Verteilte Systeme

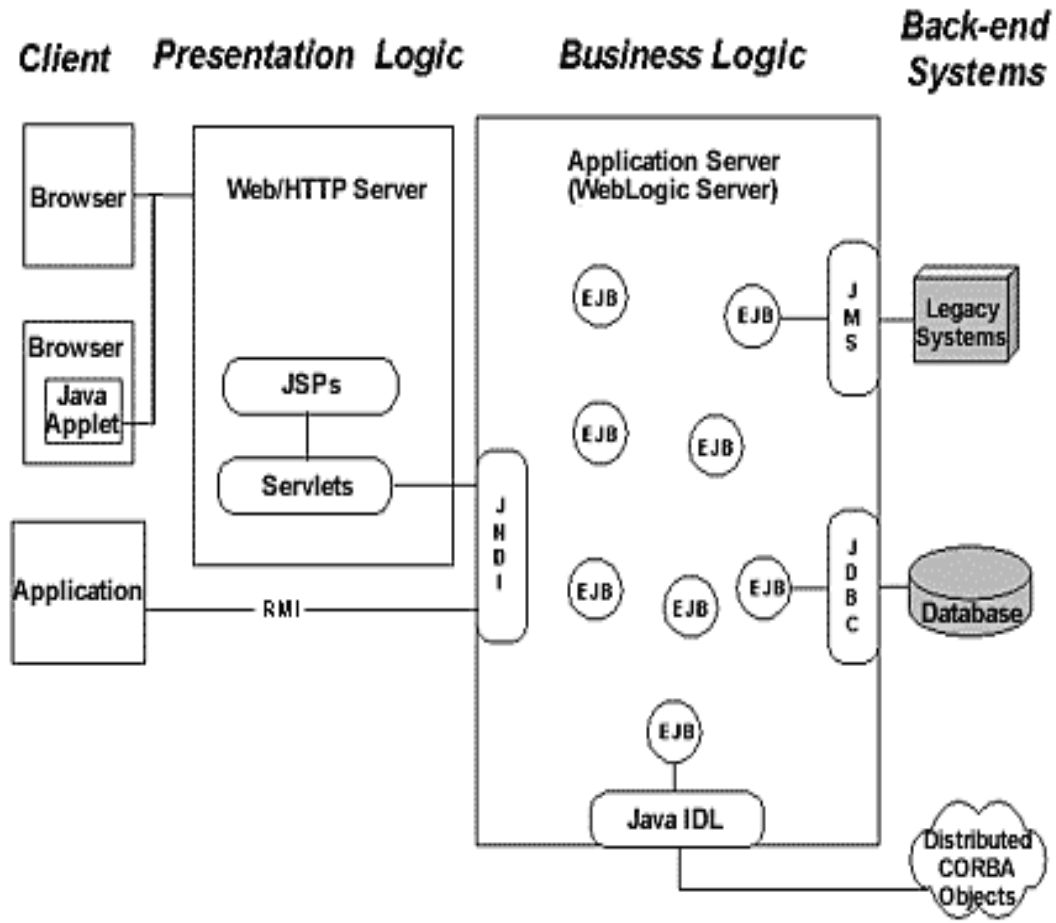
Vorlesung 12
Sebastian Iwanowski
FH Wedel

Mögliche Plattformen für Web Services

Java-Software für verteilte Systeme

J2EE: Java 2 Enterprise Edition

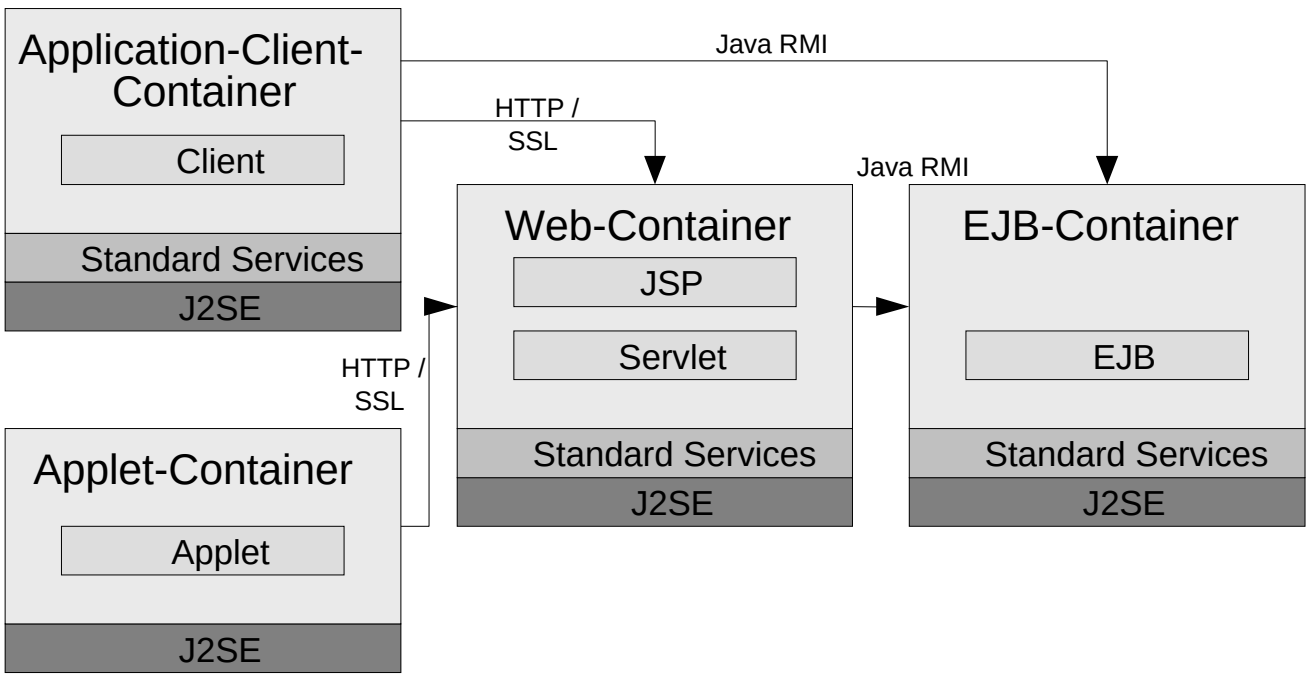
- **Namensraumverwaltung**
  (JNDI: Java Naming Directory Interface)
- **Transaktionsverwaltung**
  (EJB: Enterprise Java Beans)
- **Sicherheitsverwaltung**
- **Verbindung mit HTML-Code**
  (JSP: Java Server Pages)
- **Datenbankanbindung**
  (JDBC: Java Database Connectivity)



einiges auch schon in J2SE !

J2EE: Architektur

- Die Architektur von J2EE wird auch als *Container-Architektur* bezeichnet.
- Sie basiert auf vier Komponentenmodellen,
  jeweils mit Container und dazugehörigen Komponenten.



nach Ulrike Hammerschall: *Verteilte Systeme und Anwendungen*

J2EE: Komponentenmodelle

Applet-Container:

- Der Applet-Container entspricht dem Java Plug-In eines Browsers.
- Er läuft auf der Client-Seite einer Webanwendung.

Application-Client-Container:

- Der Application-Client-Container läuft auf der Client-Seite einer Webanwendung.
- Er unterstützt als Komponenten Java-Clients,
  die mit der Anwendungslogik des Servers kommunizieren.

Web-Container mit Servlets:

- Der Web-Container läuft in einem Webserver.
- Er unterstützt als Komponenten Webanwendungen basierend auf Servlets und JSP.

EJB-Container mit *Enterprise Java Beans*:

- Der EJB-Container läuft auf einer Zwischenschicht im Anwendungsserver.
- Er stellt die Anwendungslogik in Form von Enterprise Java Beans zur Verfügung.

nach Ulrike Hammerschall: *Verteilte Systeme und Anwendungen*

J2EE: Enterprise Java Beans

Serverseitiges verteiltes Komponentenmodell der J2EE Plattform:

- Beans als verteilte Komponenten,
- Kommunikation über Java RMI (inzwischen auch lokal möglich).

Der Container entspricht der Komponentenlaufzeitumgebung.

Der EJB Standard definiert:

- Struktur der Beans
- Aufgaben des Containers
- Schnittstellen zwischen Beans
- Schnittstellen zwischen Beans und Container
- Rollenmodell zu Entwicklung und Deployment

aus Ulrike Hammerschall: *Verteilte Systeme und Anwendungen*

J2EE: EJB-Typen

Session Beans (zustandsbehaftet und zustandslos):

- repräsentieren Abläufe (Prozesse) einer Anwendung
- sind immer mit Client-Sitzungen assoziiert
- werden Clients bei einem Aufruf zugeordnet.
- Stateless Session Beans repräsentieren eine Sitzung genau für die Dauer eines Methodenaufrufs.
- Stateful Session Beans stellen für einen vom Client festgelegten Zeitraum eine Sitzung zur Verfügung.

Message Driven Beans

- erweitern die synchrone Komponentenkommunikation (Java-RMI) um asynchrone Kommunikation.

Entity Beans

- repräsentieren die Daten einer Anwendung.
- bilden die Datenbank im Hauptspeicher nach.

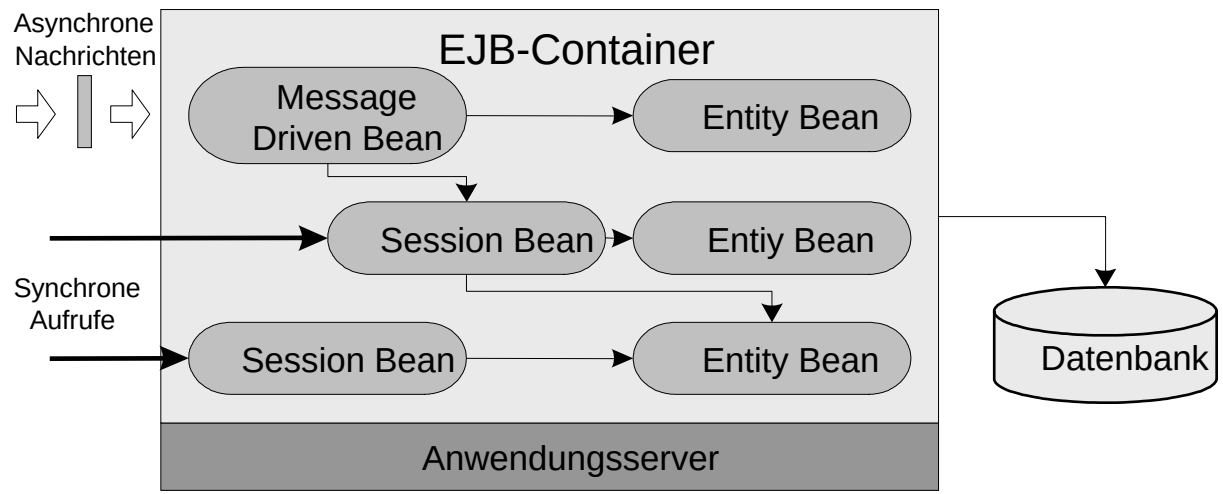
aus Ulrike Hammerschall: *Verteilte Systeme und Anwendungen*

J2EE: EJB-Typen

Funktion im EJB-Container

Die Struktur einer EJB-Anwendung orientiert sich an folgendem Aufbau:

- Session Beans bilden die Schnittstelle zum Client
- Message Driven Beans bilden die Schnittstelle zu einem JMS-Provider
- Entity Beans bilden die Schnittstelle zur Datenhaltung



nach Ulrike Hammerschall: *Verteilte Systeme und Anwendungen*

J2EE-Plattformen

Kommerziell (die bekanntesten)

- WebSphere (IBM)
- Bea Weblogic (Bea)
- Oracle Application Server (Oracle)
- Sun ONE Application Server (Sun)

Open Source

- JBoss (http://www.jboss.org/products/index)
- JOnAS (http://www.evidian.com/jonas/)

Weitere Infos:

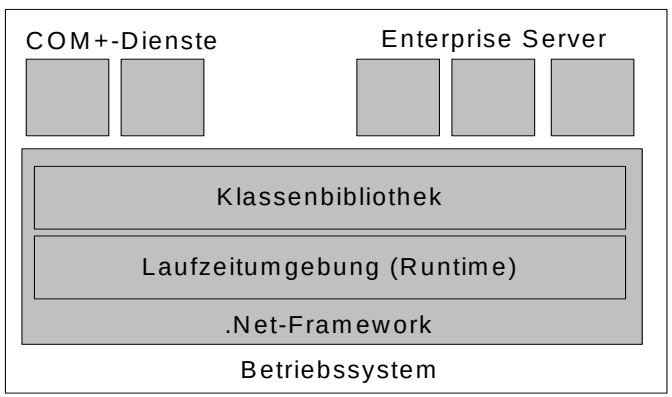
- Aktuelle Infos zu J2EE Standard und Plattformen: http://www.theserverside.com/
- Application Server Comparison Matrix: Liste aller aktuell verfügbaren J2EE Plattformen: http://www.theserverside.com/reviews/matrix.tss

aus Ulrike Hammerschall: *Verteilte Systeme und Anwendungen*

Microsoft-Software für verteilte Systeme

.Net: Plattform und Framework

- Zentrales Element der .Net-Plattform ist das .Net-Framework.
- Das .Net-Framework bietet eine Laufzeitumgebung sowie zusätzliche Funktionalität für verteilte Anwendungen.
- Kernkonzepte des .Net-Frameworks
  - Common Language Runtime (sprachunabhängige Laufzeitumgebung)
  - Framework-Library (sprachunabhängige Klassenbibliothek)



nach Ulrike Hammerschall: *Verteilte Systeme und Anwendungen*

.Net als Middleware-Plattform

Komponentenmodell(e):

- Assemblies
- COM

Laufzeitumgebung(en):

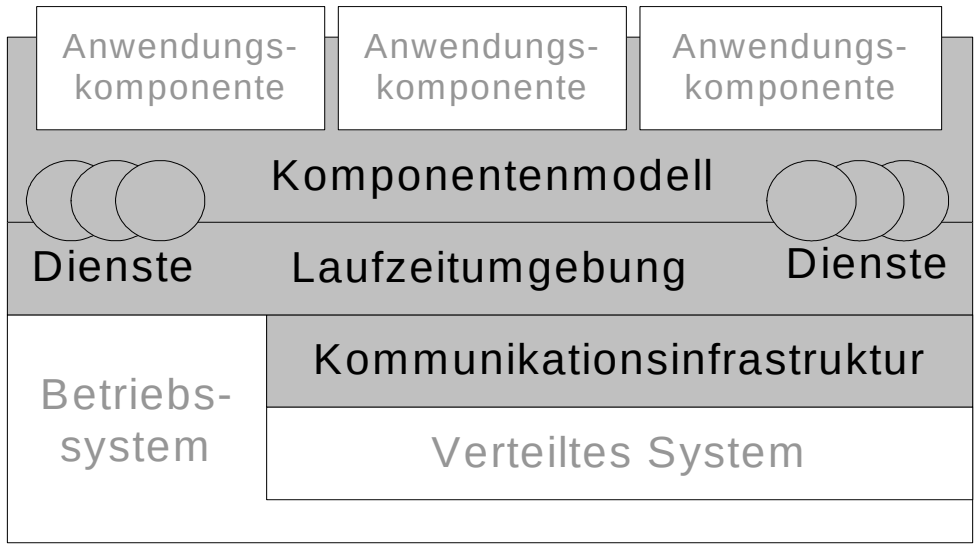
- Common Language Runtime
- COM+

Dienste:

- Klassenbibliothek
- Subsysteme (ADO.Net, ASP.Net)
- Enterprise Server
- MTS

Kommunikationsinfrastruktur(en)

- .Net Remoting
- XML Web Services
- MSMQ
- DCOM/RPC



aus Ulrike Hammerschall: *Verteilte Systeme und Anwendungen*

Zusammenfassung der Vorlesung

Verteilte Systeme

1. Innovative Beispiele aus der Praxis

   Auktionsbasierte Verkehrssteuerung, Verteilte Verkehrsinformationsgewinnung, Touristeninformationssystem

2. Allgemeine Anforderungen und Techniken verteilter Systeme

   Offenheit, Transparenz (Typen nach ISO), Skalierbarkeit, Exkurs: Verteilte Hardware

3. Die Client-Server-Beziehung und daraus entstehende Fragestellungen

   Definition, Protokolle (Callback und Polling), Transaktionen, Mehrschichtenarchitektur, Socket-Schnittstelle (mit Realisierung in Java)

4. Nebenläufigkeitstechniken in Java

   Definition, Threadkonzept, auch in Java (über Vererbung und Interfaces, Methodenaufrufe), Thread-Handling bei Callback und Polling (Prinzip und Verständnis), Risiken, Vorsorgemaßnahmen (nur Prinzip)

5. Entfernte Aufrufe

   RPC, RMI: Begriffe, Funktionsabläufe, Unterschiede.
   Java-RMI: Welche Klasse / Interface muss was realisieren.

Verteilte Systeme

6. Objektmigration

Daten, Objekte, Agenten: Unterschiede, unterschiedliche Anwendungsziele, Motivation für Migration

7. Agententechnologie

Grundlegende Agenteneigenschaften, Abgrenzung zwischen stationären und mobilen Agenten, Migrationstypen (stark vs. schwach, pull vs. push), Agentenkommunikation, Sicherheit bei mobilen Agenten

8. Dienstevermittlung

Forderungen an die Eigenschaften, unterschiedliche Aufgabenstellungen, Lösungsprinzipien, konkretes Beispiel Touristeninformationssystem

9. Synchronisation von Daten

Problemstellungen und Anwendungsgebiete in verteilten Systemen, Datenzentrierte Konsistenzforderungen und –lösungen, Clientzentrierte Konsistenzforderungen und –lösungen: Erkennen der Konsistenzforderungen an Beispielen.

Verteilte Systeme

10. Konzepte zur Erzielung von Fehlertoleranz

Fehlertypen, Fehlermaskierungsverfahren bei Servern: flach und hierarchisch, Äquivalenz verschiedener Problemstellungen, Satz von Lamport, verteilte Auswahl eines Koordinators, Problemstellung bei Leitungsfehlern: Graphentheoretische Definitionen und Aussagen

11. Web Services

Grundlegende Eigenschaften und Zielsetzungen von Webservices, Abgrenzung von Webservices gegenüber früheren Webanwendungen, Funktionalität von SOAP, WSDL und UDDI, Überblick über den inneren Aufbau von SOAP, WSDL und UDDI.
J2EE und .NET als Beispiele für Webservice-Plattformen

Nicht durchgenommen und daher auch kein Klausurthema:

Grundlagen der Kommunikation: Bewertungskriterien, Protokollschichten.
Details von Deadlockbehebung (Kap. 4), Namensverwaltung / Namenssuche, Äquivalenzbeweise zur Fehlertoleranz (mit Übungsaufgaben, Kap. 10), Sicherheit in verteilten Systemen

Das war die Vorlesung Verteilte Systeme