ab Vertragserklärung in Textform ohne Angabe von Gründen.