Brown Field Entwicklung mit ADF – Evolution einer Forms-Anwendung

Andreas Leidner
infoteam GmbH Berlin
Berlin

Schlüsselworte
Oracle Forms, ADF, Integration, Migration, Strategie, Anwendungsdesign

1. Einleitung

Der Vortrag zeigt die von der infoteam GmbH Berlin verfolgte Strategie für ihr Produkt RUBIN, einer umfassenden, hochintegrierten Organisationslösung für die Messe-, Kongress- und Veranstaltungsbranche. Das Produkt besteht im Kern aus einer sehr großen, über 14 Jahre gereiften Oracle Forms-Anwendung, die durch die Forms-Versionen 4.5, 6i, 10gR2 und 11g gegangen ist.
Die Strategie sieht vor, dass neue Funktionen mit ADF 11g entwickelt werden, sofern sie von Features des ADF und ADF Faces Framework profitieren. Um das bestehende Investment in Oracle Forms zu schützen, werden diese Funktionen zusammen mit der Forms-Anwendung integriert. Ziel ist es, dem Anwender eine einzige an der Benutzeroberfläche visuell integrierte Anwendung zu präsentieren, so dass weder eine komplette, kosten- und zeitintensive Neuentwicklung mit ADF noch eine aufwändige „Big Bang"-Migration von Forms nach ADF notwendig sind.

2. Motivation

ADF bietet mittlerweile, insbesondere mit ADF Faces 11g, eine Fülle an Funktionen, die für den Anwender sehr ansprechend präsentiert werden. Dabei erscheint die Produktivität des Entwicklers vergleichbar mit der Entwicklung in Oracle Forms, so dass nun einige wesentliche Voraussetzungen gegeben sind, um in der Anwendungsentwicklung über den Forms & Reports-Tellerrand hinweg zu blicken:

- Mit ADF 11g gestaltete Benutzeroberflächen können in vielen Aspekten mit traditionellen „Fat Clients" nicht nur mithalten, sondern sehen dabei deutlich moderner und ansprechender aus.
- Das von Forms gewohnte schnelle Prototyping von neuen Funktionalitäten sowie die zügige Umsetzung einer fertigen Lösung ist mit ADF 11g möglich.
- Oracle selbst bietet damit eine Alternative zu Forms, so dass das in einem Entwicklerteam erarbeitete Oracle Know-How weiter genutzt werden kann.

3. Strategie

3.1 Einsatz von ADF in einem bestehenden Anwendungsumfeld

Die Kernbestandteile des Produkts RUBIN der infoteam GmbH Berlin basieren überwiegend auf Oracle Forms & Reports und umfassen dabei insgesamt ein paar Tausend Module (Bibliotheken, Masken, Berichte sowie Datenbank-Packages). Dies reicht von einfachen Masken zur Stammdatenpflege bis zu Masken zur Verwaltung von Adressen, Erfassung von Aufträgen und Durchführung von Fakturen. Letztere Module können sehr komplex sein, da RUBIN als Standardsoftware die unterschiedlichsten Kundenwünsche abbildet und die entsprechenden Funktionen oder Anpassungen je nach Customizing in den Masken zur Laufzeit geschaltet werden. Damit ist die komplette Neuentwicklung des Forms-basierten Produkts mit ADF nicht die erste Option. Stattdessen sollen die bestehenden Investitionen geschützt und das existierende Produkt um

ADF-basierte Funktionen erweitert werden – getreu dem Oracle Motto zur Forms-Strategie, *Upgrade, Integrate and Evolve*:

1. *Upgrade*: Für die Forms-basierte Anwendung wird ein Upgrade auf die jeweils neueste verfügbare Oracle Forms-Version durchgeführt, derzeit 11.1.1.4.0. Nach der bedingt durch die Umstellung auf die 3-Schicht-Architektur sehr aufwändigen Migration von 6i auf 10g fällt dies von 10g auf 11g wieder deutlich leichter. Zudem bringt Forms 11g als neues Feature *External Events* mit, was die im folgenden beschriebene Integration deutlich erleichtert.
2. *Integrate*: Die Forms-Anwendung soll mit den neuen ADF-Funktionen so integriert werden, dass der Anwender weiterhin mit einer Anwendung arbeiten kann. Einen Weg dahin zeigt dieser Text auf.
3. *Evolve*: Wenn sich ADF im täglichen, produktiven Einsatz bei Kunden sowie in der Entwicklung mit dem vollständigen Entwicklerteam bewährt hat, werden neue Funktionen, die von den umfangreicheren Möglichkeiten von ADF profitieren, mehrheitlich mit ADF entwickelt werden. Der ADF-Anteil an der Gesamtlösung steigt und der Forms-Anteil wird langfristig kleiner.

In einer späteren Phase ist auch eine Migration bestehender Funktionen von Forms nach ADF möglich. Ob und wie diese Migration ggf. ablaufen könnte – z.B. manuelle Neuentwicklung einzelner Funktionen in ADF oder teil-automatisierte Migration mit Hilfe eines Tools – ist derzeit noch offen. Der hier beschriebene Weg einer Integration von Forms und ADF eröffnet einen weichen Ausweg, bei dem der Schwerpunkt der Investition bzw. des Aufwands in die Entwicklung neuer Funktionen auf Basis von ADF gelegt werden kann statt auf die Konvertierung von Funktionen von einer Technologie in die andere.

3.2 Herausforderungen und Integrationsaspekte

Die Integration der Technologien findet in der Präsentationsschicht bzw. auf dem Client statt. Damit kann ein gemeinsames Datenmodell, eine gemeinsame Datenbank sowie in der Datenbank vorliegende Anwendungslogik in PL/SQL verwendet werden. Zugleich arbeitet der Anwender nur mit einer einzigen Benutzeroberfläche.

Da die Benutzeridentität in einer Forms-Anwendung auf jeweils eigenen Datenbank-Benutzern aufsetzt, muss dies beim in ADF üblicherweise verwendeten Connection Pooling berücksichtigt werden. Ebenso muss der Tatsache Rechnung getragen werden, dass Forms und ADF jeweils eigene Datenbank-Verbindungen nutzen und zudem ADF Daten ggf. nur aus dem Cache im ADF Model liest statt diese jedesmal von der Datenbank erneut abzufragen.

Des Weiteren sollte das Forms Applet so in die von ADF Faces generierte HTML- und Javascript-basierte Seite integriert werden, dass dies für den Entwickler und Endanwender möglichst transparent ist. Hier helfen *Java to Javascript*-Kommunikation, ein geeignetes Design der ADF-Anwendung sowie die Verwendung von deklarativen Komponenten. Zudem muss eine gemeinsame Authentifizierung erfolgen, damit der Anwender sich innerhalb der in ADF eingebetteten Forms-Anwendung nicht erneut anmelden muss.

4. Integration des Forms Applets in ADF Faces

Für die Ausführung des Forms Applets innerhalb einer ADF Seite sind prinzipiell die beiden nachfolgend dargestellten Ansätze denkbar, wobei nur auf den letzten – erfolgreich implementierten – im Detail eingegangen wird.

4.1 Multi-Page / Legacy Lifecycle Ansatz

Das Forms Applet kann grundsätzlich mit Hilfe eines HTML Frames, Inline Frames oder direkt über ein *applet*-Tag in eine ADF Faces Seite eingebunden werden.

```
<f:verbatim>
        <applet height="100%" width="100%"
          code="oracle.forms.engine.Main"
          ...
          MAYSCRIPT="true">
...
</f:verbatim>

<af:inlineFrame id="formsView1" source="#{attrs.FormsUrl}"/>
```

Die direkte Einbettung des *applet*-Tags mit Hilfe von *f:verbatim* ist bei Verwendung von Forms 11g erschwert, da hier im Gegensatz zu 10gR2 ein neuer Parameter Forms Session ID zum Einsatz kommt, dessen Wert (z.B. „formsapp.2“) von den Forms Services für jede neue Session generiert wird. Insofern bietet sich die Einbettung als Inline Frame an, so dass der Code für den Applet-Aufruf wie gewohnt vom Forms Server generiert wird.
Das zentrale Problem bei der Applet-Einbettung ist die Performance bei einem Seitenwechsel in der ADF-Anwendung bzw. wiederholten Aufruf einer Seite mit eingebettetem Forms Applet. Wenn hierfür jedes Mal ein vollständiger Start einer Forms Session durchgeführt werden muss, dauert dies zu lang bzw. ist für den Anwender deutlich spürbar. Zudem würde jeweils eine eigene neue Forms Session erstellt und damit der Anwendungszustand nicht erhalten bleiben. Eine mögliche Lösung hierfür liegt in der Verwendung des mit Java 1.4 bereits vor einiger Zeit eingeführten sog. *Legacy Lifecycle* für Java Applets. Dieser sieht vor, dass Applets beim Wechsel von einer Seite zu einer anderen nicht beendet, sondern in einen ruhenden Modus versetzt werden – das Applet ist nicht mehr aktiv, bleibt aber im Speicher. Dadurch kann bei einem Wechsel zurück zu der das Applet beinhaltenden Seite dieses zum einen sehr schnell wieder reaktiviert werden und zum anderen bleibt der Zustand des Applets vollständig erhalten. Dementsprechend kehrt der Anwender genau an die Stelle der Forms-Anwendung inklusive angezeigter Daten zurück, an der er den Seitenwechsel vorgenommen hatte. Das Konzept wurde von Wilfred van der Deijl beschrieben und in der Lösung *OraFormsFaces* (siehe *http://www.commit-consulting.com/oraformsfaces/*) umgesetzt.
Die Verwendung des *Legacy Lifecycle* an sich – nicht zu verwechseln mit dem Einsatz von *OraFormsFaces* – hat sich in Tests als leider nicht besonders stabil erwiesen und ist zudem stark abhängig von dem verwendeten Update-Release des JRE. Neben einigen Bugs, an deren Behebung Oracle arbeitet (z.B. Sun Bug ID 6943350, mittlerweile laut Oracle behoben), ist die zwangsweise Beendigung von Applets im Ruhezustand bei Erreichen der konfigurierten maximalen Heap-Größe (*memory pressure condition*) Bestandteil der normalen Funktionsweise. Effektiv führt dies dazu, dass der Anwendungsentwickler keine gänzliche Kontrolle über die Ausführung des Forms Applets hat und dieses unter Umständen durch das Java Plugin beendet wird, was aber nicht im Sinne des Anwenders sein kann. Das Java Plugin ist mit dieser Verhaltensweise eher auf kleinere, wiederaufsetzbare Applets ausgerichtet als auf zustandsbehaftete Rich Clients wie Oracle Forms.

4.2 Single-Page Ansatz

Als Alternative lässt sich ein Single-Page Ansatz verfolgen, bei dem die Verwendung des *Legacy Lifecycle* nicht notwendig ist – die ADF-Anwendung wird mit der Maßgabe entwickelt, dass die komplette Anwendung in einer einzigen Seite (JSPX oder JSF) abläuft. Dies entspricht prinzipiell auch dem Wunsch, die ADF-Anwendung dem Anwender als Rich Client zu präsentieren, der eher das Gefühl vermittelt, mit einer Anwendung zu arbeiten als eine Folge von Browser-Seitenwechseln zu erleben. Weitere Informationen zu einem entsprechenden Design finden sich weiter unten. Durch diese Maßgabe bleibt das auf der Seite eingebettete Forms Applet jederzeit geladen und die Forms Session erhalten. Gleichzeitig muss jedoch dafür gesorgt werden, dass das Forms Applet nur in den gewünschten Anwendungsbestandteilen eingeblendet wird.
Da mit einer einfachen und geradlinigen Einbettung des Applets als *af:inlineFrame* die Komponente unter bestimmten Umständen durch ADF Faces im Browser wieder aus der DOM-Struktur entfernt

wird, muss das Management des Inline Frame durch direkte Manipulation des DOM mit Javascript gesteuert werden. Darunter fallen die Erzeugung des Inline Frame, Geometrie-Management sowie das Ein- und Ausblenden, wenn eine Forms-Maske dargestellt werden soll oder die Kontrolle zurück an ADF geht.

4.2.1 Deklarative Komponente *DynamicFormsView*

Die Implementierung des Inline Frames übernimmt eine deklarative Komponente *DynamicFormsView*. Die Komponente nimmt über Attribute den Namen der zu startenden Forms-Maske, Form-Parameter, den Namen der *config*-Sektion u.a. entgegen und erzeugt zur Laufzeit via Javascript im DOM ein *iframe* Element. Die Positionierung erfolgt mittels CSS *fixed positioning* und orientiert sich dabei an einem ADF Faces Anker- oder Proxy-Element wie z.B. einem *af:panelGroupLayout*, um das Ein- und Ausblenden zu steuern (siehe *Applet Display Container* in Abb. 1). Hierbei müssen Browser-Spezifika berücksichtigt werden, da z.B. Firefox ein Element beim Setzen der CSS-Eigenschaft *display:none* zerstört, was die Beendigung des Forms Applets zur Folge hat (Firefox Bug 484209). Des Weiteren führt ein durch ADF unter bestimmten Umständen ausgelöstes Neuladen der kompletten Seite im Browser ebenfalls zur ungewollten Beendigung des Applets. Dies kann verhindert werden, indem sowohl die ADF Anwendung als auch das Forms Applet in jeweils eigene Inline Frames eingesperrt werden, so dass diese wirksam voneinander getrennt sind und über eine gemeinsame Rahmenseite miteinander verbunden werden. Im Gegensatz zu normalen HTML Frames haben Inline Frames den Vorteil, dass diese übereinander positioniert werden können. Die Seiten-Struktur im Browser ist dann wie in Abb. 1 dargestellt vorstellbar.

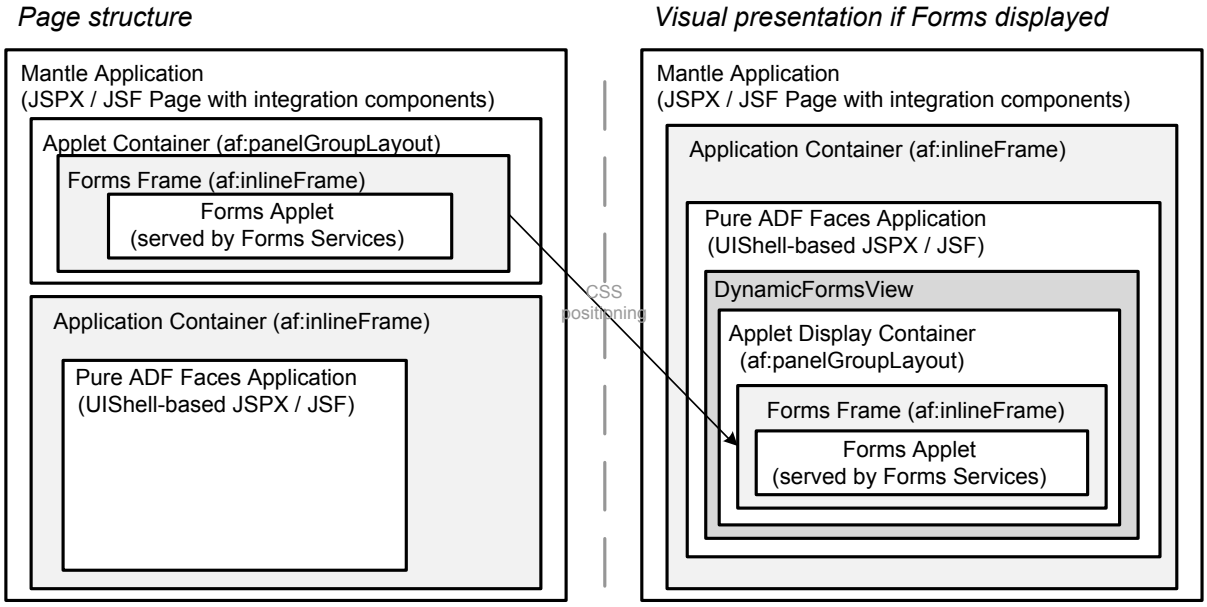

Abb. 1: Struktur und visuelle Darstellung des in die ADF-Anwendung integrierten Forms Applets

Für die client-seitige Kommunikation zwischen ADF- und Forms-Integration ist eine in die *DynamicFormsView* integrierte Javascript-Komponente zuständig, die neben dem DOM-Handling und Geometrie-Management beispielsweise auch das Starten einer Forms-Maske im Applet mit entsprechenden Parametern veranlasst, wenn dies von der ADF-Anwendung angefordert wurde. Die Trennung in jeweils eigene Frames bedingt, dass aufgrund der *Same-Origin Policy* ggf. ein Mechanismus zur *Cross Domain*-Kommunikation implementiert werden muss, sofern Forms Applet und ADF-Anwendung von unterschiedlichen Systemen stammen. Hierzu lässt sich die in HTML 5 standardisierte *postMessage*-Funktion verwenden, die in aktuellen Browserversionen unterstützt wird.

Für ältere Browser existieren entsprechende Fallback-Mechanismen. In Oracle ADF 11.1.2 wurde ein neuer Kontextparameter oracle.*adf.view.rich.security.FRAME_BUSTING* eingeführt, dessen Standardwert *differentDomain* dafür sorgt, dass beim Einbinden einer ADF-Anwendung in unterschiedlichen Domains die Anwendung ihre eigene URL selbst als top-level location im Browser setzt. Dies dient dazu auf *Cross Site Scripting* beruhende Angriffe (z.B. sog. *ClickJacking*) zu vermeiden. Falls tatsächlich verschiedene Domains verwendet werden, kann ein Teil der Lösung für den betrachteten Anwendungsfall darin bestehen, den Parameter auf den Wert *never* zu setzen.

4.2.2 Anpassungen in Forms

Für die Kommunikation der *DynamicFormsView* Komponente mit Forms kann *Java to Javascript*-Kommunikation verwendet werden, die darauf beruht von Javascript aus öffentliche Methoden der Java Applet-Klasse aufzurufen bzw. von einer Java Klasse aus einen Javascript-Ausdruck durch die Javascript Runtime des Browsers evaluieren zu lassen. Ersteres kann genutzt werden, um die Forms Applet-Methode *raiseEvent(eventName, eventPayload)* aufzurufen. Sofern der Applet-Parameter *enableJavascriptEvent* auf *true* gesetzt ist, führt dies zur Ausführung eines entsprechenden *WHEN-CUSTOM-JAVASCRIPT-EVENT* Triggers in der aktiven Forms-Maske. Die Maske kann somit auf Anforderungen der ADF-Anwendung wie z.B. „starte Modul X mit Parameter Y“ reagieren. Genauso ist es auf umgekehrtem Wege möglich, der ADF-Anwendung via Javascript mitzuteilen, dass ein Prozess-Schritt in Forms abgeschlossen ist und der aktive ADF Task Flow einen entsprechenden *outcome* interpretieren und eine Folge-Aktion durchführen soll.

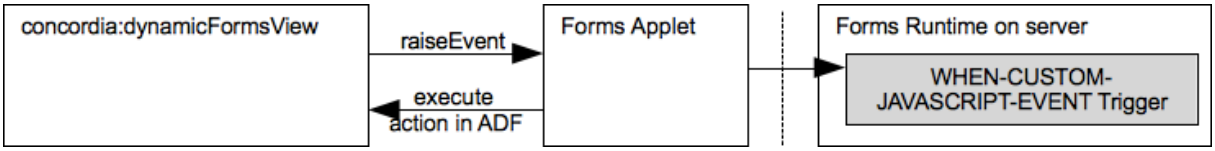

Abb. 2: Client-seitige Kommunikation zwischen ADF Faces und Forms

4.2.3 Design der ADF Anwendung

Das hier beschriebene Design der ADF Anwendung lehnt sich stark an eine von mehreren von Chris Muir beschriebenen Design-Alternativen an (siehe Präsentation *Angels in the Architecture: An Oracle Application Development Framework Architectural Blueprint* auf der Oracle Open World 2011 sowie die ADF Enterprise Methodology Group). Für die Implementierung des Single-Page-Ansatzes lässt sich als Grundlage das von Oracle zur Verfügung gestellte Page Template *UIShell* verwenden und für die Forms-Integration modifizieren. Die *UIShell* basiert ebenfalls auf dem Gedanken, nur eine einzige JSPX-Seite darzustellen und die als Bounded Task Flows (*BTFs*) implementierten verschiedenen Anwendungsbestandteile unter Verwendung von dynamischen Regionen in jeweils eigenen Tabreitern anzuzeigen. Für die Einbindung eines Tabreiters, auf dem das Forms Applet dargestellt wird, muss die *UIShell* noch um einen *TabSelectionChangeListener* erweitert werden, damit das Ein- und Ausblenden bzw. Überblenden des Inline Frames mit dem Applet veranlasst werden kann.
Die Anwendung besteht damit wie in Abb. 3 ersichtlich aus einer Reihe von JDeveloper Projekten, ggf. nach Anwendungsbereichen noch in JDeveloper Workspaces aufgeteilt, die Bounded Task Flows bereitstellen (Modul-Analogie zu Forms). Diese werden wiederum als *ADF Library JARs* gebunden zur Einbindung in andere Bounded Task Flows. Das letztendliche Deployment eines Moduls erfolgt als *WebLogic Shared Library* auf dem Server, so dass die *UIShell* die Komponente zur Laufzeit laden kann. Für die Kommunikation zwischen den lose gekoppelten Modulen können Contextual Events verwendet werden. Damit kann beispielsweise ein Bounded Task Flow, der ein Menü mit verfügbaren Anwendungsfunktionen als ADF Faces Tree darstellt, in der *UIShell* ein Ereignis *menuNavigationStarted* auslösen, das zur Aktivierung eines neuen Tabreiters führt, auf dem ein Bounded Task Flow ausgeführt wird, der die gewünschte Funktion bereitstellt.

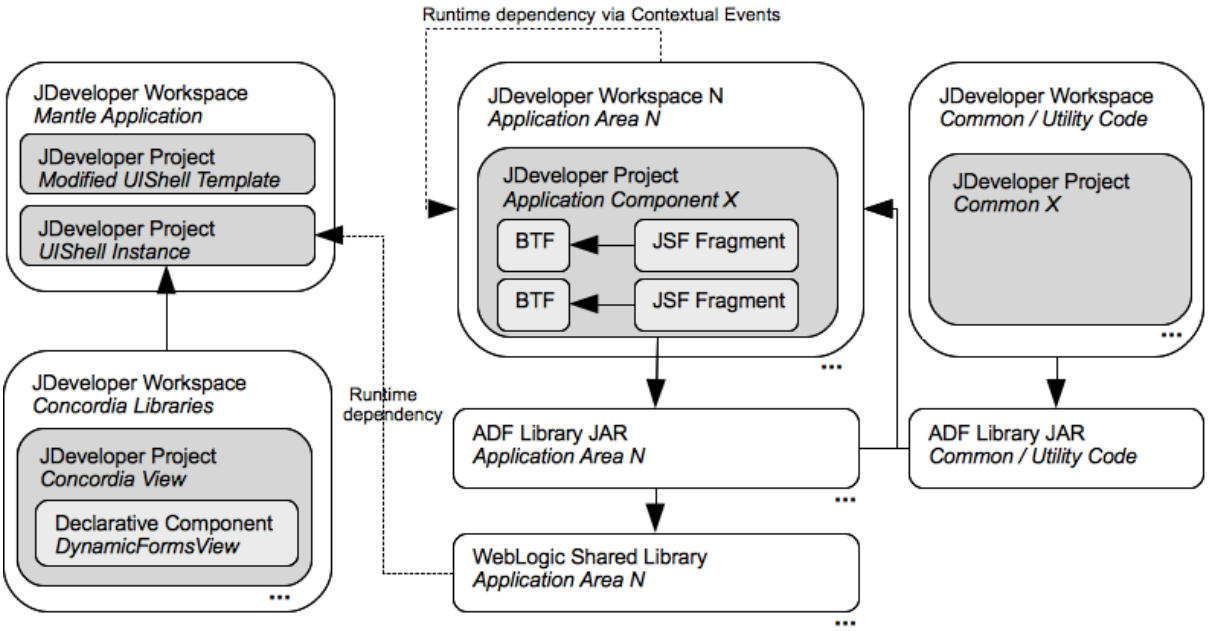

Abb. 3: Vereinfachte Darstellung der Struktur

Das beschriebene Design der ADF Anwendung ermöglicht eine stark arbeitsteilige Entwicklung und hat sich dafür bereits in der Prototypentwicklung bei der infoteam GmbH Berlin bewährt. Zudem ist eine Hotfixbarkeit von Anwendungsbestandteilen gewährleistet und die Anwendung auf ein starkes Wachstum im Funktionsumfang vorbereitet.

5. Weitere Integrationsmaßnahmen und Ausblick

Für die Authentifizierung kommen sowohl die Verwendung von *Single Sign On* als auch die Übergabe von Benutzerinformationen über die Nutzung eines gemeinsamen Kontexts in der Datenbank in Betracht. Bei Verwendung von existierenden, zustandsbehafteten Datenbank-Packages im Zusammenhang mit dem ADF Connection Pooling ist ein Konzept mit Hilfe von Wrapper-Packages denkbar (siehe auch den Vortrag von Ulrich Gerkmann-Bartels auf der DOAG Konferenz 2010). Des Weiteren kann für die Bewahrung der Benutzeridentität in der Datenbank die Proxy Authentication Funktionalität verwendet werden. Ein erster Test unter Verwendung einer eigenen *oracle.jbo.server.DatabaseTransactionFactory* und erweiterten *DBTransactionImpl2*-Klasse sowie einer überschriebenen *prepareSession*-Methode im ADF Application Module zur Öffnung einer Proxy Session ist erfolgreich verlaufen.

Insgesamt besteht damit eine gute Aussicht für die Integration von ADF und Forms in einer zukünftigen, gemeinsamen Anwendung. Für die umfangreichen Informationen und Anregungen im Zusammenhang mit der ADF-Entwicklung und den verschiedenen Integrationsaspekten möchte ich besonders den in der Oracle ADF Community Aktiven danken. Sehr hilfreich sind nicht zuletzt auch die Diskussionen in der ADF Enterprise Methodology Group, das von Frank Nimphius und Lynn Munsinger verfasste Buch *Oracle Fusion Developer Guide: Building Rich Internet Applications with Oracle ADF Business Components and Oracle ADF Faces*, eine große Zahl von Blogs mit ADF-Bezug sowie die Unterstützung durch Oracle Platform Technology Solutions gewesen.

Kontaktadresse:
Andreas Leidner
infoteam GmbH Berlin
Königsberger Straße 14

D-12207 Berlin

Telefon: +49 (0) 30-77 33 000
Fax: +49 (0) 30-77 33 004
E-Mail andreas.leidner@infoteam-berlin.de
Internet: www.infoteam-berlin.de