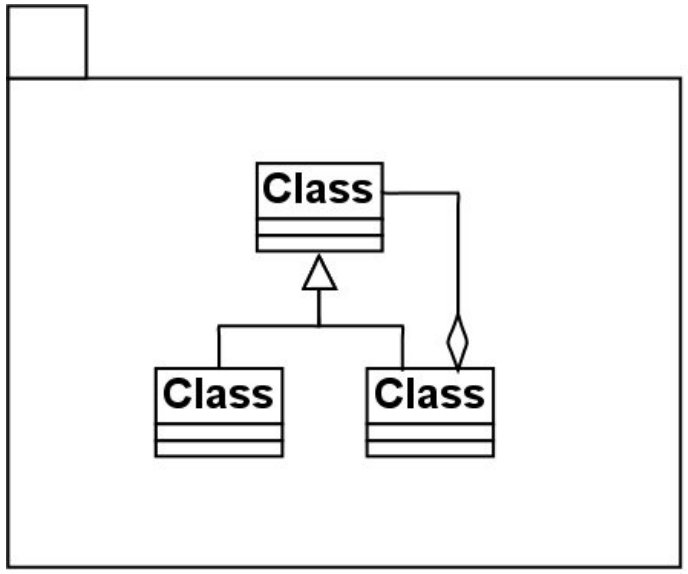
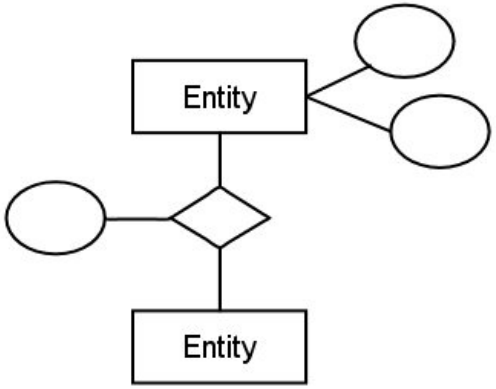
J2EE Courses

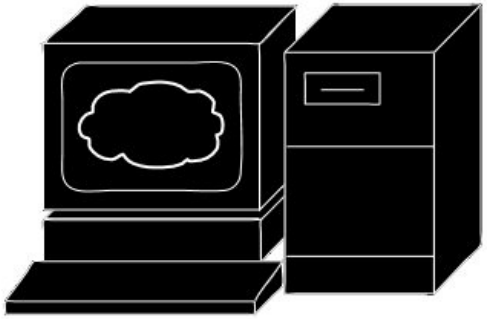
Eine Einführung in die Java 2 Enterprise Edition

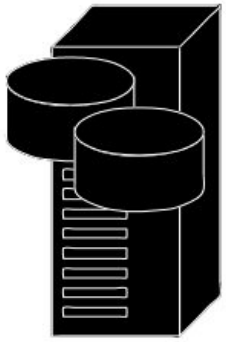
Teil 3 - EJBs - Persistenz und EJB Design Patterns

O/R Mapping

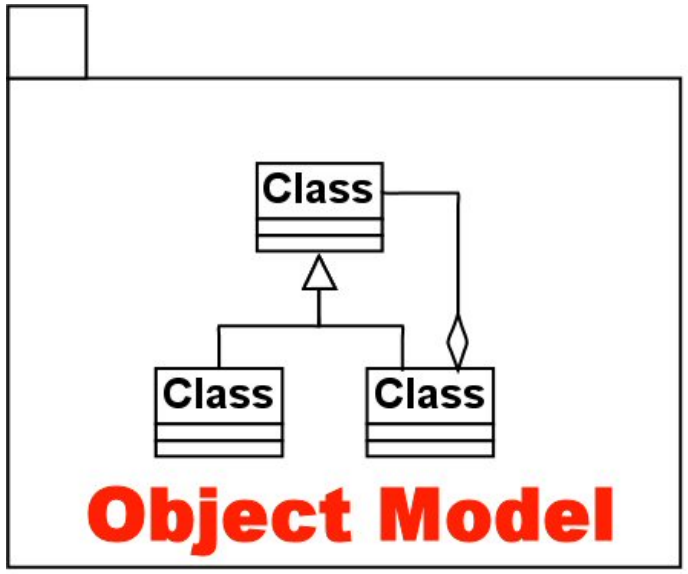


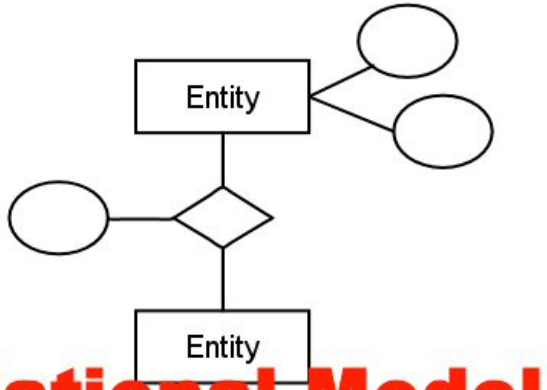


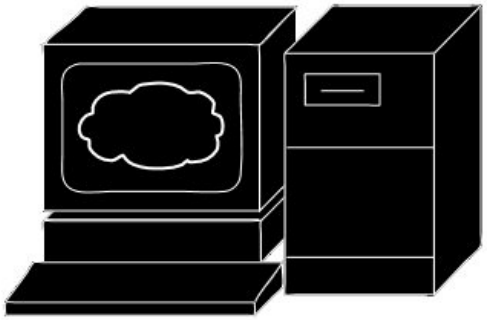


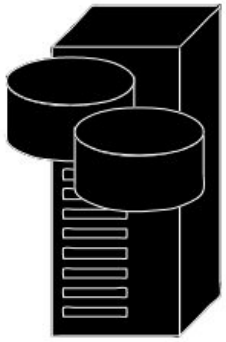


O/R Mapping

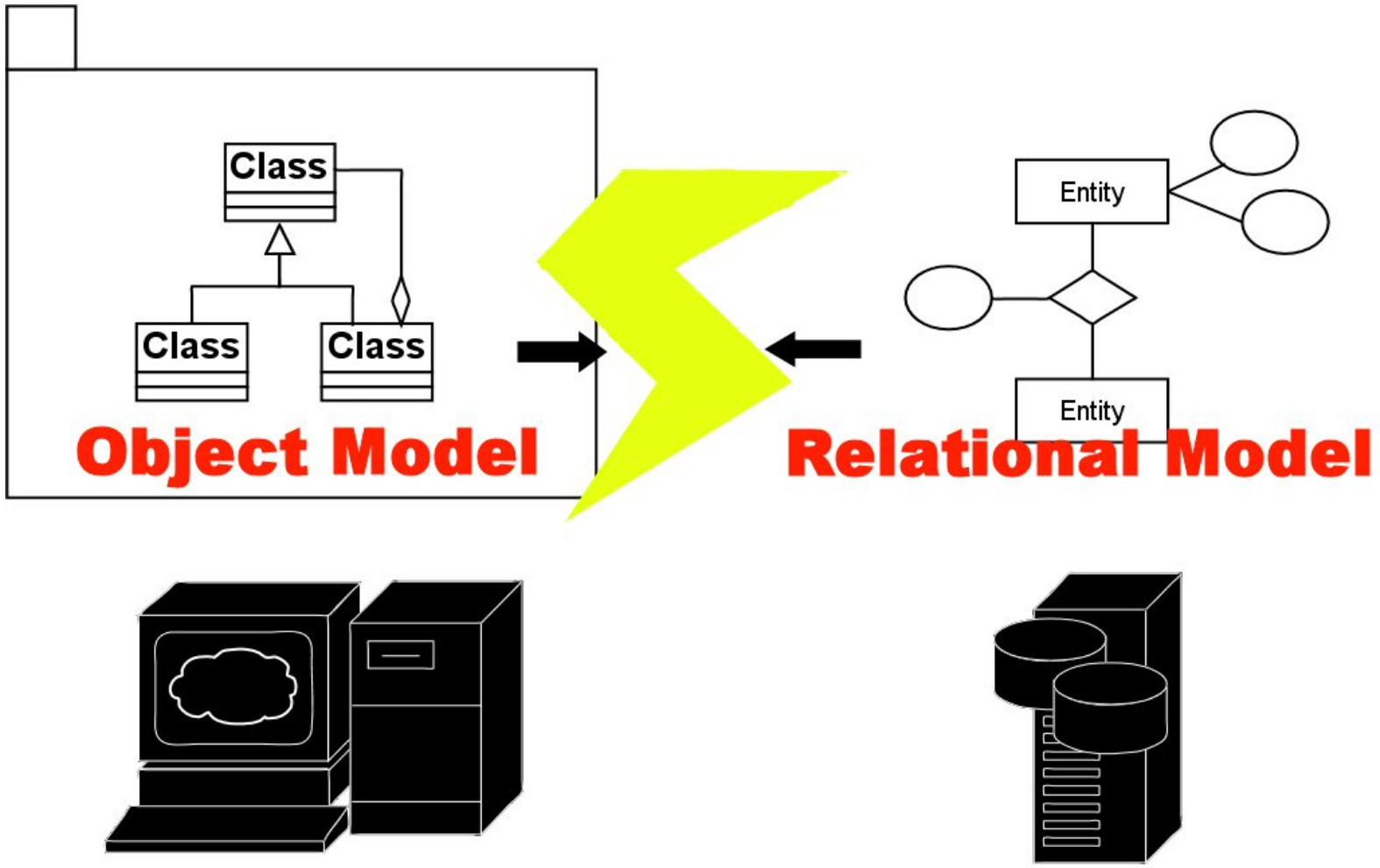

Class
Class
Class
Object Model


Entity
Entity
Relational Model





O/R Mapping


Class
Class
Class
Object Model
Entity
Entity
Relational Model

Inhalt

- Persistenz - der SEP Ansatz
- Entity Beans
  - Container Managed Persistence
  - Persistenz - SEP reworked
  - Finder und EJBQL
  - Bean Managed Persistence
  - Objektbeziehungen bei CMP-Entity Beans
- EJB Design Patterns

Persistenz - der SEP Ansatz

Direkt aus dem SEP

```
public List getTasks(String username) {
    List tasks = new LinkedList();
    Connection con = getConnection();
    PreparedStatement statement = con.prepareStatement(
        "SELECT * FROM task WHERE owner = ?");
    statement.setString(1, username);
    ResultSet rs = statement.executeQuery();
    while (rs.next()) {
        TaskVO task = new TaskVO();
        task.setTitle(rs.getString("title"));
        task.setDescription(rs.getString("description));
        tasks.add(task);
    }
    return tasks;
```

Direkt aus dem SEP

```
public int newTask(TaskVO task, String ownername) {
    Connection con = getConnection();
    int pk = parentDataSource.generatePK("task");
    PreparedStatement statement = con.prepareStatement(
        "INSERT INTO task (?, ?, ?, ?)");
    statement.setInt(1, pk);
    statement.setString(2, ownername);
    statement.setString(3, task.getTitle());
    statement.setString(4, task.getDescription());
    statement.executeUpdate();

    return pk;
}
```

Entity Beans

Auftritt der Entity Beans

- Ein Entity Bean repräsentiert persistentes Objekt
- CMP → Container kümmert sich um das Speichern
- BMP → Entwickler programmiert Speichern, Container entscheidet wann
- Container verwaltet Caching, Pooling, Locking ...

Ein erstes CMP Entity Bean

Das Component Interface

```
public interface Task extends EJBObject {
    public Long getId() throws RemoteException;
    public String getTitle() throws RemoteException;
    public String getDescription() throws RemoteException;
    public String getOwnerName() throws RemoteException;

    public void setId(Long id) throws RemoteException;
    public void setTitle(String title) throws RemoteException;
    public void setDescription(String description)
        throws RemoteException;
    public void setOwnerName(String ownerName)
        throws RemoteException; }
```

Ein erstes CMP Entity Bean

Das Home Interface

```
public interface TaskHome extends EJBHome {
    public Task create(
        Long id, String title,
        String description, String ownerName)
        throws RemoteException, CreateException;
    public Task findByPrimaryKey(Long taskId)
        throws RemoteException, FinderException;
}
```

Ein erstes CMP Entity Bean

Die Implementierungsklasse

```
public abstract class TaskBean extends EntityBean {
    public abstract Long getId();
    public abstract String getTitle();
    public abstract String getDescription();
    public abstract String getOwnerName();

    public abstract void setId(Long id);
    public abstract void setTitle(String title);
    public abstract void setDescription(String description);
    public abstract void setOwnerName(String ownerName);

    ...
```

Ein erstes CMP Entity Bean

Die Implementierungsklasse

```
...
    public void ejbActivate() {}
    public void ejbPassivate() {}
    public void ejbLoad() {}
    public void ejbStore() {}
    public void ejbRemove() {}
    public void setEntityContext(EntityContext ctx) {}
    public void unsetEntityContext() {}
...
```

Ein erstes CMP Entity Bean

Die Implementierungsklasse

```
...
    public Long ejbCreate(
        Long id, String title,
        String description, String ownerName) {
            this.setId(id);
            this.setTitle(title);
            this.setDescription(description);
            this.setOwnerName(ownerName);
            return id;
    }
    public void ejbPostCreate(Long id, String title,
        String description, String ownerName) {  }
```

Ein erstes CMP Entity Bean

Der Deployment Descriptor

```
<ejb-jar><enterprise-beans><entity>
        <ejb-name>TaskEJB</ejb-name>
        <home>j2eecourses.TaskHome</home>
        <remote>j2eecourses.Task</remote>
        <ejb-class>j2eecourses.TaskBean</ejb-class>
        <persistence-type>Container</persistence-type>
        <prim-key-class>java.lang.Long</prim-key-class>
        <reentrant>True</reentrant>
        <cmp-version>2.x</cmp-version>
        <abstract-schema-name>TaskEJB</abstract-schema-name>
        <cmp-field><field-name>id</field-name></cmp-field>
        <cmp-field><field-name>title</field-name></cmp-field>
        <cmp-field><field-name>description</field-name></cmp-field>
        <cmp-field><field-name>ownerName</field-name></cmp-field>
        <primkey-field>id</primkey-field>
</entity></enterprise-beans></ejb-jar>
```

Jboss-spezifisches

jboss.xml

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

   <enterprise-beans>
      <entity>
         <ejb-name>TaskEJB</ejb-name>
         <jndi-name>ejb/Task</jndi-name>
      </entity>
   </enterprise-beans>

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

Jboss-spezifisches

jboss-cmp.xml

```
<jbosscmp-jdbc>
   <defaults>
     <datasource>java:/NewDS</datasource>
     <datasource-mapping>mySQL</datasource-mapping>
     <preferred-relation-mapping>
        relation-table
     </preferred-relation-mapping>
   </defaults>
...
```

Jboss-spezifisches

jboss-cmp.xml

```
...
   <enterprise-beans>
      <entity>
         <ejb-name>TaskEJB</ejb-name>
         <table-name>task</table-name>
         <cmp-field><field-name>id</field-name>
            <column-name>task_id</column-name></cmp-field>
         <cmp-field><field-name>title</field-name>
            <column-name>title</column-name></cmp-field>
         <cmp-field><field-name>description</field-name>
            <column-name>description</column-name></cmp-field>
         <cmp-field><field-name>ownerName</field-name>
            <column-name>ownername</column-name></cmp-field>
      </entity>
   </enterprise-beans>
</jbosscmp-jdbc>
```

Persistenz - SEP reworked

SEP reworked

```
public class TestClient extends JFrame {
    private static Task task = null;
    ...

    c.addActionListener(new ActionListener() {

        public void actionPerformed(ActionEvent e) {
            if (task == null) {
                task = newTask(a.getText(), b.getText());
            }
        }
    });
```

SEP reworked

```
...
private static Task newTask(
        String title, String description) {
    InitialContext ctx = new InitialContext();
    Object homeRef = ctx.lookup("ejb/Task");
    TaskHome home =
        (TaskHome) PortableRemoteObject.narrow(
            homeRef,
            TaskHome.class);
    Task task = home.create(
        new Long(1), title, description, "horst");
    return task;
}
```

SEP reworked

```
public class TestClient extends JFrame {
    private static Task task = null;
    ...

    c.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (task == null) {
                task = newTask(a.getText(), b.getText());
            } else {
                task.setDescription(b.getText());
                task.setTitle(a.getText());
            }
        }
    });
```

Vorteile von Entity Beans

- Objektorientierte Repräsentation
- Entwurf rein OO - kein relationaler Entwurf
- Kein SQL-Code muss geschrieben werden
- Datenbankunabhängigkeit
- Caching und Pooling

Finder und EJBQL

Finder Methods

Das Home Interface

```
public interface TaskHome extends EJBHome {
    public Task create(
        Long id, String title,
        String description, String ownerName)
        throws RemoteException, CreateException;

    public Task findByPrimaryKey(Long taskId)
        throws RemoteException, FinderException;
}
```

Eigene Finder Methods

Das Home Interface

```
public interface TaskHome extends EJBHome {
    public Task create(
        Long id, String title,
        String description, String ownerName)
        throws RemoteException, CreateException;
    public Task findByPrimaryKey(Long taskId)
        throws RemoteException, FinderException;
    public Collection findByOwnerName(String ownerName)
        throws RemoteException, FinderException;
}
```

Eigene Finder Methods

Änderungen am Deployment Descriptor

...

```
        <primkey-field>id</primkey-field>
        <query>
            <query-method>
               <method-name>findByOwnerName</method-name>
               <method-params>
                    <method-param>java.lang.String</method-param>
               </method-params>
            </query-method>
            <result-type-mapping>Remote</result-type-mapping>
            <ejb-ql><![CDATA[
                SELECT OBJECT(o) FROM TaskEJB AS o WHERE o.ownerName=?1
            ]]></ejb-ql>
        </query>
</entity></enterprise-beans></ejb-jar>
```

Eigene Finder Methods

Aufruf vom Client

```
e.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Collection c = findTasks(d.getText());
        f.setListData(c.toArray());
    }
});
```

Eigene Finder Methods

Aufruf vom Client

```
private static Collection findTasks(String ownerName) {
    InitialContext ctx = new InitialContext();
    Object homeRef = ctx.lookup("ejb/Task");
    TaskHome home =
        (TaskHome) PortableRemoteObject.narrow(
            homeRef, TaskHome.class);
    return home.findByOwnerName(ownerName);
}
```

Elemente von EJBQL

- SQL ähnlicher Syntax
- Boolsche Operatoren (=, <, >, <> ...)
- LIKE
- BETWEEN
- Funktionen (CONCAT, LENGTH, SQRT ...)

Beispiel:

```
SELECT Object(o) FROM Task o
    WHERE   o.ownerName = ?1
    AND     LENGTH(o.description) < 50
    AND     o.daysInFuture BETWEEN 5 AND 10
```

Bean Managed Persistence

Ein BMP Entity Bean

Das Component Interface

```
public interface Record extends EJBObject {
    public Long getId();
    public String getEntry();

    public void setId(Long id);
    public void setEntry(String entry);
}
```

Ein BMP Entity Bean

Das Home Interface

```
public interface RecordHome extends EJBHome {
    public Record create(
        Long id, String entry) throws RemoteException;

    public Record findByPrimaryKey(Long taskId)
        throws RemoteException, FinderException;

    public Collection findWithEntryGreater(int length)
        throws RemoteException, FinderException;
}
```

Ein BMP Entity Bean

Die Implementierungsklasse

```
public class RecordBean extends EntityBean {
    private Long id;
    private String entry;

    public Long getId() {return id};
    public String getEntry() {return entry};

    public void setId(Long id) {this.id = id};
    public void setEntry(String entry) {this.entry = entry};
    ...
```

Ein BMP Entity Bean

Die Implementierungsklasse

```
...
    public Long ejbCreate(
        Long id, String entry) {
            ...
            ps = connection.prepareStatement(
                "INSERT INTO records VALUES (?,?)");
            ps.setLong(1, id);
            ps.setString(2, entry);
            ps.executeUpdate();
            ...
    }
...
```

Ein BMP Entity Bean

Die Implementierungsklasse

```
...
    public void ejbLoad() {
        ...
        ps = conn.prepareStatement(
            "SELECT id, entry FROM records WHERE id = ?");
        ps.setLong(1, id);
        result = ps.executeQuery();
        if (result.next()) {
            this.id = result.getLong("id");
            this.entry = result.getString("entry");
        } ... }
...
```

Ein BMP Entity Bean

Die Implementierungsklasse

```
...
    public void ejbStore() {
        ps = conn.prepareStatement(
            "UPDATE records SET entry = ? WHERE id = ?");
        ps.setString(1, this.entry);
        ps.setLong(2, this.id);
        ps.executeUpdate();
    }
...
```

Ein BMP Entity Bean

Die Implementierungsklasse

```
...
    public void ejbRemove() {
        ps = conn.prepareStatement(
            "DELETE FROM records WHERE id = ?");
        ps.setLong(1, this.id);
        ps.executeUpdate();
    }
...
```

Ein BMP Entity Bean

Die Implementierungsklasse

```
...
    public void findWithEntryGreater(int value) {
        ps = conn.prepareStatement(
            "SELECT id FROM records WHERE LENGTH(entry) > ?");
        ps.setInt(1, value);
        ResultSet result = ps.executeQuery();
        ArrayList list = new ArrayList();
        while (result.next()) {
            list.add(result.getLong("id"));
        }
        return list;
    } ...
```

Ein BMP Entity Bean

Die Implementierungsklasse

...

```
    public void ejbPostCreate(Long id, String title,
        String description, String ownerName) {  }
    public void ejbActivate() {}
    public void ejbPassivate() {}
    public void setEntityContext(EntityContext ctx) {}
    public void unsetEntityContext() {}
```

Vor- und Nachteile

Vorteile:

- Sehr flexible Lade/Abfrage/Such-Methoden

Nachteile:

- Unportabel
- n+1 Problem !

Fazit: BMP nur wenn CMP zu unflexibel ist

Objektbeziehungen bei CMP

Ein zusätzliches CMP Entity Bean

Das Component Interface

```
public interface User extends EJBObject {
    public String getUsername();
    public void setUsername(String username);
}
```

Änderungen am Task EJB

Das Component Interface - Vorher

```
public interface Task extends EJBObject {
    public Long getId() throws RemoteException;
    public String getTitle() throws RemoteException;
    public String getDescription() throws RemoteException;
    public String getOwnerName() throws RemoteException;

    public void setId(Long id) throws RemoteException;
    public void setTitle(String title) throws RemoteException;
    public void setDescription(String description)
        throws RemoteException;
    public void setOwnerName(String ownerName)
        throws RemoteException; }
```

Änderungen am Task EJB

Das Component Interface - Nachher

```
public interface Task extends EJBObject {
    public Long getId() throws RemoteException;
    public String getTitle() throws RemoteException;
    public String getDescription() throws RemoteException;
    public User getOwner() throws RemoteException;

    public void setId(Long id) throws RemoteException;
    public void setTitle(String title) throws RemoteException;
    public void setDescription(String description)
        throws RemoteException;
    public void setOwner(User owner)
        throws RemoteException; }
```

Beschreibung im Deployment Descriptor

Der Deployment Descriptor - Vorher

```
<ejb-jar><enterprise-beans><entity>
        <ejb-name>TaskEJB</ejb-name>
        <home>j2eecourses.TaskHome</home>
        <remote>j2eecourses.Task</remote>
        <ejb-class>j2eecourses.TaskBean</ejb-class>
        <persistence-type>Container</persistence-type>
        <prim-key-class>java.lang.Long</prim-key-class>
        <reentrant>True</reentrant>
        <cmp-version>2.x</cmp-version>
        <abstract-schema-name>TaskEJB</abstract-schema-name>
        <cmp-field><field-name>id</field-name></cmp-field>
        <cmp-field><field-name>title</field-name></cmp-field>
        <cmp-field><field-name>description</field-name></cmp-field>
        <cmp-field><field-name>ownerName</field-name></cmp-field>
        <primkey-field>id</primkey-field>
</entity></enterprise-beans></ejb-jar>
```

Beschreibung im Deployment Descriptor

Der Deployment Descriptor

```
<ejb-jar>
    <enterprise-beans>
        <entity>
            <ejb-name>TaskEJB</ejb-name>
            ...
            <cmp-field><field-name>description</field-name></cmp-field>
            <primkey-field>id</primkey-field>
        </entity>
    </enterprise-beans>
    <relationships>
        ...
</ejb-jar>
```

Beschreibung im Deployment Descriptor

Der Deployment Descriptor

```
<relationships>
    <ejb-relation>
        <ejb-relation-name>Task-Owner</ejb-relation-name>
        <ejb-relationship-role>
            <ejb-relationship-role-name>
                Task-has-an-Owner
            </ejb-relationship-role-name>
            <multiplicity>Many</multiplicity>
            <relationship-role-source>
                <ejb-name>TaskEJB</ejb-name>
            </relationship-role-source>
            <cmr-field>
                <cmr-field-name>owner</cmr-field-name>
            </cmr-field>
        </ejb-relationship-role>
...
```

Beschreibung im Deployment Descriptor

Der Deployment Descriptor

...

```
            <ejb-relationship-role>
                <ejb-relationship-role-name>
                    User-has-many-Tasks
                </ejb-relationship-role-name>
                <multiplicity>One</multiplicity>
                <relationship-role-source>
                    <ejb-name>UserEJB</ejb-name>
                </relationship-role-source>
            </ejb-relationship-role>
        </ejb-relation>
    </relationships>
</ejb-jar>
```

Objektbeziehungen in der EJBQL

Änderungen des alten Finders

Das Home Interface

```
public interface TaskHome extends EJBHome {
    public Task create(
        Long id, String title,
        String description, String ownerName)
        throws RemoteException, CreateException;
    public Task findByPrimaryKey(Long taskId)
        throws RemoteException, FinderException;
    public Collection findByOwner(User owner)
        throws RemoteException, FinderException;
}
```

Änderung des alten Finders

Änderungen am Deployment Descriptor

...

```
        <primkey-field>id</primkey-field>
        <query>
            <query-method>
               <method-name>findByOwnerName</method-name>
               <method-params>
                    <method-param>j2eecourses.User</method-param>
               </method-params>
            </query-method>
            <result-type-mapping>Remote</result-type-mapping>
            <ejb-ql><![CDATA[
                SELECT OBJECT(o) FROM TaskEJB AS o WHERE o.owner=?1
            ]]></ejb-ql>
        </query>
</entity></enterprise-beans></ejb-jar>
```

EJBQL Funktionen für Relationen

Navigation durch die Relationen:

```
SELECT Object(s) FROM Ship s WHERE s.reederei.besitzer.name = "Horst";
```

EJBQL Funktionen für Relationen

Navigation durch die Relationen:

```
SELECT Object(s) FROM Ship s WHERE s.reederei.besitzer.name = "Horst";
```

Auswahl von Collection-Inhalten:

```
SELECT Object(c) FROM Customer c, IN( c.adresses ) a WHERE a.street LIKE "%weg";
```

EJBQL Funktionen für Relationen

Navigation durch die Relationen:

```
SELECT Object(s) FROM Ship s WHERE s.reederei.besitzer.name = "Horst";
```

Auswahl von Collection-Inhalten:

```
SELECT Object(c) FROM Customer c, IN( c.adresses ) a WHERE a.street LIKE "%weg";
```

Test ob ein Objekt in einer Collection enthalten ist:

```
SELECT Object(crs) FROM Cruise crs, IN (crs.reservations) res, Customer cust
WHERE
    cust = ?1
AND
    cust NOT MEMBER OF res.customers
```

Probleme bei Entity Beans

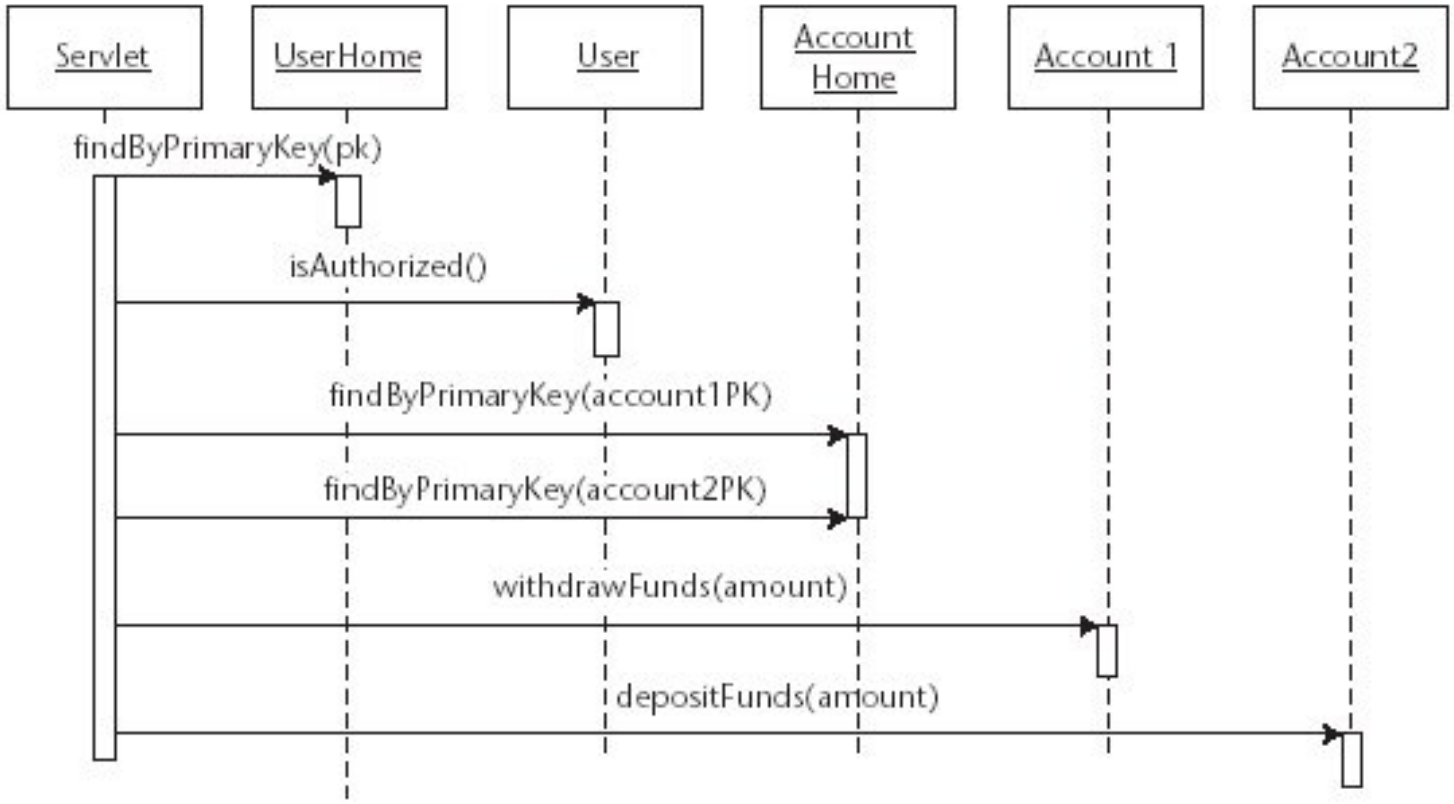
Probleme bei Entity Beans

- BMP kaum Benutzbar wegen n+1 Problem
- Problematischer Remoting-Supprt
  - Jeder get/set Aufruf ein Netzwerkaufruf
  - Schlechte Performance
- Keine Vererbung bei CMP Entity Beans
- Einschränkungen der EJBQL
  - Kein ORDER BY
  - Fehlende Date-Unterstützung
  - Fehlende Funktionen (SUM(), MAX(), MIN())
- Verstreuung der Persistenzschicht im Client
- Vermengung von Geschäftslogik und Persistenz

EJB Patterns

Session Facade

Session Facade

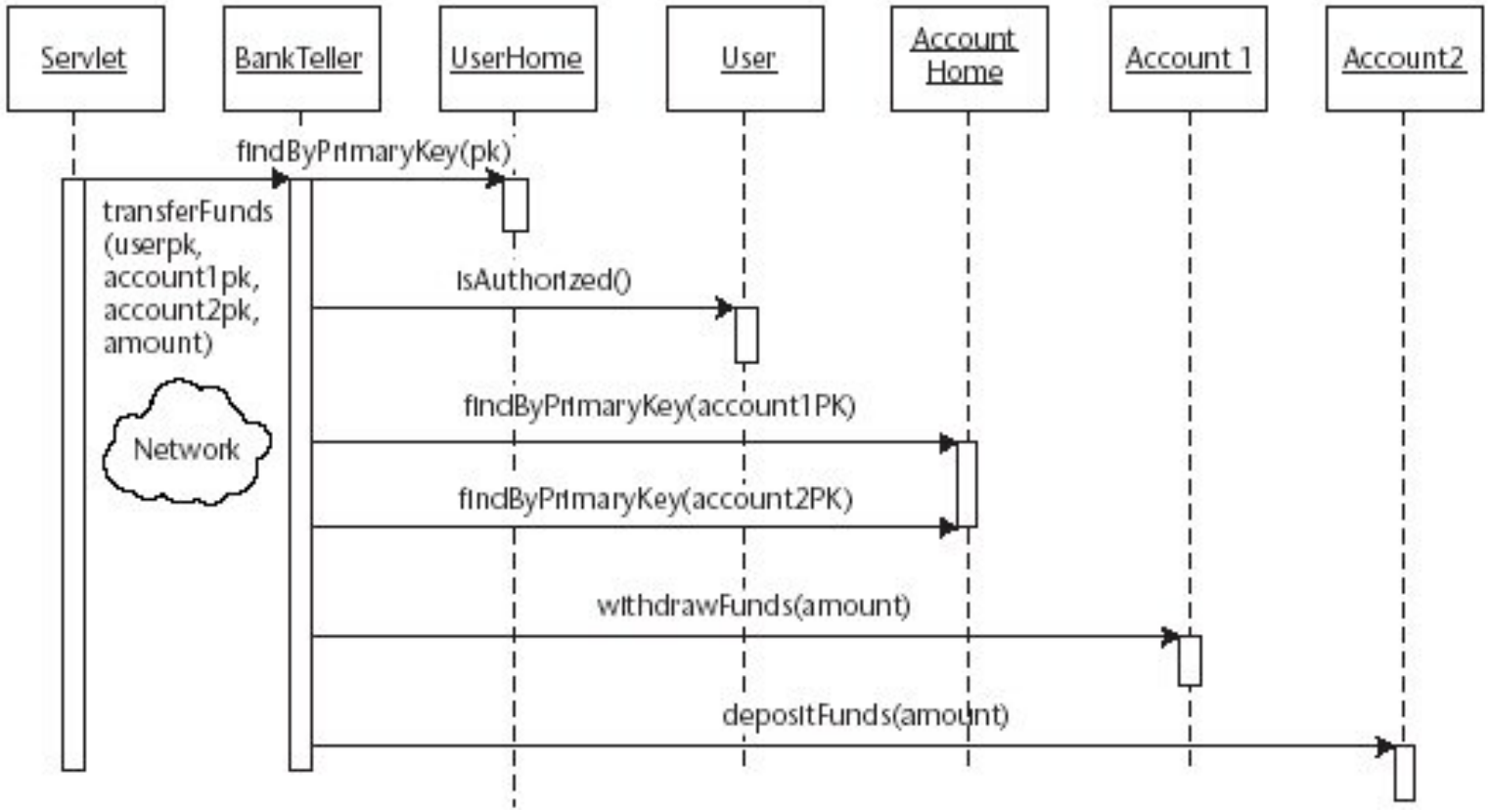


Session Facade

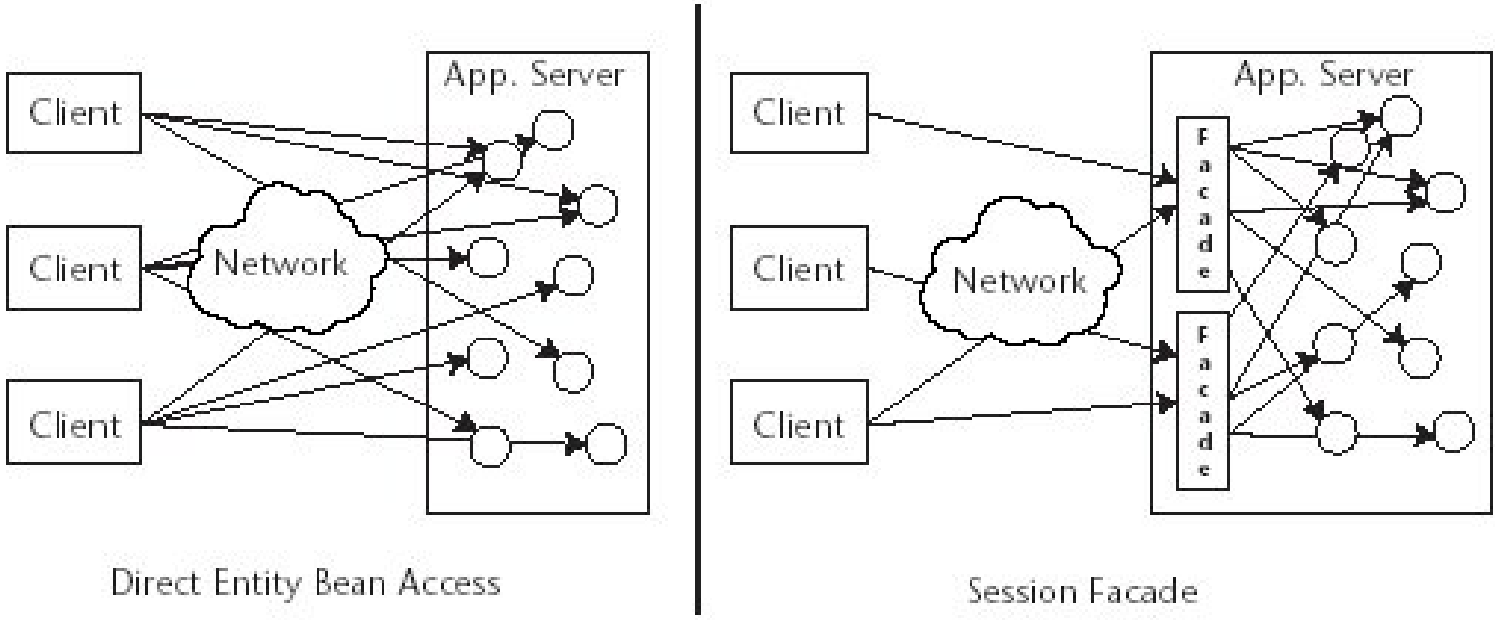
Ziele:

- Klare Schnittstelle zur Geschäftslogik
- Reduzierung des Netzwerkverkehrs
- Weniger Kopplung
- Bessere Wartbarkeit

Session Facade

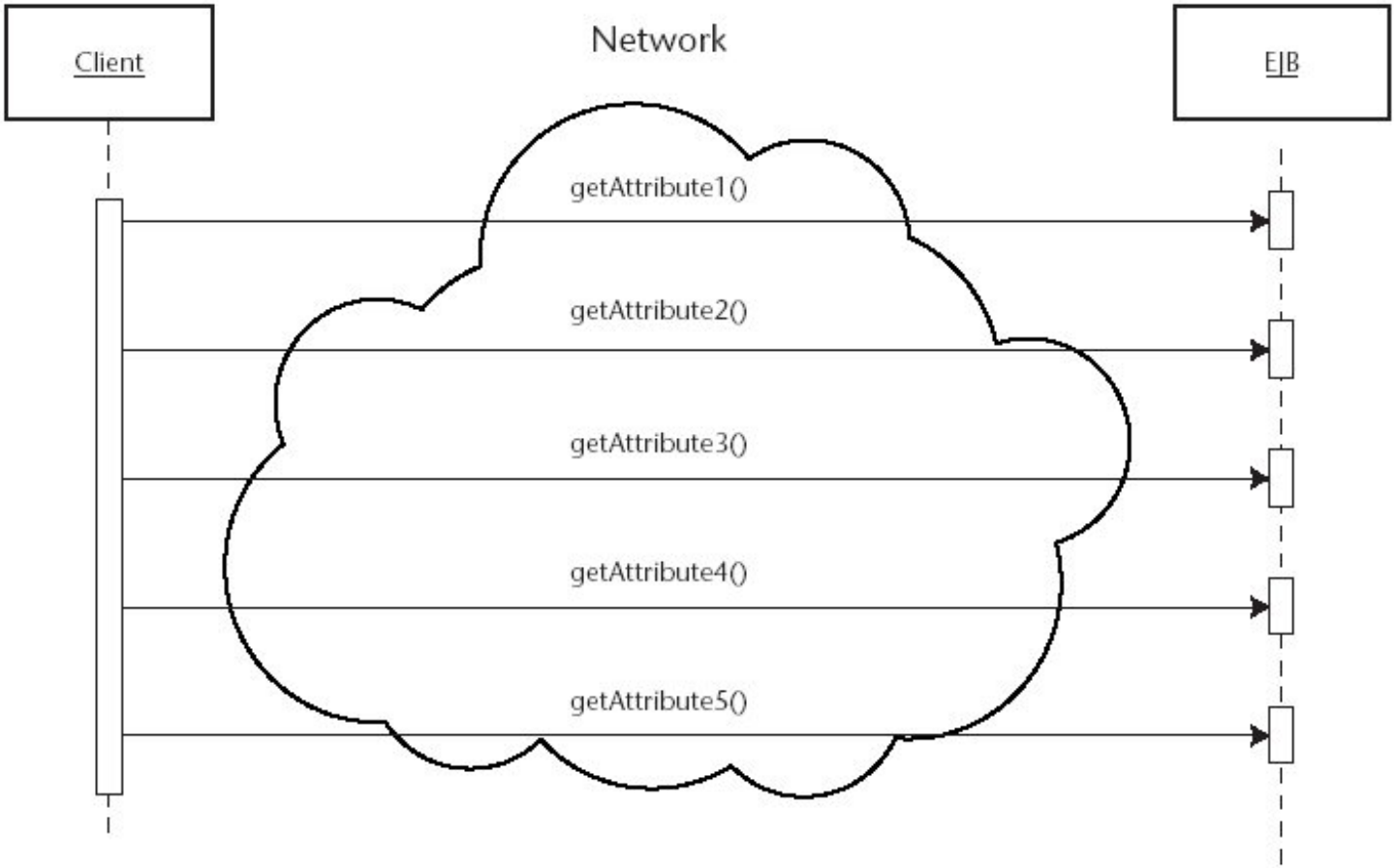


Session Facade



Data Transfer Objects

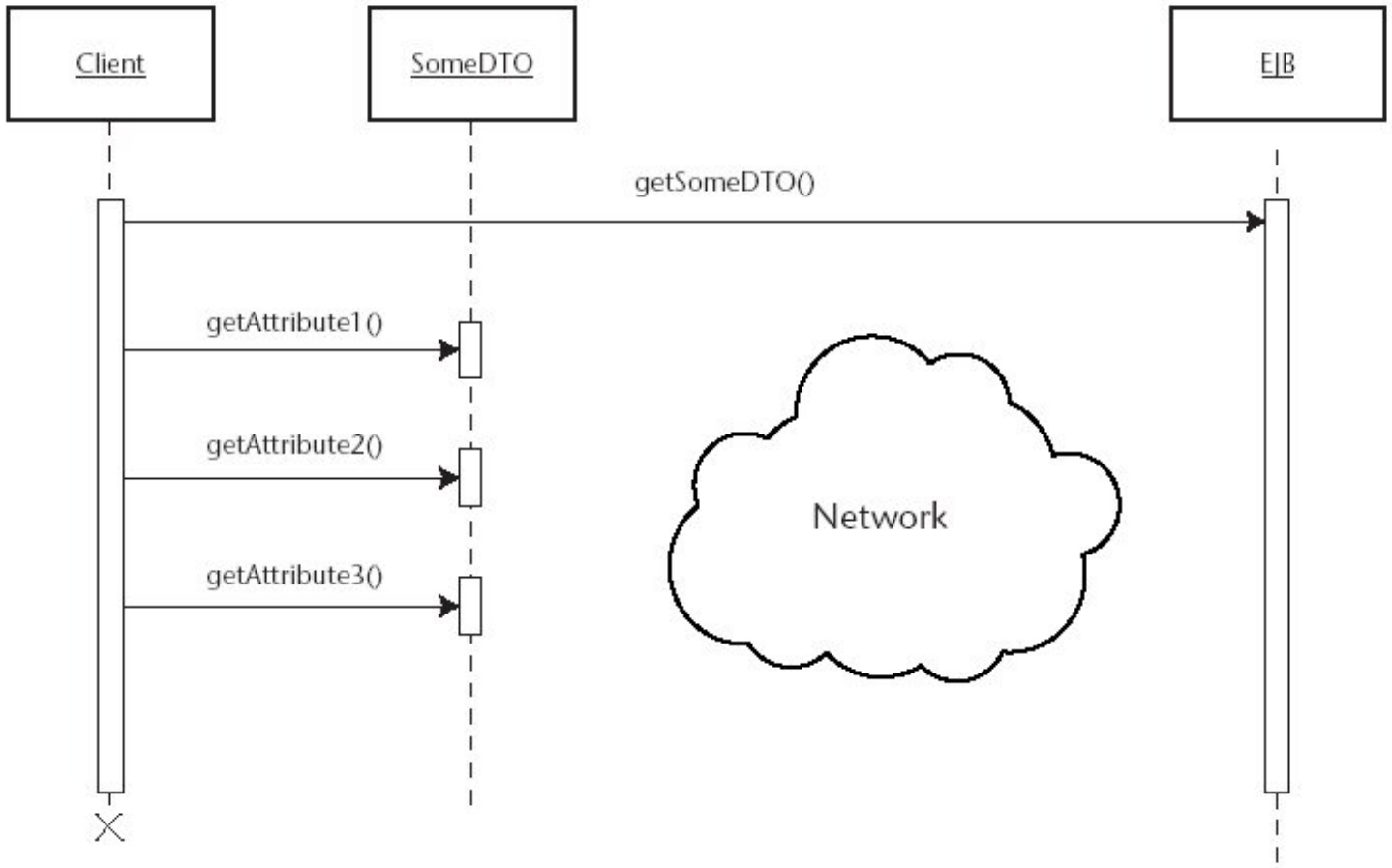
Data Transfer Objects



Data Transfer Objects

```
public class TaskDTO {
    public Long id;
    public String title;
    public String description;
    public String ownerName;
}
```

Data Transfer Objects



Noch Fragen?

Ende :)