EJB 2.0 Persistenz

Dirk M. Sohn
sohn@oio.de

Agenda

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - CMR - Beziehungen
  - EJB-QL
  - Selektoren und Finder
- Ist EJB Persistenz = Container Managed Entity Beans?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

Aktuelle Herausforderungen

Orientation in Objects

- Forderung nach verteilten Lösungen im Bereich unternehmenskritischer Anwendungen
- Rapide Verbreitung des Internets sowie mobile und „*embedded devices*“
- Wachsender Wettbewerbsdruck, verkürzte Produktentwicklungszyklen, Verbreitung des elektronischen Handels

Stolpersteine

Orientation in Objects

- Verteilte Anwendungen sind nicht trivial
  - Transparente, plattform-unabhängige Kommunikation (Netzwerkfähigkeit, Internetfähigkeit)
  - Skalierbarkeit
  - Sicherheit
  - Transaktionales Verhalten auf Anwendungslogik-Ebene
  - Quality of Service
- Qualifiziertes Personal
  - Umfangreiches Know-How wird benötigt
  - Relativ hohe Einarbeitungszeit

Low Level Services

Orientation in Objects

- Das Entwickeln verteilter Anwendungen stellt besondere Anforderungen an den Entwickler
- Java bietet bereits viele sogenannte “low level services”
  - JMS – Java Messaging Service,
  - RMI – Remote Method Invocation,
  - JDBC, JTS, JAAS, u.s.w.

High Level Services

Orientation in Objects

- Durch Anwendung der Low Level Services werden dann höhere Dienste bereitgestellt
  - Persistence, Transaction , Data caching, Security
  - Error handling, Scalability and Fail-Over
  - Portability, Manageability

Enterprise Java Beans (= EJB)

Orientation in Objects

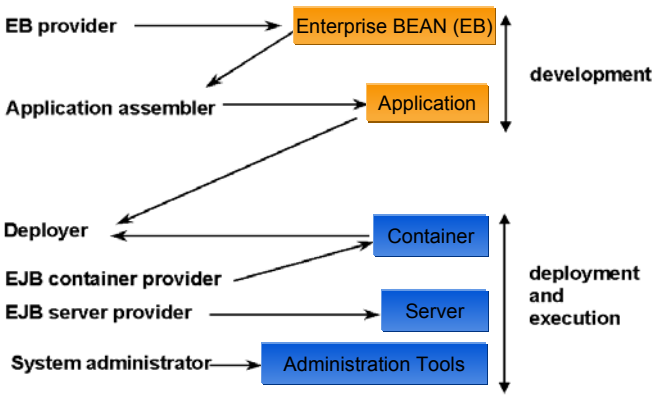
"The Enterprise JavaBeans architecture is a component architecture for the development and deployment of **component-based distributed business applications**.

Applications written using the Enterprise JavaBeans architecture are **scalable**, **transactional**, and **multi-user secure**.

These applications may be **written once**, and then **deployed on any server** platform that supports the Enterprise JavaBeans specification."

Verteilung der Rollen

Orientation in Objects



Persistenz Bean-managed

- Missverständlicher Name:
  nicht Bean-managed sondern „Bean Provider managed“
- Das bedeutet, der Geschäftslogik Entwickler ist gleichzeitig Persistenzexperte (und entwickelt deshalb eine DA-Schicht)
- Oder er kauft ein O/R-Tool

Persistenz Container-managed (CMP)

- Der Bean Provider hat die Zeit sich auf sein Aufgabenfeld zu konzentrieren, die Geschäftslogik
- Der Persistenz-Spezialist kann sein Knowhow kommerziell sehr effizient für viele nutzbar entfalten
- Die EJBs können bequem zwischen verschiedenen Application Servern und Datenbanken ausgetauscht werden.

=> Die Rollentrennung für den High-Level-Service Persistence funktioniert

Agenda

Orientation in Objects

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

Warum ist eine Entity keine Session Bean?

Orientation in Objects

- Vorschlag 1: Stateful Session Bean:

```
public class Person implements SessionBean {
   private String personId;
   private String name;
   private String christianName;

   public void ejbCreate(String personId) {
        this.personId = personId;
   }

   public String getName() {
        return name;
   }

   public void saveState() { /* SQL;SQL;SQL; */ }
   public void loadState() { /* SQL;SQL;SQL; */ }
};
```


Eine Entity ist eine Session Bean?

Orientation in Objects

- Vorschlag 1: Stateful Session Bean:

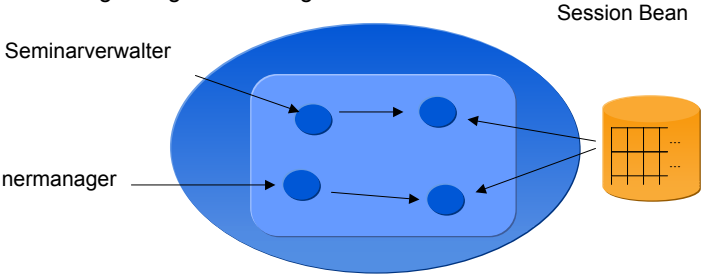
```
public class Person implements SessionBean, SessionSynchronization {

   public void afterBegin() {
        loadState();
   }

   public void beforeCompletion () {
        saveState();
   }

}
```


Eine Entity ist keine Session Bean

Orientation in Objects

- Person ist Trainer und Kursteilnehmer:
  - Seminarverwalter ändert Name wegen Schreibfehler
  - Trainermanager trägt Geburtstag ein.

Je eine Stateful Session Bean



=> Daten werden inkonsistent

=> Ressourcenverbrauch nach Ende des Gebrauchs

Warum ist eine Entity keine Session Bean?

Orientation in Objects

- Vorschlag 2: Stateless Session Bean:

```
public class Person implements SessionBean {

   public String getName(String personId) {
        Connection con = null;
        try {
          con = getConnection();
          PreparedStatement ps = con.prepareStatement(
            "SELECT Name FROM Person WHERE id=?"); // SQL;SQL;SQL;

          return resultSet.getString(1);
        } catch (/* SQL;SQL;SQL */) {
        }
   }
   public void setName() {...... //SQL;SQL;SQL;

   };
```


Vorteile Entity Beans

Orientation in Objects

- Eigene Callbacks informieren über Transaktionsfortschritt
- Unterstützung für gleichzeitigen Zugriff
- Caching zwischen Transaktionen möglich
- Persistenzcode kann generiert werden. (CMP)

Agenda

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

Finder

- Finder-Methoden sind nicht portabel
  - EJB 1.1 definiert kein Format für finder-Methoden
  - Jeder Container-Hersteller entwickelte sein eigenes
  - Verlust der Portabilität
  - Ein Bean Provider muss finder für alle gängigen Server anbieten

Finder

Orientation in Objects

- Finder-Methoden sind nicht portabel
  - EJB 1.1 (EJB QL) -Methoden
  - Jeder Container-Hersteller entwickelte sein eigenes
  - Verlust der Portabilität
  - Ein Bean Provider muss finder für alle gängigen Server anbieten

„Coarse grained“

Orientation in Objects

- Soll eine Entity Bean aus mehr bestehen als nur einer Zeile aus einer Tabelle?
  - Variante 1: Zugriff auf andere Entity Beans
    - **teuer**
    - **Relationsverwaltung infiziert Bean mit Persistenzcode**
  - Variante 2: Dependent Objects mit „normalen Objekten“
    - **Nicht im Deskriptor-Schema -> Mapping auf ER schwer**
    - **Bedarf für Container nicht erkennbar -> kein „load on demand“**
    - **Kein standard für „dirty bits“**
    - **Keine Typsicherheit in „Dependent Collections“**
    - **Probleme mit „Data aliasing“**

„Coarse grained“

Orientation in Objects

- Soll eine Entity Bean aus mehr bestehen als nur einer Zeile aus einer Tabelle?
  - Variante 1: Zugriff auf andere Entity Beans
    - **teuer**
    - **Relatio**
  - Variante 2: Dependent Objects mit „normale
    - **Nicht im Deskriptor-Schema -> Mapping auf ER schwer**
    - **Bedarf für Container nicht erkennbar -> kein „load on demand“**
    - **Kein standard für „dirty bits“**
    - **erheit in „Dependent Collections“**
    - **Probleme mit „Data aliasing“**

Local Interfaces

Persistenzcode

Container Managed Relations

Abstrakte Zugriffsmethoden

Container Managed Relations

Agenda

Orientation in Objects

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

Kompatibilität

- EJB 1.1 CMP ist EJB 2.0 CMP
  - außer einem kleinen Eintrag im DD
- Denn EJB 2.0 Container müssen mit dem EJB 1.1 CMP-Mechanismus entwickelte Beans unterstützen.

Agenda

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

EJB 1.1 Persistenzmanager

Orientation in Objects

- Zustände der Komponenteninstanz, die in einer Relationalen Datenbank persistent gemacht werden sollten, müssen:
  - In öffentlichen (public) Instanzattribute gespeichert werden
  - In dem „deployment descriptor“ als CMP Felder deklariert sein

CMP EJB 1.1 - öffentliche Attribute

Orientation in Objects

```
public class PersonBean implements EntityBean {

   public long id;
   public String name;

...
}
```

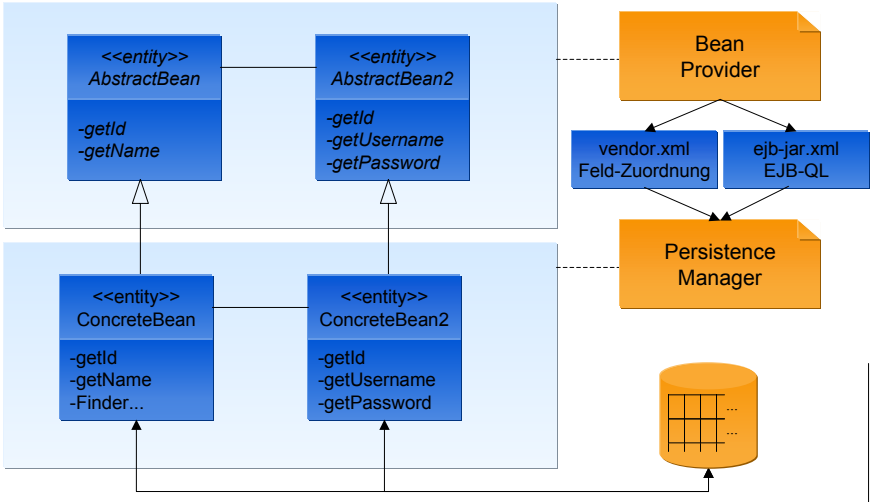

EJB 2.0 Persistenzmanager

Orientation in Objects

- Zustände der Komponenteninstanz, die in einer Relationalen Datenbank persistent gemacht werden sollten, müssen:
  - Als abstrakte „property“ (siehe JavaBeans):
    - **Abstrakte „getter“- und „setter“-Methode**
  - Dadurch ist die implementierende Beanklasse auch Abstrakt
  - In dem „deployment descriptor“ als CMP Felder deklariert sein

CMP 2.0 - abstrakte Zugriffsmethoden

Orientation in Objects



CMP 2.0 - Code

Orientation in Objects

```
public abstract class PersonBean implements EntityBean {

   public abstract long getId();
   public abstract void setId(long Id);
   public abstract String getName();
   public abstract void setName(String name);
   public abstract String getChristianName();
   public abstract void setChristianName(String cname);

}
```


CMP 2.0 - ejb-jar.xml

Orientation in Objects

```
<entity>
<ejb-name>Person</ejb-name>
..
<persistence-type>Container</persistence-type>
<cmp-version>2.x</cmp-version>
<abstract-schema-name>Person</abstract-schema-name>
<cmp-field>
<field-name>personId</field-name>
</cmp-field>
<cmp-field>
<field-name>name</field-name>
</cmp-field>
<cmp-field>
<field-name>christianName</field-name>
</cmp-field>
<primkey-field>personId</primkey-field>
..
</entity>
```

CMP 2.0 - jbosscmp-jdbc.xml

Orientation in Objects

```
<entity>
<ejb-name>Person</ejb-name>
<table-name>PERSON</table-name>
<cmp-field>
<field-name>personId</field-name>
<column-name>PERSON_ID</column-name>
</cmp-field>
<cmp-field>
<field-name>name</field-name>
<column-name>NAME</column-name>
</cmp-field>
<cmp-field>
<field-name>christianName</field-name>
<column-name>CHRISTIAN_NAME</column-name>
</cmp-field>
</entity>
```


CMP 2.0 - Feinheiten

Orientation in Objects

- Verboten:

```
public String christianName;
public String getName() {
     return name;
}
```

- Erlaubt:

```
public String myNonPersistentField;
public String getFullName() {
     return getChristianName() + " " + getName();
}
```

CMP - Mögliche Optimierungen

- Schreiboptimierung
  - Nur die veränderten Felder werden weggeschrieben
- Leseoptimierung
  - Lazy Loading / Aggressive Loading
  - Feldgruppen

Agenda

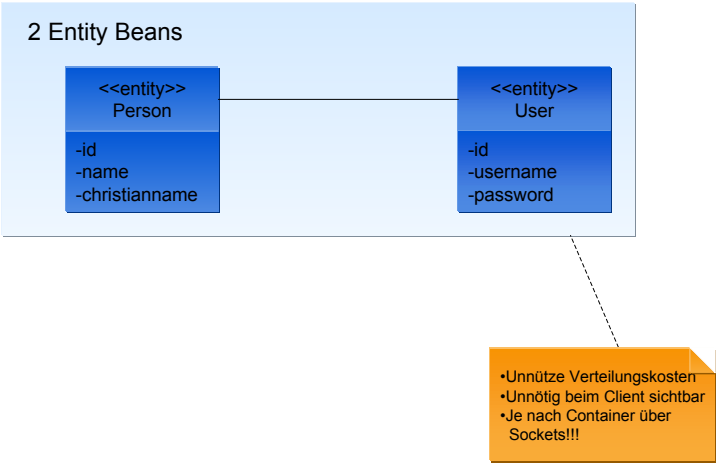
- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

Local Interfaces - Vorgeschichte

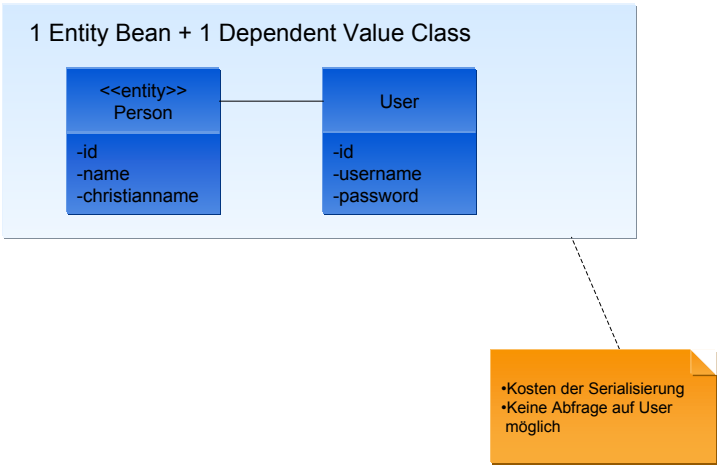
- Konsequenzen von ausschließlichen remote-Aufrufen
  - nicht viele einzelne Aufrufe - jeder ist teuer
- Konsequenzen von "call by value"
  - nicht gerne große Daten übertragen - nur per value möglich

=> wenige Aufrufe mit mittelkleinen Daten

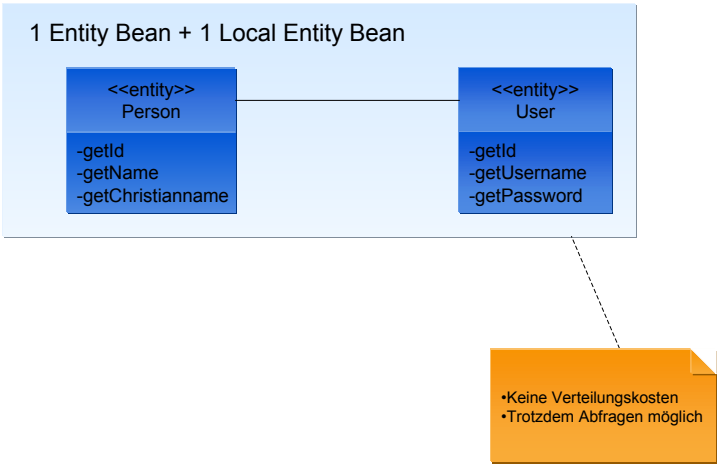
EJB 1.1 - Variante Remote Entity



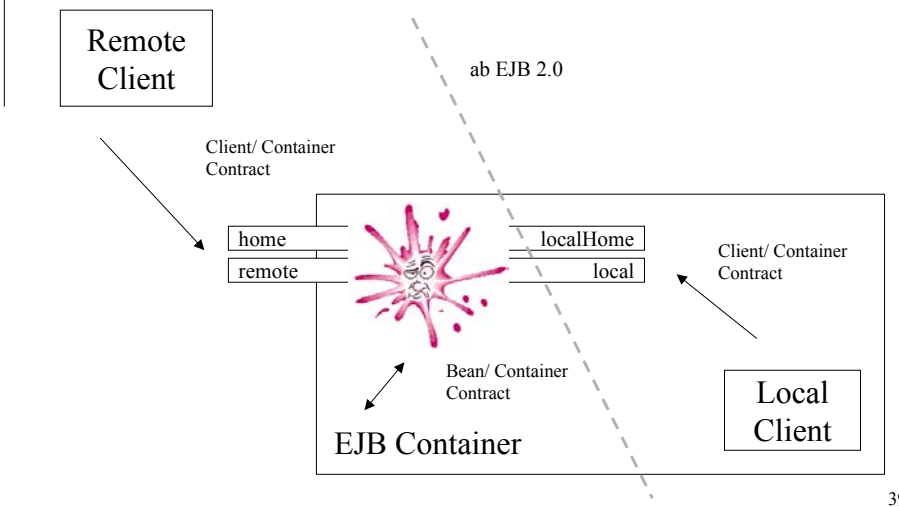
EJB 1.1 - Variante Dependent Value Class



EJB 2.0 - Variante Local Entity



Container & Bean



Local Entity <-> Dependent Value Class

Local	Value	
N	J	Kann Wert eines CMP-Felds sein
J	N	Kann Wert eines CMR-Felds sein
N	J	Kann Parameter in Remote Interface. sein
J	J	Kann Parameter in Local Interfaces sein
Cont.	Comp.	Persistenzmechanismus
Cont.	Comp.	Lebenszyklus wird gesteuert von...
J	N	Benutzbar in EJB QL Abfrage
N	J	Serialisierbar

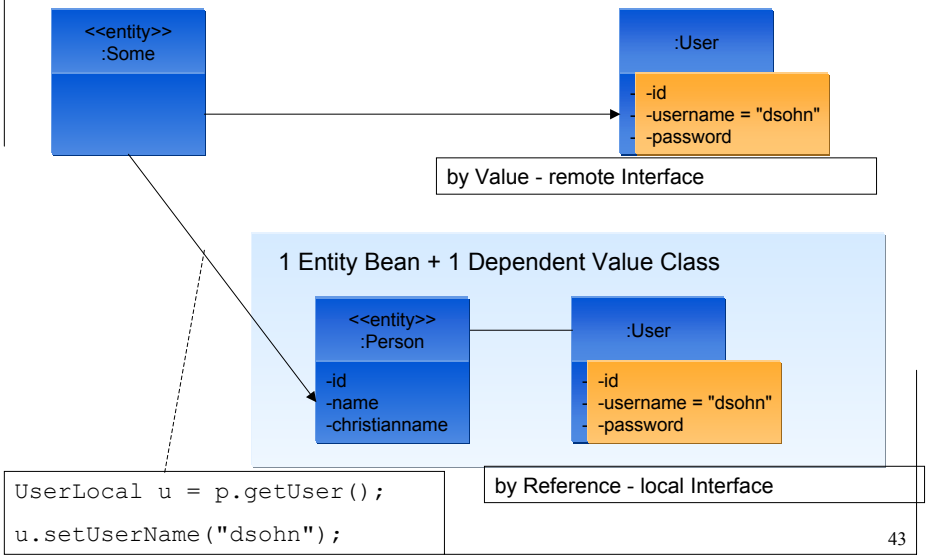
Unterschiede zu Remote Interfaces

- Objekte sind „normale lokale Java-Objekte“ - kein RMI over IIOP
- Parameterübergabe „by reference“ und nicht „by value“ statt (Vorsicht bei Migration von 1.1 nach 2.0)
- Beans sind nicht mehr ortsunabhängig.
- Dürfen nicht als Zustand anderer Beans gespeichert werden

Remote vs. Locale Interface

- Vorteile Remote Interfaces
  - Schwache Kopplung zwischen Client und Bean
  - Verteilungsunabhängigkeit
  - Isolation - call by value
- Vorteile Locale Interfaces
  - Preiswerter Zugriff
  - Möglichkeit Daten zu teilen - call by reference
  - CMR

By Reference - By Value



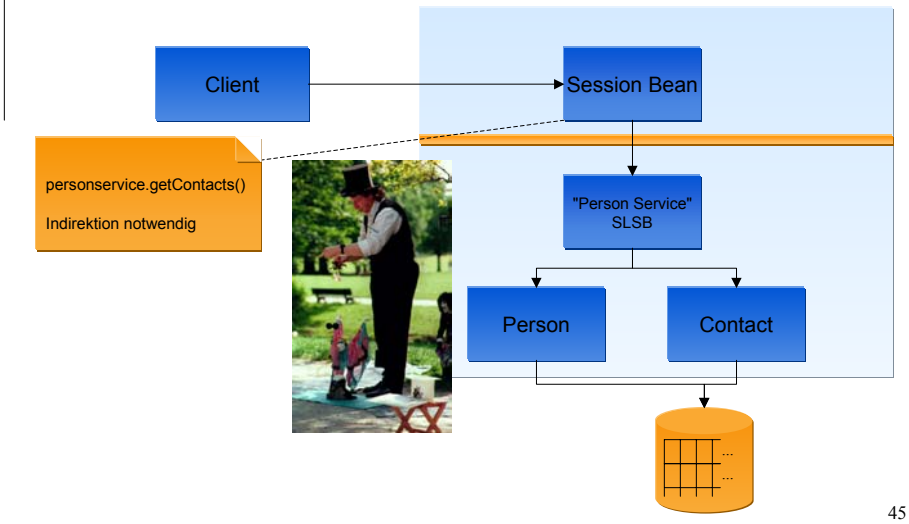

```
UserLocal u = p.getUser();
u.setUserName("dsohn");
```


Agenda

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

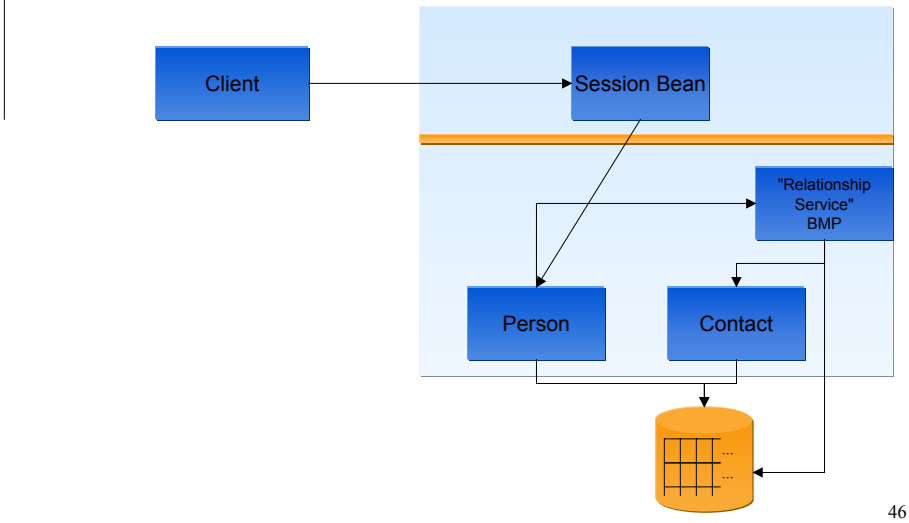
Beziehungen vor EJB 2.0 (1)

Orientation in Objects

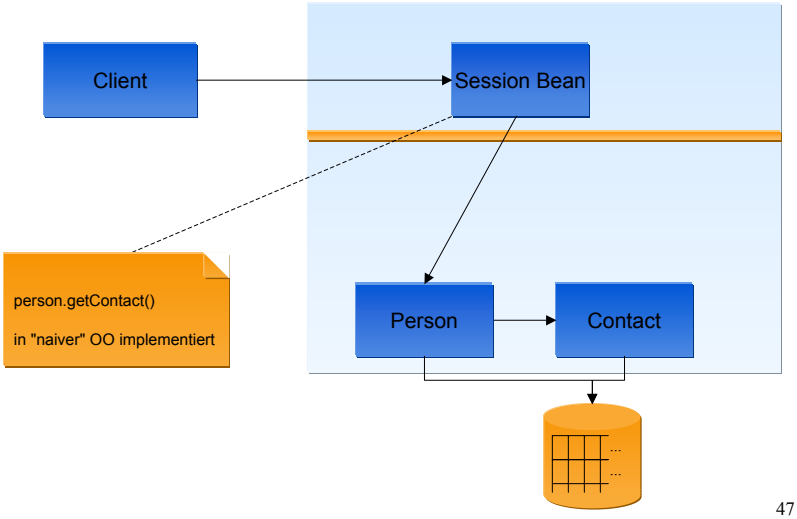


Beziehungen vor EJB 2.0 (2)

Orientation in Objects



Beziehungen mit EJB 2.0



Beziehungen

- Neue Möglichkeit zu Container-Managed Relationships CMR
- Varianten:
  1-1
  1-n
  n-m
  uni- oder bidirektional
- Das Ziel der Relation muss ein Local Interface sein, die Quelle muss dies nicht.
- Nur ohne Attributierung möglich

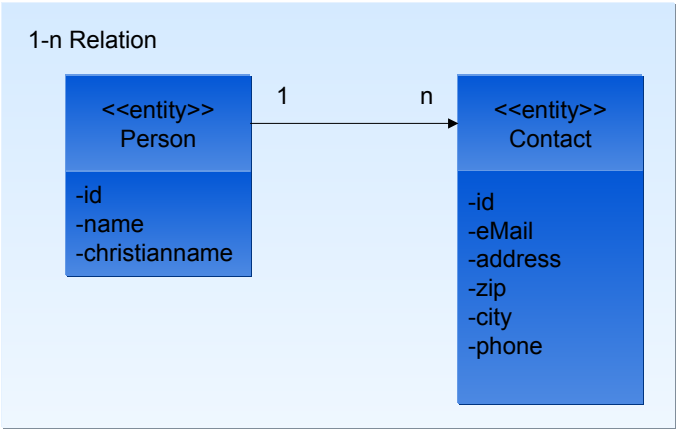
Beziehungen - Anwendungsbereich

Orientation in Objects

- Für den Zugriff auf dem aktuellen Objekt zugeordnete Objekte
- Zur Navigation über das Objektmodell bei EJB QL-Abfragen
- Automatische Verknüpfung mit dem Lebenszyklus von Komponenten möglich (Komposition)

Beziehungen - Beispiel

Orientation in Objects



Beziehungen - Code (1)

Orientation in Objects

```
public abstract class PersonBean implements EntityBean {

    public abstract long getId();
    public abstract void setId(long Id);
    public abstract String getName();
    public abstract void setName(String name);
    public abstract String getChristianName();
    public abstract void setChristianName(String cname);

    public abstract Collection getContacts();
    public abstract void setContacts(Collection contacts);

}
```


Beziehungen Code (2)

Orientation in Objects

```
public abstract class PersonBean implements EntityBean {

    public void addContact (ContactView contact)
         throws createException {

         try {
           Contact c = createContact(contact.getId(),.....);
           getContact().add(c);
         } catch (CreateException e) {}

    }
```

Manipulation von Relationen

Orientation in Objects

- Man ruft bei einer Relation mit einem singulären Ende die setXX-Methode auf.
- Bei einer Relation mit einem mehrfachen Ende benutzt man die add-Methode der Collection.
  - Die setXX-Methode dient hier zum Austausch ganzer Relationen

Beziehungen ejb-jar.xml

Orientation in Objects

```
<relationships>
<ejb-relation>
<ejb-relation-name>PersonBean-Contact</ejb-relation-name>
<ejb-relationship-role>
<ejb-relationship-role-name>person-has-many-contacts</ejb-relationship-role-name>
<multiplicity>One</multiplicity>
<relationship-role-source>
<ejb-name>Person</ejb-name>
</relationship-role-source>
<cmr-field>
<cmr-field-name>contacts</cmr-field-name>
<cmr-field-type>java.util.Collection</cmr-field-type>
</cmr-field>
</ejb-relationship-role>
<ejb-relationship-role>
<ejb-relationship-role-name>Contact-belongs-to-one-Person</ejb-relationship-role-
    name>
<multiplicity>Many</multiplicity>
<cascade-delete/>
<relationship-role-source>
<ejb-name>Contact</ejb-name>
</relationship-role-source>
</ejb-relationship-role>
</ejb-relation>
</relationships>
```

Beziehungen jbosscmp-jdbc.xml

Orientation in Objects

```
<relationships>
<ejb-relation>
<ejb-relation-name>PersonBean-Contact</ejb-relation-name>
<foreign-key-mapping/>
<ejb-relationship-role>
<ejb-relationship-role-name>person-has-many-contacts</ejb-relationship-role-name>
<key-fields>
<key-field>
<field-name>personId</field-name>
<column-name>PERSON_FK</column-name>
</key-field>
</key-fields>
</ejb-relationship-role>
<ejb-relationship-role>
<ejb-relationship-role-name>Contact-belongs-to-one-Person</ejb-relationship-role-
    name>
</ejb-relationship-role>
</ejb-relation>
</relationships>
```


Component Interfaces als CM(X)-Felder

Orientation in Objects

- Ein Remote Interface ist ein legaler Wert für ein CMP-Feld
- Ein Local Interface ist ein legaler Wert für ein CMR-Feld
  - und vermutlich meist der bessere Kandidat eine Beziehung abzubilden.
- Ein Local Interface im CMP-Feld ist so illegal wie ein Remote Interface im CMR-Feld
- Local Interface Typen, Zugriffsmethdoen von CMR und Collections aus einer CMR dürfen nicht im Remote Interface benutzt werden.

<ejb-ref> vs. <ejb-local-ref>

- Referenzen, die man über eine Zugriffsmethode auf eine Beziehung (oder eine ejbSelect-Methode) erhält, müssen nicht im DD deklariert werden (sie benutzen den JNDI Namensraum nicht)

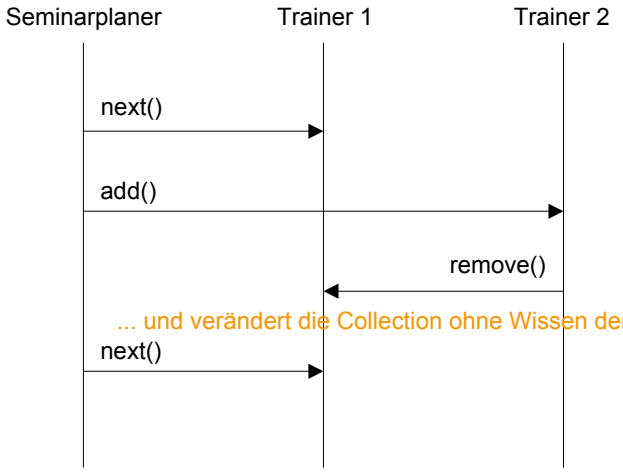
Beziehungen - Referentielle Integrität

- Der Container ist verantwortlich für die referentielle Integrität
- Die gilt für 1-1 wie auch für die eine Seite der 1-n Relationen.
- Beispiel: Ein setXX-Aufruf bei einer 1-1-Relation kann 4 Objekte beeinflussen.
- Beispiel: Wenn man einer Collection bei einer 1-n-Relation mit add einen Wert hinzufügt, der schon Mitglied einer anderen Relation ist, wird er dort gelöscht.

Fail Fast Iteration (1)

Orientation in Objects

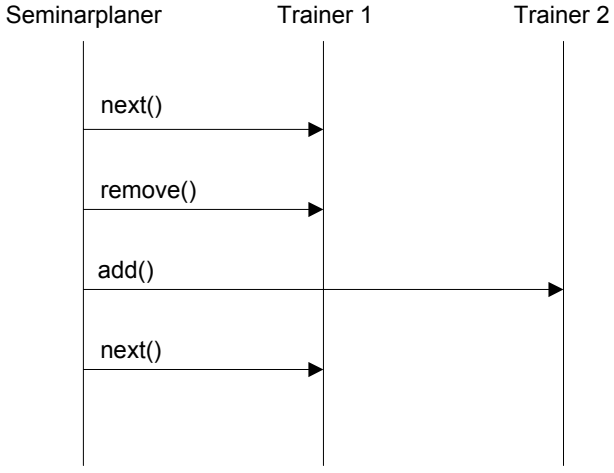
Seminarplaner gibt Kurse an einen andere Trainer...



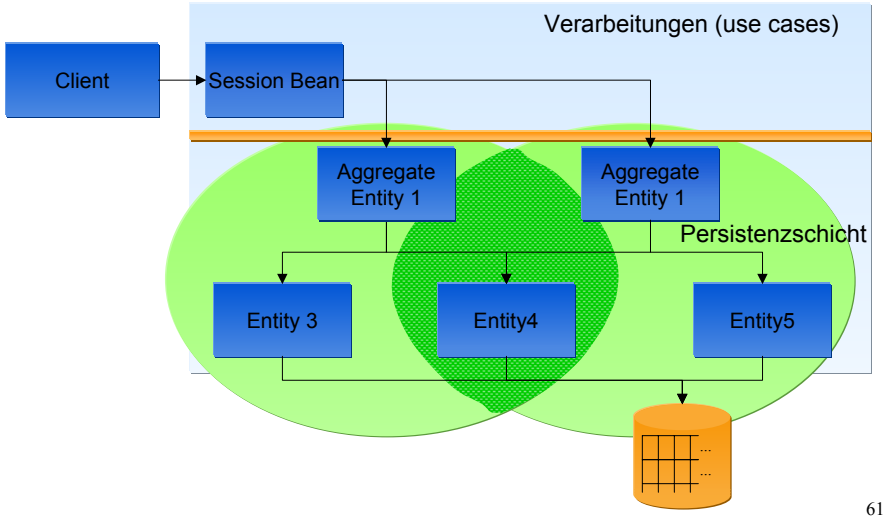
Fail Fast Iteration (2)

Orientation in Objects

Seminarplaner plant richtig, die Iteration bekommts mit



Achtung: Anti Pattern "Clotted Server"



Agenda

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

Finder in EJB 1.1 und jBoss

Orientation in Objects

- werden in der Datei „jaws.xml“ deklariert!
- Implementierte „finder“
  - Methode ejbFindXXX in Bean implementieren
  - Methode findXXX in Home deklarieren
  - Ist EJB 1.1 Standard
- Automatische „finder“ (nur in Home deklarieren!)
  - findAll()
  - findByPrimaryKey(PKKlasse pkWert)
  - findByAttributeName(AttributKlasse wert)
- Aber nur in JBoss so (es war nichts spezifiziert)

jaws.xml - Deployment descriptor

Orientation in Objects

```
<jaws>
   <enterprise-beans>
     <entity>
       <ejb-name>account</ejb-name>
       <finder>
         <name>findRichAccounts</name>
         <query>balance > {0} </query>
         <order>accountId</order>
       </finder>
     </entity>
   </enterprise-beans>
 </jaws>
```

Abfragesprache EJB QL

- DBMS-Unabhängigkeit durch eigene Abfragesprache
  - wird in ejb-jar.xml deklariert
- Finder werden portabel
- Teilmenge von SQL 92 mit Erweiterungen für die Navigation über Beziehungen

Anmutung EJB QL


```
SELECT OBJECT ( identificationVariable )
FROM range variable [ AS ] identificationVariable
[ WHERE value comparison value ]
```

```
SELECT  pathExpression
FROM collectionMemberDeclaration [ AS ] id.Var.
[ WHERE value comparison value ]
```


FROM-clause

Orientation in Objects

- Arbeitet entweder auf dem Schema einer Enitity-Bean im DD

```
abstractSchemaName [AS] identifier
```

- Oder einem Pfadausdruck über eine Beziehung aus dem DD

```
IN collectionValuedPathExpression [AS] identifier
```


FROM-clause mit Bean Schema

Orientation in Objects

- In SQL kann man, in EJB QL muß man einen *identifier* für einen *abstract schema name* vergeben:

```
FROM Event e, Seminar s
```

- oder

```
FROM Event AS e, Seminar AS s
```

FROM-clause mit "Beziehungen"

Orientation in Objects

- Collection Member Declaration:
  - Mehrwertiger Pfadausdruck in einem IN-Operator.
  - Pfadausdruck:
    Identification Variable . *CMP-Field* | *CMR-Field*

```
FROM Seminar s, IN(s.events) e
```


Pfadausdrücke - Typen

Orientation in Objects

- Mehrwertige Pfadausdrücke
  - Identifizieren mehrere Werte
  - Nur sie werden in from-clause benutzt (ganz selten in where)

- Einwertige Pfausdrücke
  - Identifizieren einzelne Werte
  - Nur sie werden in select-clause und where-clause benutzt.

Pfadausdrücke richtig / falsch

Orientation in Objects

- falsch

```
SELECT OBJECT(u) FROM TRAINER t, IN(t.user) u
```

- richtig

```
SELECT t.user FROM TRAINER t
```

- falsch

```
SELECT OBJECT(s.events) FROM Seminar s
```

- richtig

```
SELECT OBJECT(e) from Seminar s, IN(s.events) e
```


SELECT-clause

Orientation in Objects

- Ohne *Distinct* ist nichts spezifiziert

```
SELECT Object(e)
FROM Event e, Seminar s
```

- Mit *Distinct* wird's eindeutig

```
SELECT DISTINCT Object(e)
FROM Event e, Seminar s
```

- Entweder OBJECT(s) oder s.events
  Nicht s und nicht OBJECT(s.events)

Remote oder Local ale Ergebnis?

Orientation in Objects

- Man siehts der Abfrage nicht an - was kommt zurück?:

```
SELECT Object(s) FROM Seminar s
```

- Bei finder in einem local home — local interface
- Bei finder in einem remote home — remote interface
- Bei ejbSelect ohne `<result-type-mapping>` — local interface
- Bei ejbSelect mit `<result-type-mapping>` — je nach mapping

EJB-QL Funktionen

Orientation in Objects

- String Funktionen:
  - CONCAT(String, String) returns a String
  - SUBSTRING(String, start, length) returns a String
  - LOCATE(String, String [, start]) returns an int
  - LENGTH(String) returns an int
- Arithmetische Funktionen:
  - ABS(number) returns a number (int, float, or double)
  - SQRT(double) returns a double

Beispiele für EJB QL

Einfache Abfrage

```
SELECT OBJECT(s)
FROM Seminar AS s
```

Abfrage mit Parameter (aus Query oder Select)

```
SELECT OBJECT(s)
FROM Seminar AS s
WHERE s.title = ?1
```

Abfrage über Relationen hinweg:

```
SELECT OBJECT(e)
FROM Seminar s, IN( s.events) e
WHERE e.place < ?1
```


EJB-QL Probleme

- (Datum und Zeit **Literale** müssen als Millisekunden (primitiver Datentyp LONG) angegeben werden)
- (Keine Festkommazahl in arithmetischen Vergleichen möglich)
- (Unterstützt keine Kommentare)

- Keine Aggregatfunktionen
- Keine Multi-Object Selects
- Keine dynamische Erzeugung
- ...

- EJB 2.1 tut hier was

Agenda

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

Finder

- Finder-Methoden
  - werden immer in „home interfaces“ deklariert:
    - **im „remote home interface“**
      - liefern „remote interfaces“ oder entsprechende „collections“ zurück
    - **im „local home interface“**
      - liefern „local interfaces“ oder entsprechende „collections“ zurück
  - Implementierung von *findByPrimaryKey* ist Pflicht
- EJB 1.1. finder nicht zwingend umsetzbar

EJB QL im Deployment descriptor

Orientation in Objects

```
<query>
   <query-method>
        <method-name>findByBalance</method-name>
        <method-params>
               <method-param>long</method-param>
        </method-params>
   </query-method>
   <ejb-ql>
        SELECT OBJECT(a)
        FROM Account a
        WHERE a.balance = ?1
   </ejb-ql>
</query>
```


Selektoren

Orientation in Objects

- Rückgabewerte können auch CMP und CMR-Felder (inklusive Collections) sein.
- Sind nur in der Bean sichtbar (nicht für Clients!!!)
- Werden deshalb nicht im Home, sondern in der Bean-Klasse spezifiziert.

Selektoren Code


```
public abstract class PersonBean implements EntityBean {

    public abstract long getId();
    public abstract void setId(long Id);
    public abstract String getName();
    public abstract void setName(String name);
    public abstract String getChristianName();
    public abstract void setChristianName(String cname);

    public abstract Collection getContacts();
    public abstract void setContacts(Collection contacts);

    public abstract Collection ejbSelectAllPrivatContacts()
        throws javax.ejb.FinderException;
}
```


Agenda

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

BMP / CMP bei EJB 2.0

Orientation in Objects

- CMP 2.0 spart viel Arbeit
- aber das Speichermedium ist nicht spezifiziert
  - ein Container, der serialisiert, kann EJB 2.0 konform sein

=> Es kann unvermeidbare Container geben, deren CMP nicht auf die Bedürfnisse passt.

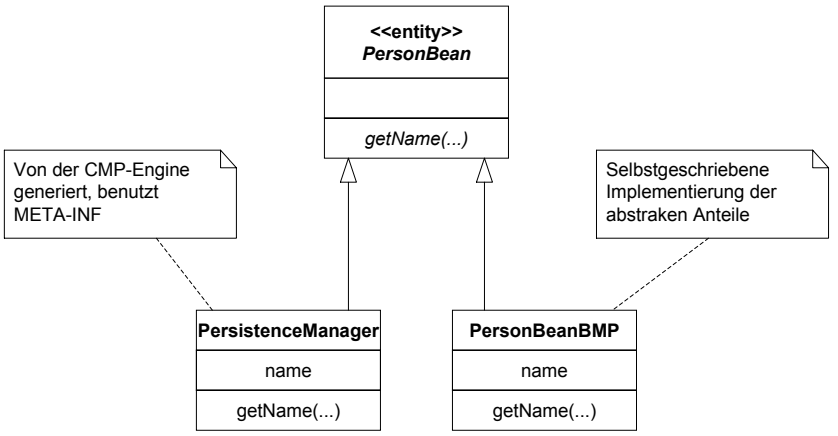
- Kann Ihr CMP-Mechanismus auf ERP-Daten mappen?
- Oder wenigstens auf Ihr bestehendes RDB-Schema?

Wege zur Vermeidung von BMP

Orientation in Objects

- Container wechseln
  - die Finanzen oder die Strategie oder der Kunde oder...
- O/R-Mapping Tool einsetzen
  - die Finanzen oder das Proprietäre oder der Kunde oder...
- Design ändern
  - immer nur eine Tabelle für eine Entity is doof....
- Aufgeben
  - dann aber wenigstens mit Stil

Harmonischer Umgang CMP / BMP



Nutzen von BMP

- Umgehung der Grenzen der CMP 2.0
  - Bulk update
  - Multi object select
  - Aggregate
- Zugriff auf EIS über Connector
- Generell legacy Behandlung

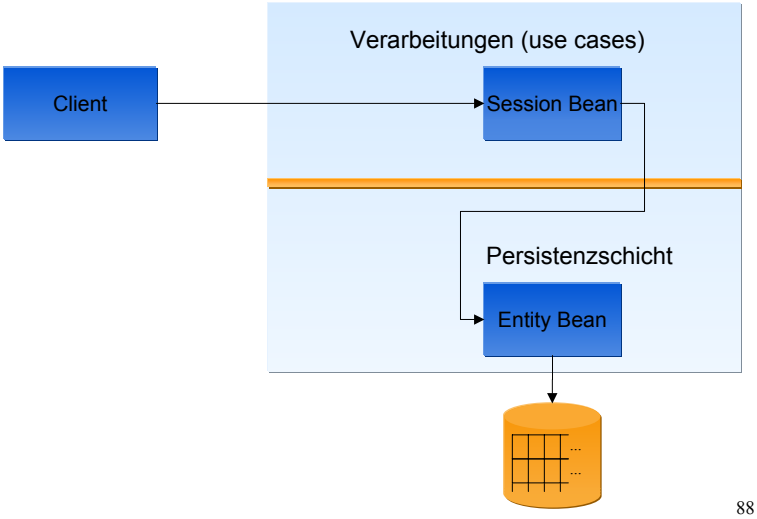
Agenda

Orientation in Objects

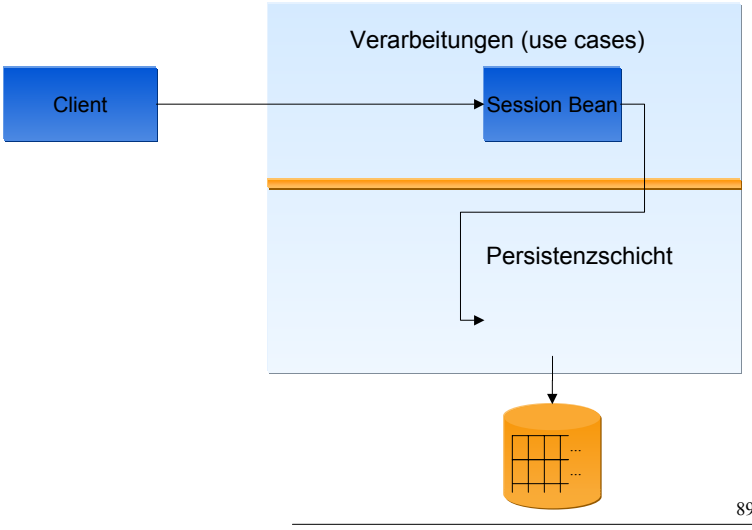
- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

Naive Vorstellung - Facade

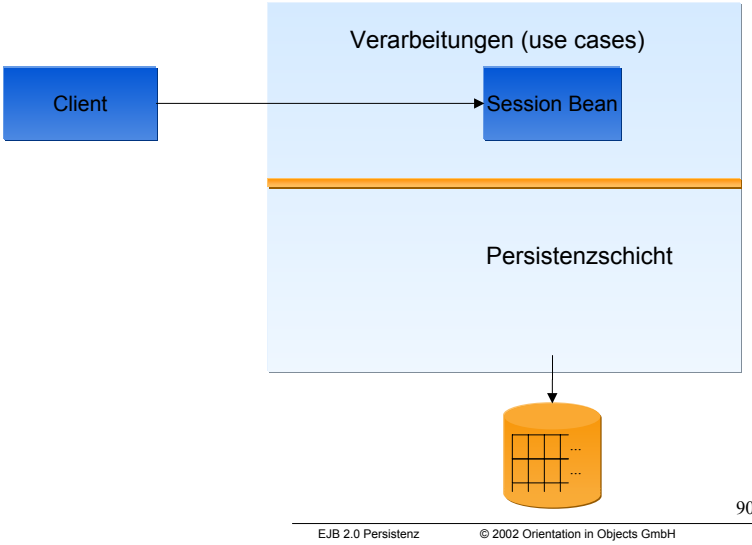
Orientation in Objects



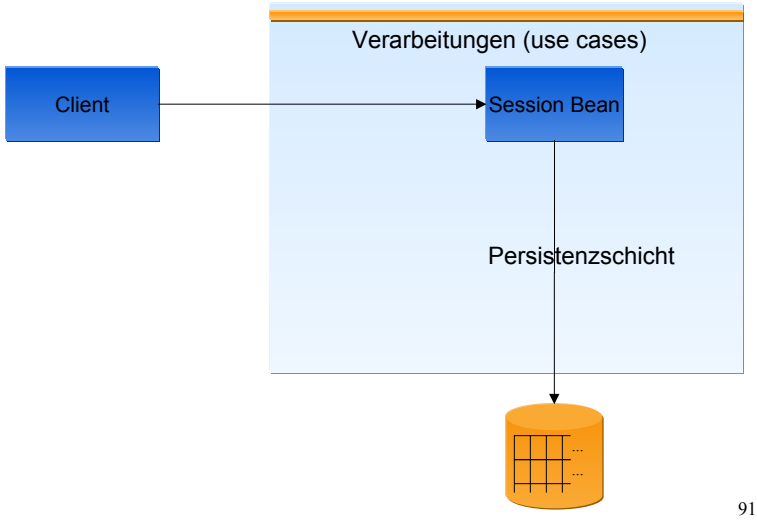
Naive Vorstellung - Facade



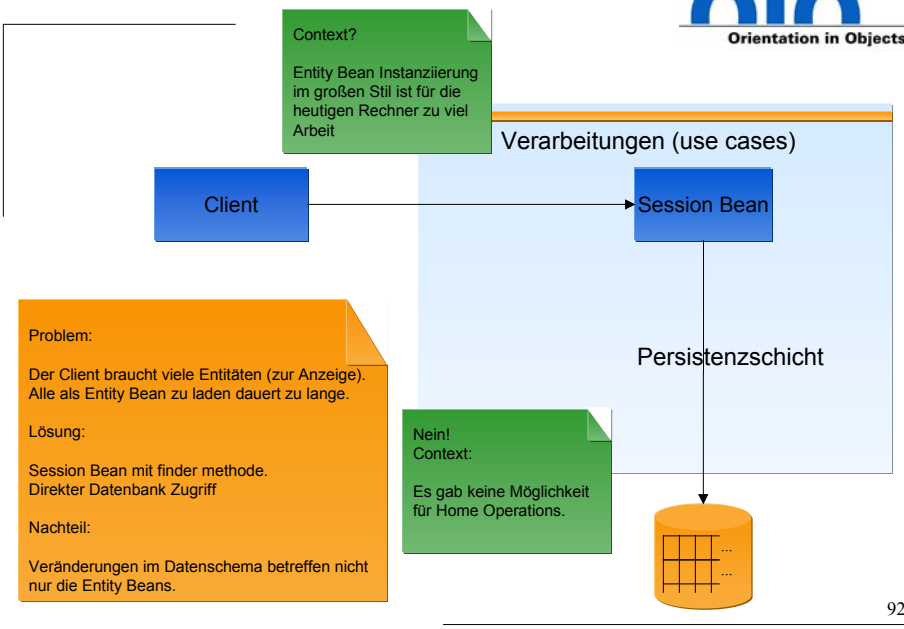
Naive Vorstellung - Facade



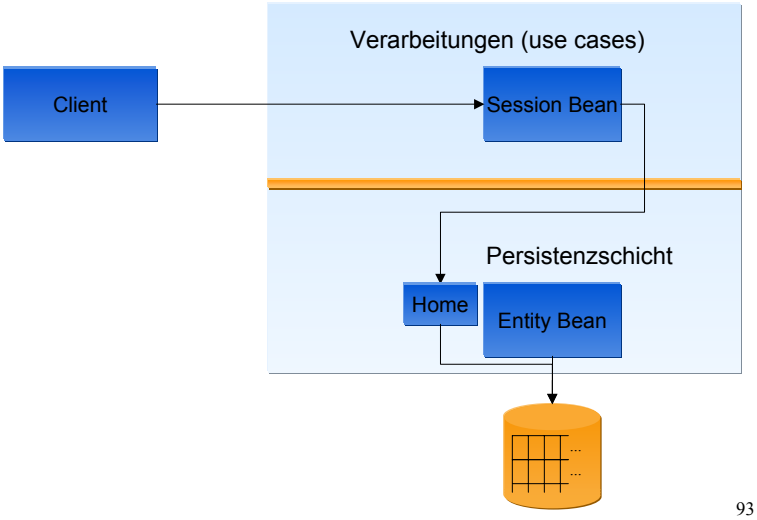
Naive Vorstellung - Facade



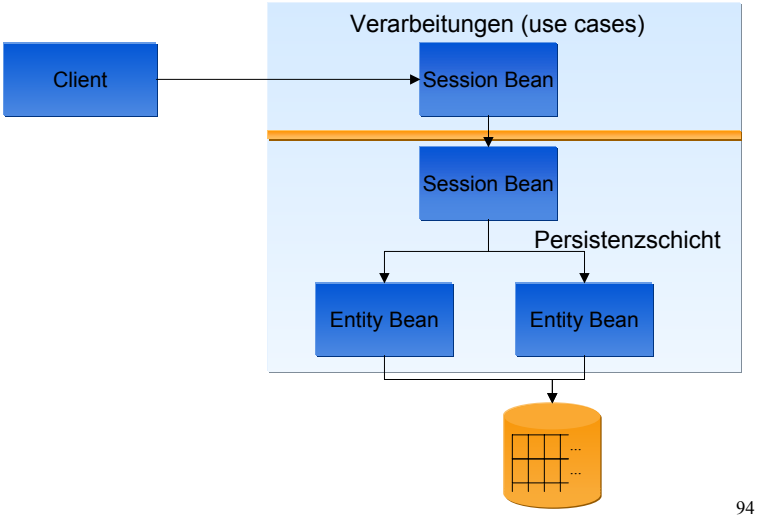
Pattern: Bulk Reader



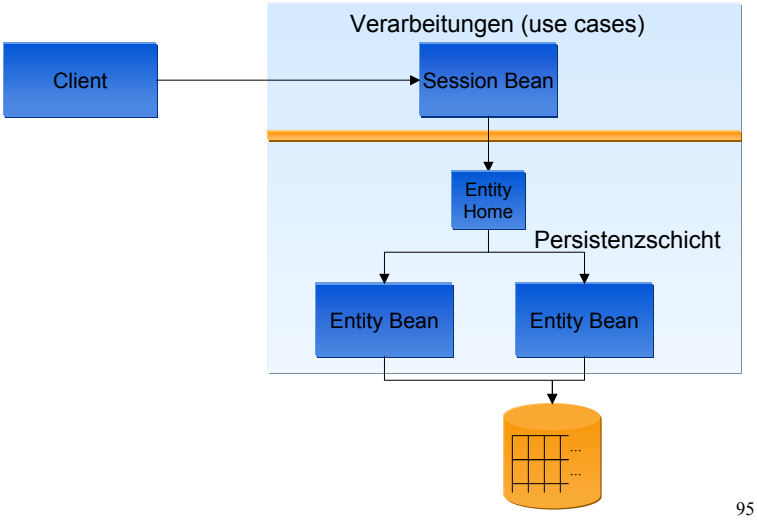
Bulk Reader mit Home Operation



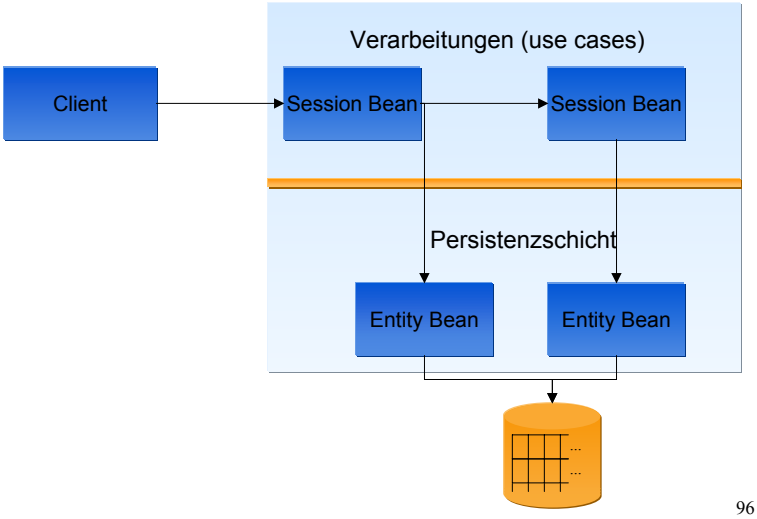
Service Facade



Service Facade



Session Facade



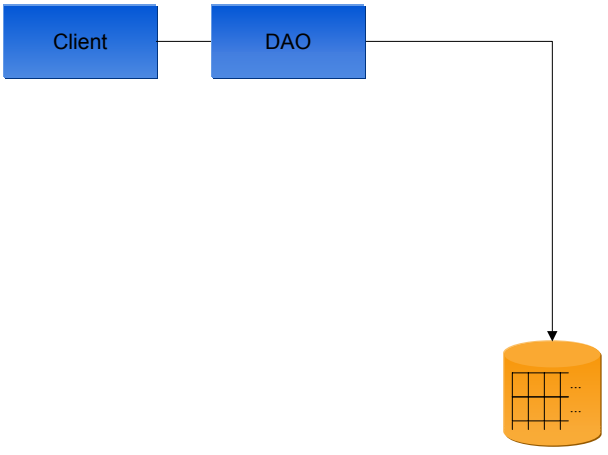
Weitere Kontextänderungen für Patterns

Orientation in Objects

- Relationship Service
  - Für Relationen ohne Zusatzinformationen nicht mehr benötigt, man benutzt einfach CMR
- Data Access Object
  - Bevölkerungsabnahme erwartet, wegen zunehmender Nutzung der CMP
- Dependent Object
  - Sind weniger wichtig wegen Local Entity Beans

Noch ein J2EE Pattern: Keine EJBs

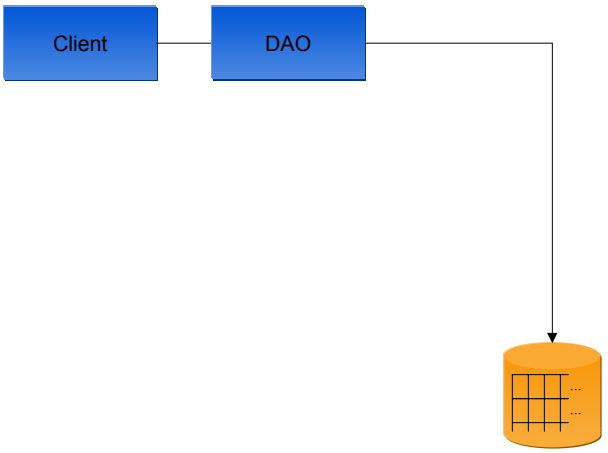
Orientation in Objects



Fast Lane Reader

Orientation in Objects

http://unimut.fsk.uni-heidelberg.de/unimut/schwob?schwob_url=http://www.oio.de



Agenda

Orientation in Objects

- Was ist EJB Persistenz
- Wozu braucht man Entity Beans
- Einige Probleme der EJB 1.1 Persistenz
- EJB 2.0 Persistenz
  - Container Managed Persistence
  - Local Interfaces
  - Container Managed Relationships
  - EJB-QL
  - Selektoren und Finder
- EJB Persistenz = Entity Beans mit CMP?
- Einige mögliche Designänderungen durch EJB 2.0
- Schlussgeplänkel

Ankündigungsdaten J2EE 1.4 / EJB 2.1

- Derzeit nur aus den JSR-Verläufen:
  - EJB 2.0
    - **Participant Review** **26.05.2000**
    - **Public Review** **06.07.2000**
    - **Proposed Final Draft** **25.10.2000**
    - **Proposed Final Draft** **26.04.2001**
    - **Final Release** **24.09.2001**
  - EJB 2.1
    - **Expert Group formation** **23.10.2001**
    - **Community review** **27.05.2002**

Generelle Aussichten EJB 2.1 (JSR 153)

- Web Services
- Message-driven Beans Erweiterungen
- Web Services
- Timer Service
- Web Services
- EJB QL Erweiterungen
- Web Services

EJB 2.1 Aussichten Persistenz

Orientation in Objects

- Leider durch Web Services und Web Services eingeschränkt
- ORDER BY

```
SELECT OBJECT(c) FROM Person p, IN(p.contacts) c WHERE
  p.name = 'bayer' ORDER BY c.street
```

- Aggregate Operators
  - avg, count, max, min, sum

```
SELECT AVG(p.assessment) FROM EVENT e,
  IN(e.participations) p WHERE e.place = 'Mannheim'
```

- null value handling und query results werden geklärt

```
SELECT c.phone FROM Person p, IN(p.contacts) c WHERE
  p.user.name = %1 AND c.email IS NOT NULL
```


Wünsche an EJB 2.2

Orientation in Objects

- Read-Only Entity Beans mit CMP
- Plug-In-Mechanismus für Persistence Manager
- Wege zur Behandlung großer ResultSets
  - Kein Iterator-Support
  - Keine Bulk-Manipulation
- Weitere Datenbankintegration
  - z.B. für Stored Procedures
- Vererbung

J2EE 1.3-Server 03.05.2002

Orientation in Objects

- Borland Borland Enterprise Server 5.0.1
- Fujitsu INTERSTAGE Application Server Enterprise Edition V5.0L10
- Macromedia Macromedia Jrun Server 4.0
- Pramati Pramati Server 3.0
- SilverStream Silverstream Application Server 4.0
- Sun Java 2 SDK, Enterprise Edition 1.3.1
- Sybase Sybase Enterprise Application Server 4.1
- Trifork Trifork Application Server 3.0
- BEA Bea Weblogic Preview 7.0
- IBM Websphere Techology for Developers
- CA Advantage Joe
- SAS AppdevStudio 2.0.2 Preview release

http://java.sun.com/j2ee/compatibility.html

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit

Dirk M. Sohn
sohn@oio.de
http://www.oio.de

ToDo Montag

ToDO

- Zugriff auf Attribute vom Typ Collection nur innerhalb einer Transaktion möglich -> mehrwertige Assoziationen nur innerhalb einer Bean und nicht in einem Client nutzbar. (JDO-Artikel) - was bedeute das
- ein stücken outparkt-navigationscode bei beziehungen dranbringen?
- Ein artikel hat behauptet, daß der zwang zu ejbload und ejbstore gelockert wurde in 2.0??
- Müssen locals im gleichen Container oder in der gleichen VM oder wo bleiben.??

ToDo außerhalb Vortrag

Orientation in Objects

- „Lebensweg einer Java-Anwednung wieder in J.MAN zurücktun
- DTO-Factory in JAPS aufnehmen
- DataTransfer RowSets ebenso

EJB 2.0 Testsuite - Aliasing

Orientation in Objects

- Aliasing Beispiel Code
  - Gemeinsames Dependent Value Object für zwei Entities, die eine Entity sagt der anderen, sie soll was ändern, und kuckt dann, ob sich bei der eigenen was geändert hat.
  - Gemeinsames EntityBean über lokales Interface und gleiches Spiel Wenns Reicht dann auch Sequenzen malen.

EJB 2.0 Testsuite - Integrität bei Relationen

Orientation in Objects

- Refernetielle Integrität bei Relationen
  - vielleicht auch tests - einen 1-1 realtion ist exklusiv, keiner der beiden partner darf an einen anderen gleichzeitig gebunden werden (was passiert) - eine 1-n ist auch in eine richtung exklusiv
  - Wenn dann auch noch Folie (bei Collections 1-n, wenn man eine added, die schon eine hatte, wird die alte gelöst.)

EJB 2.0 Testsuite - Update

Orientation in Objects

- Kontrolle des Appservers über Konstruktion eines Updates durch abstrakte getter und setter
  - Eine Person wird vor jedem Test mit Userdaten angelegt.
  - Dann wird setUserData an Person aufgerufen mit
    - **einem UserData-Objekt ohne Änderungen -> Kein Update**
    - **mit einer Änderung -> Update mit einem set**
    - **mit nur Änderungen -> volles Update**

EJB 2.0 Testsuite - DTO-Notwendigkeit

Orientation in Objects

- Mit Einführung der Local Interfaces sollte es unnötig geworden sein, DTOs über Local Interfaces zu verwenden.
  - Bei einer Person sollen die Userdaten aktualisiert werden.
  - Person bietet 2 Methoden in seinem remote Interface an:
    - **setUserDataBulk()**
    - **setUserDataSingleCalls()**
  - Probleme: Derzeit verlieren wir die meiste Zeit mit den remote calls, die eventuellen kleinen Unterschiede gehen unter.
    - **Gelöst duch log4j**
  - Neue Frage, warum ist Buld 40% schneller als Single Calls obwohl beide nur 1 Transaktion in die DB Committen mit dem letzte User
    - **Theorie: wegen Hotspot? (mal mit -classic testen)**

EJB 2.0 Testsuite - Riesen Batch

Orientation in Objects

- Sperre allen Trainern, deren Durchschnittsnote in mehr als 50% ihrer Seminare unter 50% liegt, den User.
  - Zuerst mit voller EJB-Navigation:
    - **finde alle Trainer - für alle trainer:**
    - **getSeminars,getEvents,getParticipations**
    - **calc PartAvg for Seminar. Count Seminars, compare(#PartAvg <50%, CountSem * 50%)**
    - **If necessary user=invalid.**

EJB 2.0 Testsuite - Rollen als Beans

Orientation in Objects

- Variante 2 Beans 1 Tabelle
  - Trainer: Person mit Qualifiaktion für Seminar
  - Teilnehmer: Person mit Teilnahme an Event
- Was liegt näher als 2 EntityBeans mit der entsprechenden Geschäftslogik auf die Tabelle Person zu persistieren.
- Dies scheint auch zu gelingen, bis auf die CMR-Felder von Person
- Beim Mapping von den 2 Enden der Relationen Trainer->User und Teilnehmer->User behauptet der Container, beide können nicht in der selben Spalte landen.

EJB 2.0 Testsuite - Rollen als Interfaces

Orientation in Objects

- Variante 2 Remote Interfaces 1 Bean 1 Tabelle
  - Trainer: Person mit Qualifiaktion für Seminar
  - Teilnehmer: Person mit Teilnahme an Event
- Was liegt näher als 2 Rollen als unterschiedliche Interfaces ein und derselben Bean für verschiedene Clienten zu implementieren.

EJB 2.0 Testsuite - Collection vs. ejbselect

- Viele Contacts durchiterieren von Person aus vs. ejbSelectMethode um einen bestimmten Contact zu finden.
- Ab wieviel Contacts ist "break even"

EJB 2.0 Testsuite - TimeTracer-Framework

- Fertigstellen (Suite schon halbwegs von jUnit abgekupfert) Doku, sinnhaftigkeit des abgekupferten refactorn.
- Log4j Verwendung Dokumentieren, möglichst einen Weg finden für logs ohne jboss-log anteile (eigener message-level-abschnitt?) und für automatisierte auswertung dieser logs (csv-format?)

EJB 2.0 Testsuite - Remote vs. Local calls.

- Dieseslbte Methdoe auf dem selben Rechner aufrufen - einmal im lokalen interface einmal im remoten (jboss sollte sich dabei sparen, sockets aufzumachen, vielleicht kann man das herausbekommen?)
- Nun Dieselbe Methode noch über eine Rechnergrenze hinweg aufrufen.

EJB 2.0 Testsuite - By Value mit BLOBs

- Ein grosser Datenklops (vielleicht noch Multimedia-Feld in JAPS_DB aufnehmen, wäre auch für die Demo von Field-Groups nützlich) wird über ein remote-interface abgefragt und manipuliert-> er muss kopiert werden.
- Nun über ein local interface -> die referenz müsste viel schneller sein.

EJB 2.0 Testsuite - Rollen und EJBS

Orientation in Objects

- 2 Beans auf 1 Tabelle mit je Home,Remote und Implementierung. (person -> Trainer,Teilnehmer) brachte das Probelm des zweifachen Relationsendes im selben Feld mit sich, was jboss nicht akzepierte. Ist dies ein jboss, oder ein prinzipielles problem? (jboss?)
- Alternative1 : BMP für die Realtionen
- Alternative 2: Dependant Value Class

Einschränkungen

Orientation in Objects

- Eine Entity-Bean soll ihrem Client
  - keine Collections des Persistence Managers weitergeben
  - keine getter/setter anbieten, die auf CMR-Felder gehen, die eine 1:1 oder n:1 Relation bezeichnen

EJB und JDO

Orientation in Objects

JDO

- Objektorientierung (auch Vererbung) fast ohne Einschränkungen
- Java-Syntax ähnliche Abfragesprache JDOQL
- Quellcode vollständig frei von Persistenzanteil
- Bestehende Schemata nur mit Application Identity (nicht standardisiert)

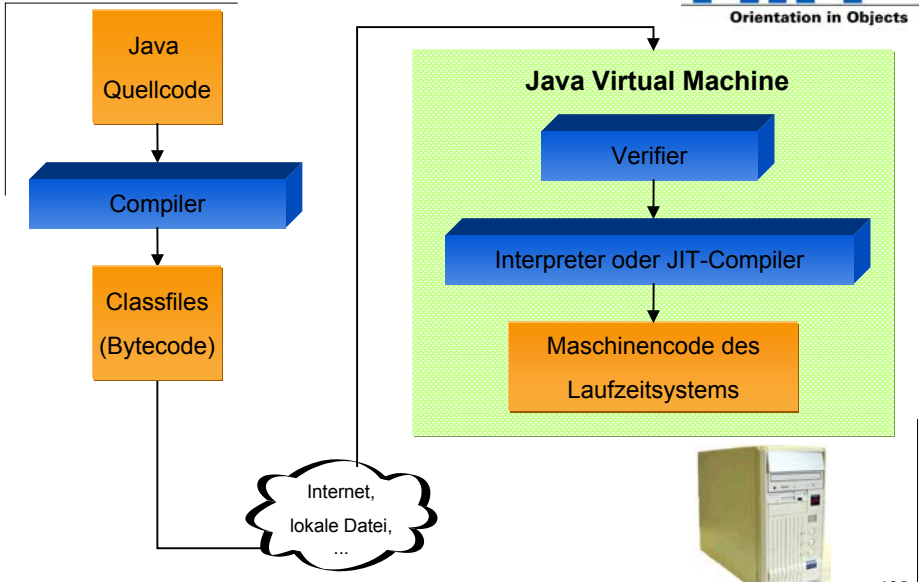
EJB

- Starke Ausrichtung am relationalen Datenmodell, jetzt Assoziationen
- EJBQL (ähnelt SQL u. OQL)
- Bei CMP auch frei
- Jede Entity-Bean definiert ihren Hauptschlüssel selbst. Simples „Je Bean eine Tabelle“

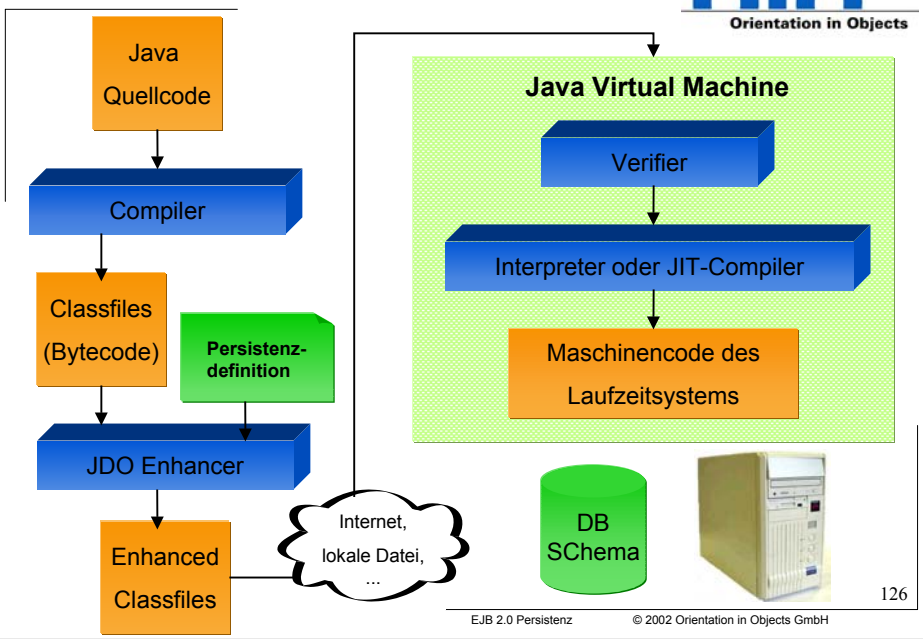
JDO Stand der Spezifikation

Orientation in Objects

Lebensweg einer Java-Anwendung



JDO Lebensweg einer Java-Anwendung



JDO Produkte

- TechTrader Kodo
- Signsoft IntelliBO

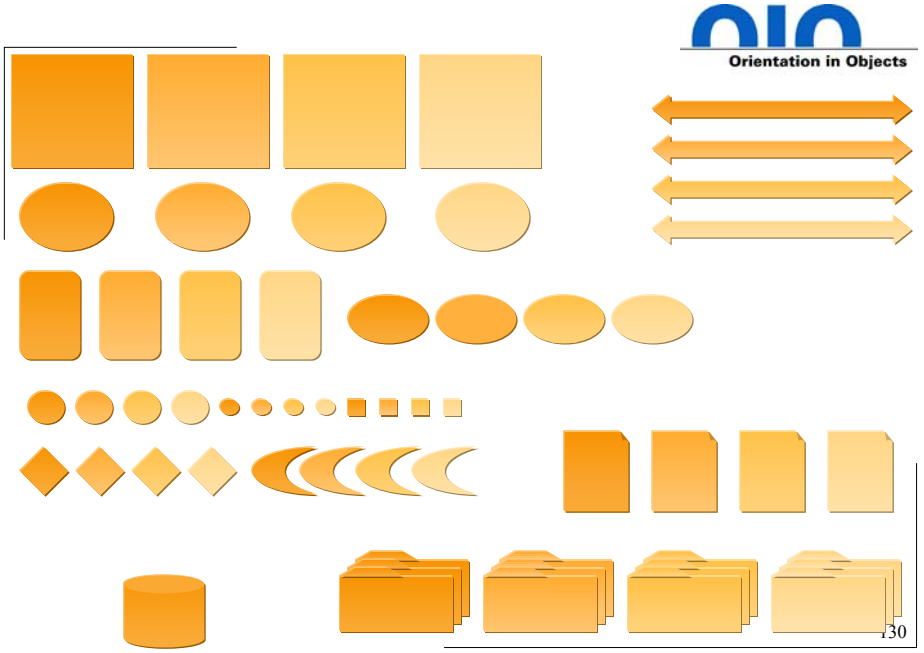
Vererbung und EJBs

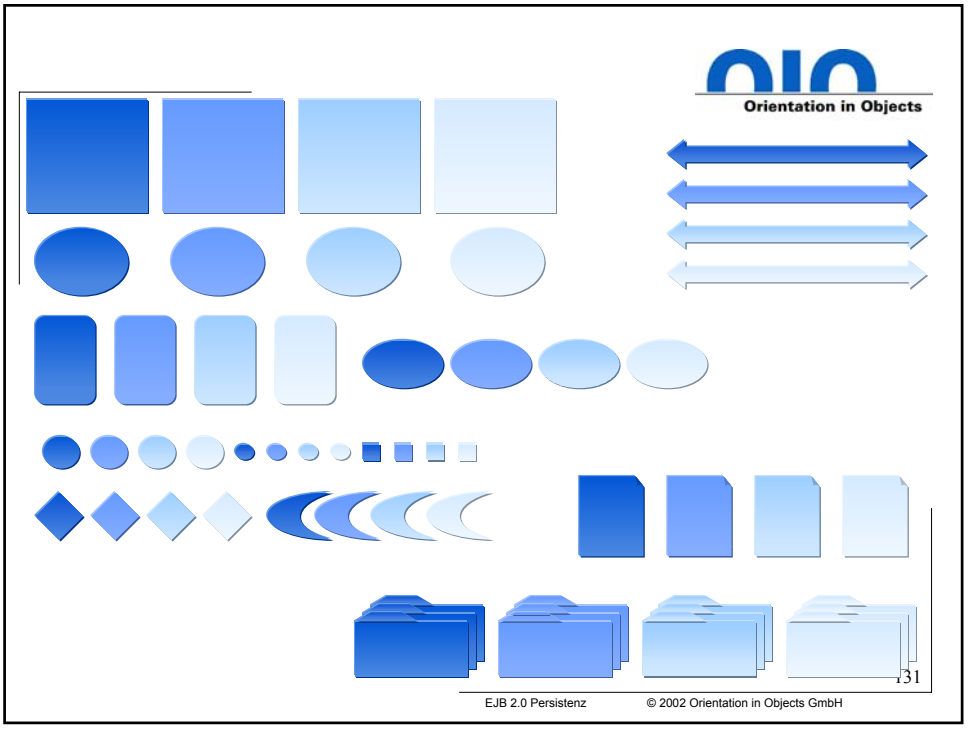
- Interface Vererbung
  - Kann für home und component interfaces verwendet werden.
- Klassenvererbung
  - Kann für die EJB Klasse benutzt werden.
- Komponenten-Vererbung
  - Ist nicht vorgesehen

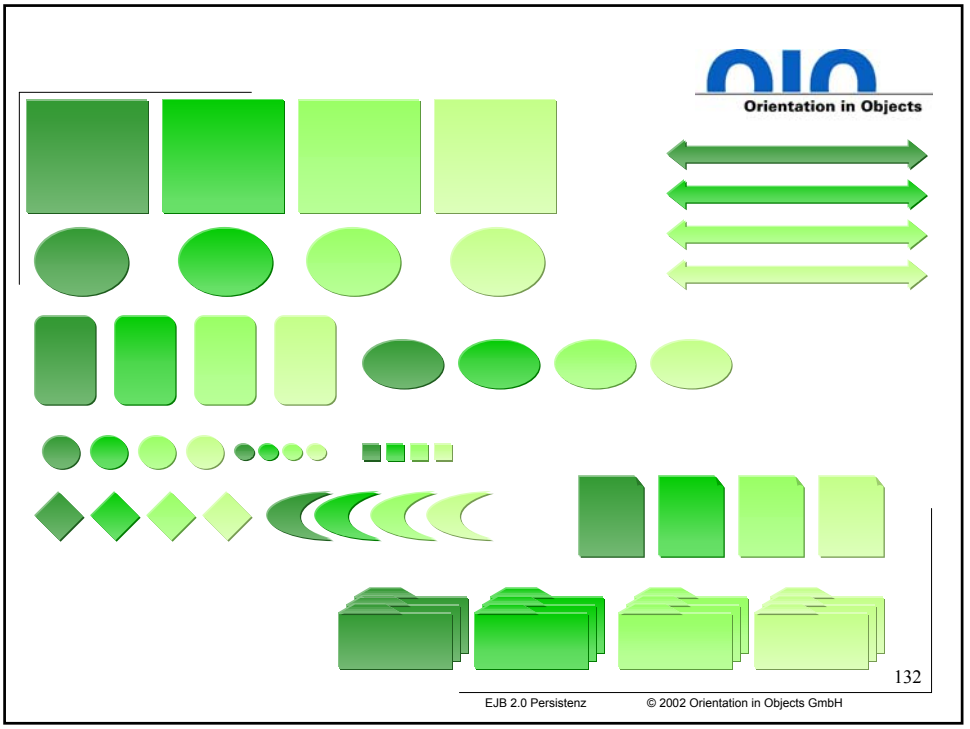
Überblick

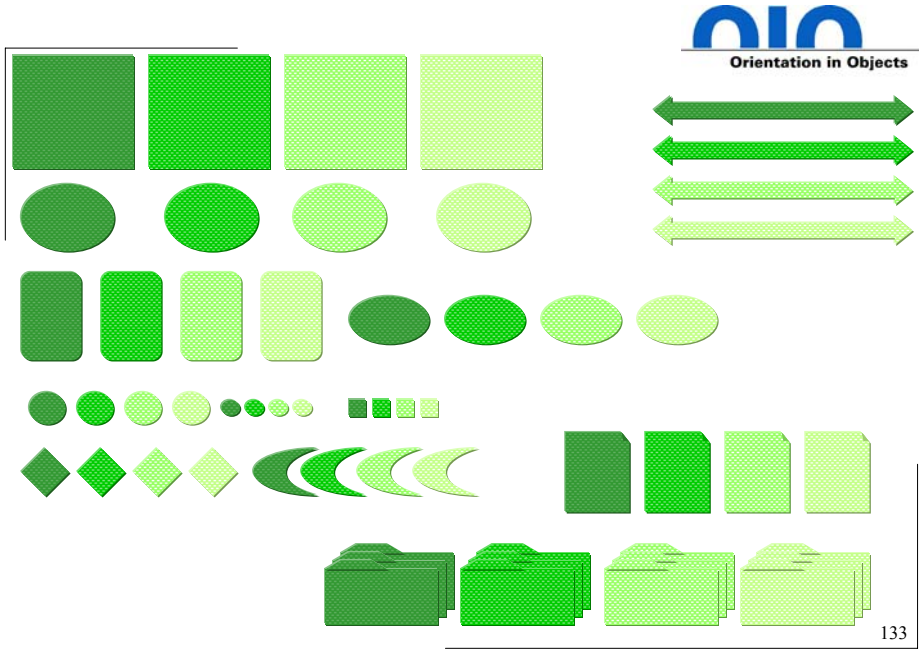
Orientation in Objects

- Einführung
- Komponentengröße
- Das CORBA Komponentenmodell (CCM)
- Integration von CORBA Komponenten mit EJB 1.1
- Enterprise JavaBeans 2.0
- Vergleich des Vorgehens bei der Entwicklung der Modelle
- Einschätzung des Reifegrades von CCM

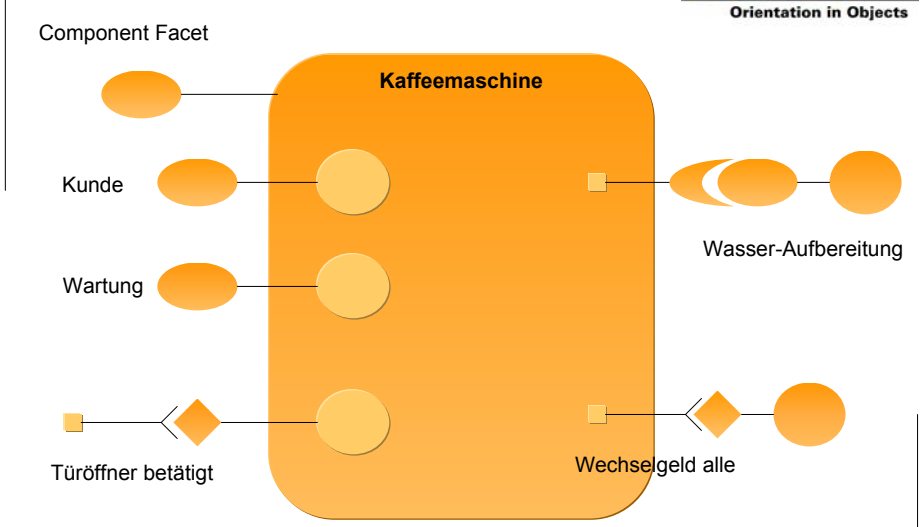



Orientation in Objects

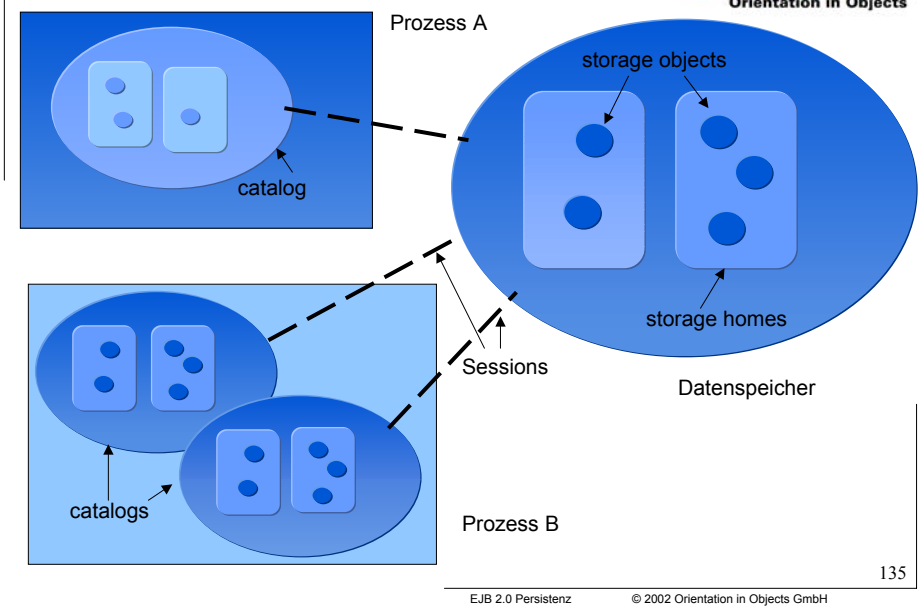

Orientation in Objects



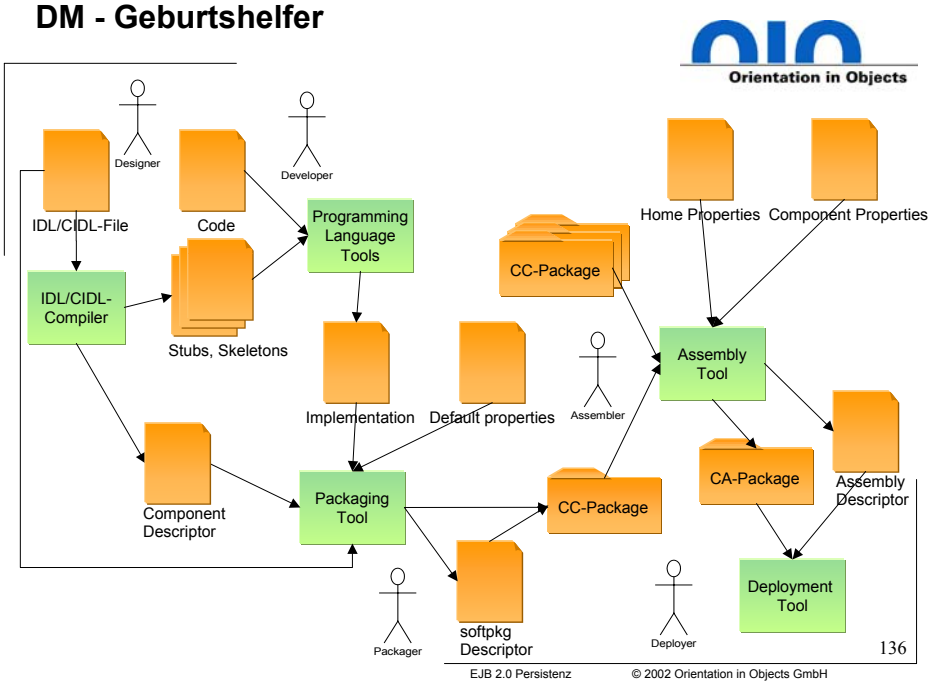
Kaffeemaschine - Events



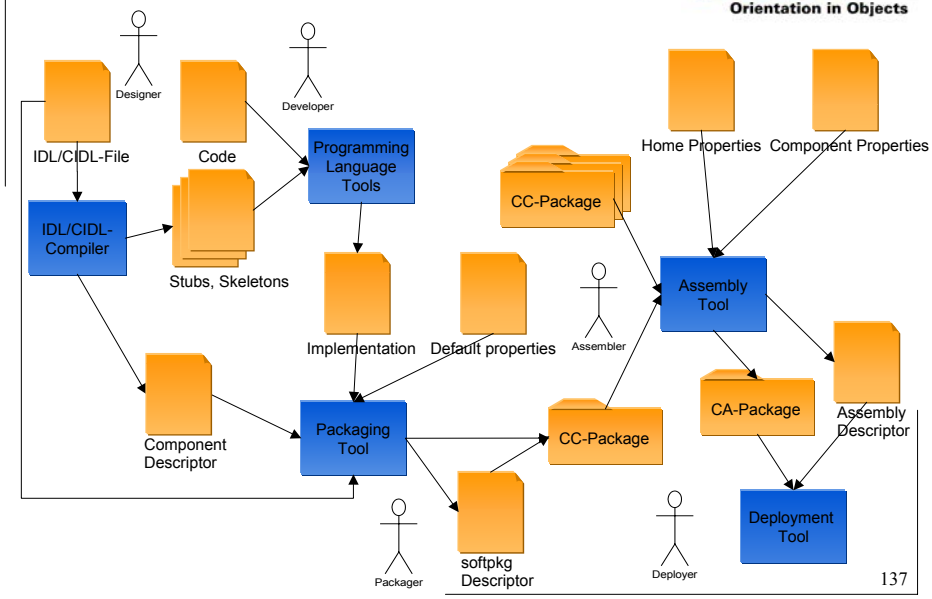
PSS



DM - Geburtshelfer



DM - Geburtshelfer



Ein CORBA EJB-Container