

Client/Server-Systeme

Prof. Dr.-Ing. Wilhelm Spruth

SS 2006

Teil 13

Web Application Server

Literatur

**Dustin R. Callaway: „Inside Servlets“.
Addison Wesley, ISBN 0201379635**

**<http://edocs.bea.com/>
(BEA Weblogic Dokumentation)**

**<http://www-306.ibm.com/software/websphere/sw-library/#Books>
(sehr große Anzahl von Lehrbüchern, welche vom Web heruntergeladen werden können)**

**<http://www.oio.de/sap-netweaver-components.htm>
(SAP Netweaver Dokumentation)**

**Berstein P., Hadzilacos V.: Goodman N., *Concurrency Control and Recovery in Database Systems*.
<http://research.microsoft.com/pubs/control>
(Dies ist ein vollständiges Buch, welches vom Web heruntergeladen werden kann)**

http://research.microsoft.com/~gray/hpts99/talks/Gray_Jim.ppt

HTTP 1.1 Spec: <http://www.ietf.org/rfc/rfc2616.txt>

HTML 4.01 Spec: <http://www.w3.org/TR/html401>

CGI/Perl Tutorial: <http://www.cgi101.com/class>

Apache User's Guide: <http://www.apache.org/docs>

Web Application Server

Plattform für die Ausführung von
Java Servlets, Java Server Pages und EJBs

BEA Web Logic
IBM WebSphere
SAP Netweaver



verfügbar auf allen Plattformen
Windows, Unix, Linux, z/OS

Mehr als 2/3 aller in der Wirtschaft eingesetzten Web
Application Server. Verwenden die Firmen-eigenen
Transaktionsmonitore:
Tuxedo, CICS, SAP/R3 Tx Monitor

Public Domain

JBOSS, Tomcat

Java Ausprägungen

Java Script

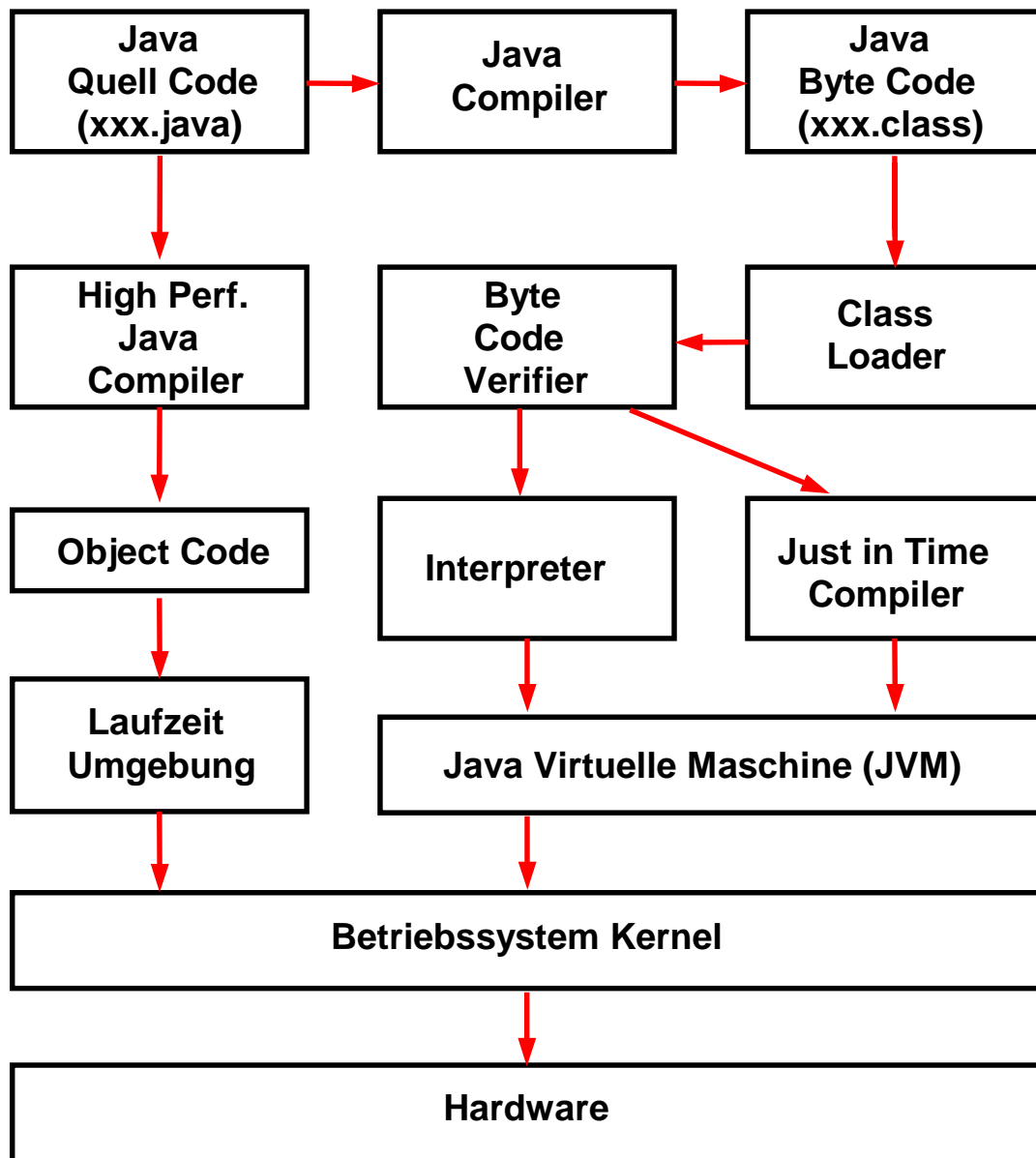
Java Applet

Java Servlet

Java Server Pages

Java Bean

Enterprise Java Bean



Java Virtuelle Maschine

JavaScript

JavaScript ist eine Scripting Language

Nicht Teil von Java, der Java Virtuellen Maschine oder des Java tool set.

Object basiert, hat keine Klassen und keine Vererbung

Ursprünglich als Erweiterung von HTML definiert, für Code, der in ein HTML Dokument eingebettet, und "on the fly" ausgeführt wird.

Beispiel:

```
<script>  
function calculate(obj) { .....  
.....  
}  
</script>
```

JavaScript Programme können auf dem Klienten oder auf dem Server laufen.

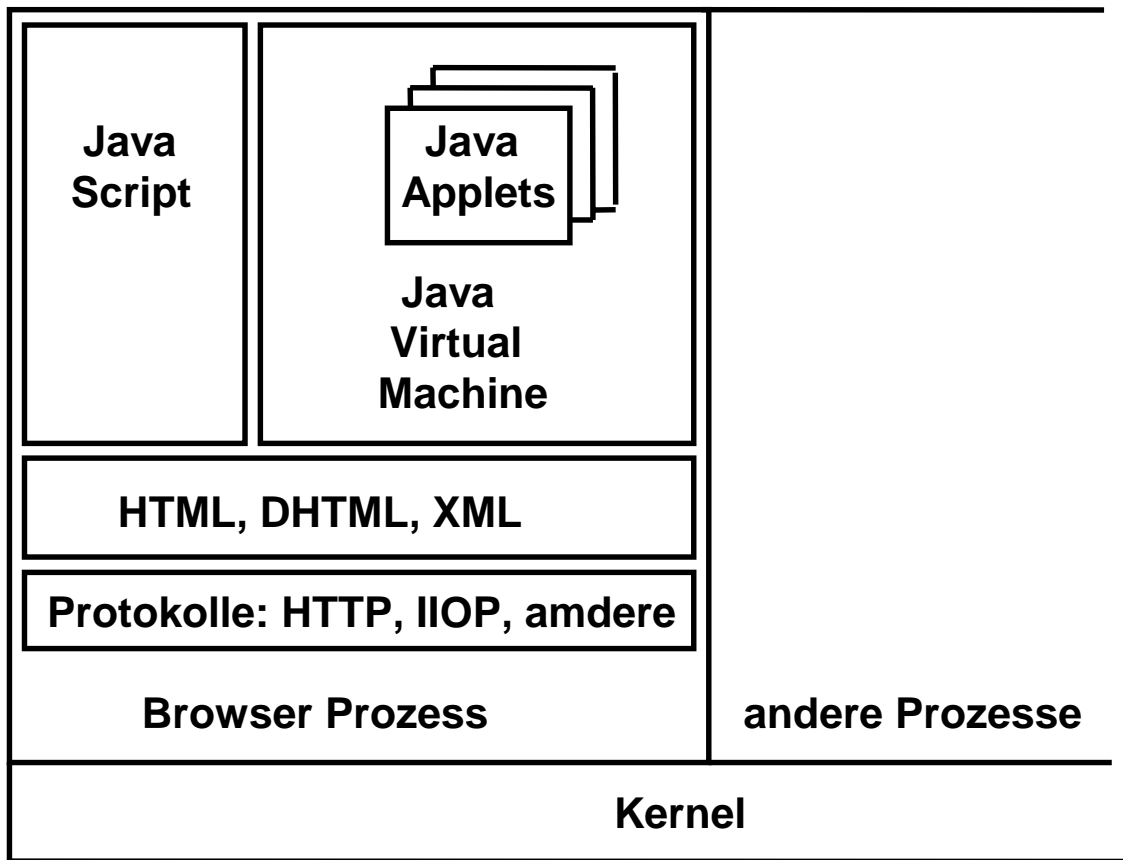
Java Applet

Unterschiede zu Java Script

- **Volle Java Funktionalität**
- **Läuft innerhalb der Java Virtuellen Maschine**

Wird als Teil der HTML Seite über das HTTP Protokoll in den Klienten geladen

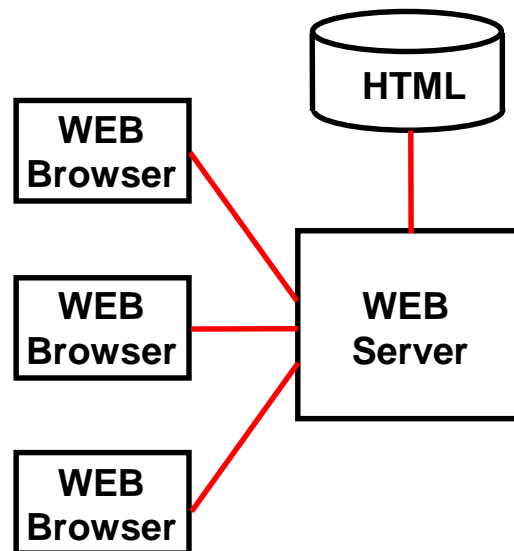
- **Kann eingesetzt werden, um für die nachfolgende Kommunikation zwischen Klienten und Server ein anderes Protokoll zu benutzen, z.B. IIOP, RMI, 3270**



Client Rechner

Der Browser ist ein standardisierter Client Prozess, der mit dem Server über Protokolle wie HTTP oder IIOP kommuniziert und Ausgabedaten des Servers im HTML oder XML Format empfängt.

Ein Browser ist leicht zu installieren und erfordert wenig administrativen Aufwand. Es existiert ein Trend, Klienten-Rechner nur mit einem Browser und keinen anderen Anwendungen auszurüsten (Thin Client).



Java und der Web Browser

HTML Seiten werden zum Web Browser mit Hilfe des HTTP Protokolls übertragen.

HTTP wird auch als „Web RPC“ betrachtet. Wie der eigentliche RPC ist HTTP zustandslos: Request/Response. Keine Session.

Erlaubt die Übertragung selbstbeschreibender Daten. Bei jeder Verbindungsaufnahme müssen Datenformate neu ausgehandelt werden.

Beispiele für Schicht 5 Protokolle:

Telnet, FTP, 3270, HTTP, SOAP, IIOP, RMI/JRMP,

HTTP Protokoll

**HTTP ist ein „connectionless“ und ein „stateless“
Protokoll. Ein Web Zugriff erfolgt in 4 Schritten:**

- 1. Klient öffnet eine Verbindung mit den (Web) Server,
benutzt hierfür TCP (connection oriented),
Socket Verbindung nach Port 80.**
- 2. Klient sendet eine Request an den Server.**

Wird z.B. das URL

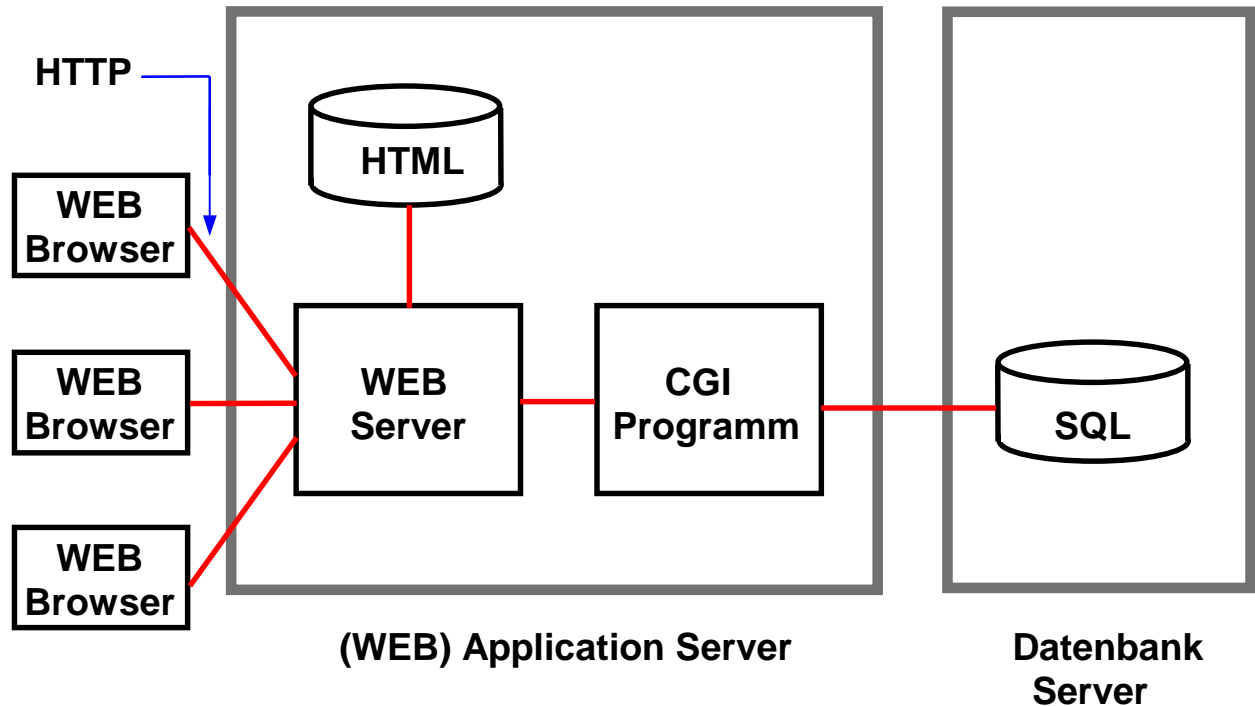
www.qrx.de

**eingegeben, so sendet der Klient (Browser) die
HTTP Nachricht**

**GET /index.html HTTP/1.0
User-Agent: Mozilla/4.7 [en] (Win2000; I)
Accept: image/gif, image/jpeg, */***

an den Server.

- 3. Der Server sendet eine HTML Seite als Antwort.**
- 4. Die TCP Verbindung wird geschlossen.**



Dynamischer WEB Seiten Inhalt (1)

Der Web Browser kommuniziert mit dem Web Server über das HyperText Transfer Protokoll (HTTP). HTTP ist das ursprüngliche Transport Protocol für das World Wide Web.

Zwei Alternativen:

1. Web Server sendet statische Seite aus der HTML Datenbank an den Web Browser.
2. Web Server erstellt dynamische Seite. Hierzu ruft er z.B. über die CGI Schnittstelle ein Anwendungsprogramm auf. Dieses kann z.B. Daten aus einer Datenbank verwenden, um eine dynamische HTML Seite zu erstellen. 60% aller derartigen Daten werden in z/OS Datenbanken gehalten.

CGI Programme werden häufig in einer Script Sprache, z.B. PERL, PHP, REXX oder Tcl/Tk erstellt, können aber auch in einer beliebigen anderen Sprache, z.B. C++, Cobol oder Java geschrieben werden.

Login to Secure Site

Username:

Password:

Submit

Clear

HTML Form

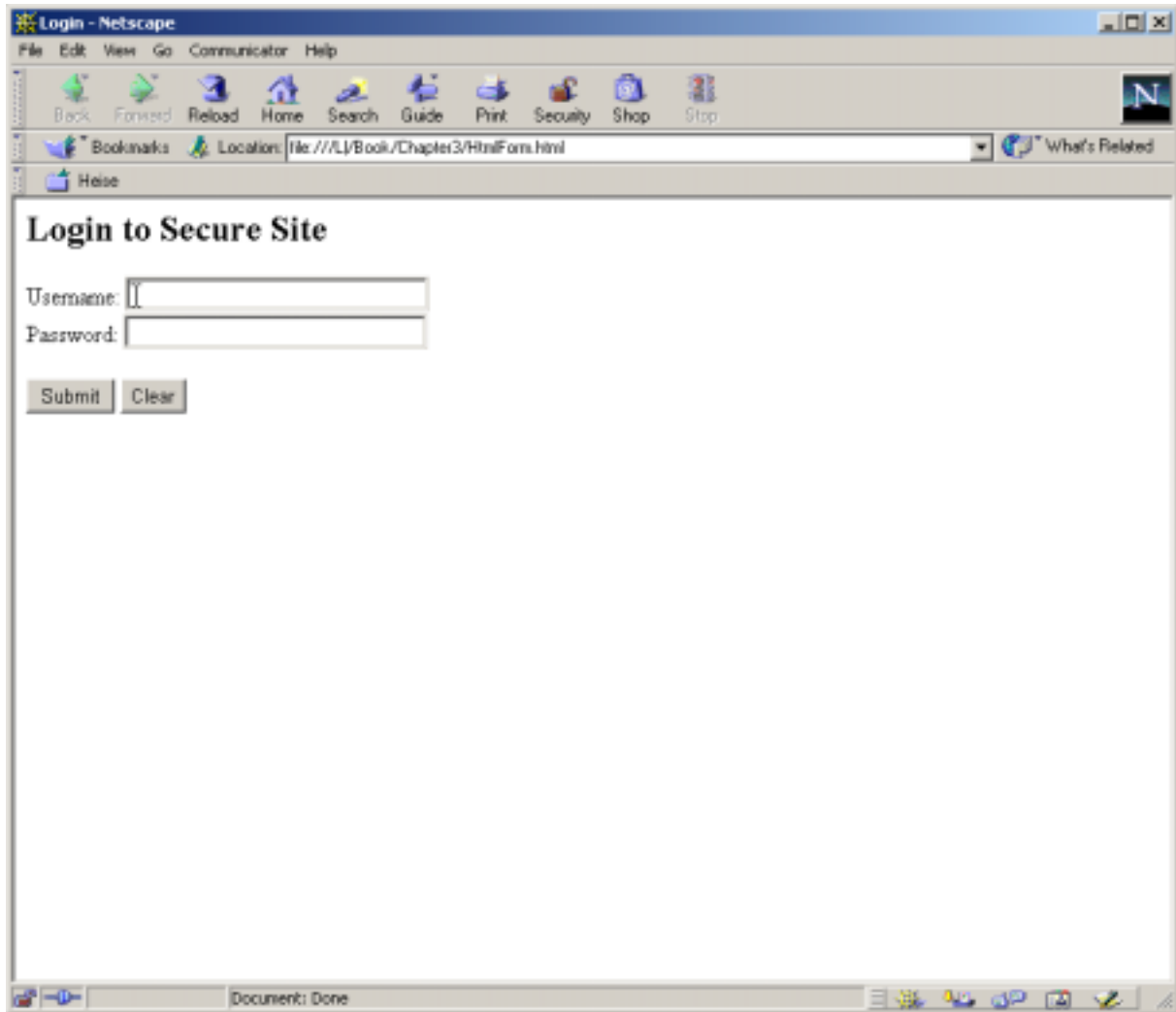
```
<HTML>
<HEAD><TITLE> Login </TITLE> </HEAD>
<BODY>
<H2>Login to Secure Site</H2>
```

```
< FORM METHOD=POST
ACTION="http://abc.de/servlet/xyz.servlet" >

Username: <INPUT TYPE="TEXT" NAME="username"
SIZE="25"><BR>
Password: <INPUT TYPE="PASSWORD"
NAME="password" SIZE="25"><P>

<INPUT TYPE="SUBMIT" VALUE="Submit">
<INPUT TYPE="RESET" VALUE="Clear">
</FORM>
```

```
</BODY> </HTML>
```



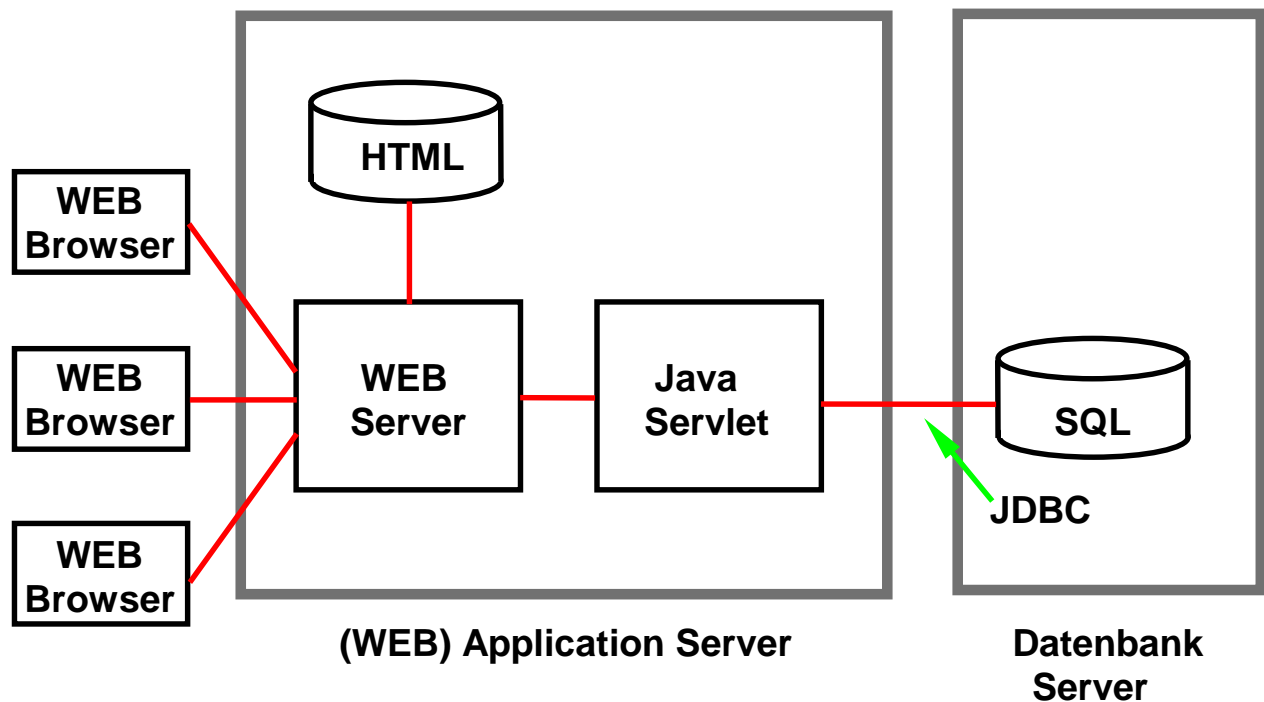
GET /login.html HTTP/1.0
User-Agent: Mozilla/4.7 [en] (Win2000; I)
Accept: image/gif, image/jpeg, */*

GET /login.html?username=Dieter&password=pluto HTTP/1.0
User-Agent: Mozilla/4.7 [en] (Win2000; I)
Accept: image/gif, image/jpeg, */*

http://www.xyz.com/login.html?username=Dieter&password=pluto

POST /abc.html HTTP/1.0
User-Agent: Mozilla/4.7 [en] (Win2000; I)
Accept: image/gif, image/jpeg, */*
Content-Length: 34

username=Dieter&password=pluto HTTP/1.0

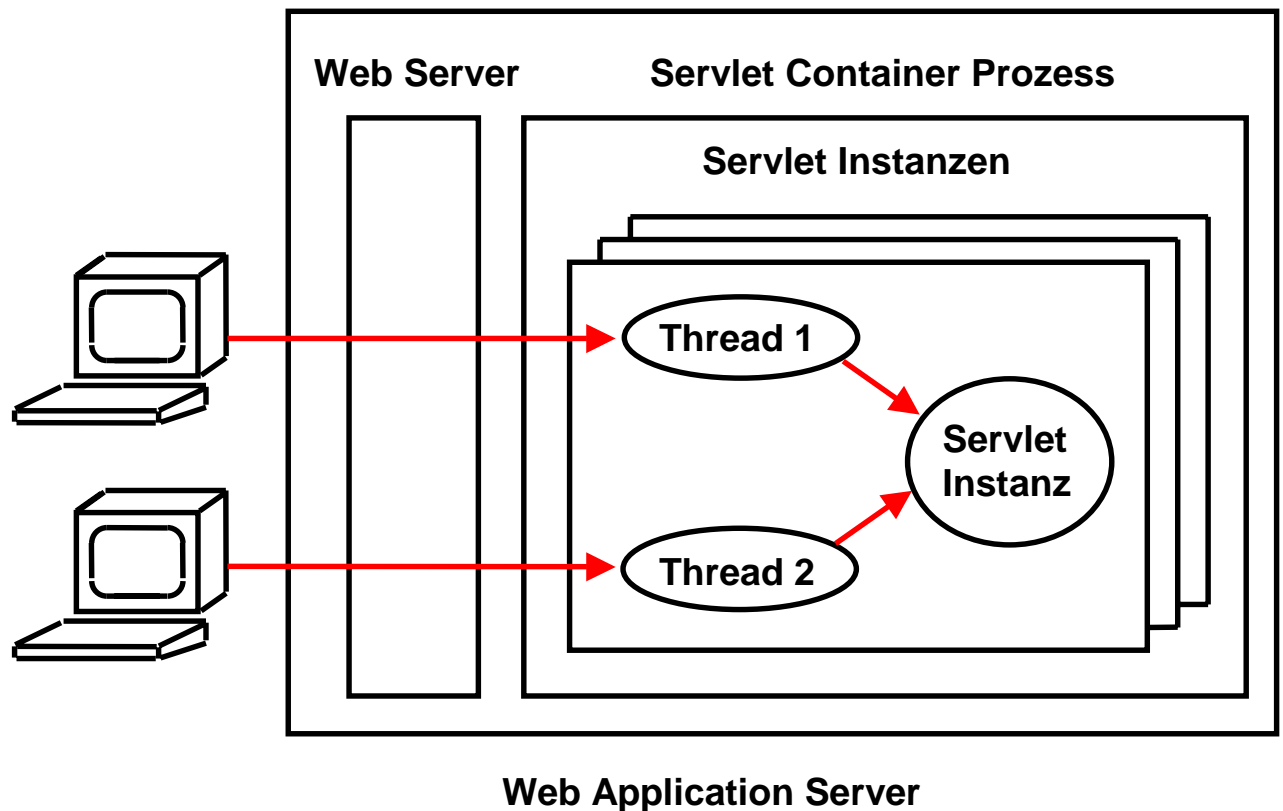


Dynamischer WEB Seiten Inhalt (2)

Im Gegensatz zu CGI erfordert das Java Servlet nur light weight Context Switches. Daher deutlich besseres Leistungsverhalten.

Servlets verfügen über alle Java API's, einschließlich JDBC (Java Data Base Connectivity).

Java Server Pages (JSP) sind eine Erweiterung der Servlet API. Verwenden in Java geschriebene XML - ähnliche Tags und Scriptlets.



Servlet Instanzen und Threads

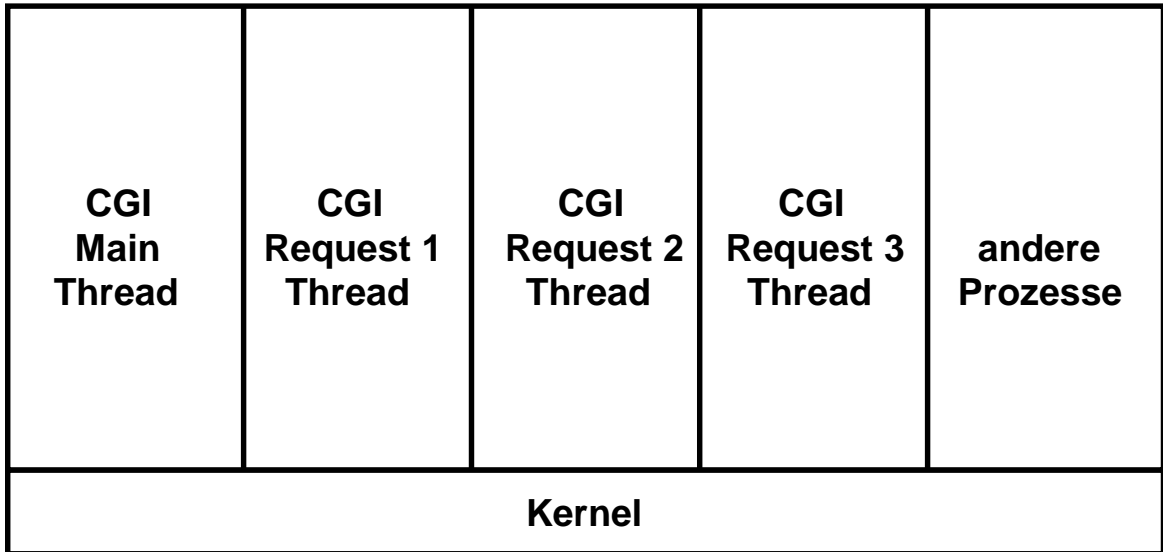
Es existiert nur eine einzige Instanz des Servlets

mehrfache Threads, die mehrere Klienten Requests gleichzeitig befriedigen können

Servlets werden dynamisch geladen, wenn sie erstmalig angefordert werden

oder statisch laden beim Hochfahren des Containers

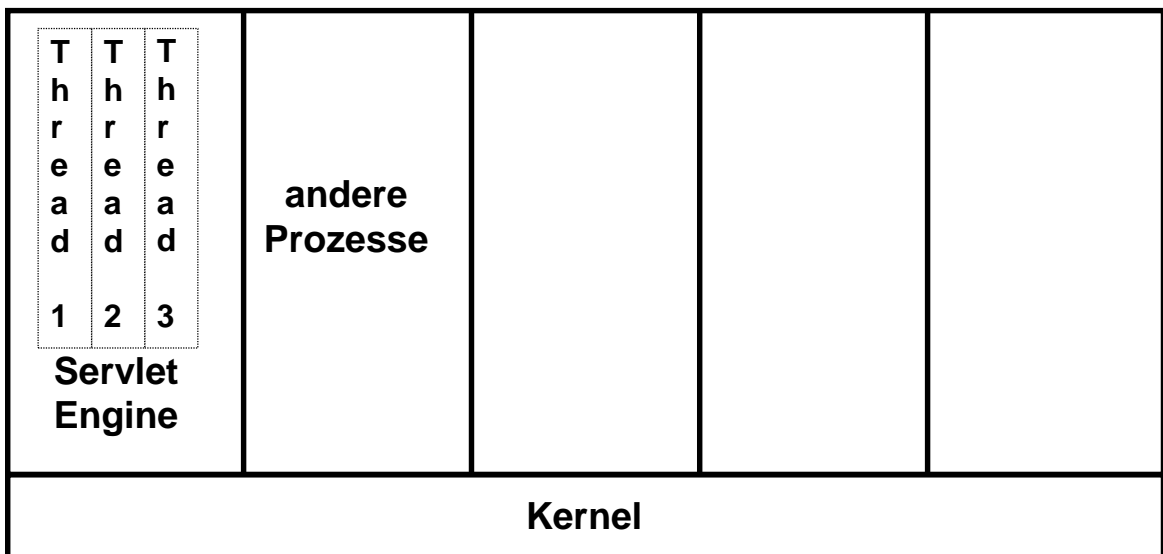
FF..FF



00..00

CGI Ansatz

FF..FF



00..00

Servlet Ansatz

Ein Servlet kann über ein Session Objekt verfügen, welches Session Information über mehrfache Zugriffe verwaltet

Ein Servlet kann einen Pool von Data Base Connections verwalten und eine Data Base Connection über Zugriffsgrenzen aufrechterhalten

Servlet Container

Ein Servlet ist eine reguläre Java Klasse, die über **extends HttpServlet** die Servlet Eigenschaften erbt, und nur in einer Servlet Laufzeitumgebung ausführbar ist. Die Servlet Laufzeitumgebung wird auch als Container oder Servlet Engine bezeichnet.

Ein Servlet Container ist ein Programm innerhalb einer JVM, das Requests für Servlets und Java Server Pages (JSP) behandelt. Der Servlet Container ist verantwortlich für:

- Erstellung von Servlet-Instanzen,
- Initialisierung von Servlets,
- Dispatching von Requests,
- Verwaltung des Servlet-Kontextes für die Nutzung durch die Web-Anwendungen.

Servlet Container haben keine Transactions-, Persistence- und Sicherheitseigenschaften. Sie verbessern u.a. die Servlet-Ausführungszeit.

HelloWeltServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public
class HelloWeltServlet extends HttpServlet
{

    public final static String message = "<html>\n" +
        "<head><title>Hallo Welt</title></head>\n" +
        "<body>\n" +
        "<h1>Hallo Welt</h1>\n" +
        "</body></html>\n";

    public void init()
    {
        System.out.println("In HelloWeltServlet init");
    }

    public void destroy()
    {
        System.out.println("In HelloWeltServlet destroy");
    }

    public void service(ServletRequest req, ServletResponse res)
    throws ServletException, IOException
    {
        PrintWriter out = res.getWriter();
        out.println(message);
    }
}
```

Basic Servlets

Try our basic servlets for common tasks:

- [Echo a request \(RequestInfo\)](#)
- [Query a database \(JDBC\)](#)
- [Process form input \(FormProcessing\)](#)
- [Display form input \(FormDisplay\)](#)

Travel

Are you looking for an exciting vacation? Xtreme Adventures knows which site you were visiting and offers you trips you'll love. And while you're browsing, you can exchange messages with other potential travelers.

WOM Bank

Open accounts, deposit, transfer, or withdraw money. WOM Bank connects to a DB2 database to authenticate customers and maintain transaction information.

Ticket Server

TicketCentral searches pre-loaded DB2 database tables to find available seats at your favorite events. Order your tickets and purchase them with your credit card.

<http://jedi.informatik.uni-leipzig.de/IBMWebAS/samples/>

Servlet Beispiele

Auf unserem OS/390 Server an der Uni Leipzig sind mehrere Lehrbeispiele für Servlets installiert. Der Quellcode dazu kann eingesehen werden. Sie können auf dem OS/390 WebSphere Server <http://jedi.informatik.uni-leipzig.de> mit den folgenden URLs adressiert werden:

<http://jedi.informatik.uni-leipzig.de/servlet/sm390.SMJDBCTestServlet>

<http://jedi.informatik.uni-leipzig.de/servlet/sm390.GuestBookServlet>

<http://jedi.informatik.uni-leipzig.de/IBMWebAS/samples>

Teilweise erfolgt ein Zugriff auf die OS/390 DB2 Datenbank

Für die Erstellung dieser Beispiele existieren ausführliche Tutorials.

Java Server Pages (JSP)

Java Server Pages sind in der Java Programmiersprache geschrieben.

Benutzen XML-artige Tags und Scriplets um die Logik zu kapseln, die den Inhalt der Seite generiert.

Alternativ kann die Anwendungslogik woanders liegen, und die Java Server Page greift hierauf mit den Tags und Scriplets zu.

Trennung der Seiten-Logik vom Seitenentwurf und der Seitenwiedergabe.

Experimentieren sie mit unseren OS/390 Servlets at

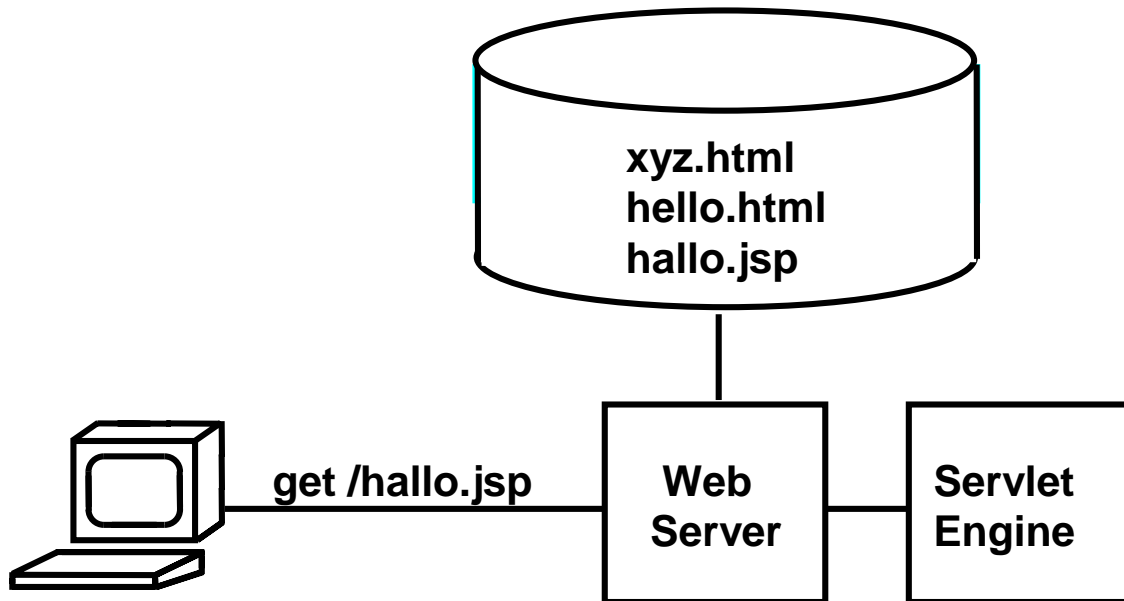
<http://jedi.informatik.uni-leipzig.de/IBMWebAS/samples>

(Vorsicht, Groß- und Kleinschreibung beachten)

siehe auch Tutorials 8 und 9

JSP files are similar in some ways to server-side includes in static HTML because both embed servlet functionality into the Web page. However, in a server-side include, a call to a servlet is embedded within a special servlet tag; in JSP, Java servlet code (or other Java code) is embedded directly into the HTML page.

One of the many advantages of JSP is that it enables you a programmer to effectively separate the HTML coding from the business logic in your Web pages. You can use JSP to access reusable components, such as servlets, Java beans, enterprise beans, and Java-based Web applications



Einfache Java Server Page

Den folgenden Text in einer Datei mit der `.jsp` extension speichern (z.B. `hallo.jsp`) und mit einem Browser ansehen:

```
<html>
<head>
<title>JSP Example </title>
</head>
<body>
<% out.println ("HALLO JSP"); %>
</body>
</html>
```

Die Zeichenfolgen `<%` und `%>` schließen Java Ausdrücke ein. Diese werden zur Run Time ausgewertet.

Mit dem Ausdruck `<%= new java.util.Date() %>` wird die gültige Zeit bei jedem Reload der HTML Seite in den Browser wiedergegeben.

Komponenten einer JSP Page

JSP Scripting ist ein Mechanismus um Code Fragmente direkt in eine HTML Seite einzubauen. Beispiele:

1. Declaration

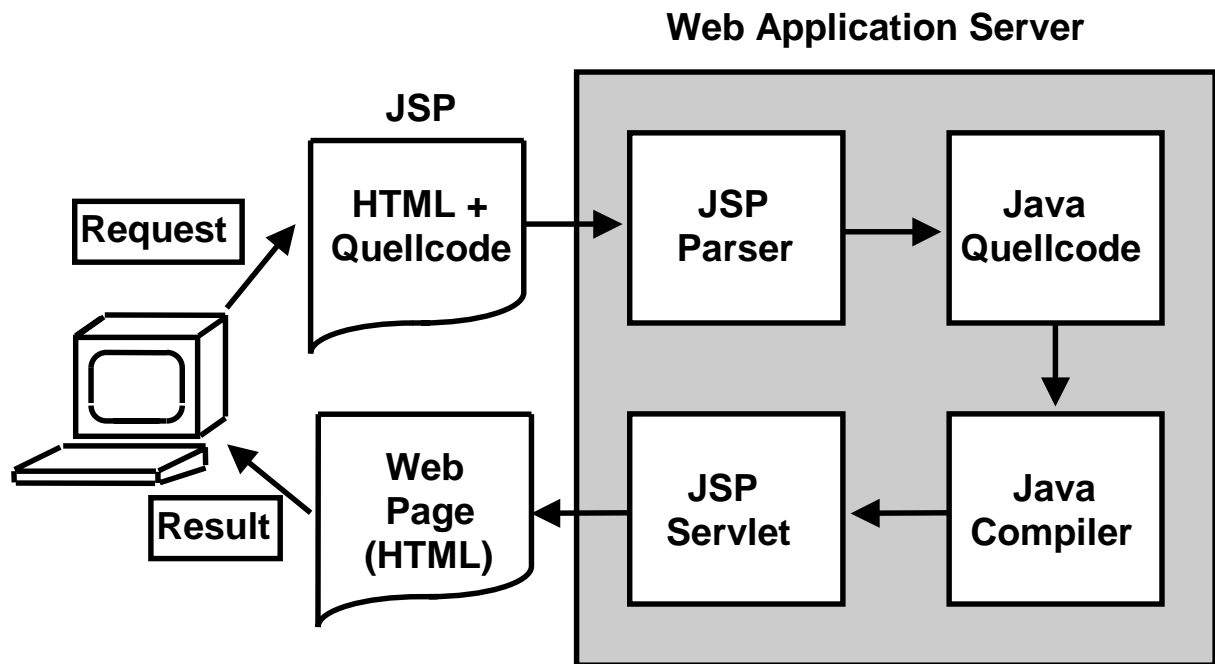
```
<%! String name = new String("Vee"); %>
```

2. Expression

```
<%= customer.getName() %>
```

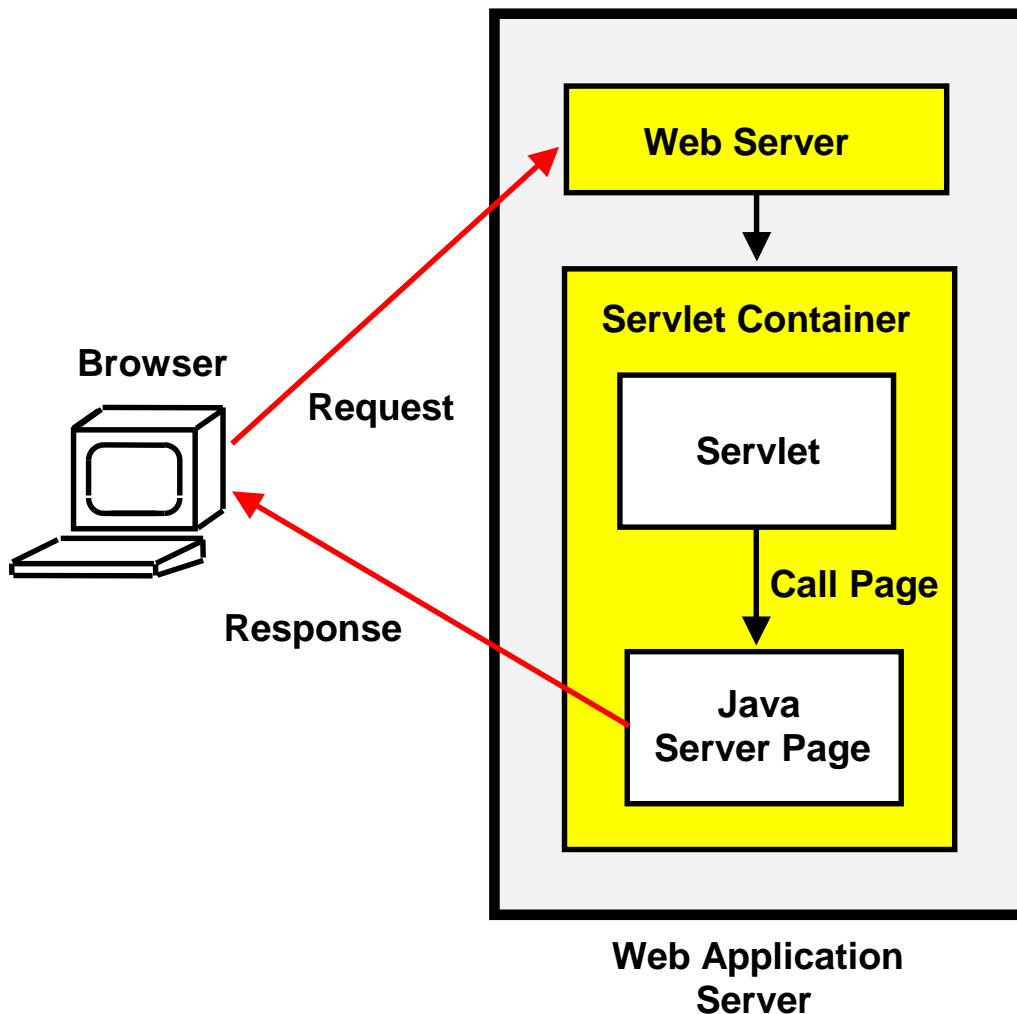
3. Scriptlets – contain any valid java code statements.

```
<% ... %>
```



Java Server Page (JSP)

1. Der Web Browser sendet eine Request an die JSP Seite.
2. Die JSP Engine parses den Inhalt der JSP File. Sie erstellt temporären Servlet Quellcode basierend auf dem Inhalt der JSP.
3. Der Servlet Quellcode wird durch den Java Compiler in eine Servlet Class File übersetzt.
4. Das Servlet wird instantiated. Die init and service Methoden des Servlets werden aufgerufen; die Servlet Logic wird ausgeführt.
5. Die Kombination von statischem HTML, kombiniert mit den dynamischen Elementen spezifiziert in der ursprünglichen JSP Definition, geht an den Web Browser zurück durch den Output Stream des Servlet Response Objektes.



Interaktion Servlet - JSP

Es ist durchaus möglich, innerhalb eines Browsers eine .jsp Seite mittels ihrer URL direkt aufzurufen.

Üblicher ist der Aufruf eines Servlets mittels eines Form Tags. Das Servlet wiederum ruft eine Java Server Page auf, welche die Ausgabedaten erzeugt.

Microsoft Active Server Pages (ASP)

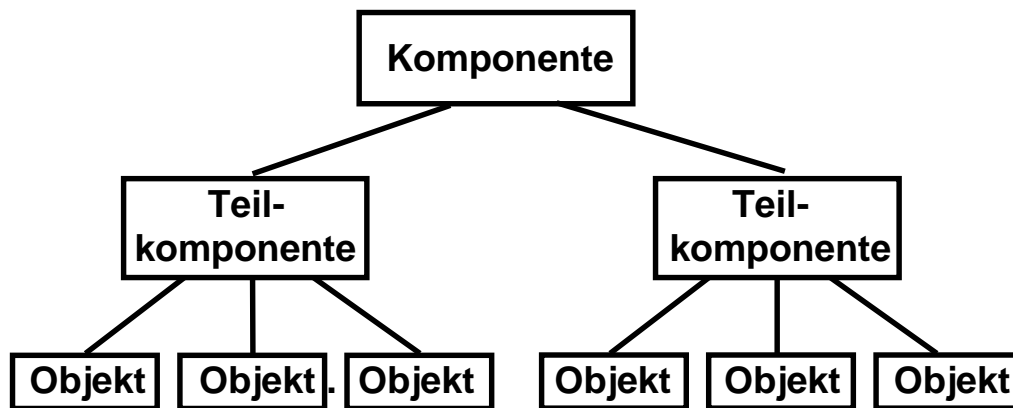
Ähnliche Technologie wie Java Server Pages. Unterschied:

- **Der dynamische Teil wird nicht in Java, sondern in Visual Basic (oder einer anderen Microsoft Sprache) geschrieben**
- **ASPs können nur auf Microsoft Windows Servern und Microsoft Web Servern eingesetzt werden.**

Angenommen:

Servlet mit 1 Million Lines of Code

transaktionssicher (ACID)



Komponenten

Komponenten sind unabhängige, in sich abgeschlossene, wohl definierte Software Einheiten, die eine spezifische Leistung über standardisierte Schnittstellen bieten.

Komponenten lassen sich mit anderen Komponenten zu größeren Einheiten zusammenfügen, die wiederum Komponenten oder eigene Anwendungen sind.

Komponenten setzen sich typischerweise aus Objekten zusammen.

Komponenten lassen sich durch geeignete Parameterisierung in einer Vielzahl von Entwicklungsumgebungen einsetzen, und sind unabhängig von Sprache, Betriebssystem und Hardware.

Eine Komponente hat

- **eine Art "Stecker", mit dem sie sich verbinden kann.**
- **eine Art "Steckdose", welche benutzt wird, um verschiedenen Komponenten die Möglichkeit zu geben, sich "einzustecken".**
- **die Möglichkeit Informationen über sich selbst bekannt zu geben.**
- **eine spezifizierte Menge von Eigenschaften**

Begriffe

Business Object, System Level Object, Metadaten

Ein Business Object ist eine Komponente der Anwendungsschicht, die in nicht voraussehbaren Kombinationen eingesetzt werden kann.

Ein Business Object repräsentiert auf eine erkennbare, nachvollziehbare Art ein Objekt (eine Entity) des täglichen Lebens.

Das Konzept eines Business Objektes steht dem betriebswirtschaftlich motivierten Anwendungskontext sehr viel näher als dem technisch motivierten Konstrukt eines programmiersprachlichen Objektes.

Ein System Level Objekt repräsentiert eine Komponente, mit der der Benutzer nie direkt etwas zu tun hat. Sie wird lediglich von Informationssystemen und Programmen genutzt.

Componentware ist Software, deren Teile sich baukastenartig zusammensetzen lassen.

Metadaten sind sich selbst beschreibende Information, welche die dynamische Struktur eines Systems definieren. Im Falle einer Datenbank beschreiben Metadaten die Datenbank Struktur und werden im „Repository“ verwaltet.

Business Objekt „Kunde“

Komponente	1	Name, Vorname		
	2	Titel, Anrede		
	3	Straße		
	4	PLZ	}	Firmenanschrift 1
	5	Ort		
	6	Straße	}	Firmenanschrift 2
	7	PLZ		
	8	Ort	}	Privatanschrift
	9	Straße		
	10	PLZ	}	
	11	Ort		
	12	Straße	}	
	13	Kunden-Nr. =		
		f(Object Nr., Objekt Identifikation		
	14			
	•			
	•			
	•			
	•			

ca. 500 Attribute:

Fax Nr.
Tel. Nr.
e-mail Adresse
Info über Partner
Info über Kinder
Zeiger auf Kinder
Arbeitgeber
Stellung im Betrieb
Mitglied im Tennisclub Blau Weiss
Rentendaten
•
•
•
•

Java Beans

Objektorientiertes Java Komponenten Modell, JavaBeans sind Java binary parts

häufig für visuelle Komponenten eingesetzt (etwa Buttons und Scrollbalken)

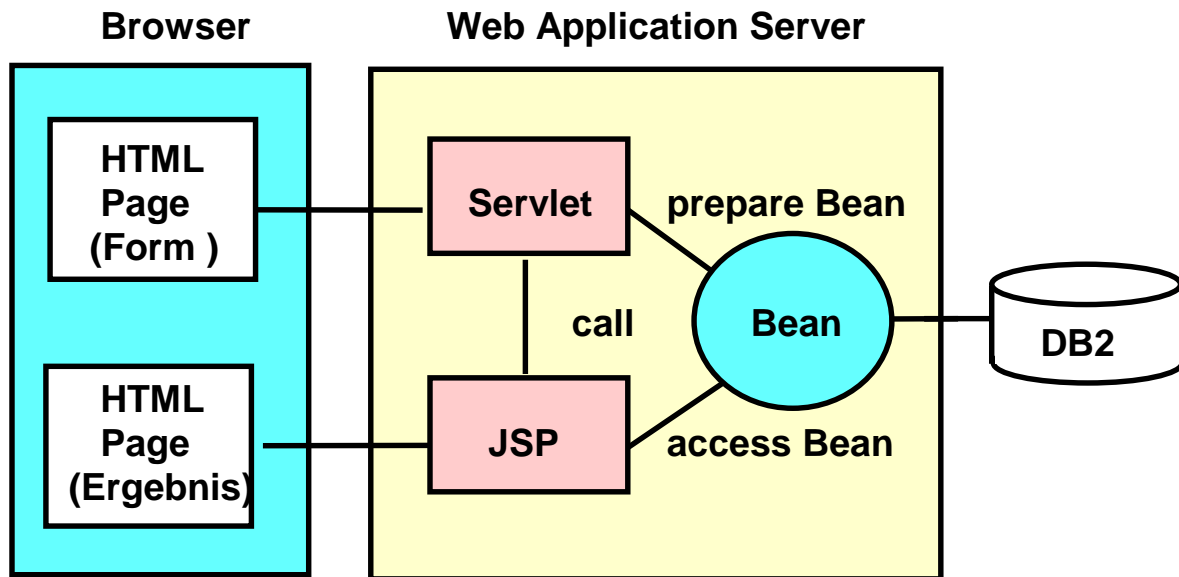
Hauptmerkmale :

- **Properties (Eigenschaften, z.B. get und set)**
- **Methoden und**
- **Events (Ereignisse)**
- **Namens Konventionen**
- **Introspection (BeanInfo Klasse)**

Der JavaBeans Komponenten Modell Teil des JDK Lieferumfangs unterstützt:

- **Sicherheit (benutzt den Java Security Manager)**
- **Versionsmanagement**
- **Life-Cycle Management**
- **Event Notification,**
- **Configuration und Property management**
- **Scripting**
- **Meta-Daten und Introspection**
- **Persistenz (über Serialisierung)**
- **Benutzbarkeit (die „BeanBox“ des JDK ist ein Prototyp einer grafischen Umgebung für das Zusammensetzen von Beans)**
- **Eigeninstallation(über Java Archiv Files)**

Für unternehmensweite Anwendungen (Enterprise Applications) fehlen Schlüsseleigenschaften, z.B Transaktionsdienste, Namensdienste und Sicherheitsdienste. Werden JavaBeans hiermit angereichert, spricht man von Enterprise JavaBeans.



Nutzung von Java Beans

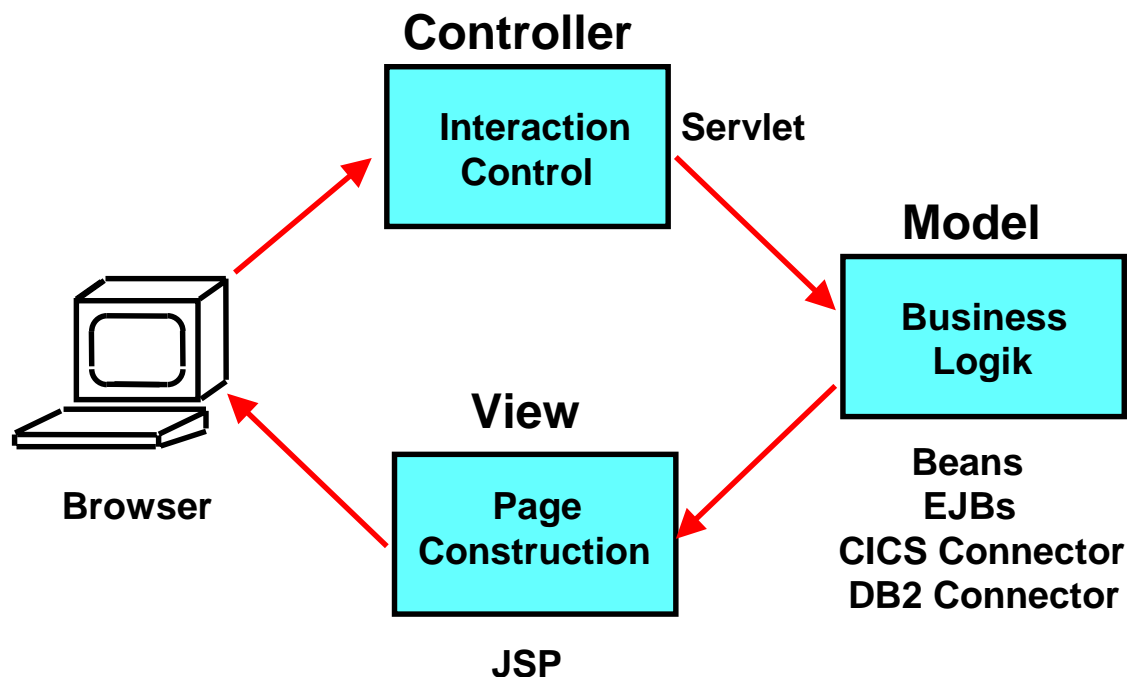
Ein Servlet ist ein Java Programm, das Bildschirm Output in der Form einer HTML Datei produziert.

Eine JAVAServerPage ist eine HTML Seite mit zusätzlichen JSP Tags.

Wird eine JSP Seite aufgerufen, so kompiliert sie ein JSP Übersetzer in ein Servlet.

In der Praxis: Servlets und JSP werden von verschiedenen Leuten erstellt (Model-View-Controller Ansatz). Eine JSP ist zwar eine vollwertige Java Komponente, aber der Java Code Anteil innerhalb der JSP wird in der Regel auf ein Minimum reduziert.

Es existieren (wie für HTML Seiten) spezielle Werkzeuge für das Erstellen von JSP's, die das Hand-coding von HTML Statements automatisieren.



Model/View/Controller Triade (MVC)

Das „Modell“ ist ein Anwendungsobjekt und kapselt die Business Logik. Der „View“ ist die Screen Darstellung dieses Objektes. Der „Controller“ definiert, wie die Benutzerschnittstelle auf Benutzereingaben reagiert.

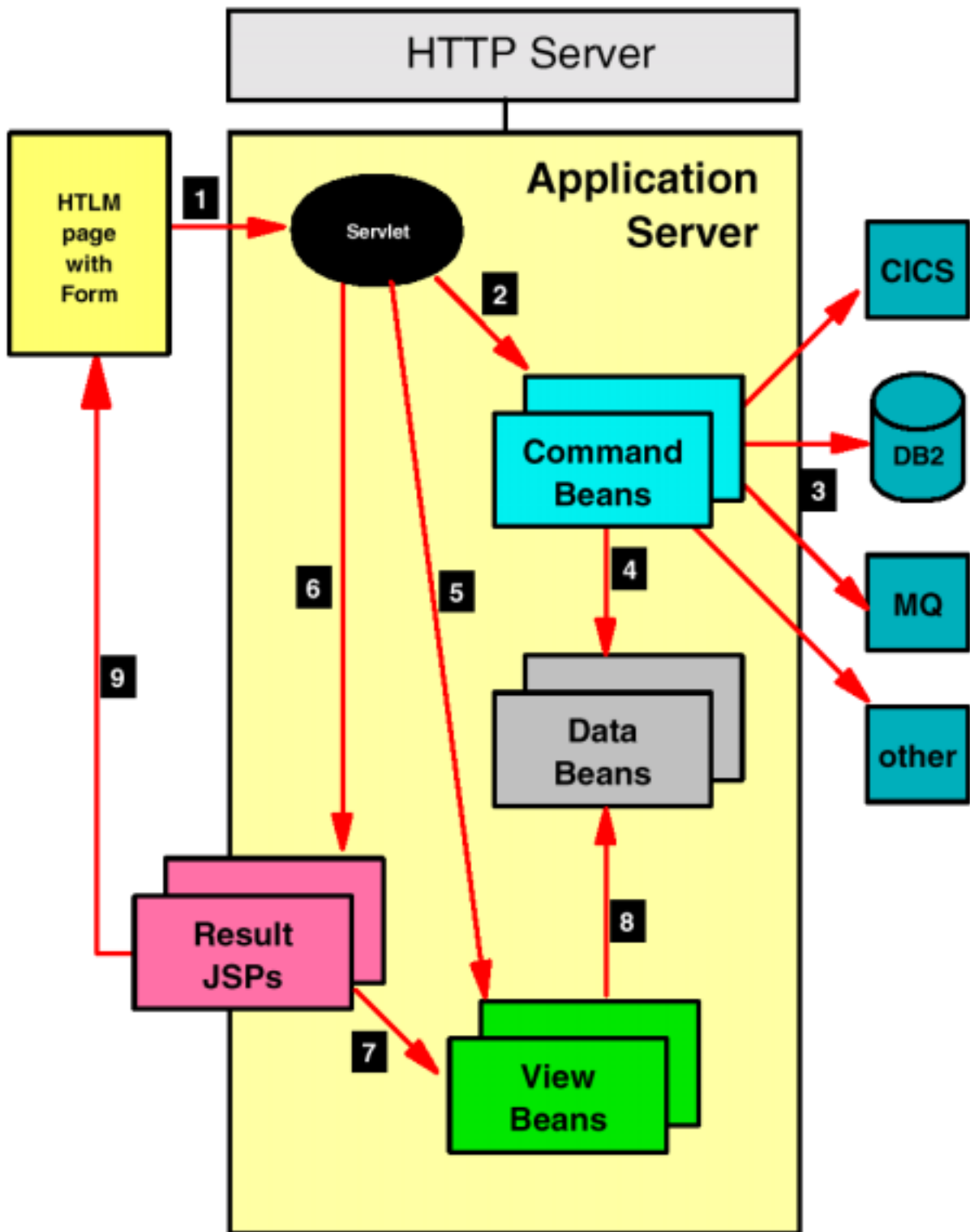
Command- und Data Beans oder Enterprise Java Beans (plus häufig CICS, IMS Programme, oder Stored Procedures) sind das „Modell“ (=Business Logik).

JSP's und View Beans sind der „View“

Das Servlet ist der „Controller“

MVC entkoppelt Modell und View zur Verbesserung von Flexibilität und Re-Use. Der Entwickler der Browser Darstellung arbeitet nur mit der Java Server Page.

Zentrisches Programmier Modell. Die gesamte Anwendungslogik (EJB, Servlet, JSP) läuft auf dem Server. Der Klient (*thin client*) braucht nur einen Browser.

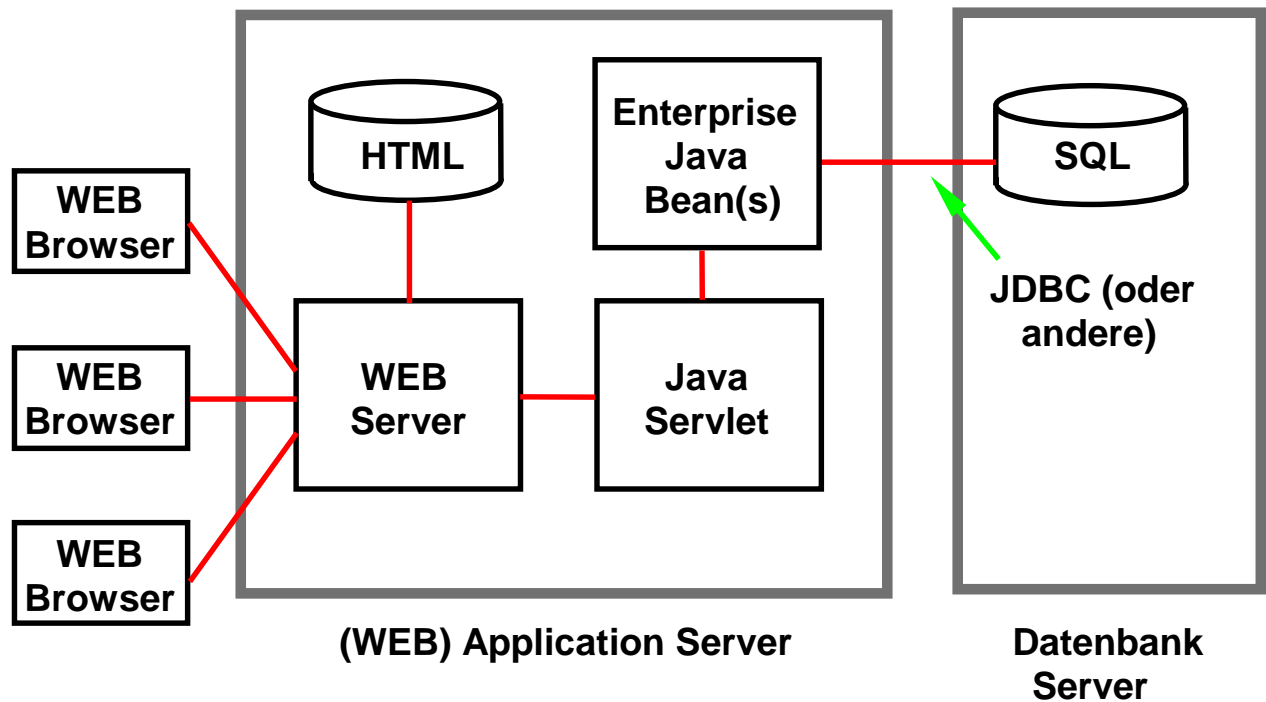


Architektur einer JSP Web Anwendung

Architektur einer JSP Web Anwendung

1. **HTML page:** static or dynamic HTML page, created from a previous step, contains one or multiple forms that invoke a servlet for processing of the next interaction.
2. **Servlet gets control** from the Application Server to perform validation and control of flow; sets up and calls command beans that perform the business logic.
3. **Command beans control** the processing of the business logic; logic may be imbedded in the command bean, or it can be delegated to back-end or enterprise systems, such as relational databases, transactions systems (CICS, MQSeries, IMS, and so forth); command bean may perform one specific function or it may contain many methods, each for a specific task (task wrappers). Command beans invoke database and transaction systems using „connectors“.
4. **Results of command beans** (or back-end systems) processing are stored in data beans. Data beans could contain an SQL result or a CICS communication area (COMMAREA).
5. **View beans provide** the contract between the output producing JSPs and the data beans that contain the dynamic data to be displayed in the output; servlet initializes the view beans and registers them with the request block so that the JSPs can find them.
6. **Servlet calls a JSP** for output processing and formatting depending on the results of the command beans; JSPs generate the output for the browser.
7. **JSP use tags** to declare the view beans and get access to all the dynamic data that must be displayed in the output.
8. **View beans contain** one or multiple data beans and provides tailored methods so that the JSP has access to the data stored in the data beans; data beans may not provide the necessary methods for a JSP to access the data.
9. **JSP assembles** the output and sends it back to the browser as an HTML page with dynamic data; in many cases, that output again contains form(s) to enable the user to continue the dialog with the application.

Servlet is the controller
Command beans provide the model
JSP is the view



Dynamischer WEB Seiten Inhalt (3)

Im einfachsten Fall enthält das Java Servlet die Anwendungslogik. In komplexeren Fällen lohnt es sich, die Anwendung in Komponentenform zu implementieren. Java Beans implementieren das Java Komponentenmodell.

Enterprise Java Beans sind Java Beans mit zusätzlicher Funktionalität, besonders Transaktionseigenschaften (ACID), Persistenz und Sicherheit.

Enterprise Java Beans (EJB)

Java basiertes Server Komponentenmodell, März 1998. Teil der J2EE-Architektur. Version 3.0 verfügbar seit Februar 2005.

- EJB Komponenten sind serverseitige Komponenten, die ausschließlich in Java geschrieben sind. EJB Komponenten enthalten nur Business Logik, keine Präsentationslogik und keine Systemfunktionen.
- EJBs sind in einen „Container“ eingebettet (Laufzeitumgebung). Die EJB Architektur ist inhärent
 - transaktionsorientiert,
 - distributed,
 - portierbar,
 - multi-tier,
 - scalierbar und
 - sicher.
- Diese und weitere Systemfunktionen wie
 - Life-cycle management,
 - threading und
 - Persistenzwerden automatisch für die EJB Komponente von dem EJB Container zur Verfügung gestellt.
- EJB Komponenten werden deklarativ (über einen *Deployment Descriptor*) zur Laufzeit angepaßt. Die Anpassung bezieht sich auf Transaktionsverhalten, Sicherheitseigenschaften, life-cycle und state management und Persistenz.
- Interoperabilität von EJBs und CORBA konformen Objekten ist möglich.

Der J2EE (Java 2 Platform, Enterprise Edition) Standard kombiniert Technologien wie Servlet, JSP, EJB, JMS, JCA und den JDK.

Persistenz

Die permanente Speicherung eines Objektes auf einem Plattenspeicher wird als Persistenz bezeichnet.

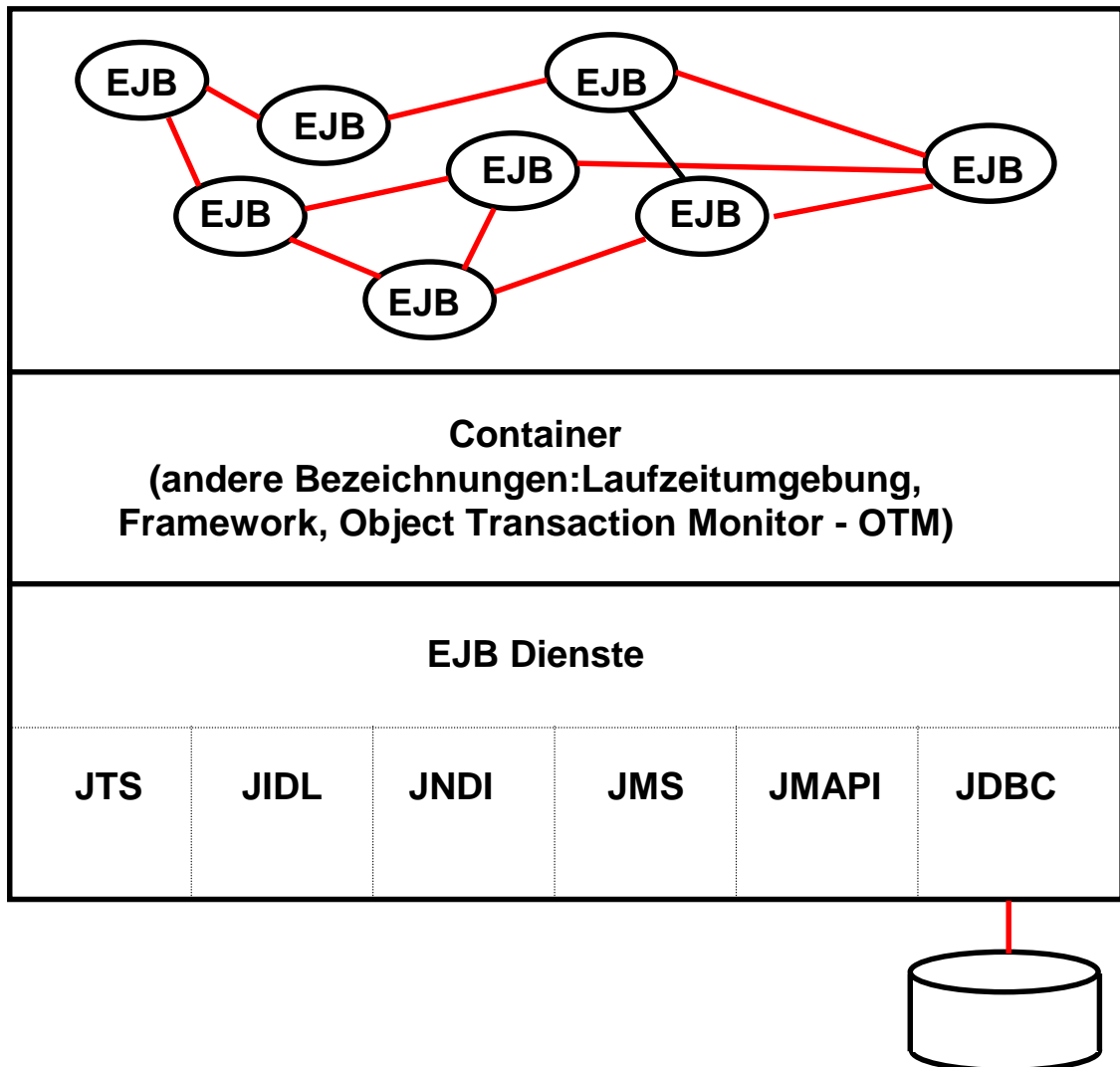
Konzeptuell können Objekte in einer Objektdatenbank (z.B. POET oder Jasmine) gespeichert werden. In der Praxis werden SQL (oder IMS, ADABAS oder VSAM) Daten als Objekte gekapselt; der Zugriff erfolgt z.B. über eine JDBC (Java Data Base Connectivity), SQLJ oder DB2Connect Schnittstelle.

Persistente Objekte existieren permanent außerhalb des Gültigkeitsbereichs des Programms, das sie erzeugt hat.

Persistenz wird implementiert, indem der Status (die Attribute) eines Objekts zwischen den einzelnen Programmausführungen gespeichert wird. Wenn das Objekt erneut benötigt wird, wird es aus seiner gespeicherten Form wieder hergestellt. Der Herstellungsprozeß erzeugt ein neues Objekt, das mit dem ursprünglichen identisch ist.

Bei der Persistenz werden den gespeicherten Daten alle Objektattribute (etwa Klassenname, Feldname und Zugriffs-Modifizier) zugeordnet, so dass verhindert wird, daß die Daten versehentlich mit einem falschen Objekttyp abgelegt werden.

Enterprise Java Beans (EJB)



Enterprise Java Beans sind Java Beans mit erweiterter Funktionalität. Dies sind unter anderem

- **JTS** Java Transaction Service
- **JNDI** Java Naming directory Interface
- **JMS** Java Messaging Services
- **JDBC** Java Data Base Connectivity
- **JMAPI** Java Management API
- **JIDL** Java interface definition language

Enterprise Java Beans (EJB)

Enterprise Java Beans sind Java Beans mit erweiterter Funktionalität. Dies sind unter anderem

- **JTS** **Java Transaction Service**
- **JNDI** **Java Naming Directory Interface**
- **JMS** **Java Messaging Services**
- **JDBC** **Java Data Base Connectivity**
- **JMAPI** **Java Management API**
- **JIDL** **Java Interface Definition Language**

JTS ist der Java Transaction Service. Der Transaction Manager ist eine API für den aufruf Des Transaction Service.

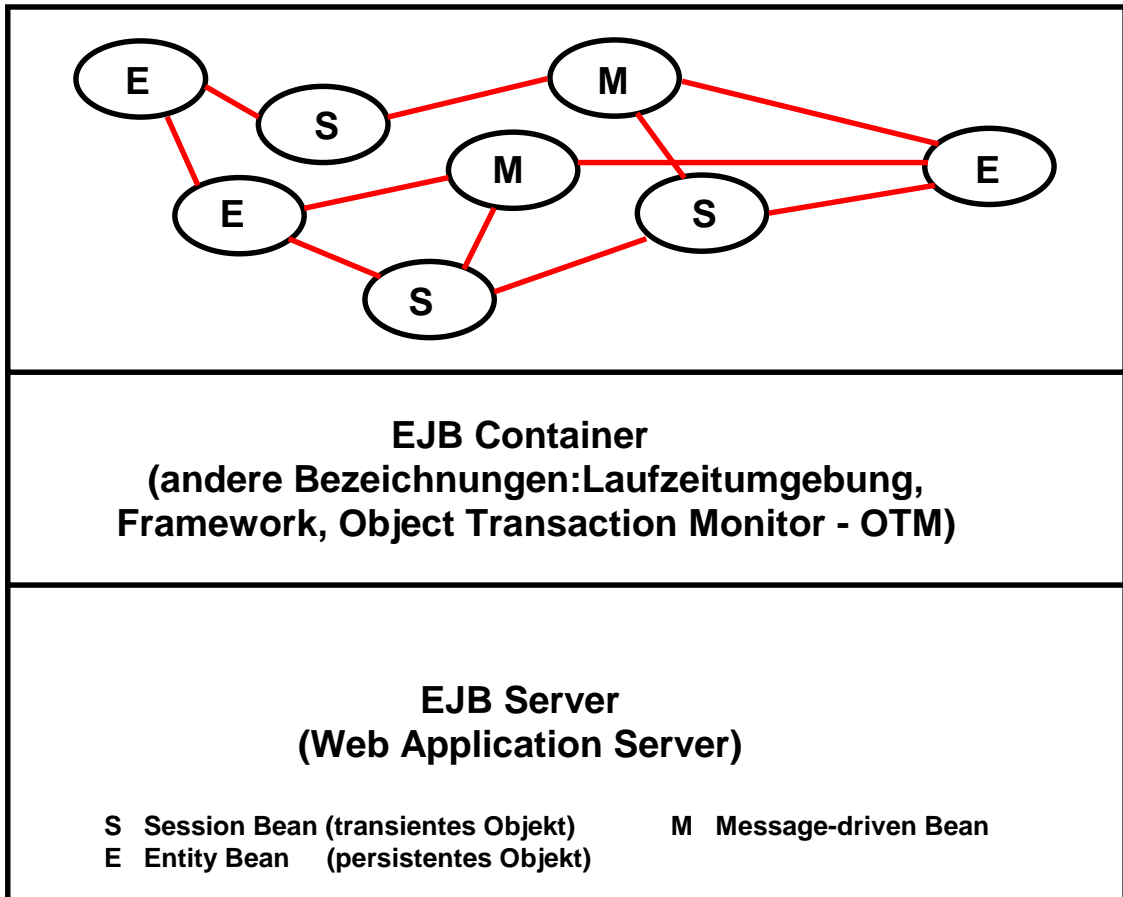
JNDI, die Java Naming and Directory Interface, ist eine API für den Zugriff auf Naming und Directory Services.

JMS, der Java Message Service, ist eine API für asynchrone Message Delivery Services.

JDBC , die Java Database Connectivity API, ermöglicht den Zugriff auf existierende Datenbanken über eine gemeinsame Schnittstelle.

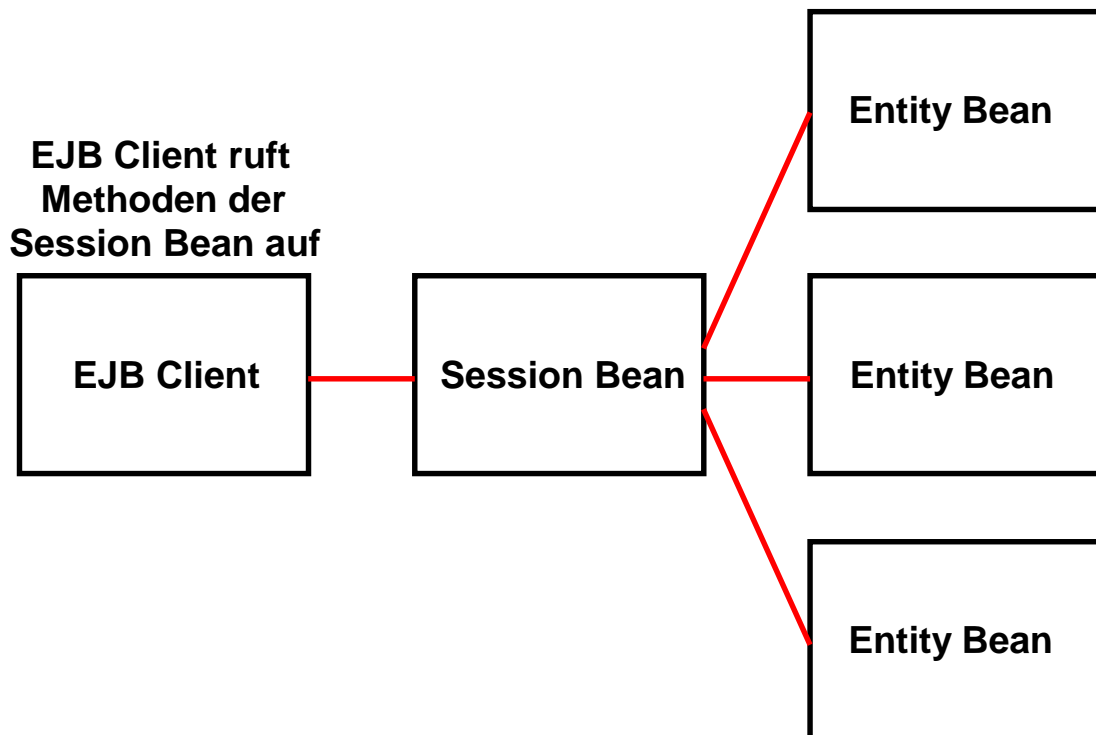
JMAPI bezeichnet die Java Management API. Sie definiert den Zugriff auf eine Reihe von Services für das Management von Java Ressourcen.

JIDL bezeichnet die Java Interface Definition Language. Dies ist eine Interface für distributed computing CORBA Services.



Drei verschiedene Arten von EJBs

- **Entity Beans** sind Komponenten, die eine objektorientierte Sicht auf persistent gespeicherte Entitäten, also eindeutig identifizierbare und über Attribute beschreibbare Informationseinheiten, repräsentieren. Diese Entitäten sind beispielsweise in einer Datenbank gespeichert. Handelt es sich dabei um eine relationale Datenbank, so korrespondieren Entity Beans jeweils einer Zeile der entsprechenden Datenbanktabelle. Entity Beans modellieren Geschäftskonzepte, beispielsweise ein Konto.
- **Session Beans** sind Komponenten, die auf dem Server ablaufende Geschäftslogik implementieren. Sie modellieren Geschäftsprozesse, wie z.B. eine Überweisung.
- **Message-Driven Beans** entsprechen Session Beans, nur werden sie durch den Empfang einer Nachricht ausgelöst.



Session Fassade EJB Architektur Modell

Session Beans:

- führen Geschäftsprozesse (Business Logik) aus
- manipulieren persistente Objekte (Daten), die als Entity Beans modelliert sind

Probleme mit dem Session Fassade EJB Architektur Modell:

- Entity Beans sind verteilte Objekte, die von beliebigen Klienten aufgerufen werden können
- Entity Beans sind transaktionsfähig

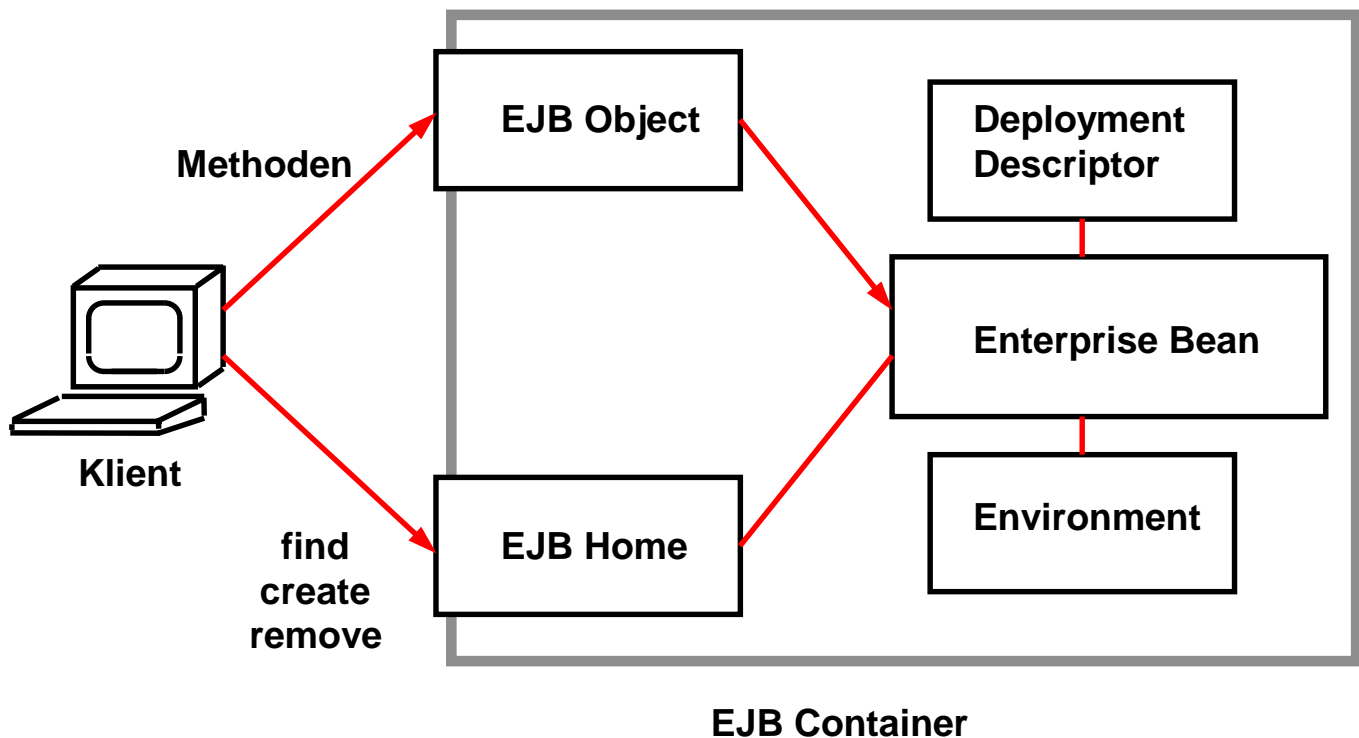
Beides ist bei einer Session Fassade unnötiger Overhead.

Lösungsansatz : „Java Data Objects“ (JDO) an Stelle der Entity Beans. JDOs sind reguläre Java Klassen, die im Vergleich zu Entity Beans über einen reduzierten Funktionsumfang verfügen.

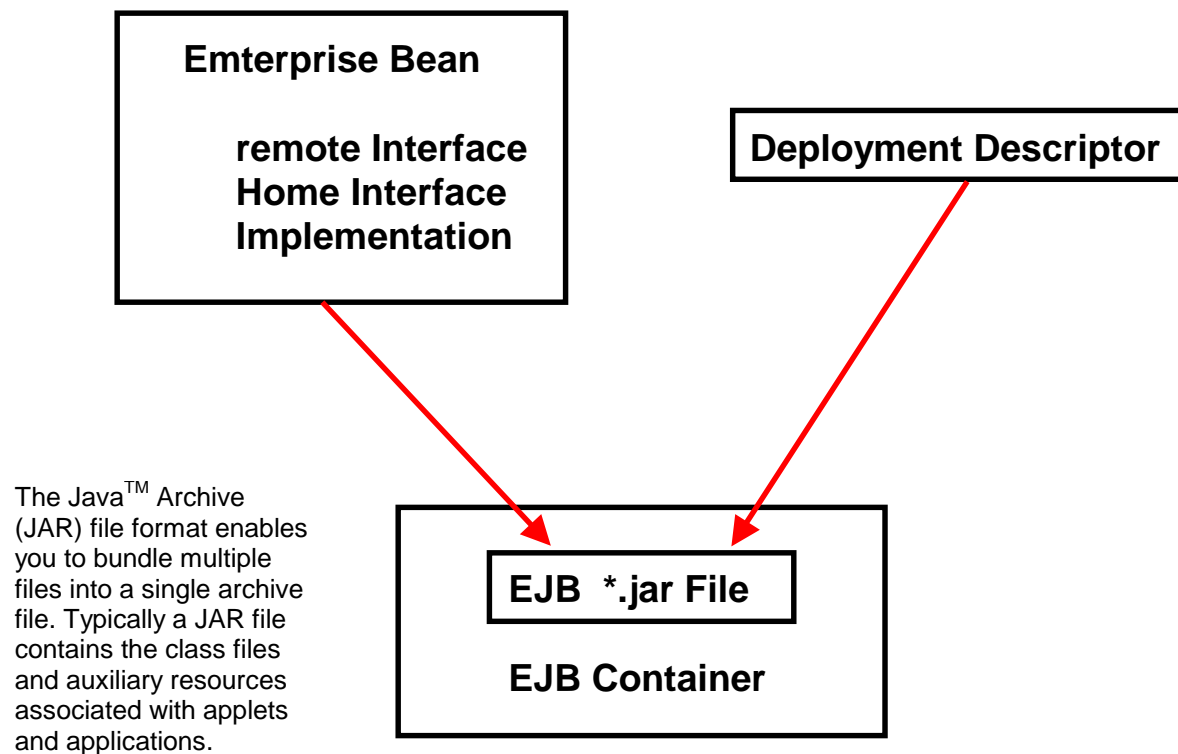
Weitere Alternative: „Konnektoren“ sind Java Klassen der Java Connection Architecture (JCA), die einen Zugriff auf CICS, Oracle, DB2 ermöglichen.

EJB Schnittstellen

Die „EJB Object“ Schnittstelle empfängt alle Methodenaufrufe. Sie implementiert die Transaktions-Zustandsverwaltungs, Persistenz- und Sicherheitsdienste für die Bean je nach den Angaben des Deployment Descriptors



„EJB Home“ dient der Identifizierung des Beans. Auf die EJB Home Schnittstelle kann mit JNDI zugegriffen werden. Sie implementiert alle Life-Cycle Dienste für die Bean



Deployment Descriptor

legt die Laufzeit Parameter einer EJB fest

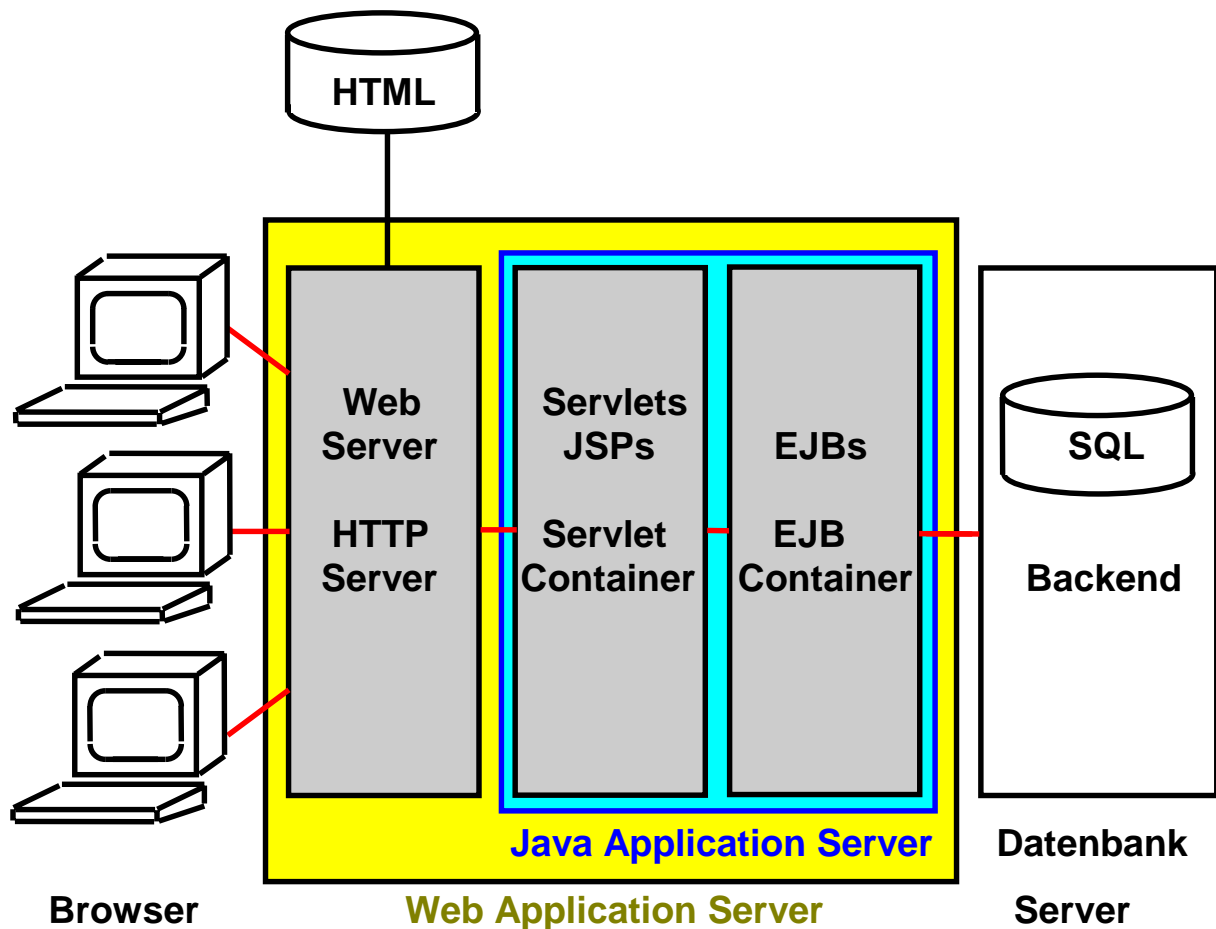
Parameter können statisch zur Assembly Zeit oder dynamisch zur Laufzeit festgelegt werden

Beispiele für Parameter:

- **Datenbank Name**
- **Verbindung zu Legacy Anwendungen**
- **JNDI Namensraum des Containers**
- **Transaktions-Semantik**
- **Umgebungseigenschaften**

typischerweise durch den EJB Entwickler angelegt

kann durch einen Administrator mit Hilfe eines Tools abgeändert werden



Application Server Hierarchie

Der Web Application Server ist ein Prozess, der normalerweise in seinem eigenen virtuellen Adressenraum läuft. Er besteht aus mehreren Komponenten:

1. Der Web Server ist vielfach Apache.
2. Der Java Application Server unterhält u.a. eine Java Virtuelle Maschine
3. Der Servlet Container (Servlet Engine) ist eine Java Laufzeit Umgebung (runtime component) für die Ausführung von Servlets und Java Server Pages.
4. Der EJB Container ist eine Laufzeit Umgebung für die Ausführung von deployed Enterprise Java Beans.

Transaktionsverarbeitung mit Enterprise JavaBeans

Enterprise JavaBeans vereinfachen die Verwaltung von Transaktionen, da der EJB-Container die meisten Aufgaben bei der Ausführung von Transaktionen übernimmt. Die Transaktions-attribute werden im Deployment Descriptor deklariert.

Sie werden daher auch Container-gesteuerte Transaktionen genannt. Transaktionssteuerung durch den EJB-Container wird auch als deklarative (implizite) Transaktionssteuerung bezeichnet, da Transaktionsattribute im Deployment Descriptor deklariert werden.

Der Ablauf von Transaktionen kann aber auch explizit in der Bean-Klasse gesteuert werden. Man spricht dann von Bean-gesteuerten Transaktionen.

Formal unterscheidet man bei dem J2EE Standard zwischen einem *Transaktionsmanager* (TM) und einer *Transaction Service Implementation* (TS). Der TM ist die Schnittstelle zum TS. Der TS implementiert einen Transaktionsmonitor vergleichbar zu CICS oder Tuxedo und ist die mit Abstand umfangreichste Komponente eines Web Application Servers.

Enterprise JavaBeans unterstützen neben lokalen Transaktionen, bei denen nur auf eine Datenbank zugegriffen wird, auch Transaktionen in verteilten Umgebungen mit mehreren Datenbanken.

Bestandteile eines EJB Transaktionsverarbeitungssystems

- **Transaktionsmanager:**

Die Schnittstelle zum Transaktionsmanager ist die EJB Java Transaktion API (JTA)

- **EJB-Container:**

Der EJB-Container übernimmt in Zusammenarbeit mit dem Transaktionsmanager Verwaltungsaufgaben (z.B. Aufzeichnung des Transaktionskontexts). Container-gesteuerte Transaktionen werden vom EJB-Container gestartet.

- **Ressourcenmanager:**

Ressourcenmanager ermöglichen den Zugriff auf transaktionsgesicherte Daten (z.B. den Zugriff auf relationale Datenbanken über JDBC-Verbindungen). Sie registrieren sich beim Transaktionsmanager, sobald sie in eine Transaktion einbezogen werden. Ressourcenmanager können Resource Adapter der Java Connection Architecture (JCA) beinhalten.

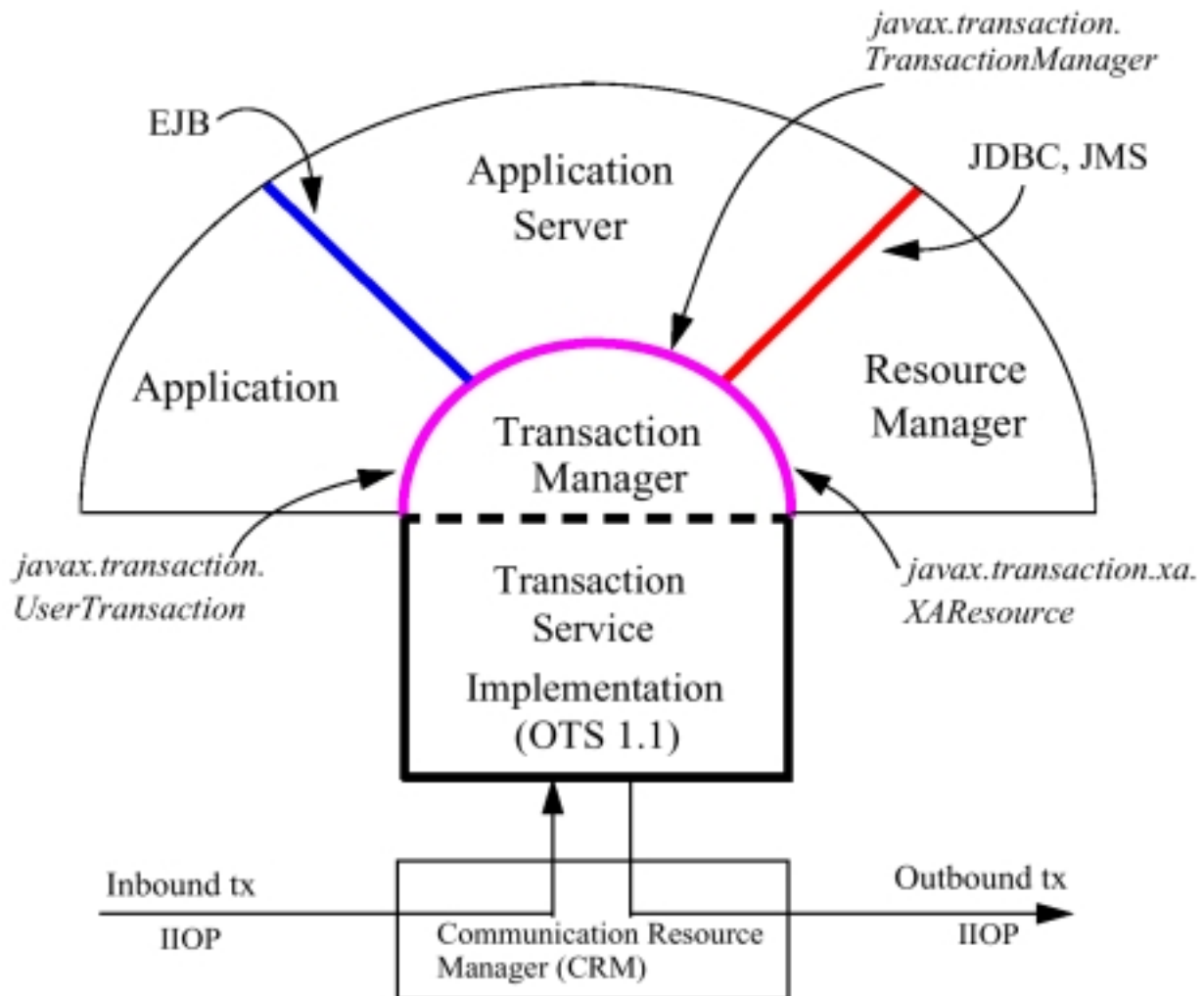
- **Transaktionale Objekte:**

Transaktionale Objekte einer Anwendung nehmen an Transaktionen teil und greifen auf die von Ressourcenmanagern verwalteten Daten zu.

- **Klienten:**

Klienten können Transaktionen starten. Java-Anwendungen, Applets, Servlets und Enterprise Beans sind Beispiele für Klienten von EJB-Anwendungen. Bei Container-gesteuerten Transaktionen haben Klienten keinen Einfluss auf den Ablauf von Transaktionen.

Java Transaction Service (JTS) Java Transaction API (JTA)



Der Java Transaction Service (JTS) ist eine Java Abbildung (mapping) des OTS. JTS Kommunikation zwischen zwei Web Application erfordert daher IIOp (Beispiel 2 Phase Commit) .

Der Transaction Manager implementiert die JTA. Die Schnittstelle zwischen Transaction Manager und Transaction Service ist Hersteller abhängig. In der Praxis wird der Transaction Service häufig durch einen existierenden Transaktionsmonitor implementiert:

- BEA Web Logic benutzt Tuxedo
- IBM WebSphere benutzt CICS und Encina

<https://jsecom15c.sun.com/ECom/EComActionServlet/LegalPage:-:com.sun.sunit.sdlc.content.LegalWebPageInfo;jsessionid=jsecom15c.sun.com-23665%3A40c1ecfb%3A7c3aab86ae6384cd>

Skalierbarkeit

Leistungsverhalten eines Windows NT Web Servers

- 600 statische Zugriffe pro Sekunde
- 100 Java Zugriffe pro Sekunde

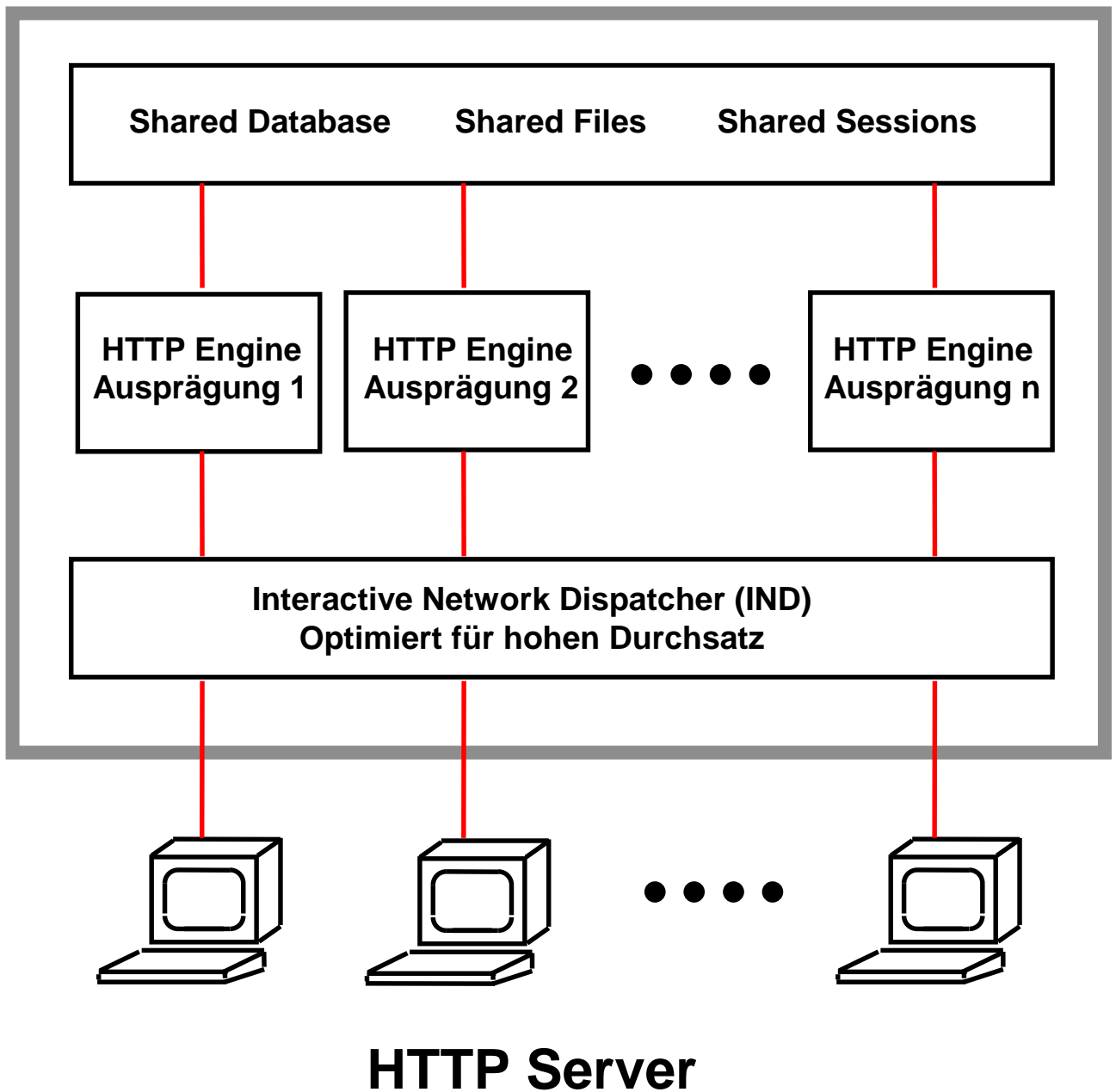
Anforderungen eines größeren Unternehmens in 1999

- Spitzenbelastung mehrere tausend dynamische Zugriffe pro Sekunde

Verhältnis 5 : 1 Spitzen- zu Durchschnittsbelastung.
Verhältnis 10 : 1 nicht selten.

Beispiel: Fernsehwerbung für ein e-Commerce Unternehmen kann Belastung dramatisch anwachsen lassen.

Faktor 10 Wachstum erwartet in wenigen Jahren



Der HTTP Server behandelt Anforderungen für (meistens) statische Ressourcen: HTML Seiten, GIF Dateien und CGI Aufrufe

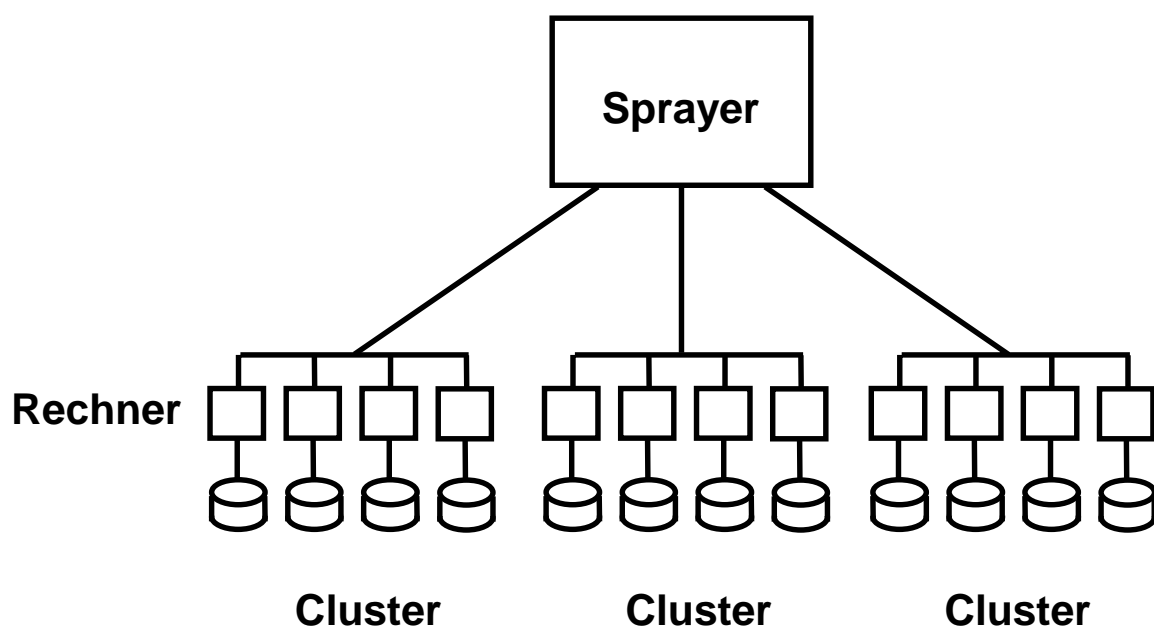
Hohes Verkehrsaufkommen, kurzlebige Anforderungen

Skalierung durch mehrfache Web Server Engines

Der Interactive Network Dispatcher (auch als „Sprayer“ oder Load Balancer bezeichnet) verteilt die Anforderungen auf die einzelnen Web Engines

www.google.com

Google unterhält in 5 Rechenzentren ca. 10 000 Rechner. Mehrere Cluster in jedem Rechenzentrum. Jeder Cluster dupliziert den ganzen Google Datenbestand.

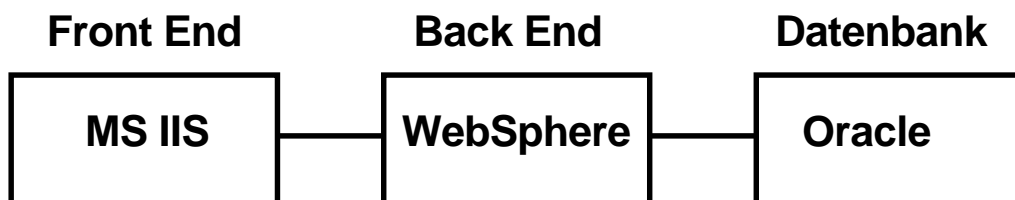


je 30 - 50 Rechner pro Cluster
je 3 - 5 Tbyte pro Cluster
Kopie aller Daten auf jedem Cluster

Sprayer verteilt Anfragen auf die einzelnen Cluster. Jeder Cluster ist in der Lage, jede Art von Anfrage zu bearbeiten.

Einfacher Workload Algorithmus.

www.ebay.com



44 Mill. Artikel

889 Mill. Aufrufe/Tag , etwa 50 000/s

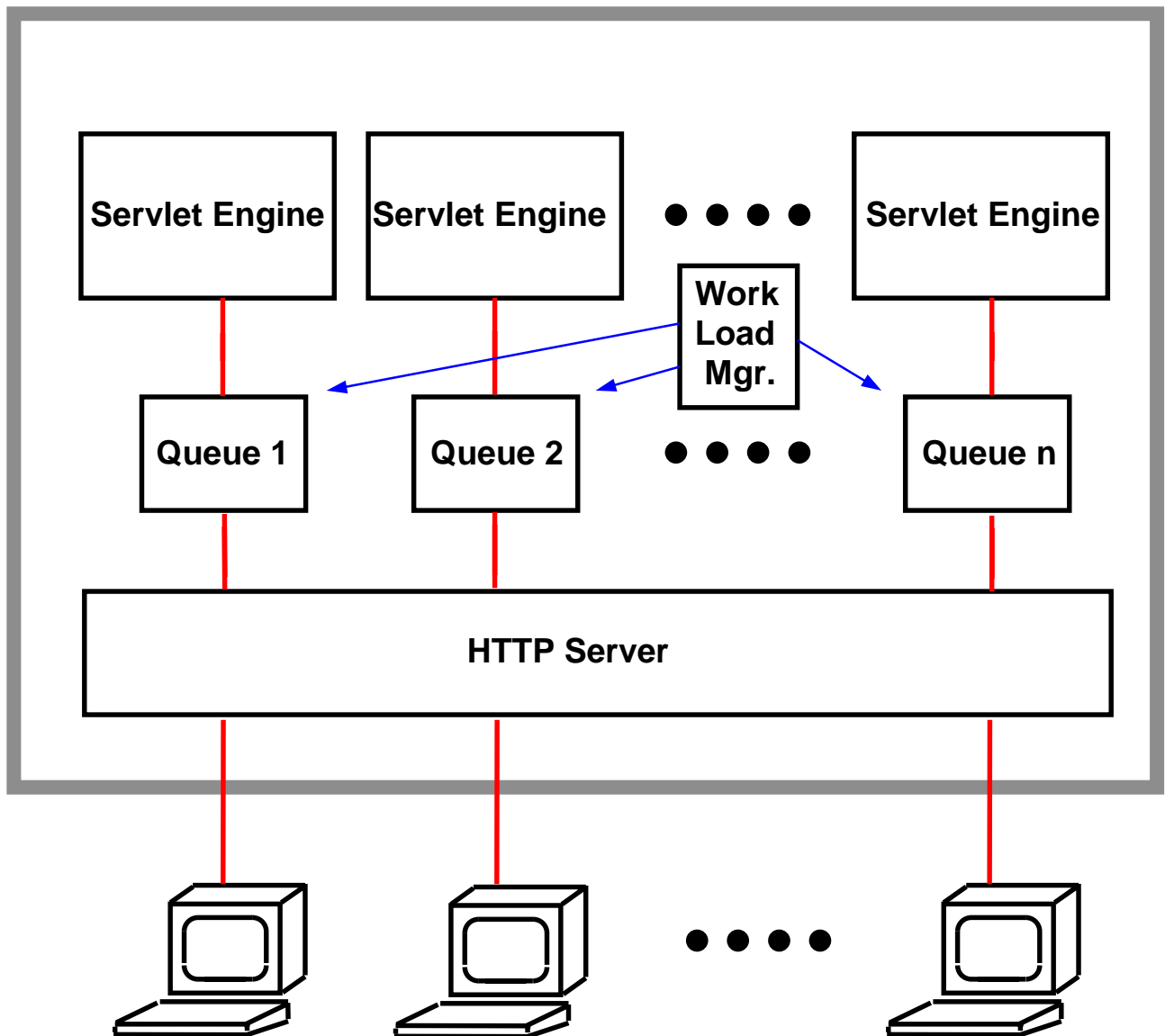
270 Mill. Suchanfragen/Tag etwa 13 000/s

15 Mill. Gebote/Tag etwa 300/s

4 Data Center in 4 Lokationen, 50 SUN Server / Lokationen

Abrechnung auf zSeries CICS System

Mai 2005



Anwendungs-Queues und mehrfache Prozesse

Zur Verbesserung des Leistungsverhaltens laufen mehrere Servlet Prozesse auf dem Applikations-Server. Anforderungen von dem Web Server gehen (je nach Policy) zu einer von mehreren Queues. Jede Queue wird von mehreren Java Prozessen bedient.

Die Queue Policy bestimmt

- URLs, die von der Queue bedient werden
- Anzahl der Prozesse für diese Queue
- Sicherheitsumgebung

Der Administrator legt die Anzahl und die Policies jeder Queue fest

Aufgaben der Servlet Queues

**Availability und Reliability 24 Stunden/Tag, 7Tage/Woche.
Verabschiedet sich ein Prozess, läuft der Rest weiter**

Lastverteilung

Schutz der Anwendungen gegeneinander

Austesten neuer Anwendungen

WebSphere Funktionen unter z/OS und OS/390

- **Parallel Sysplex**
- **Workload Manager**
- **RACF**
- **Crypto**
- **Common Connector Framework**
- **Virtuelle Server**