ADF – eine Universallösung für Web-Anwendungen?

Christian Schmitz und Stefan Glase, OPITZ CONSULTING GmbH

ADF, das Application Development Framework von Oracle, wird als Komplettlösung für die Anwendungsentwicklung im Web empfohlen. Der 4-GL-Ansatz soll dabei insbesondere Forms-Entwicklern den Technologiewechsel erleichtern. In diesem Artikel wird den Fragen nachgegangen, wie und in welchen Szenarien sich ADF in der Praxis bewährt hat und welche Vor- und Nachteile gegenüber einem vergleichbaren Open-Source-Stack existieren.

Das Oracle Application Development Framework (ADF) ist ein Framework zur Erstellung von webbasierten Anwendungen auf Basis der Java Enterprise Edition. Es wurde im Jahr 2003 von Oracle mit dem Ziel vorgestellt, die Entwicklung von Java-EE-Anwendungen zu vereinfachen und die Umsetzung bewährter Java-EE-Entwurfsmuster zu fördern. Der gewählte 4GL-Ansatz für die Entwicklung von ADF-Applikationen soll insbesondere Oracle-Forms-Entwicklern den Umstieg in die Java-Entwicklung erleichtern. ADF vereint für die Umsetzung dieser Ziele in allen Schichten der resultierenden Anwendung gut integrierte Lösungen – sei es die Anbindung der Persistenzschicht, die Abstraktionsschicht der zugrunde liegenden Services oder das breit aufgestellte Angebot von Technologien zur Gestaltung der Benutzungsoberflächen.

Nach dem Motto „Productivity with Choice“ ist es bei der Entwicklung möglich, an diversen Stellen in der Applikation Entscheidungen über die zu verwendenden Technologie zu treffen. Die Oberfläche kann beispielsweise als Swing Client oder als Web-Anwendung mit JavaServer Pages, JavaServer Faces sowie ADF Faces implementiert werden.

Die ADF-Modelschicht entkoppelt die Oberflächen-Schicht von der Business-Schicht, sodass hier beliebige Technologien (ADF Business Components, Web Services, Enterprise Java Beans, Plain Old Java Objects etc.) für die Bereitstellung von Daten zur Verfügung stehen. Der von Oracle bevorzugte und durch die Entwicklungsumgebung am besten unterstützte Weg sieht die Verwendung von ADF Faces für die Benutzungsoberfläche und ADF Business Components (ADF BC) für die Persistenzschicht vor. Dieser Weg stellt auch die Grundlage für den weiteren Vergleich in diesem Artikel dar.

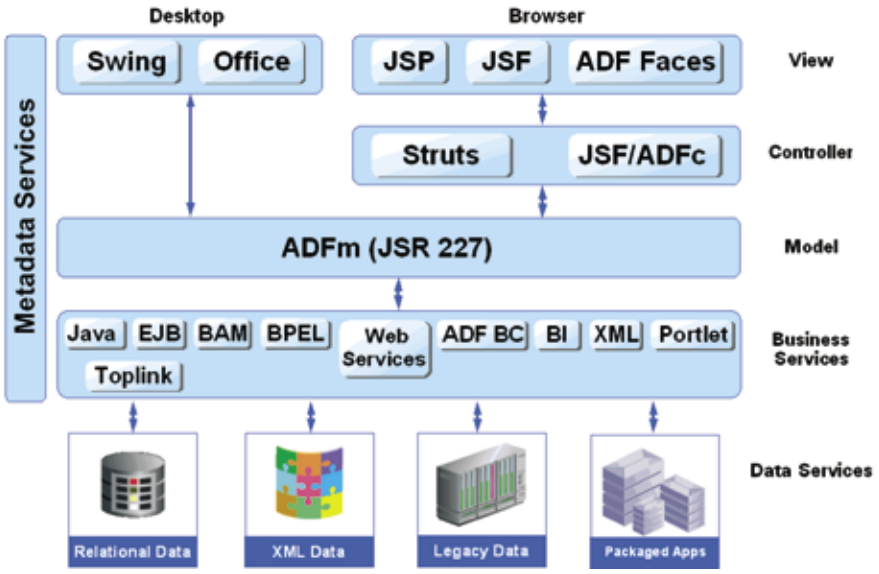

Abbildung 1: Darstellung von Oracle ADF im Developer's Guide 11gR1

Entwicklungsumgebung

Als produktive Entwicklungsumgebung ist man beim Einsatz von ADF an den Oracle JDeveloper gebunden, der die deklarative Entwicklung der Applikationen mit zahlreichen Wizards und grafischen Editoren weitreichend unterstützt. Beginnend bei den Entitäts-Objekten, die der Abbildung von Tabellen und Views in der Datenbank dienen, über die View-Objekte als Sicht auf Entitäts-Objekte in Form von SQL-Abfragen, bis hin zu den Applikationsmodulen zur Zusammenfassung von View-Objekten und Steuerung von Transaktionen, Locking und Pooling, können alle Elemente über Wizards generiert und erweitert werden. Auch die Benutzungsschnittstelle kann über einen grafischen Editor entworfen werden. So lassen sich sowohl einzelne Ein- und Ausgabe-Komponenten als auch View-Objekte mittels Drag & Drop direkt in die Oberfläche ziehen und mittels der gelieferten Komponenten visualisieren.

Datenobjekte werden über sogenannte Bindings an die Oberflächenkomponenten gebunden. Der Ansatz über Wizards erspart die Erzeugung der

notwendigen, umfangreichen XML-Metadaten. In der von den Autoren zuletzt im Projekt verwendeten Version 10.1.3.4 ist es allerdings trotzdem hilfreich, über Wissen um die Inhalte und Funktionsweise dieser Konfigurationsdateien im Team zu verfügen. Von Fall zu Fall ist ein manueller Eingriff hier durchaus erforderlich.

Vergleichbarer Open-Source-Stack

Oracle ADF bringt als Framework viele Funktionalitäten mit, die von den meisten auf dem Markt existierenden Produkten im Open-Source-Bereich allein nicht abgedeckt werden. Aus diesem Grund ist ein Vergleich mit Open-Source-Alternativen nur sinnvoll, wenn hier ein kompletter Stack an Open-Source-Produkten zu einem ähnlichen Framework kombiniert wird.

Für einen Vergleich haben sich die Autoren im vorliegenden Artikel für den Produkt-Stack entschieden, den wir im Folgenden vorstellen. Das Spring Framework dient als Dreh- und Angelpunkt der gesamten Applikation. Als Integrationsframework zwischen Business Service und View-Schicht kann es mit der Model-Schicht im zuvor dargestellten Oracle ADF verglichen werden. Hibernate ist ein Objekt-Relationales-Mapping-Framework (ORM) und dient der Abbildung der Entitäten auf die Datenbankstruktur.

Abbildung 2: Open-Source-Stack für den Vergleich mit Oracle ADF

Als Oberflächen-Technologie wird das Produkt Apache MyFaces als Implementierung des JavaServer-Faces-Standards in Kombination mit der Komponenten-Bibliothek JBoss RichFaces verwendet. Diese Komponenten-Bibliothek liefert eine große Menge an Oberflächen-Komponenten, die mit jenen von Oracle ADF Faces vergleichbar sind.

Stärken und Schwächen

Schnell wird bei der Betrachtung der Menge der oben genannten Produkte im Open-Source-Stack deutlich, welchen funktionalen Umfang Oracle ADF liefert – ein großer Vorteil von ADF: Die Technologien sind aufeinander abgestimmt, sodass ein reibungsloser Betrieb gewährleistet ist. Der Entwickler muss sich nicht mehr um die Verzahnung diverser eingesetzter Technologien kümmern.

Der mächtige Funktionsumfang bringt allerdings auch einen Nachteil mit sich: Da in kleineren Projekten oft nur ein geringer Teil benötigt wird, gestaltet sich der Einarbeitungsaufwand in ADF schnell umfangreicher als bei einem vergleichbaren Open-Source-Stack. Für die Einarbeitung stehen indes diverse Tutorials, Dokumentationen und Beispiele zur Verfügung, worauf man sich bei Open-Source-Projekten nicht immer verlassen kann. Ein Problem könnte hier weniger darin bestehen, dass es nicht ausreichend Material gibt, sondern, dass das breite Angebot den Entwickler erst einmal erschlägt und die relevanten Stellen herausgefiltert werden müssen.

Wie zu Anfang schon erwähnt, empfiehlt Oracle, bedingt durch die Menge von verfügbarer Dokumentation und die Unterstützung durch den JDeveloper, einen Weg, der den Einsatz von ADF Business Components (ADF BC) benötigt – auch bekannt unter dem Namen „Business Components for Java“ (BC4J). ADF BC wirkt im Vergleich zu den Open-Source-Technologien Spring und Hibernate, verbunden mit den Möglichkeiten von Java-5-Sprachmitteln wie Annotationen und generische Typen, schwerfällig und erfordert umfangreiche Java-Sourcen und XML-Konfigurationen.

Die Pflege der XML-Konfigurationsdateien erfolgt dabei ausschließlich über die Wizard-Funktionen im JDeveloper, wobei nicht selten ausschließlich die Reihenfolge der XML-Tags und nicht der tatsächliche Inhalt geändert werden. Dieser Umstand kann bei paralleler Entwicklung mit mehreren Entwicklern schnell zu Frustrationen führen, da unnötige Konfliktsituationen bei der Verwendung einer Versionsverwaltung beschworen werden. Auch die Erweiterung der im Framework existierenden Basisklassen ist zwar möglich, aber durch den Umfang der Basisfunktionalitäten keinesfalls trivial. Fundierte Kenntnisse dieser Funktionalitäten sind daher unabdingbar.

Der erfahrene Java-Entwickler wird sich wahrscheinlich durch die Einschränkung auf die Verwendung des JDevelopers etwas eingeengt fühlen. Dennoch wirkt der JDeveloper mit jeder Version erwachsener. Bewährte Funktionalitäten aus anderen Entwicklungsumgebungen fließen verstärkt in die Weiterentwicklung ein. Ein berechtigter Kritikpunkt an dieser Stelle ist die Verzögerung, mit der diese Funktionalitäten im Vergleich zu anderen Entwicklungswerkzeugen am Markt verfügbar sind.

Obwohl auch mit Eclipse die Integration der gewünschten Erweiterungen (in Form von Plugins) beziehungsweise das Zusammenspiel der Plugins nicht immer reibungslos funktioniert, so vermisst man beispielsweise in der Version 10.1.3.4 des JDevelopers die Möglichkeit mit dem Versionskontrollsystem Subversion 1.5 aus der Entwicklungsumgebung heraus zu arbeiten. Die geringere Zahl an Plugins ist ansonsten leichter zu verschmerzen, denn der Entwickler bekommt bereits mit der Installation eines JDevelopers eine solide Basis für die Entwicklung von Applikationen im Java-Umfeld.

Genauso wie ADF BC für die Implementierung der Business Services in der Regel als gesetzt gilt, so wird für die Oberfläche ADF Faces propagiert. ADF Faces erweitert als Komponenten-Bibliothek den in der Java Enterprise Edition 5 fest verankerten Standard JavaServer Faces. Im Bereich der Komponenten-Bibliotheken auf Basis von JavaServer Faces setzt Oracle schon seit einigen Jahren Maßstäbe mit einer umfangreichen Palette von ausgereiften und interaktiven Komponenten. Zwar gibt es auch in der Version 10.1.3.4 noch einige Altlasten, die auf UIX-Zeiten zurückgehen, doch ein Blick in die Dokumentation der aktuellen Version 11*g*, in der die Komponenten-Bibliothek nun den Namen „ADF Faces Rich Client“ trägt, genügt, um zu

sehen, dass auch diese Probleme weitestgehend adressiert wurden.

Als Beispiel wird die horizontale Ausrichtung von Inhalten in einer Tabellenspalte betrachtet. Seit Version 11*g* wird die Ausrichtung von Inhalten in Tabellenspalten (<af:column />) über das Attribut align gesetzt, das auch im HTML-Umfeld bekannt ist. In den Vorgängerversionen findet die horizontale Ausrichtung über das Attribut formatStyle mit den gültigen Werten „text", „number" und „icon" für links- und rechtsbündige sowie zentrierte Positionierung statt.

Im Bereich der Testbarkeit des Codes kann der Open-Source-Stack Punkte sammeln: Das durchgängige Konzept der Dependency Injection hilft, Abhängigkeiten zwischen Komponenten und Objekten zu minimieren. In der Folge können einzelne Anwendungsbereiche in Isolation getestet und entwickelt werden.

Die im September veröffentlichte Version 11*g* des Oracle JDevelopers und des Oracle Application Development Frameworks (ADF) bietet etwa 150 mächtige JSF-Komponenten mit Unterstützung von AJAX-Funktionalität für die Gestaltung von intuitiven und interaktiven Benutzungsoberflächen für Web-Anwendungen. Der Einsatz erfordert allerdings einen Java EE 5 Container, sodass die neu erstellte Anwendung auf einem vorhandenen Application Server in einer Version kleiner 11 nicht betrieben werden kann. Dies bedeutet aktuell, dass eine Migration auf den von BEA Systems übernommenen WebLogic Server notwendig ist, da es bis dato keine 11*g*-Version des bekannten Oracle Application Servers mit Unterstützung für Java EE 5 gibt.

Fazit

Die Erfahrungen in diversen Projekten mit ADF und dem hier vorgestellten Open-Source-Stack zeigen, dass es keine generelle Empfehlung für die eine oder die andere Variante gibt. In beiden Varianten ist für die Erstellung von Applikationen mit größerer Komplexität schnell ein Expertenwissen notwendig. Für die Einarbeitung neuer Mitarbeiter ist die Lernkurve stark von dem jeweiligen Hintergrundwissen abhängig. Erfahrene Java-Entwickler werden sich schneller in die flexiblen Open-Source-Frameworks einarbeiten, Entwickler mit prozeduralem und 4GL-Hintergrund wird ein schnellerer Einstieg in ADF gelingen. Ob die Verschleierung von objektorientierten Design-Prinzipien auf Dauer positiv für die Weiterbildung eines Java-Entwicklers ist, erscheint allerdings fragwürdig.

Der Trend der ADF Rich Faces ist grundsätzlich positiv zu beurteilen, wäre da nicht das schwergewichtig wirkende ADF BC als Empfehlung für die Business-Schicht und die Unsicherheit, ob mit der Entscheidung für den Einsatz auch eine Migration der Laufzeitumgebung auf Oracle Weblogic Server notwendig ist. Die Entscheidung für den Einsatz sollte darüber hinaus mit den fachlichen Anforderungen abgeglichen werden. Können die meisten Anforderungen mit den bestehenden Funktionalitäten abgedeckt werden, spricht einiges für den Einsatz von ADF. Existieren jedoch viele Anforderungen, die auf den ersten Blick nicht mit den Standardmitteln abgedeckt werden, kann dies die Wahl eines flexibleren Open-Source-Stacks Aufwandstreiber in einem Projekt vermeiden.

In Anbetracht der Tatsache, dass das Geschäftsmodell, das auf Beratung und Support zu einem Open-Source-Produkt basiert, zunehmend an Popularität gewinnt, wird die Entscheidung zudem erleichtert, wenn neben den eingesetzten Produkten auf Dienstleister in Anspruch genommen werden können, die kritische Probleme während der Entwicklung oder des Betriebs abfangen.

Weiterführende Literatur

Oracle ADF
- Oracle ADF: http://www.oracle.com/technology/products/adf/
- Oracle ADF Faces Rich Client: http://www.oracle.com/technology/products/adf/adffaces/

Open Source
- Hibernate: http://www.hibernate.org/
- Spring Framework: http://www.springframework.org/
- Apache MyFaces: http://myfaces.apache.org/
- JBoss RichFaces: http://www.jboss.org/jbossrichfaces/

Entwicklungsumgebung
- Oracle JDeveloper: http://www.oracle.com/technology/products/jdev/
- Eclipse: http://www.eclipse.org/

Kontakte:
Christian Schmitz
christian.schmitz@opitz-consulting.de
Stefan Glase
stefan.glase@opitz-consulting.de

Unsere Inserenten

BzYxS.com Database Optimisation www.bzyxs.com	Seite 47	ORACLE Deutschland GmbH www.oracle.com	Umschlagseite 3
Hunkler GmbH & Co KG www.hunkler.de	Seite 3	PROMATIS software GmbH www.promatis.de	Seite 9
Imining GmbH www.imining.de	Seite 27	Quest Software GmbH www.quest.com	Seite 5
MuniQsoft GmbH www.muniqsoft.de	Seite 17	Team GmbH Paderborn www.team-pb.de	Seite 11
OPITZ CONSULTING GmbH www.opitz-consulting.de	Umschlagseite 2	Trivadis GmbH www.trivadis.com	Seite 45, Umschlagseite 4