Enterprise Java Beans

Fachhochschule Hagenberg
WS 2001 / 2002

Silbergrau Consulting & Software GmbH
Dr. Andreas Erlach

Inhaltsübersicht

- Application Server
- J2EE Architektur bzw. EJB Spezifikation
- Session, Entity und Message-Driven Beans
- EJB Deployment
- EJB Clients (Thin / Rich)
- Best Practices und Probleme
- EJB Application Server Auswahl
- Technologie-Vergleich (CORBA, COM+)

Entity Bean I

- EntityBeans stellen persistente Daten dar
  - meistens einer relationalen Datenbank
- EntityBeans werden „gemeinsam“ verwendet
  - =>SessionBeans existieren pro Client
  - rekursive Aufrufe sind innerhalb einer Transaktion möglich
- Beim Deployment muss festgelegt werden, ob Bean reentrant ist
  - führt leicht zu Inkonsistenzen
- Persistenz kann entweder vom Container oder vom Bean übernommen werden

Entity Bean II

- Container managed persistence (CMP)
  - erlaubt sind native Java Typen, serialisierbare Objekte oder Referenzen auf Home bzw. Remote Interfaces
  - Beziehungen sind erst ab CMP 2.0 möglich
    - Erlaubte Kardinalitäten 1:1, 1:N, M:N
    - Uni- und bidirektional
  - Persistenzverhalten bei Hierarchien ist nicht definiert
  - Datenbankzugriffe werden beim Deployments generiert
- Bean managed persistence (BMP)
  - Datenbankzugriffe müssen implementiert werden
  - z.B. mit JDBC bzw. SQLJ
  - ejbCreate(...), ejbRemove(), ejbFind (...), ejbLoad() und ejbStore()

Entity Bean III

- Entity Bean besteht aus
  - Home Interface zum Erzeugen bzw. Auffinden einer Entity Bean
  - Remote Interface definiert die „Business-Methoden“ der Entity Bean
  - EnterpriseBean implementiert die Methoden des Home/Remote Interface

Anm.: Methoden, Parameter und Returnwerte müssen den RMI Regeln entsprechen

Entity Bean IV

- Anforderungen an ein EntityBean Home Interface
  - Interface muss von javax.ejb.EJBHome abgeleitet sein
  - Methoden müssen den RMI/IIOP Regeln entsprechen
    - Argumente und Return values müssen gültige RMI/IIOP Typen sein und die throws-Clause muss java.rmi.RemoteException enthalten
  - Home Interface kann eine oder mehrere create(...) Methoden definieren
    - Zu jeder create-Methode muss es entsprechende ejbCreate- bzw. ejbPostCreate-Methoden geben
    - Return type einer create-Methode muss das Remote Interface der Entity Bean sein und die throws-Clause muss zumindest javax.ejb.CreateException enthalten

Entity Bean V

- Anforderungen an ein EntityBean Home Interface
  - Home Interface kann eine oder mehrere find(...) Methoden definieren
    - Zu jeder find-Methode muss es eine entsprechende ejbFind-Methode geben
    - Der Return type einer find-Methode muss das Remote Interface der Entity Bean oder eine Collection sein und die throws-Clause muss zumindest javax.ejb.FinderException und enthalten
  - jede create(...) und findXXX(...) Methode des Home Interface wird mit einem „ejb“ als Prefix implementiert
    - z.B. HomeInterface.create() - EntityBean.ejbCreate()
    - bei CMP implementiert das Bean keine ejbFindXXX(...) Methoden

Entity Bean VI

- Anforderungen an ein EntityBean Home Interface
  - ejbCreate(...) Methoden liefert einen Primary Key
  - zu jeder ejbCreate Methode muss es eine entsprechende ejbPostCreate Methode geben
  - ejbFindXXX(...) Methoden liefern entweder
    - einen Primary Key oder
    - eine Enumeration oder eine Collection von Primary Keys

Entity Bean VII

- Anforderungen an eine Entity Bean Klasse
  - Muss das javax.ejb.EntityBean implementieren
  - Muss public und darf weder final noch abstract sein
  - Muss einen public Default-Konstruktur definieren
  - Darf die finalize-Methode nicht implementieren
  - Kann (muss nicht) das Remote Interface implementieren
  - Muss die entsprechenden Business- und die ejbCreate-, ejbPostCreate- und ejbFind-Methoden implementieren
  - Die Argumente und Return types der Business- und ejbCreate-, ejbPostCreate- und ejbFind-Methoden müssen den RMI/IIOP Regeln entsprechen
  - Kann beliebige andere (Hilfs-)Methoden implementieren

Entity Bean VIII

- Container-managed Persistence in EJB V2.0
  - Container-managed persistent Fields (CMP Fields)
    - Nicht mehr „public“ Instanzvariablen wie in EJB V1.x
  - Container-managed relationship Fields (CMR Fields)
    - Nicht unterstützt durch EJB V1.x
    - Grundlage sind Local Interfaces
    - Möglich sind 1:1, 1:N und M:N
    - Referentielle Integrität wird auf Objektebene sichergestellt
  - CMP bzw. CMR Fields werden definiert durch
    - Abstrakte get bzw. set Methoden auf der Entity Bean und den Deployment Descriptor
  - Entity Bean mit CMP V2.0 muss public und abstract sein

Entity Bean IX

- Entity Bean kann Non- bzw. Reentrant sein
  - Wird beim Deployment definiert
  - Non-reentrant
    - Ein mehrmaliger Aufruf einer Entity Bean im selben Transaktions-Context ist nicht erlaubt (verursacht eine java.rmi.RemoteException durch den Container)
    - Problematisch bei Loopback-calls (z.B. A ruft B auf und B ruft wiederum A auf)
  - Reentrant
    - Erfordert besondere Berücksichtigung bei der Implementierung!

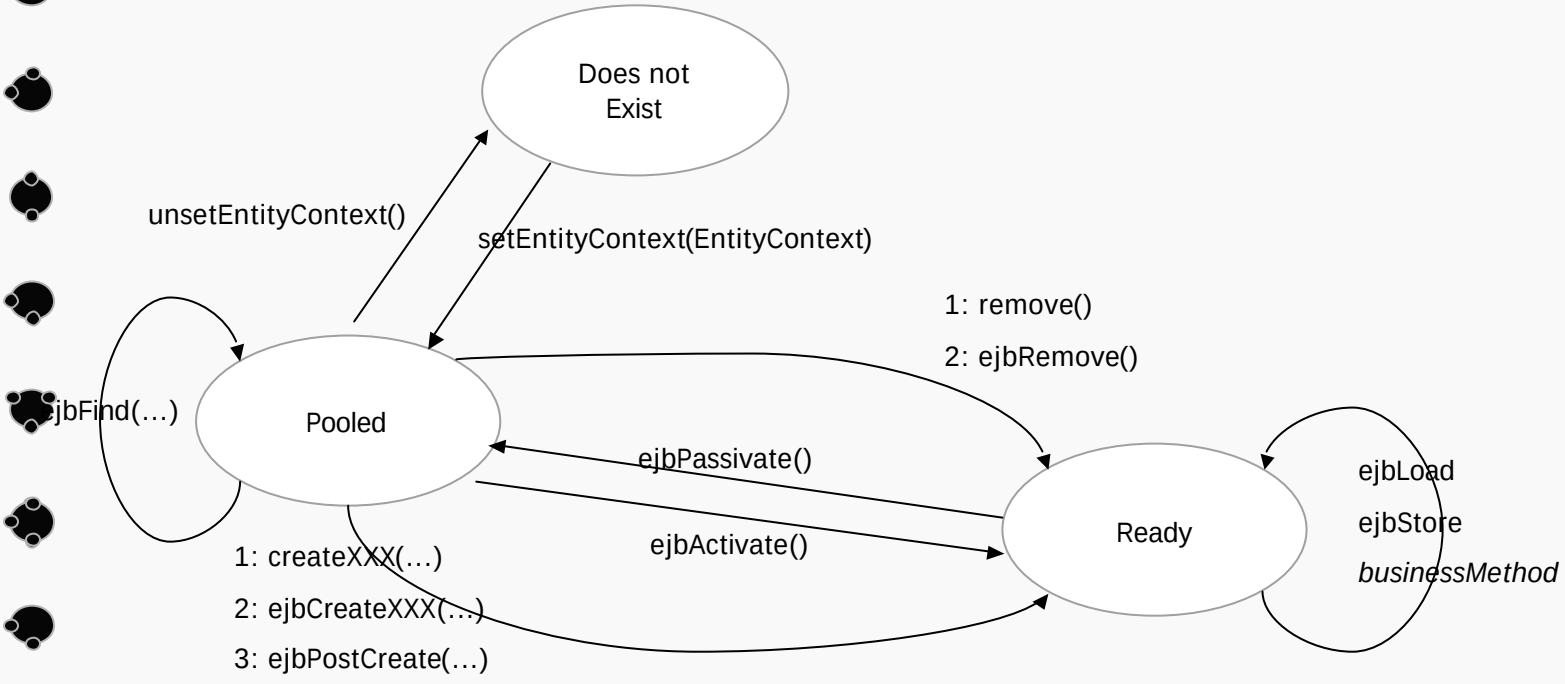
Entity Bean X

- **public interface EntityBean extends EnterpriseBean**
  - **The EntityBean interface is implemented by every entity enterprise Bean class**
- **void setEntityContext(EntityContext ctx) throws EJBException, RemoteException;**
  - **Set the associated entity context**
- **void unsetEntityContext() throws EJBException, RemoteException;**
  - **Unset the associated entity context**
- **void ejbRemove() throws RemoveException, EJBException, RemoteException;**
  - **A container invokes this method before it removes the EJB object that is currently associated with the instance**

Entity Bean XI

- **void ejbActivate() throws EJBException, RemoteException;**
  - **A container invokes this method when the instance is taken out of the pool of available instances to become associated with a specific EJB object**
- **void ejbPassivate() throws EJBException, RemoteException;**
  - **A container invokes this method on an instance before the instance becomes disassociated with a specific EJB object**
- **void ejbLoad() throws EJBException, RemoteException;**
  - **A container invokes this method to instruct the instance to synchronize its state by loading it state from the underlying database**
- **void ejbStore() throws EJBException, RemoteException;**
  - **A container invokes this method to instruct the instance to synchronize its state by storing it to the underlying database**

Entity Bean XII



Life Cycle einer Entity Bean

Entity Bean XIII

Bean method	Erlaubte Operationen in einer Bean Methode
Constructor	–
setEntityContext unsetEntityContext	EntityContext methods: *getEJBHome* JNDI access to java:comp/env
ejbCreate	EntityContext methods: *getEJBHom*e, *getCallerPrincipa*l, *getRollbackOnl*y, *isCallerInRol*e, *setRollbackOnly* JNDI access to java:comp/env, Resource manager access Enterprise bean access
ejbPostCreate	EntityContext methods: *getEJBHom*e, *getCallerPrincipa*l, *getRollbackOnl*y, *isCallerInRol*e, *setRollbackOnl*y, *getEJBObjec*t, *getPrimaryKey* JNDI access to java:comp/env, Resource manager access Enterprise bean access
ejbRemove	EntityContext methods: *getEJBHom*e, *getCallerPrincipa*l, *getRollbackOnl*y, *isCallerInRol*e, *setRollbackOnl*y, *getEJBObjec*t, *getPrimaryKey* JNDI access to java:comp/env, Resource manager access Enterprise bean access

Entity Bean XIV

Bean method	Erlaubte Operationen in einer Bean Methode
ejbFind	EntityContext methods: *getEJBHom*e, *getCallerPrincipa*l, *getRollbackOnl*y, *isCallerInRol*e, *setRollbackOnly* JNDI access to java:comp/env, Resource manager access Enterprise bean access
ejbActivate ejbPassivate	EntityContext methods: *getEJBHom*e, *getEJBObjec*t, *getPrimaryKey* JNDI access to java:comp/env
ejbLoad ejbStore	EntityContext methods: *getEJBHom*e, *getCallerPrincipa*l, *getRollbackOnl*y, *isCallerInRol*e, *setRollbackOnly*, *getEJBObjec*t, *getPrimaryKey* JNDI access to java:comp/env, Resource manager access Enterprise bean access
business method from remote interface	EntityContext methods: *getEJBHom*e, *getCallerPrincipa*l, *getRollbackOnl*y, *isCallerInRol*e, *setRollbackOnly*, *getEJBObjec*t, *getPrimaryKey* JNDI access to java:comp/env, Resource manager access Enterprise bean access

Entity Bean XV

- **public interface EntityContext extends EJBContext**
  ...
- **Object getPrimaryKey() throws IllegalStateException;**
  - **Returns the entity bean‘s primary key**

  ...

  - *A1: Ansonsten analog zu SessionContext*

Entity Bean XVI

- Anforderungen an die Primary Key Klasse
  - Muss ein gültiger RMI-IIOP Typ sein
  - Muss passende hashCode() und equals(Object) implementieren
  - Muss die Fields mit den selben Namen wie Entity Bean implementieren

  - *A1: Seit EJB Spezifikation 1.1 ist auch String als Primary Key Klasse erlaubt*

Entity Bean XVII

- Entity Bean sollte bei Fehlern die keine Anwendungsfehler sind eine EJBException signalisieren
  - in EJB 1.0 eine java.rmi.RemoteException

Local und Remote Interfaces I

- Entity- und auch Session-Beans können in EJB V2.0 sowohl ein
  - Remote Interface (wie bisher) als auch ein
  - Local Interface haben
- Typischerweise bietet eine Enterprise Bean nur eins davon an
- Stellen Grundlage für CMR Fields dar
  - Entity Bean A referenziert B über das Local Interface von B

Local und Remote Interfaces II

- Remote Interfaces erlauben „verteilte“ Clients
- Local Interfaces erlauben „lokalen“ Clients einen „lightweight“ Zugriff
  - Standard Java Interface (kein RMI)
  - Basis Interfaces sind EJBLocalHome und EJBLocalObject

Beispiel: BMP Entity Bean I

- Verwaltung eines Kontos mittel BMP
  - Einzahlung bzw. Abhebung
- Erstellt werden müssen folgende Interfaces bzw. Klassen
  - AccountHome – Home Interface
  - Account – Local Interface
  - AccountBean – Entity Bean
- Session Bean für
  - Kontoinformationen auslesen
  - Überweisung durchführen

Beispiel: BMP Entity Bean II

```
import java.rmi.*;
import javax.ejb.*;

public interface AccountHome extends EJBLocalHome {
   AccountEntity create(String number,
       double balance, String currency)
       throws CreateException;
   AccountEntity findByPrimaryKey(String number)
       throws FinderException;
}
```

Beispiel: BMP Entity Bean III

```
import java.rmi.*;
import javax.ejb.*;

public interface Account extends EJBLocalObject {
   String getNumber();
   double getBalance();
   String getCurrency();
   void deposit(double amount, String currency)
       throws IllegalCurrencyException;
   void withdraw(double amount, String currency)
       throws InsufficientBalanceException,
        IllegalCurrencyException;
}
```

Beispiel: BMP Entity Bean IV

```
import javax.naming.*;
import java.rmi.*;
import javax.ejb.*;
import javax.rmi.*;

public class AccountBean implements EntityBean {
   private String currency;
   private double balance;
   private String number;
   public void setEntityContext(EntityContext ctx) {}
   public void ejbActivate() {}
   public void ejbPassivate() {}
   public void unsetEntityContext() {}
}
```

Beispiel: BMP Entity Bean V

```
public String getNumber() {
    return number;
}
public double getBalance() {
    return balance;
}
protected void setBalance(double newBalance) {
    balance = newBalance;
}
public String getCurrency() {
    return currency;
}
```

Beispiel: BMP Entity Bean VI

```
public String ejbCreate(String newNumber,
    double newBalance, String newCurrency)
    throws CreateException {
    number = newNumber;
    balance = newBalance;
    currency = newCurrency;
    // Insert Account into Database
    return null;
}
public void ejbPostCreate(String newNumber,
    double newBalance, String newCurrency) {}
```

Beispiel: BMP Entity Bean VII

```
public String ejbFindByPrimaryKey(String number)
    throws FinderException {
    // Check Account exists in Database
    return number;
}
public void ejbRemove()throws RemoveException  {
    // Remove Account from Database
}
public void ejbLoad() {
    // Restore Account from Database
}
public void ejbStore() {
    // Store Account into Database
}
```

Beispiel: BMP Entity Bean VIII

```
public void deposit(double amount, String currency)
   throws IllegalCurrencyException {
    if (!currency.equals(getCurrency()) {
       throw new IllegalCurrencyException();
    }
    setBalance(getBalance() + amount);
}
```

Beispiel: BMP Entity Bean IX

```
  public void withdraw(double amount, String currency)
     throws InsufficientBalanceException,
       IllegalCurrencyException {
     if (!currency.equals(getCurrency()) {
        throw new IllegalCurrencyException();
     }
     if (getBalance() < amount) {
        throw new InsufficientBalanceException();
     }
     setBalance(getBalance() - amount);
  }
}
```

Beispiel: BMP Entity Bean X

```
import java.rmi.*;
import javax.ejb.*;

public interface eBankingHome extends EJBHome {
   eBanking create()
       throws CreateException, RemoteException;
}
```

Beispiel: BMP Entity Bean XI

```
import java.rmi.*;
import javax.ejb.*;
public interface eBanking extends EJBObject {
  AccountInfo getAccountInfo(String number)
      throws RemoteException, FinderException;
  void transfer(String fromAccount,
      String toAccount,
      double amount, String currency)
      throws InsufficientBalanceException,
       IllegalCurrencyException, RemoteException;
}
```

Beispiel: BMP Entity Bean XII

```
public class eBankingBean implements SessionBean {
   public void ejbCreate() {}
   public void setSessionContext(SessionContext ctx) {}
   public void ejbRemove() {}
   public void ejbActivate() {}
   public void ejbPassivate() {}
   public AccountInfo getAccountInfo(String number)
       throws FinderException {
       // Lookup Account Home
       // Create and Populate AccountInfo
       return accountInfo;
   }
```

Beispiel: BMP Entity Bean XIII

```
  public void transfer(String fromAccount,
      String toAccount,
      double amount, String currency)
      throws FinderException,
      InsufficientBalanceException,
       IllegalCurrencyException {
      // Lookup fromAccount
      // Lookup toAccount
      // Withdraw amount from fromAccount
      // Deposit amount into toAccount
  }
}
```

Beispiel: BMP Entity Bean XIV

- Entity Bean muss deployed werden
  - Was ist zu beachten?

  => Übungen

Entity Bean XVIII

- Auffinden des Home Interface

```
Context ic = new InitialContext();
Object obj =
  ic.lookup(“java:comp/env/ejb/MyEntityBean”);
MyHome home =
  (MyHome) PortableRemoteObject.narrow(obj,
  MyHome.class);
```

  - *A1: Der Cast-Operator anstelle von PortableObject.narrow kann bei einem Container mit RMI-IIOP einen Fehler verursachen*

Entity Bean XIX

- Erzeugen einer Entity Bean

```
MyEntity entity = home.create(„Garbage“);
```

  - *A1: Mögliche Exceptions sind RemoteException, CreateException und DuplicateKeyException*
  - *A2: Das Home Interface kann null oder mehrere create-Methoden definieren*
  - *A3: Die Entity Bean muss entsprechende ejbCreate() Methode implementieren!*

Entity Bean XX

- Erzeugen einer Entity Bean aus der Datenbank

```
MyEntity entity = home.findByPrimaryKey(„Garbage“);
```

  - *A1: Mögliche Exceptions sind RemoteException, FinderException und ObjectNotFoundException*
  - *A2: Das Home Interface kann weitere find-Methoden definieren (z.B. findByName(String name))*
  - *A3: Die Entity Bean muss entsprechende ejbFind-Methoden implementieren!*
  - *A4: Kann eine Collection (PrimaryKeys) liefern. Unter JDK 1.1 eine java.util.Enumeration und ab Java2 eine java.util.Collection*

Entity Bean XXI

- Löschen einer Entity Bean

```
entity.remove();
home.remove(entity.getHandle());
Home.remove(entity. getPrimaryKey());
```

  - *A1: Mögliche Exceptions sind RemoteException und RemoveException*
  - *A2: Die Entity Bean muss eine ejbRemove() Methode implementieren!*
  - *A3: Die Entity Bean wird dadurch aus der Datenbank gelöscht!*

silbergrau
Consulting & Software GmbH

Danke für die Aufmerksamkeit!

Silbergrau Consulting & Software GmbH
Dr. Andreas Erlach