Vorlesung
Software aus Komponenten

3. Komponenten-Modelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2006/07

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

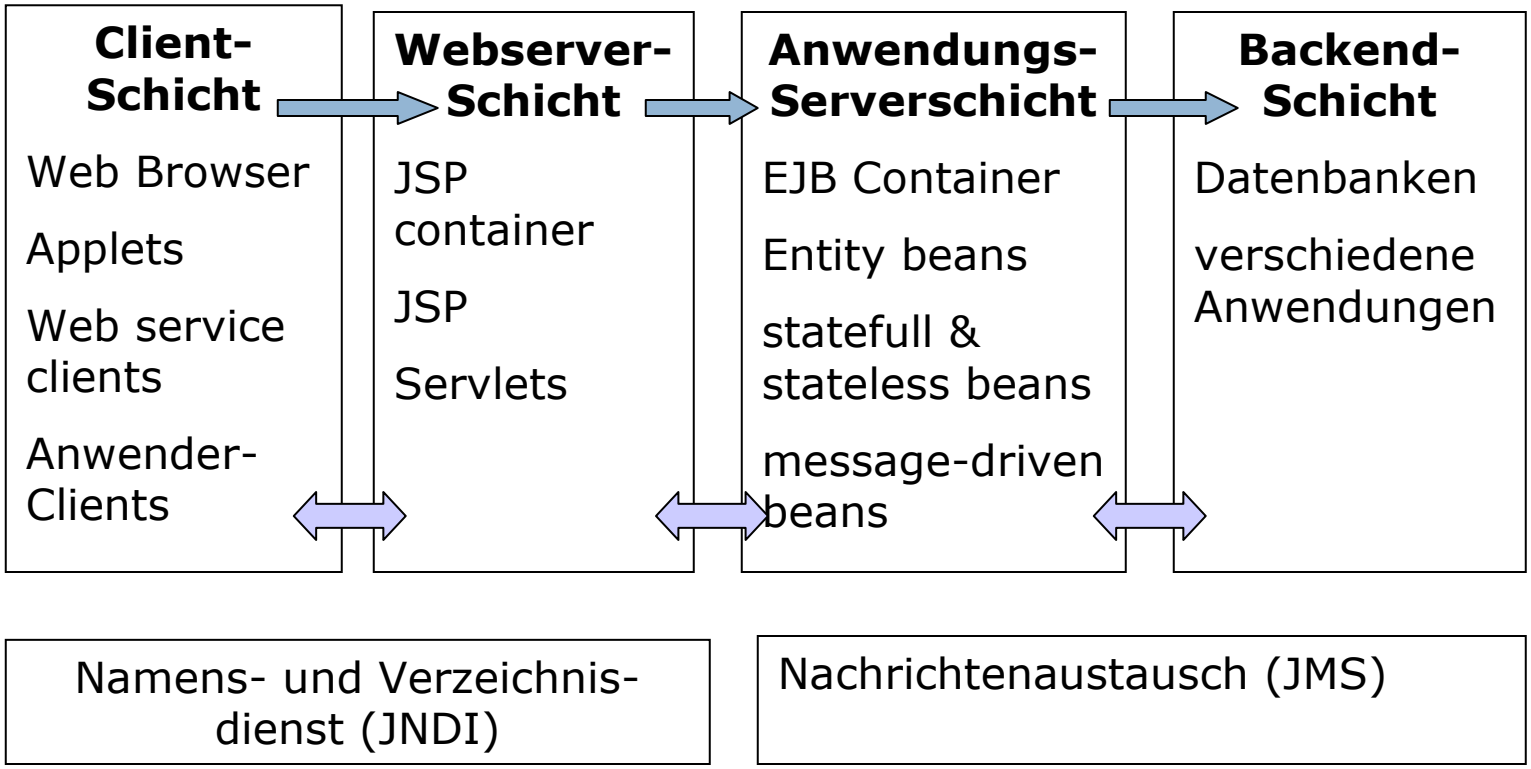
J2EE Architektur und Javas Komponentenmodelle für Middleware-Anwendungen

- 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
  - zentraler Ansatz: 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**
  - Im Gegensatz zum Prinzip des Webbrowsers konsistentere Verteilung der Funktionalität zwischen Client und Server (Rich Client)

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

- **<u>Realisierung 1</u>:**
  Einbettung von Code direkt in das Markup einer HTML-Datei
  - Realisierung durch Active Server Pages (ASP) und Java Server Pages (JSP)
  - Aus den Script-Teilen wird HTML-Code generiert
  - Webserver ersetzt den Script-Code durch den generierten Code
- Beispiel einer einfachen JSP-Seite:

```
<HTML><BODY>
<%
    java.util.Calendar calendar = new java.util.GregorianCalendar();
    int hour = calendar.get(currTime.HOUR);
    int minute = calendar.get(currTime.MINUTE);
%>
The time is: <%= hour %>:<%= minute %> - or it was when I looked.
</BODY></HTML>
```

- **<u>Realisierung 2</u>: Erzeugung von Markup durch Java Code**
  - Java Servlets: Interaktion mit dem Nutzer auf der Browser-Seite über den Webserver

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Das gleiche Beispiel als implementiertes Servlet:

```
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet
{
   protected void doGet (HttpServletRequest req, HttpServletResponse resp)
   throws ServletException, java.io.IOException
   {
       java.util.Calendar calendar = new java.util.GregorianCalendar();
       int hour = calendar.get(currTime.HOUR);
       int minute = calendar.get(currTime.MINUTE);
       ServletOutputStream out = resp.getWriter();
       out.print("<HTML><BODY>\r\n");
       out.print("The time is: " +hour+ ":" +minute+
                    " - or it was when I looked.\r\n");
       out.print("</BODY></HTML>\r\n");
   }
}
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

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
- Ausdrucksmächtigkeit von JSP kann durch JSP Standard Tag Library (JSTL) aufgebohrt werden.
  - Besonders in Richtung XML-Verarbeitung (SOAP, Web Services)
  - Erweiterung des Sprachumfangs muss durch den Webserver unterstützt werden

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 Geschäftsanwendungen 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

Einleitung

- keine Gemeinsamkeiten mit **JavaBeans**
  - Komposition durch verbindungsorientierte Programmierung
    - Beans können Ereignisquellen und –beobachter sein
    - Ereignisfluss wird festgelegt durch Anbinden von Quellen in einer Bean an Beobachter in anderen Beans

- Das **EJB-Konzept** realisiert dagegen 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
  - generische verbindungsorientierte Kompositionskonzepte werden nicht unterstützt (erst mit CCM als Erweiterung von EJB)
    - Komposition von e-beans nur über eigenes Design

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 vergleichbar mit statischen Methoden, Beans mit Instanzmethoden einer Java-Klasse
  - Container ist der „Pate" der Beans, über welchen die gesamte Kommunikation läuft
  - EJB Container werden von EJB-Servern bereitgestellt
    - J2EE Standard: J2EE application server
- Beschreibung (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

- 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
- Zu beiden gibt es **Local**-Varianten für den Einsatz in einer lokalen Umgebung

- Implementation von J2EE 1.4 (Auswahl)
  - BEA WebLogic Server 9.2, IBM WebSphere, Sun Application Server Plattform Edition 8, Oracle Application Server 10.1
  - Apache Geronimo 1.0-M5, JBoss Application Server 4.0

Siehe: http://java.sun.com/j2ee/compatibility.html