Oracle ADF macht mobil

Unternehmens-anwendungen mit Oracle ADF Mobile

Volker Linz, Jürgen Menge

Die Anbieter von Unternehmenssoftware stehen immer häufiger vor der Herausforderung, ihre Anwendungen mit zusätzlichen Mehrwertdiensten (z. B. location-based Services) für mobile Endgeräte bereitzustellen. Hier wäre ein Ansatz zielführend, der getreu dem Java-Slogan "Develop once, run everywhere" ein einheitliches Programmiermodell und Toolset für die verschiedenen Mobil-Plattformen vorsieht und idealerweise auch noch die Entwicklung für den traditionellen Desktop einschließt. Das Oracle Application Development Framework (ADF) und speziell die mobile Komponente ADF Mobile gehen dieses Problem an.

Anforderungen an mobile Unternehmensanwendungen

In den Unternehmen stellt sich immer häufiger die Anforderung, Anwendungen auch auf mobilen Endgeräten bereitzustellen. Im Unterschied zu Applikationen (Apps) für den privaten Konsumenten müssen diese jedoch einige spezielle Anforderungen erfüllen:

- In größeren Unternehmen kommen Geräte unterschiedlicher Plattformen zum Einsatz. Neben den traditionell verbreiteten Geräten mit Windows Mobile und Blackberry drängen immer mehr Geräte auf Basis von iOS und Android in die Unternehmen. Dieser Trend zur Vielfalt wird durch das ByoD (Bring your own Device) noch verstärkt. Unternehmens-Applikationen müssen deshalb auf möglichst vielen Plattformen lauffähig sein und einen konsistenten Funktionsumfang bieten. Parallele Entwicklungen für unterschiedliche Plattformen sind kostenintensiv und schwer zu koordinieren. Idealerweise sollte eine mobile Anwendung nur einmal entwickelt und dann auf verschiedenen Plattformen getreu dem Motto "Develop once, run everywhere" ablauffähig sein.
- Auch für mobile Anwendungen ist es erforderlich, sich in die IT-Infrastruktur des Unternehmens zu integrieren. Dazu gehört unter anderem die Anbindung an zentrale Unternehmenssysteme, wie z. B. Datenbanken und LDAP-Verzeichnisse.
- Da nicht in jeder Situation eine Online-Verbindung garantiert werden kann, müssen mobile Anwendungen in der Lage sein, Session-Informationen zwischenzuspeichern und sich mit zentralen Instanzen zu synchronisieren, sobald wieder eine Verbindung besteht.
- Das Thema Sicherheit spielt eine große Rolle, da über die mobilen Endgeräte der Zugriff auf sensible Unternehmensdaten erfolgt. Verlust bzw. Missbrauch der mobilen Geräte stellt eine besondere Herausforderung dar.
- Für die Bereitstellung (Provisionierung) der Applikationen wird eine Distributions-Plattform (App Store) benötigt. Diese kann entweder von einem Anbieter (z. B. Apple Enterprise Distribution) oder innerhalb des Unternehmens betrieben werden.

Entwicklung mobiler Unternehmensanwendungen

Architektur

Eine mobile Anwendung kann hinsichtlich der Architektur in drei Ausprägungen auftreten:

- webbrowser-basierend,
- nativ oder
- hybrid.

Anwendungen, die ausschließlich für den Webbrowser des mobilen Gerätes entwickelt sind, eignen sich für leichtgewichtige mobile Webanwendungen auf den unterschiedlichsten Mobil-Plattformen (iOS, Android, Windows Mobile). Es wird ständig in Echtzeit Zugriff auf die Daten benötigt und clientseitig werden Interaktionen im Browser ausgelöst. Dabei nutzen sie die vorhandene serverseitige Geschäftslogik.

Im Gegensatz dazu benötigt eine rein native, mobile Anwendung zusätzliche Geschäftslogik auf dem Client. Sie ermöglicht aber auch die Offline-Nutzung, ohne permanent auf das Backend-System zugreifen zu müssen. Zum Beispiel bei Technikern, die in Räumen ohne Internetzugang arbeiten und einen Serviceauftrag mit ihrem mobilen Gerät bearbeiten, ist diese Lösung zu bevorzugen. Ein weiterer Vorteil der nativen Variante ist die Möglichkeit, gerätespezifische Dienste, wie z. B. Kamera, Barcode Scanner, Adressbuch, Kalender und andere, innerhalb der Anwendungen zu nutzen.

Um die Vorteile der nativen und webbasierenden Anwendungen zu vereinen, entwickelten sich bereits hybride Ansätze, bei denen unter anderem folgende Technologien zum Einsatz kommen können:

- PhoneGap
- Titanium
- Rhodes Mobile
- JavaScript-Frameworks & HTML5
- iOS oder Android SDK mit „Single Web View Control" oder Webkit

Hybride Clients besitzen eine native Applikationsschicht und integrieren webbasierende Inhalte (Content). Die hybride Anwendung läuft in einem separaten Applikationscontainer auf dem mobilen Endgerät. Die native Applikationsschicht unterstützt drei wesentliche Funktionen:

- gerätespezifische Darstellung, die an das Look & Feel des Gerätes angepasst ist,
- Anzeigen von Webinhalten (Content) innerhalb der Applikation (Web View),
- Integration mit systemspezifischen Gerätediensten, wie z. B. Kamera und Kalender.

Eine Vielzahl an „nativen" Anwendungen im Apple Store oder Android Market sind tatsächlich hybride Anwendungen. Des Weiteren gibt es die Möglichkeit, komplette Anwendungsoberflächen (UIs) mit HTML5 zu gestalten.

Zunehmend gewinnen Open-Source-Frameworks wie PhoneGap an Bedeutung, die die gerätespezifischen Services mit Weboberflächen verbinden und per JavaScript-Funktionen (JS API) aufrufen.

Damit kombinieren hybride Ansätze die Vorteile nativer und webbasierender Anwendungen. Die Integration mit gerätespezifischen Diensten und Offline-Operationen (z. B. Anwendungslogik und Datenspeicherung in einer lokalen Datenbank) werden unterstützt. In der Konsequenz muss komplexe Geschäftslogik in JavaScript implementiert werden.

Abbildung 1 zeigt eine hybride Architektur am Beispiel von Oracle ADF Mobile. Das Besondere an dieser Architektur ist, dass neben der Web View und der gerätespezifischen JavaScript API für *PhoneGap* eine Java Runtime (CDC) im An-

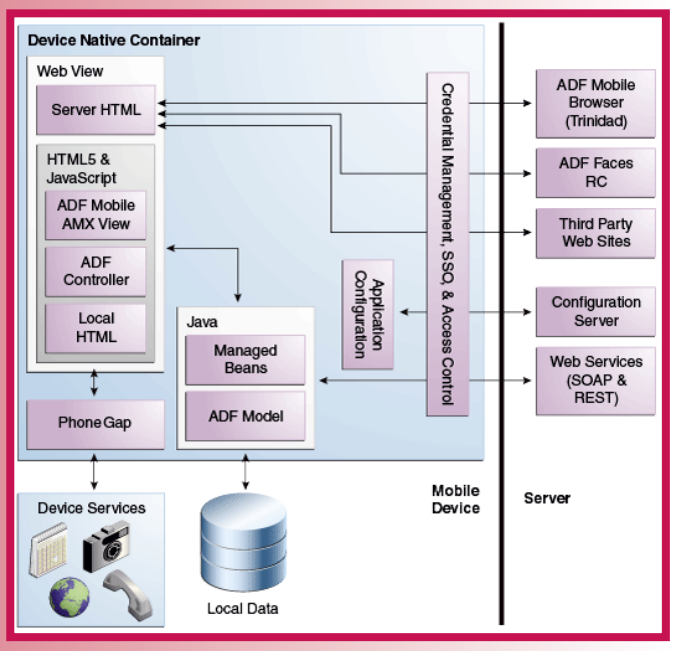

Abb. 1: Hybride Beispielarchitektur von mobilen Anwendungen (Developer Guide Oracle ADF Mobile)

wendungscontainer integriert ist. Dies bietet den Vorteil, dass Java sowohl für die Geschäftslogik der mobilen Anwendung als auch für die serverseitige Logik verwendet wird.

Die hybride Client-Architektur zielt auf die optimale Balance zwischen Funktionalität und Entwicklungsaufwand. Während bei nativen Anwendungen der Entwickler die Sprache der Mobil-Plattform (z. B. Objective-C bei iOS) erst erlernen muss, um hochwertige Anwendungen zu entwickeln, erzielt der Entwickler hybrider Anwendungen mit etablierten, bekannten Standards (HTML5, JavaScript, CSS) schnell das gewünschte Ergebnis.

Entwicklungs-Framework

Für die Unterstützung der verschiedenen Mobil-Plattformen wird ein Entwicklungs-Framework mit der notwendigen Infrastruktur benötigt, das es dem Entwickler ermöglicht, ein einheitliches Programmiermodell zu verwenden. Idealerweise sollte das Entwicklungs-Framework sowohl für Mobil- als auch für Desktop-Anwendungen im Unternehmen eingesetzt werden können.

Das Oracle Application Development Framework (ADF) kombiniert Java EE-Standards und herstellerspezifische Frameworks, um mehrschichtige Unternehmens-Applikationen zu entwickeln. In der aktuellen Version von ADF (11.1.2.3) sind Komponenten zur Entwicklung mobiler Anwendungen hinzugekommen. Eine Besonderheit von ADF besteht in der *Model*-Schicht, einem universellen *Data Binding Layer*, der den Zugriff auf verschiedenartige Implementierungen des *Business Service* (EJB/JPA, Web Service, ADFbc, ...) ermöglicht.

Als Entwicklungsumgebung (IDE) können der Oracle JDeveloper oder das Oracle Enterprise Pack for Eclipse (OEPE) eingesetzt werden.

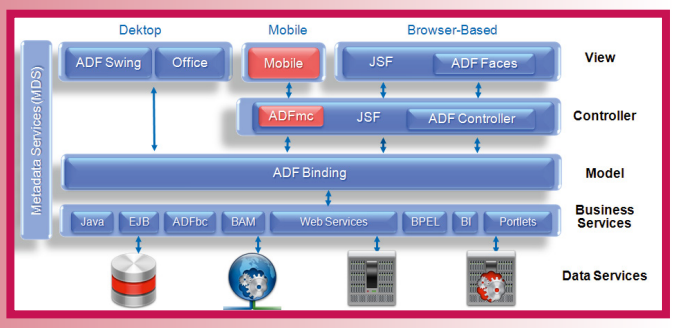

Abb. 2: Architektur des Oracle Application Development Framework (ADF)

Spezifische UI für mobile Anwendungen

Es hat sich gezeigt, dass es in den wenigsten Fällen sinnvoll ist, eine für alle Endgeräte vom Desktop bis zum Smartphone identische Seitengestaltung zu definieren.

In der Konsequenz heißt dies, dass für unterschiedliche Geräteklassen (Desktop, Tablet, Smartphone) spezielle Oberflächen entwickelt werden müssen. ADF Mobile stellt dazu eine Bibliothek mit Oberflächen-Komponenten zur Verfügung (s. Abb. 3). Die Definition der Seiten erfolgt in einer JSF-ähnlichen Syntax, die auf XML basiert (ADF Mobile XML, Listing. 1).

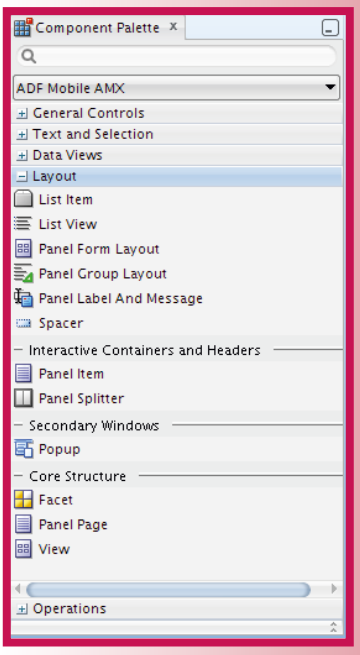

Abb. 3: Komponenten-Palette für Mobile Pages

```
<?xml version="1.0" encoding="UTF-8" ?>
<amx:view xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns:amx="http://xmlns.oracle.com/adf/mf/amx"
                  xmlns:dvtm="http://xmlns.oracle.com/adf/mf/amx/dvt">
  <amx:panelPage id="pp1">
    <amx:facet name="header">
      <amx:outputText id="outputText1" value="Departments"/>
    </amx:facet>
    <amx:listView var="row"
      value="#{bindings.departments.collectionModel}"
      fetchSize="#{bindings.departments.rangeSize}"
      id="listView1">
      <amx:listItem id="listItem1" action="details">
        <amx:outputText value="#{row.deptName}" id="outputText2"/>
        <amx:setPropertyListener to="#{pageFlowScope.DeptNo}"
        from="#{row.rowKey}"/>
      </amx:listItem>
    </amx:listView>
  </amx:panelPage>
</amx:view>
```

Listing 1: ADF Mobile Page (amx)

Der Seitenfluss wird in Form von *Task Flows* (s. Abb. 4) definiert, die aus Page Views, Methodenaufrufen, Routern sowie den Navigationsfällen zwischen diesen bestehen können. Der *Task Flow* enthält die steuernden Anweisungen für den mobilen Controller (*ADFmc*), der als Teil der Anwendung auf dem Endgerät deployed wird.

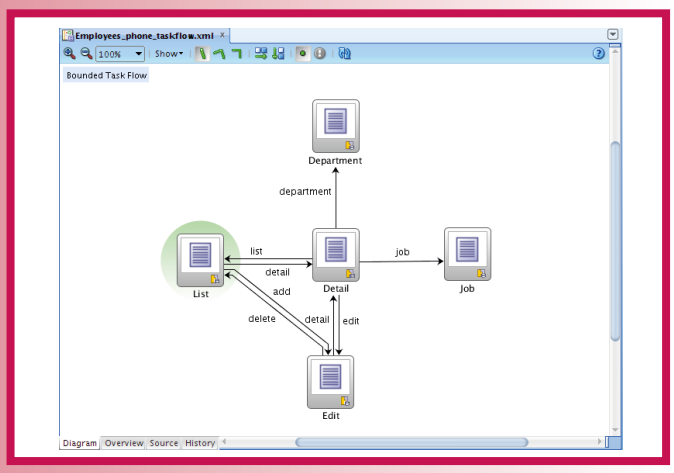

Abb. 4: ADF Mobile Task Flow einer mobilen Anwendung

Datenzugriff und -Synchronisation

Für Unternehmens-Applikationen besteht eine wichtige Frage darin, auf welche Weise der Zugriff auf die Unternehmensdaten erfolgen soll.

Eine Möglichkeit besteht darin, Webservices zu nutzen. Kommt Oracle ADF zum Einsatz, kann für einen SOAP- oder REST-basierten Webservice ein sogenanntes Data Control im JDeveloper generiert werden. Data Controls beschreiben das Public Interface eines Business-Service, in diesem Fall des ausgewählten Webservice. Für die Generierung wird die WSDL des SOAP-Service oder der URL Endpoint des REST-Service benötigt. Das generierte Data Control kann anschließend in einer mobilen Seite verwendet werden, um Attribute oder Methoden des Webservice zu konsumieren. Die Verbindung (Data Binding) wird pro Seite in einer Page Definition-Datei beschrieben, wobei die JSF Expression Language (Beispiel: `#{row.deptName}`) zum Einsatz kommt.

Die Nutzung von Webservices im Backend setzt allerdings voraus, dass eine Verbindung zum Server hergestellt werden kann. Kann dies nicht garantiert werden, besteht die Alternative darin, die Anwendungsdaten lokal auf dem Gerät zu speichern und zu bestimmten Zeitpunkten mit zentralen Datenbanken zu synchronisieren.

Für die lokale Datenhaltung können in ADF Mobile wahlweise SQLite oder die Berkeley DB zum Einsatz kommen. Da die Berkeley DB in der aktuellen Version das SQLite API 3.7.1 implementiert, können für beide Datenbanken die gleichen Funktionsaufrufe (SQL) verwendet werden.

In Oracle ADF Mobile werden die Datenzugriffe in Java Beans mittels JDBC programmiert. Aus diesen Beans können anschließend Data Controls generiert werden, die analog zu den Webservices in den mobilen Seiten zu verwenden sind.

Konkrete Beispiele zur Verwendung von Webservices und lokalen Datenbanken innerhalb der mobilen Anwendung sind für eine spätere Fortsetzung dieses Artikels vorgesehen.

Nutzung nativer Gerätefunktionen

Das Open-Source-Framework PhoneGap (Apache-Projekt Cordova) hat sich durch eine breite Plattformunterstützung bereits in der Praxis bewährt. Externe Applikationen bzw. Webanwendungen, die Zugriff auf die Gerätefunktionalitäten erhalten möchten, können nur mittels des JavaScript APIs von PhoneGap darauf zugreifen.

Mit ADF Mobile besteht für lokale HTML5 bzw. ADF Mobile AMX-Seiten die komfortable Möglichkeit, über ein spezielles Data Control (Device DataControl, s. Abb. 5) auf die von PhoneGap angebotenen nativen Gerätefunktionalitäten zuzugreifen.

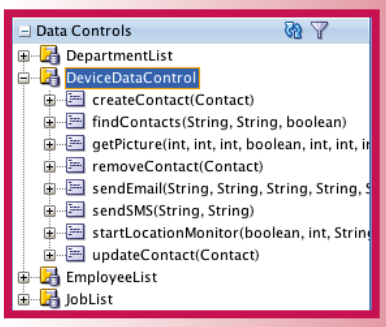

Abb. 5: Data Control für native Gerätefunktionen

Vom Device Data Control werden folgende gerätespezifischen Funktionalitäten unterstützt:

- Geolocation Services (Maps)
- Adressbuch
- Kamera
- Zugriff auf Fotos
- E-Mail
- SMS
- Kontakte
- GPS

Konkrete Code-Beispiele zur Verwendung von Kamera und GPS, zum Versenden von SMS und E-Mails sind ebenfalls für eine spätere Fortsetzung dieses Artikels vorgesehen.

Hybride Applikationen

Wie zeigt sich nun aber der hybride Charakter einer Applikation? Mit ADF Mobile ist es möglich, eine Applikation aus sogenannten Features zusammenzusetzen (*adfmf-feature.xml*, s. Abb. 6), die unterschiedliche Technologien verwenden. Wie in Abbildung 1 zu erkennen, können innerhalb der *Web View* folgende Technologien miteinander kombiniert werden:

- Server HTML (Remote URL): Die Anwendungslogik befindet sich auf einem zentralen Server, der das HTML erzeugt.
- ADF Mobile AMX Pages: AMX-Seiten, die lokal ausgeführt werden und zur Laufzeit HTML5 und JavaScript erzeugen.
- Lokales HTML: HTML-Seiten, die lokal auf dem Gerät ausgeführt werden.
- Native View: Komponenten, die speziell für eine bestimmte Plattform (z. B. in Objektive-C) entwickelt werden und damit nicht plattform-übergreifend zur Verfügung stehen.

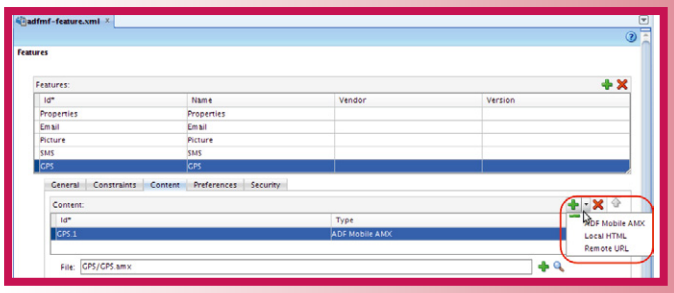

Abb. 6: Komposition einer ADF Mobile-Applikation (adfmf-feature.xml)

Erst zum Zeitpunkt des Deployments wird aus der Applikation eine ausführbare Anwendung erzeugt, indem die plattformspezifischen Artefakte (Deployment-Deskriptoren, Metadaten,

Projekt-Dateien, ...) für die jeweilige Plattform generiert und mit den notwendigen Bibliotheken zu einem Bundle zusammengefasst werden. Dabei werden Werkzeuge aus dem SDK der jeweiligen Zielplattform genutzt.

Sicherheit in ADF Mobile

Zur Authentifizierung bzw. zur Autorisierung des Endanwenders für einzelne Features (kleinste abzusichernde Einheit in ADF Mobile-Applikationen) werden JavaScript, ein *PhoneGap*-Plug-in und verschiedene native *PhoneGap*-Command-Handler genutzt. Die genannten Komponenten kümmern sich um:

- die Interaktion zwischen Login-Seite und Teilen der Applikation (Features),
- die Navigation zwischen den einzelnen Application-Features einer mobilen Anwendung und
- die Interaktion zwischen Oracle Identity Connect IDM Mobile SDK, Oracle Access Manager (OAM) und Applikation.

Die Klassen dieser Security API stellen die Kommunikation und die Verifikation der Nutzer-Credentials sicher. Es gibt spezifische clientseitige Security-Services, um auch einen gesicherten Offline-Betrieb zu gewährleisten. Die Credentials sind dann abgesichert auf dem Gerät abgelegt.

Die Login-Seite wird vom Framework bereitgestellt oder ist nach Kundenbedürfnissen individuell erstellbar. Der komplette Login-Prozess wird nativ, d. h. ohne eingebettetes Java, im ADF Mobile Framework ausgeführt.

Das Login ist notwendig, wenn das Attribut *credential* für ein Application-Feature in der Konfigurationsdatei *adfmf-feature.xml* gesetzt bzw. das Timeout des gesicherten Features überschritten worden ist. In der genannten Datei wird auch festgelegt, ob die Authentifizierung lokal oder remote erfolgt.

Bei Verwendung einer eigenen Login-Seite wird in der *adfmf-application.xml* ein zusätzlicher Tag eingefügt:

```
<adfmf:login defaultConnRefId="LoginConnection">
  <adfmf:localHTML url="newlogin.html"/>
</adfmf:login>
```

Der Eintrittspunkt für den Authentifizierungsprozess eines Application-Features beginnt mit dem Event *activateLifecycle*.

Die Implementierung der Sicherheit wird anhand eines konkreten Beispiels in einer späteren Fortsetzung dieses Artikels dargestellt.

Vorgehensweise

Oracle ADF Mobile zielt auf den Bereich von Unternehmensanwendungen, die auf mobilen Endgeräten sowohl im Online- als auch Offline-Modus bereitgestellt werden sollen. In vielen Fällen wird es sich um Erweiterungen bestehender Applikationssysteme handeln, bei denen bestimmte Funktionalitäten zusätzlich auf mobilen Endgeräten zur Verfügung gestellt werden sollen. Ein praktisches Beispiel wäre die Erweiterung des unternehmensweiten CRM-Systems um mobile Komponenten für den Kundendienst.

So könnte beispielsweise der Service-Manager sich über das Smartphone informieren, ob neue Serviceaufträge eingegangen sind, und diese nach bestimmten Kriterien, wie z. B. Expertise, Verfügbarkeit und Entfernung zum Einsatzort, den Mitarbeitern seines Teams zuweisen. Bei dieser Entscheidung kann der Manager durch die Anzeige der aktuellen Position seiner Mitarbeiter in einer Karte unterstützt werden.

Nach der Auswahl eines Technikers erhält dieser eine Nachricht auf seinem Smartphone mit allen notwendigen Informationen zum Service Request. Über die mobile Applikation kann er sich über die Details des Service Request informieren, die Daten im Verlaufe der Bearbeitung aktualisieren und den Service Request schließen.

Der Manager wiederum kann sich jederzeit über den aktuellen Bearbeitungsstatus der Service Requests informieren und statistische Auswertungen (mobile BI) anfordern.

Die mobile Unternehmensanwendung würde aus verschiedenen Modulen bestehen:

- Kundendatenverwaltung,
- Auftragsbearbeitung,
- Beschaffung von Ersatzteilen
- Terminplaner,
- E-Mails,
- Zugriff auf eigene Wissensdatenbank,
- Weiterleiten von Service Requests an verschiedene Hersteller.

Um praktische Erfahrungen mit dem Framework Oracle ADF und speziell mit der mobilen Komponente ADF Mobile zu sammeln, empfiehlt es sich, zunächst mit einem überschaubaren Pilotprojekt zu beginnen. Wenn die mobile Anwendung eine hohe Sichtbarkeit nach außen hat (zum Beispiel Konferenzplaner, Kantinenanwendung), so kann dies für die Motivation aller Beteiligten von Vorteil sein. Mit ADF Mobile werden einige kleinere Beispiel-Anwendungen ausgeliefert, die eine Hilfestellung geben.

Es ist geplant, ausgewählte Aspekte der Entwicklung anhand eines Anwendungsszenarios in einer späteren Fortsetzung des Artikels zu vertiefen.

Links

[ADFM] ADF Mobile im Oracle Technet, `http://www.oracle.com/technetwork/developer-tools/adf/overview/adf-mobile-096323.html`

[EDA] Enterprise Distribution Apps (iOS), `http://developer.apple.com/library/ios/#featuredarticles/FA_Wireless_Enterprise_App_Distribution/Introduction/Introduction.html`

[OWP10a] White Paper – Oracle Berkeley DB SQL API vs. SQLite API – A Technical Evaluation, 2010, `http://www.oracle.com/technetwork/database/berkeleydb/learnmore/bdbvssqlite-wp-186779.pdf`

[OWP10b] White Paper – Oracle Berkeley DB SQL API vs. SQLite API – Integration, Benefits and Differences, 2010, `http://www.oracle.com/technetwork/database/berkeleydb/bdb-sqlite-comparison-wp-176431.pdf`

Volker Linz arbeitet bei der Oracle Deutschland B.V. & Co. KG in Potsdam und engagiert sich bei der deutschen Oracle ADF Community. Neben der Java Enterprise Anwendungsentwicklung (Java EE 5/6, ADF, JavaFX) und dem Management der Applikationsserverinfrastruktur (WebLogic Server, Enterprise Manager) beschäftigt er sich mit der Social-Business-Plattform Oracle WebCenter.
E-Mail: volker.linz@oracle.com

Dr. Jürgen Menge ist Systemberater für Oracle Fusion Middleware bei der Oracle Deutschland B.V. & Co. KG in München und engagiert sich bei der DOAG (Deutsche ORACLE-Anwendungsgruppe e. V.). Sein fachlicher Schwerpunkt liegt auf Entwicklungswerkzeugen, sowohl auf klassischen (Oracle Forms, Reports, Designer) als auch auf neuen, standardbasierten Werkzeugen (JDeveloper/ADF, BI Publisher, ...).
E-Mail: juergen.menge@oracle.com