silbergrau
Consulting & Software GmbH

Enterprise Java Beans (EJB)

Fachhochschule Hagenberg
WS 2002 / 2003

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)
- Probleme, Design Patterns, Best Practices
- EJB Application Server Auswahl
- Technologie-Vergleich (CORBA, .Net)

Ressourcen I

- Literatur
  - Mastering Enterprise Java Beans & J2EE, Wiley & Sons
  - EJB Design Patterns, Wiley Computer Publishing
  - Professional Java Server Programming, Wrox Press Ltd.
  - Enterprise JavaBeans, O'Reilly & Associates
  - Enterprise JavaBeans – Developing Component-Based Distributed Applications, Addison Wesley
  - Designing Enterprise Applications with J2EE
  - Java Enterprise in a Nutshell, O'Reilly & Associates
  - Java 2 EE: Plattform and Component Specification
  - Java Distributed Computing, O'Reilly & Associates
  - ...

Ressourcen II

- Tutorials
  - developer.java.sun.com/developer/onlineTraining/Beans/EJBTutorial/
  - java.sun.com/j2ee/tutorial/
  - www.orionserver.com/tutorials/ejbtutorial
  - www.execpc.com/~gopalan/java/java_tutorial
  - www.ejbtut.com/
  - webster.cs.uga.edu/~sanjeev/ejb/
  - ...

Ressourcen III

- W3 Quellen
  - www.javasoft.com/j2ee/
  - www.javasoft.com/blueprints/
  - www.javaworld.com/
  - www.theserverside.com/
  - www.mgm-edv.de/ejbsig/ejbsig.html
  - www.ejbean.com/
  - www.flashline.com/components/appservermatrix.jsp
  - …

Application Server I

- Erweiterbar durch anwendungsspezifische Komponenten (zB. EJB)
- Komponenten können remote „aufgerufen“ werden
  - Remote Object Method Call
- Wesentliche Eigenschaften
  - Skalierbarkeit (Benutzer, Transaktionen, Plattform)
  - Verfügbarkeit (Ausfallsicherheit, 7*24 h Betrieb)
  - Sicherheit (Authentification, Authorization)
  - Transaktionsorientiert (Mehrbenutzerbetrieb)
  - Konfiguration vs. Programmierung (Security, Transaktionen, ...)
  - ...

Application Server II

- Stateless
  - für den Client werden nur für kurze Zeit (pro Aufruf) Ressourcen allokiert
  - skalieren sehr gut
  - Beispiele: EJB, Web Server, CICS, CORBA
- Stateful
  - für den Client werden für längere Zeit (Session) Ressourcen allokiert
  - skalieren schlechter
  - Beispiele: EJB, CORBA

Application Server III

- EJB Server
  - BEA WebLogic, IBM WebSphere, Oracle App Server, ...
- Web Server
  - Erweiterbar durch CGI, Servlet, JSP, ASP, ...
- CORBA
  - VisiBroker, ComponentBroker, ...
- Transaktionsmonitor
  - CICS, Tuxedo, MTS, ...
- Datenbanksystem
  - DB2, Oracle, Gemstone, ...
- ...

Java 2 Enterprise Edition I

- J2EE Produkte bzw. APIs
  - Enterprise Java Beans (EJB)
  - Java Naming & Directory Interfaces (JNDI)
  - Java Database Connectivity (JDBC)
  - Java Connector Architecture (JCA)
  - Java Message Service (JMS)
  - Java Transaction API (JTA) und Service (JTS)
  - Servlets und Java Server Pages (JSP)
  - CORBA (RMI/IIOP, Java IDL, JTS/OTS)
  - Extensible Markup Language (XML)
  - Web Services (& UDDI)
  - ...

Java 2 Enterprise Edition II

- Enterprise Java Beans (EJB)
  - eine Spezifikation - kein Produkt
    - 1.0, 1.1 und 2.0
    - J2EE SDK enthält Referenzimplementierung (RI)
  - „keine“ Gemeinsamkeiten mit Java Beans
  - verwendet andere J2EE Technologien
    - JNDI, JDBC, JTA, RMI/IIOP, XML, ...
  - Implementierungen
    - kommerziell (IBM WebSphere, BEA WebLogic, ...) bzw. OSS (Enhydra, jBoss, ...)

Java 2 Enterprise Edition III

- Java Naming & Directory Service (JNDI)
  - Zugriff auf Naming und Directory Services
    - LDAP, COS Naming, Active Directory, ...
- Java Database Connectivity (JDBC)
  - Zugriff auf relationale Datenquellen
    - SQL92 bzw. SQL99
- Java Connector Architecture (JCA)
  - Zugriff auf Legacy Systeme
    - CICS, SAP, Baan, ...

Java 2 Enterprise Edition IV

- Java Messaging Service (JMS)
  - Zugriff auf Message Queues
    - MQ Series, ...
- Java Transaction API (JTA/JTS)
  - Programmierung von Client Demarcated Transactions
- Servlets und Java Server Pages (JSP)
  - Programmierung von Thin Clients
- ...

Enterprise Java Beans I

- Designziele
- EJB Environment
- Rollenkonzept
- Anwendungsentwicklung
- EJB Typen
  - Session Bean
  - Entity Bean
  - Message-Driven Bean

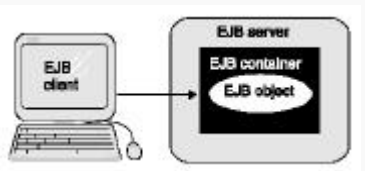
Enterprise Java Beans II

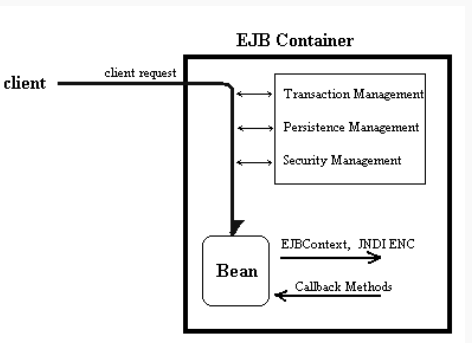
- Designziele
  - Komponentenarchitektur für verteilte Java Anwendungen
  - Hersteller- und Plattformunabhängig
  - Trennung zw. Anwendung und Infrastrukturdiensten (Persistenz, Transaktionen, Sicherheit, ...)
  - definiert Schnittstellen und Semantik im Hinblick auf
    - Server Provider bzw. Container Provider
    - Bean Provider
    - Client Anwendungen

Enterprise Java Beans III

- EJBs werden in einem speziellen Environment ausgeführt – dem EJB Container
  - Ohne EJB Container funktionieren EJBs nicht
    - EJBs interagieren mit dem EJB Container mittels Callback Methoden, dem EJBContext Interface oder JNDI (ENC)
  - Isoliert die EJB vom direkten Zugriff durch Client Anwendungen
    - Stellt Security, Persistenz, Transaktionen, Concurrency sicher
  - Stellt Pooling von Ressourcen bereit

Enterprise Java Beans IV





- EJB Container wird vom EJB Server verwaltet
  - EJB Server stellt Naming, Transaction, Database Connectivity, usw. bereit.

Enterprise Java Beans V

- Callback Methoden
  - EJB implementiert einen Subtype von EnterpriseBean (Session, Entity, Message-Driven Bean)
  - Subtypes definieren Methoden, die vom EJB Container zu bestimmten Zeitpunkten im Lifecycle der EJB aufgerufen werden
    - Aktivieren der EJB, Transaktionsende, ...
  - Ermöglichen der EJB den Eingriff zu bestimmten Zeitpunkten im Lebenszyklus

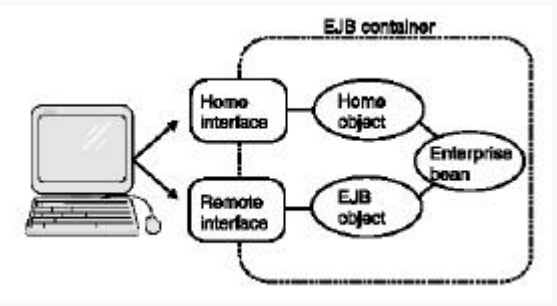
Enterprise Java Beans VI

- EJBContext
  - EJB erhält beim „Erzeugen“ eine Referenz auf ein EJBContext Objekt
    - Eigentlich eine direkte Referenz auf den EJB Container
  - EJBContext Interface stellt Methoden bereit für den Zugriff auf das Environment der EJB
    - Identität des Clients, Zustand einer Transaktion, ...
- Environment Naming Context (ENC) ist ein spezieller Naming Context

Enterprise Java Beans VII

- Rollenkonzept
  - Enterprise Bean Provider programmiert EJB Komponenten
  - Application Assembler stellt aus EJB Komponenten Anwendungen zusammen
  - Deployer „installiert“ EJB Anwendungen in einem EJB Server
  - EJB Server Provider stellt EJB Server her
  - EJB Container Provider stellt EJB Container her
  - System Administrator administriert EJB Server

Enterprise Java Beans VIII



- Client greift über Home bzw. Remote Interface zu
- Zugriff auf „eigentliche“ Enterprise Java Bean nur möglich über Home bzw. EJB Object
- EJB Container stellt Interceptor dar

Enterprise Java Beans IX

- Enterprise Java Bean besteht aus
  - Home Interface zum Erzeugen bzw. Auffinden (nur EntityBean) einer EnterpriseBean
  - Remote Interface definiert die Methoden der EnterpriseBean
  - EnterpriseBean implementiert die Methoden des Home/Remote Interface

Enterprise Java Beans X

- EJB Komponenten werden deployed
  - Deployment Descriptor wird erstellt
  - Implementierungen der Interfaces (Home bzw. Remote) werden generiert
- im Deployment Descriptor werden für die Komponente
  - Transaktions- bzw. Sicherheitseigenschaften
  - die verwendeten Ressourcen und
  - das Environment (Properties) der EJB festgelegt

Enterprise Java Beans XI

- EJB Client
  - findet das Home Interface über JNDI
  - „erzeugt“ mittels Home Interface das entsprechende Remote Interface
  - ruft mittels Remote Interface die Business Methoden der EJB auf
  - kann ein „normaler“ Java Client (Application, Applet) oder ein Servlet sein
  - abhängig vom EJB Server ist auch ein CORBA- bzw. MS-Client möglich

Enterprise Java Beans XII

- Session Bean
  - nicht persistent – existiert nur solange der Server „läuft“
  - Stateless vs. Stateful
    - für den Client kann von der Session Bean zwischen den Aufrufen Zustandsinformation gehalten werden
  - existiert nur für einen Client
  - muss nach einem Absturz des Servers neu erzeugt werden
  - entspricht einem Use Case
  - Container vs. Bean managed transaction

Enterprise Java Beans XIII

- Entity Bean
  - Persistent – existiert „quasi“ permanent
  - Meist in einer Datenbank gespeichert
  - existiert für „alle“ Clients
    - hat eine Identität (Primary Key)
    - überlebt den Absturz des Servers
  - entspricht einem Business Object (?)
  - Container vs. Bean managed persistence
    - das Laden und Speichern des Objekts kann vom Container übernommen werden

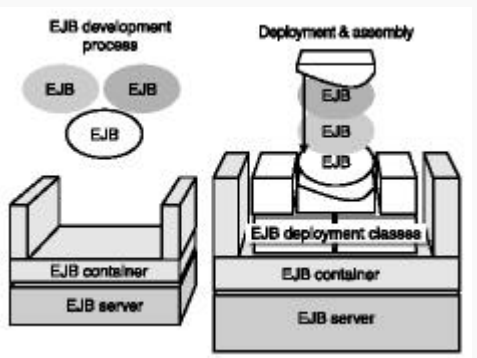
Enterprise Java Beans XIV

- Message-Driven Bean
  - Stateless, transaktional
  - zur Verarbeitung von JMS Messages
  - „besteht“ nur aus Bean Klasse und Deployment Descriptor
    - Unterschied zu Session- bzw. Entity Bean
  - Home und Remote Interface fehlen
    - kann über kein RMI Interface angesprochen werden
    - verarbeitet „nur“ asynchrone Messages

Development Process I

- Bean Provider implementiert Business-Logik
  - erstellt EJBs (Home-, Remote-Interface, Bean-Klasse), Deployment Descriptor für EJBs, JAR bestehend aus EJBs und DDs
- Container Provider erstellt EJB Container
- Server Provider erstellt EJB Server
- Deployer „installiert“ EJBs
  - generiert vom EJB Container benötigte „Helper“-Klassen
- Application Assembler implementiert EJB Client

Development Process II



Beispiel: Stateless Session Bean I

- Soll das aktuelle Datum und die Uhrzeit des Servers liefern
- Erstellt werden müssen folgende Interfaces bzw. Klassen
  - CurrentDateHome – Home Interface
  - CurrentDate – Remote Interface
  - CurrentDateBean – Session Bean
- Einfacher Test-Client

Beispiel: Stateless Session Bean II

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

Beispiel: Stateless Session Bean III

```
import java.util.*;
import javax.ejb.*;
import java.rmi.RemoteException;
public interface CurrentDate extends EJBObject {
  Date getCurrentDate() throws RemoteException;
}
```


Beispiel: Stateless Session Bean IV

```
import java.rmi.*;
import java.util.*;
import javax.ejb.*;
public class CurrentDateBean implements SessionBean {
  public CurrentTimeBean() {}
  public void setSessionContext(SessionContext context)
      throws RemoteException {}
  public void ejbActivate() throws RemoteException {}
  public void ejbCreate() {}
  public void ejbPassivate() throws RemoteException {}
  public void ejbRemove() throws RemoteException {}
  public Date getCurrentDate() {return new Date();}
}
```

Beispiel: Stateless Session Bean V

- Jede Methode des Home Interface wird mit einem ejb als Prefix implementiert
  - z.B. HomeInterface.create() - SessionBean.ejbCreate()
- Methoden des Remote Interface müssen den RMI Regeln entsprechen
  - throws RemoteException
  - Argumente und Returnwerte müssen serialisierbar oder EJBObject Instanzen sein
- Session Bean sollte bei Anwendungsfehler eine EJBException signalisieren
  - in EJB 1.0 eine java.rmi.RemoteException

Beispiel: Stateless Session Bean VI

- Deployment Descriptor muss erstellt werden
  - Transaktions- und Security-Eigenschaften
  - Umgebungsinformationen für die Bean
  - …
- Session Bean muss deployed werden
  - Während des Deployments werden Implementierungen der Interfaces generiert

Beispiel: Stateless Session Bean VII

- Deployment Descriptor am Beispiel CurrencyConverter

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE ...>

<ejb-jar>
  <display-name>CurrentDateEJB</display-name>
  <enterprise-beans> ... </enterprise-beans>
  <assembly-descriptor>
    <method-permission> ... </method-permission>
    <container-transaction> ... </container-transaction>
  </assembly-descriptor>
</ejb-jar>
```


Beispiel: Stateless Session Bean VIII

```
<enterprise-beans>
   <session>
     <display-name>CurrentDateBean</display-name>
     <ejb-name>CurrentDateBean</ejb-name>
     <home>examples.ejb.CurrentDateHome</home>
     <remote>examples.ejb.CurrentDate</remote>
     <ejb-class>examples.ejb.CurrentDateBean</ejb-class>
     <session-type>Stateless</session-type>
     <transaction-type>Container</transaction-type>
     <security-identity>
       <description></description>
       <use-caller-identity></use-caller-identity>
     </security-identity>
     <resource-ref></resource-ref>
   </session>
 </enterprise-beans>
```

Beispiel: Stateless Session Bean IX

```
<assembly-descriptor>
  <method-permission>
    <unchecked />
    ...
    <method> ... </method>
    ...
  </method-permission>
  <container-transaction>
    <method> ... </method>
    <trans-attribute>Supports</trans-attribute>
  </container-transaction>
</assembly-descriptor>
```


Beispiel: Stateless Session Bean X

```
<method-permission>
  <unchecked />
  ...
  <method>
    <ejb-name>CurrentDateBean</ejb-name>
    <method-intf>Remote</method-intf>
    <method-name>getCurrentDate</method-name>
    <method-params/>
  </method>
  ...
</method-permission>
```

Beispiel: Stateless Session Bean XI

```
<container-transaction>
  <method>
    <ejb-name>CurrentDateBean</ejb-name>
    <method-intf>Remote</method-intf>
    <method-name>getCurrentDate</method-name>
    <method-params/>
  </method>
  <trans-attribute>Supports</trans-attribute>
</container-transaction>
```

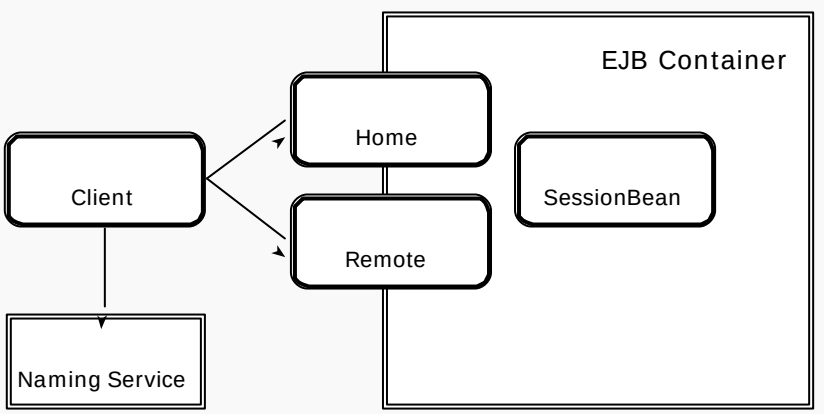

Beispiel: Stateless Session Bean XII

```
import java.io.*;
import javax.ejb.*;
import javax.naming.*;
import javax.rmi.PortableRemoteObject;
public class CurrentDateClient {
  public static void main(String[] args) {
    // lookup home
    // create bean
    // call business code
  }
}
```

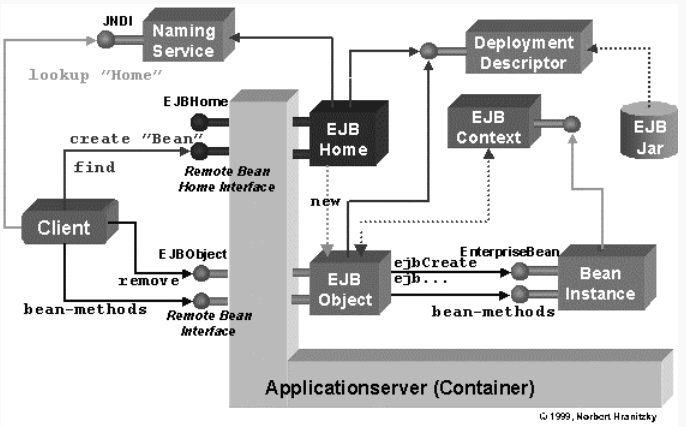
Beispiel: Stateless Session Bean XIII

```
try {
  InitialContext context = new InitialContext();
  Object ref = context.lookup("CurrentDate");
  CurrentDateHome home =
    (CurrentDateHome) PortableRemoteObject.narrow(
      home, CurrentDateHome.class);
  CurrentDate bean = home.create();
  System.out.print("Date = ");
  System.out.println(bean.getCurrentDate());
  bean.remove();
} catch (NamingException ex) {
  ex.printStackTrace();
} catch (CreateException ex) {
 ex.printStackTrace();
}
```


Beispiel: Stateless Session Bean XIV



Enterprise Java Beans XV



silbergrau Consulting & Software GmbH

Danke für die Aufmerksamkeit!

Silbergrau Consulting & Software GmbH
Dr. Andreas Erlach