Vorlesungsreihe
Entwicklung webbasierter Anwendungen

Java Enterprise Applications

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

Gliederung

- Komplexe Webanwendungen mit Java
  - J2EE
  - EJB - Enterprise JavaBeans
  - Struts
  - Java Server Faces
  - Hibernate , Spring
- Performancevergleich aller Technologien

Quelle(n):

- **Allgemeiner Java-Technolgieüberlick**

  http://www.oracle.com/technetwork/java/index.html
- **Speziell J2EE**
- http://www.oracle.com/technetwork/java/javaee/downloads/index.html

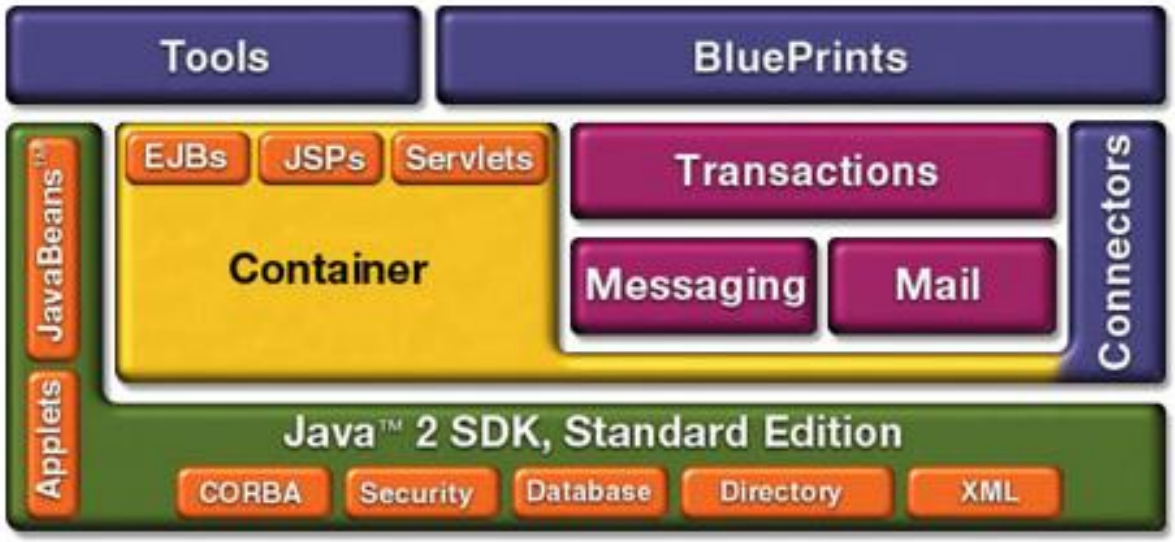
Java - Technologien

Java ist in verschiedenen Plattformen verfügbar :

- J2SE - Java 2 Standard Edition (Core/Desktop)
- J2EE - Java 2 Enterprise Edition (Enterprise/Server)
- J2ME - Java 2 Micro Edition (Mobile/Wireless)

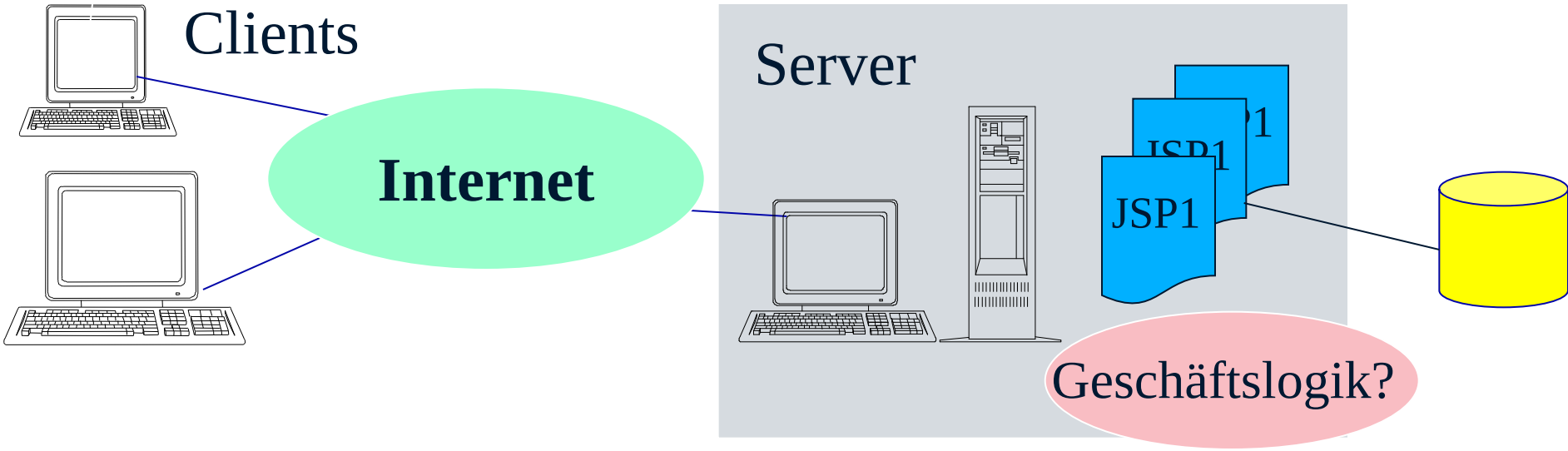
abgeleitet davon auch als

- JavaCard, XML, Java Web Services



Quelle: Sun (www.sun.com)

Motivation für spezielle Business-Logik-Technologien



Probleme bei der Verwendung von JSV / JSP

1. Die Webanwendung besteht aus einer relativ großen Anzahl von Servletts / JSP´s
2. Zugriffs- und Sicherheitsmechanismen müssen selbst implementiert werden
3. Die Geschäftslogik lässt sich nur aufwändig über die Einzel-JSP´s realisieren

➔ für komplexe Anwendungen sind bessere Technologien notwendig

Anforderungen an die Business-Logik-Technologien

Bei komplexen Anwendungen sind folgende Aufgaben zu unterstützen :

- **Nebenläufigkeit** (Multitasking über viele Requests)
- **Ressourcenverwaltung** (Lastverteilungs)
- **Transaktionskonzepte** (sicherer Abschluß komplexer Aktionen)
- **Nachrichtenmanagement** (synchron / asynchron: Menschliche Interaktionen, welche auch komplett ausfallen können)
- **Fehlertoleranz** (Ausfall Netz / Störung Backend-Systeme)
- **Sicherheit** der Kommunikation und der Prozesse

Aktuelle Anforderungen an die Technologien :

- **Portabel** / unabhängig von Betriebssystemen und Herstellern
- **offen** und **erweiterbar**
- schnelle und kostengünstige Entwicklungszyklen
- Einfache Wartung und Investitionsschutz (z.B. bei Updates)

Definition Java Enterprise Edition- Java Enterprise Beans

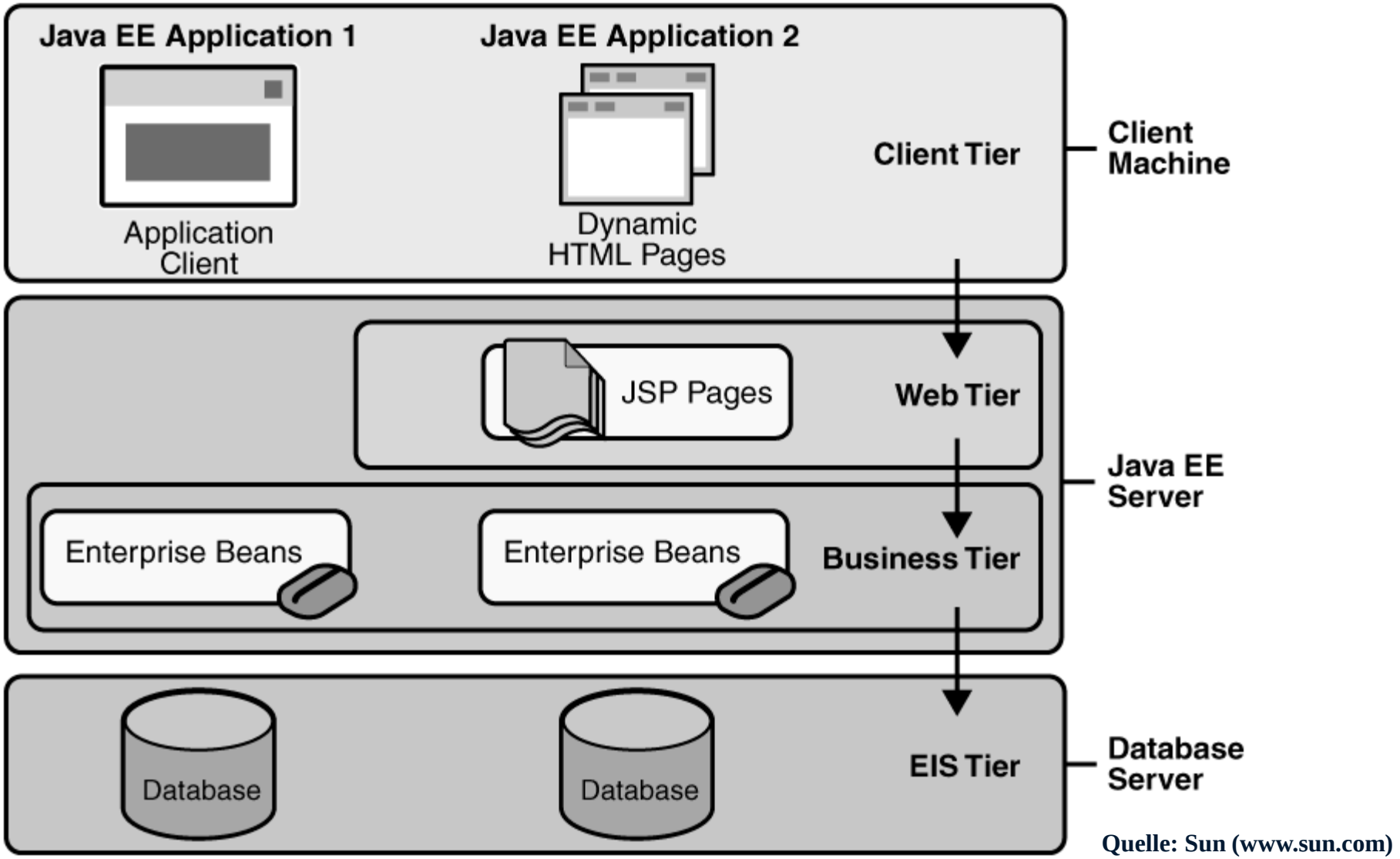
Definitionen (by Sun - Dok. zu JavaEE)

- „The aim of the **Java™ Platform, Enterprise Edition (Java EE),** platform is to provide developers a powerful set of APIs while reducing development time, reducing application complexity, and improving performance of **distributed, transactional, and portable (Enterprise) applications**“.
- **Java ENTERPRISE (EJB) beans** are **Java EE components** that implement Enterprise Java-Beans (EJB) technology. Written in the Java programming language, an *enterprise bean* is a **server-side** component that encapsulates the business logic of an application.

- Hinweis: **Java Beans** allein sind dagegen „normale“ (=lokale) Komponenten analog zu anderen Komponentensystemen (vgl. ActiveX / OCX). **Enterprise** Beans implizieren i.d.R. immer die verteilte Implementierung von Diensten, meist über das Web !

JEE – Multi-Tier-Applikationen

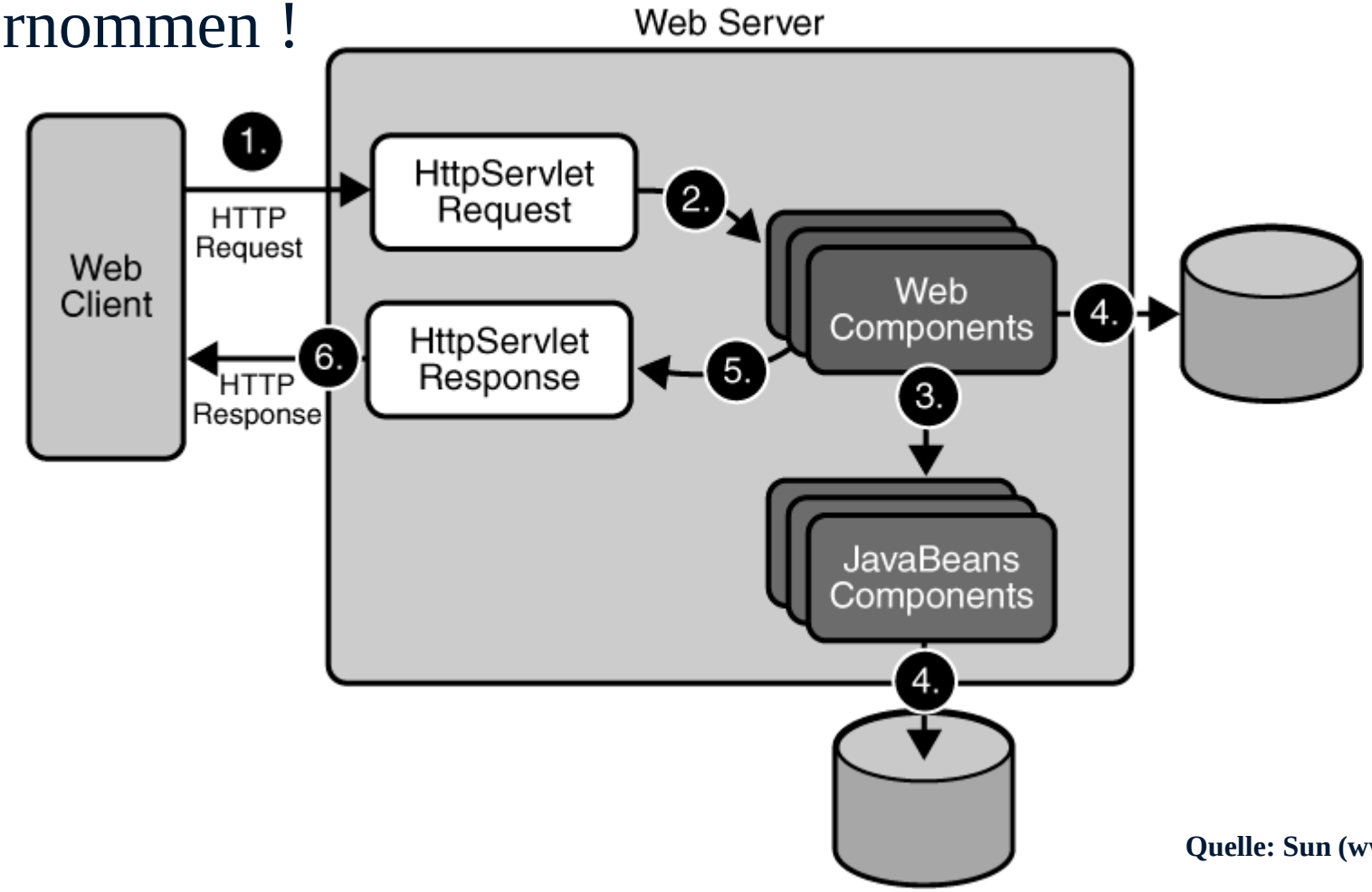
- Auf der Basis von JEE und EJB können komplexe Applikationen in Schichten (Tiers) aufgeteilt werden.



Quelle: Sun (www.sun.com)

Webapplikationen mit Java

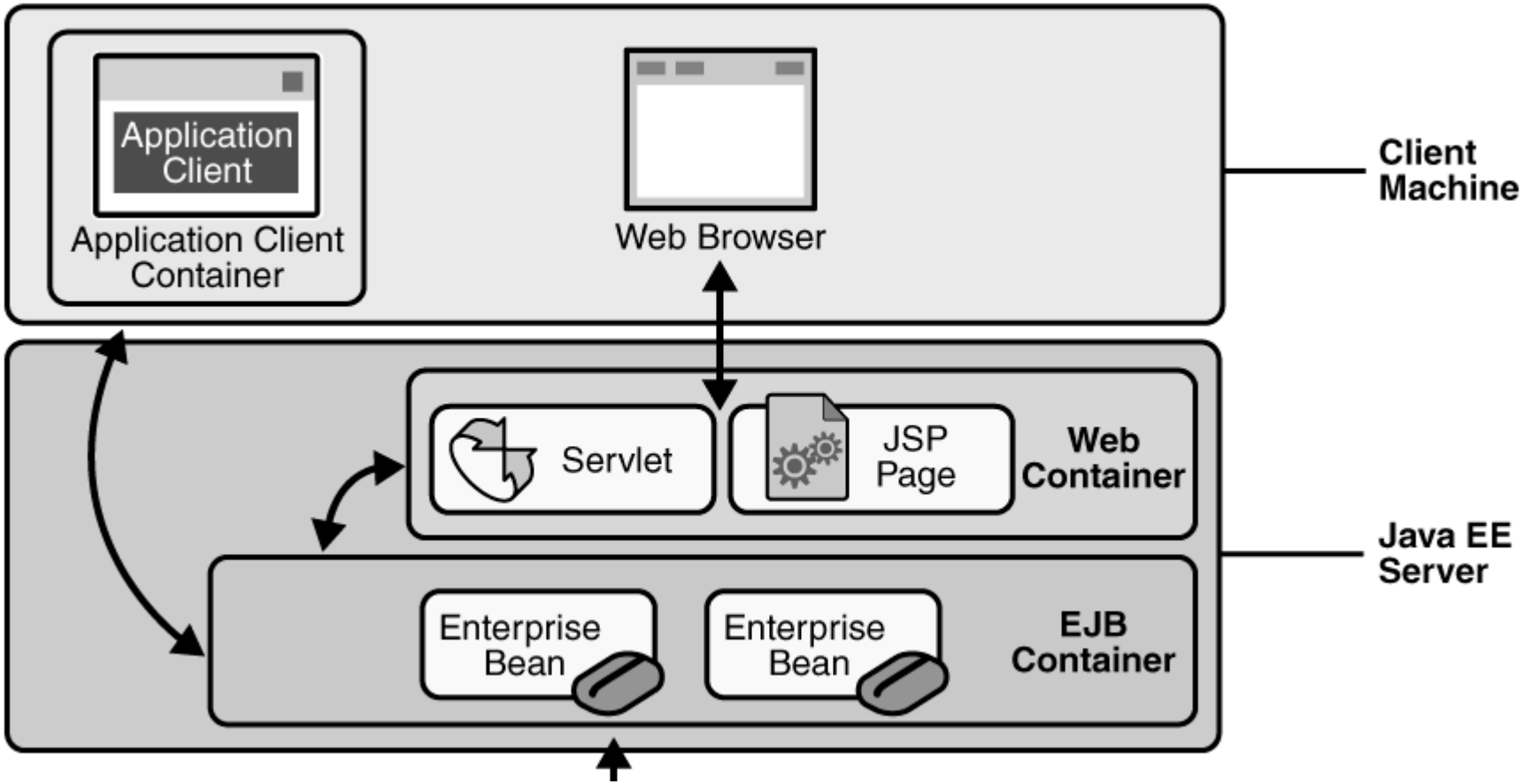
- Die Kommunikation mit dem Web-Client verbleibt beim Servlett oder JSP.
- Die Backendkommunikation wird von den Beans übernommen !



Quelle: Sun (www.sun.com)

EJB – Container

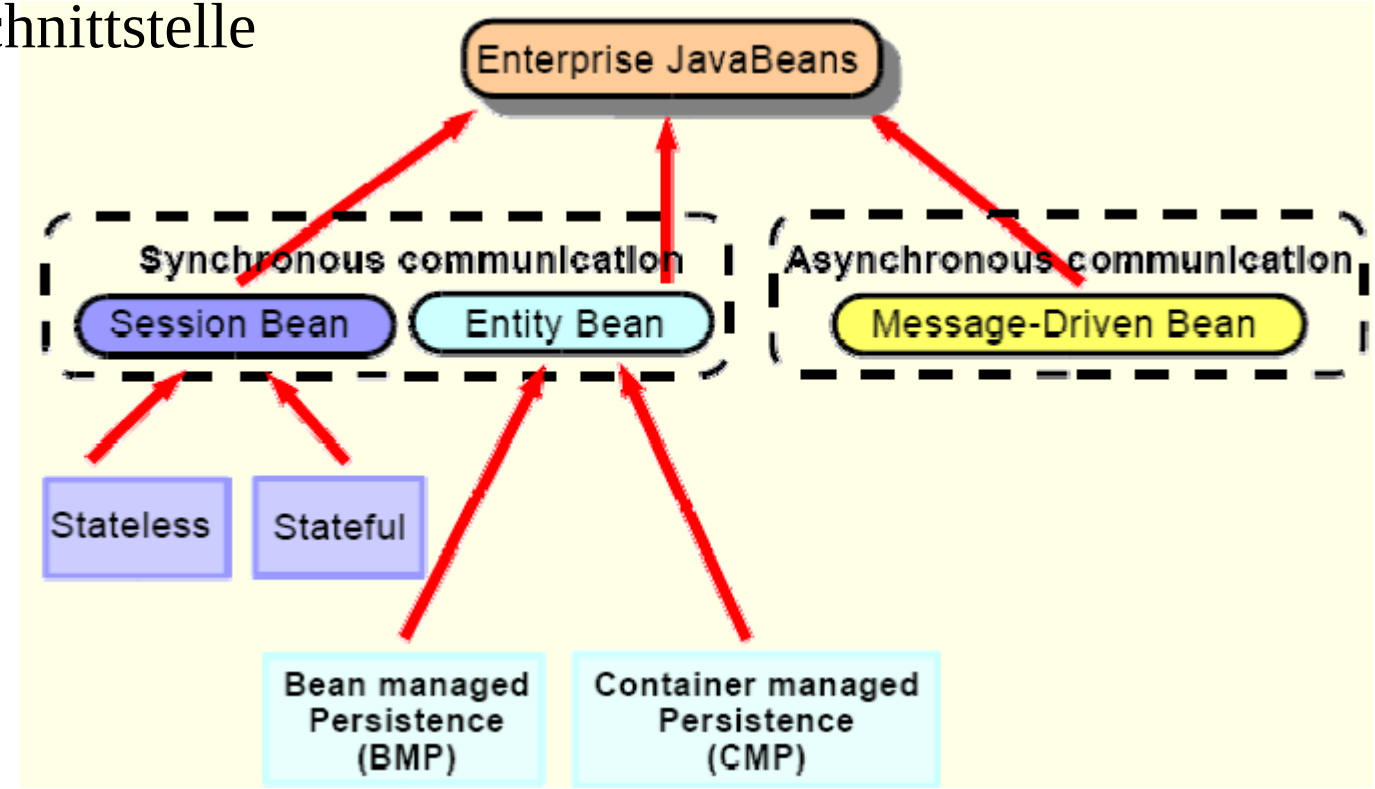
- Container umfassen eine oder mehrere Komponenten und sichern die Transaktionen ab.
- Die Container wiederum laufen in einem Java EE-Applikationsserver.



Enterprise Java Beans - Typen

Typen von Enterprise Java Beans :

- SessionBeans : repräsentiert eine Komponente für die Dauer einer Session und realisiert bestimmte Aufgaben
- EntityBeans : kapselt eine Entity, z.B. eine Datenbanktabelle oder ähnliche Ressource
- MessageDrivenBeans : agiert als async. Dienst (listener) für eine Message-Schnittstelle



Enterprise Java Beans – State-Typen

Stateful Session Beans :

- In a *stateful* session bean, the instance variables represent the state of a unique client-bean session. Because the client interacts (“talks”) with its bean, this state is often called the *conversational state*.
- The **state is retained for the duration of the client-bean session**. If the client removes the bean or terminates, the state disappears.

Stateless Session Beans

- A *stateless* session bean **does not maintain a conversational state** with the client.
- When a client invokes the method of a stateless bean, the bean’s instance variables may contain a state, but only for the duration of the invocation.
- Stateless session beans are **never written to secondary storage** !

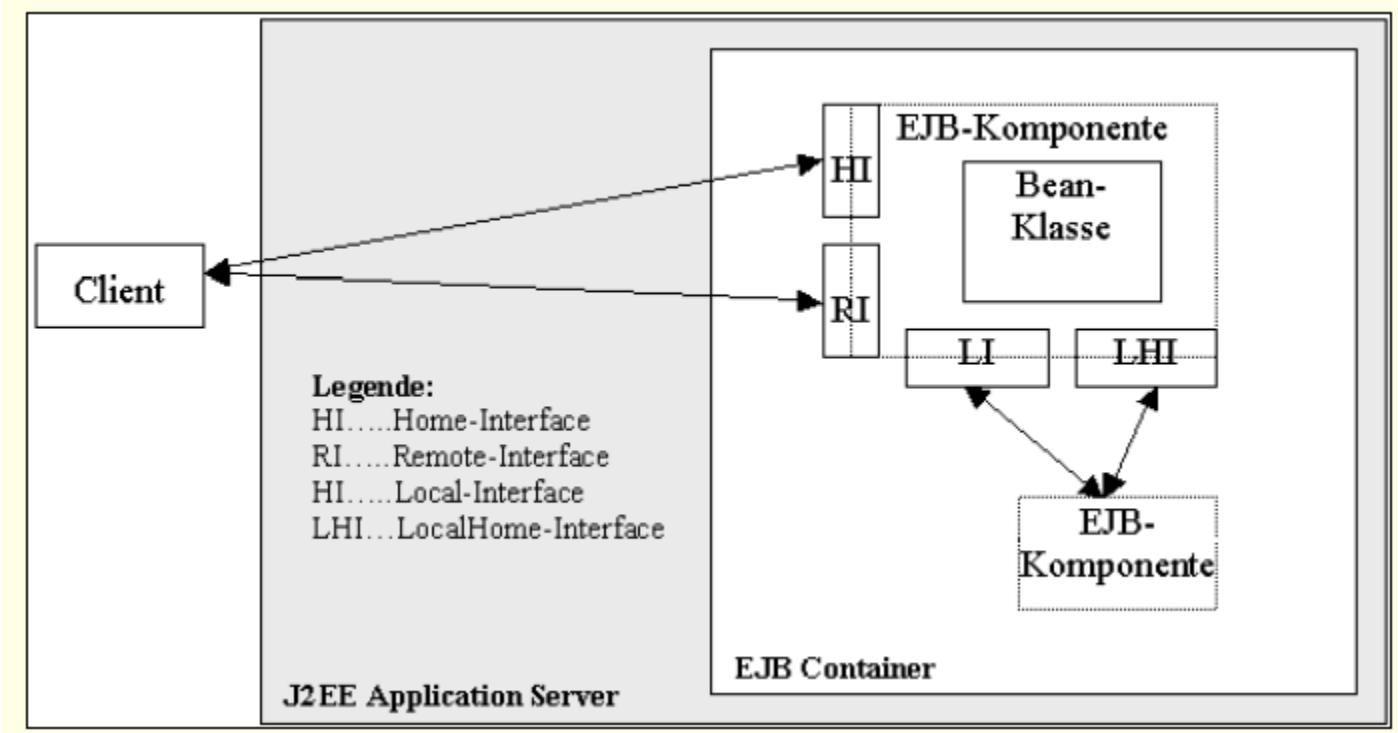
Bewertung und Vergleich :

- stateless session beans can support multiple clients and offer better scalability for applications that require large numbers of clients.

Enterprise Java Beans – Aufbau

Eine Enterprise Java Bean besteht aus:

- Bean-Klasse (bei EntityBeans + Primärschlüsselklasse)
- 2 bis 4 Interface + evtl. eigene Interfaces
- XML-Deploymentdeskriptor



J2EE konforme Application Server

J2EE konforme Application Server:

- Sun Glassfish–J2EE-Opensource http://glassfish.dev.java.net
- IBM Websphere
- Bea Weblogic
- Sybase Enterprise Application Server (Jaguar)
- Oracle Application Server
- JBoss (Open Source Project)

Aufgaben

- Pooling und Caching
- AOP - Aspekt orientierte Programmierung (XML)
- Vorteile: verteilter Zugriff, zentral, relativ performant,
- flexibel (portabel, kompatibel, wartungsarm)

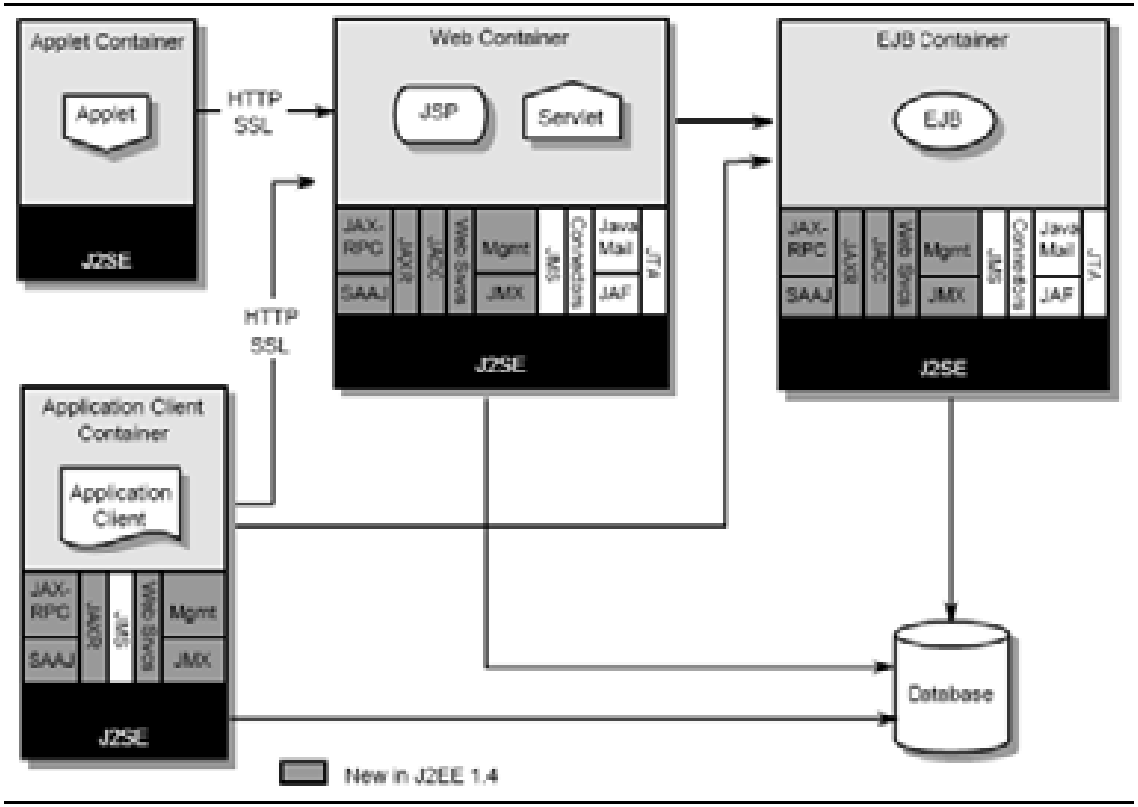
Nachteile:

- höhere Netzlast
- sehr stark steigende Komplexität (1200 Seiten Dokumentation)

EJB – Containerarten

Containerarten

- Web Container
- EJB Container
- Applet Container
- ApplicationClient



Verwaltung von Containern

- Deployment = „installieren“ von Applikationskomponenten – gegenwärtig meist über XML-Konfigurations-dateien
- Komponenten in Archiven zusammen-gefasst : JAR, WAR, EAR, RAR, SAR

Anschluß von Datenbanken unter EJB / J2EE

(unter dem **Stichwort** Persistence (dt. Fortdauern / Speichern) gelistet)

Anschluß von Datenbanken unter EJB / J2EE :

- JDBC als Ausgangspunkt (aber leider sehr unterschiedliche Komplexitätsstufen => Impedance Mismatch...)
- über EJB Entity Beans (=> sehr großer Aufwand => z.B. über 300 Klassen für relativ kleine Website)

Alternative Persistence Frameworks (teilweise auch in J2SE nutzbar)

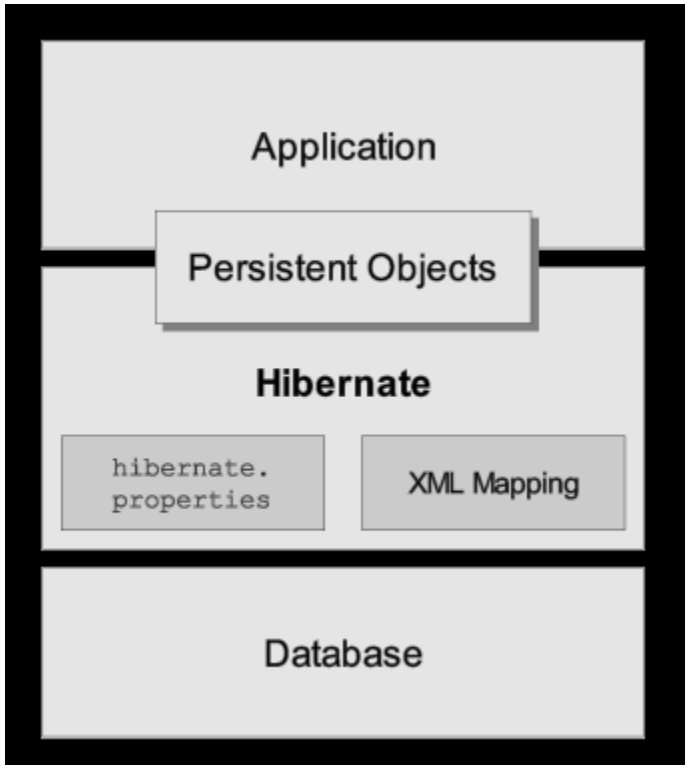
- JDO - Java Data Objects (Sun API)
- Open Source Projekte
  - Hibernate (JBoss Subproject)
  - OJB - Object Relational Bridge (Apache Subproject)
- Oracle Technologien
  - TopLink
  - BC4J - Business Components for Java -> OC4J

Datenbankinterface Hibernate

- Hibernate ist ein Open-Source-Persistenz-Framework für Java
- (für .NET ist eine Version NHibernate verfügbar)
- Entwickelt von JBOSS (Red Hat) - http://www.hibernate.org

- Hibernate mappt Datenbanken auf entsprechende Java-Strukturen :

Java	**DB**
Klasse <->	Tabelle
Objekt <->	Tabellenzeile
Attribut <->	Tabellenspalte

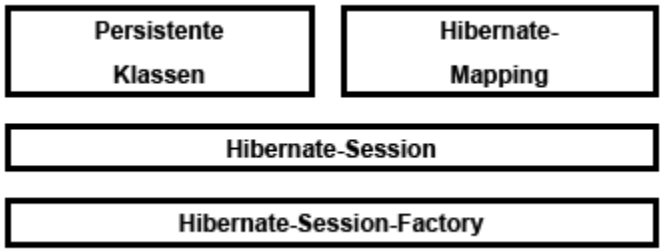


Datenbankinterface Hibernate II

- Das Hibernate – Framework
  - die erzeugten Java-Strukturen sind POJOs (*Plain Old Java Objects*) – also ganz „normale “ Java-Objekte
  - ist portabel bzgl. der Datenbanken !!! (wesentlicher Vorteil)
  - generiert SQL-Anweisungen
  - verfügt über HQL (Hibernate Query Language) –ähnlich SQL
  - unterstützt Transaktionen, Pooling, Caching

Die Hibernate-Architektur

- Session-Factory : einzelne Instanz, erzeugt HB-Sessions, hält DB-Connection (ggf. DB-Cache)
- HM-Session: Lesen und Schreiben in DB mit den Methoden - Session.load(Class, Object); .delete(Object) .createQuery(String), .save(Object)
- HM-Mapping : XML-Mapping von Java und Datenbank-Struktur



Datenbankinterface Hibernate - Beispiel – Basis

```
public class Event
{ private Long id; private String title;
   private Date date;
    public Event() {}
    public Long getId() { return id; }
    void setId(Long id) { this.id = id; }
 //  ... Weitere Setter / Getter für die anderen Attribute
```

Java-Klasse

Hibernate-Mapping

```
<hibernate-mapping>
  <class name="events.Event" table="EVENTS">
      <id name="id" column="EVENT_ID">
      <generator class="native"/> </id>
      <property name="date" type="timestamp" column="EVENT_DATE"/>
 <property name="title"/>
</class> </hibernate-mapping>
```

Datenbankinterface Hibernate - Beispiel - Speichern

```
public class EventManager {
    public static void main(String[] args) {
        EventManager mgr = new EventManager();
        if (args[0].equals("store")) {
            mgr.createAndStoreEvent("My Event", new Date());
 }    HibernateUtil.getSessionFactory().close();   }

    private void createAndStoreEvent(String title, Date theDate)
     {  Session session =
     HibernateUtil.getSessionFactory().getCurrentSession();
        session.beginTransaction();
        Event theEvent = new Event();
              theEvent.setTitle(title);          theEvent.setDate(theDate);
        session.save(theEvent);  // generiert SQL-Anweisungen
        session.getTransaction().commit();  // führt Transaktion aus ...
} }
```

Quelle und weitere Bsp.: http://www.hibernate.org/hib_docs/v3/reference/en/html/

Framework Jakarta Struts

Jakarta Struts ist ein **Framework**, das viele

- Standardaufgaben von JSP/Servlet-Anwendungen bereits enthält.
- Mit dem Framework werden häufig wiederkehrende Aufgaben bereits als **Gerüst** bereitgestellt !
- Bereits seit 1998 in der Entwicklung und relativ gut eingesetzt !

Unterstützung für :

- Mehrsprachigkeit, Internationalisierung
- Fehlerbehandlung
- einfacher Zugriff auf Request Parameter
- viele spezielle HTML-Tags für Formulare
- Umfangreiche Konfigurierungsmöglichkeiten (aber viele XML Dokumente, teilweise sehr komplexe und aufwändig !!!)

Jakarta Struts ist ein Open Source Produkt

- => http://jakarta.apache.org/struts

Java Server Faces (JSF)

Java Server Faces ist ebenfalls ein **Framework** :

- basierend auf Sun API
- stellt eine API zur Erstellung von GUI-Komponenten :
- Entwicklungsfirmen sollen API für GUI-Komponenten nutzen
- Anwendungsentwickler sollen mit dieser API Komponenten ansprechen
- Serverseitig (sind keine Applets => ALLE Nachteile von HTML)
- Bibliothek von wiederverwendbaren GUI-Komponenten
- Möglichkeit eigene GUI-Komponenten zu schreiben
- unabhängig von Protokollen und Markup Languages
- Event-Modell
- Verarbeitung der Formulare
- Sessionverwaltung

Vergleich der Frameworks

Jakarta Struts

- wird zur Zeit am häufigsten eingesetzt und funktioniert stabil
- konzentriert sich auf den Controllerteil
- basiert in einigen Kerneigenschaften auf Entwürfen von 1998 und gilt als leicht veraltet und weniger flexibel

Java Server Faces

- entstand ab 2004 als Antwort auf die Kritik an Java Struts
- Höhere Flexibilität durch besseren Austausch von Renderkomponenten und anderen zentralen Sub-Komponenten

Empfehlung aus IX 10/2005 (S. 133)

- für kritische Projekte mit wenige Spielraum -> Struts
- bei mehr Zeit für Tests und strategischer Ausrichtung -> JSF
- Bei kleinen UND kritischen Projekten (Kosten!) sind Alternativ-Frameworks wie Springs eventuell sinnvoller !

Neue, leichtgewichtige Frameworks

Warum noch mehr Frameworks ?

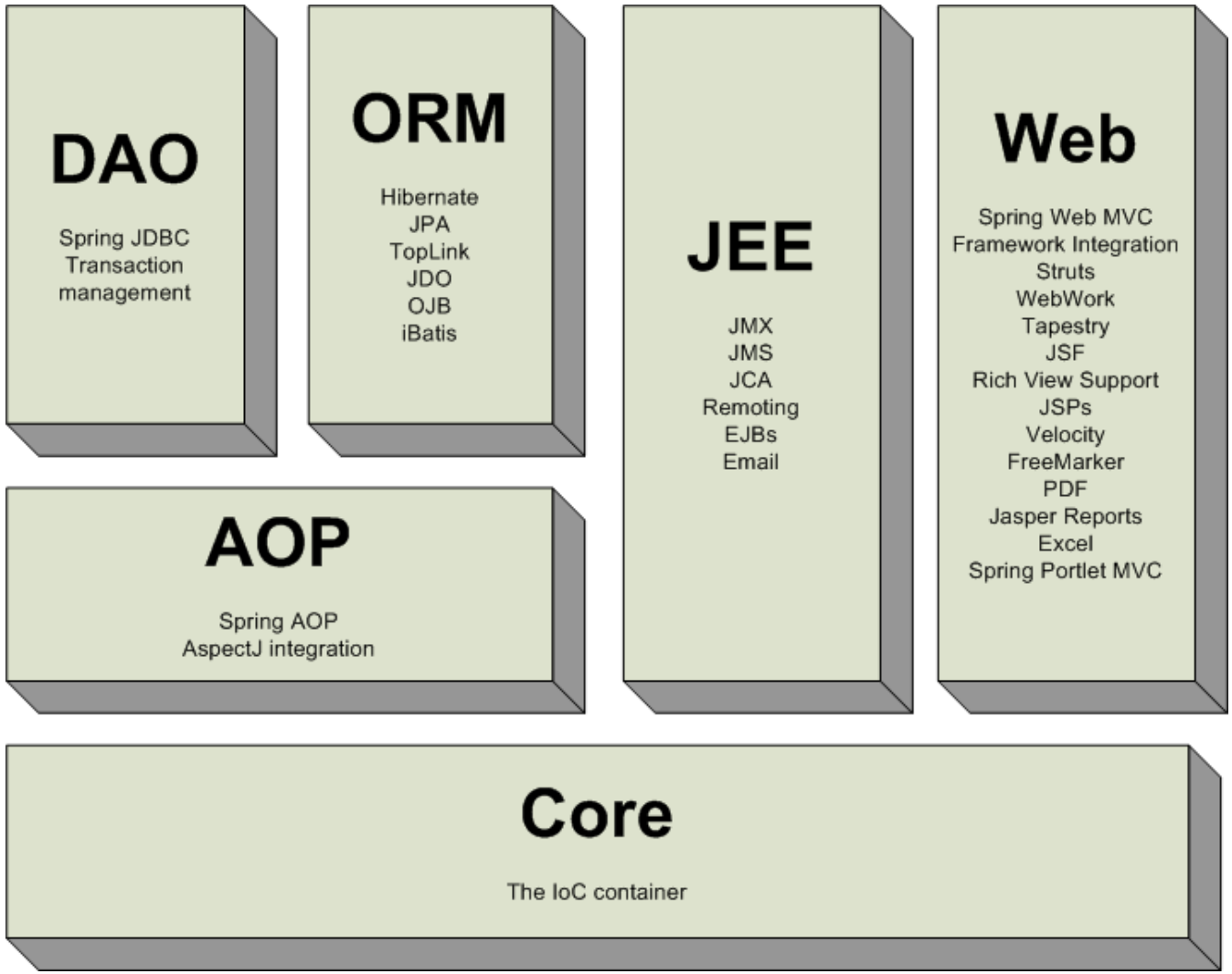
- Java EE gilt mittlerweile als kompliziert
- Projekte müssen viel Rücksicht auf Technologie nehmen
- Entwickler zu sehr mit API‘s beschäftigt anstatt mit Geschäftslogik
- Anzahl der API‘s und Bibliotheken sehr groß – siehe 20 API´s unter http://de.wikipedia.org/wiki/J2ee
- Keine Möglichkeit infrastrukturelle Dienste der Container anzupassen (Bsp.: EJB)
- Komplett neue Ansätze mit Ruby on Rails / Grails

Aktuelles Projekt im J2EE-Umfeld - SPRING

- erster Ansatz bereits 2002 von Rod Johnson auf der Basis seines Buchs "Expert One-On-One J2EE Design and Development“
- 2003 erste Release in der Version 0.9 unter der Apache 2.0 Lizenz, Aktuell Entwicklung durch die Firma SpringSource
- Download : http://www.springframework.org/

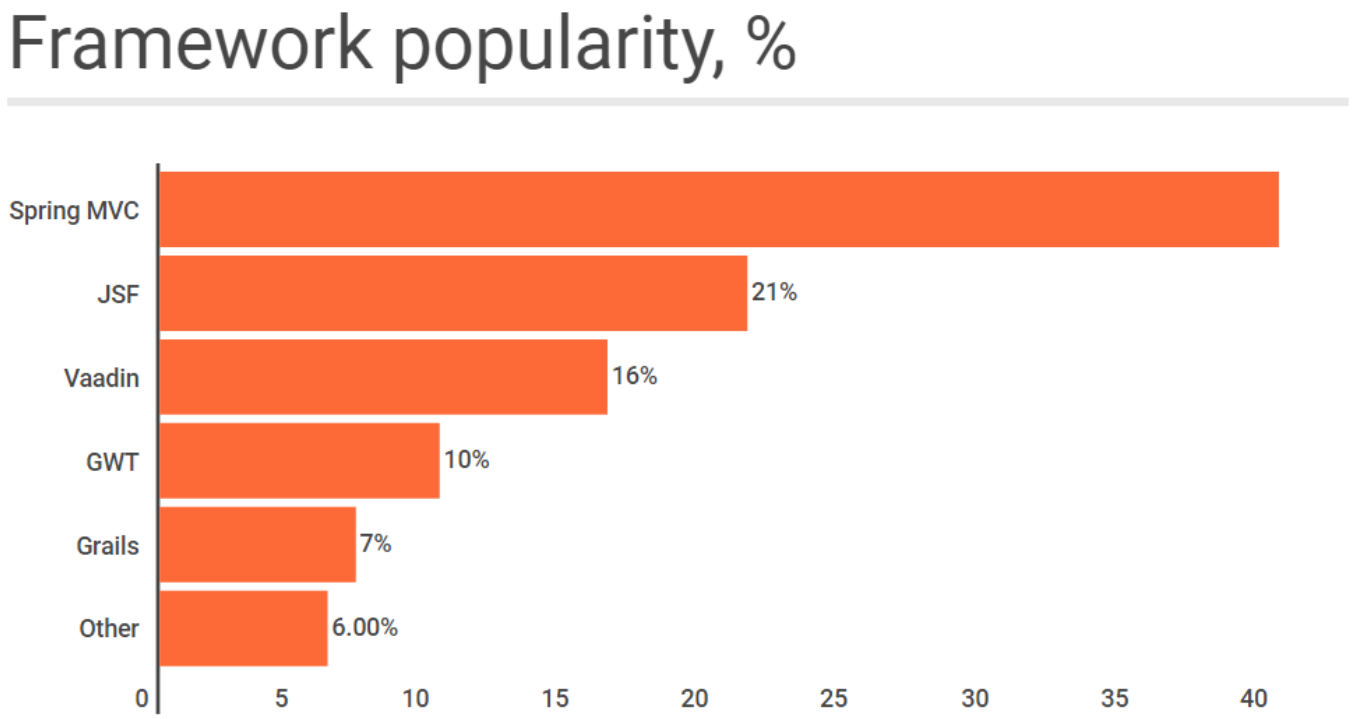
Spring - grundlegende Eigenschaften und Architektur

- Verknüpfung mehrerer unabhängiger Sub-Frameworks
- konsistente Architektur mit modularen Aufbau
- Minimierung der Komponentenabhängigkeiten (Inversion of Control (IoC)



Anwendungsvergleich der Java-Frameworks

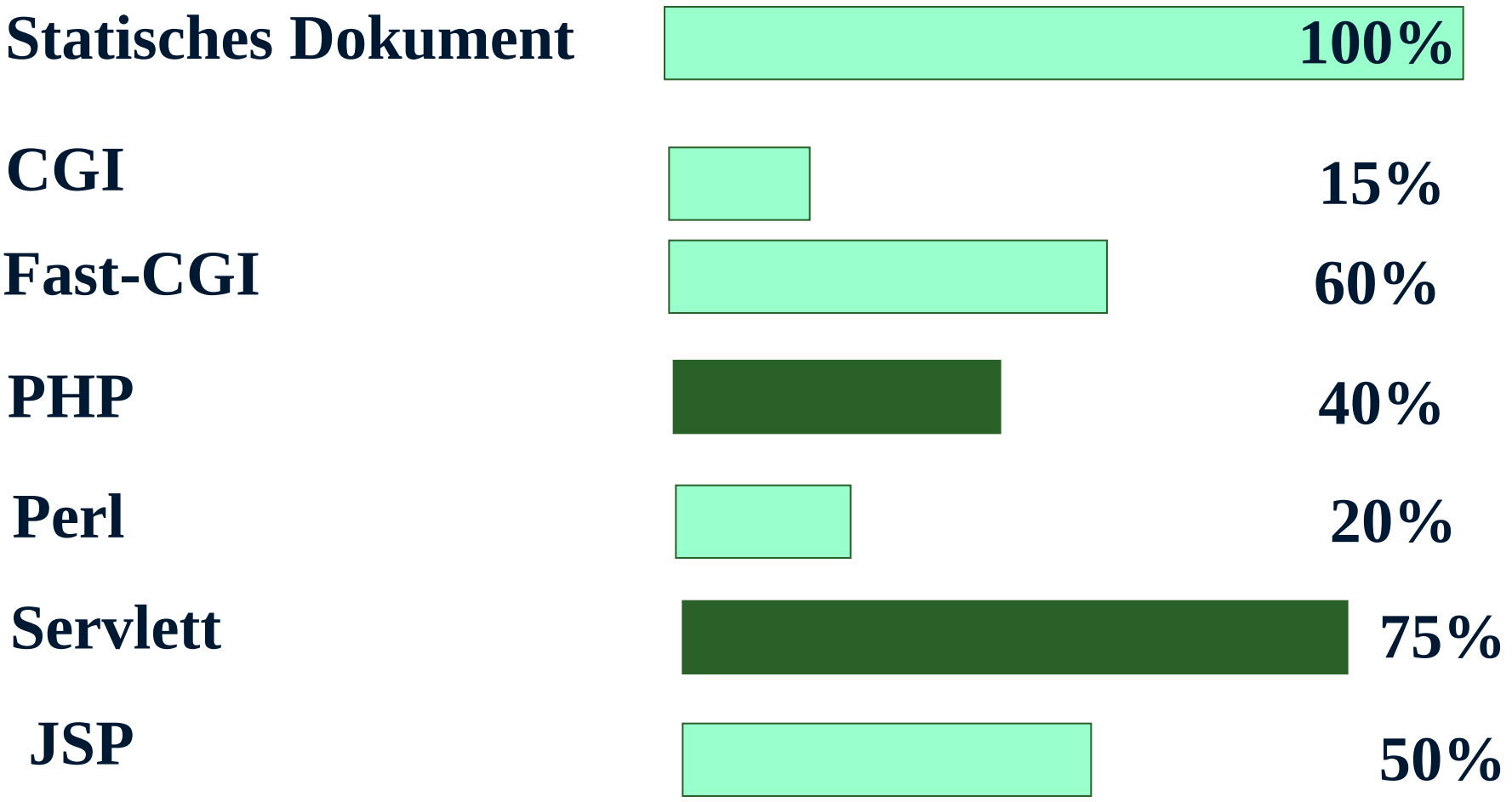
- Spring MVC und JSF Favoriten (2016)
- Bei komplexen Frameworks wird die Anpassbarkeit der Basiskomponenten (speziell der GUI-El.) als kritisch eingeschätzt



Quelle: https://www.romexsoft.com/blog/7-best-java-frameworks-for-2016/

Vergleich der Java Technologien mit anderen Optionen

- Vergleich der Performance bei der Auslieferung eines statischen Dokumentes (angesetzt mit 100%) - nach IX 10/2005



Zusammenfassung und Ausblick

Java 2 Enterprise Edition

- stark zunehmende Kritik an vergangenen Releases bzgl. stetig anwachsenden Entwicklungsaufwandes in den letzten Jahren
- letzte Releases scheinen wieder etwas zurück zu rudern:
  - vieles wird einfacher und zu besser überschaubaren Einheiten zusammengefasst
  - Dokumentation geht (wahrscheinlich) wieder unter 1000 Seiten !

Fazit :

- Wenn die Komplexität beherrscht wird, so sind Projekte mit JavaEE im Vergleich zu anderen Technologien (PHP / Perl etc.) deutlich leistungsfähiger und strategisch aussichtsreicher !
- **Für kleine Projekt und schnelle Umsetzungen einfacher Websites ist JavaEE zu groß und zu aufwändig**
- Bei einer Bereitstellung von PHP-ähnlichen Skriptingmöglichkeiten mit fast gleicher Performance wie Servletts könnte dies anders sein !!
- => das GRAILS –Framework könnte eine Option sein !