

# Internetprogrammierung 2006

## Enterprise JavaBeans—Einführung

Peter Thiemann

Universität Freiburg, Germany

Vorlesung Internetprogrammierung, 28.06.2006

# Inhalt

## Allgemeines

Motivation

Rollen

Aufbau einer EJB

Arten von Beans

## Session Beans

Stateless Session Bean

Deployment

Stateful Session Bean

Besonderheiten

# Enterprise JavaBeans (EJB)

- ▶ Serverseitige Software-Komponenten (auf Application-Server)
  - ▶ kapseln Applikationslogik
  - ▶ wiederverwendbar
  - ▶ verteilt
  - ▶ leben in Container, der Infrastruktur (Dienste) zur Verfügung stellt
- ▶ EJB-Spezifikation
  - ▶ Standard für Entwicklung und Deployment von Server-Komponenten
  - ▶ definiert Interfaces zwischen
    - ▶ EJB und Container
    - ▶ Container und Application-Server
    - ▶ Container und Client

# EJB Motivation

- ▶ Wichtige Dienste werden vom Container zur Verfügung gestellt
  - ▶ Management verteilter Transaktionen
  - ▶ Sicherheit
  - ▶ Persistenz
  - ▶ Ressourcen-Pooling
  - ▶ Load balancing
  - ▶ Multithreading
  - ▶ Verzeichnisdienst
- ▶ Unabhängigkeit von bestimmtem Anbieter (J2EE-Spezifikation)
- ▶ Trennung von Business-Logik und Präsentation

# EJB Rollen

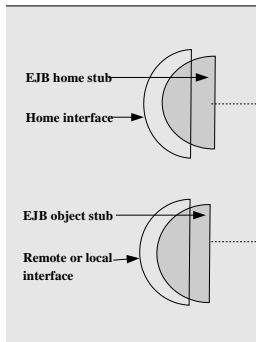
Die EJB-Spezifikation definiert verschiedene Rollen für die unterschiedlichen Aufgaben bei Entwicklung, Deployment und Wartung einer J2EE-Anwendung

- ▶ Server-Provider:
  - ▶ Anbieter eines Application-Servers
  - ▶ z.B. BEA oder IBM
- ▶ Container-Provider:
  - ▶ Anbieter von Containern für bestimmte Komponenten (z.B. EJB-Container)
  - ▶ Die meisten Server-Provider sind auch Container-Provider

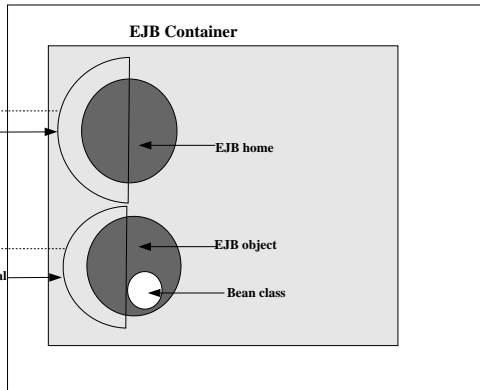
## EJB Rollen/2

- ▶ EJB-Provider:
  - ▶ Entwickler von EJBs
  - ▶ Muss sich nicht um Infrastruktur kümmern
- ▶ Application-Assembler:
  - ▶ Kümmert sich um Architektur einer Anwendung und setzt Komponenten entsprechend zusammen
  - ▶ Verwendet vorgefertigte EJBs
  - ▶ Muss nicht programmieren können
- ▶ Deployer:
  - ▶ Kennt Anwendungsarchitektur
  - ▶ Konfiguration der Anwendung und des Application-Servers

## Client



## EJB Server



# EJB Interfaces

- ▶ Container:  
Schnittstelle zwischen EJBs und Außenwelt
- ▶ Bean-Klasse: Implementierung; implementiert Interface zum Container
- ▶ Klienten greifen stets über Interfaces auf EJB zu
  - ▶ Home-Interface: Lifecycle Methoden (create, remove, find, ...)
  - ▶ Interface: Business Methoden (Anwendungslogik)
- ▶ Diese Interfaces kommen jeweils in zwei Varianten
  - Remote** zum Zugriff außerhalb des Containers
  - Local** für Beans innerhalb des gleichen Containers  
effizienter, da kein Remote-Protokoll erforderlich



## EJB Interfaces/2

Interfaces definiert durch EJB-Provider (Entwickler)

- ▶ Remote-Home-Interface

<: javax.ejb.EJBHome <: java.rmi.Remote

- ▶ Remote-Interface

<: javax.ejb.EJBObject <: java.rmi.Remote

- ▶ Local-Home-Interface <: javax.ejb.EJBLocalHome

- ▶ Local-Interface <: javax.ejb.EJBLocalObject

- ▶ Objekte mit Interfaces EJBHome und EJBObject werden vom Application-Server generiert

- ▶ **Achtung!**

Bean-Klasse implementiert **keines** dieser Interfaces, **aber** sie muss Methoden mit ähnlichen Namen und gleicher Signatur haben.

# Arten von Beans

- ▶ Session Beans
  - ▶ Kapseln die Applikationslogik
  - ▶ Unterarten: stateful und stateless
- ▶ Entity Beans
  - ▶ Repräsentieren persistente Business-Objekte
  - ▶ Beispiel: Kunde, Adresse
  - ▶ Unterarten: container-managed und bean-managed persistency (CMP/BMP)
- ▶ Message Driven Beans
  - ▶ Asynchrone Nachrichtenverarbeitung

# Session Beans

- ▶ Session Beans kapseln die Applikationslogik
- ▶ Stateless Session Beans  
(Bsp: Konverterfunktionen, Logger)
  - ▶ Keine feste Bindung an einen Klienten
  - ▶ Kein klientenspezifischer Zustand  
(wohl aber offene Dateien, Datenbankverbindungen, etc)
  - ▶ Alle Instanzen äquivalent
- ▶ Stateful Session Beans (Bsp: Einkaufswagen)
  - ▶ Feste Bindung an einen Klienten
  - ▶ Klientenspezifischer Zustand für die Dauer einer Sitzung
  - ▶ Activation/Passivation zur Lastregulierung

# Remote-Home-Interface eines Stateless Session Bean

```
import java.rmi.RemoteException;  
import javax.ejb.CreateException;  
  
public interface ConverterHomeRemote  
    extends javax.ejb.EJBHome {  
  
    public ConverterRemote create()  
        throws RemoteException, CreateException;  
  
}
```

# Remote-Interface eines Session Bean

```
import java.rmi.RemoteException;
import javax.ejb.FinderException;

public interface ConverterRemote
    extends javax.ejb.EJBObject {

    // business methods
    public double kmToMiles(double km)
        throws RemoteException;

    public double milesToKm(double miles)
        throws RemoteException;
}
```

# Bean-Klasse eines Session Bean

```
import ConverterRemote ;
import ConverterHomeRemote ;

public class ConverterBean
    implements javax.ejb.SessionBean {

    public void ejbCreate () {
        // Do nothing.
    }
}
```

- ▶ `ejbCreate` Methode wird vom Container aufgerufen, wenn neue Beaninstanz benötigt wird
- ▶ je eine `ejbCreate` Methode pro `create` Methode im Remote-Home-Interface

# Bean-Klasse eines Session Bean

## Business Methoden

```
public double kmToMiles(double km) {  
    return km / 1.6;  
}  
  
public double milesToKm(double miles) {  
    return miles * 1.6;  
}
```

- ▶ werden vom Klienten über Remote-Interface aufgerufen
- ▶ Argumente und Ergebnisse müssen serialisierbar sein

# Bean-Klasse eines Session Bean

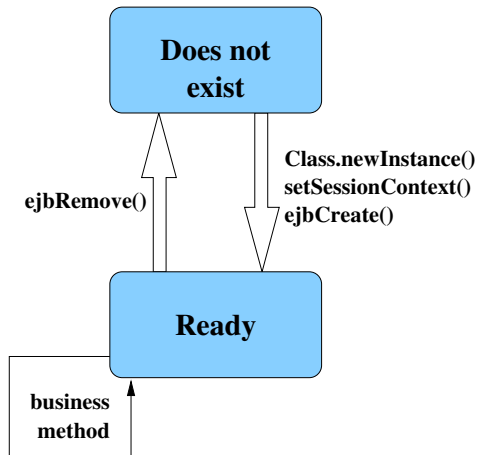
## Container Callback Methoden

```
public void ejbRemove(){}  
public void ejbActivate(){}  
public void ejbPassivate(){}  
public void setSessionContext(  
    javax.ejb.SessionContext cntx){}  
}
```

- ▶ werden vom Container bei Lifecycle-Events aufgerufen
- ▶ meist leer bei stateless Session Beans



# Lebenslauf Stateless Session Bean



# Deployment von Beans

- ▶ Deployment = Bereitstellen und Konfigurieren des Beans auf dem Application-Server (Container)
- ▶ Warum Deployment?
  - ▶ Wiederverwendbarkeit erfordert Flexibilität und Konfigurierbarkeit
  - ▶ Anpassung an tatsächliche Zielplattform
    - ▶ Nebenläufigkeit, Verteilung
    - ▶ Datenbanklayout
    - ▶ Sicherheitsmodell
- ▶ Deployer: erstellt **Deployment Descriptor**
  - ▶ Datei META-INF/ejb-jar.xml

# Deployment Descriptor für ConverterBean

## 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>
    <session>
      <display-name>Convert km to miles.</display-name>
      <ejb-name>ConverterEJB</ejb-name>
      <home>ConverterHomeRemote</home>
      <remote>ConverterRemote</remote>
      <ejb-class>ConverterBean</ejb-class>
      <session-type>Stateless</session-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

# Stateful Session Bean

## Interfaces

```
import java.rmi.RemoteException;
import javax.ejb.CreateException;
public interface CounterHomeRemote
    extends javax.ejb.EJBHome {
    public CounterRemote create()
        throws RemoteException, CreateException;
}
```

```
import java.rmi.RemoteException;
import javax.ejb.FinderException;
public interface CounterRemote
    extends javax.ejb.EJBObject {
    public int bump(int step)
        throws RemoteException;
}
```

# Stateful Session Bean

## Bean-Klasse

```
import CounterRemote;  
import CounterHomeRemote;  
  
public class CounterBean  
    implements javax.ejb.SessionBean {  
    private SessionContext ctx;  
    private int count;  
  
    public void ejbCreate() {  
        count = 0;  
    }  
  
    public int bump(int step) {  
        return count += step;  
    }  
}
```

# Stateful Session Bean

Bean-Klasse/2

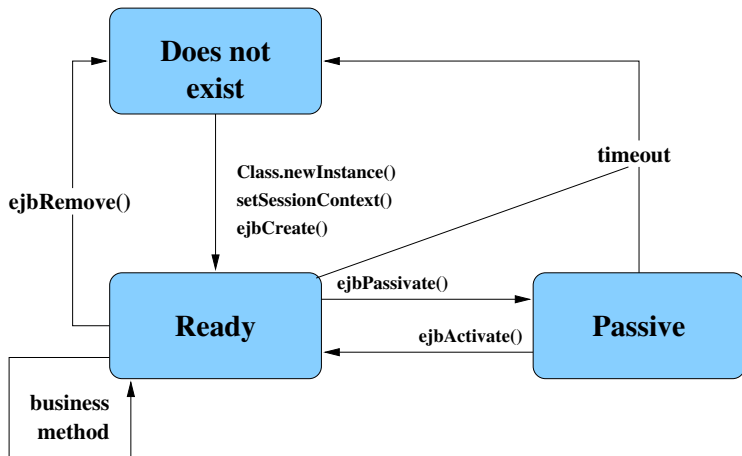
```
// container callback methods  
public void ejbRemove(){}  
public void ejbActivate(){}  
public void ejbPassivate(){}  
public void setSessionContext(SessionContext cntx) {  
    this.ctx = cntx;  
}  
}
```

# Deployment Descriptor für CounterBean

## 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>
    <session>
      <display-name>Counter.</display-name>
      <ejb-name>CounterEJB</ejb-name>
      <home>CounterHomeRemote</home>
      <remote>CounterRemote</remote>
      <ejb-class>CounterBean</ejb-class>
      <session-type>Stateful</session-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

# Lebenslauf Stateful Session Bean





# Besonderheiten

- ▶ Stateless Session Bean
  - ▶ `create` parameterlos
  - ▶ klientspezifischer Zustand wie Datenbankverbindung
  - ▶ (Erhaltung von Instanzen nicht garantiert)
  - ▶ oft direkte Datenbankzugriffe
- ▶ Stateful Session Bean
  - ▶ `create` kann Startzustand als Parameter haben
  - ▶ **Passivierung**
    - ▶ Zustand der Sitzung (*conversational state*) wird serialisiert oder anderweitig gespeichert
    - ▶ Beaninstanz wird zerstört oder anderweitig verwendet
  - ▶ **Aktivierung** (Umkehrung)

# Ansprechen eines Beans

```
Context jndiContext =
    new javax.naming.InitialContext ();
Object ref = jndiContext.lookup
    ("java:comp/env/ejb/CounterHomeRemote");
CounterHomeRemote counterHome =
    (CounterHomeRemote)
        PortableRemoteObject.narrow
            (ref, CounterHomeRemote.class);

// create a new counter
CounterRemote myCounter =
    counterHome.create();
myCounter.bump (15);
```