Enterprise Java Beans

Überblick

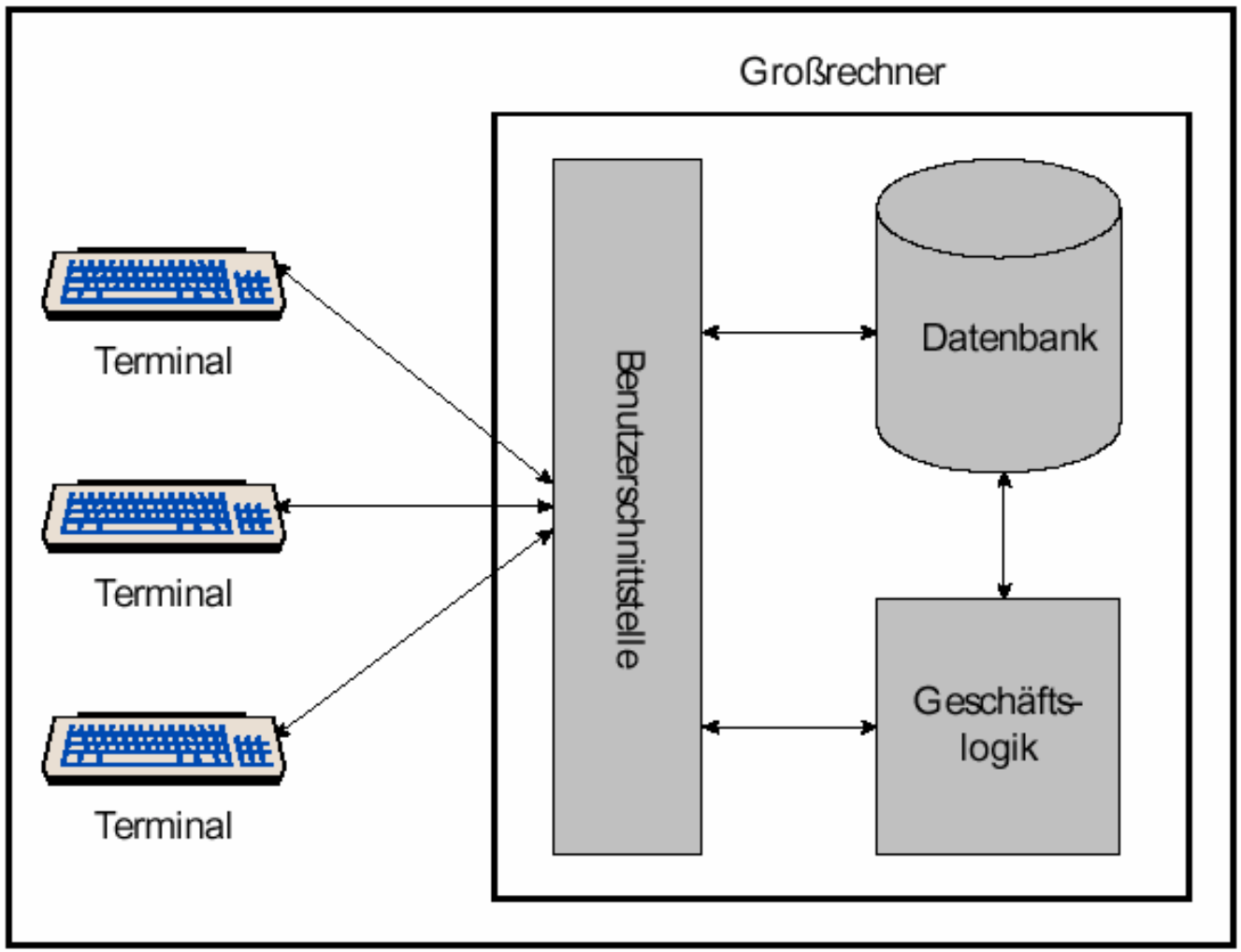
- Architektur
- EJB in der Praxis
- Dienste und Probleme / Nachteile

Motivation für Applikation Server

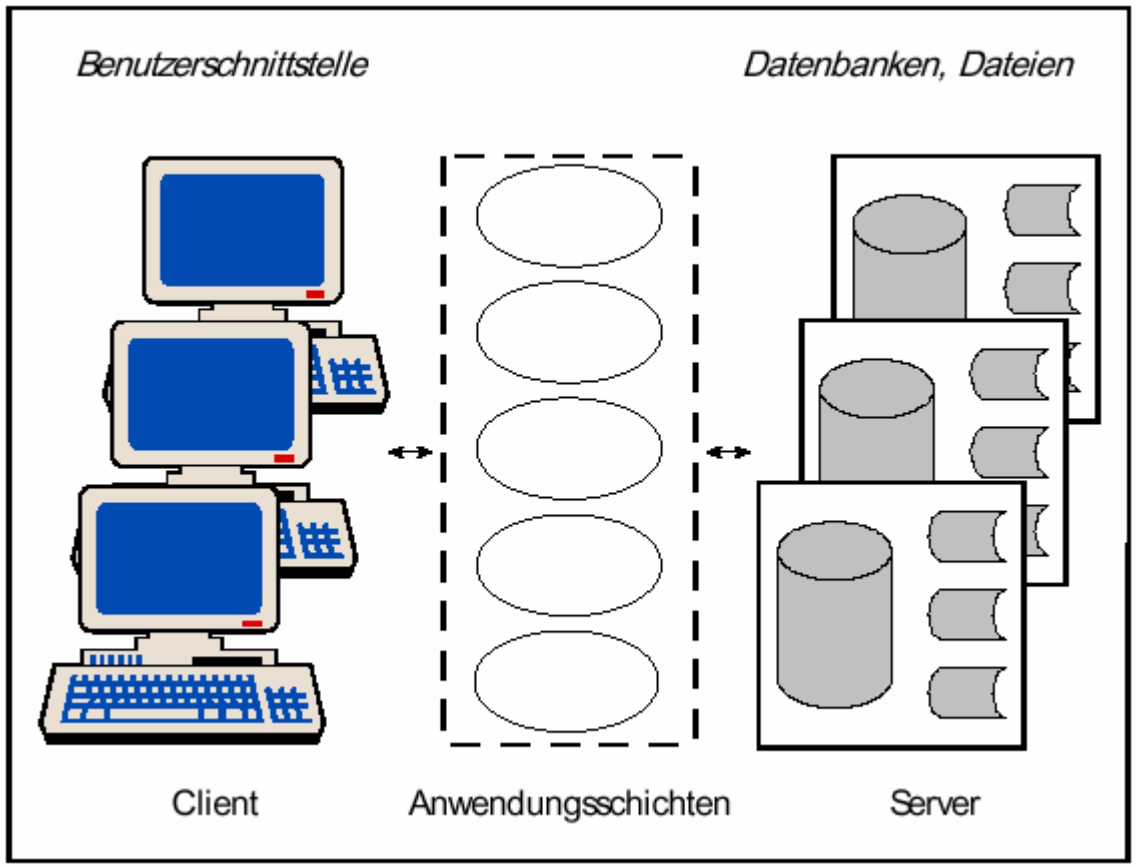
„Die Auseinandersetzung wird sich in diesem Jahrzehnt bei Middleware abspielen, nicht Betriebssystemen“

John Swainson,
General Manager in der IBM Software Group

Mainframe Architektur



3-Schicht Architektur

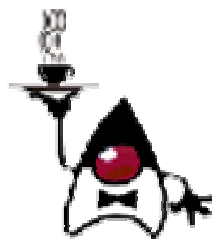


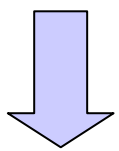
Anforderungen an Application Server

- Erweiterbarkeit
- Skalierbarkeit
- flexibel hinsichtlich Lastenverteilung
- Hochverfügbarkeit

Java-basierte Application Server

- Anforderungen:
  - standardisierte Schnittstellen
  - transparente Services
  - Transaktionssicherheit
  - Komponentenbasiert





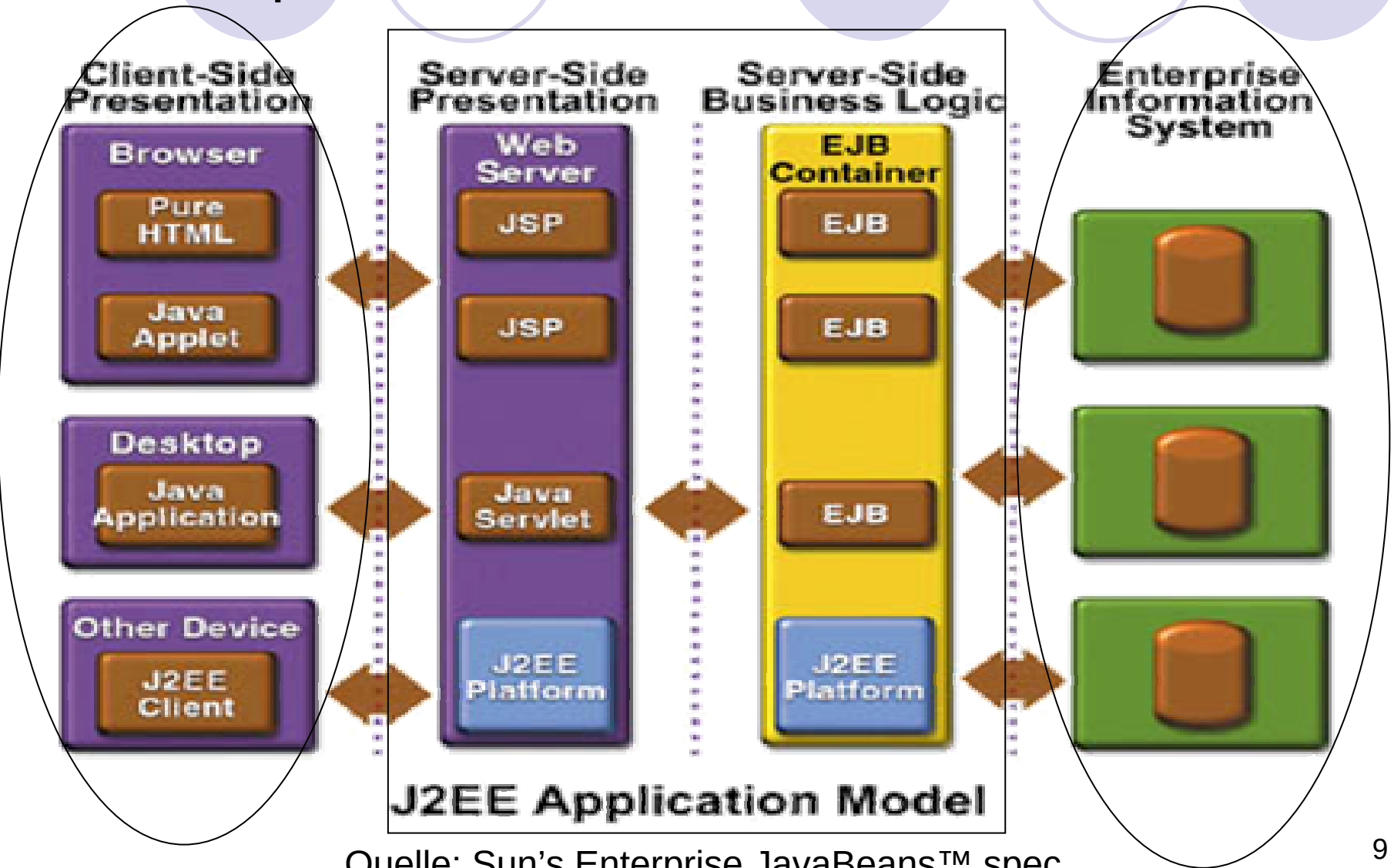
Enterprise Architektur

Was ist die Enterprise Architektur?

- Architektur für eine Serverseitige Plattform, die verteilte Komponenten zur Verfügung stellt
  - “a standard for multi-tier, server-oriented, component development”.
- Dienste: “lifecycle”, Transaktionen, Sicherheit, Verbindung zum Client, Datenbankzugriff, “Pooling”,…
- Entwickler soll sich auf die Business-Logik konzentrieren können

Quelle: Sun’s Enterprise JavaBeans™ spec

Enterprise Architekture



Quelle: Sun's Enterprise JavaBeans™ spec

Was sind ”Enterprise JavaBeans” (EJB)?

- Serverseitig wiederverwendbare Java Komponenten
- Enterprise JavaBeans werden vom J2EE Server ausgeführt
- Server bietet Dienste, die von Programmierer schwer implementiert werden können
  - einmal schreiben, in jedem Server laufen lassen

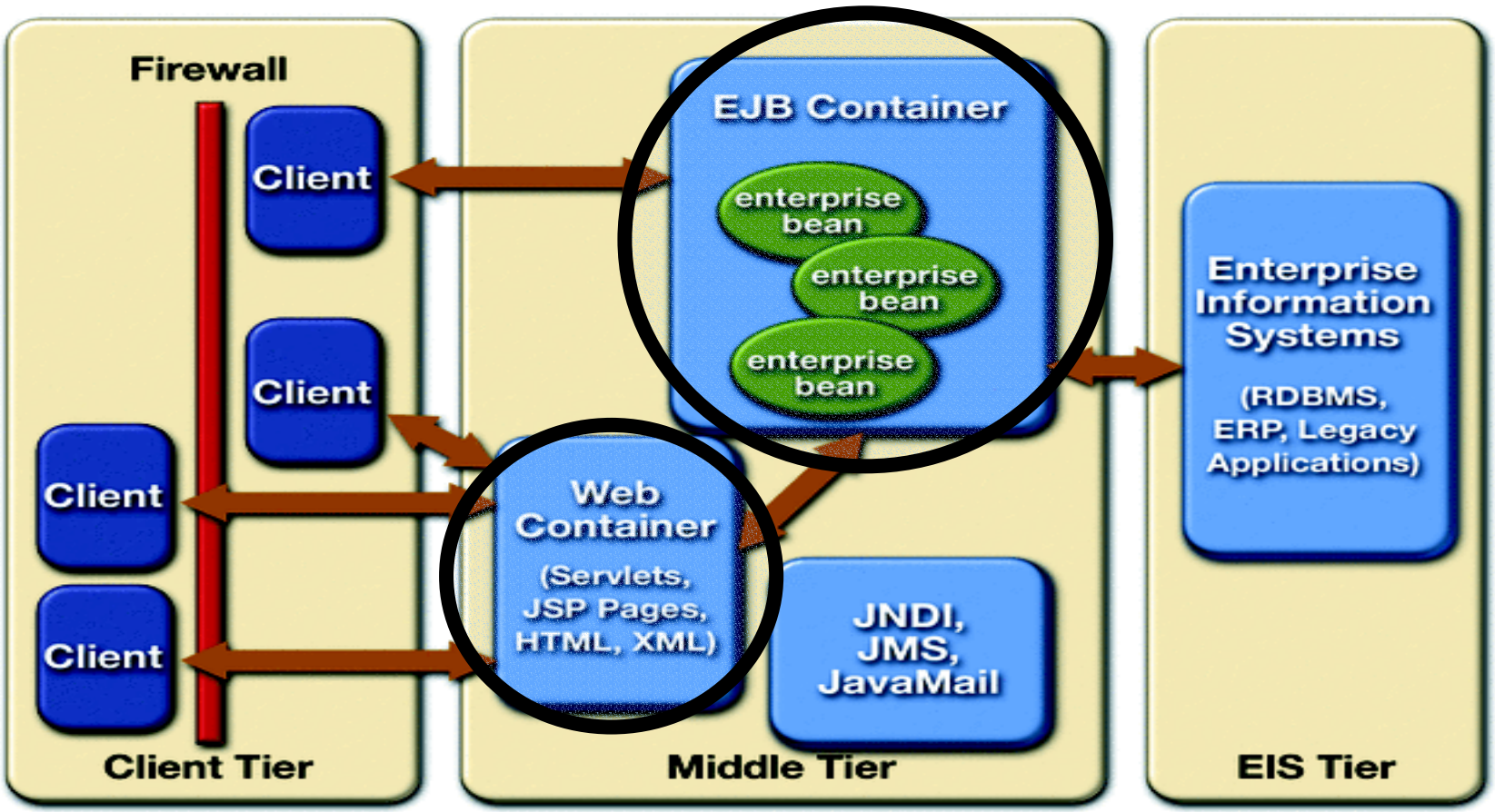
Entwicklung von EJB

- Die EJB 1.0 Spezifikation wurde auf der JavaOne ‘98 Conference veröffentlicht.
- Die Spezifikation wurde von Sun mit vielen Firmen erarbeitet:
  - Sybase
  - BEA
  - Oracle
  - IBM
  - Netscape
  - Novell
  - Tandem
  - Viele Andere
- 2.0 Spezifikation Aktuell (22. Aug. 2001)

J2EE Server Architektur

- Kann in Container und Server aufgeteilt werden.
  - Container
    - Bietet High-Level Dienste für das EJB
  - Server
    - Bietet Low-Level Dienste, z.B. Netzwerkverbindung
    - Der Server soll viele Protokolle wie RMI, IIOP oder DCOM unterstützen.
- Clients können in jeder Sprache geschrieben werden

J2EE Server Architektur



Server and Container Dienste

- Remote-Zugriff möglich, sobald Komponente im Server
- Multi-Client Unterstützung → Kopie pro Client
- Ressourcen Management (Thread, Sockets, DB-Verbindung)
- Komponenten Lebenszyklus
- Persistenz von Objekten
- Sicherheit
- Lokale Transparenz durch JNDI
  („Java Naming and Directory Interface”)
- Transaktions-Management

Clustering

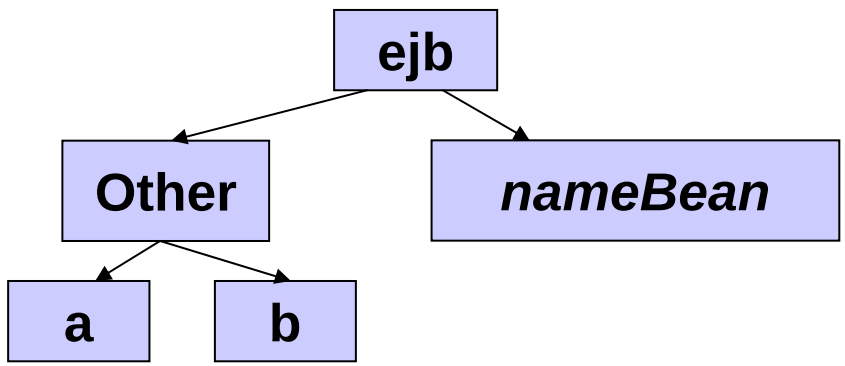
- Skalierbarkeit
  - Einen Server hinzufügen → mehr Leistung
- Hoch Verfügbarkeit
  - Wenn einer ausfällt, sind die anderen Server noch verfügbar
- Automatische Lastverteilung
- Clustering ist unsichtbar für den Entwickler

Hersteller

- Ein Hersteller muss die EJB Spezifaktion erfüllen und die Container zur Verfügung stellen
- Durch die Spezifikation ist sichergestellt, das eine EJB-Anwendung in jedem EJB-Server läuft
- EJB Container/Server Hersteller:
  - BEA WebLogic Tengah
  - NetDynamics
  - IBM WebSphere Advanced Edition
  - Oracle8i, Oracle Application Server
  - Persistence Power Tier
  - Progress Apptivity
  - Jboss (Open Source)
  - Forte, Informix, Netscape, Gemstone, Bluestone, Inprise und viele andere…

Java Naming and Directory Interface (JNDI)

- Einheitliches Namensschema zum Zugriff auf:
  - Netzwerkrechner
  - Dateien im Dateisystem
  - Objekte im Applikationserver (Servlets, JSPs, EJBs)
- JNDI wird benötigt, um auf EJBs zuzugreifen
- Bsp: jndi name: *ejb/nameBean, ejb/Other/a*



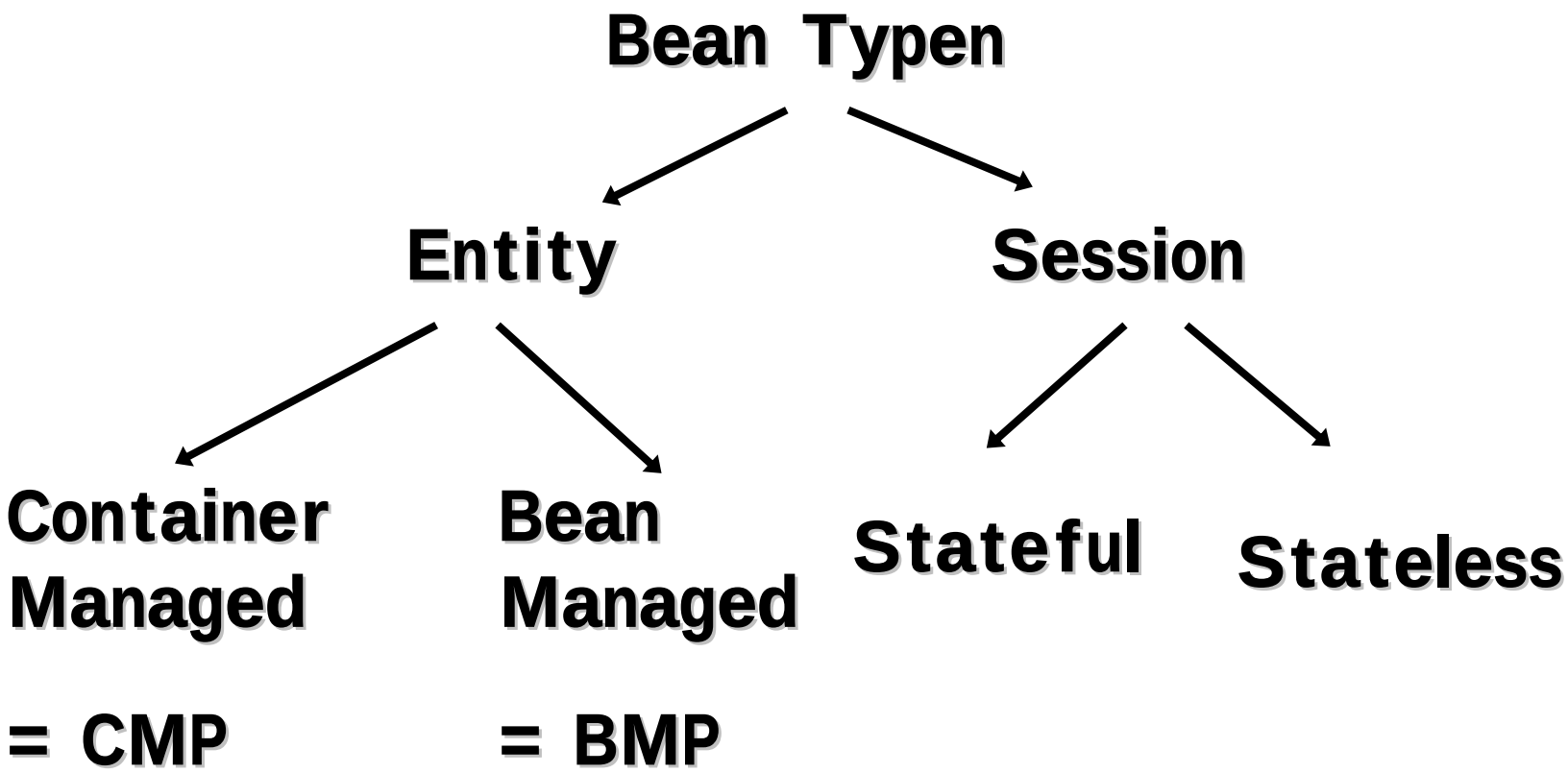
Architektur

EJBs

Beans Übersicht

Wie implementiert man EJB´s ?

Enterprise Java Beans – Übersicht



Quelle: Sun's Enterprise JavaBeans™ spec

Enterprise Java Beans – Übersicht

Entity Beans

- Persistente Daten – meistes DB - Zustand
- Finder Methoden
- PrimaryKey
- Container Managed Persistence (CMP)
- Bean Managed Persistence (BMP)

Session Beans

- Überdauern nur eine Benutzersitzung
- Applikations Logik
- Statefull Session Beans
- Stateless Session Beans

Enterprise Java Beans – Übersicht

Warum verschiedene Session-Beans?

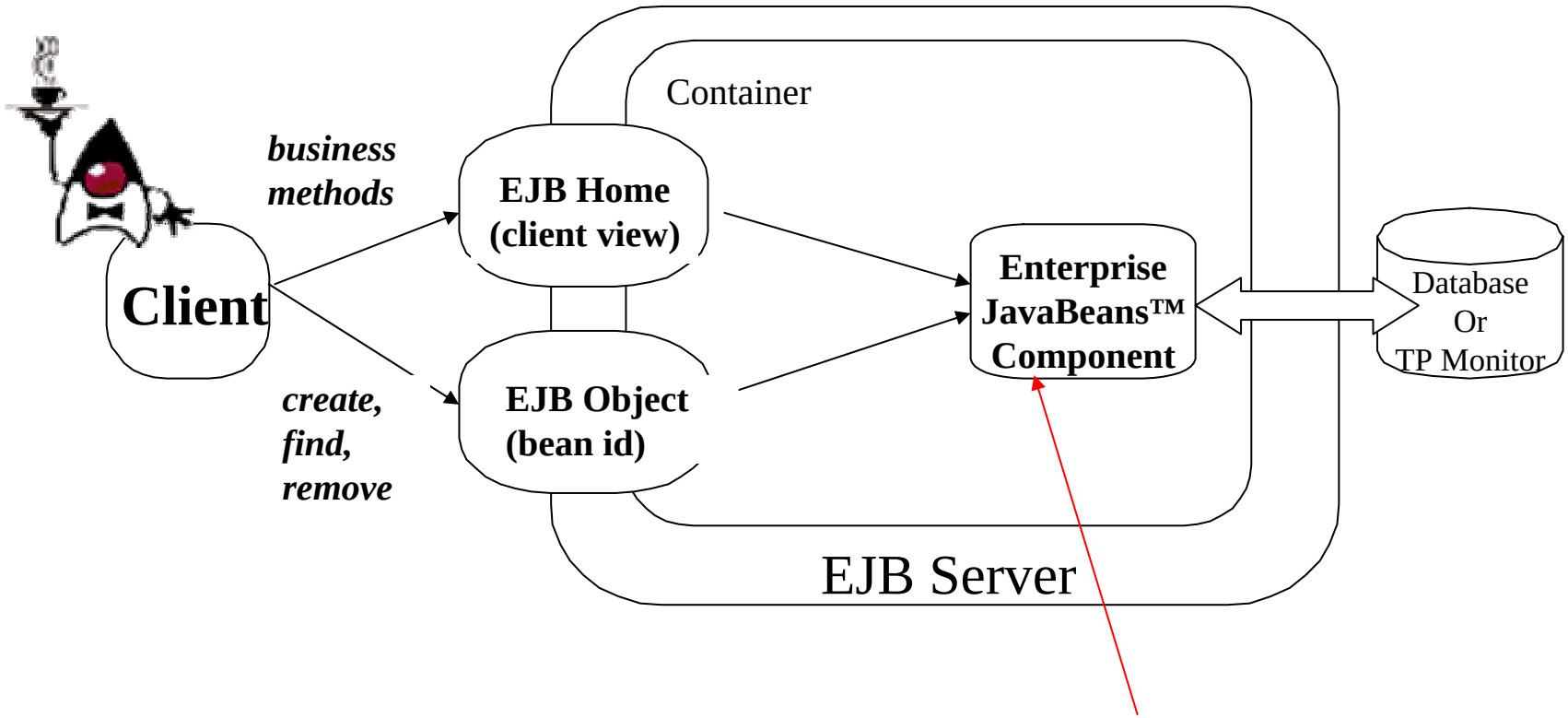
Statefull

- Haben über Methodenaufrufe hinweg einen Zustand
- Sind für eine Sitzung spezifisch
- Z.B. Abwicklung eines Workflows
- Z.B. Warenkorb des aktuellen Beutzers

Stateless

- Speichern keinen Zustand
- Vergleichbar mit statischen Methoden einer Klasse
- Eine Bean kann von verschiedenen Benutzern gleichzeitig genutzt werden
- Z.B. Loginmanager, oder Druckdienste anbieten

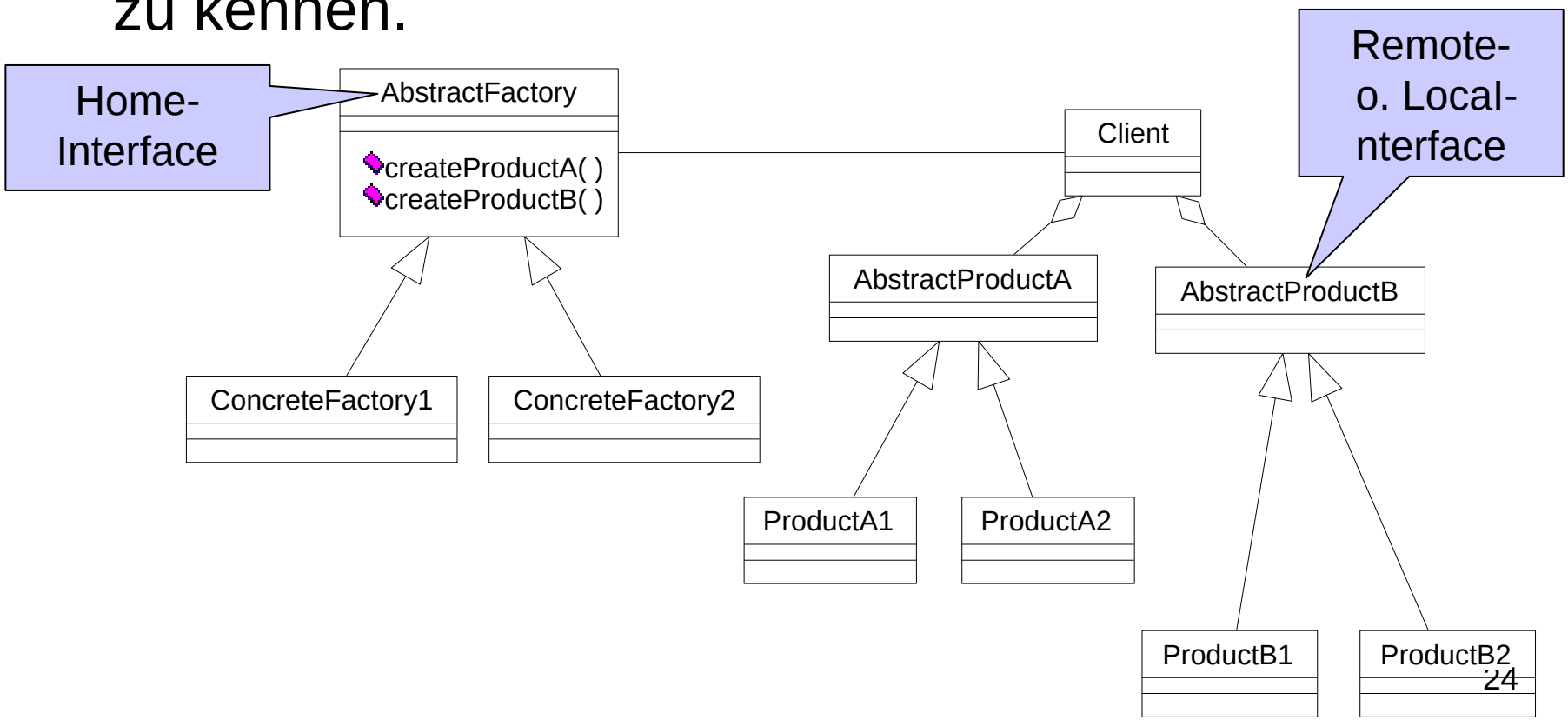
Enterprise Java Beans – Prinzip



Kann Session- oder Entity-Bean sein

Das Abstract Factory Pattern

- Hinter Java-Bans steckt das Abstract Factory Pattern. Ziel: Nutzung eines Interfaces um Objekte zu Kreieren ohne deren konkrete Implementierung zu kennen.



EJB Basiert auf Factories

- Enterprise Beans bestehen aus mindestens zwei unabhängigen Interfaces
- Home Interface: definiert eine Menge von Factory-Methoden
  - Der Client ruft diese Methoden auf um Instanzen zu erzeugen
  - *Der Container erzeugt die Konkreten Instanzen transparent*
- Remote/Local Interface: Hier werden die Methoden für den Zugriff definiert. Der Client greift **niemals** direkt auf ein Bean zu!

Home Interface

- Factory zum Erzeugen von Instanzen
- Namenskonvention:
  - *DeinBeanName***Home**.java
- Extends EJBHome
  - `public MeinBean`**`Home`**` extends EJBHome {`
    `…`
    `}`
- Methoden zum Erzeugen von Beans
  - `public void` **`create`**`(...) throws java.rmi.RemoteException, evtl. andere Exceptions`

Remote/Local Interface

- Interface für die ausführbaren Methoden des Beans
- Namenskonvention:
  - DeinBeanName.java /DeinBeanName**Local**.java
- Extends EJBObject /EJBLocalObject
  - ```
    public MeinBean Home extends EJBHome {
      public Object getPrimaryKey() throws RemoteException;
      public boolean isIdentical(EJBObject obj) throws …
      public void remove() throws RemoteException …
    }
    ```
- Enthält weiter die Business Methoden
    ```

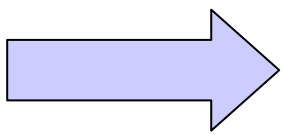
Persistence und Beziehungen

Wie implementiert man EJB´s ?

Entity-Bean = 1 DB Tupel

Datenbank

Name	Vorname
Vera	Mustermann
Klaus	Kinkel

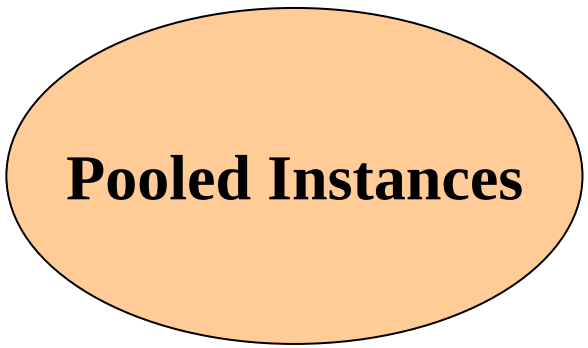


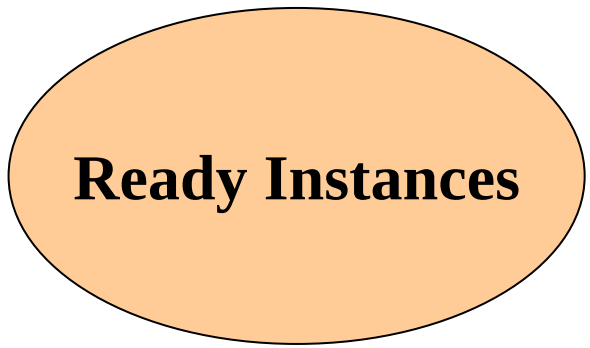
Entity-Objekt

Name=Mustermann
Vorname=Vera

Alle Beans werden vom Container verwaltet

- Der Container verwaltet einen Pool von Objekten





Korrespondieren nicht mit den “aktuellen Daten”. Verabeiten keine Anfragen. Nur leere Objekt-Hülsen eines Typs

Korrespondieren mit den “aktuellen Daten”
Sind zu einem echten EJB-Objekt assoziert. Bearbeiten Anfragen.

→ Daten-Verwaltung / Synchronisation mir einer DB muss vom Container gesteuert werden

Vom Pool zum “bereiten” Bean

- Instanzen werden mit gültigen Daten gefüllt wenn der Container ein weiters Bean benötigt
- Erzeugen: *ejbCreate*() and *ejbPostCreate*()
  - Werden aufgerufen wenn ein “neues” Objekt erzeugt wird. (Wenn es dieses also in der DB noch nicht gab)
  - z. B. eine *create()* Methode kann vom Home-Objekt aufgerufen werden
- Aktivieren: *ejbActivate*()
  - Werden aufgerufen wenn das Objekt bereits logisch (z.B. in der DB) existiert.
  - Z.B. wenn die Methode *findXXX()* vom Home Interface aufgerufen wurde

Vom “bereiten” Bean zum Pool

- Wenn eine der beiden Methoden aufgerufen wird, wird die Instanz wieder zum Pool hinzugefügt
  - ejbRemove() wenn das zugehörige Logische-Objekt gelöscht wurde (z.B. ein Löschen in der DB)
  - ejbPassivate() wenn das zugehörige Logische-Objekt existiert, aber das Bean deaktiviert werden soll. (z.B. weil es eine längere Zeit nicht mehr benutzt wurde)

Persistence Modelle – CMP/BMP

Es gibt zwei Modelle um Daten dauerhaft zu speichern ?

BMP - Die Persistence wird vom Bean verwaltet

- Man muss nun selbst das Finden, Laden, Speichern, Erzeugen, Aktivieren und Passivieren der Beans kodieren
- Die entsprechenden SQL Befehle sind alle selbst zu schreiben
- ejbCreate, ejbLoad, ejbRemove, and ejbStore Methoden müssen implementiert werden

CMP - Die Persistence wird vom Container verwaltet

- Der EJB-Container übernimmt das Datenbank-Mapping
- Der SQL-Code wird vom Container selbst generiert
- Für die Finder Methoden werden sogenannte EJB QL -Anfragen verwendet

CMP- Beispiel

- Der Code für die Persitence-Felder, welche vom Container verwaltet werden sollen sieht folgender maßen aus
- ***public abstract*** setXXXX() und getXXX()

```
public abstract class BaseattribBean implements EntityBean {
        ...
        public abstract void setBa_ID(java.math.BigDecimal baId);
        public abstract void set Ba_Name(java.lang.String baName);
          ...
}
```

CMP- Beispiel – der zugehörige EJB-Descriptor

```
<entity>
        <ejb-name>Baseattrib</ejb-name>
      <local-home>compiler.ejb.entity.BaseattribHome</local-home>
      <local>compiler.ejb.entity.Baseattrib</local>
      <ejb-class>compiler.ejb.entity.BaseattribBean</ejb-class>
        <cmp-field>
              <field-name>Ba_ID</field-name>
      </cmp-field>
      <cmp-field>
              <field-name>Ba_Name</field-name>
      </cmp-field>
</entity>
```

EJB QL

Wie werden Objekte gefunden? Abfragen?

EJB Abfrage - Sprache: EJB QL

- Standardsprache zum definieren von finderXXX()-Methoden
- Teilmange von SQL92 Standard mit einigen Erweiterungen
- SQL: *Select* .. *From* … *Where* Klauseln
- EJB QL :: = [Select_Klausel] From_Klausel [Where_Klausel]

Beispiel eines Finders

```
public interface CaseHome extends javax.ejb.EJBLocalHome {
    public Case create(BigDecimal caId) throws ...;
    public Case findBa_ID (double caId) throws ...;
}
```

```
<query>
  <query-method>
    <method-name>findBa_ID</method-name>
    <method-params>
      <method-param>int</method-param>
    </method-params>
  </query-method>
  <ejb-ql>
    <where baId = ?1>
  </ejb-ql>
</query>
```

Methodenname im Home

Parameter der Methode findXXXX()

EJB Abfrage-Sprache: EJB QL

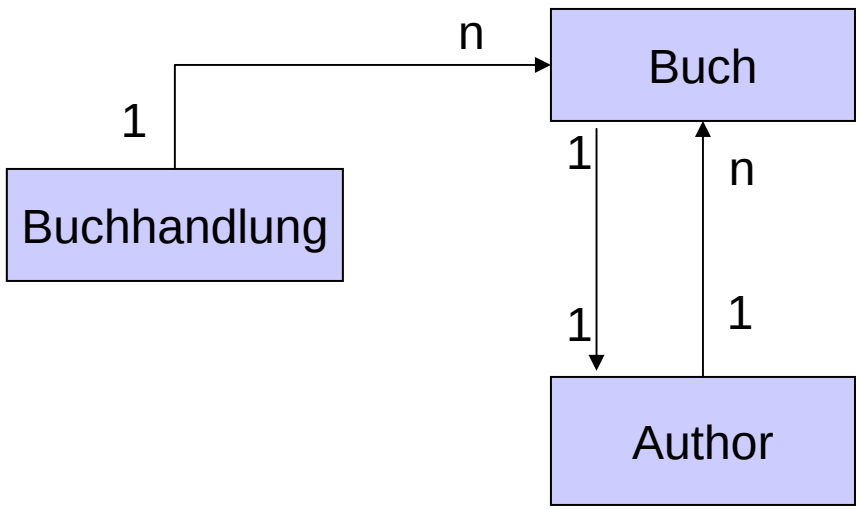
- EJB QL Parameter korrespondieren zu den Parametern welche in einer find/select Methode definiert wurden
- Entity-Beans müssen einen sogenannten “abstract schema name” besitzen, der in der Anfrage verwendet wird
- Bei Namensänderungen z.B Bean-Name muss die EJB QL Anfrage nicht verändert werden

EJB Abfrage-Sprache: EJB QL

CMR Feld 1:n

FROM Buchhandlung L, IN (L.buecher) b
WHERE b.author.vorname = ‘Katharina’ AND
b.author.nachname = ‘Morik’

CMR Feld 1:1

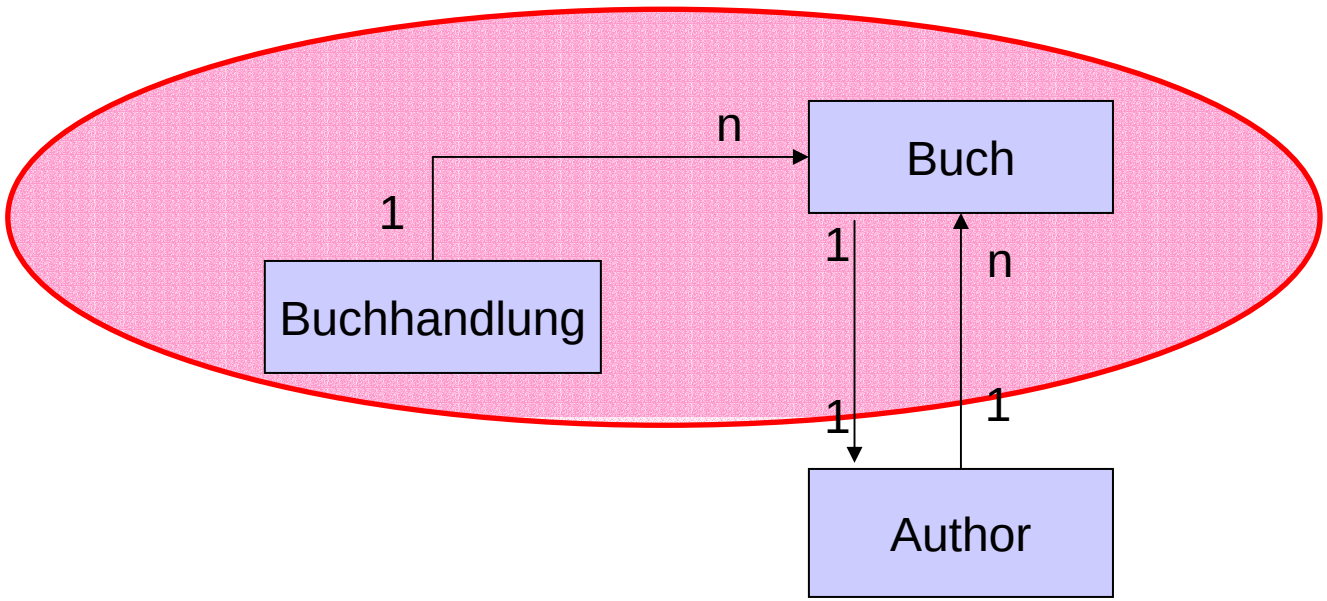


Beziehungen in EJB

Wie werden Beziehungen definiert?

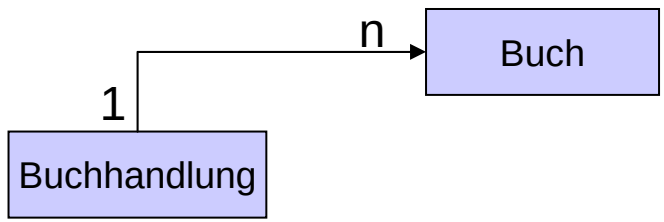
Wie werden Beziehungen definiert?

Wir wollen folgende Beziehung definieren!



Wie werden Beziehungen definiert?

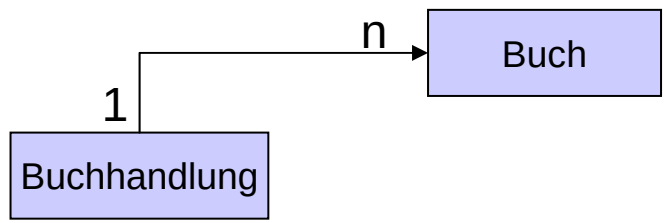
```
<ejb-relation>
    <ejb-relation-name>Buchhandlungen_haben_Buecher</ejb-relation-name>
    <ejb-relationship-role>
        <ejb-relationship-role-name>Buchhaen_Rolle</ejb-relationship-role-name>
          <multiplicity>One</multiplicity>
          <relationship-role-source>
              <ejb-name>Buchhandlung</ejb-name>
          </relationship-role-source>
          <cmr-field>
              <cmr-field-name>buecher</cmr-field-name>
              <cmr-field-type>java.util.Collection</cmr-field-type>
          </cmr-field>
    </ejb-relationship-role>
```



Quelle: Sun's Enterprise JavaBeans™ spec

Wie werden Beziehungen definiert?

```
<ejb-relationship-role>
        <ejb-relationship-role-name>Buch_Rolle</ejb-relationship-role-name>
         <multiplicity>Many</multiplicity>
                  <relationship-role-source>
                  <ejb-name>Buch</ejb-name>
         </relationship-role-source>
</ejb-relationship-role>
</ejb-relation>
```



Quelle: Sun's Enterprise JavaBeans™ spec

Der zugehörige Code im Entity-Bean

```
public abstract class Buchhandlung implements
EntityBean {

….

public abstract Collection getBuecher();

….

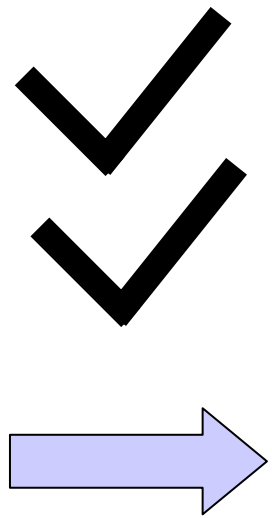
}
```

CMR Felder fangen stets
Mit dem Schlüsselwort
getXXX an!

Architektur

EJBs

Dienste, Probleme

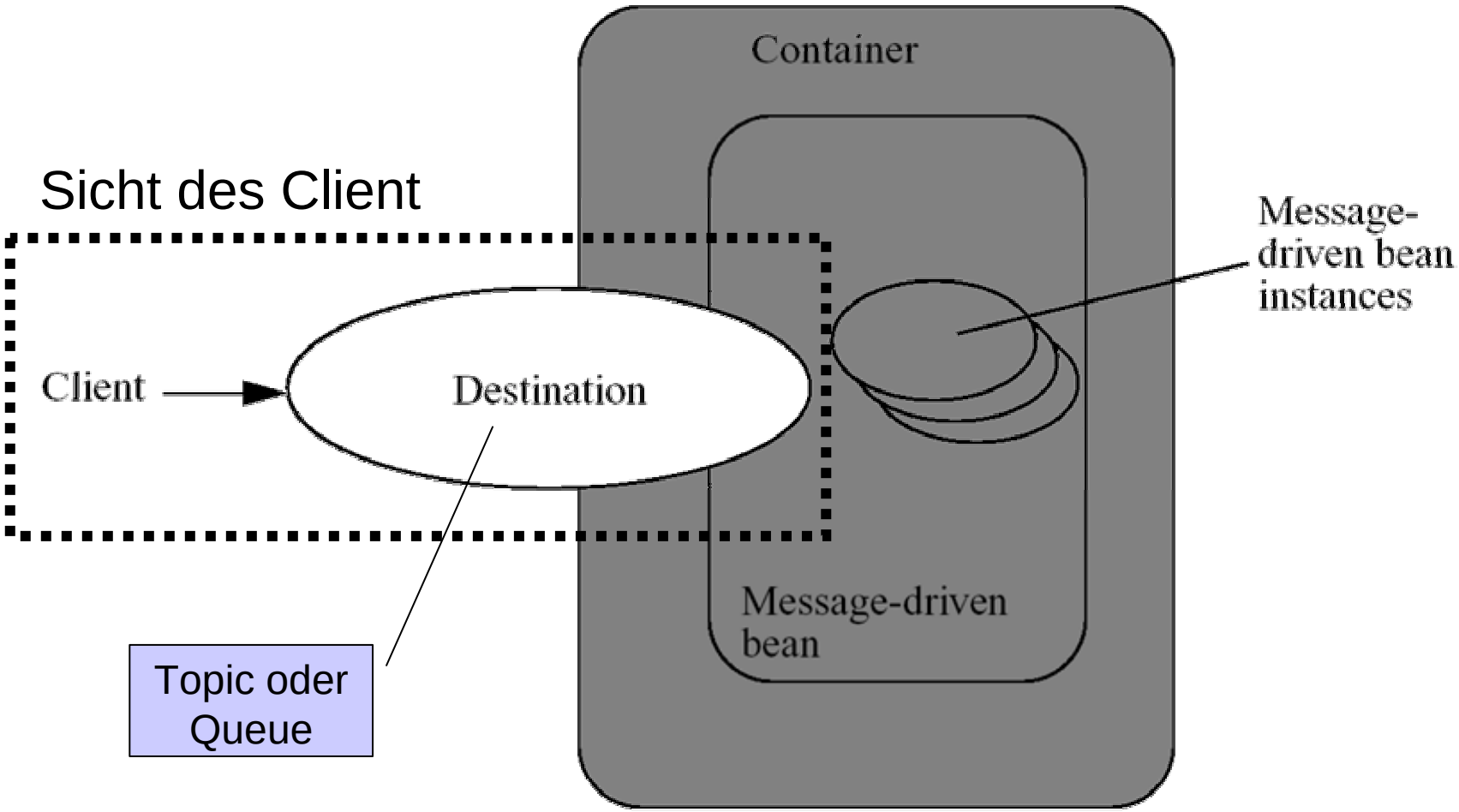


Message Driven Beans

- Problem: Nachrichten wurden bisher synchron verarbeitet

- Asynchrone Verarbeitung: Message Driven Beans
  - Point – to – point (Queue) (Client → Bean)
  - Publish – subscribe (Topic) (Client → Verteiler)
  - Es ist sichergestellt, das die Nachricht auch ankommt

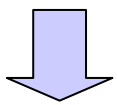
Senden einer Nachricht



Quelle: Sun's Enterprise JavaBeans™ spec

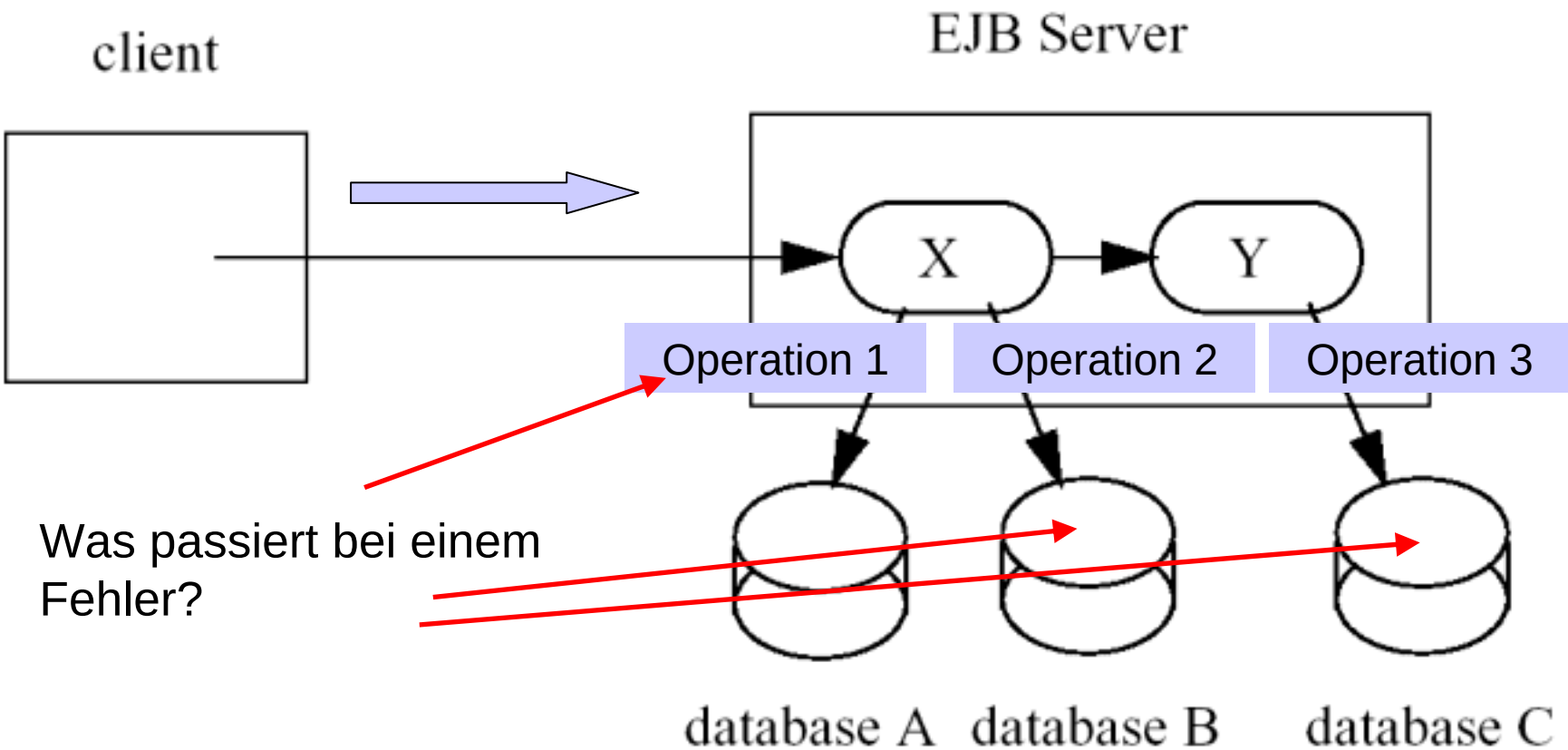
Was ist eine Transaktion?

- Eine Folge von Aktionen soll **atomar** sein
  - Entweder alle Aktionen werden wirksam
  - Oder keine Aktion soll ein Wirkung haben



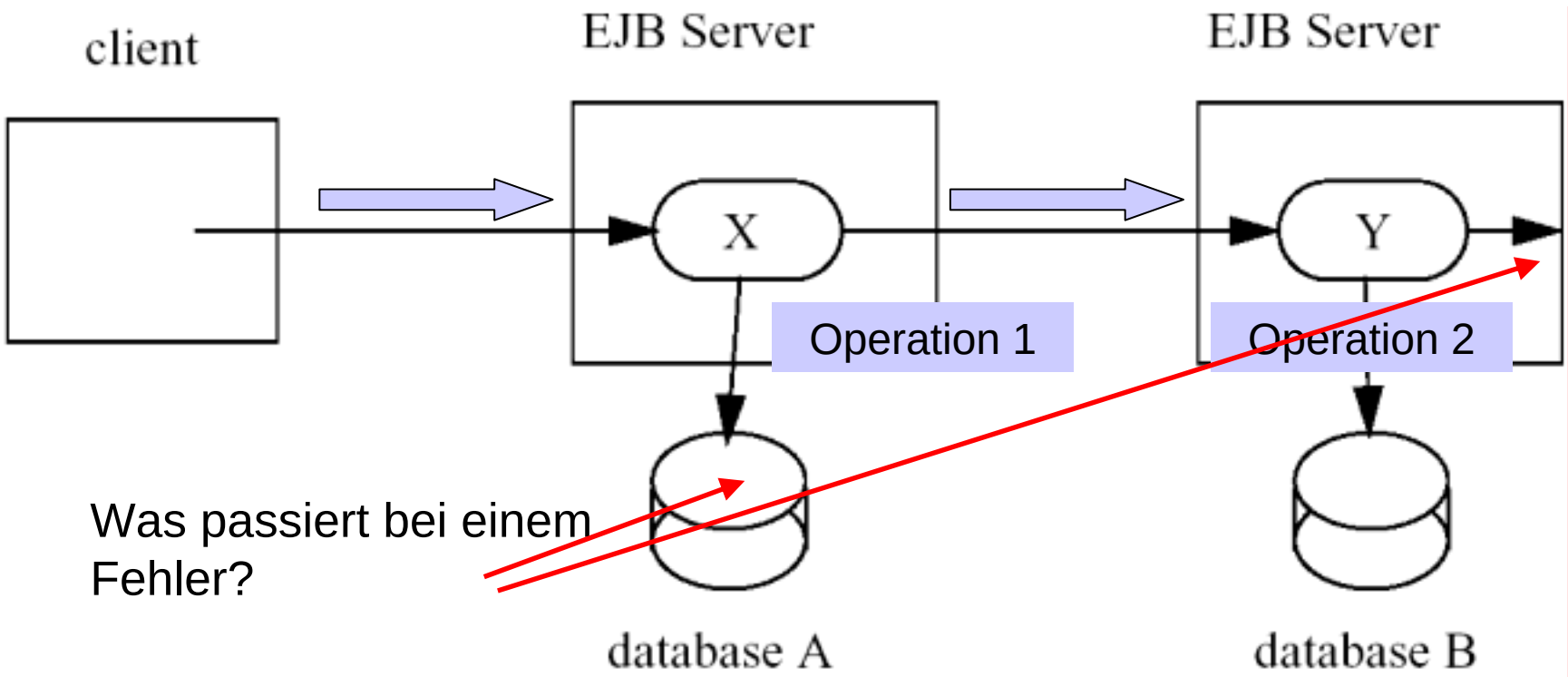
Was bedeutet das bei einem EJB-Server?

Das Problem (1/2)



Quelle: Sun's Enterprise JavaBeans™ spec

Das Problem (2/2)



Quelle: Sun's Enterprise JavaBeans™ spec

Die Lösung: Trasaktionsunterstützung bei J2EE

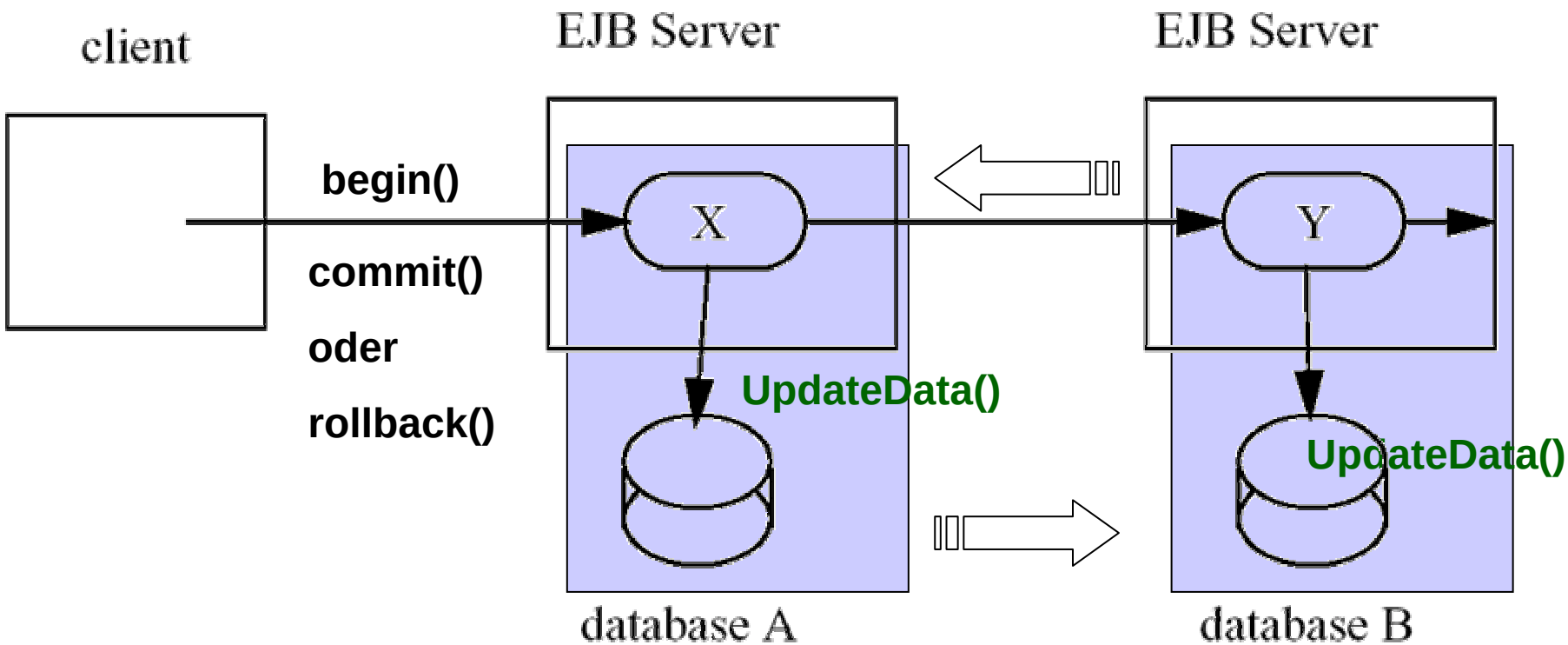
- J2EE-Server hat einen Transaktions-Monitor
  - Transaktionen werden im ”Transaction context” ausgeführt
- ”Transaction context”
  - Zugriff vom Client oder vom Bean über das UserTransaction Interface

UserTransaction Interface

- public void begin()
- public void commit()
- public void rollback()
- public void setRollbackOnly()
- public int getStatus()
- public void setTransactionTimeout(int timeout)

STATUS_ACTIVE
STATUS_MARKED_ROLLBACK
STATUS_PREPARED
STATUS_PREPARING
STATUS_UNKNOWN
STATUS_COMMITTED
STATUS_COMMITTING
STATUS_ROLLEDBACK
STATUS_ROLLING_BACK
STATUS_NO_TRANSACTION

Transaktionen im Einsatz



Nachteile - Effizienz

- **Overhead:** Alle Beans sind Transactionsfähig, Thread-sicher, verteilt
  - Dies sollte Optional sein, wie z.B. bei COM+
- **Hoher Aufwand:** Vergleich zur *normalen* RMI Implementation
  - Mehr Code (Interfaces,…)
  - Mehr Denken (Welche Beans?, Welche Interfaces?,…)

Performanz ⇩

Zeitaufwand ⇧

CMP – Problematik

- EJB-QL unterstützt nur eine Teilmenge der Funktionen in wirklichen DB-Systemen (keine Aggregatfunktionen)
- Effizienzprobleme → DB ist schneller
  - Keine Informationen für den jeweiligen DB-Optimizer verfügbar (Bsp. Oracle: Index explizit angeben, „First_Rows“)
  - Nutzung von internen DB-Funktionen nicht möglich
- Abfrage fest definiert (Descriptor)
  - Keine dynamischen Abfragen vom Client
- EntityBean kann nur auf eine Tabelle gemappt werden. Spezielle OR-Mapping Tools sind viel mächtiger.

Hersteller

- Spezialisierte Container können zusätzliche Dienste zu denen in der EJB Spezifikation bieten

→ Ein Enterprise Bean das einen solchen Dienst nutzt, kann nur in einem Container eingesetzt werden, der diesen Dienst bietet

- Hoher Preis: ca. 12000 €
- Probleme bei freien J2EE Servern?
  - Keine Technische Unterstützung
  - Schwere zu benutzen (keine Grafische Oberfläche ...)
  - Keine Integration zu Entwicklungstools tools (z.B., Jbuilder)
  - Bugs? Probleme während des Projektes?

Literatur

- Linda G. DeMichiel, L. Ümit Yalçinalp, Sanjeev Krishnan. Enterprise JavaBeansTM Specification, Version 2.0, EJB web site. Sun Microsystems, Inc. 2001.
- Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition. EJB web site. Sun Microsystems, Inc. 2000.
- Enterprise JavaBeans web site. http://java.sun.com/products/ejb
- Matena, Vlada., Harper, Mark. Enterprise JavaBeans Specification, v1.1. EJB web site. Sun Microsystems, Inc. 1999.
- Roman, Ed. Mastering Enterprise JavaBeans and the Java 2 Platform Enterprise Edition. John Wiley and Sons Inc. 1999.
- Thomas, Anne. Enterprise JavaBeans Technology Server Component Model for the Java Platform. EJB web site. Patricia Seybold Group. 1998.
- Java Enterprise in a Nutshell – David Flanagan
- Core Servlets and Java ServerPages – Marty Hall
- Designing Enterprise Applications with J2EE – Nicholas Kassem
- Enterprise JavaBeans – Richard Monson-Haefel

Ende