Java EE 5

Infopoint, 02.12.2009

Jörg Wüthrich

Inhalt

- Historie
- Neuerungen gegenüber J2EE 1.4
- EJB3
- Webservices (JAX-WS)
- Persistenz (JPA)
- Ausblick auf Java EE 6

Historie

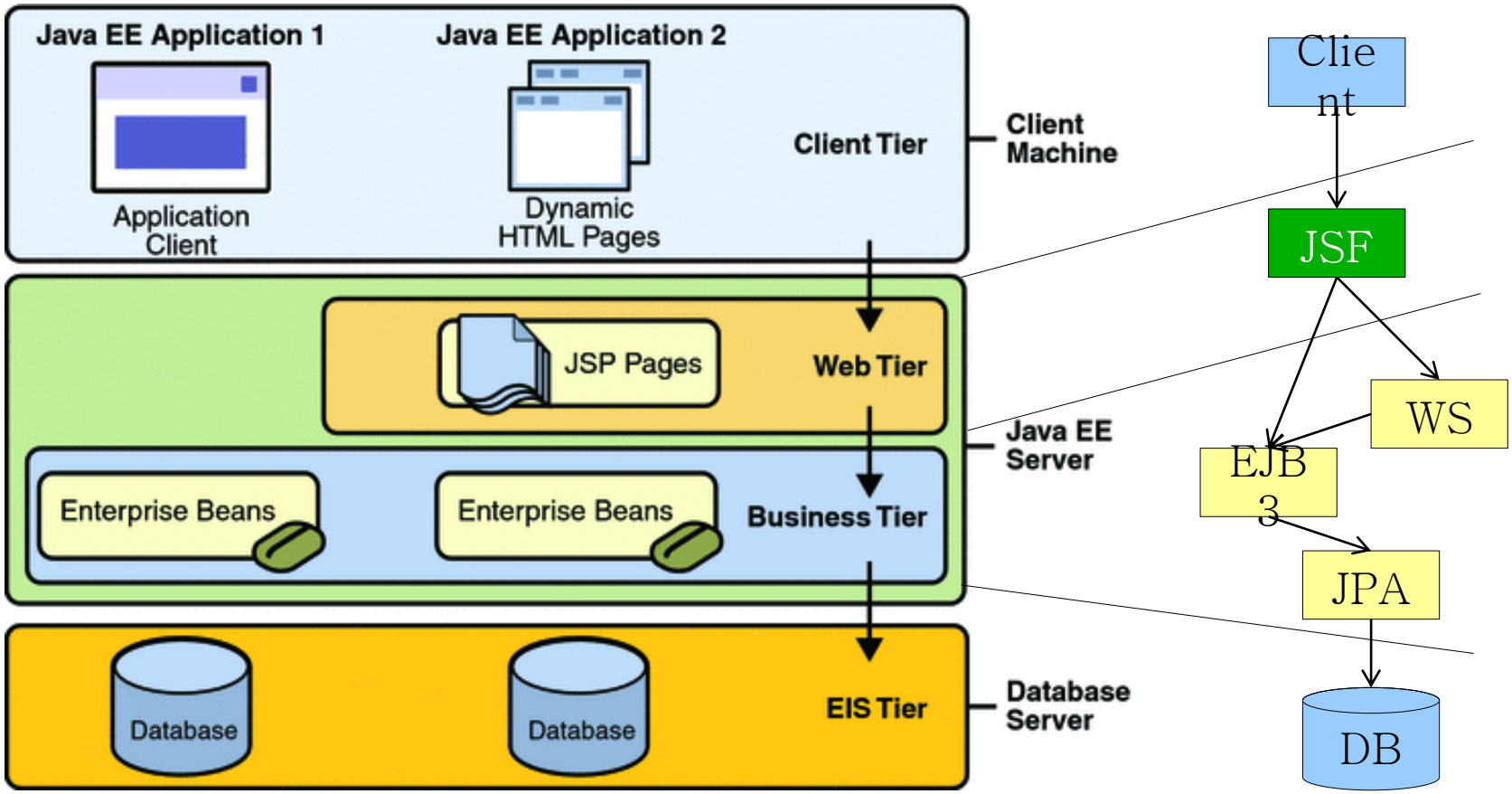
- 12.1999: J2EE 1.0
- 2000: J2EE 1.2
- 05.2000: J2EE 1.2.1
- 09.2001: J2EE 1.3
- 11.2003: J2EE 1.4
- 05.2006: Java EE 5
- Java EE 6: geplant auf 2008, seit Oktober 2009 gibts den Final Draft

Neuerungen gegenüber J2EE 1.4

Neuerungen gegenüber J2EE 1.4

- Vereinfachungen des Java EE Programmier-Modells
- EJB 3.0 mit POJOs
- JPA: Erneuerung des Objekt-relationalen Mappings (Ablösung der EJB CMP)
- Web-Services: Vereinfachungen mit JAX-WS und JAXB 2.0
- JSF / JSTL: Vereinfachte Web UI Entwicklung
- Annotationen ersetzen auf Wunsch die meisten Deployment Deskriptoren
- Streaming API für XML (StAX)

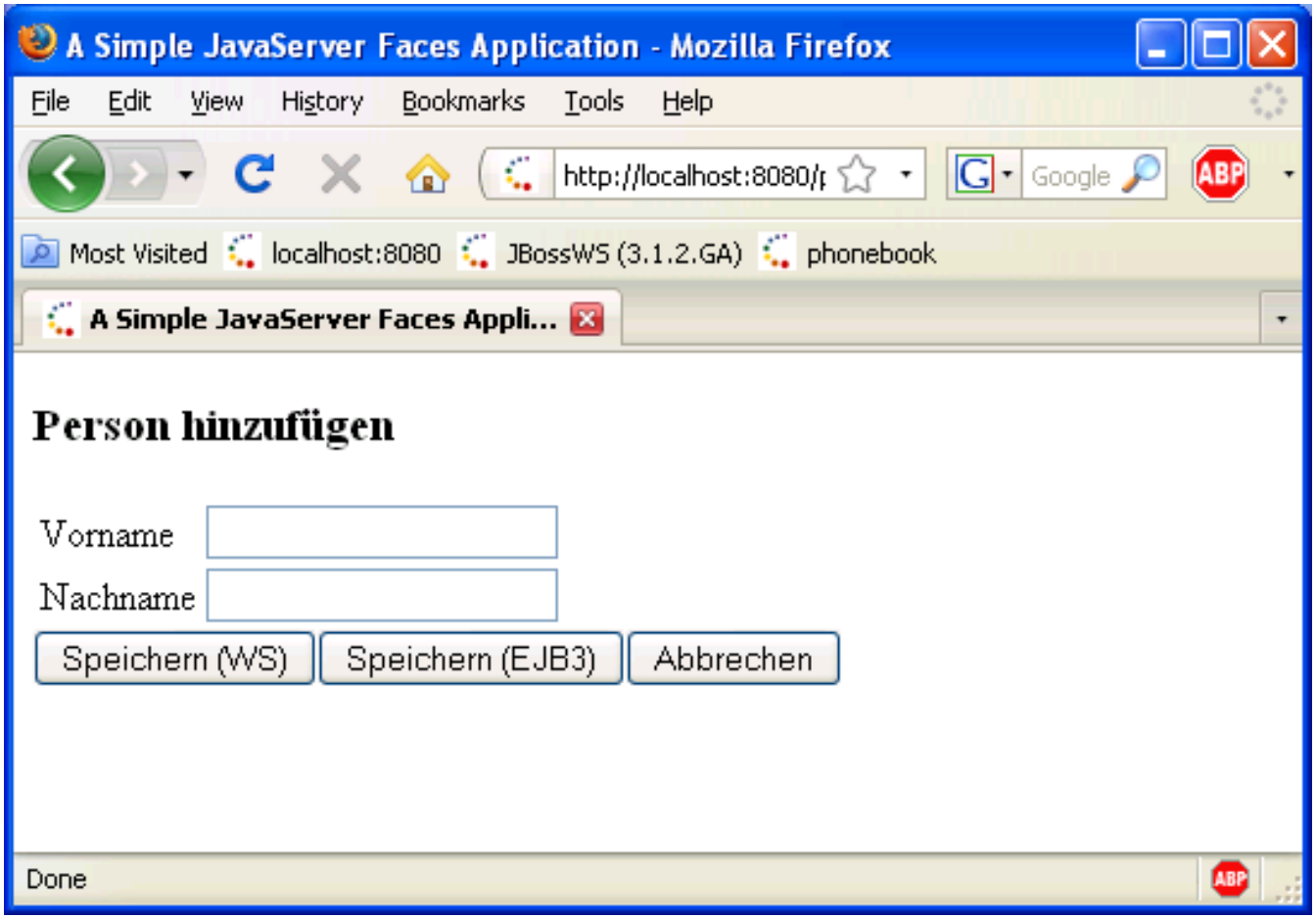
Demo-Applikation



siehe Urs' Präsentation vom 04.2007

behandelt in dieser Präsentation

Live - Demo


A Simple JavaServer Faces Application - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/
Google
Most Visited localhost:8080 JBossWS (3.1.2.GA) phonebook
A Simple JavaServer Faces Appli...
Person hinzufügen
Vorname
Nachname
Speichern (WS)
Speichern (EJB3)
Abbrechen
Done

EJB 2.1

```
interface
PersonDbService
 extends javax.ejb.EJBObject {

int addPerson(Person person)
  throws RemoteException;
```

```
interface
PersonDbServiceLocal
  extends javax.ejb.EJBLocalObject
```

```
interface PersonDbServiceHome
extends javax.ejb.EJBHome

PersonDbService create()
  throws                    RemoteException,
CreateException
```

```
interface
PersonDbServiceLocalHome
extends javax.ejb.EJBLocalHome
```

```
PersonDbServiceBean implements javax.ejb.SessionBean

public int addPerson(Person person) { … }

public void ejbCreate() {}
public void ejbRemove() {}
public void ejbActivate() {}
public void ejbPassivate() {}
public void setSessionContext(SessionContext sc) {}
```

Deployment-Deskriptor:

ejb-jar.xml

EJB 2.1 Client

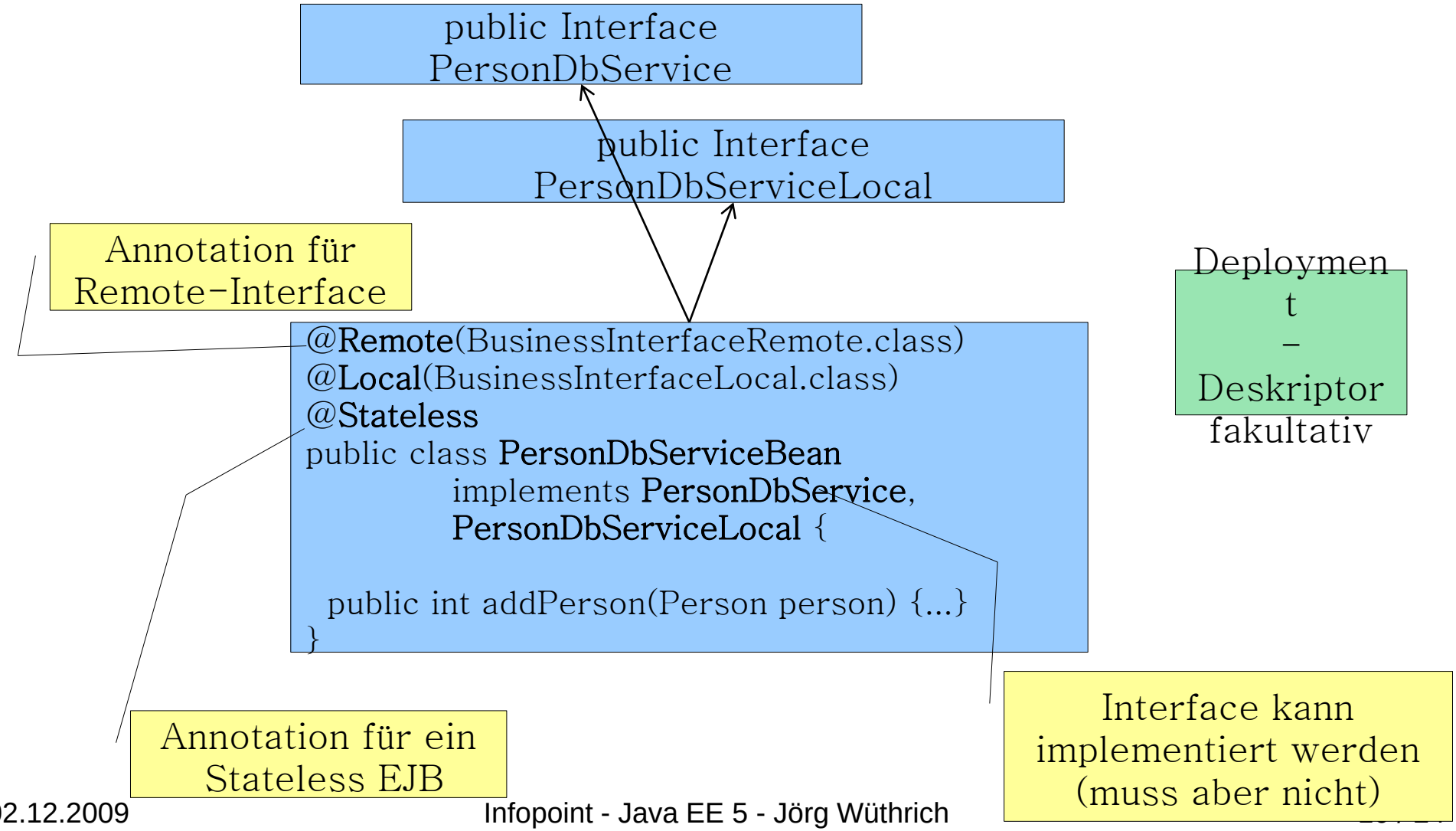
- Aufruf:

```
Context initial = new InitialContext();
Context myEnv = (Context)initial.lookup("java:comp/env");
Object objref = myEnv.lookup("ejb/PersonDbService");

PersonDbServiceHome home =
        (PersonDbServiceHome)PortableRemoteObject.narrow(objref,
                PersonDbServiceHome.class);

PersonDbService personDbService = home.create();
```

EJB3


public Interface
PersonDbService
public Interface
PersonDbServiceLocal
Annotation für
Remote-Interface
@Remote(BusinessInterfaceRemote.class)
@Local(BusinessInterfaceLocal.class)
@Stateless
public class PersonDbServiceBean
implements PersonDbService,
PersonDbServiceLocal {
public int addPerson(Person person) {...}
}
Deployment
-
Deskriptor
fakultativ
Annotation für ein
Stateless EJB
Interface kann
implementiert werden
(muss aber nicht)

EJB3 Client

- Aus Application-Container:
  - Mit Annotation; Container macht Injection des EJB

```
@EJB
private PersonDbService personService;
```

- Aus Servlet Container:
  - Einfacher Lookup (kein “narrow” mehr)

```
InitialContext ctx = new InitialContext();
return (PersonDbService)
        ctx.lookup("phonebook-business-
ear/PersonDbServiceBean/remote");
```

Webservices – J2EE 1.4

- Am Anfang war SOAP ...
- … dann kam WSDL...
- … und dann JAX-RPC
  - Standardisiert, wie Java Webservices behandelt
  - Remote Procedure Calls
  - nur Java 1.4 Features
  - Eigenes Mapping der Datentypen; deckt ca. 90% aller XML-Schema Typen ab
  - SOAP with Attachments

Webservices – Java EE 5

- voll rückwärtskompatibel, dazu kommt:
- JAX WS (Teil von Java SE 6)
  - Nachrichten-orientierte Implementierung (↔ RPC)
  - unterstützt asynchrone Aufrufe für Clients
  - macht von Java 5 Features Gebrauch
  - Mapping der Datentypen erfolgt mit JAXB → alle XML-Schema Datentypen unterstützt
  - unterstützt XML/HTTP (nicht nur SOAP/HTTP)
  - unterstützt auch MTOM (“Message Transmission Optimization Mechanism” via JAXB) → Bessere Interoperabilität

Webservices - Vergleich

- JAX-RPC 1.1 Code

```
public interface PersonDbWebService
  extends java.rmi.Remote {

  public int addPerson(Person person)
    throws java.rmi.RemoteException;
}

---------

public class PersonDbServiceBean
  implements PersonDbWebService {

  public int addPerson(Person person)
    throws java.rmi.RemoteException {

        ...
  }
}
```

- JAX-WS 2.0 Code

```
@WebService
public class PersonDbServiceBean {

  @WebMethod
  public int addPerson(Person person) {
    ...
  }

}
```

Webservice - Topdown

- Topdown: aus WSDL generiert

```
@WebService(name = "PersonDbWebService", targetNamespace =
"http://demo.javaee5.wuethrich.ch/")
public interface PersonDbService {

    @WebMethod
    @WebResult(targetNamespace = "")
    @RequestWrapper(localName = "addPerson",
                targetNamespace = "http://demo.javaee5.wuethrich.ch/",
                className = "ch.wuethrich.ws.generated.AddPerson")
    @ResponseWrapper(localName = "addPersonResponse",
                targetNamespace = "http://demo.javaee5.wuethrich.ch/",
                className = "ch.wuethrich.ws.generated.AddPersonResponse")
    public int addPerson(
        @WebParam(name = "arg0", targetNamespace = "")
        Person arg0);
```

- Implementierung realisiert generiertes Interface

Webservice - Bottom-Up

- Bottom-Up: annotiertes EJB

```
@javax.ejb.Stateless
@javax.ejb.Remote(PersonDbServiceIF.class)
@javax.jws.WebService(serviceName="PersonDbServiceClient",
        name="PersonDbService")
public class PersonDbServiceBean implements PersonDbServiceIF {

        @javax.jws.WebMethod()
        public int addPerson(Person person) {
                ...
        }

}
```

- Daraus wird WSDL generiert

Webservice Client

- aus Application-Container

```
@WebServiceRef
private PersonDbWebService webService;
```

- aus Servlet-Container

```
URL wsdlLocation
        = new
URL("http://localhost:8080/service/PersonDbServiceBean?wsdl");
QName qname = new QName("http://demo.javaee5.wuethrich.ch/",
        "PersonDbServiceClient");

PersonDbServiceClient wsClient
        = new PersonDbServiceClient(wsdlLocation, qname);
PersonDbWebService webService = wsClient.getPersonDbWebServicePort();

webService.addPerson(person);
```

Persistenz J2EE 1.4

- Persistenz wurde mittels Entity Beans realisiert
  - eigener Typ von EJBs
  - implementiert javax.ejb.EntityBean
  - haufenweise Konventionen, welche eingehalten werden müssen, damit alles klappt
- Transaktions-Verhalten konnte dem Container übergeben oder selbst implementiert werden
  - davon abhängig muss im Bean mehr oder weniger implementiert werden
  - Abfragen auf die Entity laufen über EJB QL im Deployment Deskriptor oder PreparedStatements

Persistenz Java EE 5

- JPA (Java Persistence API)
  - inspiriert von Hibernate
  - Entities sind keine Beans mehr, sondern annotierte Klassen
  - Persistenz-Kontext verwaltet Transaktionen
    - Kontext kann vom Container oder der Applikation verwaltet werden
  - Entities können “detached” werden vom Persistenz-Kontext
    - Verwendung in einem EJB-Interface kein Problem
  - verwendet JPA QL für Abfragen

Persistenz Java EE 5

- Persistente Klasse

```
@javax.persistence.Entity
@javax.persistence.Table(name="Persons")
public class Person implements Serializable {

  private static final long serialVersionUID = 1L;

  @javax.persistence.Id
  @javax.persistence.GeneratedValue
  private int id;

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }
```

Persistenz Java EE 5

- Speichern

```
public class PersonDbServiceBean implements PersonDbService {

  @javax.persistence.PersistenceContext
  private EntityManager em;

  public int addPerson(Person person) {
    em.persist(person);

    return person.getId();
  }
```

- Query

```
  public List<Person> getPersonList() {
    Query query = em.createQuery("SELECT p FROM Person p");
    List<Person> result = query.getResultList();

    return result;
  }
}
```

Ausblick Java EE 6 (1/2)

- Middle-Tier
  - EJB 3.1: optionale lokale Business-IF
  - asynchroner Aufruf von Session Beans
- Webbeans 1.0 (→ JBoss Seam, Google Guice)
- Web-Tier
  - Erweiterung Servlet-API
  - Web-Framework pluggability
  - AJAX Support
- Bessere Unterstützung von RESTful Webservices

Ausblick Java EE 6 (2/2)

- Unterstützung von Profilen
  - Idee: Reduktion der Grösse der Umgebung, wenn nicht alle Features benötigt werden
  - Profile sollen spezifische Lösungen bzw. Anforderungen abdecken
  - Aktuell geplant: Web-Profil
- Rauswerfen wenig benutzter Features
  - Kandidaten sind JAX-RPC (abgelöst durch JAX-WS) und EJB CMP (Container Managed Persistence; abgelöst durch JPA)

Referenzen

- Sun Tutorial für Java EE 5 (pdf: 1126 Seiten)
  http://java.sun.com/javaee/5/docs/tutorial/doc/
- Sun Tutorial für J2EE 1.4 (pdf: 1542 Seiten)
  http://java.sun.com/j2ee/1.4/docs/tutorial/doc/
- Evolving the Java SE and Java EE Platforms
  http://jazoon.com/download/presentations/2100.pdf
- Neue Features in Java EE 6
  http://www.javabeat.net/articles/99-new-features-in-java-ee-60-1.html
- History und Vergleich J2SE 1.4, Java EE 5
  http://de.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
- Webservice-Annotationen https://jax-ws.dev.java.net/jax-ws-ea3/docs/annotations.html

backup

Webservice - WSDL

- Eine WSDL-Datei enthält folgende Informationen:
  - types
    - Definition der Datentypen (meist XML-Schema)
  - message
    - definiert Datenelemente der Operationen
  - porttype
    - Beschreibt die vorhandenen Operationen und verwendeten Messages (Java: “Interface”)
  - binding
    - definiert das Nachrichtenformat und Protokoll-Details für jeden Porttype