Vorlesung
Software aus Komponenten

3. Komponentenmodelle

Prof. Dr. Hans-Gert Gräbe
Wintersemester 2013/14

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: Rich Client, JavaBeans, Applets
  - Webserver-Schicht: Servlets
  - Anwendungsserver-Schicht: EJB

- 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

Java-Komponentenmodelle

Komponentenmodelle in der Java 2 Enterprise Edition

- **Servlets**
- **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** als *Ausführung* der Entpackung, die eine Komponente in einer speziellen Hardware-Konfiguration verfügbar macht.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Distribution und Packung von Java-Komponenten

- Geschäftsanwendungen (enterprise applications):
  ⇒ *.ear Dateien (enterprise archive)
- Servlets
  ⇒ *.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

Servlets und Java Server Pages

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

Das Servlet-Modell

- (Konzeptionelle) Komponenten in diesem Modell heißen *Servlet*. Ein Servlet implementiert die Schnittstelle javax.servlet.Servlet, womit ein kommunikatives Grundverhalten vorgegeben ist (Komponenten-Modell).
- Eine weitere Vereinfachung ergibt sich bei Verwendung der Klasse javax.servlet.http.HttpServlet, die ein Grundgerüst implementiert, das durch Überschreiben einzelner Methoden (doGet, doPost, service) angepasst werden kann (Design Pattern: Erweiterungspunkt).
- Servlets werden in einem *Servlet-Container* ausgeführt, der entweder von einem J2EE-Server erzeugt wird oder im Rahmen anderer Webserver-Projekten (etwa Apache Tomcat) implementiert ist (Komponenten-Frameworks).
- Das Zusammenspiel von Servlets in einem Container wird durch Metainformationen beschrieben, die in einer XML-Datei namens *web.xml* bereitgestellt werden (Deployment Descriptor).

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Das Servlet-Modell

- Der Deployment Descriptor wird zusammen mit der kompilierten Servlet-Klasse (sowie ggf. weiteren Ressourcen) in einer Archiv-Datei, dem *Web-Archiv*, zusammengeführt (Auslieferbare Komponente).
- Dieses Web-Archiv wird dem Servlet-Container über eine von ihm bereitgestellte Funktionalität übergeben (deployment). Zur Laufzeit greift der Webserver auf den Servlet-Container zu, der wiederum eine Instanz des Servlets erzeugt und die entsprechende Methode startet. (Lokale Komponente)

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

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Auf Servlet-Basis gibt es verschiedene andere weniger komplexe Modelle für die strukturierte Erstellung von Webanwendungen.

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“

Auch hier: Ein standardisierter 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 auf der Basis von Beschreibungen
- Komponenten-Container-Architektur – der Container stellt die Laufzeitumgebung der Komponenten zur Verfügung und kapselt diese von der Umgebung (dem **Kontext**)
- Container überwacht die Ressourcenzuordnung, die damit für die Komponenten transparent ist.

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