

# AVID Übung 3

## Enterprise JavaBeans

Andreas I. Schmied (schmied@inf...)

Abteilung Verteilte Systeme  
Universität Ulm

SS2005

# Und nun...

## 1 Motivation

# Motivation – Wiederholung: Beans der JSF-Übung

- UserBean

```
1 class UserBean { private int id;  
2                 private String name, pass; }
```

- CalendarBean

```
1 class CalendarBean { private List entries = new ArrayList(); }
```

- EntryBean

```
1 class EntryBean { int owner;  
2                 String text; }
```

- hier umbenannt, z.B. EntryValue

# Motivation – Ziel

- Geschäftsdaten in Entity Beans
- Geschäftsmethoden in Session Beans
- Warum?
  - Abstraktion der Persistenz
  - Skalierbarkeit, Transaktionalität, ...
- Zugriff durch Servlets
  - keine direkte Interaktion mit Entity Beans
  - Value-Objects: reguläre JavaBeans zum Datentransfer
  - Durchsatz höher als bei Remote-Getter pro Attribut
- effiziente, elegante EJB-Programmierung möglich?
- Codebeispiele wie immer gekürzt

# Und nun...

## 2 Entwicklung

- Session Bean
- Bean-Referenzen
- Entity Bean
- Entity Bean (BMP)
- Entity Bean (CMP)
- Deskriptoren
- Bean-Beziehungen
- Client

# Entwicklung – Schritte

- Ermitteln von Session/Entity Beans
  - Stateless/Stateful Session Beans
  - Remote/Local Interfaces
  - Entities mit CMP/BMP, CMT/BMT
- Architekturkonzept
- Programmierung
- Deskriptoren erstellen
- EAR, WAR, Client-JAR
- Aufsetzen der Datenbank
- Deployment und Start

# Entwicklung – Architekturkonzept

- Remote Session-Beans
- Local Entity-Beans
  - Abstrakte Persistenz mit CMP
  - ebenso Pooling, Passivation
- Datentransfer mittels Value-Objects
- getrennte JVMs für EJB- und Servlet-Container
  - zumindest konzeptionell
  - Remoting zwischen Servlets und Session Beans
  - Lokale Referenzen zwischen EJBs
  - Entity Beans nur lokal für Session Beans erreichbar

# Session Bean – Überblick

- Kalender-Management, stateful
- Geschäftsmethoden
  - umfangreiche Aktionen
  - an eigenständigen Geschäftsobjekten
- Bestandteile
  - Interface `javax.ejb.SessionBean`
    - `ejbActivate/Passivate/Remove`, `setSessionContext`
  - public Default-Constructor
  - diverse `ejbCreate(...)`
  - Bean-Implementierung ohne `RemoteException`
  - Remote-Home/Bean-Interface
    - verwandte Methodensignaturen!

# Session Bean – Klassen/Schnittstellen

```
1 class CalendarManagerBean implements SessionBean {
2
3     public void ejbCreate(UserValue u) throws CreateException { ... }
4
5     public EntryValue[] getEntries() { ... }
6     public void addEntry(EntryValue e) { ...}
7
8     public void ejbRemove() {...}
9     public void ejbActivate() {...}
10    public void ejbPassivate() {...}
11    public void setSessionContext(SessionContext sc) {...}
12 }
13
14 interface CalendarManagerHome extends EJBHome {
15     CalendarManager create(UserValue u) throws RemoteException, CreateException;
16 }
17
18 interface CalendarManager extends EJBObject {
19     public EntryValue[] getEntries() throws RemoteException { ... }
20     public void addEntry(EntryValue e) throws RemoteException { ...}
21 }
```

# Bean-Referenzen – Überblick

- Austauschbarkeit/Erweiterung auf Deployment-Ebene
- Konzept
  - Clients befragen Verzeichnisdienst, hier: JNDI
  - Client kann Servlet oder anderes EJB sein
  - JNDI liefert Referenz auf Home-Interface
  - jeder Client hat eigene Konfigurationsumgebung
  - einheitlich unter `java:comp/env` im JNDI-Namensraum
  - Bean-Referenzen üblich unter `/env/ejb`
- zwei Beispiele
  - Servlet greift auf Remote-EJB zu
  - EJB verwendet Local-EJB im selben Container

# Bean-Referenzen – Konfiguration

in web.xml

```
1 <ejb-ref>
2   <ejb-ref-name>ejb/CalMgr</ejb-ref-name>
3   <ejb-ref-type>Session</ejb-ref-type>
4   <home>avid.CalendarManagerHome</home>
5   <remote>avid.CalendarManager</remote>
6 </ejb-ref>
```

in ejb-jar.xml

```
1 <ejb-local-ref>
2   <ejb-ref-name>ejb/Entry</ejb-ref-name>
3   <ejb-ref-type>Entity</ejb-ref-type>
4   <local-home>avid.EntryHome</local-home>
5   <local>avid.Entry</local>
6 </ejb-local-ref>
```

# Bean-Referenzen – Zugriff

```
1 javax.naming.Context context = new javax.naming.InitialContext();
2
3 Object ref = context.lookup("java:comp/env/ejb/CalMgr");
4
5 home = (CalendarManagerHome) javax.rmi.PortableRemoteObject
6         .narrow(ref, CalendarManagerHome.class);
7
8 CalendarManager mgr = home.create();
```

# Entwicklung – Entity Bean

- Geschäftsobjekte
- Geschäftsmethoden
  - objektbezogene Methoden
  - an abhängigen Geschäftsobjekten
- LocalHome-, LocalBean-Interface
  - Entities nur intern von Session Beans angesprochen
  - dann aber Session Bean Wrapper auf Geschäftsmethoden
- Varianten: CMP und BMP
- Bestandteile analog Session Bean

# Entity Bean (BMP) – Klasse

```
1 class EntryBean implements EntityBean {
2
3     public void ejbLoad() {...}
4     public void ejbStore() {...}
5     public void ejbRemove() {...}
6
7     public Integer ejbCreate(EntryValue e) throws CreateException {...}
8     public void ejbPostCreate(EntryValue e) {...}
9
10    public Integer ejbFindByPrimaryKey(Integer id) throws FinderException {...}
11    public Collection ejbFindInYear(int year) throws FinderException {...}
12
13    public String getText() {...}
14    public void setText(String t) {...}
15    public UserBean getOwner() {...}
16    public void setOwner(UserBean u) {...}
17 }
```

- Bean/Home-Interface siehe CMP

# Entity Bean (BMP) – JDBC in ejbCreate

```
1 InitialContext context = new InitialContext();
2 DataSource ds = context.lookup("java:comp/env/jdbc/AvidDB");
3 Connection con = ds.getConnection();
4
5 con.setAutoCommit(false);
6 String sql = "INSERT INTO entry VALUES (?, ?, ?, ?)";
7 PreparedStatement stmt = con.prepareStatement(sql);
8
9 stmt.setString(1, entryID);
10 stmt.setString(2, text);
11 ...
12 stmt.executeUpdate();
13 stmt.close();
14 con.commit();
15
16 ... // release/reuse connection
```

# Entity Bean (CMP) – Klasse (1)

```
1  abstract class EntryBean implements EntityBean {
2
3  public abstract Integer getId();
4  public abstract void setId(Integer id);
5  public abstract String getText();
6  public abstract void setText(String t);
7  public abstract UserBean getOwner();
8  public abstract void setOwner(UserBean u);
9
10 public EntryValue getValue() {...}
11 public void setValue(EntryValue e) {...}
12
13 public void setSessionContext(SessionContext sc) {...}
14 public void ejbLoad() {return;}
15 public void ejbStore() {return;}
16 public void ejbRemove() {return;}
17
18 public abstract Integer ejbFindByPrimaryKey(Integer id) throws FinderException;
19 public abstract Collection ejbFindInYear(int year) throws FinderException;
20 ...
```

# Entity Bean (CMP) – Klasse (2)

```
1  ...
2  public Integer ejbCreate(EntryValue e) throws CreateException {
3
4      setText(e.getText());
5      return null;
6  }
7
8  public void ejbPostCreate(EntryValue e) { // establish relations!
9
10     context.Object ref = context.lookup("java:comp/env/ejb/User");
11     UserHome uhome = (UserHome) PortableRemoteObject.narrow(ref, UserHome.class);
12     User u = uhome.findByPrimaryKey(e.getOwner());
13     setOwner(u);
14 }
15
16 public abstract int ejbSelectCountAlarms(int id) throws FinderException;
17
18 public int countAlarms() { // wrapper!
19
20     //Entry e = (Entry)context.getEJBLocalObject();
21     return ejbSelectCountEntries(/*e.**/getId());
22 }
23 } // class
```

# Entity Bean (CMP) – Local Interfaces

- ohne RemoteExceptions!

```
1 interface Entry extends EJBLocalObject {
2     public Integer getId();
3     public void setId(Integer id);
4     public String getText();
5     public void setText(String t);
6     public UserBean getOwner();
7     public void setOwner(UserBean u);
8
9     public EntryValue getValue();
10    public void setValue(EntryValue e);
11    public int countAlarms();
12 }
13
14 interface EntryHome extends EJBLocalHome {
15     public Entry create(EntryValue e) throws CreateException;
16     public Entry findByPrimaryKey(Integer id) throws FinderException;
17     public Collection findInYear(int year) throws FinderException;
18 }
```

# Deskriptoren – ejb-jar.xml (1)

```
1 <?xml version="1.0"?>
2 <!DOCTYPE ejb-jar ...>
3
4 <ejb-jar>
5   <enterprise-beans>
6
7     <session>
8       <ejb-name>CalendarManager</ejb-name>
9       <home>avid.CalendarManagerHome</home>
10      <remote>avid.CalendarManager</remote>
11      <ejb-class>avid.CalendarManagerBean</ejb-class>
12      <session-type>Stateful</session-type>
13      <transaction-type>Container</transaction-type>
14    </session>
15
16    <entity> ... s.u. ... </entity>
17    ...
18  </enterprise-beans>
19 </ejb-jar>
```

# Deskriptoren – ejb-jar.xml (2)

```
1 <entity>
2   <ejb-name>Entry</ejb-name>
3   <local-home>avid.EntryHome</local-home>
4   <local>avid.Entry</local>
5   <ejb-class>avid.EntryBean</ejb-class>
6
7   <persistence-type>Container</persistence-type>
8   <prim-key-class>java.lang.Integer</prim-key-class>
9   <cmp-version>2.x</cmp-version>
10  <abstract-schema-name>Entry</abstract-schema-name>
11
12  <cmp-field>
13    <field-name>id</field-name>
14  </cmp-field>
15  <cmp-field>
16    <field-name>text</field-name>
17  </cmp-field>
18  <primkey-field>id</primkey-field>
19
20  <query>... s.u. ...</query>
21 </entity>
```

# Deskriptoren – ejb-jar.xml (3)

```
1 <query>
2   <query-method>
3     <method-name>ejbSelectCountAlarms</method-name>
4     <method-params>
5       <method-param>int</method-param>
6     </method-params>
7   </query-method>
8   <ejb-ql>
9     <![CDATA[
10    SELECT count(e.alarms)
11    FROM entry e
12    WHERE e.id = ?1
13    ]]>
14   </ejb-ql>
15 </query>
```

# Deskriptoren

- weitere Container-abhängige Deskriptoren
- z.B. jboss.xml, jboss-web.xml
- Anpassung an lokale Datenbank, Transaktionsmanagement
  - Überschreiben/Erweitern der Standard-Deskriptoren

# Bean-Beziehungen

- **Container-Managed Relationships**
  - ab CMP 2.0
  - inkl. referenzieller Integrität
- vorher manuelle Implementierung
- Beispiel: Entry bedient mehrere Alarmer

# Bean-Beziehungen – ejb-jar.xml – relationships (1)

```
1 <ejb-jar>
2   <relationships>
3     <ejb-relation>
4       <ejb-relation-name>Entry-Alarms</ejb-relation-name>
5       <ejb-relationship-role>
6         <ejb-relationship-role-name>entry-has-alarms</ejb-relationship-role-name>
7         <multiplicity>One</multiplicity>
8         <relationship-role-source>
9           <ejb-name>EntryBean</ejb-name>
10          </relationship-role-source>
11         <cmr-field>
12           <cmr-field-name>alarms</cmr-field-name>
13           <cmr-field-type>java.util.Set</cmr-field-type>
14         </cmr-field>
15       </ejb-relationship-role>
16
17       <ejb-relationship-role>... s.u. ...</ejb-relationship-role>
18     </ejb-relation>
19   </relationships>
20 </ejb-jar>
```

# Bean-Beziehungen – ejb-jar.xml – relationships (2)

```
1 <ejb-jar>
2   <relationships>
3     <ejb-relation>
4       <ejb-relation-name>Entry-Alarms</ejb-relation-name>
5       <ejb-relationship-role>... s.o. ...</ejb-relationship-role>
6
7     <ejb-relationship-role>
8       <ejb-relationship-role-name>alarm-of-entry</ejb-relationship-role-name>
9       <multiplicity>Many</multiplicity>
10      <cascade-delete/>
11      <relationship-role-source>
12        <ejb-name>AlarmBean</ejb-name>
13      </relationship-role-source>
14      <cmr-field>
15        <cmr-field-name>entry</cmr-field-name>
16      </cmr-field>
17    </ejb-relationship-role>
18  </ejb-relation>
19 </relationships>
20 </ejb-jar>
```

# Entwicklung – Client

- hier Servlets
- vollständiger Java-Client auch möglich
- Schritte analog zu Referenzen zwischen Beans
  - Deklaration der Bean-Referenzen in web.xml

```
1 <ejb-ref>  
2   <ejb-ref-name>ejb/CalMgr</ejb-ref-name>  
3   <ejb-ref-type>Session</ejb-ref-type>  
4   <home>avid.CalendarManagerHome</home>  
5   <remote>avid.CalendarManager</remote>  
6 </ejb-ref>
```

- Home-Interface ermitteln
  - JNDI-Eintrag java:comp/env/ejb/CalMgr
- Bean-Instanz erzeugen/finden
- Methodenaufrufe

# Und nun...

## 3 Paketierung

# Paketierung – Vier verschränkte Archive

- **AvidEJB.jar**
  - META-INF/ejb-jar.xml, .../jboss.xml
  - Class-Dateien (EJBs, Homes, Interfaces, Hilfsklassen)
- **AvidEJB-client.jar**
  - nur Class-Dateien für Client (Homes, Interfaces, Hilfsklassen)
  - keine Deskriptoren und EJBs
- **AvidWeb.war**
  - Servlet-Code, Web-Dokumente
  - WEB-INF/web.xml, ... jboss-web.xml
  - AvidEJB-client.jar
- **AvidApp.ear**
  - JAR, „Enterprise Application aRchive“
  - AvidEJB.jar
  - AvidWeb.jar
  - META-INF/application.xml
- am Besten per ANT-Task generieren (s.u.)

# Paketierung – application.xml

```
1 ...
2 <application>
3   <display-name>AVID Calendar</display-name>
4   <module>
5     <ejb>AvidEJB.jar</ejb>
6   </module>
7   <module>
8     <web>
9       <web-uri>AvidWeb.war</web-uri>
10      <context-root>/cal</context-root>
11    </web>
12  </module>
13 </application>
```

# Und nun...

## 4 JBoss

# JBoss – Installation

- Download von [jboss.org](http://jboss.org)
- Tarball auspacken – direkt benutzbar
- Unterverzeichnisse, u.a.:
  - bin** – Start-/Stop-Skripte
  - lib** – (gemeinsam genutzte) Server-JARs
  - server** – Server-Konfigurationen (all, minimal, default)
  - server/.../deploy** – Hot-Deployment (vgl. Tomcat)
- Tomcat als Servlet/JSP-Container integriert
- JBoss-IDE für Eclipse installieren
- Konfiguration starten: `bin/run.sh [-c <config>]`
- Hot Deployment: EAR-Datei kopieren

# JBoss – Management

- Default-Adresse <http://localhost:8080>
- JBoss-Architektur
  - Minimalkern + „Plug-Ins“
  - Managed Beans (MBeans, siehe „JMX“) erbringen Funktionalität
- Datenbank wird automatisch erzeugt
  - aus CMP-Deployment
  - manuelle Anpassung
  - Deployment-Anpassung in `jbosscmp-jdbc.xml`
- <http://localhost:8080/jmx-console>
  - Information, Konfiguration
  - Aufruf von MBeans
- <http://localhost:8080/web-console>

## JBoss – JBoss-Konsole

JBoss Management Console

System

- Unified ClassLoaders
- JMX MBeans
- JMImplementation
  - jboss
    - jboss:type=Service,name
    - jboss:type=Service,name
    - jboss:service=invoker,typ
    - jboss:service=ClientUser
    - jboss:service=JNDIView
    - jboss:service=Mail
    - jboss:service=Naming
    - jboss:service=Transaction
    - jboss:service=UUIDKeyG
    - jboss:service=WebService
    - jboss:service=XidFactory
    - jboss:service=invoker,typ
    - jboss:service=invoker,typ
    - jboss:service=invoker,typ
    - jboss:service=invoker,typ
    - jboss:service=invoker,typ
    - jboss:service=invoker,typ
    - jboss:service=proxyFact
    - jboss:service=proxyFact
  - jboss.admin
  - jboss.cache
  - jboss.deployment
  - jboss.ejb
  - jboss.j2ee

Operation Name	Parameters
<b>Return Type</b> <b>Description</b>	
<b>list</b> java.lang.String Output JNDI info as text	<b>verbose</b> boolean If true, list the class of each object in addition to its name <input checked="" type="radio"/> True <input type="radio"/> False <input type="button" value="Invoke"/>
<b>listXML</b> java.lang.String Output JNDI info in XML format	<input type="button" value="Invoke"/>
<b>create</b> void Standard MBean lifecycle method	<input type="button" value="Invoke"/>
<b>start</b> void The start lifecycle operation	<input type="button" value="Invoke"/>
<b>stop</b> void The stop lifecycle operation	<input type="button" value="Invoke"/>

# Und nun. . .

## 5 XDoclet

- Entwicklungsunterstützung
- „One-Source“ Paradigma
- Java-Quellcode angereichert mit Meta-Information
- Javadoc-Tags innerhalb `/** ... */`
- manuell verfasst werden
  - Bean-Implementierungen, Servlets
  - Application-Descriptor
- generiert werden
  - Bean- und Home-Interface
  - Deskriptoren (EJB, Web, JBoss)
  - ggf. Datenbank-Skripte

# XDoclet – Servlet-Beispiel

```
1  /**
2   * @web.servlet name = "CalendarServlet"
3   * display-name = "Calendar Servlet"
4   *
5   * @web.servlet-mapping url-pattern = "/cal"
6   *
7   * @web.ejb-ref name = "ejb/CalendarManager"
8   * type = "Session"
9   * home = "avid.CalendarManagerHome"
10  * remote = "avid.CalendarManager"
11  */
12  public class CalendarServlet extends HttpServlet {
13      ...
14  }
```

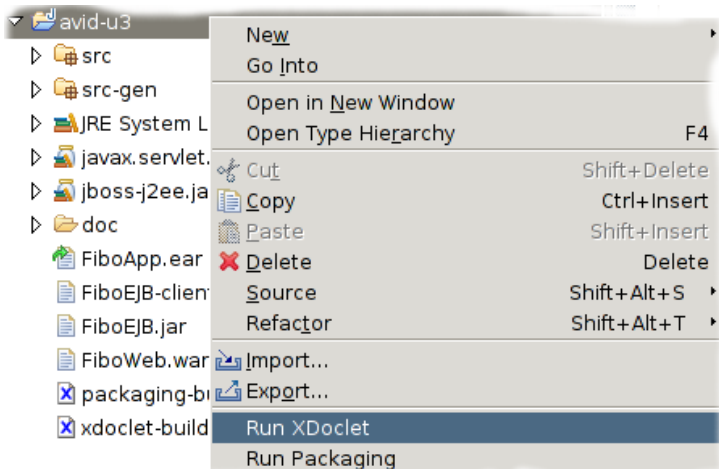
# XDoclet – EJB-Beispiel

```
1  /**
2   * @ejb.bean name="CalendarManager"
3   * jndi-name="ejb/CalendarManager"
4   * type="Stateful"
5   * view-type="remote"
6   */
7  public class CalendarManagerBean implements SessionBean {
8
9   /** @ejb.create-method */
10  public void ejbCreate() throws CreateException { ... }
11
12  /** @ejb.interface-method view-type="remote" */
13  public void doSomething() { ... }
14
15  ...
16  }
```

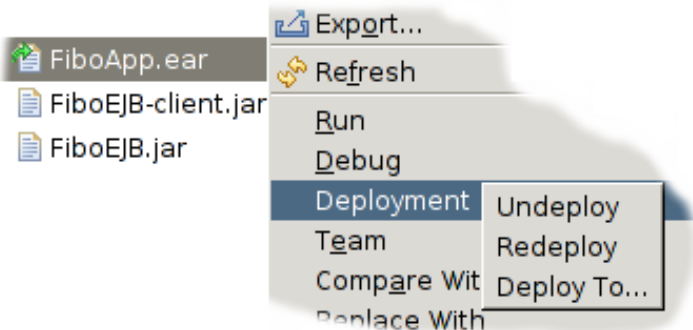
# Und nun...

## 6 JBoss-IDE + XDoclet

## JBoss-IDE + XDoclet – Projekt-Kontext



## JBoss-IDE + XDoclet – EAR-Kontext



# JBoss-IDE + XDoclet – XDoclet Konfiguration

**XDoclet Configurations**

Define the XDoclet configurations available for generation.

EJB  
 web

Up  
Down

ejbdoclet
 

- deploymentdescriptor
- fileset
- homeinterface
- jboss
- packageSubstitution
- remoteinterface

Property	Value
<input checked="" type="checkbox"/> dir	src
<input type="checkbox"/> excludes	
<input checked="" type="checkbox"/> includes	**/*Bean.java

# JBoss-IDE + XDoclet – Packaging Konfiguration

## Packaging Configurations

Define the packaging configurations available for generation.

- FiboEJB.jar
  - /avid-u3/bin
  - /avid-u3/src-gen/META-INF/ejb-jar.xml -> META-INF
  - /avid-u3/src-gen/META-INF/jboss.xml -> META-INF
- FiboEJB-client.jar
  - /avid-u3/bin
- FiboWeb.war
  - /avid-u3/bin -> WEB-INF/classes
  - /avid-u3/src-gen/WEB-INF/web.xml -> WEB-INF
  - /avid-u3/src-gen/WEB-INF/jboss-web.xml -> WEB-INF
  - /avid-u3/FiboEJB-client.jar -> WEB-INF/lib
  - /avid-u3/doc
- FiboApp.ear
  - /avid-u3/src/META-INF/application.xml -> META-INF
  - /avid-u3/FiboEJB.jar
  - /avid-u3/FiboWeb.war

# Und nun...

## 7 Ausblick EJB 3

# Ausblick EJB 3

- Annotations
- an Klassen/Methoden/Attributen
- ersetzen Deskriptorschreiben
- Ansätze von Dependency Injection
- Beispiel

```
1  @Stateful
2  public class AccountManagementBean implements AccountManagement {
3      @Inject SessionContext sessionContext;
4
5      @PostConstruct @PostActivate
6      public void initRemoteConnectionToAccountSystem() {...}
7
8      @PreDestroy @PrePassivate
9      public void closeRemoteConnectionToAccountSystem() {...}
10
11     @Remove
12     public void logOff() {...}
13 }
```

# Und nun...

## 8 Bewertung

# Bewertung

- hoher Aufwand
  - XDoclet
- alternative, leichtgewichtige Ansätze
  - Spring Framework, CUBA (Container)
  - Hibernate (Persistenz)
- JNDI-Lookups kritisch
  - Konzept schränkt Testbarkeit stark ein
  - Inversion of Control / Dependency Injection
  - siehe Spring Framework
- demnächst: EJB 3 vereinfacht Vieles

# Und nun. . .

## 9 Aufgabe

# Aufgabe

- Sun J2EE-Tutorial ab Kap. 23
- JBoss-Dokumentation/Beispiele/Wiki
- JBoss-Installation
- JBoss-Eclipse-Plugin (optional)
  - **empfehlenswerter Einstieg:** IDE-Tutorial
- Enterprise Beans erstellen
  - Integration mit Servlets/JSP
- Spielen, Debuggen, ...
- nicht besprochen, aber wichtig:
  - (Verteilte) Transaktionen (Servlet-EJB-EJB-DB)
  - Custom Databases
  - Sicherheit (JAAS)