

# Internetprogrammierung 2006

## Enterprise JavaBeans—Entity Beans

Peter Thiemann

Universität Freiburg, Germany

Vorlesung Internetprogrammierung, 03.07.2006

# Inhalt

## Entity Beans

- Entity Bean Interfaces
- Entity Bean Verwendung
- Bean-Klasse
- Lifecycle

## Deployment

- Deployment Descriptor
- Durchführung

# Entity Beans

- ▶ Entity-Bean  $\leftrightarrow$  Relation in einer relationalen Datenbank
- ▶ Entity-Bean Instanz  $\leftrightarrow$  Tupel der Relation
- ▶ Jede Entity-Bean Instanz besitzt einen Primärschlüssel
- ▶ Zugriff auf Entity-Bean wie auf Objekt
- ▶ Objektattribut entspricht Spalte der Tabelle
- ▶ Automatischer Abgleich mit Datenbank durch den Container
- ▶ Entity-Beans können von mehreren Klienten benutzt werden.

# Entity-Bean Persistenzarten

- ▶ Container regelt den Ablauf der Bean-Operationen
- ▶ Container-Managed Persistence (CMP)
  - ▶ Container generiert alle Datenbankoperationen
  - ▶ Deployment-Descriptor definiert Abbildung auf Datenbank
  - ▶ Code unabhängig von Datenbank und EJB-Container
- ▶ Bean-Managed Persistence (BMP)
  - ▶ EJB-Provider muss Callbackmethoden mit Datenbankoperationen schreiben
  - ▶ Code abhängig von Datenbank und EJB-Container
  - ▶ Nicht-standard Abbildungen realisierbar

# Beispiel: Entity Bean Address

Remote-Home-Interface und Remote-Interface

```
public interface AddressRemote
    extends javax.ejb.EJBObject {
    public String getStreetAddress()
        throws RemoteException;
    public void setStreetAddress(String sa)
        throws RemoteException;
    public String getCity()
        throws RemoteException;
    public void setCity(String city)
        throws RemoteException;
    /* get/set methods for all attributes */
}
```

- ▶ Externes Interface zum Bean

# Beispiel: Entity Bean Address

## Remote-Home-Interface

```
public interface AddressHomeRemote  
    extends javax.ejb.EJBHome {  
    public AddressRemote create(Integer pk)  
        throws RemoteException, CreateException;  
    public AddressRemote findByPrimaryKey(Integer pk)  
        throws RemoteException, FinderException;  
}
```

- ▶ Weitere `createXYZ()` Methoden möglich
- ▶ Weitere `findXYZ()` Methoden möglich
- ▶ `selectXYZ()` Methoden

# Ansprechen eines Entity Bean

```
AddressHomeRemote addressHome = /* from JNDI */  
  
// Neue Adresse erzeugen  
AddressRemote myAddress =  
    addressHome.create (new Integer (69));  
myAddress.setStreetAddress ("Georges-Koehler-Allee");  
myAddress.setCity ("Freiburg");  
myAddress.setState ("");  
myAddress.setZip ("79110");  
myAddress.setCountry ("Germany");  
  
// Suche nach einer Adresse  
AddressRemote yourAddr =  
    addressHome.findByPrimaryKey (new Integer (42));  
String yourCountry = yourAddr.getCountry ();
```

# Sonstiges

## Zugriff aufs Home-Interface

```
public class AddressUtil {  
    // Zugriff aufs Remote-Home-Interface mit Hilfe von JNDI  
    public static AddressHomeRemote getHomeRemote () {  
        try {  
            Context jndiContext = getInitialContext ();  
            Object ref =  
                jndiContext.lookup ("java:comp/env/ejb/AddressHomeRemote");  
            AddressHomeRemote home = (AddressHomeRemote)  
                PortableRemoteObject.narrow (ref, AddressHomeRemote.class);  
            return home;  
        } catch (Exception e) {  
            e.printStackTrace ();  
            return null;  
        }  
    }  
}
```



# Sonstiges

## Zugriff auf InitialContext

- ▶ Abhängig vom Server/Container-Provider

```
public static Context getInitialContext ()  
    throws javax.naming.NamingException {  
    Properties p = new Properties ();  
    /* put some magic into the properties */  
    /* depending on the EJB container */  
    return new javax.naming.InitialContext (p);  
}  
}
```

# Implementierung der Bean-Klasse

```
import javax.ejb.EntityContext ;

public abstract class AddressBean
    implements javax.ejb.EntityBean {
    // container methods
    public Integer ejbCreate (Integer id) {
        this.setId (id);
        return null;
    }
    public void ejbPostCreate (Integer id) {
```

- ▶ `home.create (args)` ruft auf
  - ▶ `ejbCreate (args)` und dann
  - ▶ `ejbPostCreate (args)`
- ▶ Alle müssen gleiche Signatur haben und in Bean-Klasse implementiert

## Implementierung der Bean-Klasse/2

```
// business methods  
public abstract void setId (Integer id);  
public abstract Integer getId ();  
  
public abstract void setStreetAddress (String sa);  
public abstract String getStreetAddress ();  
  
public abstract void setCity (String city);  
public abstract String getCity ();
```

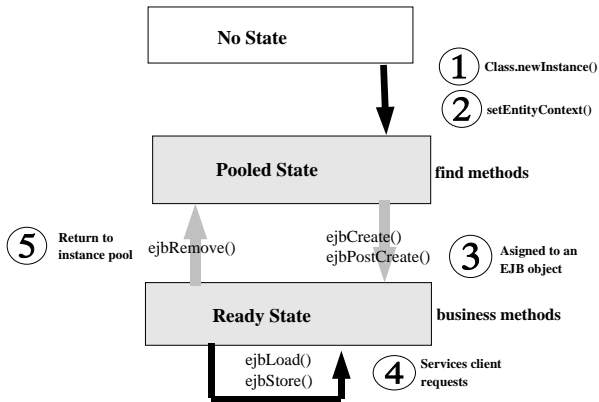
- ▶ id Feld ist Primärschlüssel
- ▶ get/set Methoden für Felder entsprechend Local- und Remote-Interface
- ▶ (restliche Methoden analog)

# Implementierung der Bean-Klasse/3

```
// further container methods  
// all have empty implementations  
public void setEntityContext (EntityContext ctx) {}  
public void unsetEntityContext () {}  
  
public void ejbActivate () {}  
public void ejbPassivate () {}  
  
public void ejbLoad () {}  
public void ejbStore () {}  
public void ejbRemove () {}  
}
```

- ▶ Aufruf durch den Container
- ▶ Implementierung des Lifecycle
- ▶ Meist leer für container-managed persistence

# Lifecycle eines Entity Beans



# Deployment Descriptor für AddressBean

## Grobstruktur

```
<?xml version="1.0" encoding="UTF-8" ?>
<ejb-jar
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">
  <enterprise-beans>
    <entity> <!-- description of entity bean --> </entity>
    <session><!-- description of session bean --></session>
  </enterprise-beans>
  <assembly-descriptor>
    <!-- security configuration -->
    <!-- transaction configuration -->
  </assembly-descriptor>
</ejb-jar>
```

# Deployment Descriptor für AddressBean

In <enterprise-beans>

```
<entity>
  <ejb-name>AddressEJB</ejb-name>
  <home>proglang.j2ee.ejb.AddressHomeRemote</home>
  <remote>proglang.j2ee.ejb.AddressRemote</remote>
  <ejb-class>proglang.j2ee.ejb.AddressBean</ejb-class>
  <persistence-type>Container</persistence-type>
  <prim-key-class>java.lang.Integer</prim-key-class>
  <reentrant>False</reentrant>
  <abstract-schema-name>Address</abstract-schema-name>
  <cmp-field><field-name>streetAddress</field-name></cmp-field>
  <cmp-field><field-name>city</field-name></cmp-field>
  <cmp-field><field-name>state</field-name></cmp-field>
  <cmp-field><field-name>zip</field-name></cmp-field>
  <cmp-field><field-name>country</field-name></cmp-field>
  <primkey-field>id</primkey-field>
  <security-identity><use-caller-identity/></security-identity>
</entity>
```

# Deployment Descriptor für AddressBean

In <assembly-descriptor>

```
<security-role>
  <description>Everyone with full access to AddressEJB.</description>
  <role-name>everyone</role-name>
</security-role>
<method-permission>
  <role-name>everyone</role-name>
  <method>
    <ejb-name>AddressEJB</ejb-name>
    <method-name>*</method-name>
  </method>
</method-permission>
<container-transaction>
  <method>
    <ejb-name>AddressEJB</ejb-name>
    <method-name>*</method-name>
  </method>
  <trans-attribute>Required</trans-attribute>
</container-transaction>
```



# Durchführen des Deployments

- ▶ Für Deployment erstelle `jar` Archiv mit
  - ▶ Interfaces
    - ▶ Remote-Home-Interface
    - ▶ Remote-Interface
  - ▶ Bean-Klasse
  - ▶ Deployment Deskriptor
- ▶ Interfaces und Bean-Klassen in Verzeichnis entsprechend der Package
- ▶ Deployment-Deskriptor (`ejb-jar.xml`) in Verzeichnis `META-INF`
- ▶ Alternativen
  - ▶ Deployment-Wizard des Application-Servers
  - ▶ Ant-Task für Deployment