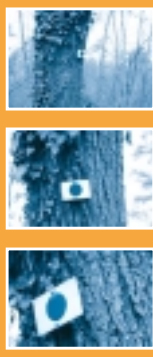


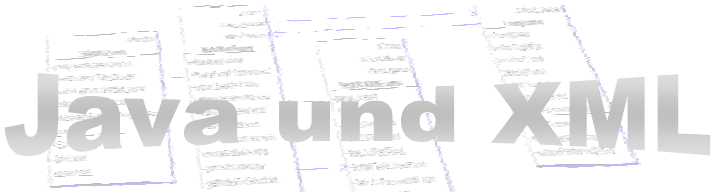
Persistenz mit Hibernate

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

„Orientierung“ in Objekten



) Akademie)

- **Schulungen**, **Coaching**, **Weiterbildungsberatung**, **Train & Solve**-Programme

) Beratung)

- **Methoden**, **Standards** und **Tools** für die Entwicklung von **offenen, unternehmensweiten Systemen**

) Projekte)

- **Schlüsselfertige Realisierung** von Software
- **Unterstützung laufender Projekte**
- **Pilot- und Migrationsprojekte**

Ihre Speaker

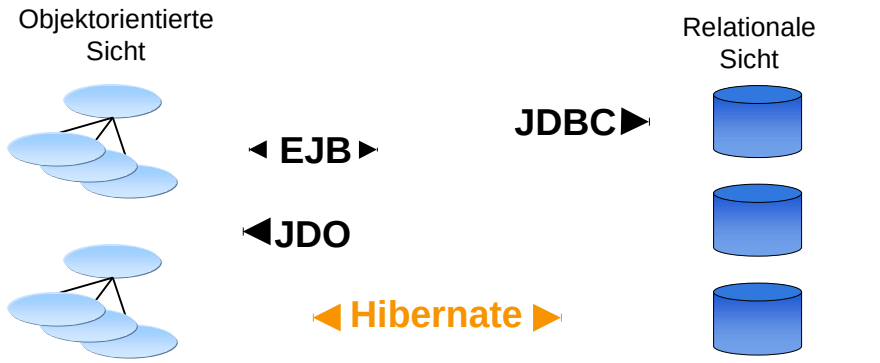
Serge Ndong
<ndong@oio.de>

Tobias Kieninger
<kieninger@oio.de>

Gliederung

- **Hibernate?**
- HelloHibernateWorld
- Leightweight J2EE
- Heute und Morgen

Technologien zur Persistenz mit RDBMS



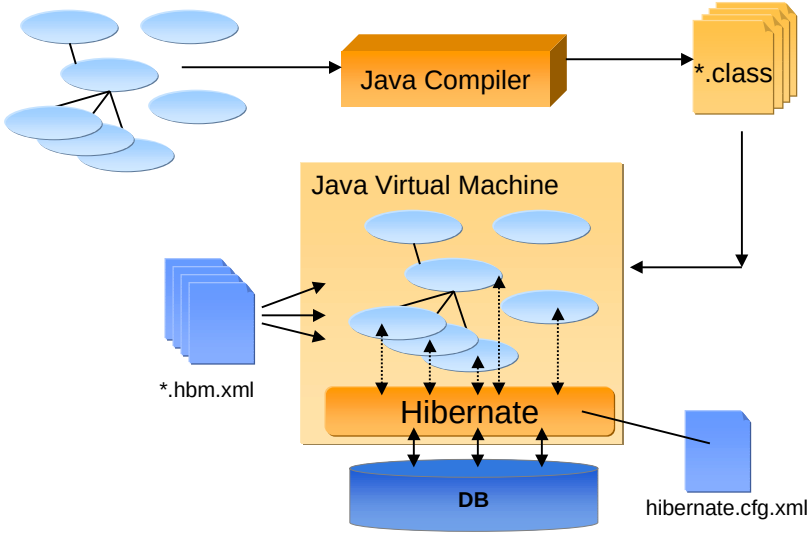
Was ist Hibernate?

- „Einfaches“ Persistenz-Framework
  - Arbeitet mit Reflection und CGLIB (Code Generation Library)
- Anwendung sowohl in managed als auch non-managed Umgebungen
- Unterstützt viele Datenbanken mit JDBC-Treibern
  - DB2, PostgreSQL, MySQL, Oracle, Sybase, MS SQL Server,
  - SAP DB, Informatix, HypersonicSQL, Ingres, Progress, MCKoiSQL,
  - Oracle, MSSQL, Interbase, Pointbase, Frontbase, Firebird

Lizenz und aktuelle Versionen

- **URL:** http://www.hibernate.org
- Versionen:
  - 3.0 (seit 31.03.2005)
  - 3.0.3 (Maintenance Relases)
- Open-Source
  - Lizenziert unter LGPL
  - Kostenlos für Entwicklung und Betrieb
- Wird inzwischen bei JBoss entwickelt

Wie funktioniert Hibernate?



Features (Auszug)

- Transparente Persistenz (~~POJOs~~) POPOs (Plain Old Persistent Object)
- Automatisches Dirty Checking
- Transitive Persistenz
- Lazy Fetching
- Mappingstrategien für Vererbung

Gliederung

- Hibernate?
- **HelloHibernateWorld**
- Leightweight J2EE
- Heute und Morgen

Java Klasse - POJO

```
public class User {

   private Long id;
   private String firstName;
   private Set addresses = new HashSet();
   ....
   public User() {}

   public User(String firstName, String lastName,...) {...}

   public Long getId() { return id;}

   private void setId(Long id) {
       this.id = id;
   }
   ... //Getter, Setter und Business Methoden
}
```


HelloHibernateWorld

```
// Konfiguration
Configuration cfg = new Configuration();
SessionFactory factory = cfg.buildSessionFactory();

// Verbindungsaufbau
Session session = factory.openSession();
Transaction tx = session.beginTransaction();

// Business Logik
User user=new User("Wade","Abdoulaye",“president@senegal.sn");
session.save(user);

tx.commit();
session.close();
```

Konfigurationsdatei hibernate.cfg.xml


```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE ...>
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.connection.driver_class">
       .hsqldb.jdbcDriver
    </property>
    <property name="hibernate.connection.url">
       :hsqldb:hsql://localhost/
    </property>
    <property name="hibernate.connection.username“>
      sa
    </property>
    <property name="hibernate.connection.password“/>
```


Konfigurationsdatei hibernate.cfg.xml


```
  <property name="dialect">
            .hibernate.dialect.HSQLDialect
    </property>
    <property name="show_sql">true</property>
    ...
    <mapping resource="de/oio/User.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

Mapping - User.hbm.xml

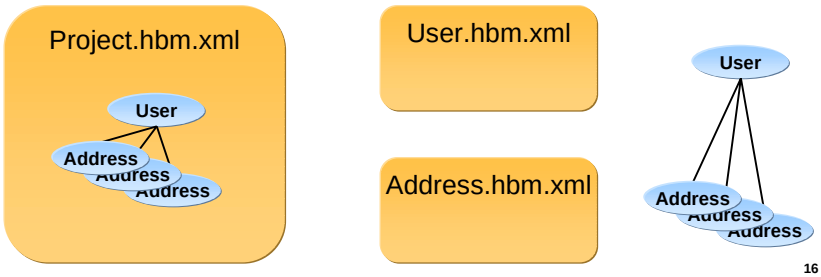
```
<?xml version="1.0"?>
<hibernate-mapping>
 <class name="de.oio.User" table="user">
   <id name="id" column="USER_ID" type="long">
     <generator class="increment"/>
   </id>
   <property name="firstName" column="VORNAME"/>
   <property name="lastName" column="NACHNAME"/>
   <property name="email" column="EMAIL" type="string"/>
  <set name="addresses">
     <key column="user_id"/>
     <one-to-many class="de.oio.Address"/>
   </set>
</class>
</hibernate-mapping>
```


Ablage der Mapping-Dateien

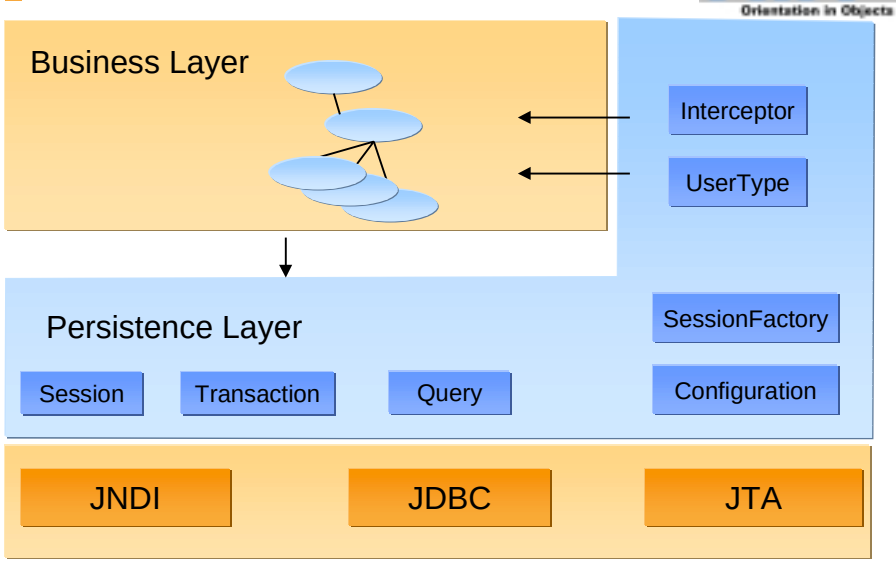
- Programmatisch hinzufügen

```
Configuration cfg = new Configuration()
cfg.addClass(de.oio.User.class);
```
oder
```
cfg.addRessource("de/oio/User.hbm.xml");
```

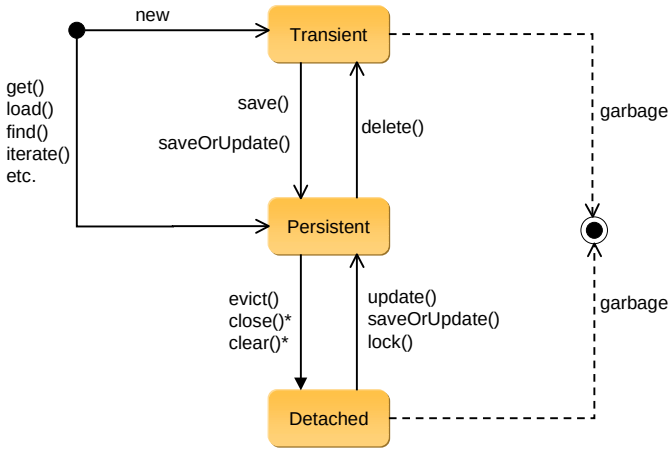
- In Konfiguration hinzufügen
  - `<mapping resource="de/oio/User.hbm.xml"/>`



Hibernate API in einem Schichtenmodell



Objektszustände



* gilt für alle Objekten in der Session

HQL - Hibernate Query Language

- SQL-Funktionen
  - Aufruf von SQL-Funktionen (sysdate, upper, ...)
  - Für manche Dialekte
- Aggregation
  - count(), min(), max(), sum(), avg()
- Gruppierungen
  - group by
  - having
- Report Queries
  - SQL ist am schnellsten
  - Bei Report Queries kein Caching, kein Overhead
  - Sehr schnell

HQL - Hibernate Query Language

- Hibernate Query Language (HQL)
  - Objektorientierter Dialekt von ANSI SQL
  - Klassen und Eigenschaften statt Tabellen und Spalten
  - Unterstützt Polimorphismus
  - Viele SQL Funktionen(Joins, Projektionen, Aggregation...)

```
Query query = session.createQuery(„from User u where
   u.firstName = :fname order by u.name asc“);

q.setString("fname", “abdoulaye");
List result = q.list();
```

HQL - Aggregationen

- Zusammenfassung von Daten und Fakten
- Bsp Minimum, Maximum etc...

```
Integer count =
(Integer) session.createQuery(
          "select count(*) from User").uniqueResult();


List results = session.createQuery(
       "select user.name, count(user), avg(user.logins)
        from User user where user.status > 1
        group by user.name").list();
```


HQL - Verschiedene Query Strategien

- Eager fetching im HQL-Query
- Asoziierte Objekte werden automatisch befüllt

```
User user = session.createQuery(
    “from User u left join fetch user.addresses
          where user.id = 5”);
//assoziation ist schon nachgeladen
user.getAdresses().iterator();
```

Queries by Criteria (QBC) API

- Queries by Criteria (QBC) API
  - Queries werden objektorientiert zusammengebaut
  - Enthält Query by Example

```
Criteria criteria = session.createCriteria(User.class);
criteria.add(Expression.like(„firstname“, „abdoulaye“));
List result = criteria.list();

User exampleUser = new User();
exampleUser.setFirstname(„abdoulaye“);
Criteria criteria = session.createCriteria(User.class);
criteria.add(Example.create(exampleUser));
List result = criteria.list();
```


Practice - Dynamische Queries


```
...
if (firstname != null) {
  queryString.append(„lower(u.firstname) like :fname “);
  conditionFound=true;
}
if (lastname != null) {
  if (conditionFound) queryString.append("and ");
  queryString.append("lower(u.lastname) like :lname ");
  conditionFound=true;
}
...
if (firstname != null)
  query.setString("fname", '%‘ + fn.toLowerCase() + '%');
...
```

Better Practice - Dynamische Queries mit Criteria API

```
Criteria crit = getSession().createCriteria(User.class);
if (firstname != null) {
  crit.add( Expression.ilike("firstname", firstname,
                                    MatchMode.ANYWHERE) );
}
if (lastname != null) {
  crit.add( Expression.ilike("lastname", lastname,
                                    MatchMode.ANYWHERE) );
}
```

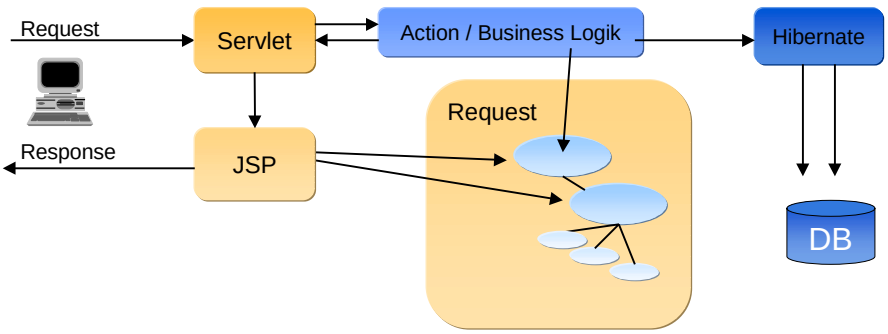

Best Practice - Dynamische Queries mit Query By Example

```
User u = new User;
u.setFirstname("abdoulaye");

Example exampleUser = Example.create(u)
                        .ignoreCase()
                        .enableLike(MatchMode.ANYWHERE);

list = getSession().createCriteria(User.class)
                   .add(exampleUser)
                   .list();
```

Gliederung

- Hibernate?
- HelloHibernateWorld
- **Leightweight J2EE**
- Heute und Morgen

Beispiel Architektur



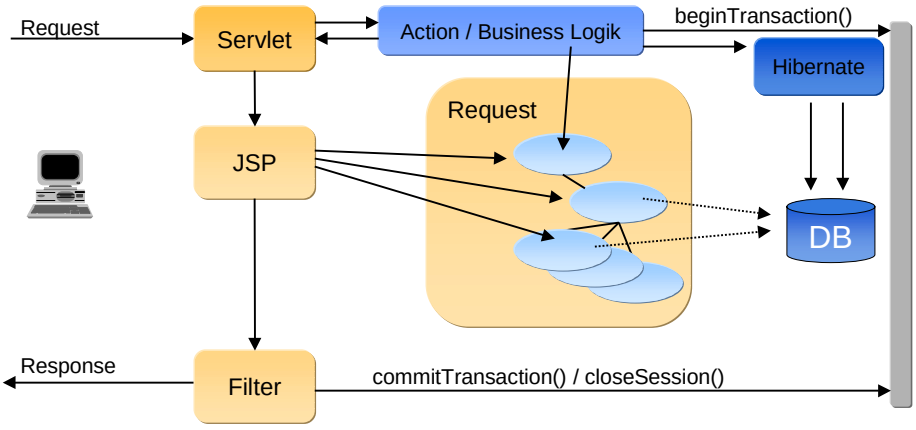
Zugriff auf SessionFactory im Webumfeld

- **Problem:** Nur eine Instanz der SessionFactory
- Möglichkeiten
  - Application Context
  - JNDI
  - Singleton
  - ThreadLocal
- Empfehlung
  - Implementierung in HibernateUtil.java je Architektur

ThreadLocal-Pattern

- ThreadLocal-Pattern
  - Eine Session wird mit **einem** Request assoziert
  - Gemeinsamer Persitenz Kontext
- ThreadLocal
  - ThreadLocal Variable bietet Kopie an jedem Thread der ihn benutzt
  - Thread sieht nur Variablen die mit ihm assoziiert sind
  - Kein Zugriff auf die anderen Variablen

ThreadLocal Beispiel Architektur



Implementierung ThreadLocal-Pattern

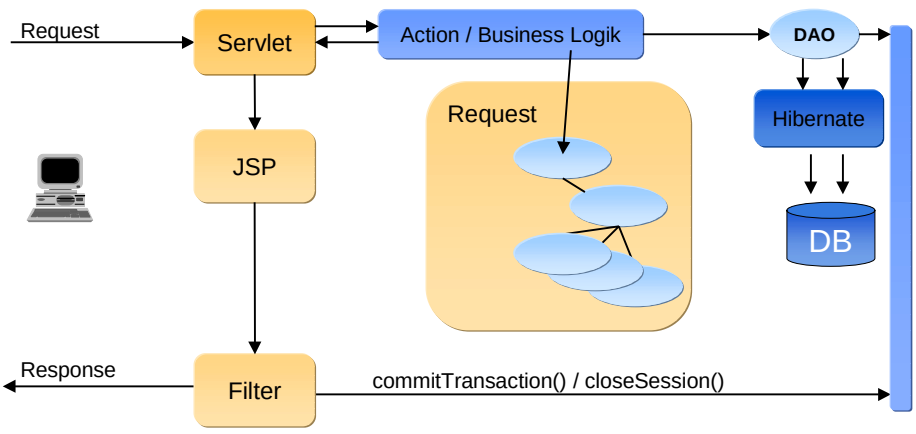
```
public class HibernateUtil {

   private static final ThreadLocal threadSession
                    = new ThreadLocal();

   public static Session getSession() {
        Session s = (Session) threadSession.get();
        if (s == null) {
               s = sessionFactory.openSession();
               threadSession.set(s);
        }
        return s;
   }

   public static Session beginTransaction() { ... }
}
```

Sessions schliessen

- Schliessen aller Sessions in ServletFilter

```
public void doFilter(...) throws ... {
   try {
        chain.doFilter(request, response);
        HibernateUtil.commitTransaction();
   }
    catch(Exception e) { ... }
    finally {
        HibernateUtil.closeSession();
   }
}
```


Verwendung eines DAOs (Data Access Object)

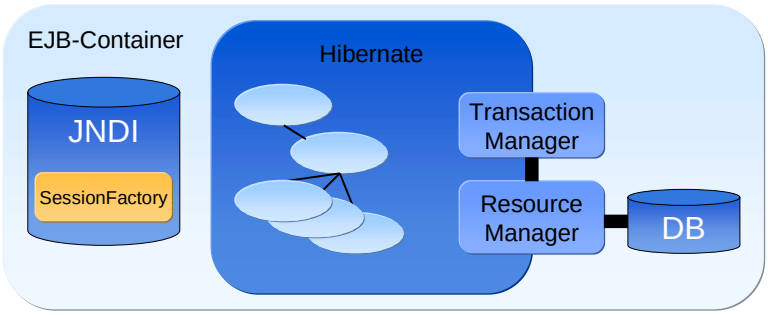


Implementierung Data Access Object

```
public class UserDAO {
   public UserDAO() {
       HibernateUtil.beginTransaction();
   }

   public User getUserById(Long userId) {
     Session session = HibernateUtil.getSession();
     User user = null;
     ...
     user = (User) session.load(User.class, userId);
     ...
     return user;
   }
}
```


Managed Environment

- Features von Container nutzbar durch JCA (J2EE Connector Architecture)
  - Externes Transactionmanagement (JTA)
  - Externes Connectionmanagement

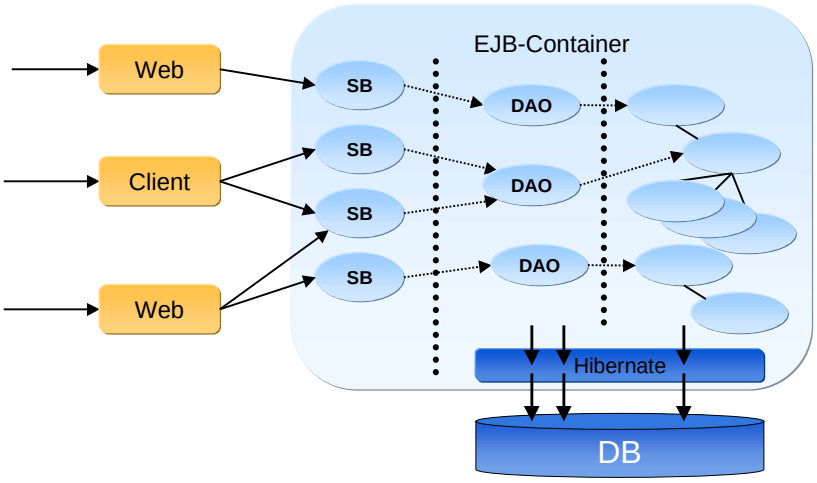


Hibernate im Application Server - Unterschiede

- SessionFactory wird in JDNI abgelegt
  - Wird automatisch an JNDI gebunden
  - Konfiguration mittels hibernate.cfg.xml

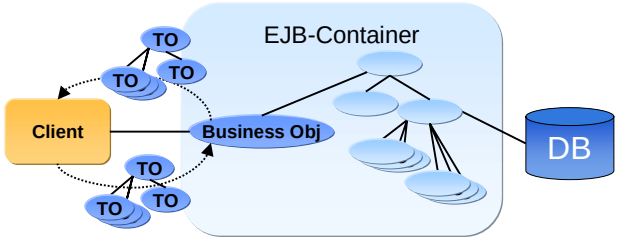
```
static {
  new Configuration().configure()
                        .buildSessionFactory();
  ...
}

public static SessionFactory getSessionFactory() {
  Context ctx = new InitialContext();
  String jndiName = "java:hibernate/HibernateFactory";
  return (SessionFactory) ctx.lookup(jndiName);
}
```


Session Facade



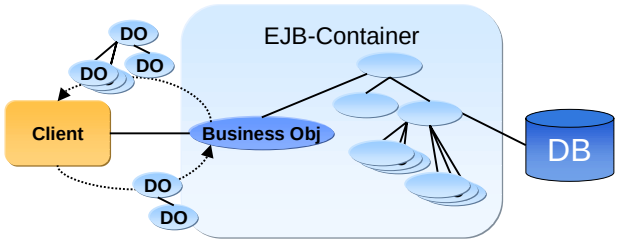
Braucht Hibernate Data Transfer Objects?

- Value Objects (VO) bzw. Data Transfer Objects (DTO)
- Datenkontainer ohne Logik
- Eventuell Value Object Assembler der Value Objects behandelt
- Entkopplung vom Objektmodell



Anwendung von Detached Objects

- Objekte, ganze Teilbäume können aus Session entfernt werden
- Verlieren Identität nicht
- Können wieder persistent gemacht werden



Detaching von Objekten - Beispiel

```
...
  Session session1 = sf.openSession();
  Transaction tx1 = session.beginTransaction();
  User user = (User) session.get(User.class, userId);
  tx.commit();
  session.close();
...
   user.setPassword(„secret“);
   Session session2 = sf.openSession();
   Transaction tx2 = session2.beginTransaction();
   session2.update(user);  // session2.saveOrUpdate(user);
   tx2.commit();
...
```

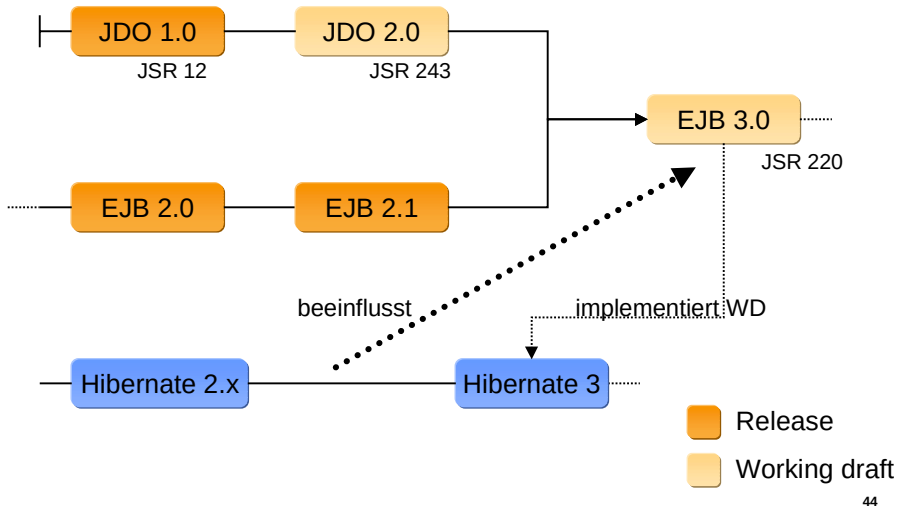

DTO - Data Transfer Objects

- Vorteile
  - Flacht die Sicht auf stark hierarchisches Domain Model ab
  - Bereitet Daten für View-Komponenten auf
- Nachteile
  - Es erfolgt Duplizierung der Daten
  - Erweiterung des Modells muss in DTOs nachgezogen werden
  - Versionierung, Überspielen der Daten schwierig

Gliederung

- Hibernate?
- HelloHibernateWorld
- Leightweight J2EE
- **Heute und Morgen**

Standards und zukünftige Persistenz-Standards



Annotations

- Metainformationen können in Java-Klasse abgelegt werden
- Verfügbar ab Java 1.5
- Stehen in der Bytecode(.class) und
- Auch zur Laufzeit zur Verfügung

```
...
@Entity(access = AccessType.FIELD)
public class User implements Serializable {
...
```


Annotation Entity Bean


```
@Entity(access = AccessType.FIELD)
public class User implements Serializable {
   @Id;
   Long id;

   String firstName;
   String lastName;
   Date birthday;
   @Transient Integer age;

   .....
    // Getter/setter und business Methoden
 }
```

Gossip

- Versant sponsort VOA (Versant Open Access) als Eclipse Subproject
- Oracle will eigenes Eclipse EJB3 Subproject unterstützen
- Viele Unterstellungen im Persistenzkrieg

Fazit / Ausblick

- EJB 3 vielversprechend
- Open Source Markt wird angereichert
- Hibernate starke Stabilität und hohe Popularität



Vielen Dank für Ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de



Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

„OIO-JAX Scrabble“ Gewinnspiel

- Spielen Sie mit uns eine Runde OIO-JAX Scrabble und gewinnen Sie ein **OIO Seminar** aus dem Bereich

- „Scrabble-Felder“ bekommen Sie an unserem Konferenzstand.
- Abgabe bis Donnerstag, 12.05.05 - 11.45 Uhr
- Gewinnbekanntgabe: 12.05.05 - 14.10 Uhr

- Wir freuen uns über Ihren Besuch.
  Viel Spass beim Scrabbeln!