Vorlesung
Software aus Komponenten

3. Komponentenmodelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2011/12

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Standardkomponenten (CORBAfacilities)

Standardisierung von häufig benötigten Anwendungsbestandteilen

- Komponentenrahmen zur einfachen Integration von Anbieterlösungen

Abgrenzung von Bereichen horizontal oder vertikal

- horizontal: Fokus auf generellem Anwendungsmodell
  - Standards für Nutzerschnittstellen, System- und Aufgabenverwaltung
- vertikal: Focus auf bereichsspezifischen Einsatzfeldern
  - im Rahmen von OMG SIG‘s oder Domain Task Forces

Standards zur Integration häufig benötigter Dienste als „Plugins“ in bestehende Komponentenrahmen (component frameworks)

- vereinfacht und standardisiert das Vorgehen bei der Integration von Komponenten verschiedener Anwender

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Horizontale (generale) Standardkomponenten

Fokus auf generellem Anwendungsmodell. OMG hatte hier ursprünglich Standards für folgende Rahmen im Auge

- **Benutzerschnittstelle** (user interface)
- **Informationsverwaltung** (information management)
- **Systemverwaltung** (system management)
- **Aufgabenverwaltung** (task management)

Wird heute nur noch wenig vorangetrieben und stärker auf übergreifende Dienste konzentriert wie

- Internationalisierung, mobile Agenten, Zeit- und Druckdienst-Standards
- Fokus heute nicht auf Standard-Komponenten (ready to use), sondern auf Komponenten-Standards (Architektur)

Standardisierungsbemühungen der ursprünglichen Bereiche spielen heute praktisch so gut wie keine Rolle mehr.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Vertikale Standardkomponenten

Ursprünglich lag hier der Fokus auf Basisfunktionalitäten für unterschiedliche Marktsegmente. Ergebnisse bekommen zunehmend segmentüberschreitende Bedeutung

- Auch hier Komponenten-Standards statt Standard-Komponenten

Ausgehandelt werden dieses Standards in Aktivitäten verschiedener Domain Task Forces

- Business Enterprise Untegration
- Command, Control, Communications
- Finanzbereich
- Bereich Gesundheitsvorsorge
- Lebenswissenschaften
- Produktionsstrukturen
- Telekommunikation usw.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Sun und Java

- Java: Geschichte und Konzepte
- Wichtige von Java unterstützte Grundkonzepte
- Die J2EE-Architektur
- Java Komponentenmodelle
- Java Servlets / Java Server Pages (JSP)
- Enterprise Java Beans
- Ein Beispiel

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Java: Geschichte und Konzepte

große Erfolgsstory

- 1995/96 erste Anfänge (Sun Microsystems)
- Um 2000 eines der am häufigsten gebrauchten Schlagworte

Zwei Ansätze, womit Java wirklich zur Killerapplikation wurde

1. Das Sicherheitskonzept

- Applet wird doppelt geprüft (Übersetzungszeit und Ladezeit)
- strenge Sicherheitsregeln (security policies), die mit keiner anderen Programmiersprache erreicht werden
- Sicherheit auch im compilierten Code eines JIT-Compilers
- Sicherheitsaussagen durch Erzeugerzertifikate möglich
  - „signed applets“ (unter Nutzerkontrolle)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2. Die Java virtual machine

- Plattformunabhängigkeit
- großer Vorteil für Applikationen, die übers Web verteilt werden
- Vorteil vor allem im Standard
  - Java class-File Format und Java JAR-Archiv-Format
- beides nicht neu, jedoch in der Kombination und zu diesem Zeitpunkt durchschlagend

Java 2 (seit 1998)

- Fokus auf Applets aufgegeben
- Applets heute nur noch marginal
- Plattform-Editionen – Funktionalitätsbündel für verschiedene Klassen von Nutzern
  - J2SE als Plattform für Einzelanwendungen
  - J2EE als Serverplattform (seit Ende 1999)
  - J2ME für mobile und eingebettete Anwendungen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Java 2 (Fortsetzung)

- Formalisierung der Bezeichnungen Laufzeitumgebung (JRE), Entwicklungsumgebung (JDK) und Referenzimplementierung
- Referenzimplementierung der J2SE von Sun/Oracle auf der Basis der HotSpot-JVM
- J2EE als Standard mit Implementierungen von verschiedenen (unabhängigen) Anbietern
  - (nicht laufzeitoptimierte) Referenzimplementierung von Sun/Oracle als Beispiel-Implementierung im Quellcode verfügbar
- Prüfung der Unterstützung von Standards durch Kompatibilitätstest-Reihen
- Java BluePrints als Sammlung von Design-Richtlinien und Mustern, um spezielle Technologien zu unterstützen.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Wichtige von Java unterstützte Grundkonzepte

- **Methoden** (Verhalten, behavior) und **Attribute** (Status, state)
- **Schnittstellen:** Es können Interfaces und abstrakte Klassen definiert werden, die später (mittels ‚*implements*') implementiert werden sollen
  - Mehrfachvererbung von Schnittstellen (ohne Status und Verhalten)
- **Klassen:** Implementierungen von Schnittstellen. Es können von bestehenden Klassen spezielle Unterklassen (mittels ‚*extends*') abgeleitet werden.
  - Einfachvererbung von Implementierungen
  - vermeidet das Diamant-Problem
  - unveränderbare Implementierungen (final class, method, attribute)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- **Pakete** und **Pakethierarchien** als Modularisierungskonzept jenseits von Klassen
  - Namensgebung und Namensräume auf dieser Basis
  - keine Unterstützung von Mehrfachversionen
  - company-name.productname Präfix als Standard
  - Namensraum-Importe
- **Sichtbarkeitsklassen** von Attributen und Methoden (default, public, protected, private)
- **Ausnahmebehandlung** (exception handling): Es besteht die Möglichkeit, über Ausnahmen (mittels ‚try-catch'-Blöcken) vom Standard-Kontrollfluss abzuweichen
- **Threads und Synchronisation:** Nebenläufige Programmabläufe lassen sich mit Threads erzeugen und synchronisieren
- **Garbage Collection:** Nicht mehr referenzierte Objekte werden automatisch auf kontrollierbare Weise (‚*finalize*') zerstört
  - Anwender hat darauf aus Sicherheitserwägungen keinen Einfluss

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- **Objektserialisierung:** Objekte, welche die Schnittstelle *Serializable* implementieren, können in einen Datenstrom geschrieben oder aus einem solchen aufgebaut werden (z.B. Speichern in eine Datei)
- **Ereignisse (events):** werden einige Folien später genauer besprochen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

J2EE Architektur und Javas Komponentenmodelle für Middleware-Anwendungen

- Im Zentrum steht **Familie von Komponentenmodellen**
  - Client-Schicht: Anwenderkomponenten, JavaBeans, Applets
  - Webserver-Schicht: Servlets, JSP
  - Anwendungsserver-Schicht: EJB in vier Varianten (stateless session, statefull session, entity, message-driven session)

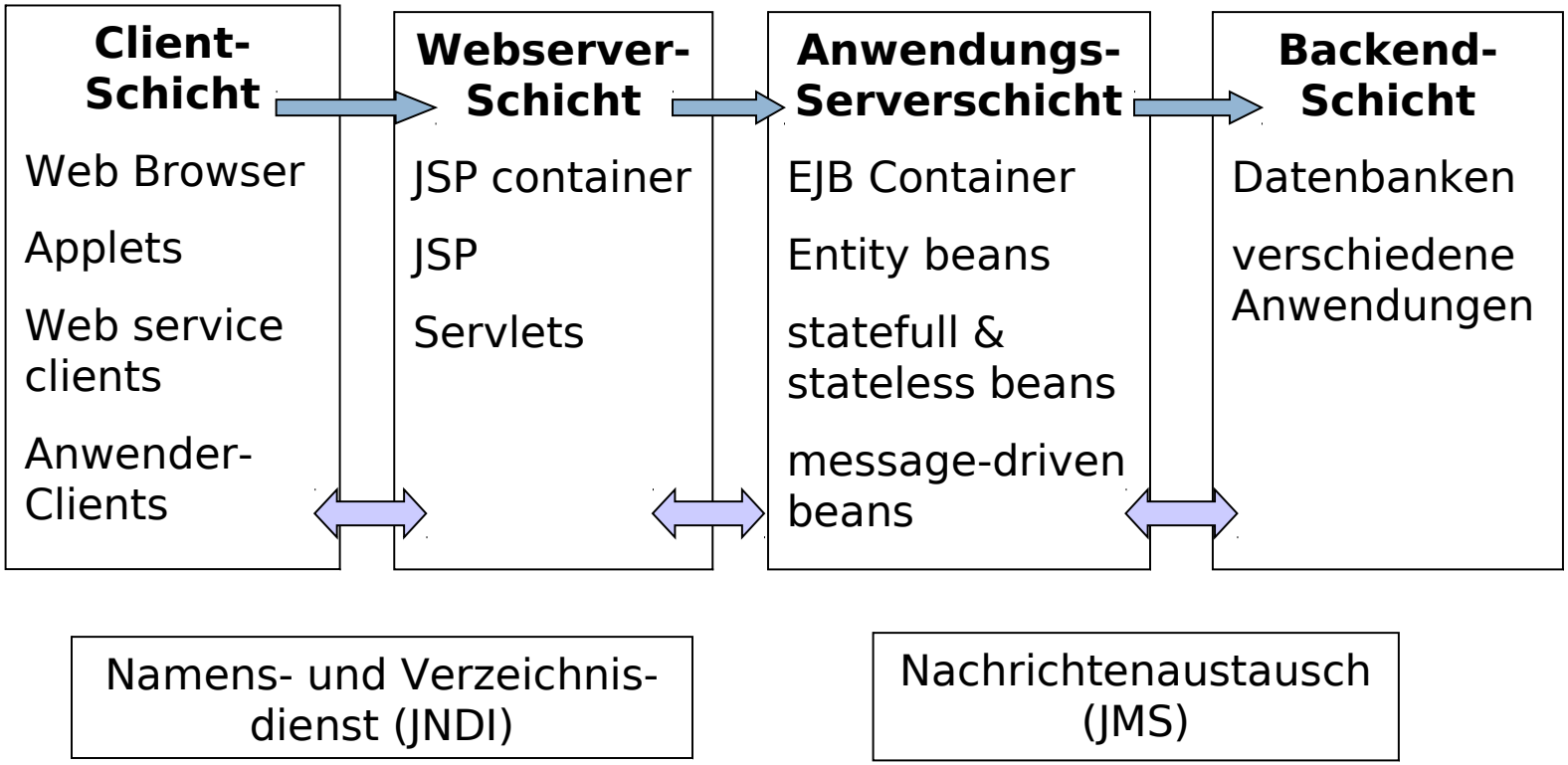
- Integrations-Ebenen (Basisdienste):
  - Namens- und Verzeichnis-Infrastruktur (naming and directory interface, JNDI) sowie Nachrichten-Infrastruktur (Java messaging service, JMS) bilden die Klammern zwischen den verschiedenen Schichten
  - weitere I.-E.: Transaktionskoordinierung, Sicherheitsdienste

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

J2EE Architektur

Kontrollfluss:

Datenfluss:



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Java-Komponentenmodelle

Komponentenmodelle in der Java 2 Enterprise Edition

- **Servlets / Java Server Pages**
- **Enterprise JavaBeans** und
- **Application Client Components**

Alle drei Modelle sind serverseitige Komponentenmodelle.

Wichtige Gemeinsamkeit ist die **Absetzung der Entpackungsphase** (deployment): Gesteuert durch Deployment-Deskriptor im XML-Format

- **Entpackung** = Prozess der Vorbereitung einer Komponente auf den Einsatz in einer speziellen **Umgebung** (context)
- Herstellen/Prüfen der (komponentenspezifischen) Voraussetzungen, unter denen die Komponente arbeitsfähig ist.
  - Beispiel: Datenbank-Anbindung, Persistenzfragen
- wird unterschieden von der **Installation**, die (plattformspezifisch) eine entpackte Komponente in einer speziellen Hardware-Konfiguration verfügbar macht.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Java Komponentenmodelle im Überblick

Applets: herunterladbare leichtgewichtige Komponenten für den Einsatz in Webbrowser-Clients. Einsatzgebiet heute durch andere Technologien abgedeckt

JavaBeans: unterstützt verbindungsorientiertes Programmieren über Beobachter und Ereignisse

- wenig Gemeinsamkeit mit EJB (außer dem Namen)

Java Servlets greifen den Gedanken von Applets auf, laufen jedoch auf dem Server ab und sind (meist) leichtgewichtige Komponenten

- werden durch einen Web Server instanziiert
- **Java Server Pages (JSP)**: verwandte Technologie, mit der dynamische Webseiten deklarativ definiert werden können
- Vergleichbar mit den Microsoft Active Server Pages .NET (ASP.NET)

Application Client Components

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Distribution und Packung von Java-Komponenten

- Geschäftsanwendungen (enterprise applications):
  ⇒ *.ear Dateien (enterprise archive)
- Servlets und JSP
  ⇒ *.war Dateien (web archive)
- Applets, JavaBeans, EJB
  ⇒ *.jar Dateien (Java archive)
- Entpackungs-Beschreibung (Deployment descriptor)
  - Formulierung von Anforderungen der Komponente durch Komponenten-Entwickler
  - Setzen offener Parameter (der „Blanks“) der Komponente durch Komponenten-Assemblierer
    - hart verdrahtet während der Entpackungsphase
    - Komponenten sind sonst zustandslos
  - Assemblierer ist eine andere Rolle als Komponentenentwickler, oft sogar in einer anderen Organisation
    - andere Ansätze trennen diese beiden Rollen deutlicher

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Java Server Pages und Servlets

dynamische Webseiten = Kombination dreier grundlegender Funktionen

- ankommende Anfragen akzeptieren, auf Gültigkeit und Autorisierung prüfen und an geeignete Komponente zur Weiterverarbeitung abgeben
- relevante Information aus den Informationsquellen extrahieren und den angefragten Inhalt (content) zusammenstellen
- Inhalt an den Anfrager übermitteln

Prototypisches Modell: wird von einem **Webserver** abgehandelt

- HTTP-Anfragen empfangen
- URL und enthaltene Parameter auswerten
- statische oder dynamische HTML-Seite (generiert mittels Aktivierung einer Komponente, z.B. über eine einfache Schnittstelle wie CGI) zurücksenden

Modell ist nicht auf HTML-Anfragen beschränkt

- Szenario liegt allen typischen Web-Diensten (Web Services) zu Grunde
- Dienste-Komponenten müssen nur eine simple Server-Schnittstelle implementieren

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Realisierung 1: Java Server Pages

- Einbettung von Code direkt in das Markup einer HTML-Datei
- Aus den Script-Teilen wird HTML-Code generiert
- Webserver ersetzt den Script-Code durch den generierten Code

Realisierung 2: Java Servlets

- Erzeugung von Markup durch Java Code

JSP versus Servlets:

- JSP erzeugen im Prinzip genau den Code des äquivalenten Servlets
- JSP existieren nur auf der Ebene von Java-Instanzen
- keine Unterstützung der natürlichen Abstraktionsmechanismen von Java (Pakete, Klassen, Methoden)
- JSP können wie Servlets auf externen Java-Code zugreifen
  - Regel: Java Code in JSP auf absolut notwendigen „Klebstoff“ reduzieren, funktionalen Teil in separate Klassen auslagern
- JSP: Konfusion mit clientseitigem JavaScript-Code möglich

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Vorteile des Servlet-Modells

- Inhalts-Generierung kann über mehrere Servlets verteilt werden
- Trennung in Präsentationsgenerierung und Geschäftslogik
- weitergehende Faktorisierung von Servlets längs Aufgabengrenzen möglich
- Abstraktions- und Modellierungsprinzipien des klassischen Software-Entwurfs sind anwendbar
- Servlets als Komponentenmodell

Servlets bieten sich auch als Einstiegspunkt in komplexere Geschäfts-anwendungen an, die beispielsweise auf Enterprise JavaBeans aufsetzen

Probleme mit der Mischung unterschiedlicher Komponenten-Modelle:

- mehrere Infrastrukturen müssen vorgehalten werden und interferieren
- Kommunikation zwischen den Modellen muss entworfen werden

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Auf der Basis gibt es verschiedene Modelle für die strukturierte Anwendungserstellung.

Beispiel Struts

- Klare Umsetzung des MVC-Modells
- Action-Dispatch-Konzept als Erweiterungspunkt für Controler
- JSP-Scripting und Tiles als View-Baukasten
- Deployment Descriptor beschreibt das „Wiring“

Strukturierter Ablauf definiert genau beschriebene **Erweiterungspunkte**, an denen die Teile der Anwendung eingehängt werden können, welche die Fachlogik enthalten.

Ansatz der **Basisdienste** findet sich dahingehend wieder, dass die Anwendungen nach einem gemeinsamen Architekturentwurf aufgebaut sind, in dem Funktionalität für Querschnittsaufgaben als fertige jar-Bündel zur Verfügung stehen.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Einleitung

Das **EJB-Konzept** realisiert einen klassischen OO-Zugang

- Kommunikation über Methodenaufrufe und Objektgenerierung
- Spezifikation, kein konkretes Produkt

Im Mittelpunkt steht ein **kontextuelles Kompositionskonzept**

- automatische Komposition von Komponenteninstanzen mit zugehörigen Ressourcen und Diensten
- Komponenten-Container-Architektur – der Container stellt die Laufzeitumgebung der Komponenten zur Verfügung und kapselt diese von der Umgebung (dem **Kontext**)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Das EJB-Konzept basiert auf **e-Beans** und **EJB Containern**

In der Deployment-Phase erfolgt über den **EJB Container** eine kontextuelle Zusammenführung der Beans mit Diensten und Ressourcen

- Container ist der „Pate“ der Beans, über welchen die gesamte Kommunikation läuft. Kein direkter Zugriff auf Attribute und Methoden von Beans, nicht innerhalb desselben Containers und nicht zwischen den Beans.
- Bean-Instanzen „leben“ als Objekte (in unserem Sinne) in der Container-Laufzeitumgebung
- EJB Container werden von EJB-Servern bereitgestellt, zum Beispiel vom J2EE application server
- Beschreibung des Zusammenspiels (Inhalt, Relationen, Rollen-, Sicherheits- und Transaktionsverhalten) in einem speziellen **Deployment-Deskriptor**
- da solche Deskriptoren umfangreich sein können, ist Werkzeugunterstützung sowohl zur Erstellung als auch zur Entpackung erforderlich

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Unterscheide (a) Methoden der Bean-Instanzen sowie (b) Methoden zum Management des Lebenszyklus der Beans.
- EJB 2 bietet dafür zwei Schnittstellen, EJB 3 verwendet (a) POJO – plain old java objects – und (b) Annotationen
- Kommunikation zwischen (clientseitigen) Stub-Klassen und (serverseitigen) Klassen im Container durch Java RMI
- Komponente in unserem Sinne ist also die Schnittstellen-Information, die Bean-Implementierung und die Metainformationen über das Zusammenspiel.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Enterprise JavaBeans EJB 2.1

- **Modell**: Beans = ununterscheidbare Objekte in einem Container
- Container-Abstraktion repräsentiert die spezielle Art, in welcher Beans an Ressourcen gebunden sind.
- kein direkter Zugriff auf Attribute und Methoden von Beans, auch nicht innerhalb desselben Containers
  - Verhältnis wie zwischen DB-Server und Datensätzen
  - Zugriff nur über zwei Schnittstellen, die **javax.ejb.EJBHome** (für Container) und **javax.ejb.EJBObject** (für Beans) erweitern
  - EJBHome: Methoden zum Management des Lebenszyklus der Beans
  - EJBObject: Methoden der Bean-Instanzen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Bean-Schnittstelle EJBObject

- **interface MyEJBObject extends javax.ejb.EJBObject**
- Aufruf-Schnittstelle, ergänzt EJBObject um die Fachlogik, über welche ein Client auf den Dienst zugreifen kann
- **Beschreibt** Dienstleistung
  - Darstellung von Attributen über get/set-Methoden
- Nichtlokale Clients rufen Schnittstelle auf Stub-Objekt, welches über RMI oder RMI-über IIOP mit dem EJB Container kommuniziert
  - deklarierte Operationen müssen Exception **java.rmi.RemoteException** auslösen können
  - Abfangen von Kommunikationsproblemen im Netz
- Lokale Clients können über lokale Version der Schnittstelle (Erweiterung von **EJBLocalObject**) zugreifen, wenn von der e-bean bereitgestellt

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Container-Schnittstelle EJBHome

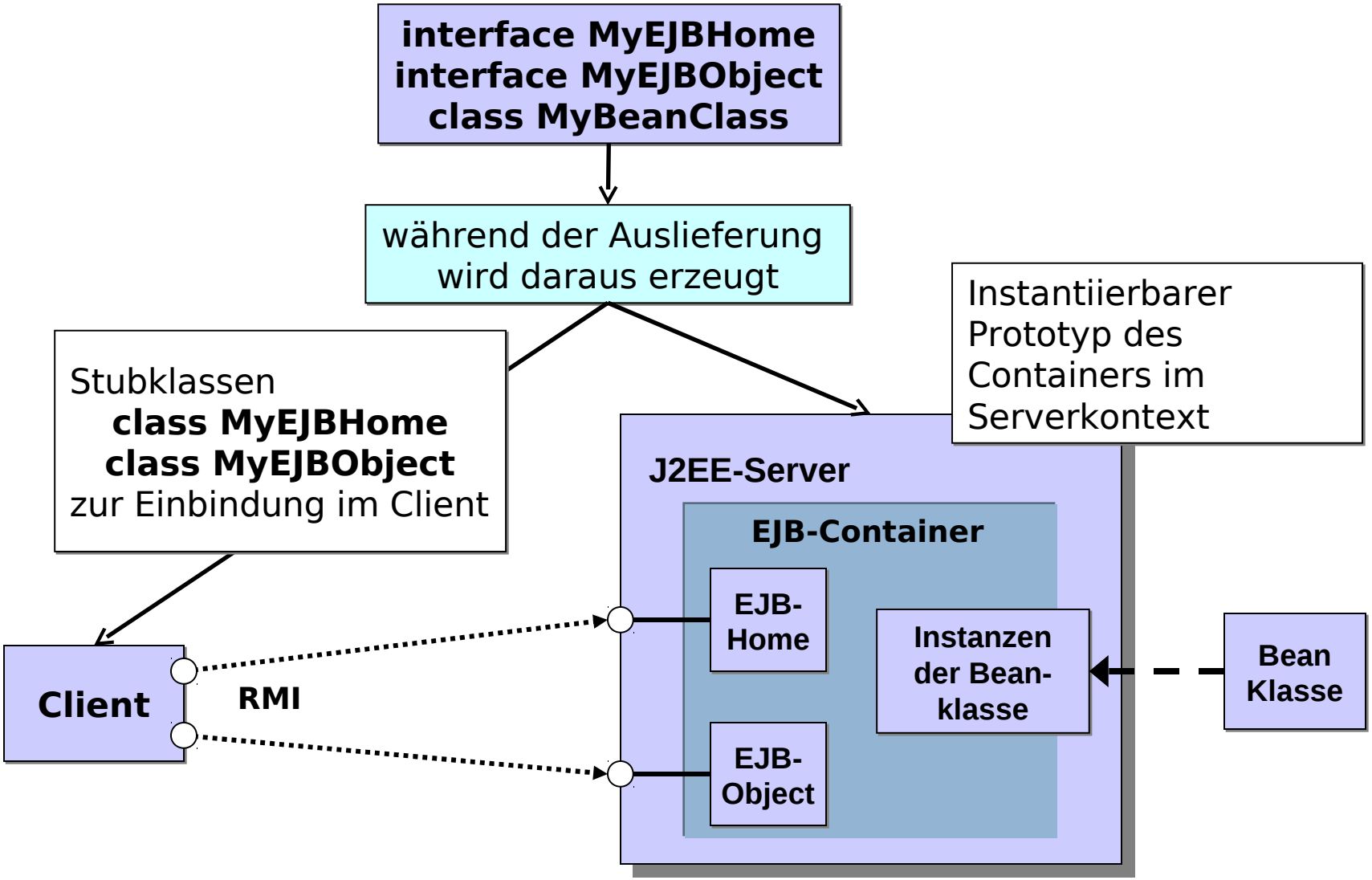
- **interface MyEJBHome extends java.ejb.EJBHome**
- Verwaltungs-Schnittstelle: verwaltet Bean-Instanzen im Container
  - Erzeugen neuer Instanzen
  - Auffinden vorhandener Instanzen
  - serialisiert alle Zugriffe auf seine Beans
- Operationen **create** und **findByPrimaryKey** (entity beans)
- optional weitere Methoden, die nicht mit einzelnen Beans zu assoziieren sind (analog zu statischen Methoden einer Klasse)
- begleitet Lebenszyklus seiner Beans
  - **setEntityContext** oder **setSessionContext** erzeugt Rückverweis auf den Container-Kontext
  - **ejbCreate** / **ejbRemove**
    - Ressourcenallokation bzw. –freigabe
  - **ejbPassivate** / **ejbActivate**
    - zeitweise Trennung von den Ressourcen und Speichern in serialisierter Form auf externem Medium

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Bean-Klassen

- **public class MyBeanClass implements javax.ejb.xxBean**
- es gibt **EntityBeans**, **SessionBeans** und **MessageDrivenBeans**, alle sind Subklassen von **EnterpriseBeans**
- **Implementierung** der zur Verfügung gestellten Operationen
  - also eigentlich auch **... implements MyEJBObject**
  - wird aber nur informell überwacht und diese Verbindung erst bei der Installation hergestellt (ist in Wirklichkeit noch etwas komplexer)
  - Entwickler muss auf Übereinstimmung der Signaturen selbst achten

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Anforderungen an den EJB–Container-Kontext

- JVM wird benötigt, ist aber allein nicht ausreichend
- EJB-Standard spezifiziert Schnittstelle und Verhalten von Container und J2EE
- Container benötigt zur Verwaltung seiner Beans
  - Namensdienst (Java Name and Directory Interface)
  - Transaktionsmonitor (Java Transaction API)
  - Datenbankzugriff (Java Data Base Connectivity)
  - eMail (Java Mail API)
  - Standard Java API
- Greifen dabei gewöhnlich auf weitere Dienste im Rahmen des J2EE Applikationsservers zu