

Informationsintegration und Web-Portale

Kapitel:

*Mehrschichtenarchitekturen
und Komponentenumgebungen*

Erik Buchmann

buchmann@ira.uka.de



Geschäftsprozesse und „Enterprise Applications“

- Geschäftsprozess = *Verkettungen von Aktivitäten, die zur Erstellung der Unternehmensleistung beitragen*
 - Vertriebsprozess, Materialbereitstellungsprozess, ...
- „Gute Geschäftsprozesse“ sind wichtig für den Erfolg eines Unternehmens
- Verbesserung von Geschäftsprozessen:
 - Optimierung
 - Automatisierung → „Enterprise Applications“
- Unsere Frage heute:
 - Wie entwickle ich Software zur Unterstützung von Geschäftsprozessen? → Methodik und Technologie

Einführung

Probleme

Lösungsansätze

Komponenten

EJB

Tutorial

Schluss





Agenda

- „Enterprise Applications“ – Herausforderungen
- Grundlegende Lösungsansätze
- Komponenten und Komponentenumgebungen
- Konkretes Beispiel: Enterprise JavaBeans

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





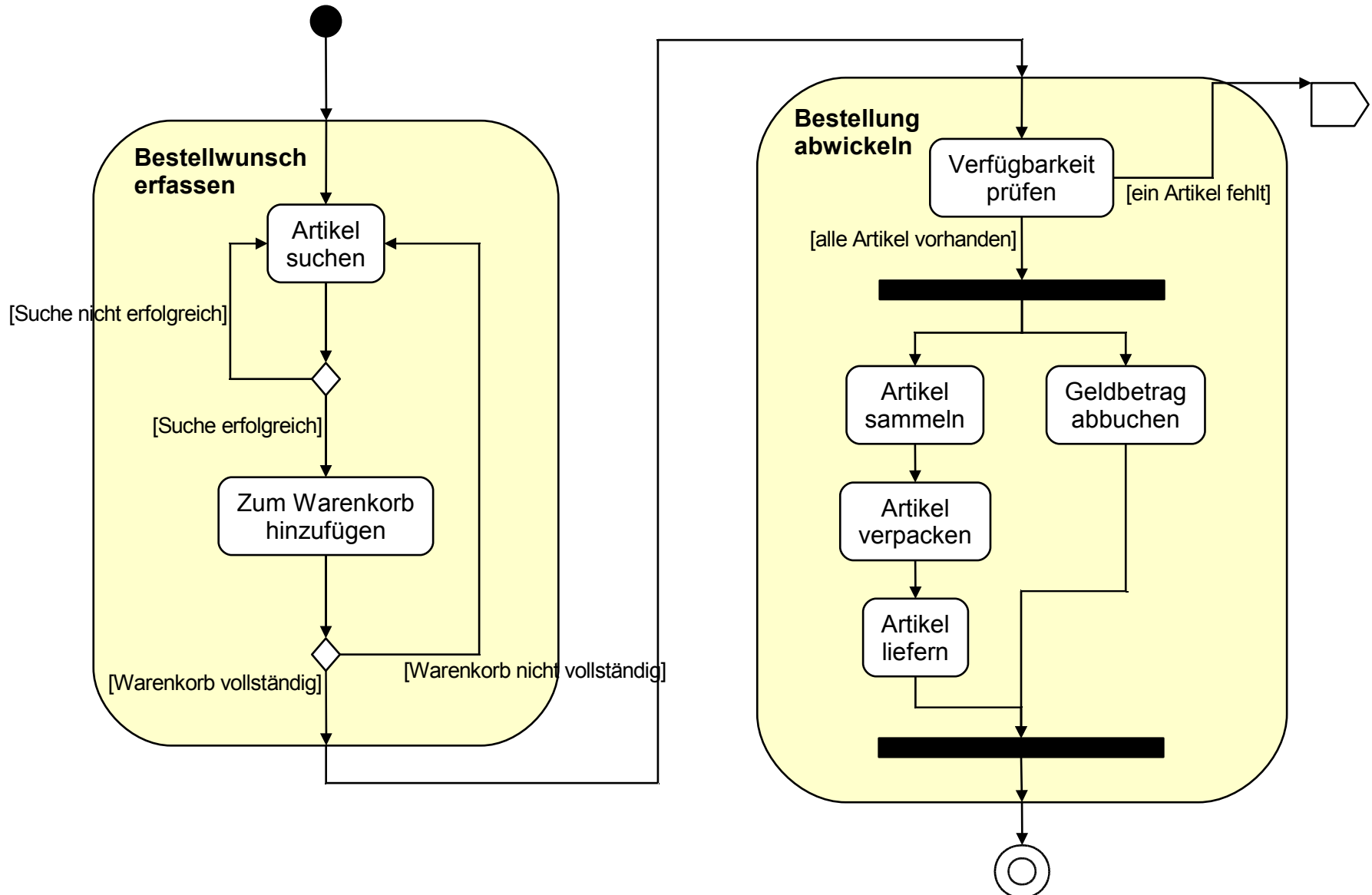


Historie

Ansatz:	Mainframes	Client-Server-Architektur	Mehrschichtenarchitektur	Dienstorientierte Architektur
Zeit:	1975 - 1985	1985 - 1995	1995 - ...	2002 - ...
Motivation:	Digitale Erfassung und Verarbeitung von Daten	Unterstützung komplexer Arbeitsvorgänge (CAD, CASE, ...)		
Rahmenbedingungen:	Extrem teure Rechnerleistung	Arbeitsplatzrechner werden erschwinglich	HTML/HTTP, CGI, Java, Servlets, ...	XML, WebServices, ...
Grundidee:	Ein Großrechner für die Datenverarbeitung + viele einfache Terminals zur Datenerfassung	Ein zentraler Rechner mit DBMS, ERP o.ä. (Server) + leistungsfähige Arbeitsplatzrechner (Clients)		



Geschäftsprozesse heute: Online-Kauf





Problemfelder aus Unternehmenssicht (1)

- Outsourcing
 - Abgabe von Geschäftsfunktionen an externe Firmen
 - horizontales / vertikales Outsourcing
 - Ziele:
 - Rationalisierung von Geschäftsprozessen durch Spezialisten
 - Reduzierung der Prozesskomplexität
 - Konzentration auf das Kerngeschäft
 - Beitrag von Komponentenarchitekturen:
 - Zukauf von IT-Dienstleistungen über definierte, systemunabhängige Schnittstellen
 - Auslagern von selbstentwickelten Komponenten ohne Änderungen an der internen Softwarelandschaft

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Problemfelder aus Unternehmenssicht (2)

- Just-in-Time-Fertigung
 - Materialbereitstellung zum exakt nötigen Zeitpunkt
 - Komplexe Logistik, Ganzheitliche Systeme
 - Ziele:
 - Einsparung von Lagerhaltung
 - Beitrag von Komponentenarchitekturen:
 - Zugriff auf die Komponenten mit den Geschäftsmethoden der Sub-Unternehmer und Lieferanten
 - Lagerverwaltung
 - Auftragsstatus
 - Logistik

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Problemfelder aus Unternehmenssicht (3)

- Integration von Geschäftsprozessen
 - Zusammenlegung von firmeninternen Systemen
 - Firmenzusammenschlüsse
 - global Player, weltweit verteilte Workflows
 - Beitrag von Komponentenarchitekturen:
 - bereits existierende Komponenten können weiter genutzt werden
 - technische Infrastruktur (DB, App-Sever) kann zusammengelegt werden, ohne dass die Business-Komponenten geändert werden müssen

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Problemfelder aus Unternehmenssicht (4)

- Neue Geschäftsmodelle, neue Märkte
 - Anbieten von Standardkomponenten
 - einmal entwickeln und testen, mehrfach vertreiben
 - z.B. Lohnbuchhaltung, CRM, Logistik
 - Hosten von Business-Logic
 - nicht nur einfache Miet-Server, sondern ganze Miet-Geschäftsprozesse
 - „SAP aus der Steckdose“
 - Beitrag von Komponentenarchitekturen:
 - machen dies überhaupt erst möglich

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Problemfelder aus technischer Sicht (1)

- Verteilte Entwicklungsprozesse
 - geschäftskritische Komponenten vom Spezialisten vor Ort entwickeln, Standardkomponenten in Indien
- Sicherheit
 - Komponenten einzeln testen
 - getestete Komponenten müssen nicht bei jeder Anpassung am System neu übersetzt werden
- Lastverteilung / Ausfallsicherheit
 - integrierte Lokationsdienste; es ist für den Geschäftsprozess egal auf welcher Maschine welche Komponenten laufen
 - Komponenten mit hoher Last lassen sich leicht auf stärkere Server migrieren

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Problemfelder aus technischer Sicht (2)

- Integration neuer Technologien
 - z.B.:
 - Web 2.0, AJAX, Webportale
 - Thin Clients, Blackberry-Pushdienste, Mobiltelefon
 - Änderungen am Frontend unabhängig von den Komponenten mit den Geschäftsmethoden

Einführung

Probleme

Lösungsansätze

Komponenten

EJB

Tutorial

Schluss





Zentrale Herausforderungen

- Einbindung externer Akteure
 - Kunden, Lieferanten, Bank (B2C, B2B)
 - Ähnliches Problem: global agierende Unternehmen
- Vielzahl und Vielfältigkeit der Akteure
 - Wie viele Kunden erwarte ich? Welche Endgeräte?
- Automatisierung transaktionaler Vorgänge
 - „Geld abbuchen“ und „Artikel liefern“
- Integration existierender Systeme
 - Zugriff auf das ERP in „Verfügbarkeit prüfen“
- Ständige Veränderungen (→ Geschäftsprozessoptimierung)
 - Morgen verkaufen wir auch über eBay...
- Qualitätszusagen

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Rahmenbedingungen

- Knapper Zeitrahmen
 - Time-to-Market ist essentiell in globalen Märkten mit vielen Konkurrenten
- Knappe personelle Ressourcen
 - Entwicklung von Unternehmenssoftware erfordert Kenntnis über die installierten Geschäftsprozesse
→ Spezialisten erforderlich
- Knapper Kostenrahmen für
 - die Entwicklung
 - den Betrieb (hohe Wartbarkeit notwendig)

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Grundlegende Lösungskonzepte



Ausgangslage

Klienten mit unterschiedlichen Fähigkeiten



Einführung

Probleme

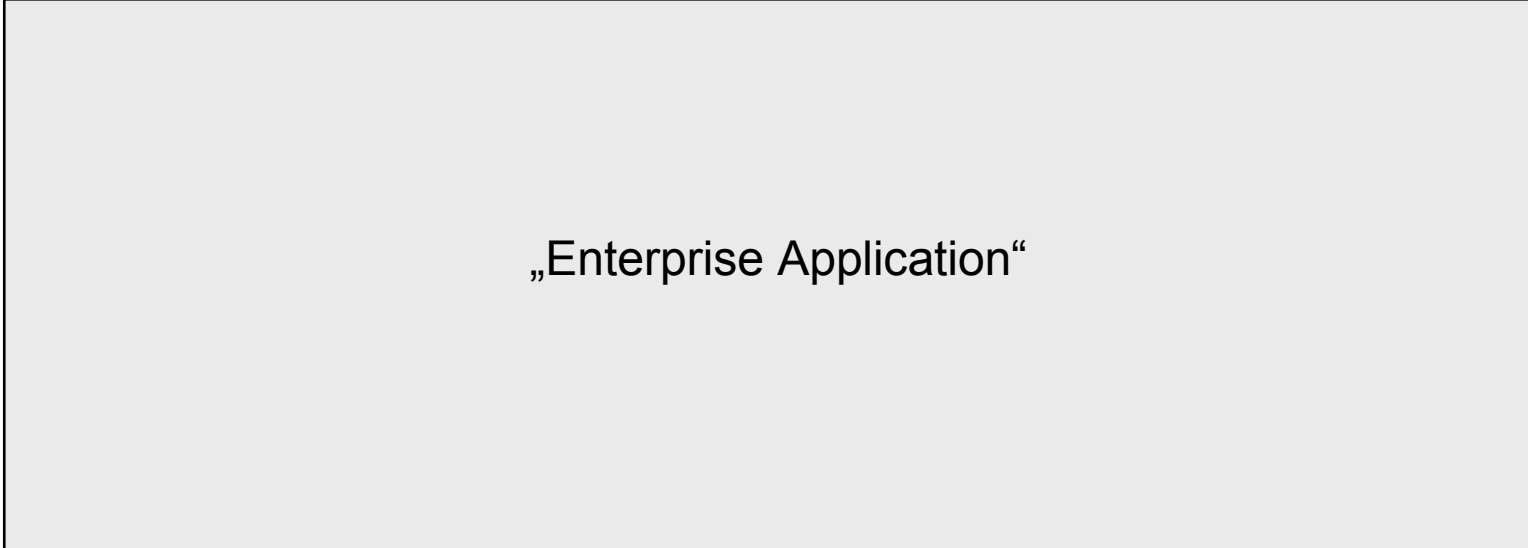
Lösungsansätze

Komponenten

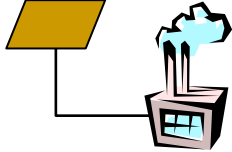
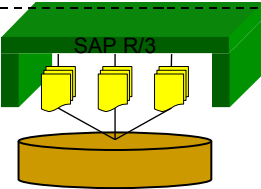
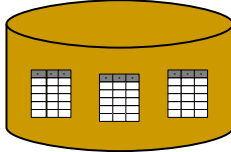
EJB

Tutorial

Schluss



Rohdaten und -dienste (nicht integriert)





Mehrschichtenarchitekturen

Klienten mit unterschiedlichen Fähigkeiten



Einführung

Interaktion

Probleme

Geschäftsprozesse (auf Basis der Geschäftsobjekte)

Lösungsansätze

Geschäftsobjekte (technisch und inhaltlich integrierte Daten)

Komponenten

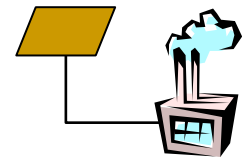
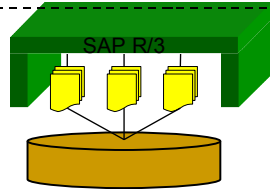
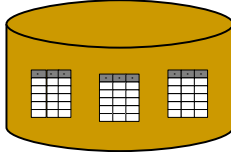
Datenhaltung und Systemintegration

EJB

Tutorial

Schluss

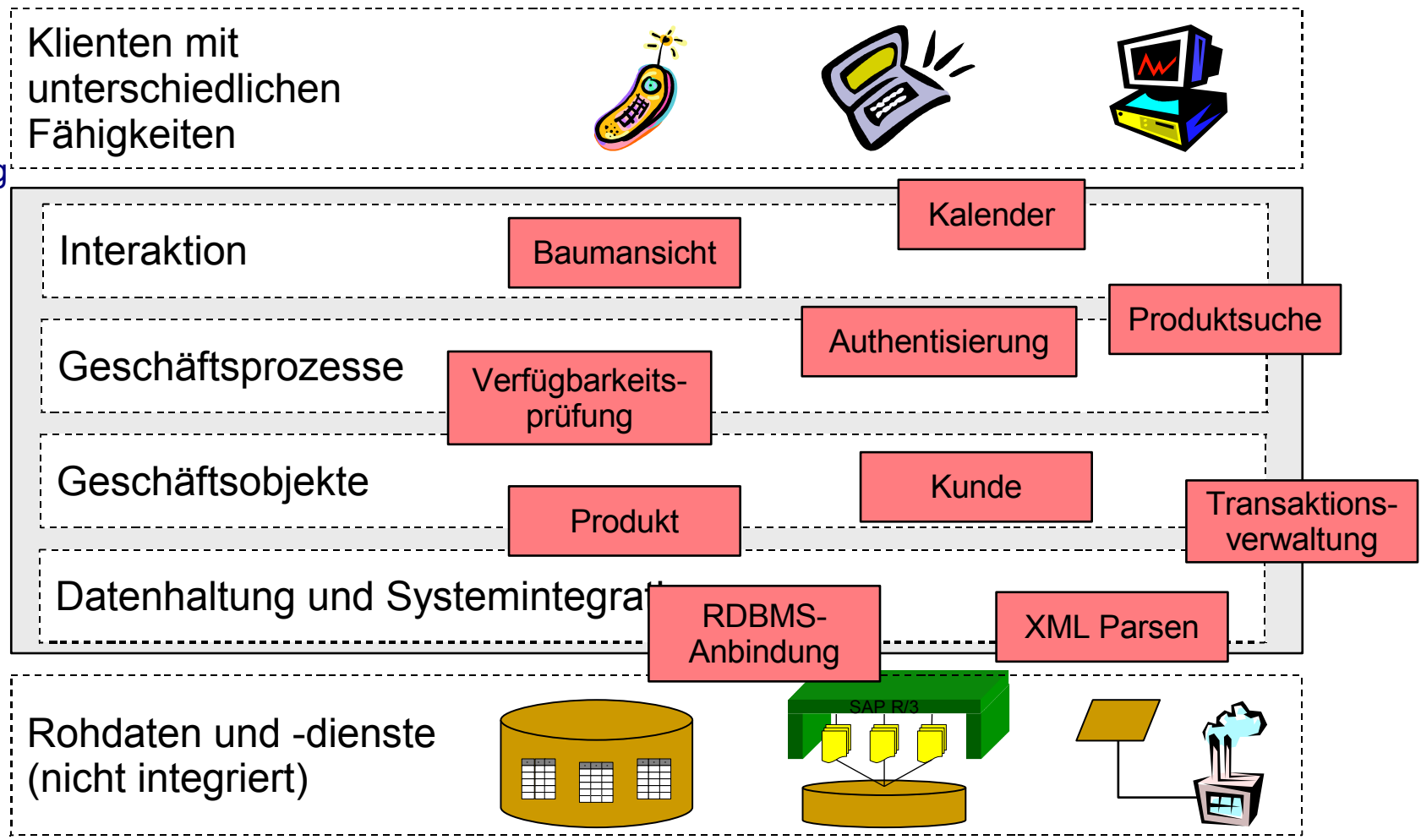
Rohdaten und -dienste (nicht integriert)



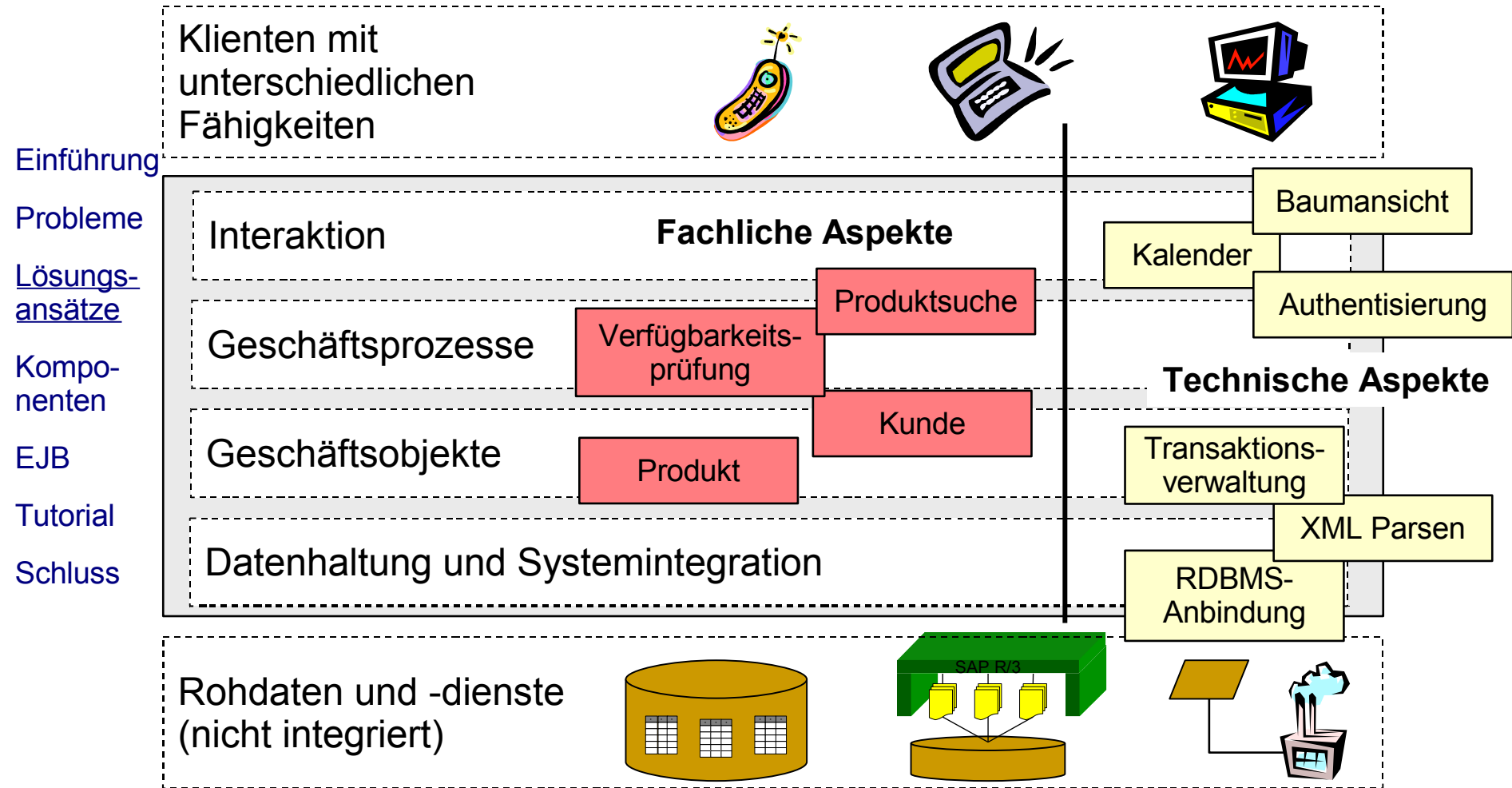


Modularisierung

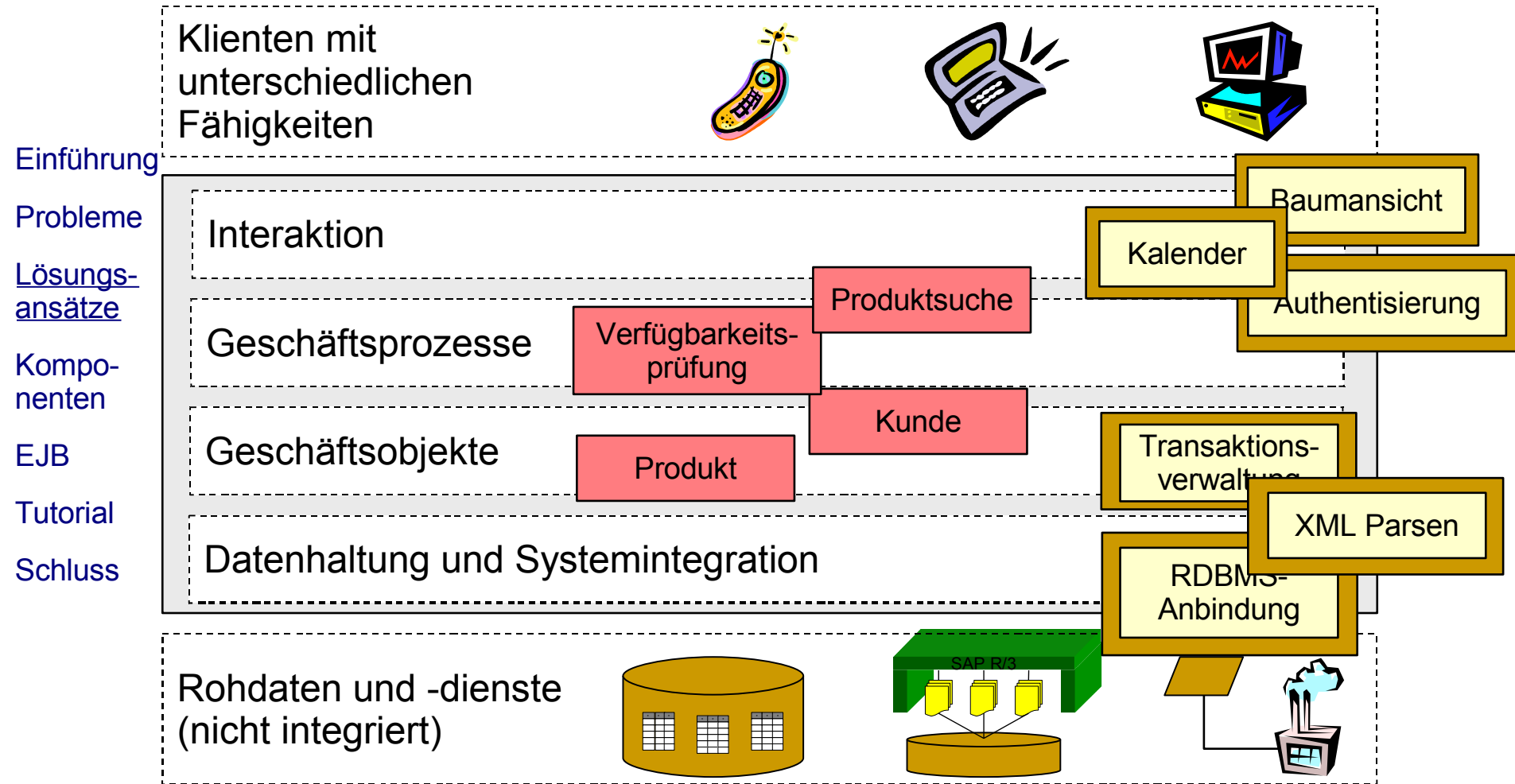
Einführung
Probleme
Lösungsansätze
Komponenten
EJB
Tutorial
Schluss



„Separation of Concerns“



Standardisierung





Die Vorteile auf einen Blick

- Mehrschichtenarchitektur
 - Austauschbarkeit einzelner Schichten
 - 1:N-Lösungen (z.B. 1 Prozess, N Oberflächen)
- Modularisierung
 - Strukturierung des Entwicklungsprozesses
 - Wiederverwendbarkeit von Teillösungen
- „Separation of Concerns“
 - Wiederverwendbarkeit und Austauschbarkeit technischer Module (hier noch unternehmensintern!)
- Standardisierung
 - Offene Märkte für Teillösungen werden möglich
 - Profilierung als Spezialist für Teillösungen möglich

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Komponenten und Komponentenumgebungen

Komponentenumgebungen

Klienten mit unterschiedlichen Fähigkeiten



Interaktion

Geschäftsprozesse

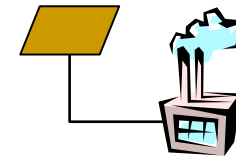
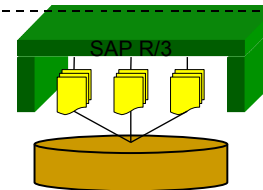
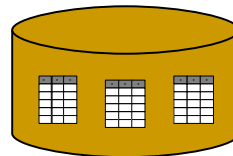


Geschäftsobjekte



Datenhaltung und Systemintegration

Rohdaten und -dienste
(nicht integriert)



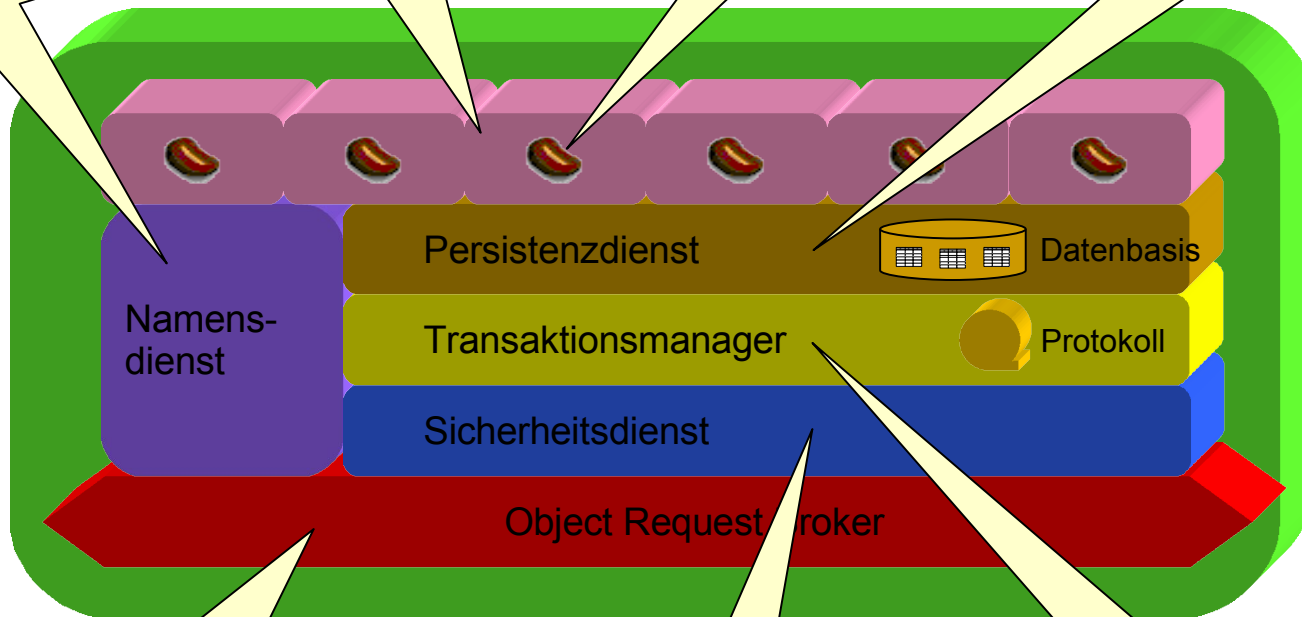
Dienste einer Komponentenumgebung

5. Auffinden von Komponenten und Diensten

2. Container für Komponenten

1. Geschäftskomponenten

3. Transparente Zustandsspeicherung



6. Transparente Kommunikation zwischen verteilten Objekten

7. Transparente Zugriffskontrolle

4. Transparente Transaktionsverwaltung



Was sind Komponenten?

"A component is a piece of software that is small enough to create and maintain, big enough to deploy and support, and with standard interfaces for interoperability."

(J. Harris, President, CI Labs, 1995)

"A component is a reusable, self-contained piece of software that is independent of any application."

(R. Orfali, D. Harkey, J. Edwards:
The Essential Distributed Objects Survival Guide)

Zur Definition siehe auch C. Szyperski oder F. Griffel

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Eigenschaften von Komponenten

- vielfältig einsetzbare, vermarktbare, in sich abgeschlossene Funktionseinheiten.
- interagieren nur über wohldefinierte Schnittstellen mit ihrer Umgebung.
- können leicht und flexibel zu größeren Funktionseinheiten gruppiert werden.
- Beispiele:
 - Kunde (Geschäftsobjekt)
 - Bestellung (Geschäftsobjekt)
 - Kundenverwaltung (komplexe Komponente)
 - Produktkatalog (komplexe Komponente)

→ *Es gibt keine optimale Granularität aus technischer Sicht!*

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

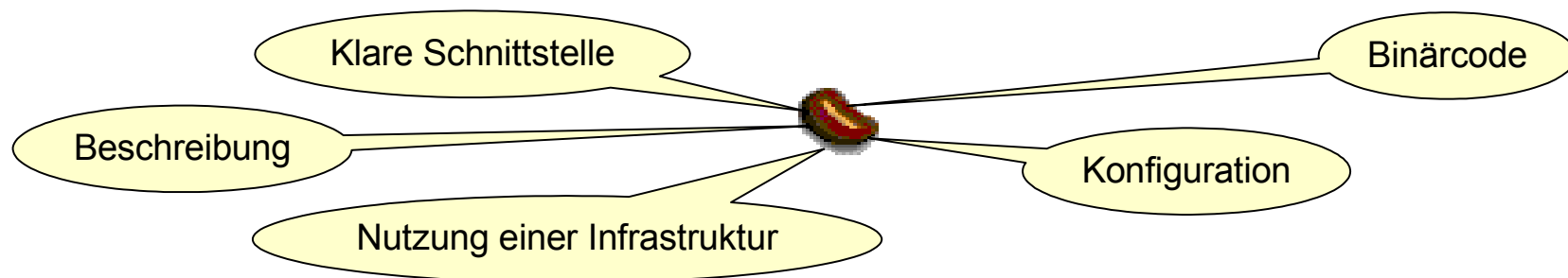
Schluss





Komponenten: Softwaretechnische Sicht

- Komponenten sind Softwaremodule, die auf **Binärcodeebene** wiederverwendbar sind.
- Komponenten sind **selbstbeschreibend** sowohl hinsichtlich ihrer exportierten als auch ihrer importierten Dienste.
- Komponenten sind möglichst umfassend **konfigurierbar**.
- Komponenten **implementieren Infrastrukturdienste nicht selbst**, sondern beziehen sie aus ihrer Einsatzumgebung.



Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





**Enterprise JavaBeans:
Der offene Standard
für Komponentenumgebungen**



Java 2 Enterprise Edition (J2EE) und die Enterprise JavaBeans (EJB)

- Java 2 Enterprise Edition:
 - „J2EE defines the standard for developing component-based multitier enterprise applications.“ [java.sun.com/j2ee/]
 - Entwickelt von Sun Microsystems (1999)
- Enterprise JavaBeans:
 - „...stellt in J2EE die zentrale Technologie zur Abbildung von Geschäftslogik und Geschäftsdaten dar.“ [Bachschat/Gardon]
 - aktuell ist Enterprise JavaBeans 3.0

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss



J2EE-Komponentenumgebung

Enterprise JavaBeans (EJB)

Java Database Connectivity (JDBC),
Container Managed Persistence (CMP)

Java Naming and Directory
Interface (JNDI)

EJB-Container

Namens-
dienst

Persistenzdienst



Datenbasis

Transaktionsmanager



Protokoll

Sicherheitsdienst

Object Request Broker

Java Transaction API (JTA)
Java Transaction Service (JTS)

Remote Method Invocation (RMI)
Java Messaging Service (JMS)

Java Authentication
and Authorization Service
(JAAS)



Wesentliche Inhalte des EJB-Standards

- EJB spezifiziert
 - einen Komponentenprozess mit diversen Rollen
 - die grundsätzlichen Arten von Komponenten
 - die Klientensicht von Komponenten
 - die Regeln für die Implementierung von Komponenten
 - den Lebenszyklus von Komponenten und die Interaktion zwischen Komponente und Container
 - die Beschreibung und Konfiguration von Komponenten
 - Anfragesprache zum Zugriff auf persistente Komponenten

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

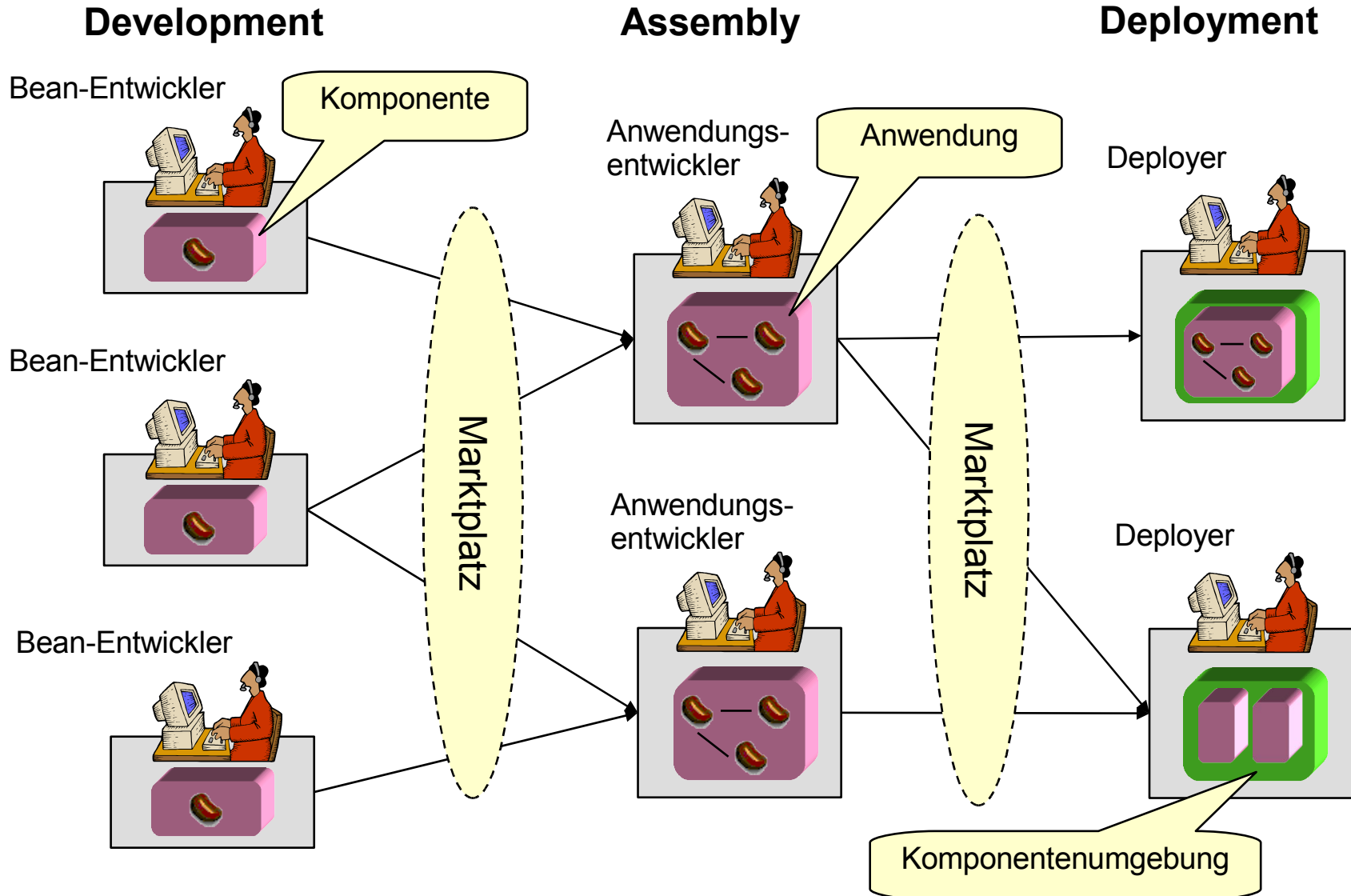
EJB

Tutorial

Schluss



EJB-Prozess





Komponentenarten in EJB

- **Entity Beans**
 - Datenobjekte
- **Session Beans**
 - Geschäftsmethoden
- **Message Driven Beans**
 - ereignisgetriebene Prozesse

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Entity Beans

- **Eigenschaften:**
 - Langlebige, von mehreren Klienten genutzte (Daten)Objekte
 - Bean- und Container-Managed Persistence möglich
 - Identifikation durch Objektreferenz und Primärschlüssel
- **Einsatzgebiet:**
 - Entity Beans modellieren Geschäftsobjekte: Kunden, Lieferanten, Produkte etc.
 - Sinnvoll zur objektorientierten Kapselung von Datensätzen

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Session Beans

- **Eigenschaften:**
 - Kurzlebige, nur von einem Klienten genutzte Objekte
 - Zustand existiert nur für die Dauer des Geschäftsprozesses
 - Identifikation nur durch Objektreferenz
- **Einsatzgebiet:**
 - Session Beans modellieren Geschäftsprozesse: Kaufvorgang, Buchung, Bonitätsprüfung etc.
 - Sinnvoll zur Kapselung von Geschäftslogik und Diensten

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Bestandteile einer Bean

- **Bean:** Das Bean-Objekt mit Zustandsdaten und den Implementierungen der Geschäftsmethoden
- **Home Interface:** definiert Methoden zum Erzeugen, Aufsuchen und Zerstören der Bean
- **Remote Interface:** definiert für Klienten nutzbare „Geschäftsmethoden“
- **Primary Key (*nur Entity Bean*):** ein serialisierbares Objekt; Datenbankschlüssel für die persistente Speicherung der Bean
- **Deployment Descriptor:** XML-Datei, die Struktur, Persistenzanforderungen, Zugriffsrechte, transaktionales Verhalten und referenzierte Namen der Bean beschreibt.

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Entity Beans: Beispiel

- Angenommen, klick-and-bau.com verwaltet Artikelstammdaten in der folgenden Tabelle:

```
ARTIKEL (  
    Id                CHAR (12) ,  
    Name              VARCHAR (25) ,  
    Beschreibung      VARCHAR (512) ,  
    Photo_URL         VARCHAR (512) ,  
    Hersteller_Id     CHAR (12)  
    Kategorien        VARCHAR (256) ,  
    Status            INT ,  
    Preis             NUMERIC (8 , 2)  
)
```

- Die Datensätze sollen nun als "Artikel"-Entity Beans gekapselt werden.

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Artikel-Bean: Remote Interface

```
package com.klick-and-bau;

import java.rmi.RemoteException;

public interface Artikel extends javax.ejb.EJBObject {

    public String getName() throws RemoteException;
    public void setName(String name) throws RemoteException;

    public String getBeschreibung() throws RemoteException;
    public void setBeschreibung(String beschr) throws
        RemoteException;

    ...

    public double getPreis() throws RemoteException;
    public void setPreis(double preis) throws
        RemoteException;

}
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Artikel-Bean: Home Interface

```
package com.klick-and-bau;

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

public interface ArtikelHome extends javax.ejb.EJBHome {

    public Artikel create(String id)
        throws RemoteException, CreateException;

    public Artikel findByPrimaryKey(ArtikelPK artikelPk)
        throws RemoteException, FinderException;

    // remove() Methoden werden von EJBHome geerbt
}

```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Artikel-Bean: Primärschlüsselklasse

```
package com.klick-and-bau;

public class ArtikelPK implements java.io.Serializable {

    // Schlüsselfelder - hier muss eine Untermenge der Felder
    // der Beanklasse stehen.

    public String id;

    // Spezielle Hash- und Vergleichsmethoden, da die Schlüsselwerte
    // und nicht die Java-Objekte gehasht bzw. verglichen werden sollen.

    public int hashCode() {
        return id.hashCode();
    }

    public boolean equals(Object obj) {
        return (obj instanceof ArtikelPK) && id.equals(((ArtikelPK)obj).id);
    }
}
```





Artikel-Bean: Beanklasse

```
package com.klick-and-bau;
```

```
public class ArtikelBean implements javax.ejb.EntityBean {
```

```
    // Zustandsvariablen - werden vom Container automatisch  
    // anhand der Informationen im Deployment Descriptor  
    // mit den entsprechenden Tabellenfeldern abgeglichen.
```

```
    public String id;  
    public String name;  
    public String beschreibung;  
    public String photoURL;  
    public String herstellerId;  
    public String kategorien;  
    public int    status;  
    public double preis;
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Artikel-Bean: Beanklasse

```
// Initialisierungsmethode - wird vom Container aufgerufen,  
// wenn Klient create()-Methode des Home Interface aufruft.  
// Für jede create()-Methode des Home Interface muss eine  
// entsprechende ejbCreate()-Methode der Beanklasse existieren.
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss

```
public ArtikelPK ejbCreate(String id) {  
    this.id = id;  
    // CMP: sonst nichts zu tun  
    // BMP: „INSERT INTO ARTIKEL VALUES (?, ?, ...)“  
    return new ArtikelPK(id);  
}
```

```
// Zusätzliche Initialisierungsmethode - wird vom Container  
// aufgerufen, nachdem Datensatz in der Datenbank angelegt  
// wurde.
```

```
public void ejbPostCreate(String id) {  
    // Dient vor allem zur Initialisierung von Bean-Referenzen.  
}
```





Artikel-Bean: Beanklasse

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss

```
// Implementierungen der Geschäftsmethoden.  
  
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
// etc.
```





Artikel-Bean: Beanklasse

```
// Lebenszyklusmethoden - werden vom Container aufgerufen.  
  
public void ejbLoad() {  
    // Zustandsvariablen werden aus der Datenbank gelesen.  
    // CMP: nichts zu tun.  
    // BMP: „SELECT * FROM ARTIKEL WHERE Id = ?“  
}  
  
public void ejbStore() {  
    // Zustandsvariablen werden in die Datenbank geschrieben.  
    // CMP: nichts zu tun.  
    // BMP: „UPDATE ARTIKEL SET Name = ? ... WHERE Id = ?“  
}  
  
public void ejbRemove() {  
    // Bean und Datensatz werden gelöscht.  
    // CMP: nichts zu tun.  
    // BMP: „DELETE FROM ARTIKEL WHERE Id = ?“  
}  
  
// Einige weitere Lebenszyklusmethoden sind nicht gezeigt.  
}
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Artikel-Bean: Deployment Descriptor

Einführung

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

Probleme

```
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise  
JavaBeans 2.0//EN" "http://java.sun.com/dtd/ejb-jar_2_0.dtd">
```

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Artikel-Bean: Deployment Descriptor

```
<ejb-jar>
  <enterprise-beans>
    <entity>
      <description>Bean für Artikel-Stammdaten</description>
      <ejb-name>Artikel</ejb-name>
      <home>com.klick-and-bau.ArtikelHome</home>
      <remote>com.klick-and-bau.Artikel</remote>
      <ejb-class>com.klick-and-bau.ArtikelBean</ejb-class>
      <persistence-type>Container</persistence-type>
      <prim-key-class>com.klick-and-bau.ArtikelPK</prim-key-class>
      <reentrant>False</reentrant>
      <cmp-field><field-name>id</field-name></cmp-field>
      <cmp-field><field-name>name</field-name></cmp-field>
      <cmp-field><field-name>beschreibung</field-name></cmp-field>
      <cmp-field><field-name>photoURL</field-name></cmp-field>
      <cmp-field><field-name>herstellerId</field-name></cmp-field>
      <cmp-field><field-name>kategorien</field-name></cmp-field>
      <cmp-field><field-name>status</field-name></cmp-field>
      <cmp-field><field-name>preis</field-name></cmp-field>
    </entity>
  </enterprise-beans>
```

Allgemeine
Angaben

Persistenz



Artikel-Bean: Deployment Descriptor

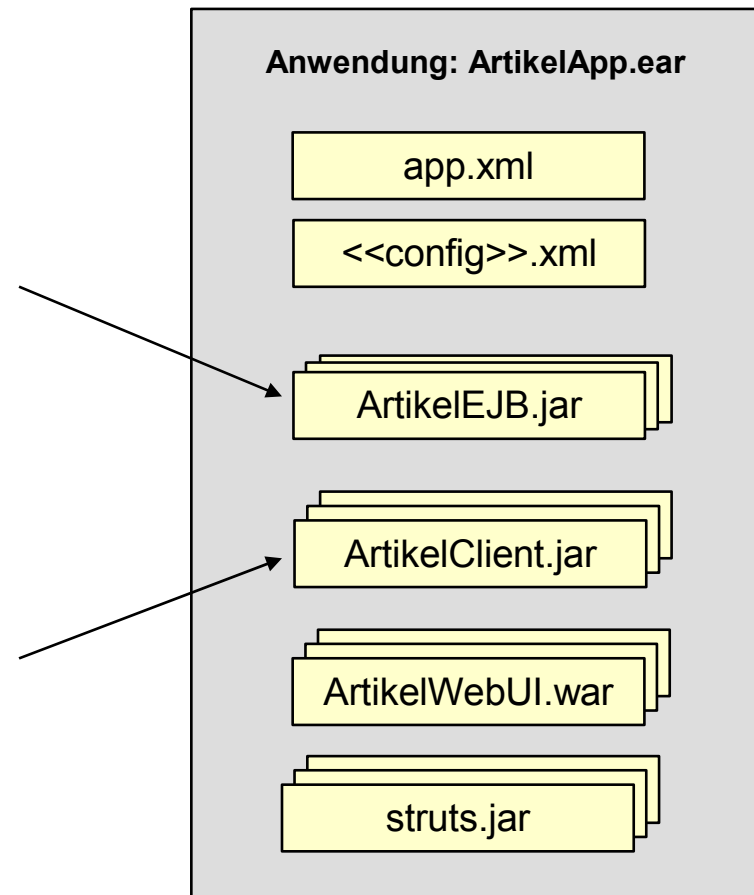
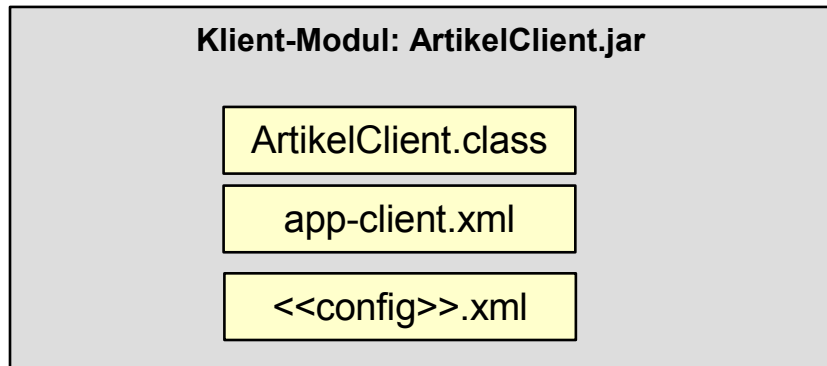
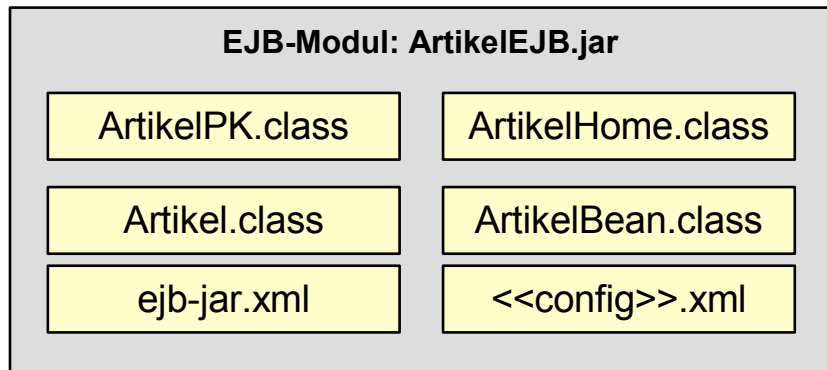
```
<assembly-descriptor>
  <security-role>
    <description>Benutzer mit Vollzugriff</description>
    <role-name>Vollzugriff</role-name>
  </security-role>
  <method-permission>
    <role-name>Vollzugriff</role-name>
    <method>
      <ejb-name>Artikel</ejb-name>
      <method-name>*</method-name>
    </method>
  </method-permission>
  <container-transaction>
    <method>
      <ejb-name>Artikel</ejb-name>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
  </container-transaction>
</assembly-descriptor>
</ejb-jar>
```

Sicherheit

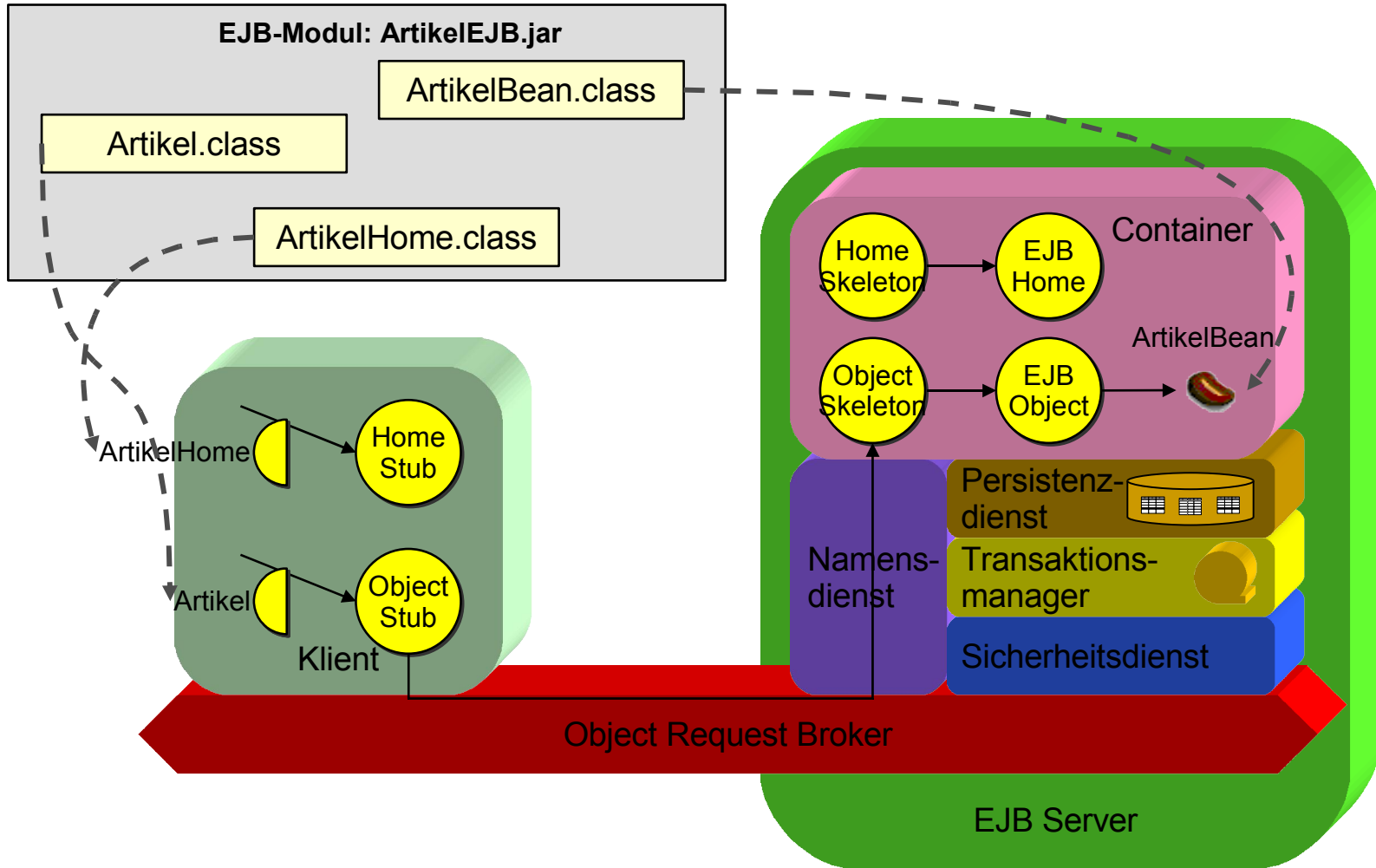
Transaktionales Verhalten



EJB-Archive

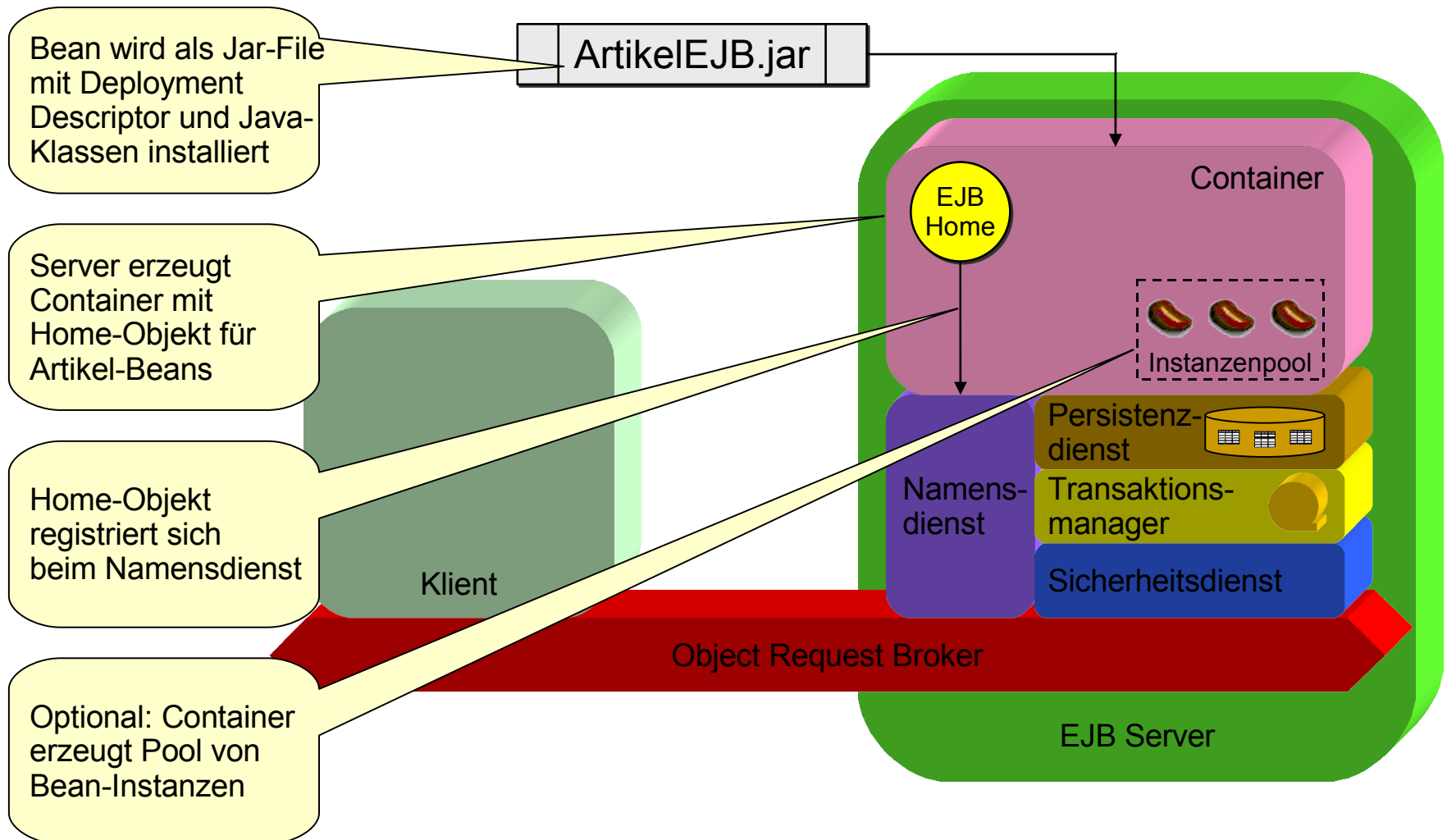


„Interception“



- Einführung
- Probleme
- Lösungsansätze
- Komponenten
- EJB
- Tutorial
- Schluss

Lebenszyklus einer Entity Bean: Installation





Lebenszyklus einer Entity Bean: Erzeugen einer Bean

```
package com.klick-and-bau;

import java.rmi.RemoteException;
...

public class ArtikelClient {

    public static void main(String args[]) throws Exception {

        // Eine Referenz auf ArtikelHome besorgen.
        InitialContext ctx = new InitialContext();
        Object artikelHomeRef = ctx.lookup(„ArtikelHomeInterface“);
        ArtikelHome artikelHome = (ArtikelHome)PortableRemoteObject.narrow(
            ref, ArtikelHome.class);

        // Einen Artikel erzeugen.
        Artikel artikel = artikelHome.create(„77“);
        ...
    }
}
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

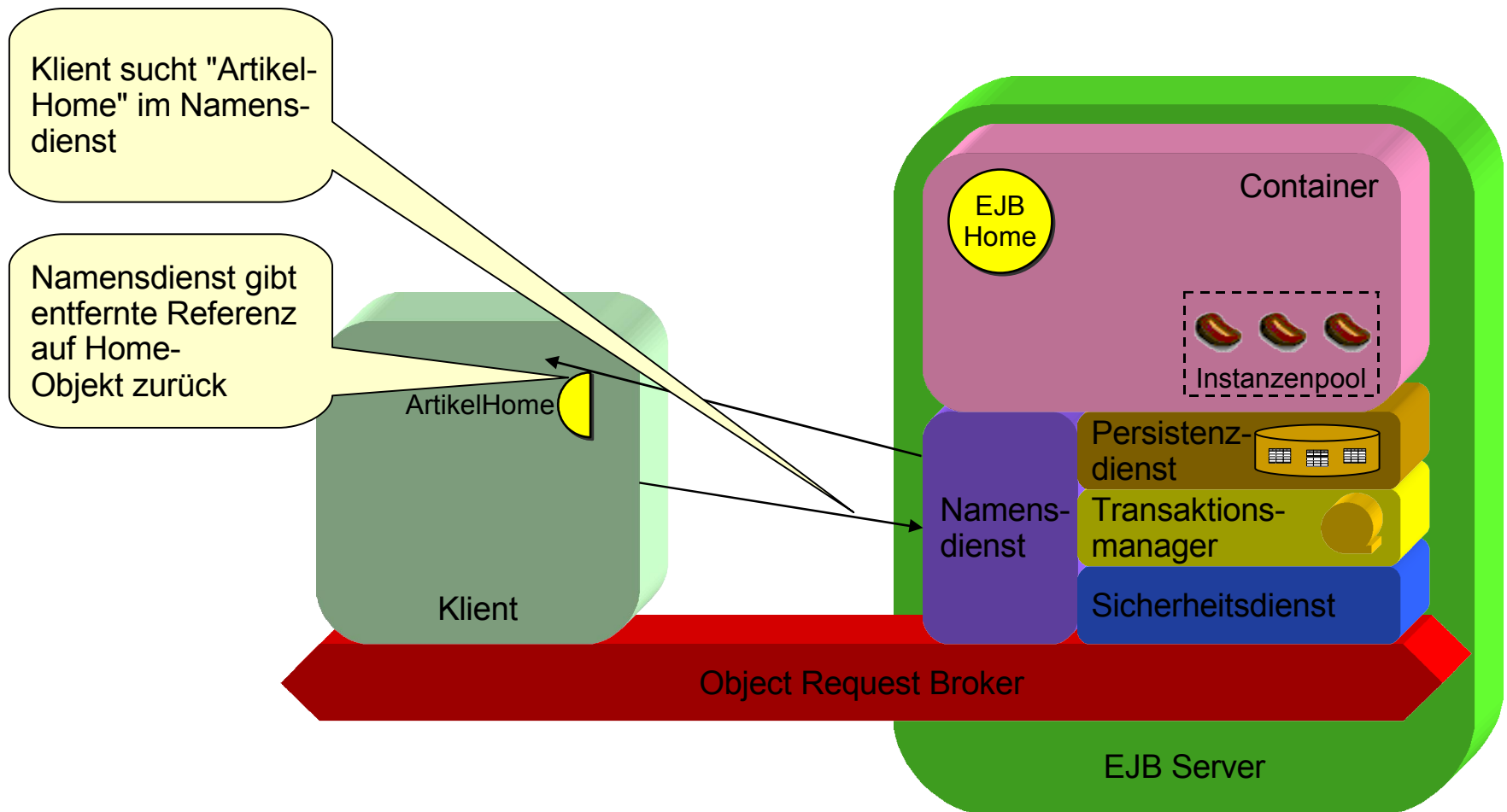
EJB

Tutorial

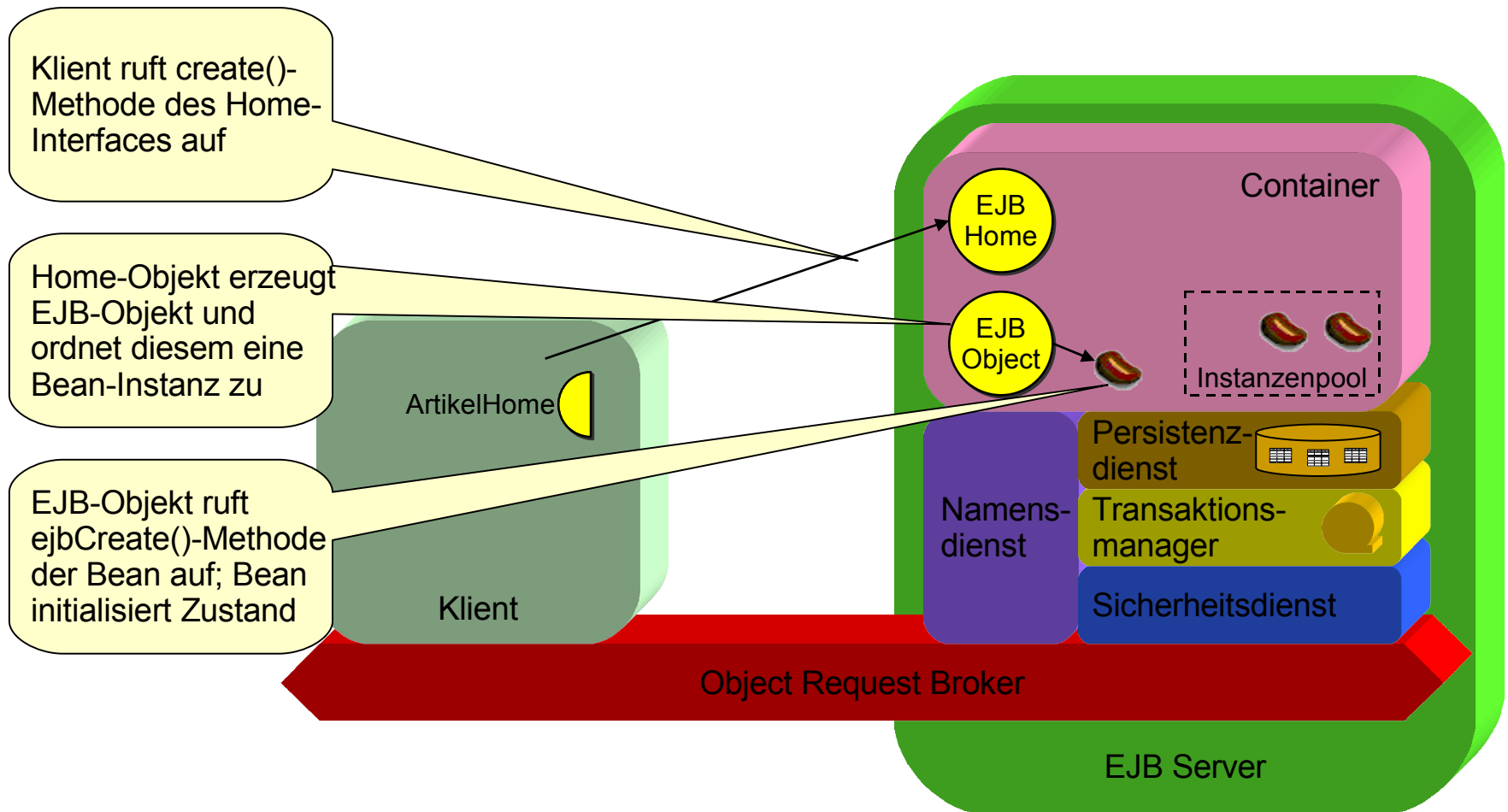
Schluss



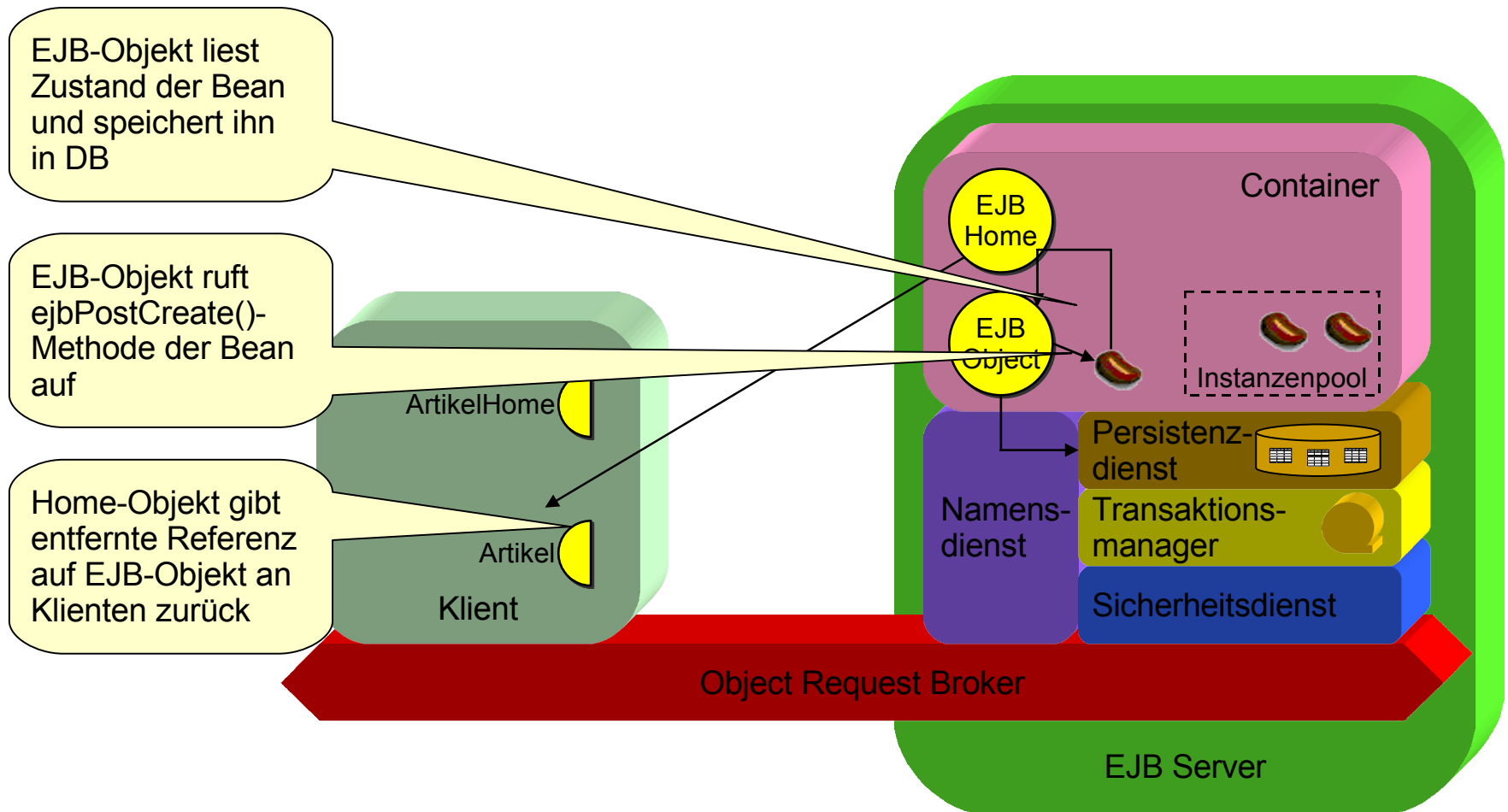
Lebenszyklus einer Entity Bean: Erzeugen einer Bean



Lebenszyklus einer Entity Bean: Erzeugen einer Bean



Lebenszyklus einer Entity Bean: Erzeugen einer Bean





Lebenszyklus einer Entity Bean: Aufruf einer Geschäftsmethode

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss

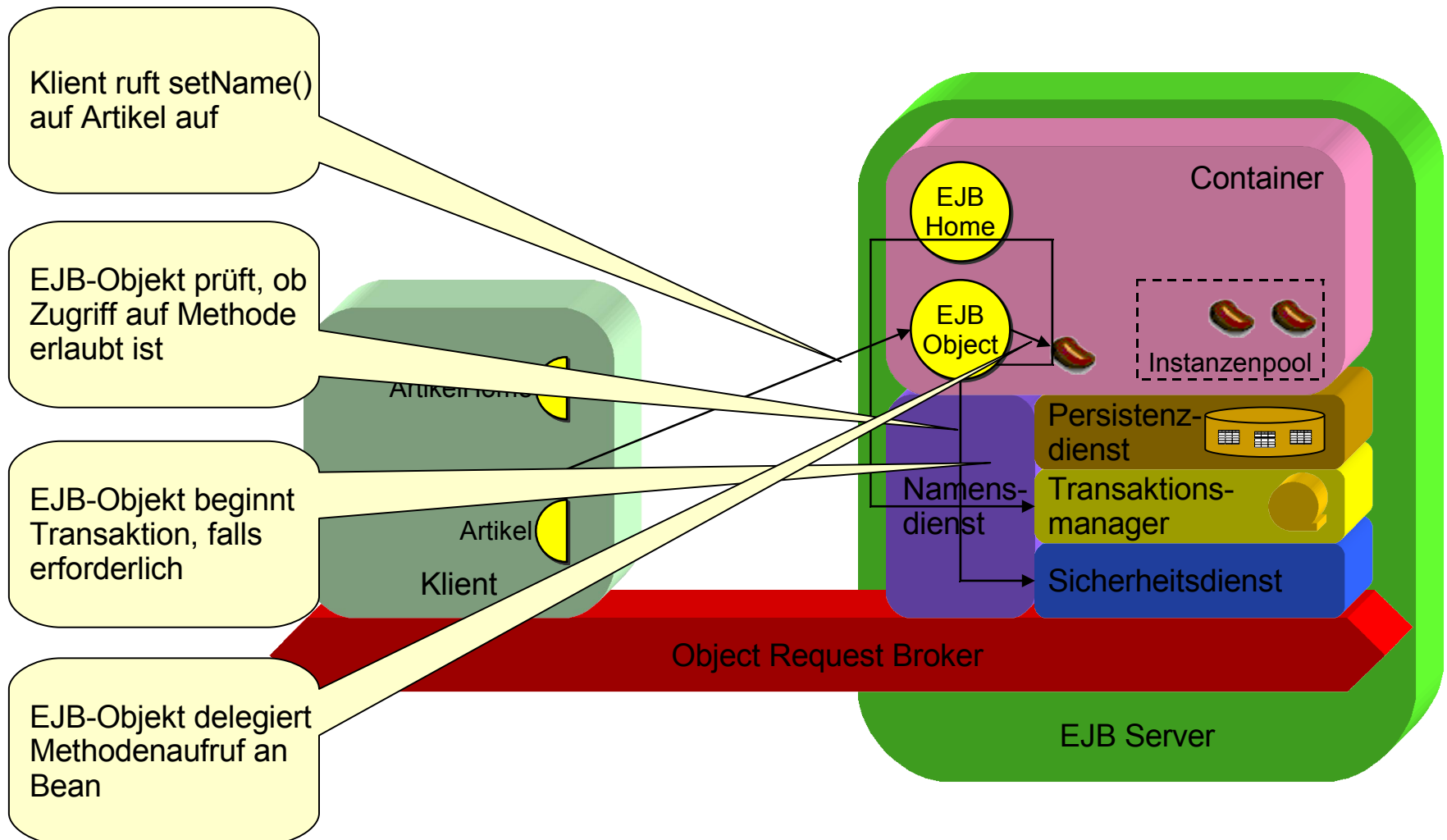
...

```
// Eine Geschäftsmethode aufrufen.  
String name = artikel.getName();
```

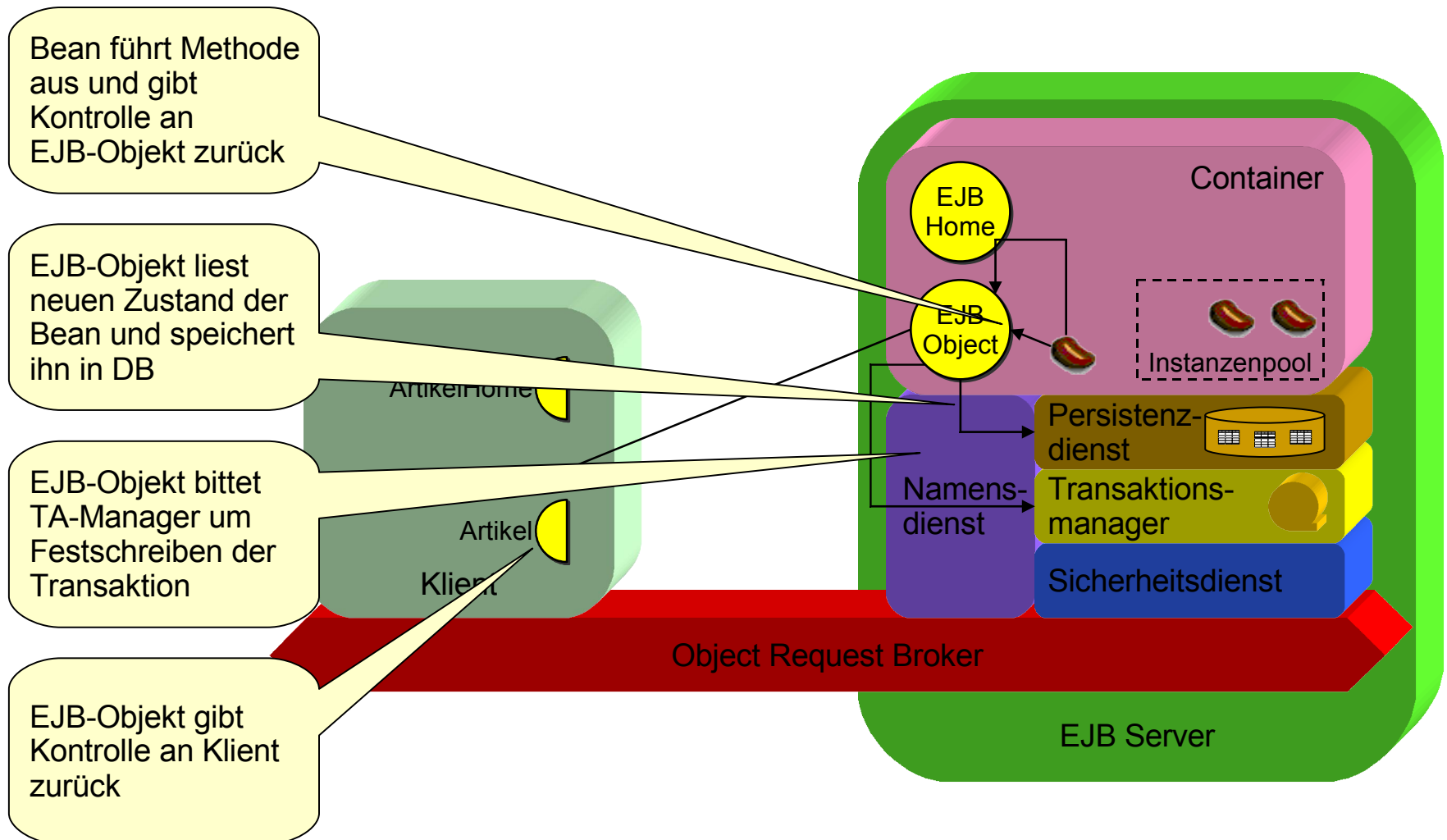
...



Lebenszyklus einer Entity Bean: Aufruf einer Geschäftsmethode



Lebenszyklus einer Entity Bean: Aufruf einer Geschäftsmethode





Lebenszyklus einer Entity Bean: Zerstören einer Bean

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

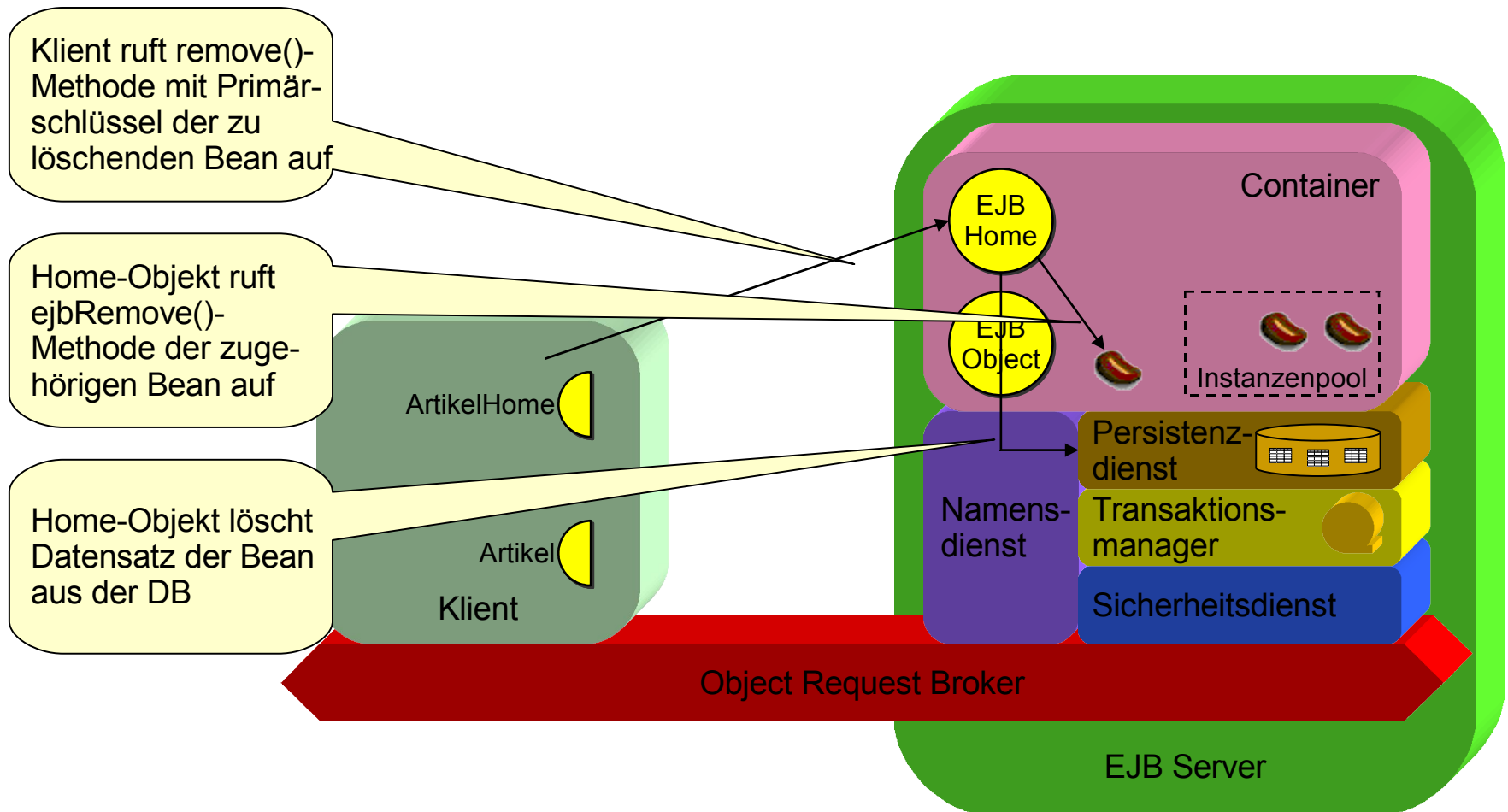
Tutorial

Schluss

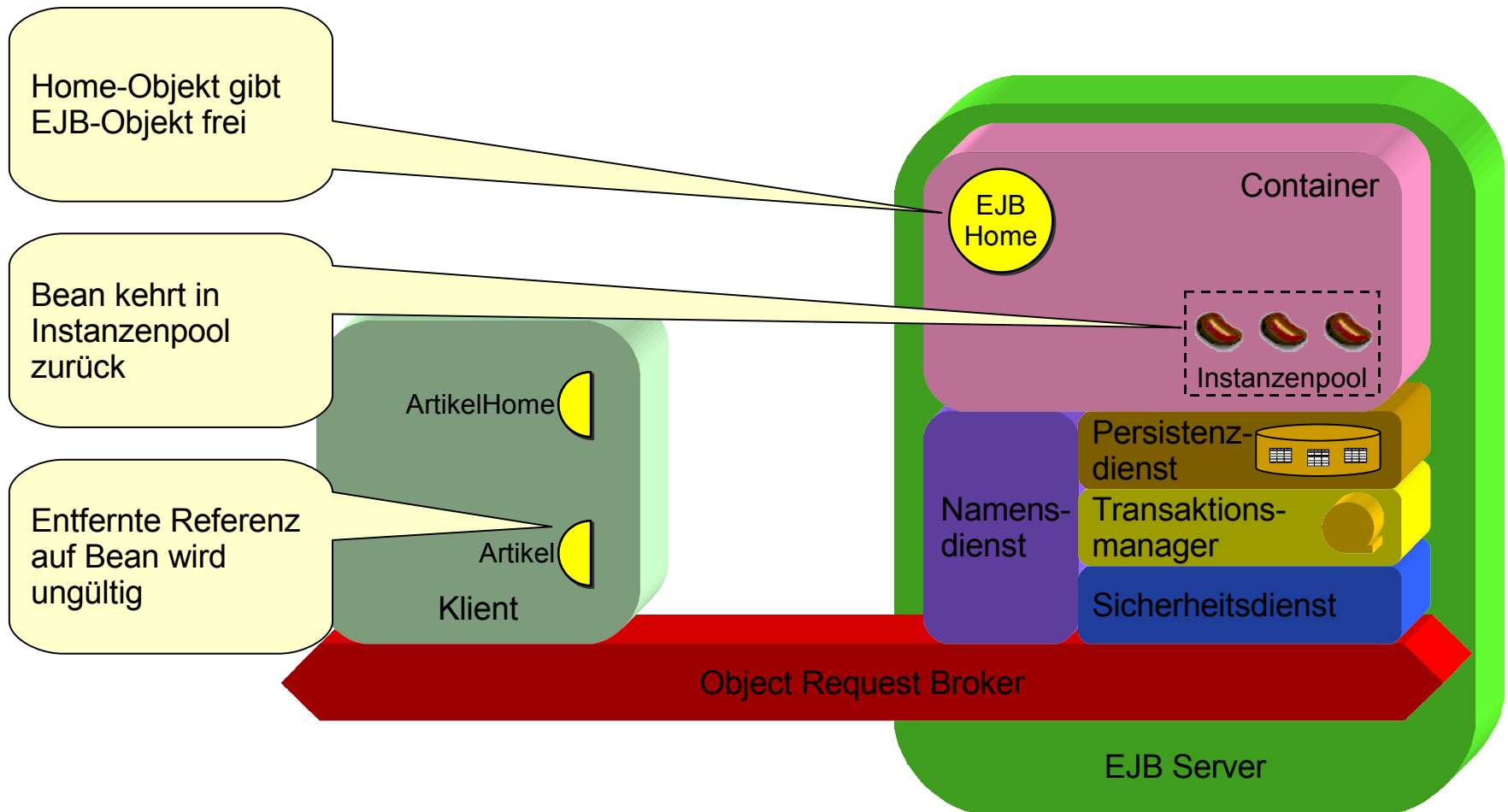
```
...  
  
// Den Artikel löschen.  
artikel.remove();  
  
}  
}
```



Lebenszyklus einer Entity Bean: Zerstören einer Bean



Lebenszyklus einer Entity Bean: Zerstören einer Bean





Session Beans: Beispiel

- Session Beans kommen in zwei Varianten:
 - **Stateless Session Beans** beschreiben Prozesse, die mit einem einzigen Methodenaufruf abgewickelt werden können.
 - **Stateful Session Beans** beschreiben Prozesse, die sich über mehrere Methodenaufrufe erstrecken ("Konversationen").
- **Beispiel: Adressprüfer**
 - Adressprüfer-Bean ist stateless und dient zur Prüfung von Adressen anhand der Postleitzahl
- **Beispiel: Kaufvorgang**
 - Die Kaufvorgang-Bean ist stateful. Sie implementiert Methoden zum Füllen des Warenkorbs, Festlegen der Zahlungsmodalität, Auslösen der Zahlung etc.

Einführung

Probleme

Lösungsansätze

Komponenten

EJB

Tutorial

Schluss





Adressprüfer-Bean: Prinzip

- Die folgende Postleitzahl-Tabelle stehe zur Verfügung:

```
POSTLEITZAHLEN (  
    Postleitzahl NUMERIC(5,0),  
    Stadt        VARCHAR(50),  
    Strasse      VARCHAR(100),  
    Min_Nummer  NUMERIC(5,0) DEFAULT 0,  
    Max_Nummer  NUMERIC(5,0) DEFAULT 99999  
)
```

- Zur (naiven) Prüfung einer Adresse (plz, std, str, num) wird geschaut, ob ein passender Datensatz existiert:

```
select * from POSTLEITZAHLEN  
where Postleitzahl = plz  
and Stadt = std and Strasse = str  
and Min_Nummer <= num and Max_Nummer >= num
```

Einführung
Probleme
Lösungs-
ansätze
Kompo-
nenten
[EJB](#)
Tutorial
Schluss





Adressprüfer-Bean: Remote Interface

```
package com.klick-and-bau;  
  
import java.rmi.RemoteException;  
  
public interface Adressprüfer extends javax.ejb.EJBObject {  
    public boolean gültig(int plz, String std, String str, int num)  
        throws RemoteException;  
}
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Adressprüfer-Bean: Home Interface

```
package com.klick-and-bau;

import java.rmi.RemoteException;
import javax.ejb.CreateException;

public interface AdressprüferHome extends javax.ejb.EJBHome {

    public Adressprüfer create() throws RemoteException,
        CreateException;

    // find()-Methoden gibt es für Session Beans nicht,
    // remove()-Methoden werden von EJBHome geerbt.
}
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Adressprüfer-Bean: Beanklasse

```
package com.klick-and-bau;

import java.sql.*;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;
import javax.ejb.EJBException;

public class AdressprüferBean implements javax.ejb.SessionBean {

    // Zustandsvariablen (auch in stateless Beans erlaubt, solange
    // nicht versucht wird, klientenspezifischen Zustand zu halten)

    // JDBC-Verbindung zur PLZ-Datenbasis
    Connection con = null;

    // Vorbereitete SQL-Suchanfrage
    PreparedStatement ps = null;
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Adressprüfer-Bean: Beanklasse (Forts.)

```
// Initialisierungsmethode - wird vom Container irgendwann zwischen
// dem Anlegen der Instanz und dem Aufruf der ersten Geschäftsmethode
// aufgerufen.
// Hier: Öffne Verbindung zu PLZ-Datenbasis und bereite SQL-Anfrage vor.

public void ejbCreate() {
    try {
        InitialContext ctx = new InitialContext();
        DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/plzDB");
        con = ds.getConnection();
    } catch (NamingException ne) {
        throw new EJBException(ne);
    }
    ps = con.prepareStatement(
        "select * from POSTLEITZAHLEN"
        + "where Postleitzahl = ?"
        + "and Stadt = ? and Strasse = ?"
        + "and Min_Nummer <= ? and Max_Nummer >= ?"
    );
}
```



Adressprüfer-Bean: Beanklasse (Forts.)

```
// Implementierung der Geschäftsmethode.  
  
public boolean gültig(int plz, String std, String str, int num) {  
    try {  
        ps.setInt(1,plz);  
        ps.setString(2,std);  
        ps.setString(3,str);  
        ps.setInt(4,num);  
        ps.setInt(5,num);  
        ResultSet res = ps.executeQuery();  
        // res.next() gibt false zurück, wenn kein Datensatz vorh.  
        return res.next();  
    } catch (SQLException se) {  
        throw new EJBException(se);  
    }  
}
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Adressprüfer-Bean: Beanklasse (Forts.)

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss

```
// Lebenszyklusmethoden - werden vom Container aufgerufen.  
  
public void ejbRemove() {  
    // Bean wird gleich gelöscht - gebe Ressourcen frei.  
    try {  
        if (ps != null) ps.close();  
        if (con != null) con.close();  
    } catch (SQLException se) {  
    }  
}  
  
// Einige weitere Lebenszyklusmethoden sind nicht gezeigt.  
}
```



Adressprüfer-Bean: Deployment Descriptor

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise
JavaBeans 1.1//EN" "http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">
<ejb-jar>
  <enterprise-beans>
    <session>
      <description>Bean zur Adressprüfung</description>
      <ejb-name>Adressprüfer</ejb-name>
      <home>com.klick-and-bau.AdressprüferHome</home>
      <remote>com.klick-and-bau.Adressprüfer</remote>
      <ejb-class>com.klick-and-bau.AdressprüferBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
      <resource-ref>
        <description>Postleitzahl-Datenbasis</description>
        <res-ref-name>jdbc/plzDB</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
      </resource-ref>
    </session>
  </enterprise-beans>
```

Allgemeine
Angaben

Benötigte
Ressourcen





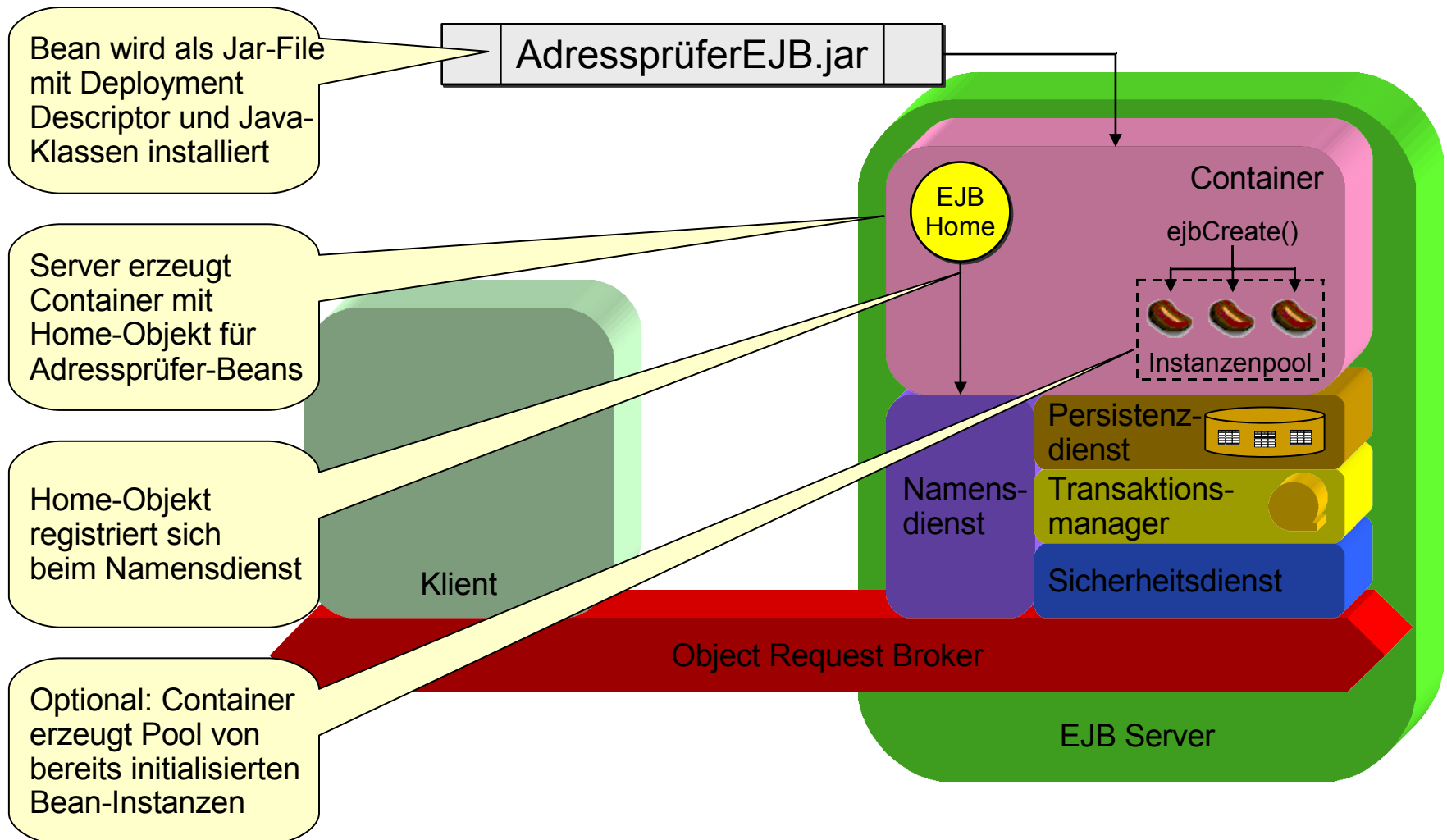
Adressprüfer-Bean: Deployment Descriptor (Forts.)

```
<assembly-descriptor>
  <security-role>
    <description>Benutzer mit Vollzugriff</description>
    <role-name>Vollzugriff</role-name>
  </security-role>
  <method-permission>
    <role-name>Vollzugriff</role-name>
    <method>
      <ejb-name>Adressprüfer</ejb-name>
      <method-name>*</method-name>
    </method>
  </method-permission>
  <container-transaction>
    <method>
      <ejb-name>Adressprüfer</ejb-name>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
  </container-transaction>
</assembly-descriptor>
</ejb-jar>
```

Sicherheit

Transaktionales Verhalten

Lebenszyklus einer Stateless Sess. Bean: Installation





Lebenszyklus einer Stateless Sess. Bean: Erzeugen einer Bean

```
package com.klick-and-bau;
```

```
import java.rmi.RemoteException;
```

```
...
```

```
public class AdressprüferClient {
```

```
    public static void main(String args[]) throws Exception {
```

```
        // Eine Referenz auf AdressprüferHome besorgen.
```

```
        InitialContext ctx = new InitialContext();
```

```
        Object prüferHomeRef = ctx.lookup(„AdressprüferHomeInterface“);
```

```
        AdressprüferHome prüferHome =
```

```
            (AdressprüferHome) PortableRemoteObject.narrow(ref, ArtikelHome.class);
```

```
        // Einen Adressprüfer erzeugen.
```

```
        Adressprüfer prüfer = prüferHome.create();
```

```
        ...
```

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

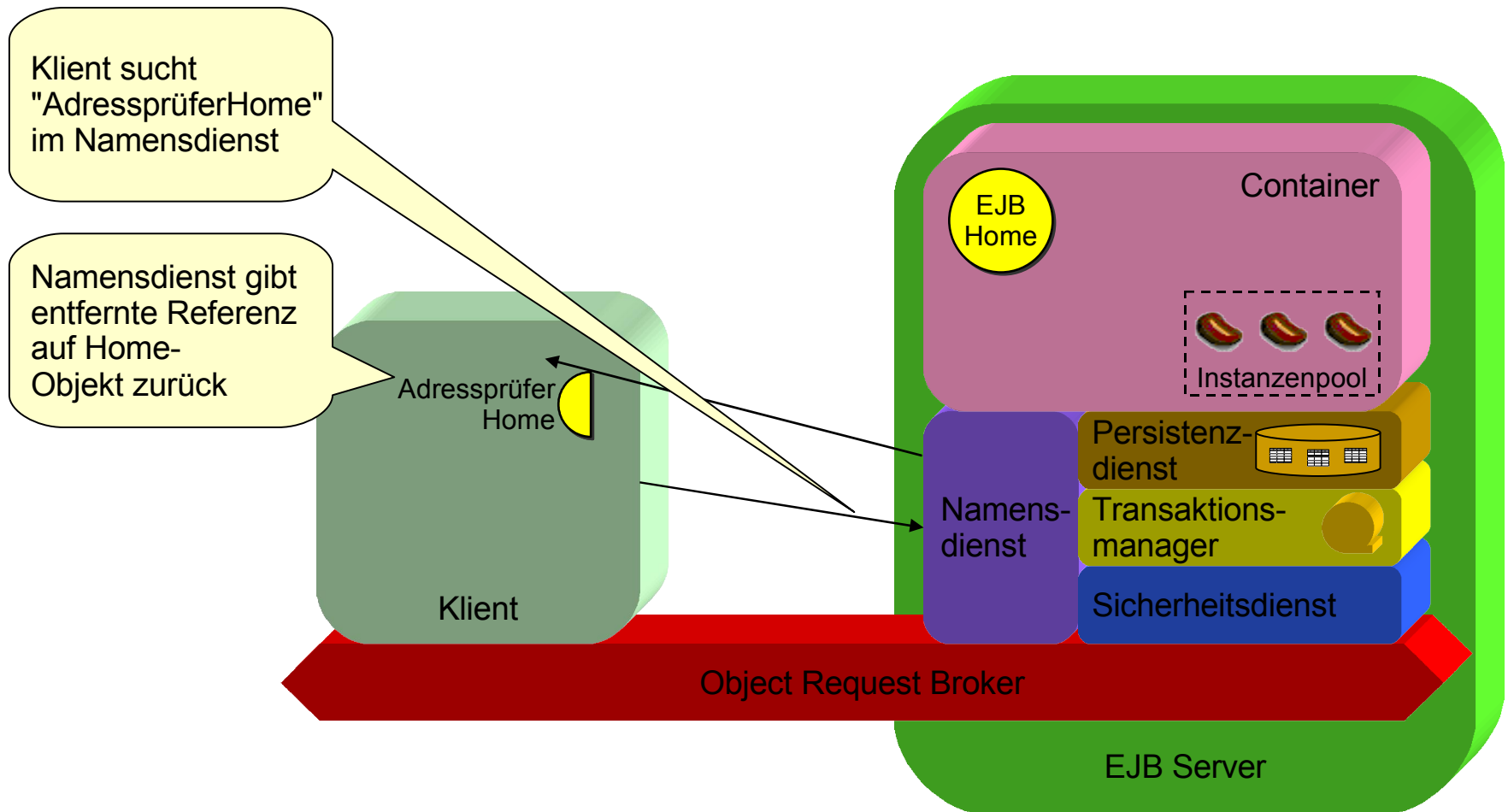
EJB

Tutorial

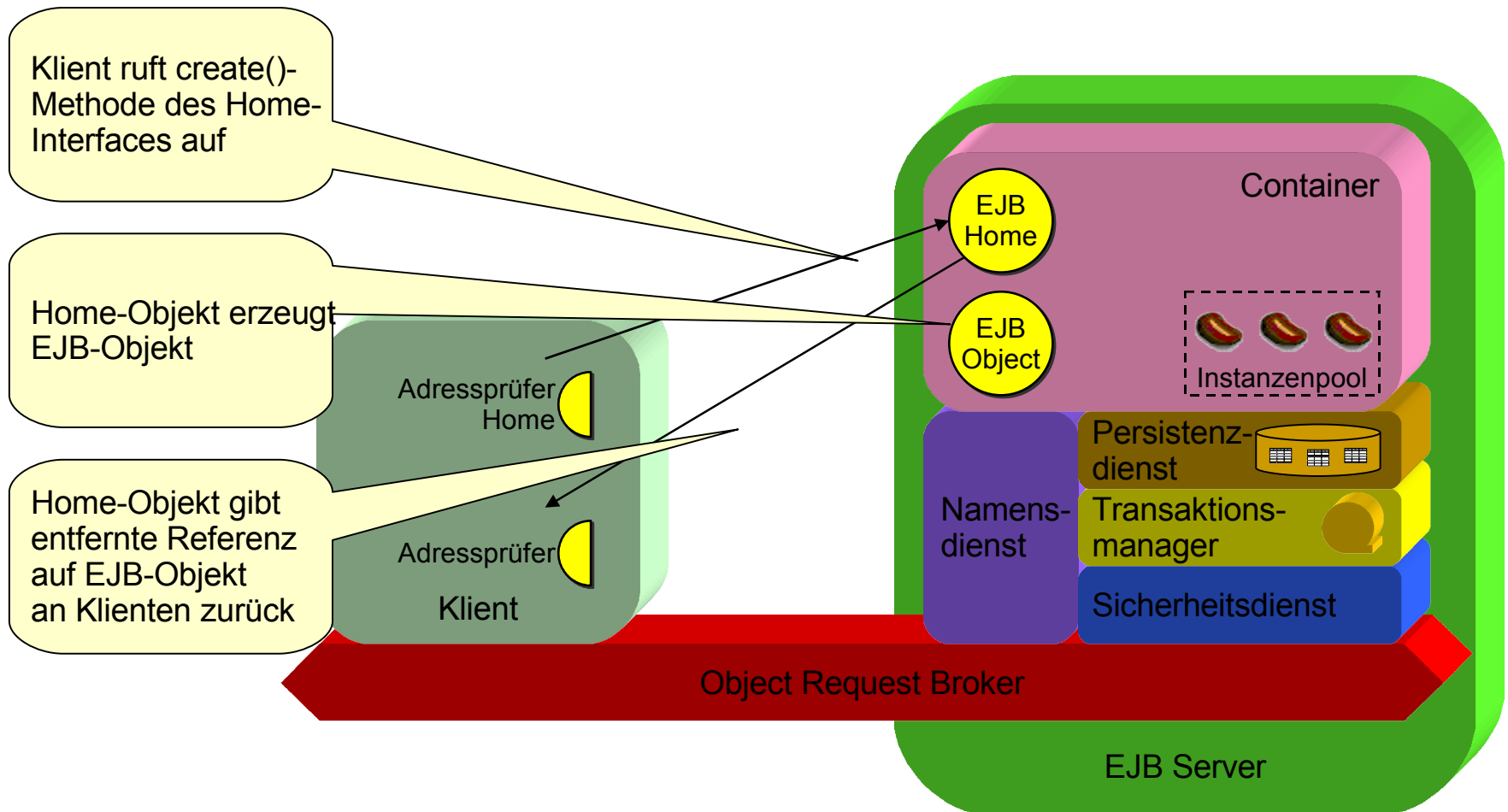
Schluss



Lebenszyklus einer Stateless Sess. Bean: Erzeugen einer Bean



Lebenszyklus einer Stateless Sess. Bean: Erzeugen einer Bean (Forts.)





Lebenszyklus einer Stateless Sess. Bean: Aufruf einer Geschäftsmethode

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss

...

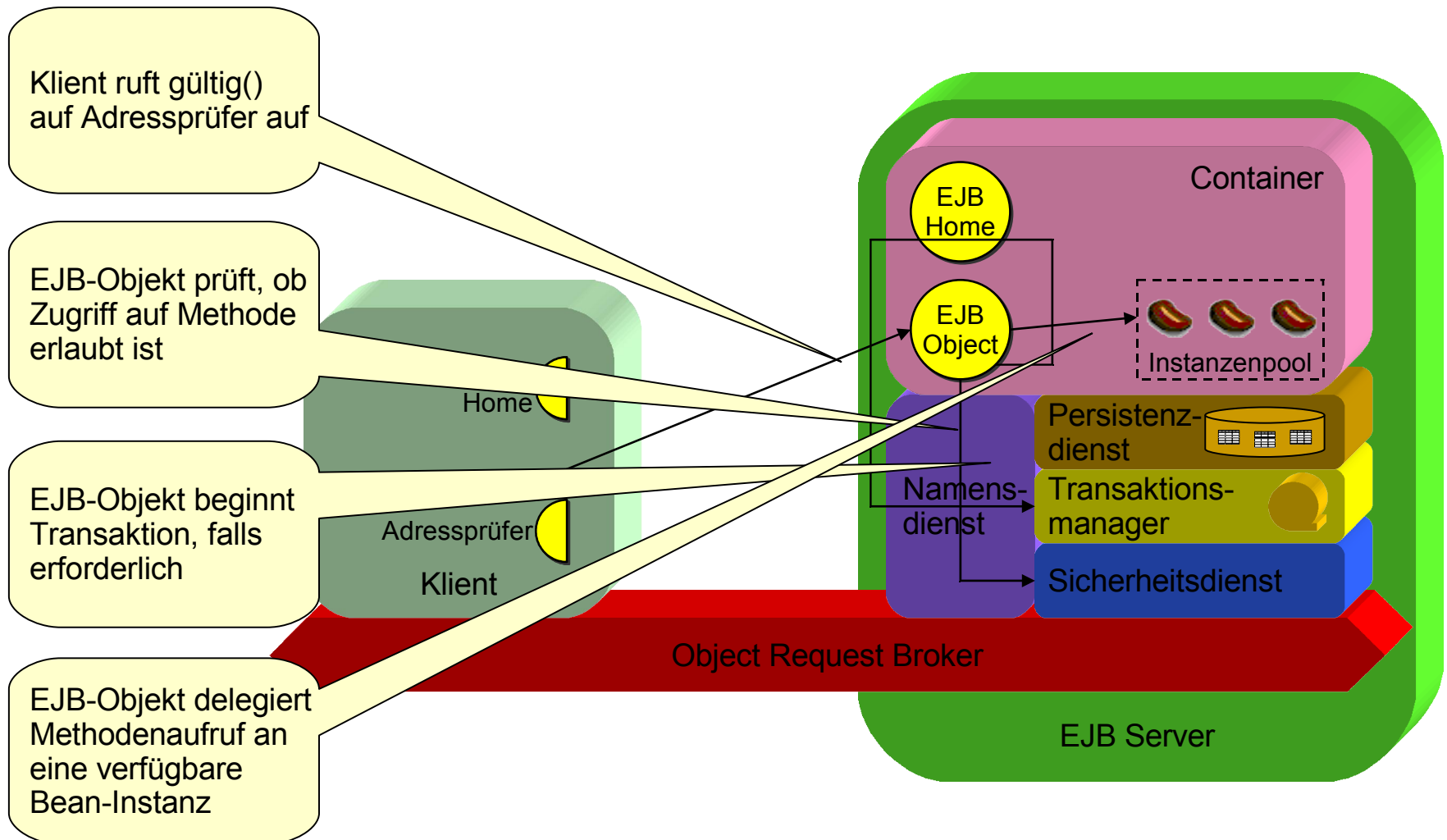
```
// Eine Adresse auf Gültigkeit prüfen.
```

```
boolean adresseIstOk = pruefer.gueltig(76131, „Karlsruhe“,  
    „Haid-und-Neu-Straße“, 10);
```

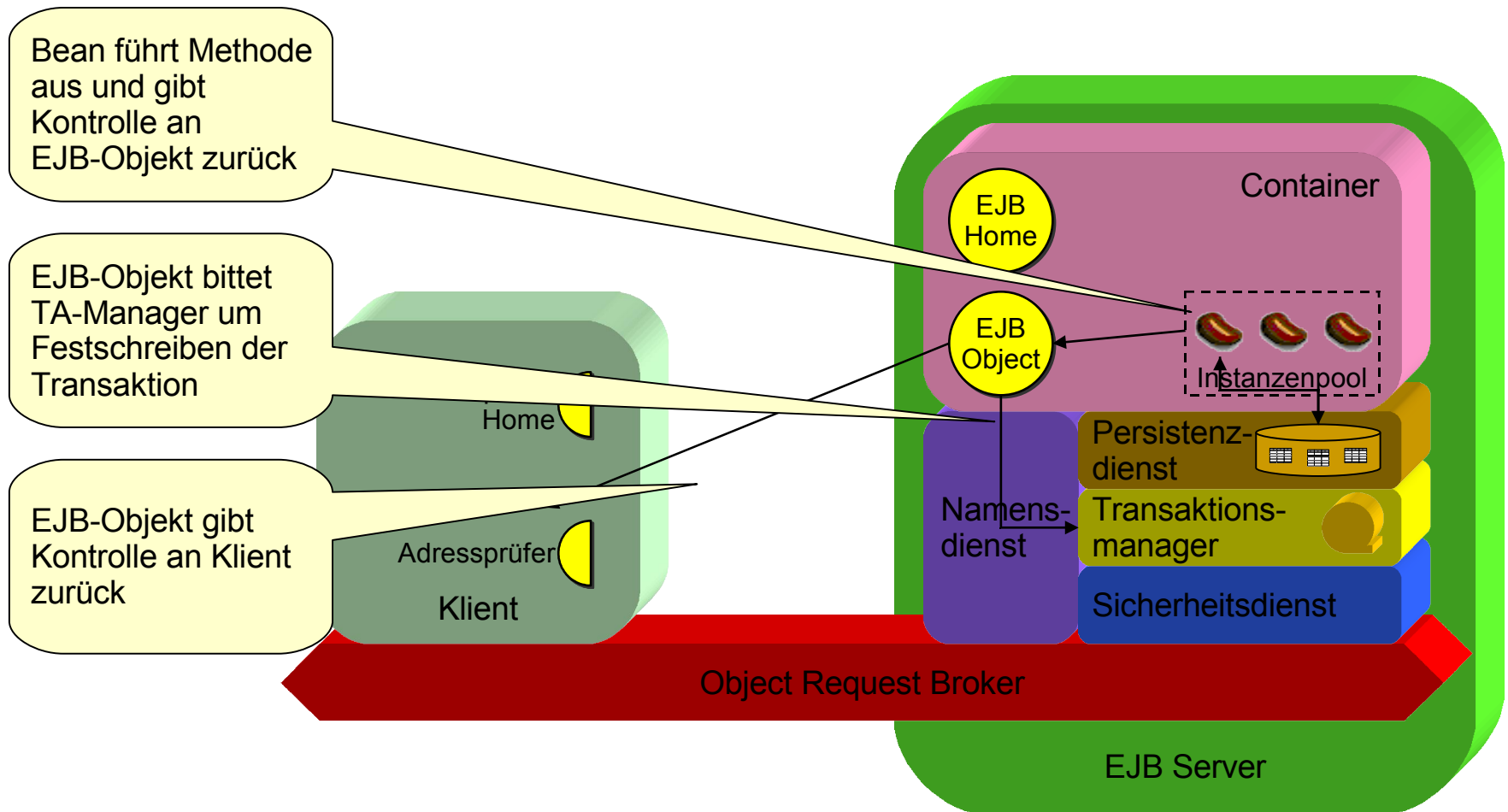
...



Lebenszyklus einer Stateless Sess. Bean: Aufruf einer Geschäftsmethode



Lebenszyklus einer Stateless Sess. Bean: Aufruf einer Geschäftsmethode (Forts.)





Lebenszyklus einer Stateless Sess. Bean: Zerstören einer Bean

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

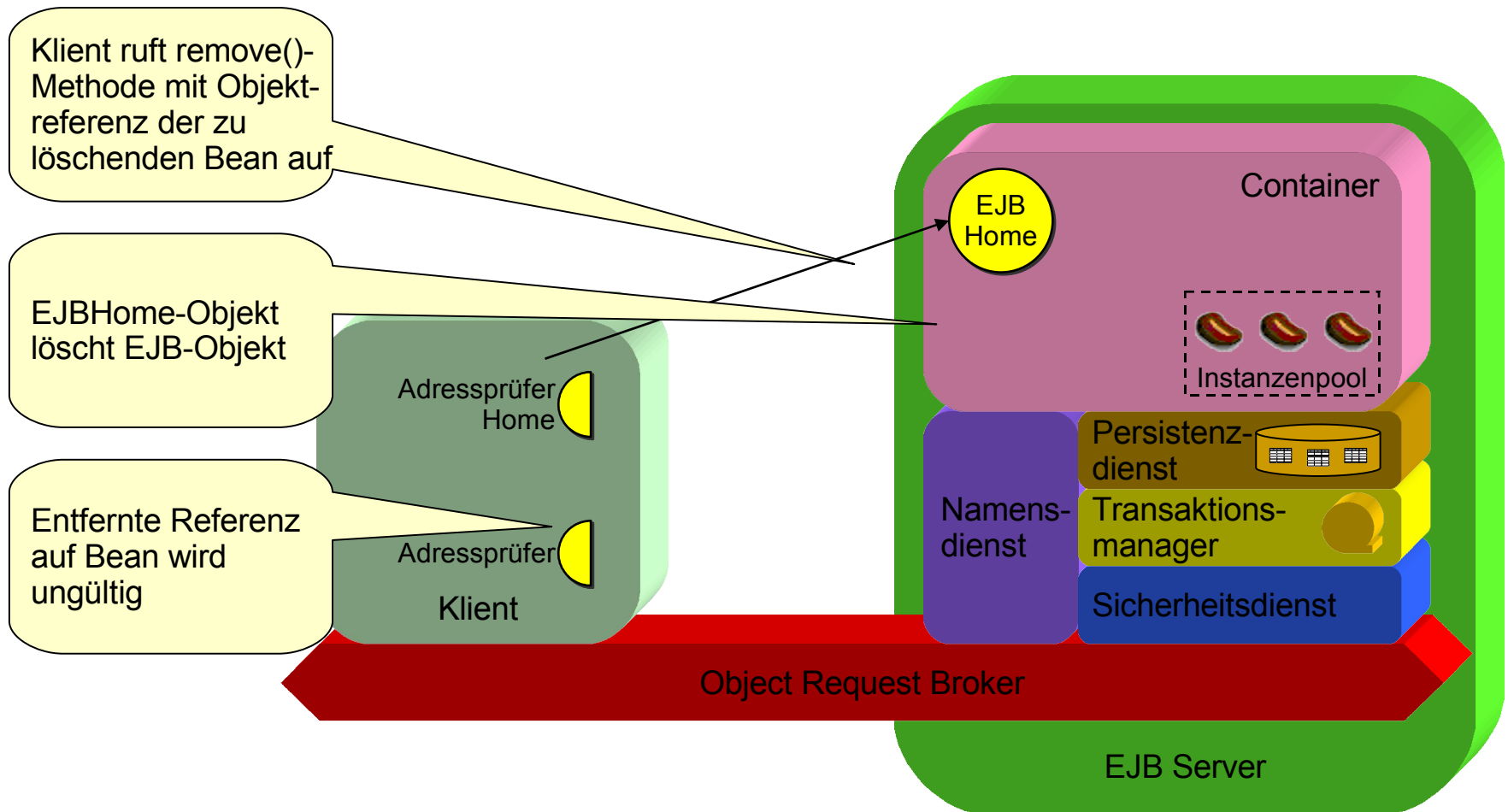
Tutorial

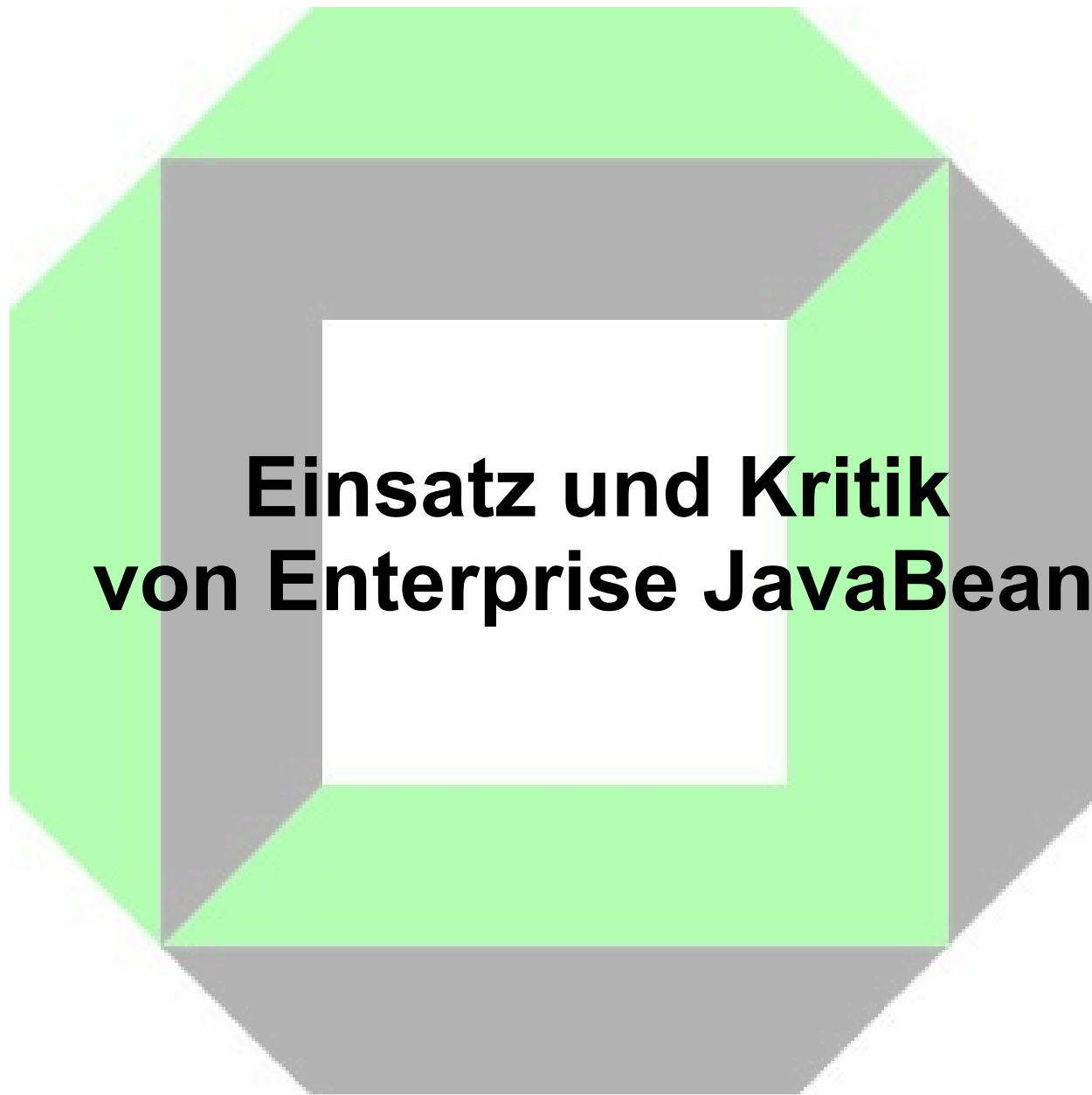
Schluss

```
...  
  
// Den Adressprüfer löschen.  
prüfer.remove();  
  
}  
}
```



Lebenszyklus einer Stateless Sess. Bean: Zerstören einer Bean





Einsatz und Kritik von Enterprise JavaBeans



Vorteile von EJB

- Bewährte Vorgehensweisen wurden öffentlich wiederverwendbar gemacht
- EJB-Server bieten eine fertige Infrastruktur für umfangreiche, verteilte Anwendungen
- Know-how ist „portabel“ geworden:
 - leichter Umgang mit mehreren Plattformen
 - stärkerer Erfahrungsaustausch unter den Entwicklern
 - erhöhtes Verständnis fremder Software
- Deutlich vereinfachte Portierung von Anwendungen

Einführung

Probleme

Lösungsansätze

Komponenten

EJB

Tutorial

Schluss





Nachteile von EJB

- Hoher Lernaufwand
- Aufwändige Entwicklung
- Schlechte Handhabung und Fehleranfälligkeit
 - Deployment Descriptoren, Testen, Debugging
- Konzeptuelle Schwächen
 - Technische und fachliche Aspekte z.T. stark verzahnt
 - Warum braucht man sonst so viele J2EE-Patterns?
- Begrenzte Integrierbarkeit von und in bestehende IT-Landschaften
 - siehe später: Dienstorientierte Architektur und WebServices

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Nachteile im Detail

- Immer wieder genannte Probleme:
 - Implementierung vieler Schnittstellen notwendig
 - Viele überflüssige Code-Passagen sind zu schreiben
 - Deployment Descriptoren sind komplex und fehleranfällig
 - CMP ist kompliziert aber nicht mächtig genug (-> EJBQL)
 - Der Ansatz ist nicht objektorientiert
 - Testen und Debuggen von EJBs ist ein Alptraum
 - Komplizierter Zugriff auf Komponenten per JNDI

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Lösungen: Effiziente Kommunikation

- Problem:
 - Bis EJB 2.0 nur entfernter Zugriff auf EJBs
→ Feingranulare Zugriffe ineffizient (getter/setter)
- Lösungsansatz:
 - Einführung lokaler Schnittstellen in EJB 2.0 zur Kommunikation innerhalb des Servers
 - `javax.ejb.EJBLocalObject`, `javax.ejb.EJBLocalHome`
 - J2EE-Muster „Facade“ und „TransferObject“ für den Datenaustausch zwischen Server und Klient
- Nebenwirkungen:
 - Unerwünschte Kopplung von fachlichen und technischen Aspekten
 - Ein nochmals erhöhter Implementierungsaufwand

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

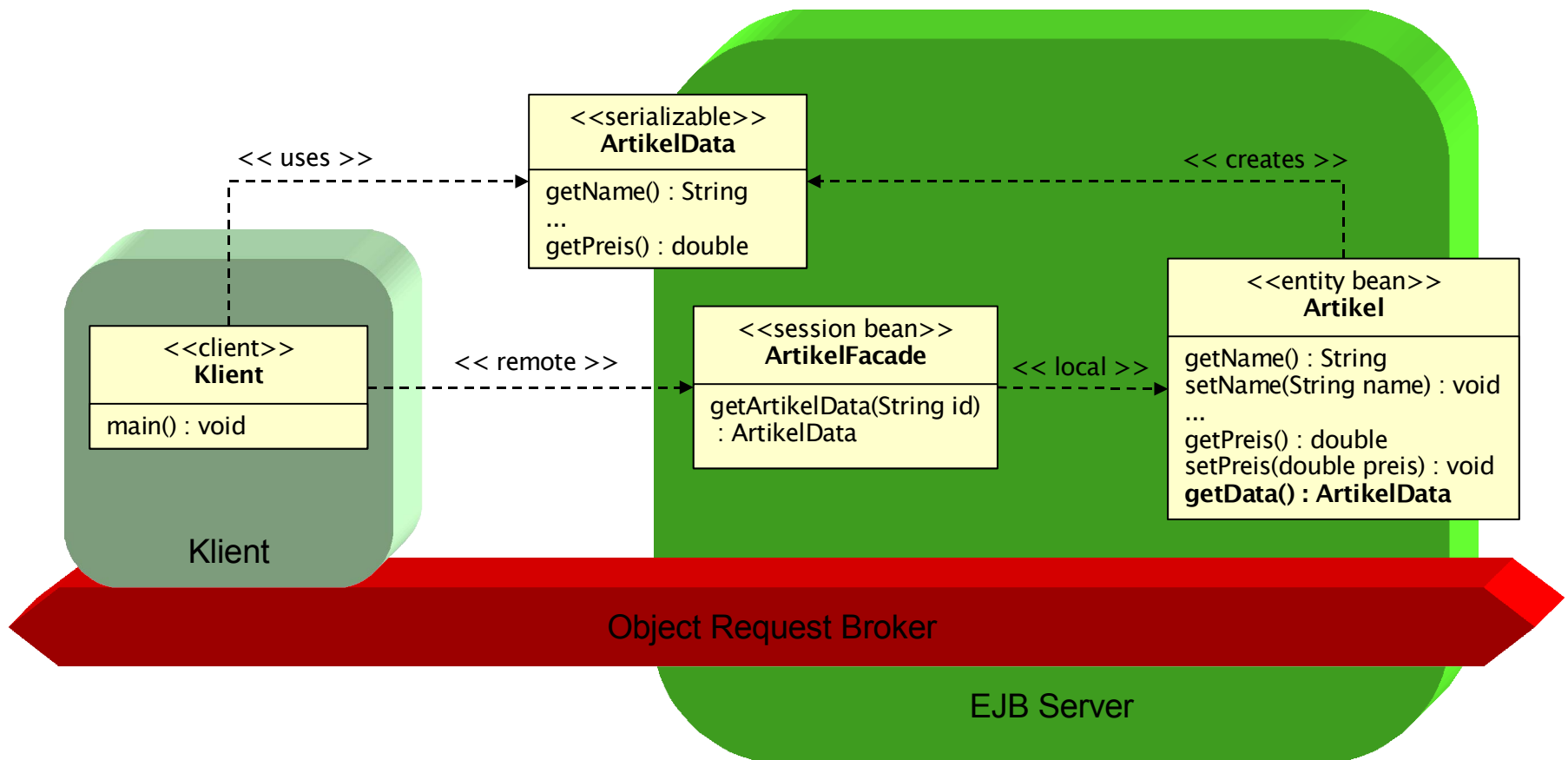
EJB

Tutorial

Schluss



Muster „Facade“ und „TransferObject“





Lösungen: Aufwandsreduktion

- Problem:
 - Redundante/überflüssige Code-Passagen sind zu schreiben
 - Komplexe und fehleranfällige Elemente (→ DD)
- Lösungsansätze:
 - xdoclet – Generierung von Schnittstellen und DD aus einer annotierten Bean-Implementierung

```
@Remote @Stateless public class HelloWorldBean {  
    public String sayHello(String s) {  
        System.out.println("Hello:" + s);  
    }  
}
```

- Model-Driven Architecture (MDA) – Generierung kompletter EJBs aus einem annotierten UML-Klassendiagramm

Einführung

Probleme

Lösungsansätze

Komponenten

EJB

Tutorial

Schluss





Zusammenfassung

- Unterstützung moderner Geschäftsprozesse ist eine sehr komplexe und kostspielige Angelegenheit
- Grundlegende Lösungsansätze
 - Mehrschichtenarchitekturen
 - Modularisierung
 - „Separation of Concerns“
 - Standardisierung
- Komponentenumgebungen sind ein Versuch, diese Ansätze in ein ganzheitliches Rahmenwerk zu gießen
- J2EE und Enterprise JavaBeans sind Java-basierte Standards für Komponentenumgebungen

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss





Literatur

- Bücher:

- Backschat, M., Gardon, O. (2002): *Enterprise JavaBeans*. Spektrum Akademischer Verlag.
- Alur, D., Crupi, J., Malks, D. (2002): *Core J2EE Patterns*. Prentice Hall.
- Bien, A. (2003): *J2EE Patterns*. Addison-Wesley.
- Johnson, R. (2004): *J2EE Development without EJB*. Wiley Publishing.

- News und Artikel im Web:

- www.theserverside.com, www.ejbsig.de
- www.javamagazin.de

- Zeitschriften:

- JavaMagazin, JavaSpektrum
- Bien, A.: *J2EE Patterns*. In: JavaMagazin 11/2002-08/2003

Einführung

Probleme

Lösungs-
ansätze

Kompo-
nenten

EJB

Tutorial

Schluss

