ADF Persistenzframeworks im Vergleich –

JPA/EJB vs. ADF BC

Hendrik Gossens
MT AG
Ratingen

Schlüsselworte
ADF, ORM, Persistenz, JPA, EJB

Einleitung
Durch die Trennung von Datenhaltung, Anwendungslogik und -darstellung ermöglicht es ADF auf den verschiedenen Schichten unterschiedliche Technologien zu verwenden. Neben den Oracle-Lösungen können dabei auch OpenSource-Technologien zum Einsatz kommen. So bietet ADF beispielsweise zur Anbindung von Business Services verschiedene vorimplementierte Data Controls an, die die Verwendung von Persistenzmechanismen wie ADF BC oder EJB/JPA "out-of-the-box" auf deklarative Weise ermöglichen. Der Vortrag stellt die Oracle-Lösung "Business Components" der OpenSource Variante "EJB/JPA" gegenüber und vergleicht Vor- und Nachteile anhand von ShowCases.

Objektrelationales Mapping
Objektorientierte Programmiersprachen (OOP) wie Java kapseln Daten und Verhalten in Objekten, wohingegen relationale Datenbanken Daten in Tabellen verwalten. Diese beiden Paradigmen sind grundlegend verschieden und führen zu Schwierigkeiten, wenn beide Welten miteinander interagieren sollen. So ist im OOP-Bereich der Aufbau teils komplexer Vererbungshierarchien obligatorisch, während dieses Konzept in relationalen Datenbanken gänzlich unbekannt ist, Die teilweise Inkompatibilität beider Konzepte wird als *Object-relational Impedance Mismatch* bezeichnet.

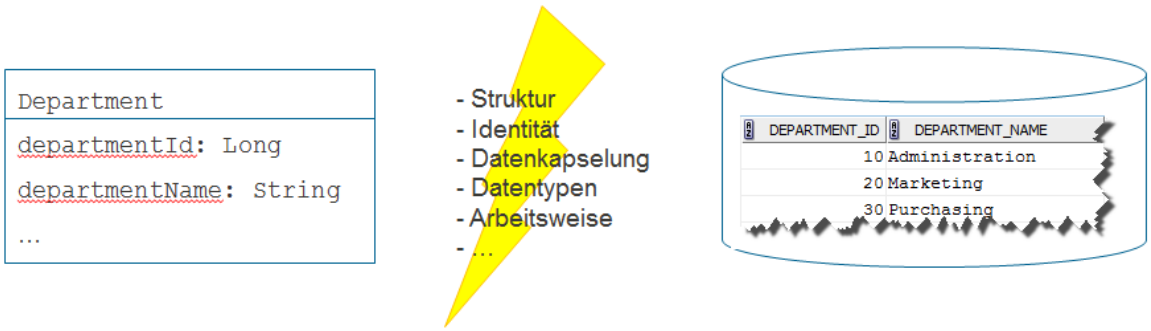

Abb. 1: Object-relational impedance mismatch

Grundlegende Unterschiede zwischen OOP und relationalen Datenbanken sind:

- **Struktur:** In der OOP kapselt ein *Objekt* Daten und Verhalten und kann darüber hinaus Bestandteil einer Klassenhierarchie sein. Ein solches Konzept fehlt in relationalen Datenbanken

- **Identität:** Ein *Objekt* hat eine vom Zustand (Daten) unabhängige Identität. Die Identität eines *Tupels* wird durch dessen Daten (bzw. Primärschlüssel) determiniert.
- **Datenkapselung:** Ein **Objekt** schützt seine Daten vor Veränderungen bzw. grenzt durch Methoden (das Verhalten) die Art, wie Daten verändert werden können, ab.
- **Arbeitsweise:** Daten und Zustände eines *Objekts* werden durch Methoden modifiziert, Daten einer relationalen Datenbank werden durch Transaktionen von einer verbundenen Anwendung modifiziert.

Um diese konzeptuellen Inkompatibilitäten ohne großen manuellen Aufwand ausgleichen zu können, empfiehlt sich der Einsatz von ORM-Frameworks wie ADF Business Components oder JPA/EJB.

ADF Schichtenarchitektur

ADF ermöglicht aufgrund der Schichtentrennung die Verwendung verschiedener Persistenzframeworks!

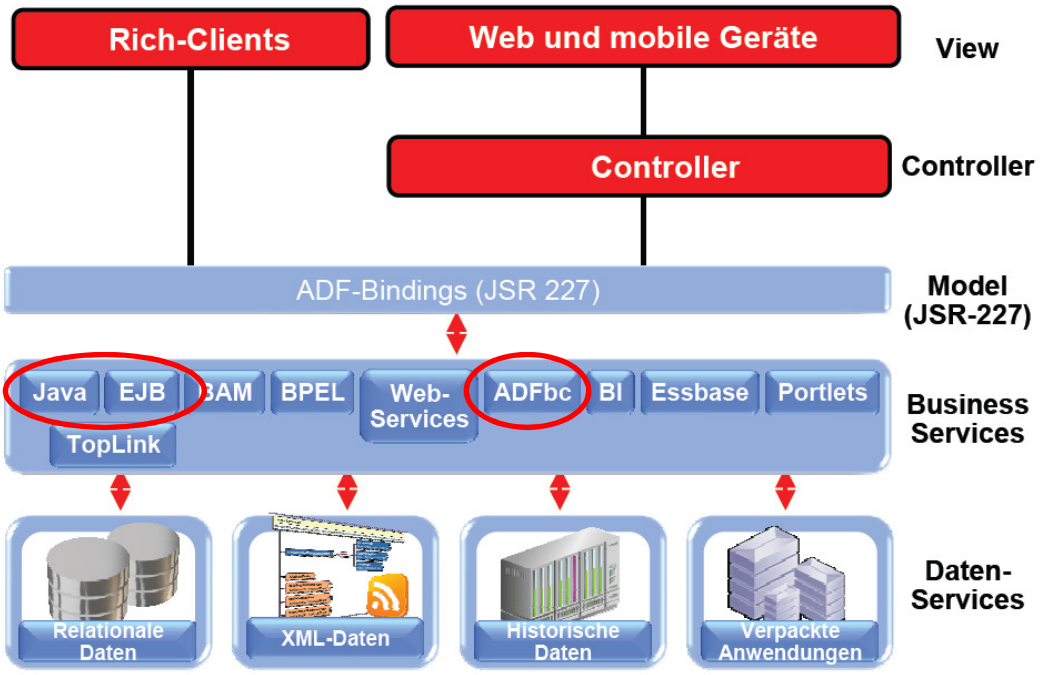

Abb. 2: ADF Schichtenarchitektur

ADF Business Components (ADF BC)

Bei den ADF Business Components handelt es sich um das „Standard"-Persistenz-Framework von ADF. Wesentliche Features sind:

- Objektrelationales Mapping
- Transaktion und Locking
- Möglichkeiten der Manipulation ähnlich zu Forms (doDML, before lock, …)
- LOVs auf Attributen

Wesentliche Komponenten von ADF sind

- Entity Objects
- View Objects
- Assoziationen
- View Links
- Application Modules

JPA/EJB

Bei JPA handelt es sich um eine standardisierte Schnittstelle, mit der relationale Daten in Java-Applikationen verwaltet werden können. Wesentliche Features sind:

- Objektrelationales Mapping
- Leichtgewichtig
- Industrie-„Standard"
- Vielfältige Persistence Provider (BatooJPA, EclipseLink, Hibernate, OpenJPA, …)
- EclipseLink (früher Oracle TopLink) Referenzimplementierung von JPA 2.0
- Plain Old Java Objects (POJO)
- Nutzung von JPA ist nicht auf Enterprise Java Beans (EJB) beschränkt
    - JPA ist Untermenge von EJB3, Implementierung bereitgestellt durch Persistence Provider
    - EJB Implementierung bereitgestellt durch EJB Container des Application Servers
- Konfigurierbar via Annotations und/oder XML

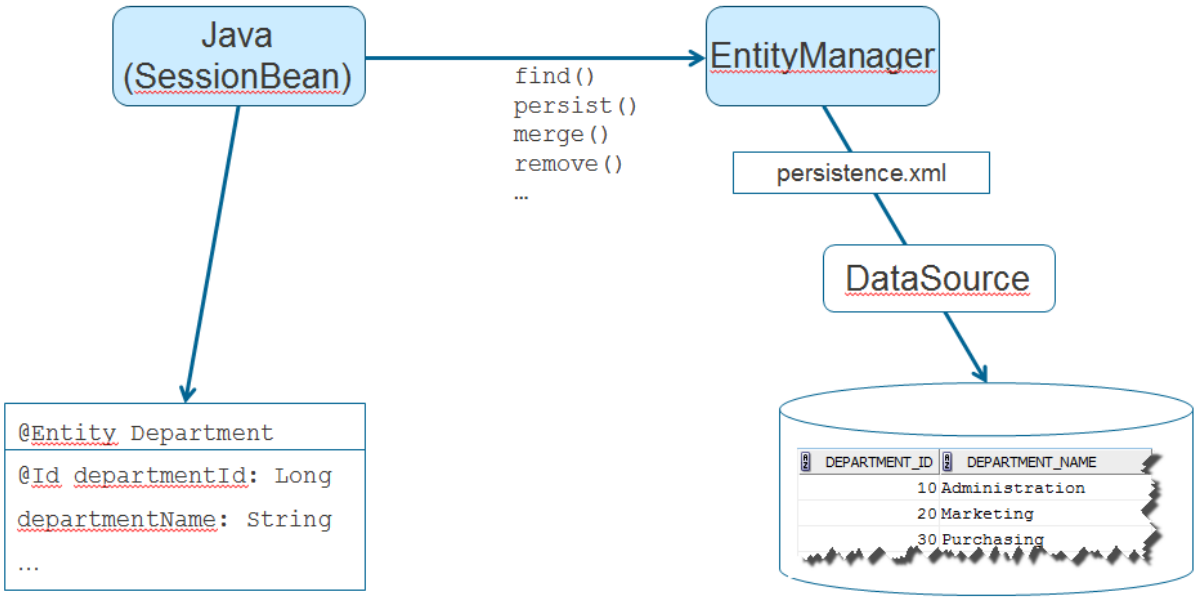

Abb. 3: Konzeptuelle Sicht auf JPA/EJB

Gegenüberstellung ADF BC und JPA/EJB
ADF BC und JPA verfolgen ähnliche Konzepte. Dies soll folgende Gegenüberstellung verdeutlichen:

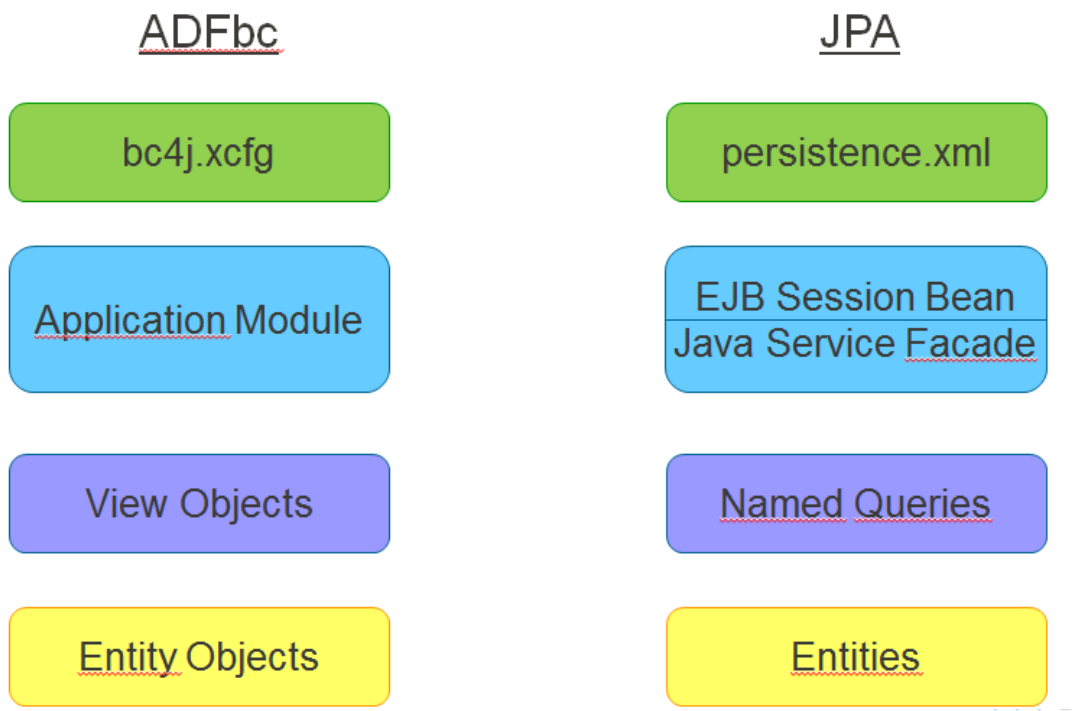

Abb. 4: Gegenüberstellung ADF BC und JPA/EJB

Gründe für ADF BC	**Gründe für JPA/EJB**
▪ Gute Integration mit JDeveloper ▪ Gute Integration mit Oracle Datenbank ▪ Metadatengetrieben (XML) ▪ Deklarative Herangehensweise ▪ Einfach zu erlernen für Umsteiger aus der „alten“ Oracle-Welt ▪ Java Implementierung anpassbar (Vererbung, Hook Points, …) ▪ Gute Testbarkeit durch Business Components Tester	▪ Einfach erlernbar ▪ Breite Anzahl an Java Entwicklern ▪ Gute Dokumentation und Hilfestellung durch Java Community ▪ Datenbankunabhängig ▪ Offener Standard => verringert Gefahr des „Vendor Lock-In“ ▪ Kaum Konfigurationsaufwand ▪ Eigene, SQL-ähnliche Abfragesprache (JPQL) ▪ Criteria Queries ▪ Multitenant Entities ▪ Breite Toolunterstützung (JDeveloper, Eclipse, NetBeans, …) ▪ Plain Old Java Objects (POJO) => Jede Java-Klasse persistierbar ▪ Gute Testbarkeit, z.B. mit Hilfe von JUnit ▪ Metadatengetrieben (XML oder Annotations)

Fazit

In einem Oracle-dominierten Stack empfiehlt sich erfahrungsgemäß die Verwendung von ADF BC. Besonders bei der Migration von Forms-Anwendungen nach ADF können diese ihre Vorteile ausspielen. In der Regel wird in einem solchen Szenario das bestehende Datenmodell zumindest größtenteils weitergenutzt. Hier stößt man mit JPQL schnell an die Grenzen. Der Einsatz von JPA kann dann sogar soweit führen, dass native SQL-Queries verwendet werden müssen, die die Spezifika der Oracle Datenbank nutzen. Hierdurch verliert man die Vorteile von JPA bei gleichzeitig steigender Komplexität der Persistenzschicht. Hier haben die ADF Business Components aufgrund der guten Integration mit der Oracle Datenbank die Nase vorn.

Quellen

- http://docs.oracle.com/html/E24396_01/ejb3_overview_arch.html
- ADF 11g Fundamentals Schulung
- Update/Insert With JPA and EJB using ADF (Shay Schmelzer) http://www.youtube.com/watch?v=VyPOhmWD5Y0
- http://docs.oracle.com/cd/E28280_01/web.1111/b31974/bcquerying.htm
- http://openejb.apache.org/jpa-concepts.html
- Java EE 6 – Anwendungen entwickeln mit JSF, CDI, EJB und JPA, Michael Kulla, Addison Wesley

Kontaktadresse:

Hendrik Gossens
MT AG
Balcke-Dürr-Allee 9
D-40882 Ratingen

Telefon:	+49 2102 30961-0
Fax:	+49 2102 30961-101
E-Mail	hendrik.gossens@mt-ag.com
Internet:	www.mt-ag.com