J2EE Courses

Eine Einführung in die Java 2 Enterprise Edition

Teil 2 - EJBs - Remoting, Transaktionssupport, Security

Inhalt

- Was ist eigentlich ein EJB
- Session Beans - Stateless und Statefull
- Remoting-Fähigkeit
- Deklarativer Transaktionssupport
- Security in EJBs

Was ist eigentlich ein EJB

- serverseitige Komponente
- fest definierte Interfaces
- callback-Methoden
- Schnittstelle für Application-Server Dienste

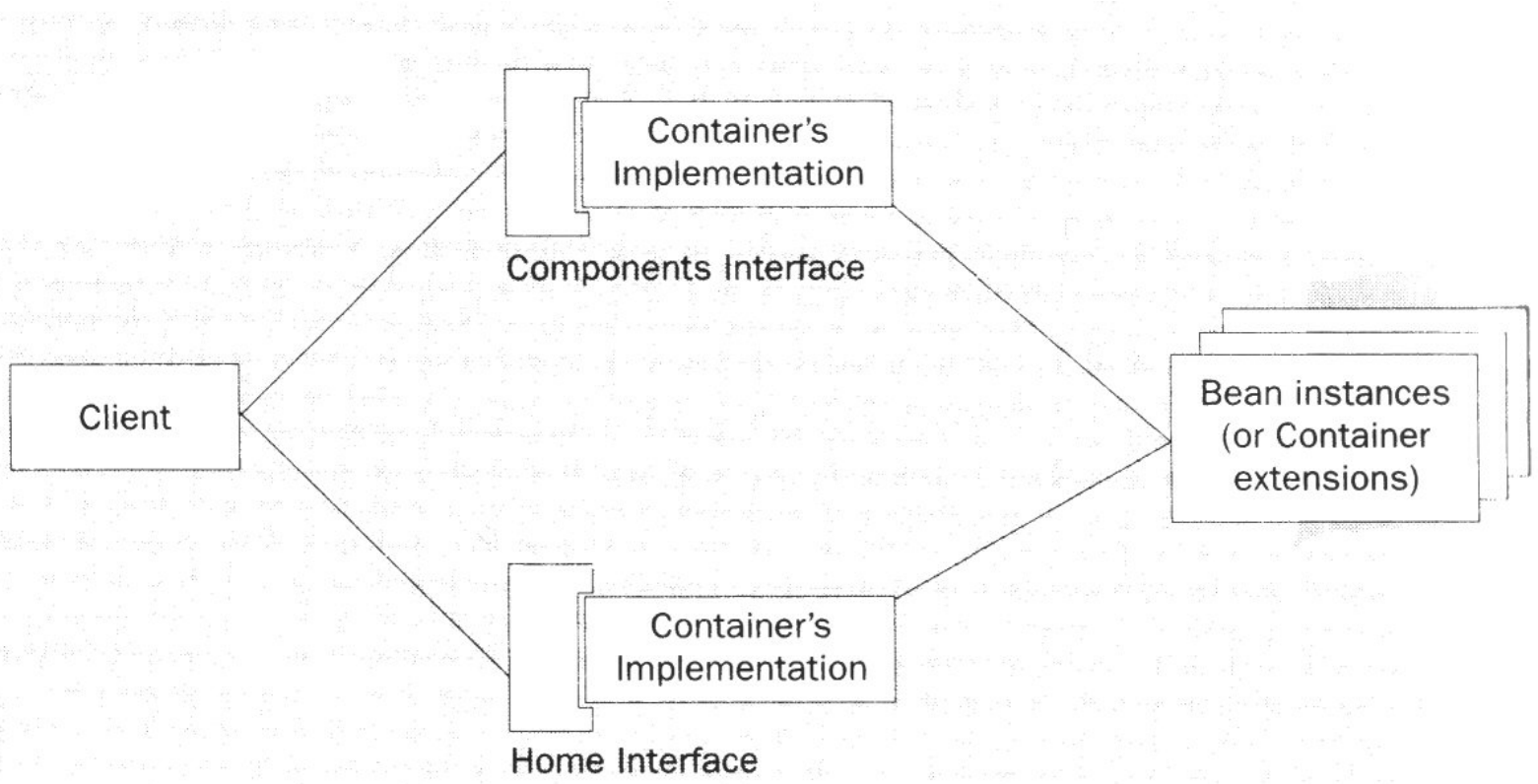
Typen von EJBs

- Session Beans
  - Stateless Session Beans
  - Statefull Session Beans
- Entity Beans
  - CMP Entity Beans
  - BMP Entity Beans
- Message Driven Beans

Woraus besteht ein EJB

- Einer EJB-Hauptklasse
  - Beinhaltet die tatsächliche Implementierung
- Home-Interfaces
  - Enthält Methoden zum Anlegen, Auffinden
- Component-Interfaces
  - Schnittstelle für den EJB-Client

Woraus besteht ein EJB



Stateless Session Beans

- Methoden mit Geschätslogik
- Nach der Methode gleicher Zustand

Unser erstes Stateless Session Bean

Das Component-Interface

```
public interface Account extends EJBObject {
    public double addMoney(double ammount, double oldValue)
        throws LimitExceededException, RemoteException;
}
```

Das EJBObject Interface

```
public interface EJBObject extends Remote {
    public EJBHome getEJBHome() throws RemoteException;

    public Handle getHandle() throws RemoteException;

    public Object getPrimaryKey() throws RemoteException;

    public boolean isIdentical(EJBObject ejb)
        throws RemoteException;

    public void remove()
        throws RemoteException, RemoveException;
}
```

Unser erstes Stateless Session Bean

Die EJB-Implementierung

```
public class AccountEJB implements javax.ejb.SessionBean {

    public void ejbCreate() {};
    public void ejbRemove() {};
    public void ejbActivate() {};
    public void ejbPassivate() {};
    public void setSessionContext(SessionContext ctx) {};

...
```

Unser erstes Stateless Session Bean

Die EJB-Implementierung

...

```
    private final double minimumAmmount = -5000;

    public double addMoney(double ammount, double oldValue)
        throws LimitExceededException
    {
        if (oldValue + ammount < minimumAmmount) {
            throw new LimitExceededException("pleite");
        }
        return oldValue + ammount;
    }
}
```

Unser erstes Stateless Session Bean

Das Home-Interface

```
public interface AccountHome extends javax.ejb.EJBHome {
    public Account create()
        throws RemoteException, CreateException;
}
```

Das EJBHome Interface

```
public interface EJBHome extends Remote {

    public EJBMetaData getEJBMetaData() throws RE;

    public HomeHandle getHomeHandle() throws RE;

    public void remove(Object obj)
        throws RE, RemoveException;

    public void remove(Handle handle)
        throws RE, RemoveException;

}
```

Der Deployment Descriptor

```
<ejb-jar>
    <description>Krasse Bank-Anwendung</description>
    <display-name>Banking-Application</display-name>
    <enterprise-beans>
        <session>
            <description>Unser erstes SLSB</description>
            <display-name>Bank Bohne</display-name>
            <ejb-name>FirstAccountBean</ejb-name>
            <home>ejbs.AccountHome</home>
            <remote>ejbs.Account</remote>
            <ejb-class>ejbs.AccountEJB</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
</enterprise-beans></ejb-jar>
```

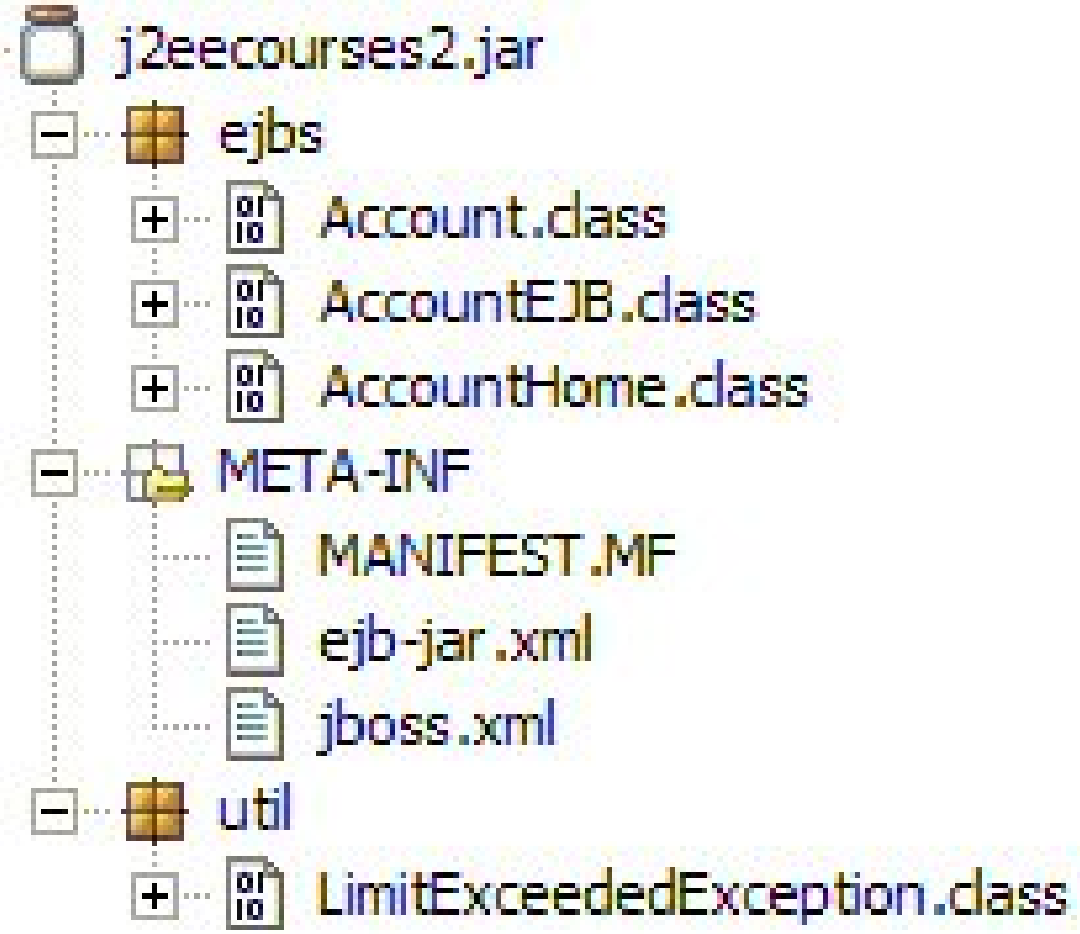
Jboss-spezifisches

```
<jboss>
   <unauthenticated-principal>nobody</unauthenticated-principal>

   <enterprise-beans>
      <session>
         <ejb-name>FirstAccountBean</ejb-name>
         <jndi-name>ejb/Account</jndi-name>
      </session>
   </enterprise-beans>

   <resource-managers></resource-managers>
</jboss>
```

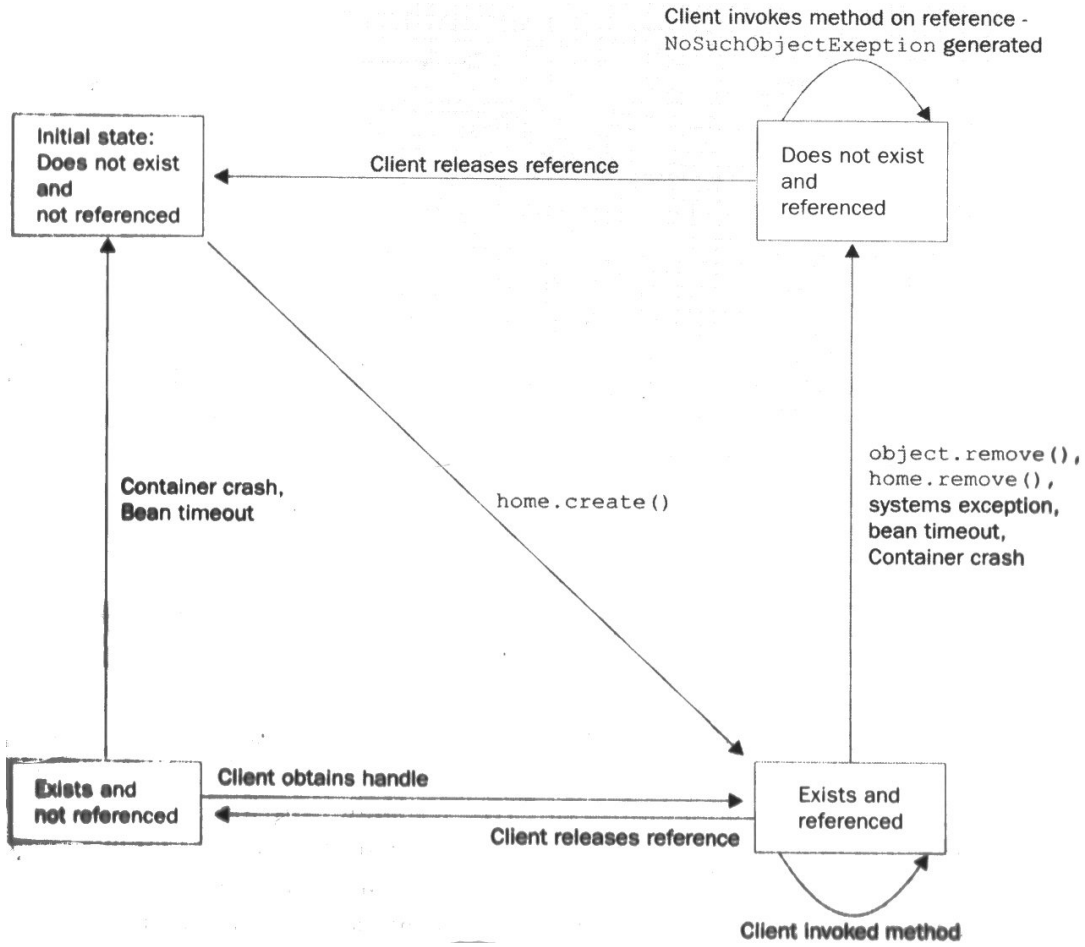
Verpackung fürs Deployment



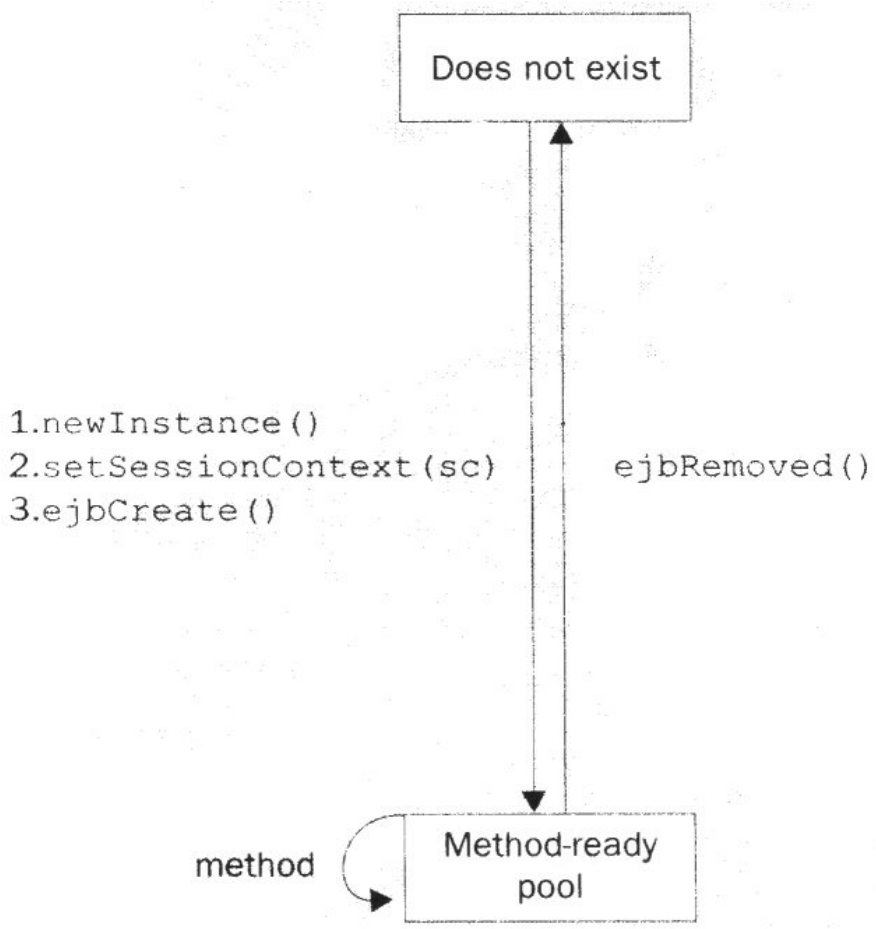
Aufrufen des EJB vom Client

```
private double calcNewSum(double oldValue, double toAdd) {
    try {
        InitialContext ctx = new InitialContext();
        Object homeRef = ctx.lookup("ejb/Account");
        AccountHome home =
            (AccountHome) PortableRemoteObject.narrow(
                homeRef, AccountHome.class);
        Account account = home.create();
        double newValue = account.addMoney(oldValue, toAdd);
        account.remove();
        return newValue;
    } catch (LimitExceededException e) {
    throw new RuntimeException(e); ...
}
```

Aus dem Leben eines SLSB



Zustände eines SLSB



Stateful Session Beans

- Methoden mit Geschätslogik
- Zustand im Bean speicherbar
- immer einem Client zugeordnet

Vom Stateless zum Stateful SB

Das Component-Interface

```
public interface StatefulAccount extends EJBObject{
    public void addMoney(double ammount)
        throws LimitExceededException, RemoteException;
    public double getAccountContent()
        throws RemoteException;
}
```

Vom Stateless zum Stateful SB

Die EJB-Implementierung

```
public class AccountEJB implements javax.ejb.SessionBean {

    public void ejbCreate() {};
    public void ejbRemove() {};
    public void ejbActivate() {};
    public void ejbPassivate() {};
    public void setSessionContext(SessionContext ctx) {};

...
```

Vom Stateless zum Stateful SB

Die EJB-Implementierung

...

```
    private final double minimumAmmount = -5000;

    private double accountContent;

    public double getAccountContent() {
        return accountContent;
    }
```

...

Vom Stateless zum Stateful SB

Die EJB-Implementierung

...

```
    public void addMoney(double ammount)
        throws LimitExceededException
    {
        if (accountContent + ammount < minimumAmmount) {
            throw new LimitExceededException("pleite");
        }
        accountContent += ammount;
    }
}
```

Der Deployment Descriptor

```
...
<enterprise-beans>
    <session>
        <description>Unser erstes SFSB</description>
        <display-name>Dauer Bank Bohne</display-name>
        <ejb-name>StatefulAccountBean</ejb-name>
        <home>ejbs.StatefulAccountHome</home>
        <remote>ejbs.StatefulAccount</remote>
        <ejb-class>ejbs.StatefulAccountEJB</ejb-class>
        <session-type>Stateful</session-type>
        <transaction-type>Container</transaction-type>
    </session>
</enterprise-beans>
</ejb-jar>
```

Jboss-spezifisches

```
<jboss>
   <unauthenticated-principal>nobody</unauthenticated-principal>
   <enterprise-beans>
      <session>
         <ejb-name>FirstAccountBean</ejb-name>
         <jndi-name>ejb/Account</jndi-name>
      </session>
      <session>
         <ejb-name>StatefulAccountBean</ejb-name>
         <jndi-name>ejb/StatefulAccount</jndi-name>
      </session>
   </enterprise-beans>
   <resource-managers></resource-managers>
</jboss>
```

Aufrufen des EJB vom Client

```
public class StatefulTestClient {

    private StatefulAccount account;

    public StatefulTestClient() {
        InitialContext ctx = new InitialContext();
        Object homeRef = ctx.lookup("ejb/StatefulAccount");
        StatefulAccountHome home =
            (StatefulAccountHome) PortableRemoteObject.narrow(
                homeRef,
                StatefulAccountHome.class);
        account = home.create();
    }
    ...
```

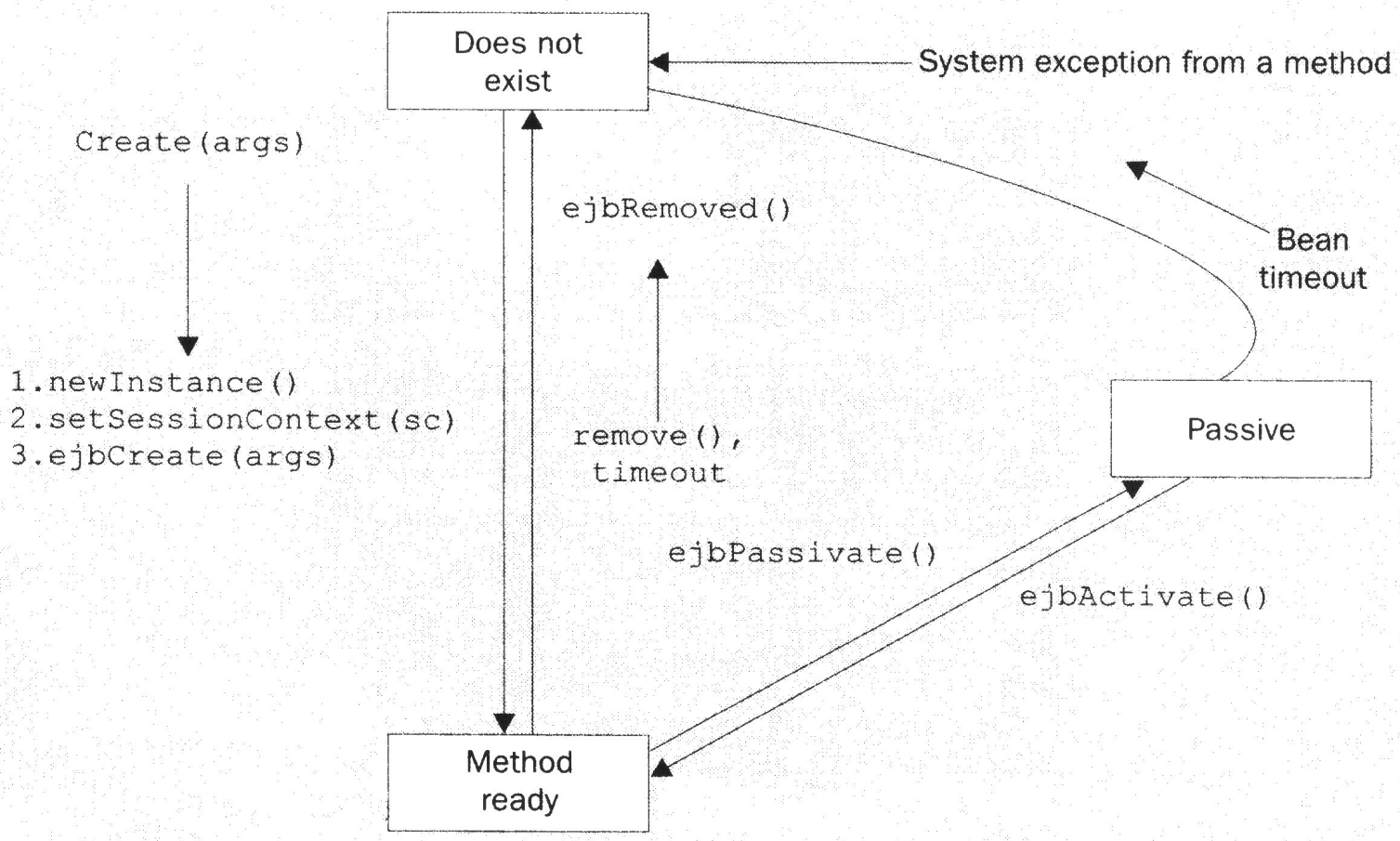
Aufrufen des EJB vom Client

```
...
    public void actionPerformed(ActionEvent e) {
            double v1 = Double.parseDouble(field1.getText());

            account.addMoney(v1);

            field2.setText("" + account.getAccountContent());
    }
```

Zustände eines SLSB



Benutzung von Handles

- Serialisierbare Verweise auf EJBs
- EJBs nicht serialisierbar
- Handle erhalten mit EJBObject.getHandle()
- EJB zurückbekommen mit Handle.getEJBObject()

Benutzung von Handle

Speichern des Handles

```
public void windowClosing(WindowEvent e) {
        ObjectOutputStream stream =
            new ObjectOutputStream(
                new FileOutputStream("handle.store"));
        stream.writeObject(account.getHandle());
        stream.flush();
        stream.close();

        System.exit(0);
}
```

Benutzung von Handle

Laden des Handles

```
if (account == null && new File("handle.store").exists()) {
    ObjectInputStream stream =
        new ObjectInputStream(
            new FileInputStream("handle.store"));
    Handle handle = (Handle) stream.readObject();
    account = (StatefulAccount) handle.getEJBObject();
    stream.close();
}
```

Kommunikation zwischen EJBs

- Kommunikation über das Remote-Interface
- Probleme bei Remote-Aufrufen
- Local Interfaces

EJB Kommunikation - Deployment Descriptor

```
...
<enterprise-beans>
    <session>
        <description>Unser erstes SFSB</description>
        ...
        <transaction-type>Container</transaction-type>
        <ejb-ref>
            <ejb-ref-name>ejb/Account</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>
            <home>ejbs.AccountHome</home>
            <remote>ejbs.Account</remote>
        </ejb-ref>
    </session>
</enterprise-beans>
</ejb-jar>
```

EJB Kommunikation - JBoss-spezifisch

```
<enterprise-beans>
   <session>
      <ejb-name>FirstAccountBean</ejb-name>
      <jndi-name>ejb/Account</jndi-name>
   </session>
   <session>
      <ejb-name>StatefulAccountBean</ejb-name>
      <jndi-name>ejb/StatefulAccount</jndi-name>
      <ejb-ref>
         <ejb-ref-name>ejb/Account</ejb-ref-name>
         <jndi-name>ejb/Account</jndi-name>
      </ejb-ref>
   </session>
</enterprise-beans>
```

EJB Kommunikation - Benutzung im Bean

```
public void addMoney(double ammount)
        throws LimitExceededException {
    if (accountContent + ammount < minimumAmmount) {
        throw new LimitExceededException("pleite");
    }
    InitialContext ctx = new InitialContext();
    AccountHome home =
        (AccountHome) PortableRemoteObject.narrow(
                ctx.lookup("java:comp/env/ejb/Account"),
                AccountHome.class);
    Account act = home.create();

    accountContent = act.addMoney(ammount, accountContent);
}
```

EJB Kommunikation - Probleme bei Remote-Interfaces

- Jeder Methodenaufruf über das Netz
- Serialisierungsaufwand der Parameter/Rückgabewerte
- Lösung: Local Interfaces
  - Zusätzliches Component- und Home-Interface
  - Nur Lokal im Container verfügbar
  - Keine Serialisierung → normaler Methodenaufruf

EJB Kommunikation - Local Interfaces

Zwei neue Interfaces ...

```
public interface AccountLocal extends EJBLocalObject {
    public double addMoney(double ammount, double oldValue)
        throws LimitExceededException;
}

public interface AccountLocalHome extends EJBLocalHome {
    public AccountLocal create()
        throws CreateException;
}
```

EJB Kommunikation - Local Interfaces

Deklaration im Deployment Descriptor

```
<session>
    <description>Unser erstes SLSB</description>
    <display-name>Bank Bohne</display-name>
    <ejb-name>FirstAccountBean</ejb-name>
    <home>ejbs.AccountHome</home>
    <remote>ejbs.Account</remote>
    <local-home>ejbs.AccountLocalHome</local-home>
    <local>ejbs.AccountLocal</local>
    <ejb-class>ejbs.AccountEJB</ejb-class>
    <session-type>Stateless</session-type>
    <transaction-type>Container</transaction-type>
</session>
```

EJB Kommunikation - Local Interfaces

Änderungen am Deployment Descriptor

```
...
<enterprise-beans>
    <session>
        <description>Unser erstes SFSB</description>
        ...
        <transaction-type>Container</transaction-type>
        <ejb-local-ref>
            <ejb-ref-name>ejb/Account</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>
            <local-home>ejbs.AccountLocalHome</local-home>
            <local>ejbs.AccountLocal</local>
        </ejb-local-ref>
    </session>
</enterprise-beans>
```

EJB Kommunikation - Local Interfaces

Änderungen am JBoss-Descriptor

```
<enterprise-beans>
   <session>
      <ejb-name>FirstAccountBean</ejb-name>
      <jndi-name>ejb/Account</jndi-name>
      <local-jndi-name>ejb/AccountLocal</local-jndi-name>
   </session>
   <session>
      <ejb-name>StatefulAccountBean</ejb-name>
      <jndi-name>ejb/StatefulAccount</jndi-name>
      <ejb-local-ref>
         <ejb-ref-name>ejb/Account</ejb-ref-name>
         <local-jndi-name>ejb/AccountLocal</local-jndi-name>
      </ejb-local-ref>
   </session>
```

EJB Kommunikation - Local Interfaces

Geändertes Lookup

```
InitialContext ctx = new InitialContext();
AccountLocalHome home =
    (AccountLocalHome) PortableRemoteObject.narrow(
            ctx.lookup("java:comp/env/ejb/Account"),
            AccountLocalHome.class);
AccountLocal act = home.create();

accountContent = act.addMoney(ammount, accountContent);
```

Transaktionssupport

Probleme:

- Transaktionsdemarkation
  - Wo Anfang, wo Ende
- Transaktionspropagation
- Transaktionsverwaltung
  - Verteilte Transaktionen

Transaktionssupport

J2EE Lösungsansatz:

- Zusammenfassung der transaktionsfähigen Resourcen
- Transaktionen auf Methodenebene
- Deklarative Transaktionen

Transaktionssupport

```
public interface Transact extends EJBObject {
    public void doDBStuff() throws RemoteException;
}

public interface TransactHome extends EJBHome {
    public Transact create()
        throws RemoteException, CreateException;
}
```

Transaktionssupport

```
public void doDBStuff() {
        InitialContext context = new InitialContext();
        DataSource ds =
            (DataSource) PortableRemoteObject.narrow(
                context.lookup("java:comp/env/ejb/Account"),
                DataSource.class);
        Connection conn = ds.getConnection();
        Statement statement = conn.createStatement();
        statement.executeUpdate(
            "UPDATE konto SET summe = summe + 42");
        statement.executeUpdate(
            "UPDATE bargeld SET summe = summe - 42");
        statement.close(); conn.close();
}
```

Transaktionssupport

Deployment Descriptor

```
...
    <session>
        <description>EJB mit Transaktion</description>
        <display-name>Transact Bohne</display-name>
        <ejb-name>TransactBean</ejb-name>
        <home>ejbs.TransactHome</home>
        <remote>ejbs.Transact</remote>
        <ejb-class>ejbs.TransactEJB</ejb-class>
        <session-type>Stateless</session-type>
        <transaction-type>Container</transaction-type>
    </session>
</enterprise-beans>
```

Transaktionssupport

Deployment Descriptor

```
    ...
    </enterprise-beans>
    <assembly-descriptor>
        <container-transaction>
            <method>
                <ejb-name>TransactBean</ejb-name>
                <method-name>*</method-name>
            </method>
            <trans-attribute>Required</trans-attribute>
        </container-transaction>
    </assembly-descriptor>
</ejb-jar>
```

Transaktionssupport

Verfügbare Transaction-Attributes

- Required
- RequiresNew
- Mandatory
- NotSupported
- Supports
- Never

Transaktionssupport

Auswirkung von Exceptions auf Transaktionen

- Application Exceptions
  - Transaktion wird committed
  - Exception wird zum Client weitergegeben
  - Manuelles rollback der Transaktion: SessionContext.setRollbackOnly()
- Runtime Exceptions
  - Transaktion wird zurückgerollt
  - Exception als EJBException oder andere passende Runtime-Exception zum Client

Fragen?