

Client/Server-Systeme

Prof. Dr.-Ing. Wilhelm Spruth

SS 2004

Teil 13

Web Application Server

Web Application Server

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

BEA Web Logic

IBM WebSphere



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

Auch im Rennen:

SAP

Oracle

Public Domain

JBOSS, Tomcat

Lehrbücher

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

www.redbooks.ibm.com

(sehr große Anzahl von Lehrbüchern, welche vom Web heruntergeladen werden können, unter dem Stichwörtern Java, VisualAge oder WebSphere suchen)

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>

cs 1505 ww6

wgs 08-00

Java Ausprägungen

Java

Java Script

Java Applet

Java Servlet

Java Server Pages

Java Bean

Enterprise Java Bean

cs 1413 ww6

wgs 03-00

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.

cs 1432 ww6

wgs 03-00

Java Applet

Unterschiede zu Java Script

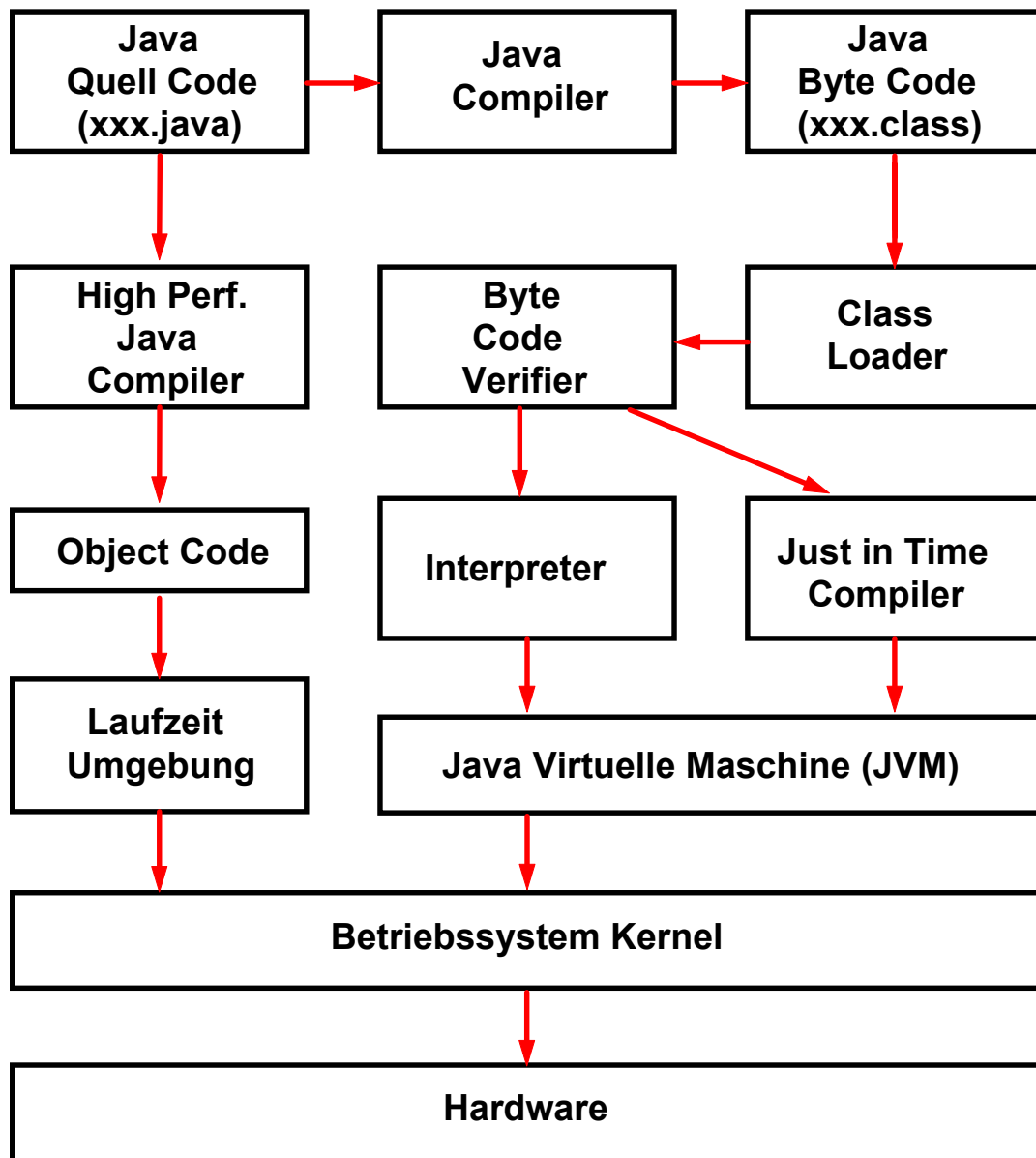
- Volle Java Funktionalität
- Läuft innerhalb der Java Virtuellen Maschine

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

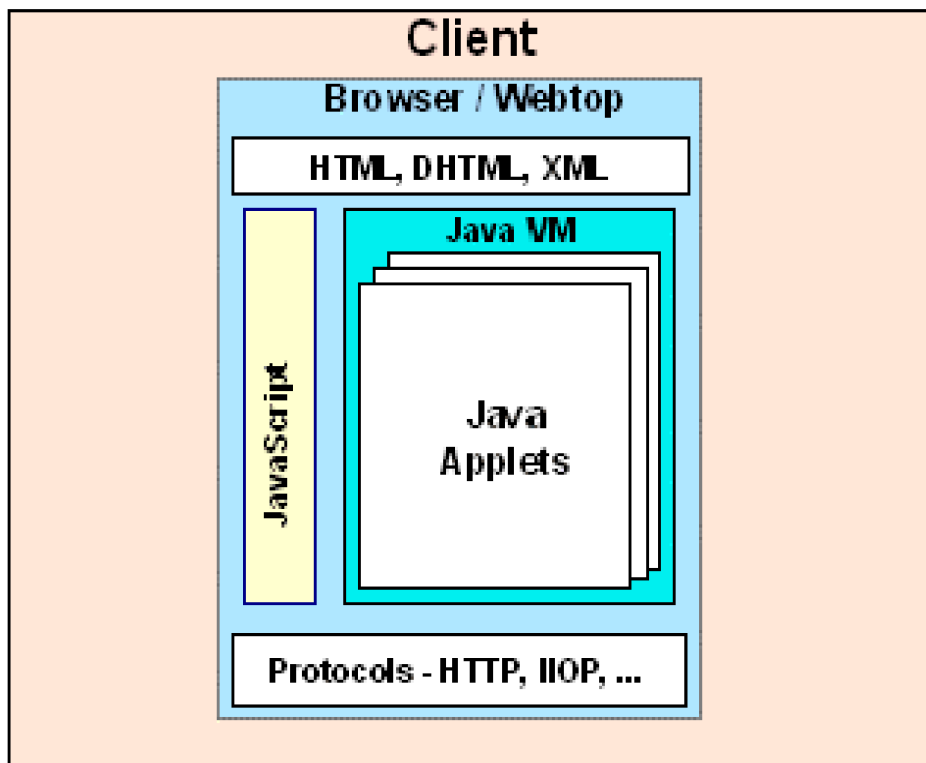
cs 1419 ww6

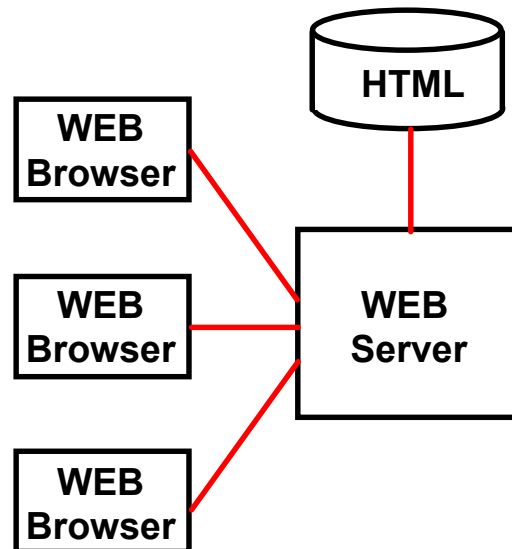
wgs 06-00



Java Virtuelle Maschine

Client Technologies





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

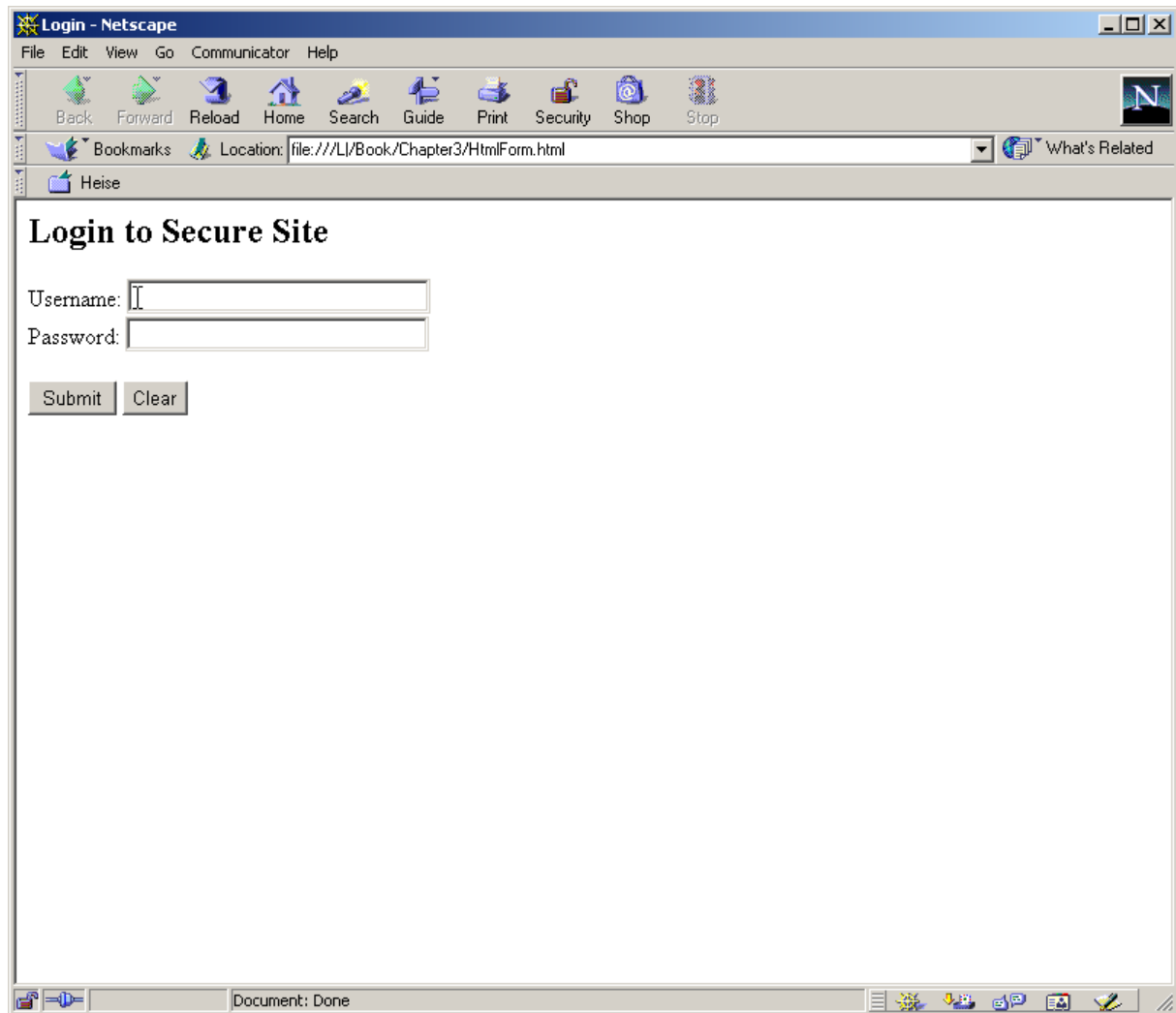
www.qrx.de

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.**



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

HTML Forms

HTML Forms sind ein einfache Werkzeuge, mit dem ein Benutzer Daten mit Hilfe des HTTP-Protokolls an einen Server schicken kann.

Ein HTML-Form besteht aus einem Code-Block, der mit dem `<FORM>` Tag anfängt und dem `</FORM>` Tag aufhört. Eine HTML-Seite kann mehrere Forms enthalten.

```
<HTML>
<HEAD><TITLE>Mailing LIST</TITLE></HEAD>
<BODY>
<FORM METHOD="POST" ACTION="/servlet/xyz.servlet"
ENCTYPE="application/x-www-form-encoded">
<P>Name: <INPUT TYPE="TEXT" NAME="name"
SIZE="25"></P>
<P>Email Address: <INPUT TYPE="TEXT" NAME="email"
SIZE="25"></P> <P><INPUT TYPE="SUBMIT"
VALUE="Submit">
</FORM>
</BODY>
```

Der FORM Tag spezifiziert:

- Die zu benutzende HTTP-Methode. Hier ist dies POST; die Daten werden innerhalb des Bodys der Nachricht übertragen.
- Die Action. Dies ist meistens die URL, es kann aber auch die Action mit ihrem Namen angegeben werden.
- Der Typ der MIME-Enkodierung der Daten in der FORM. Der Default ist "application/x-www-form-encoded".

Login to Secure Site

Username:

Password:

HTML Form

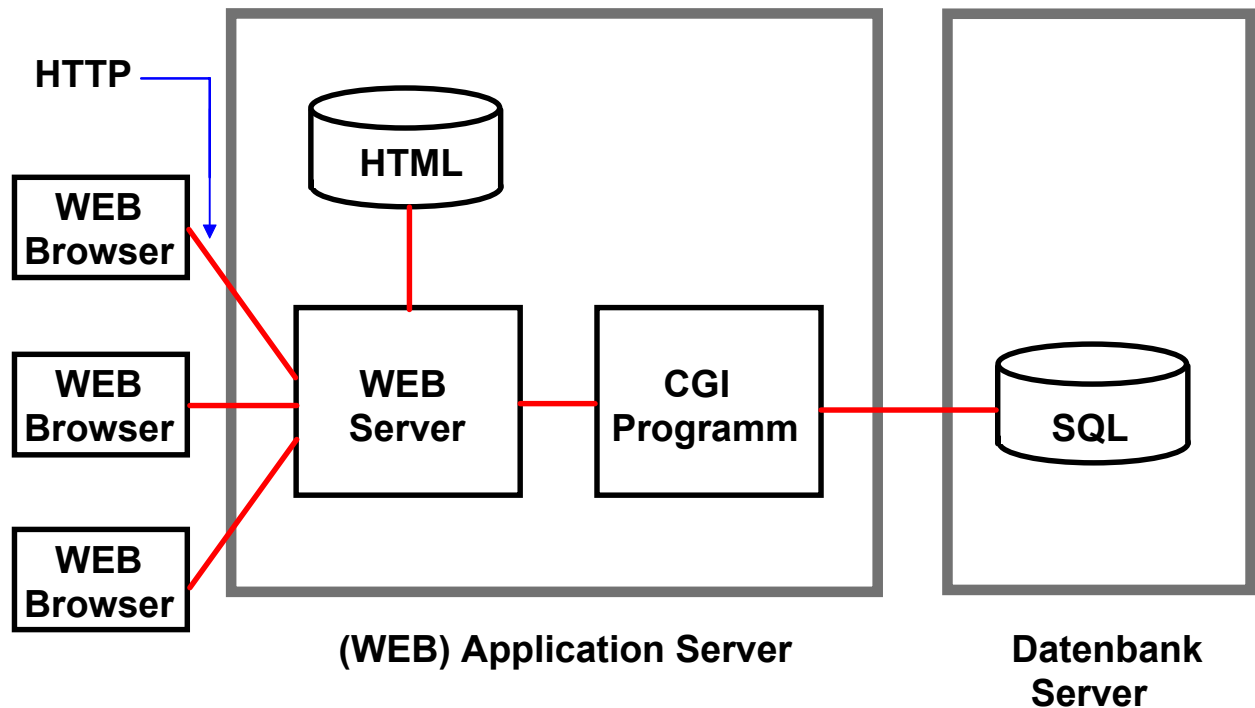
```
<HTML>
<HEAD><TITLE> Login </TITLE> </HEAD>
<BODY>
<H2>Login to Secure Site</H2>

< 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>
```



Dynamischer WEB Seiten Inhalt (1)

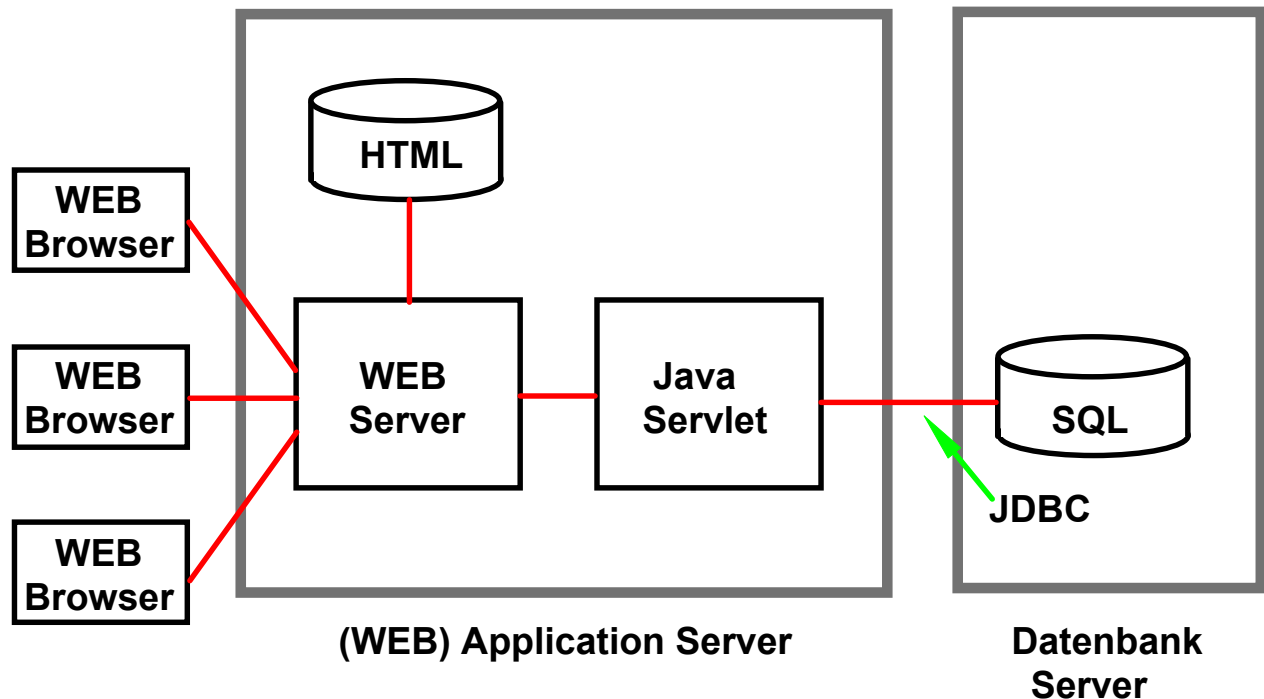
Der Web Browser kommuniziert mit dem Web Server über das HyperText Transfer Protokoll (HTTP). HTTP ist das ursprüngliche Transport Protocol für das World Wide Web.

Zwei Alternativen:

- Web Server sendet statische Seite aus der HTML Datenbank an den Web Browser.
- Web Server ruft über die CGI Schnittstelle Anwendungsprogramm auf. Dieses kann z.B. Daten aus einer OS/390 DB2 Datenbank verwenden, um eine dynamische HTML Seite zu erstellen.

CGI Programme werden häufig in einer Script Sprache, z.B. PERL (Practical Extraction and Reporting Language) oder Unix Shell Script erstellt, können aber auch in einer beliebigen anderen Sprache, z.B. C++ oder Java geschrieben werden.

•

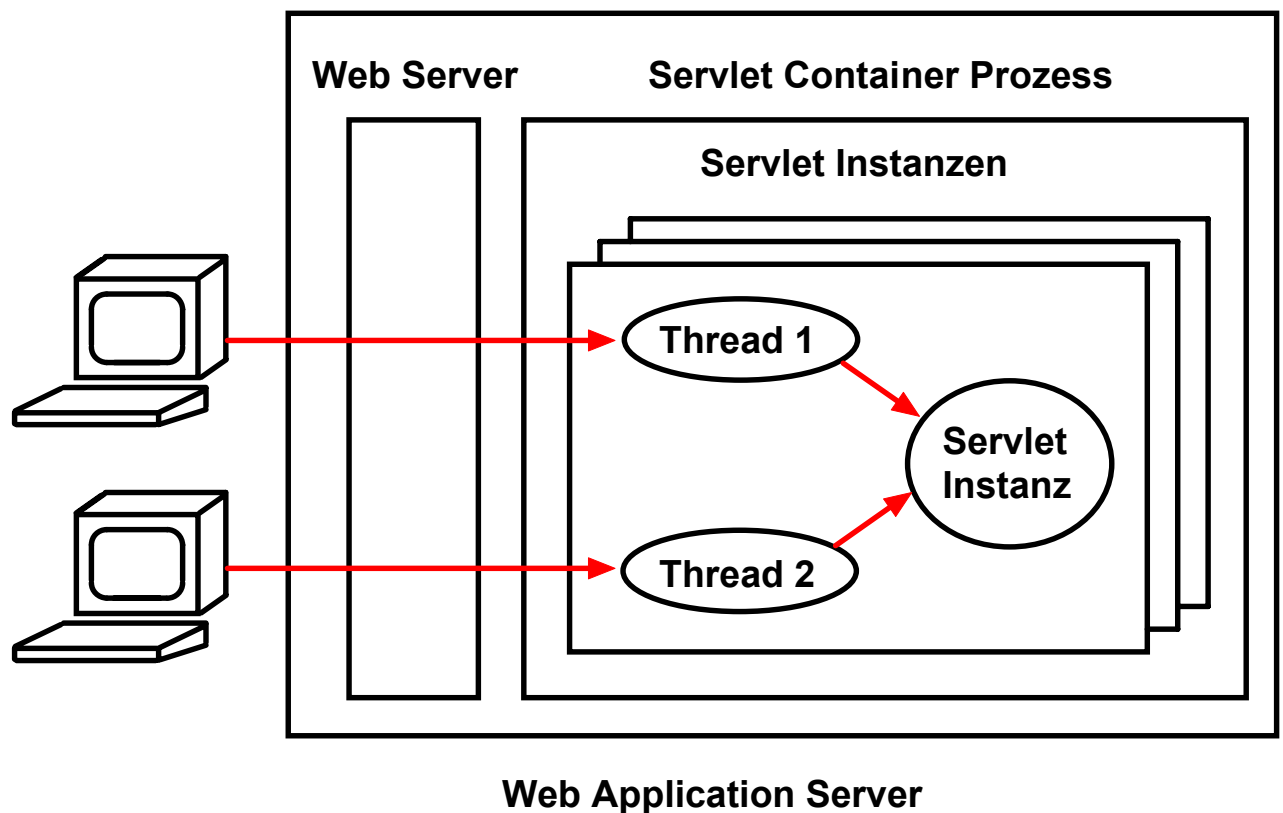


Dynamischer WEB Seiten Inhalt (2)

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

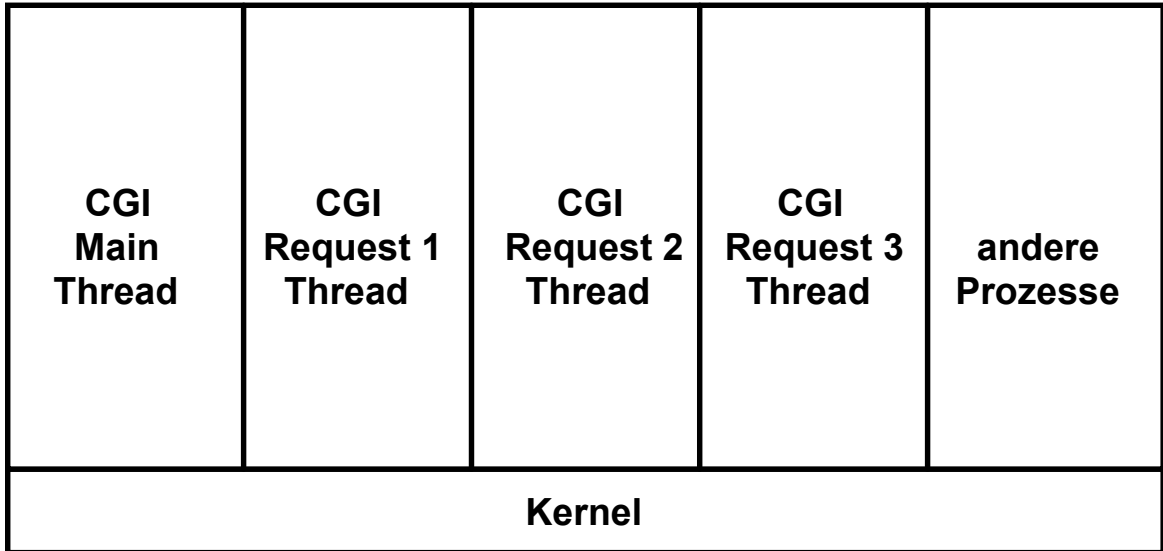
Es existiert nur eine einzige Instanz des Servlets

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

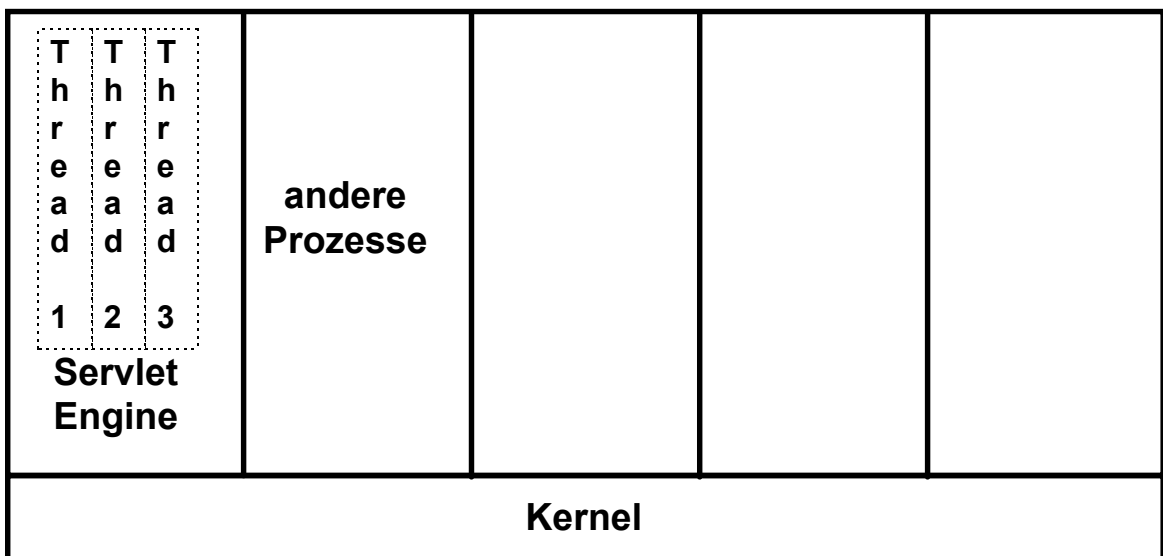
FF..FF



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

Servlets laufen in speziellen Servlet-spezifischen Containern, die auch als Servlet Engines bezeichnet werden. Sie verbessern u.a. die Servlet-Ausführungszeit. Servlet Container haben keine Transactions-, Persistence- und Sicherheitseigenschaften. Ein Servlet Container ist ein Programm, 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.**

HalloWeltServlet.java

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

public
class HalloWeltServlet 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 HalloWeltServlet init");
    }

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

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

Java Server Pages (JSP)

Java Server Pages sind in der Java Programmiersprache geschrieben.

Benutzen XML-artige Tags und Scriptlets 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 Scriptlets 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)

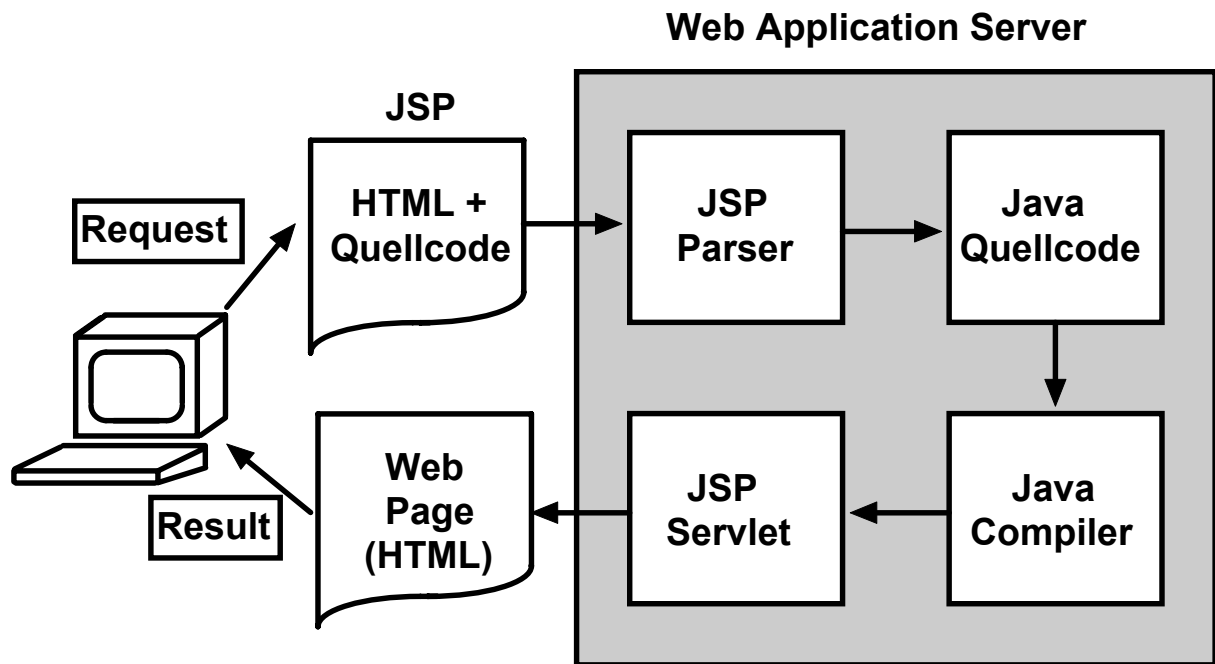
siehe auch Tutorials 14 und 15

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. He can use JSP to access reusable components, such as servlets, Java beans, enterprise beans, and Java-based Web applications

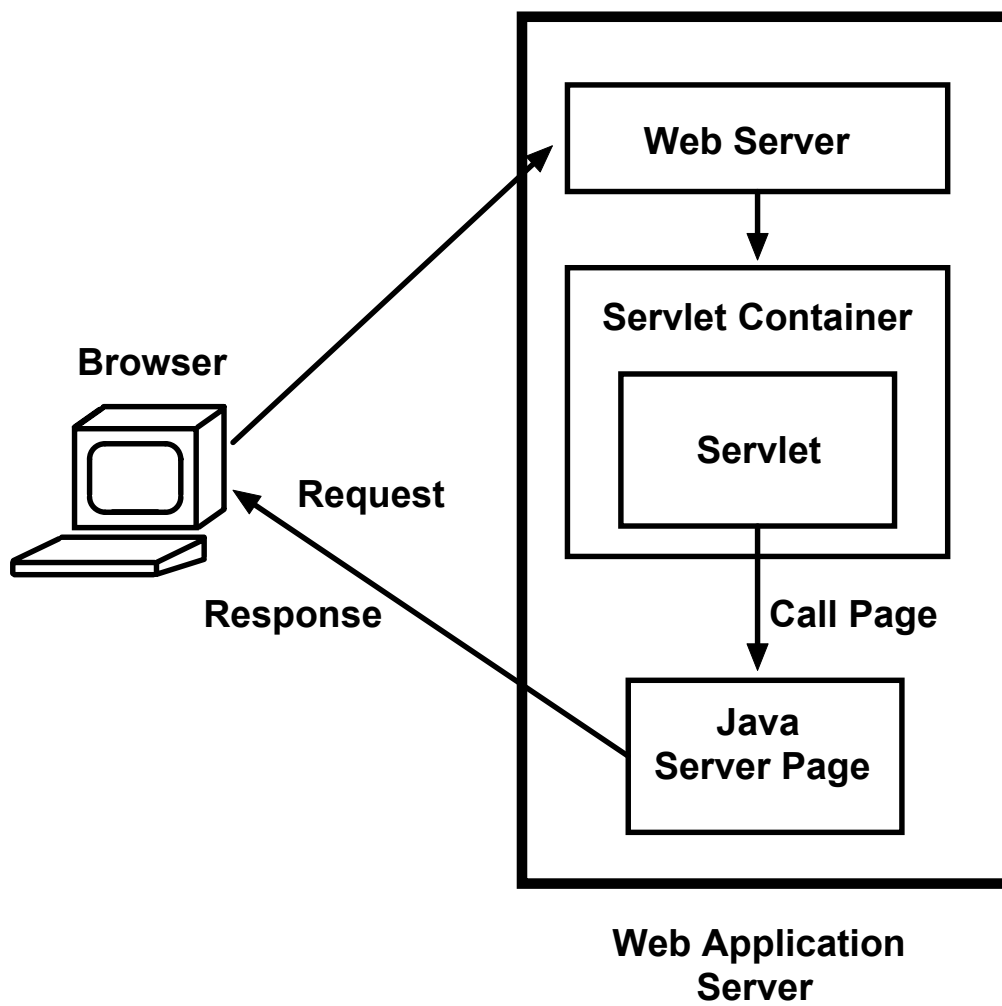
Einfache Java Server Page

```
1| <!doctype html public "-//w3c//dtd html 4.0 transitional//en">
2| <html>
3| <head>
4| <title>JSP Example 1</title>
5| <meta http-equiv="Content-Type" content="text/html;
6|   charset=iso-8859-1">
7| <meta name="Author" content="Paul Tremblett">
8| <meta name="GENERATOR" content="Mozilla/4.51 [en] (X11; I;
9|   Linux 2.2.5-15 i586) [Netscape]">
10| </head>
11| <body bgcolor="#FFFFFF">
12| <p>
13| <font face="Arial, Helvetica, sans-serif"><b><font size="+2">
14| JSP Example 1
15| </font></b></font>
16| <br>
17| <br>
18| <font face = "Arial, Helvetica"><font size="+1">
19| It is now
20| <%=
21| new java.util.GregorianCalendar(new java.util.SimpleTimeZone
22|   (-5*60*60*1000,"EDT")).getTime()
23| %>
24| </font>
25| </body>
26| </html>
```



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

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.

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

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.

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>

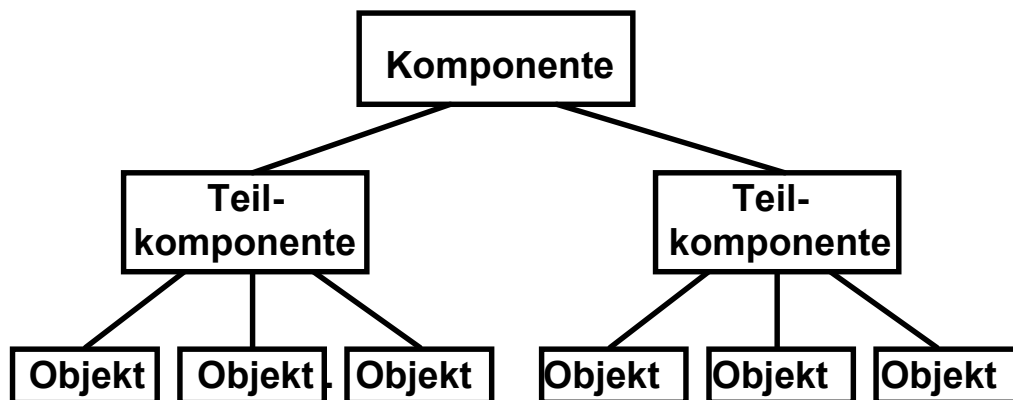
Teilweise erfolgt ein Zugriff auf die OS/390 DB2 Datenbank

Für die Erstellung dieser Beispiele existieren ausführliche Tutorials.

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.**



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)**

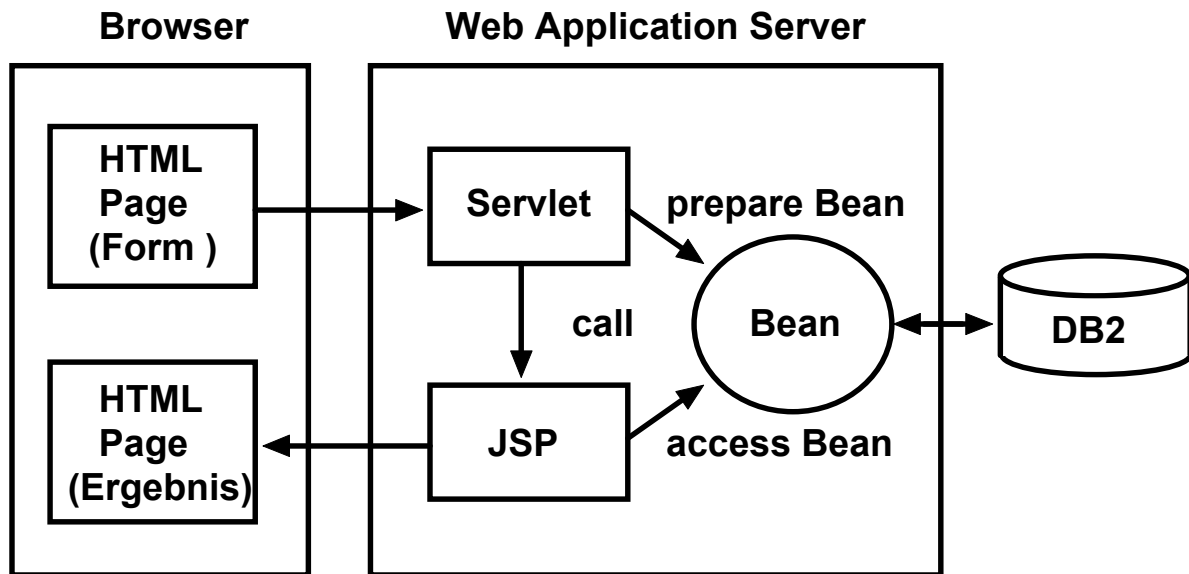
Namens Konventionen

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 Schlüsseigenschaften, z.B Transaktionsdienste, Namensdienste und Sicherheitsdienste. Werden JavaBeans hiermit angereichert, spricht man von Enterprise JavaBeans.



Java Server Pages (JSP)

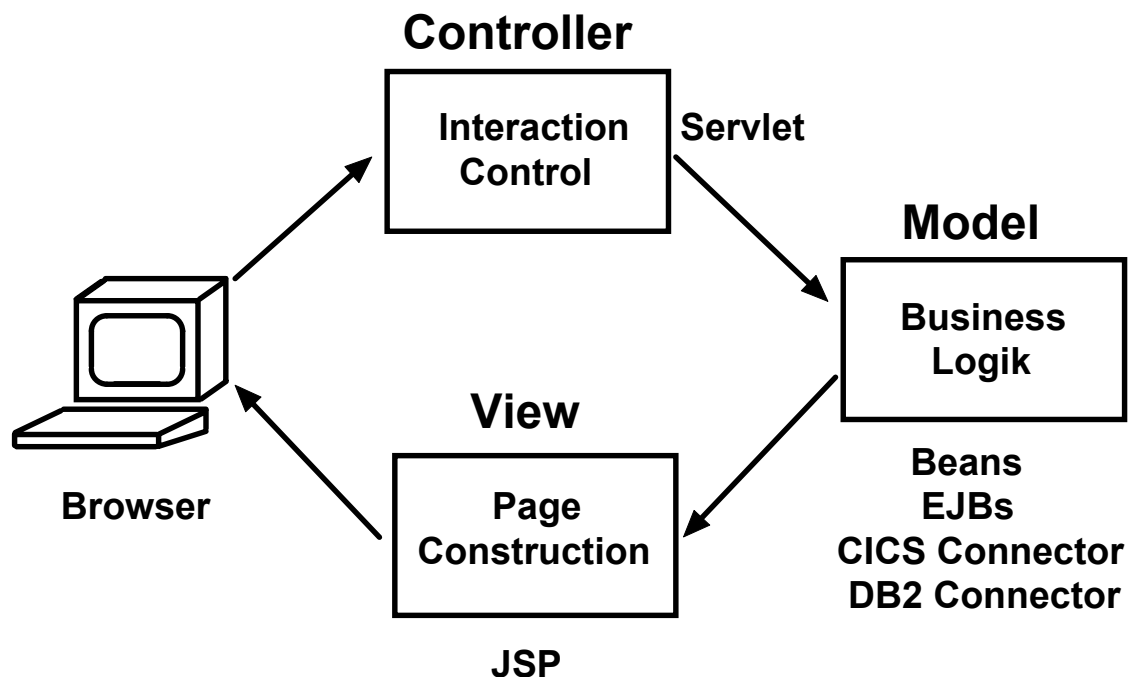
Ein Servlet ist ein Java Programm, das Bildschirm Output in der Form einer HTML Datei produziert.

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.

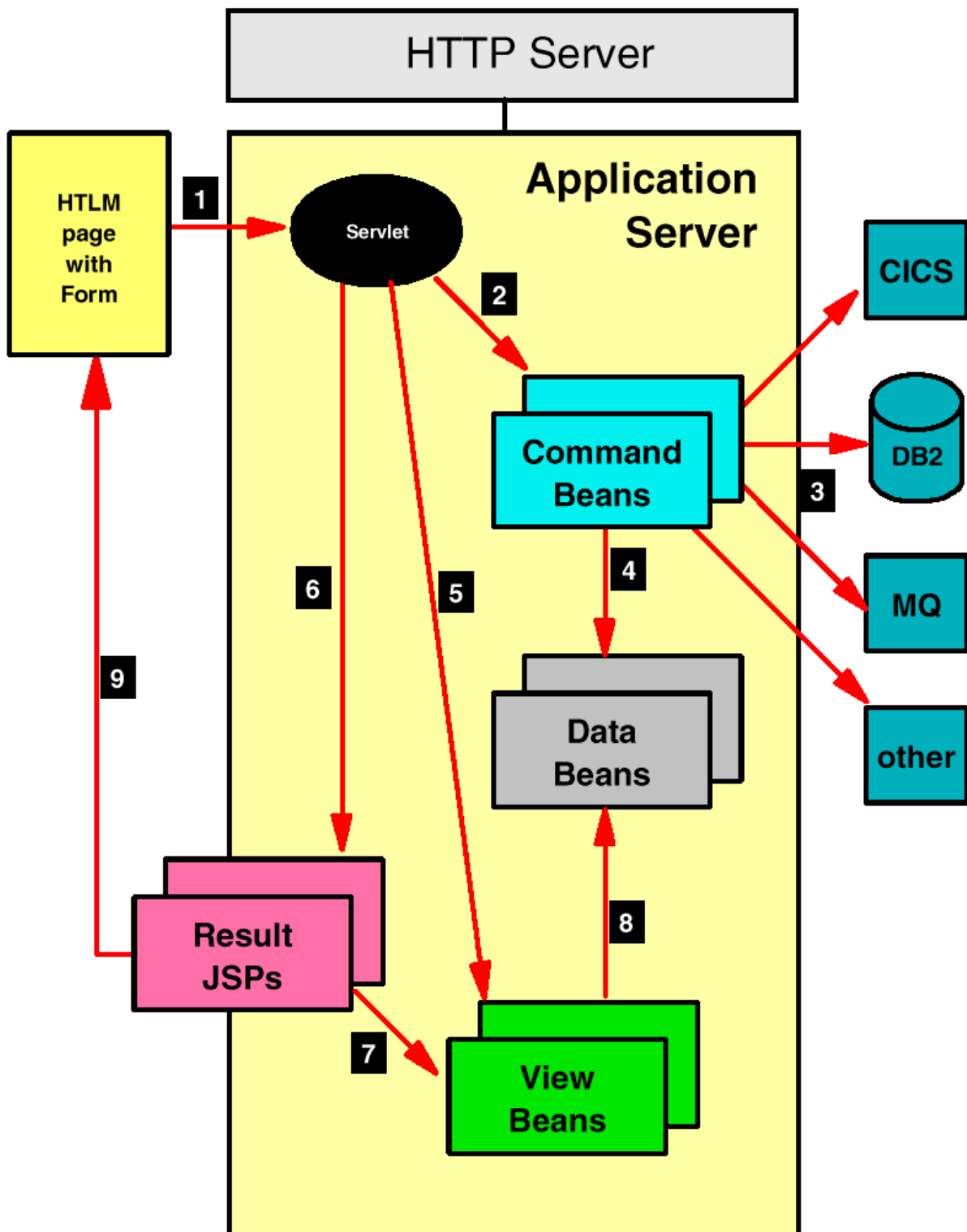
Command- und Data Beans oder Enterprise Java Beans (plus häufig CICS, IMS Programme, oder Stored Procedures) sind das „Modell“ (=Business Logik).

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.

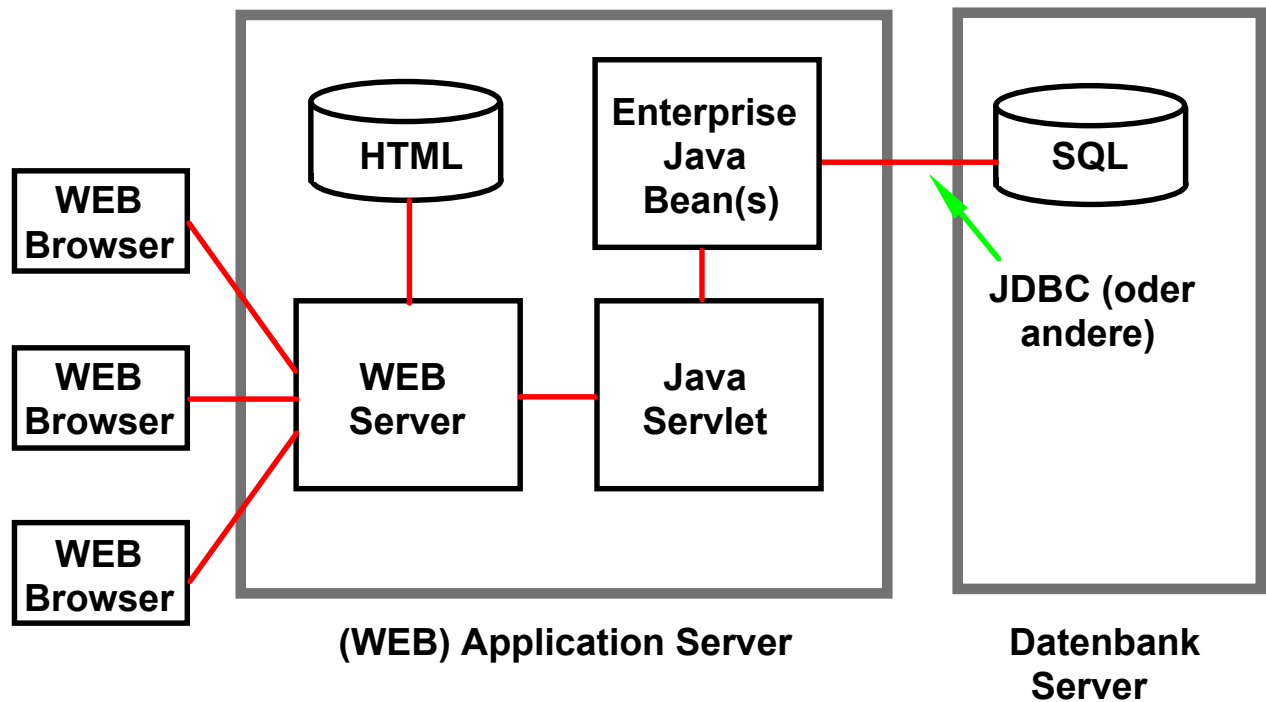


Architektur einer JSP Web Anwendung

Model - View - Controller Ansatz

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.
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. Final Release der Version 1.1 der EJB Spezifikation erfolgte im Dezember 1999

- 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, skalierbar und sicher. Diese und weitere Systemfunktionen wie, Life-cycle management, threading und Persistenz werden 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, Persistenz, usw.
- Die permanente Speicherung eines Objektes auf einem Plattenspeicher wird als Persistenz bezeichnet. Konzeptuell können Objekte in einer Objektdatenbank (z.B. POET) gespeichert werden. In der Praxis werden SQL (oder IMS oder VSAM) Daten als Objekte gekapselt; der Zugriff erfolgt z.B. über eine JDBC (Java Data Base Connectivity), SQLJ oder DB2Connect Schnittstelle.
- Interoperabilität von EJBs und CORBA konformen Objekten ist möglich.

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

Enterprise Java Beans (EJB)

Java basiertes Server Komponentenmodell, März 98

EJBs sind in einen „Container“ eingebettet (über die EJB API angeschlossen).

EJB Spezifikation legt fest:

- Regeln, denen eine EJB entsprechen muß
- Funktionalität, die der Container über entsprechende Schnittstellen erbringt
- Art der Beschreibung einer EJB für die Installation und Verteilung
- Abbildung auf das CORBA IIOP Protokoll

Eigenschaften einer EJB und deren Anforderungen an die Ablaufumgebung (z.B. Sicherheit, Transaktionssteuerung) werden durch die Attribute eines „DeploymentScriptors“ (Metadaten) deklarativ beschrieben.

Interoperabilität von EJBs und CORBA konformen Objekten ist möglich

Erweiterungen von Transaktionsmonitoren wie CICS (IBM) oder UTM (Siemens) ermöglichen Verteilung, Integration und Message Queuing für EJB Komponenten (Object Transaction Monitor, OTM)

cs1223z ww6

wgs 11-98

Persistenz

Das Objektmodell macht zunächst keine Annahmen über die permanente Speicherung der Objekte. Konzeptuell können Objekte in einer Objektdatenbank (z.B. POET) gespeichert werden.

In der Praxis werden SQL (oder IMS oder VSAM) Daten als Objekte gekapselt; der Zugriff erfolgt z.B. über eine JDBC (Java Data Base Connectivity) Schnittstelle.

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

cs 1418 ww6

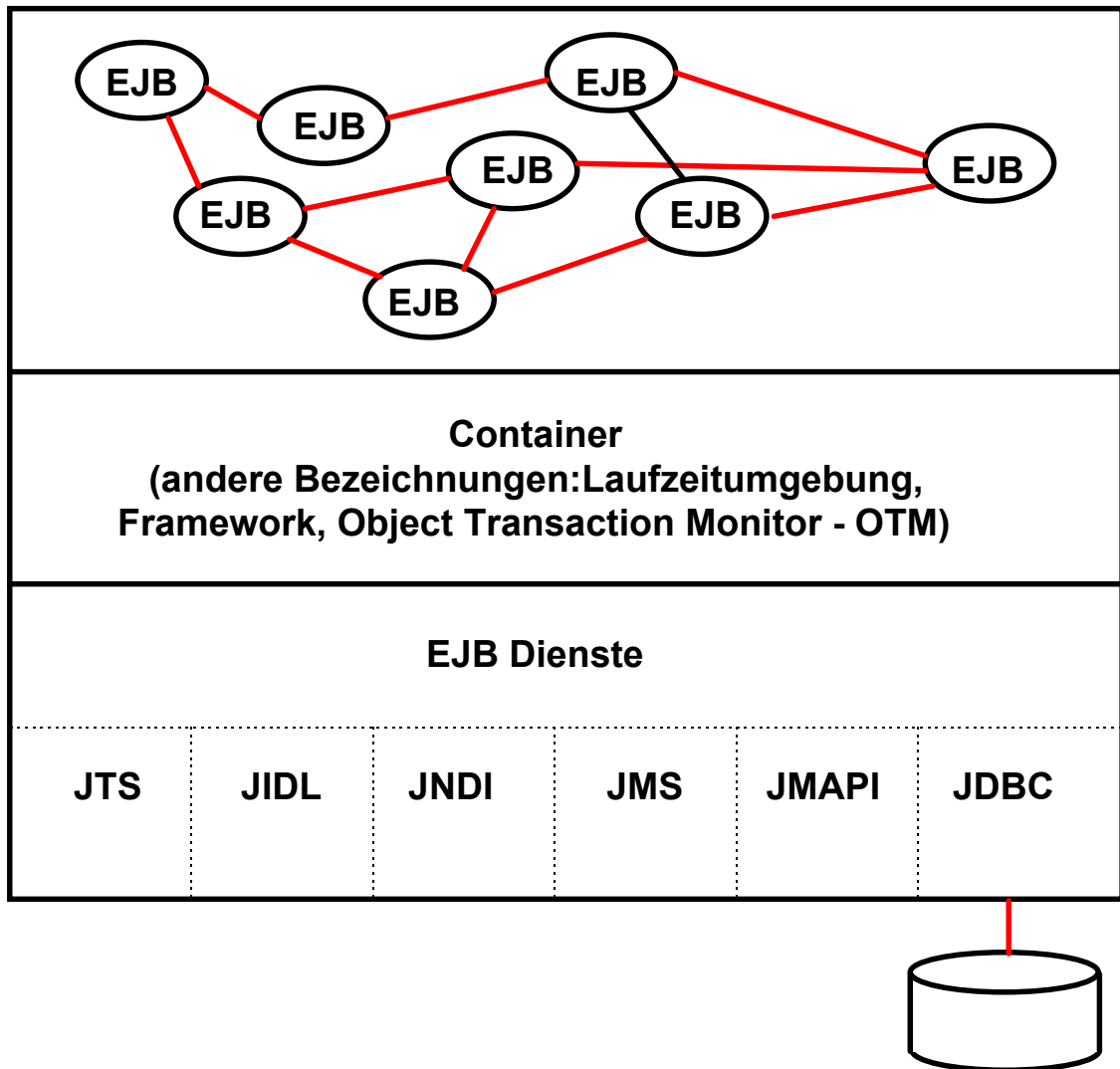
wgs 03-00

Schlüsseigenschaften der EJB Technologie

- **EJB Komponenten sind serverseitige Komponenten, die ausschließlich in Java geschrieben sind**
- **EJB Komponenten enthalten nur Business Logik, keine Systemfunktionen**
- **Systemfunktionen wie Transaktionsdienste, Sicherheit, Life-cycle, threading, Persistenz werden automatisch für die EJB Komponente von dem EJB Server zur Verfügung gestellt**
- **Die EJB Architektur ist inhärent transaktionsorientiert, distributed, portierbar, multi-tier, skalierbar und sicher**
- **EJB Komponenten werden declarativ zur Laufzeit angepaßt. Die Anpassung bezieht sich auf: Transaktionsverhalten, Sicherheitseigenschaften, life-cycle, state management, Persistenz, usw.**
- **EJB Komponenten sind voll portierbar über Betriebssystem und EJB Server Grenzen hinweg**
- **Für die Zusammenarbeit mit relationalen Datenbanken und anderen Datenquellen vorgesehen**

Final Release der Version 1.1 der EJB Spezifikation erfolgte im Dezember 1999

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 is the Java Transaction Service, an API for invoking transaction services.

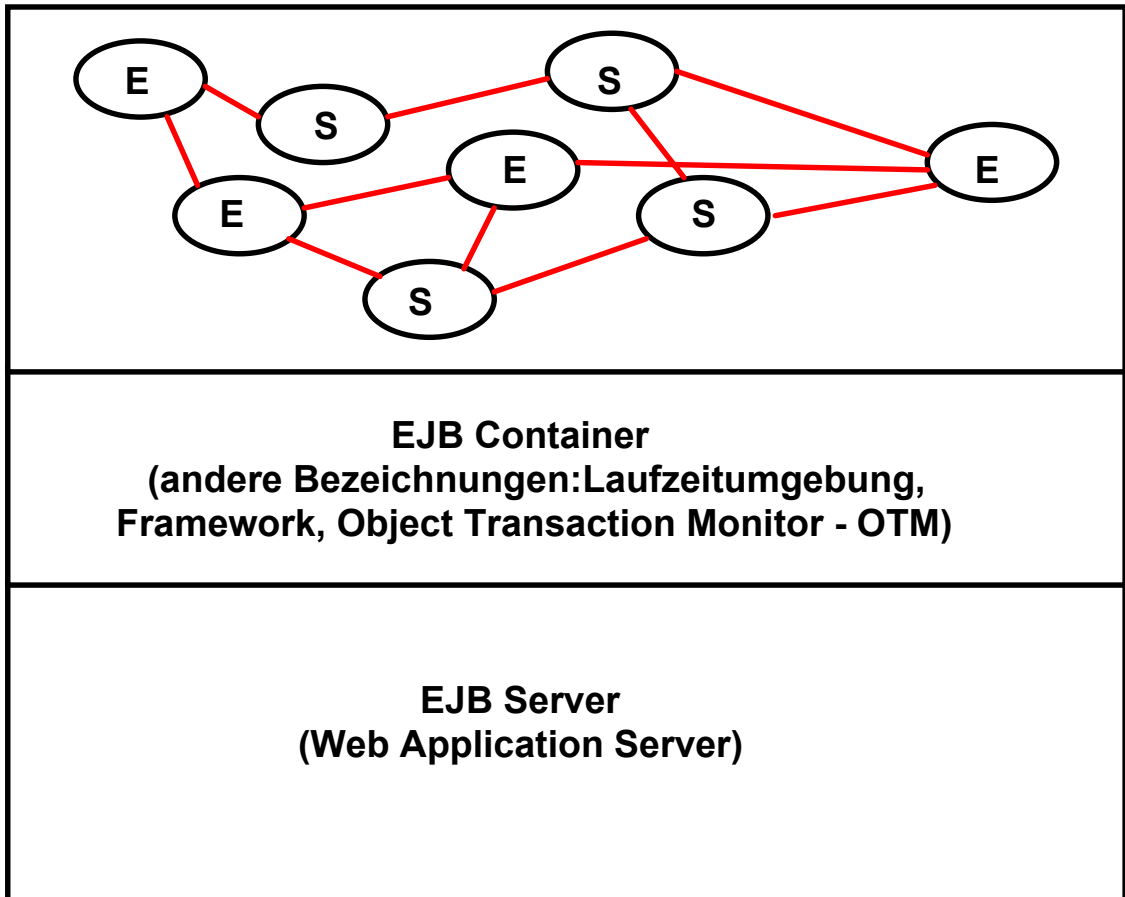
JNDI, the Java Naming and Directory Interface, is an API for accessing naming and directory services.

JMS, the Java Message Service, is an API for invoking asynchronous message delivery services.

JDBC , the Java Database Connectivity API, accesses data in existing databases through a common interface.

JMAPI refers to the Java Management API, which defines access to a set of services for managing Java resources.

JIDL refers to Java interface definition language, an interface to the CORBA set of services for distributed computing.



S Session Bean (transientes Objekt)
 E Entity Bean (persistentes Objekt)

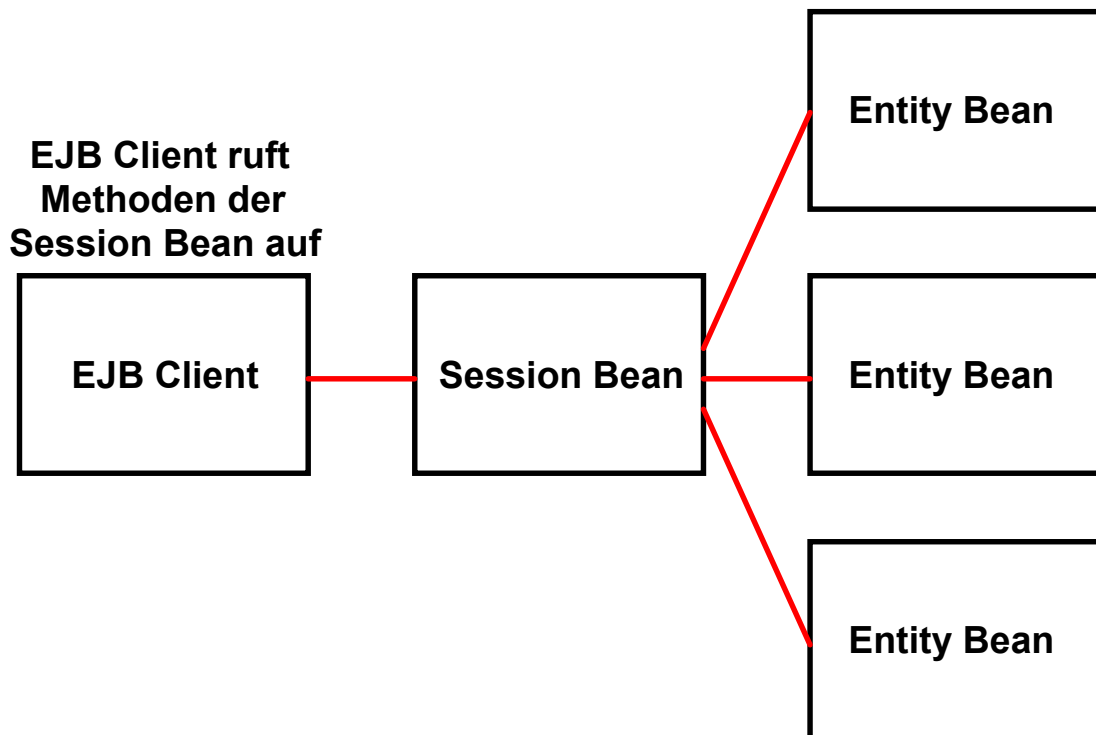
Arten von EJBs

Entity Beans

- repräsentieren spezifische Daten (z.B. eine Reihe in einer SQL Datenbank)
- Methoden ermöglichen die Manipulation der Daten, welche die Bean repräsentiert
- überleben, so lange die Daten in der Datenbank überleben.

Session Beans

- können den internen Zustand einer Session speichern, der aber nicht persistent ist
- Lebensdauer ist häufig auf die Zeit einer einzigen Client/Server Session begrenzt.



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:

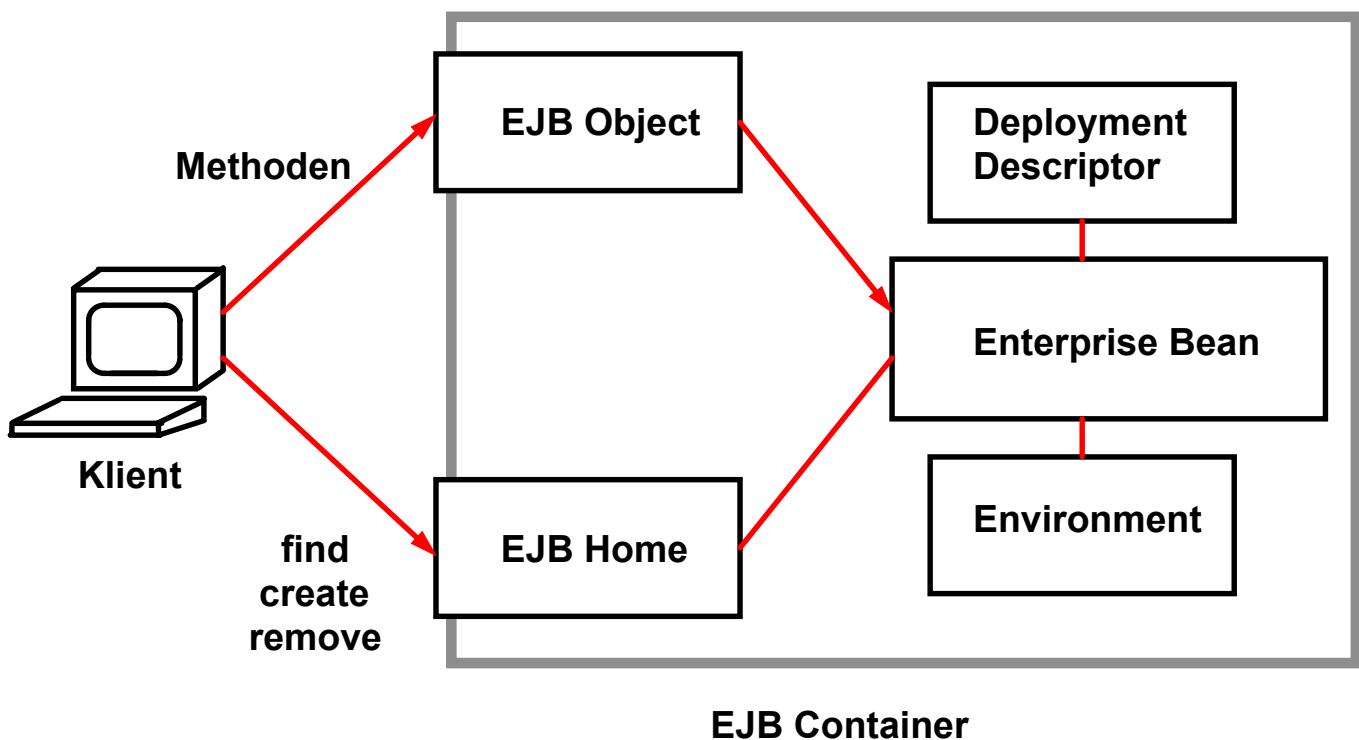
- Entity Beans sind verteilte Objekte, die von beliebigen Klienten aufgerufen werden können
- Entity Beans sind transaktionsfähig

Beides ist bei einer Session Fassade unnötiger Overhead.
 Lösungsansatz : „Java Data Objects“ (JDO) an Stelle der Entity Beans. JDOs sind reguläre Java Klassen, die über einen Persistence Encapsulation Mechanismus verfügen.

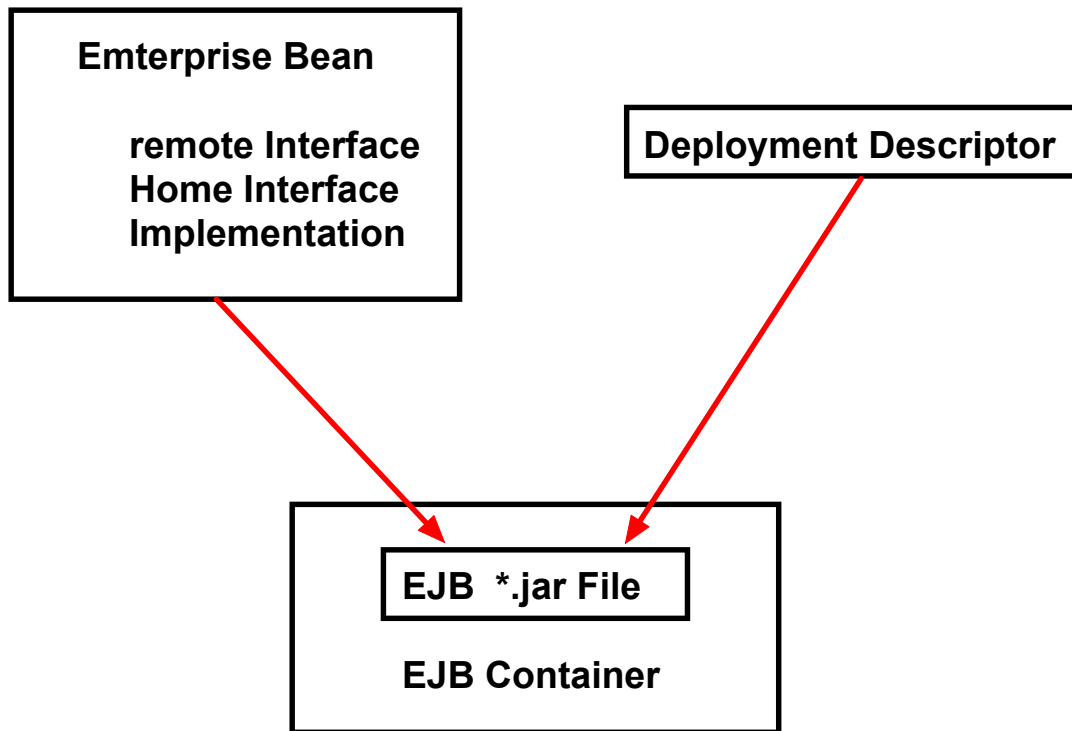
Weitere Alternative: „Konnektoren“ sind Java Klassen, die einen Zugriff auf CICS, Oracle, DB2 ermöglichen.

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

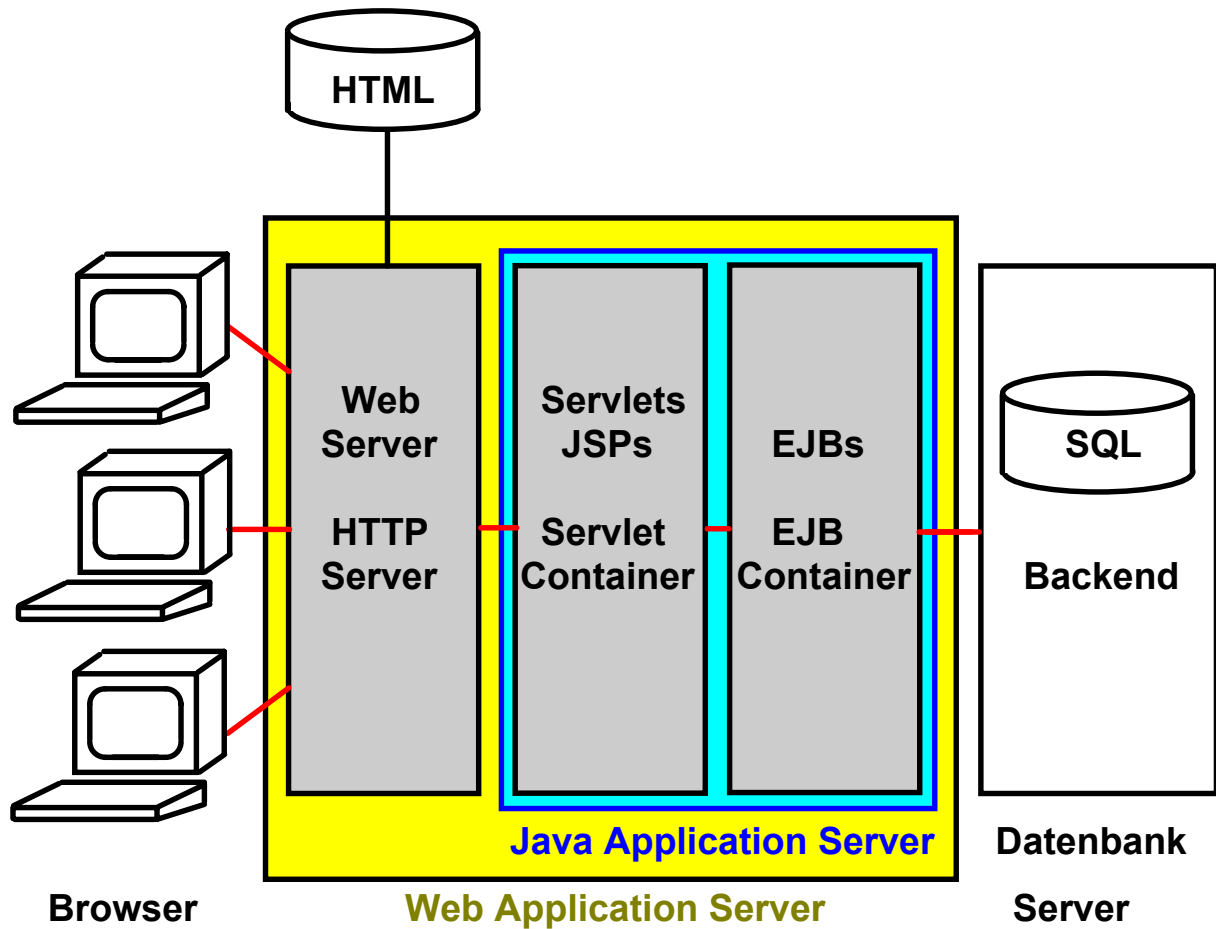
Parameter können statisch zur Assembly Zeit oder dynamisch zur Laufzeit festgelegt werden

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



Führende Vertreter

BEA WebLogic
IBM WebSphere

Weitere Implementierungen

SAP
Oracle
Sun One Application Server (bisher iPlanet)
JBoss - Tomcat
Iona Orbix
Inprise

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.**

2839 ww6

wgs 04-01

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**

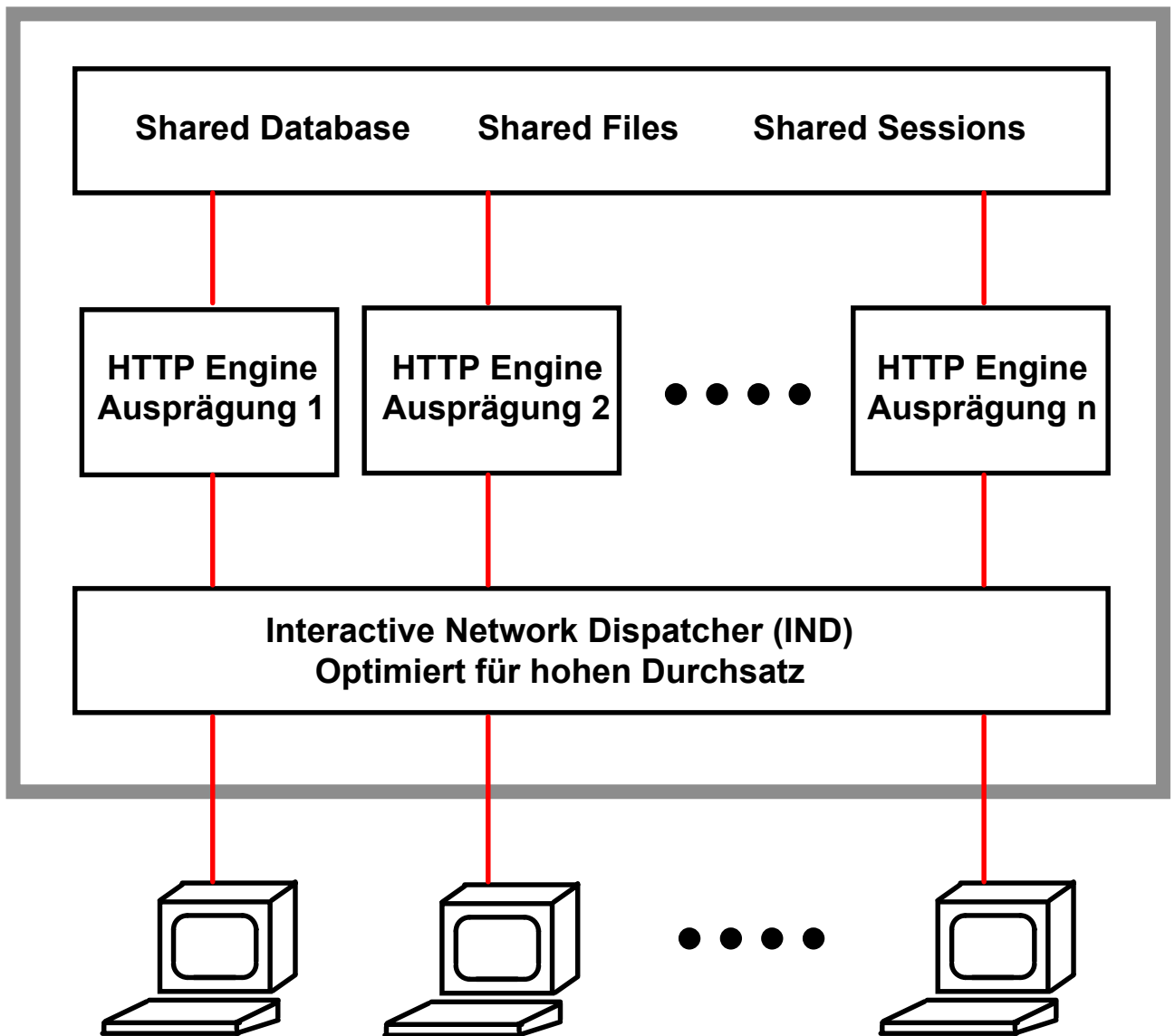
Verhältnis 5 : 1 Spitzen- zu Durchschnittsbelastung. Verhältnis 10 : 1 nicht selten.

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

Faktor 10 Wachstum erwartet in wenigen Jahren

cs 1454 ww6

wgs 03-00



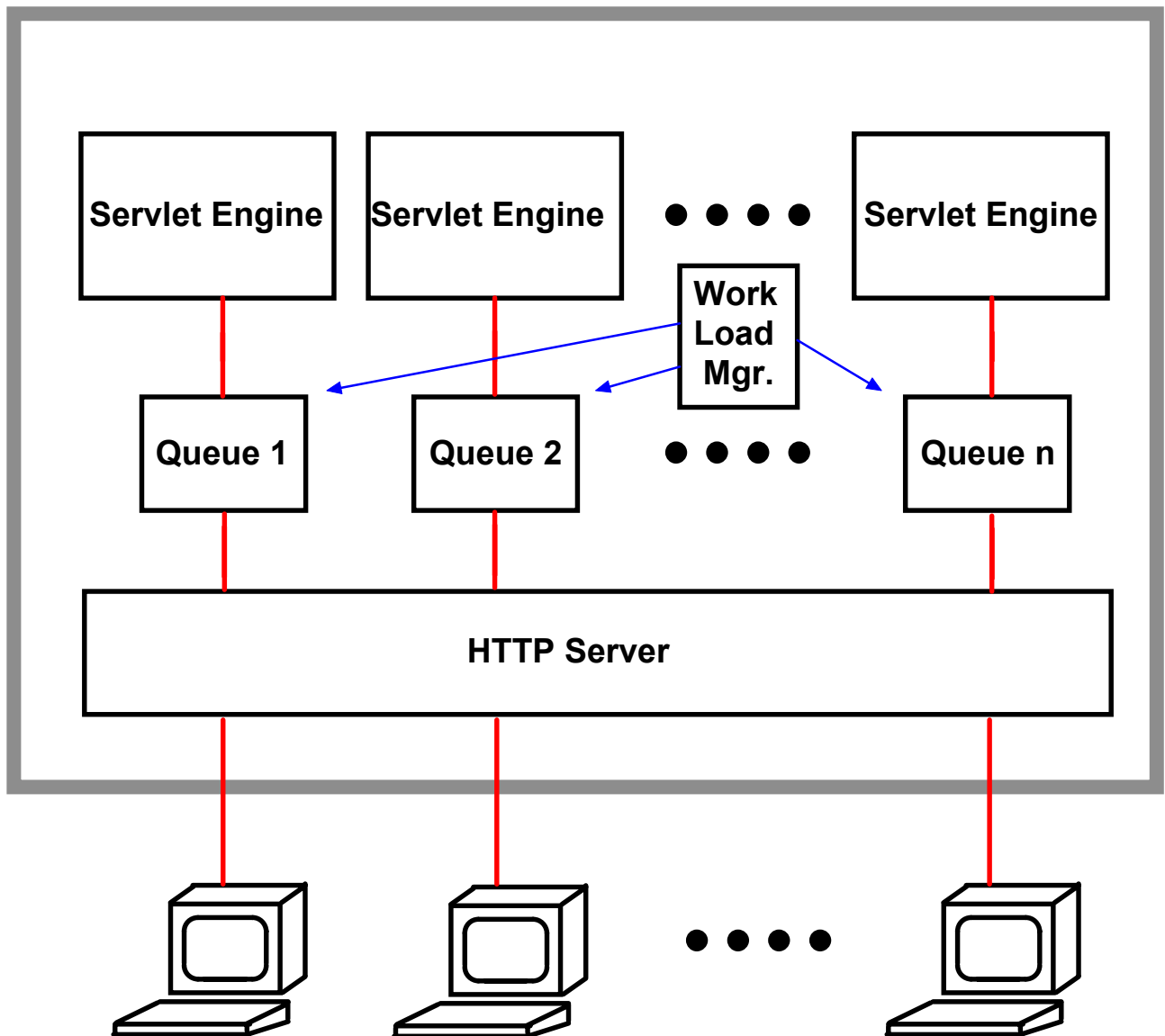
HTTP Server

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



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

cs 1451 ww6

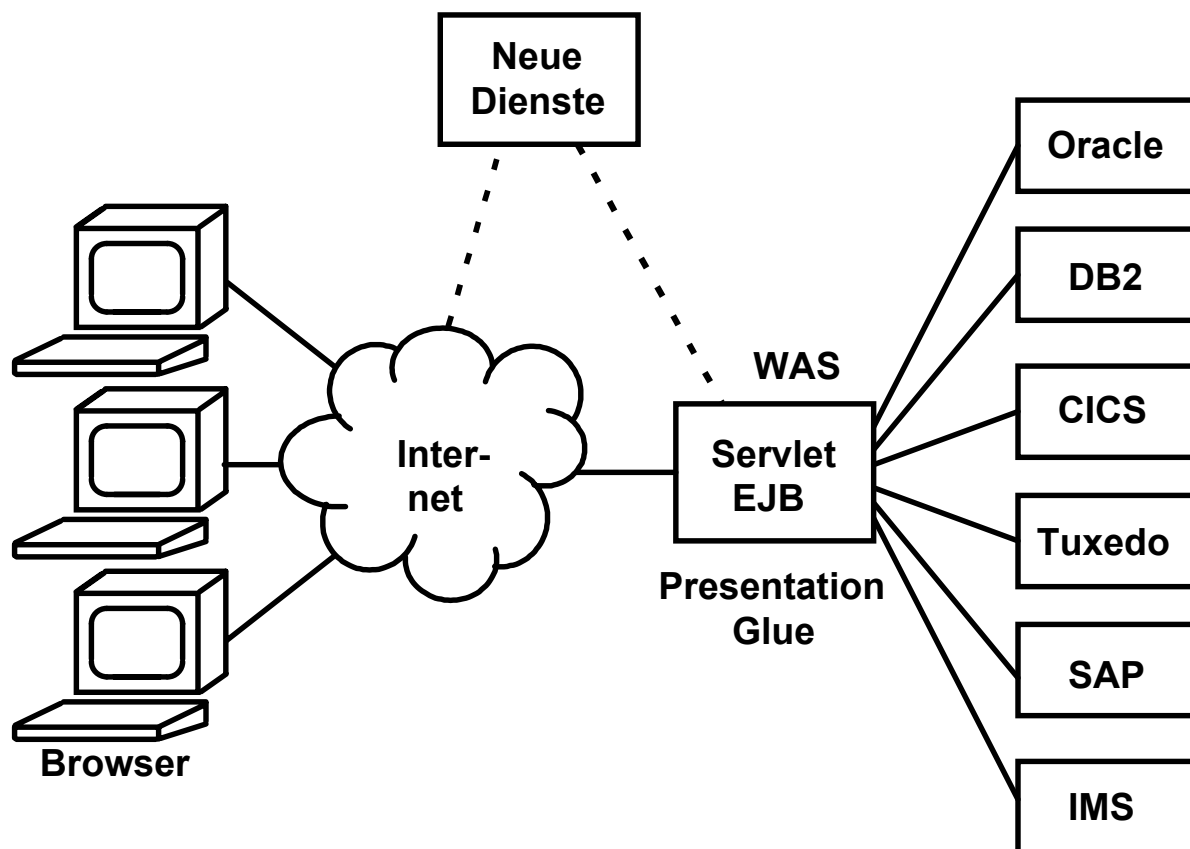
wgs 03-00

WebSphere Funktionen unter OS/390

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

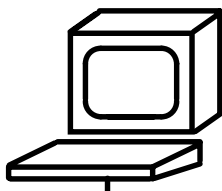
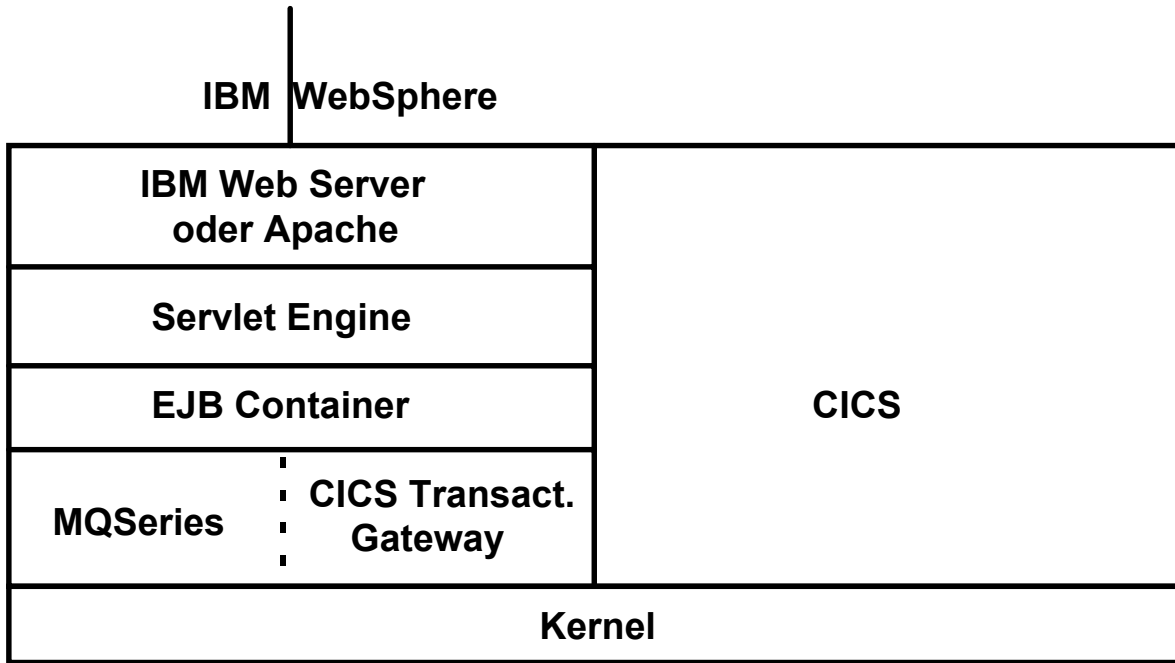
cs 1452 ww6

wgs 03-00

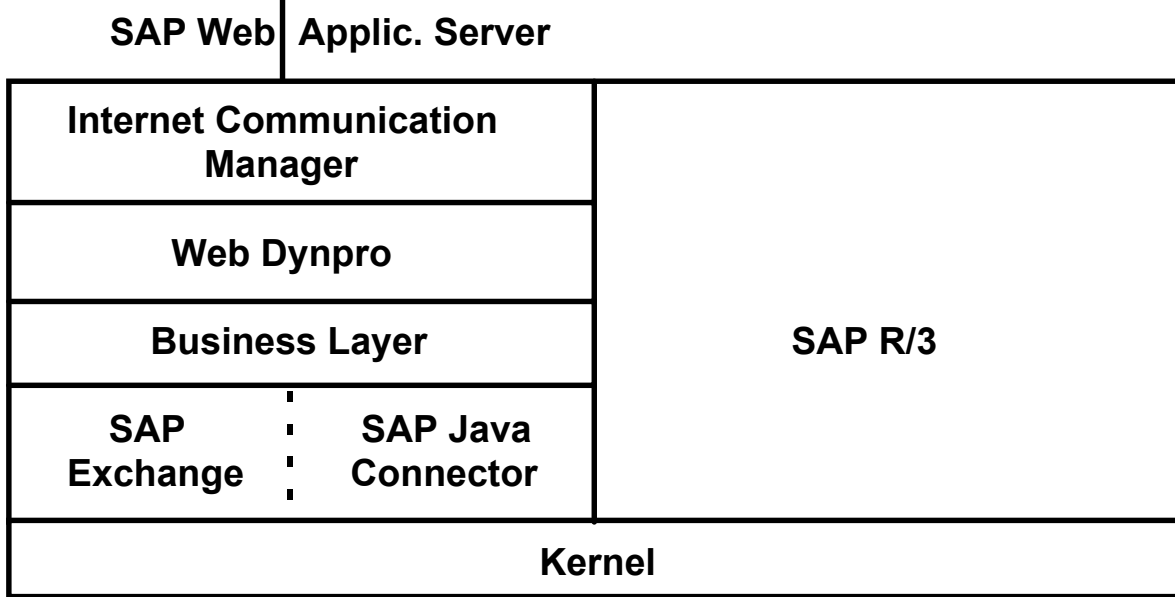


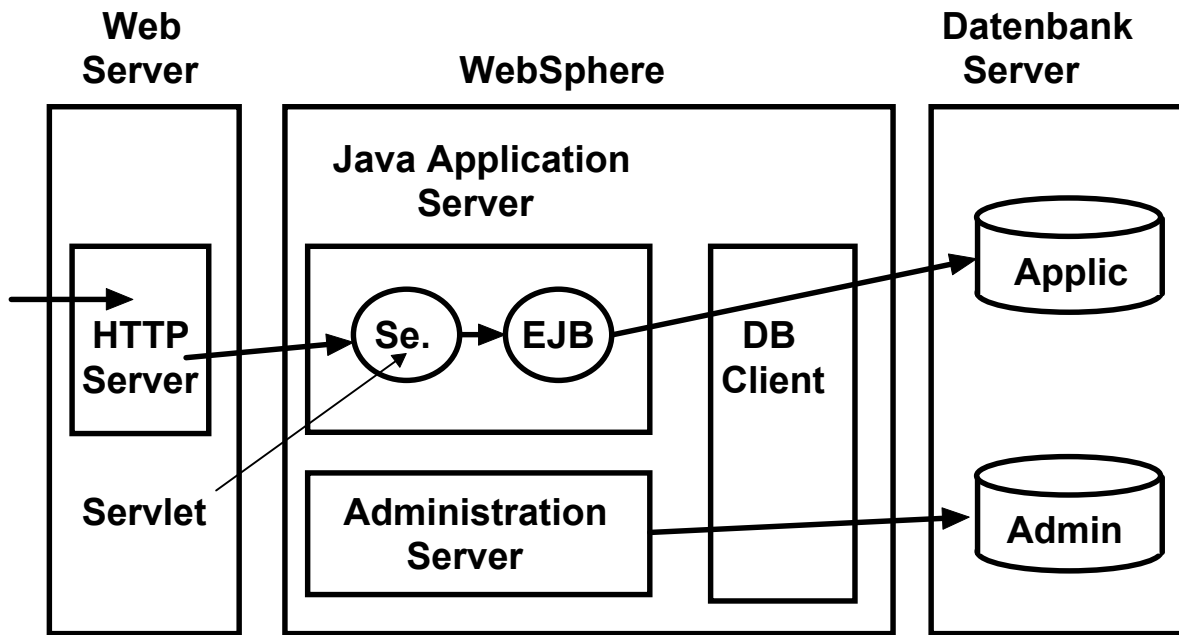
Client/Server Aufgabenstellung

- Browser orientierter Web Zugang
- Datenhaltung in existierenden Datenbanken
- Dominierender Anteil der Business Logik in existierenden Transaktionsprogrammen und/oder Stored Procedures
- Neue Software (z.B. EJBs) stellen Querverbindungen zwischen existierenden Komponenten her (Glue)
- System Management - TCO



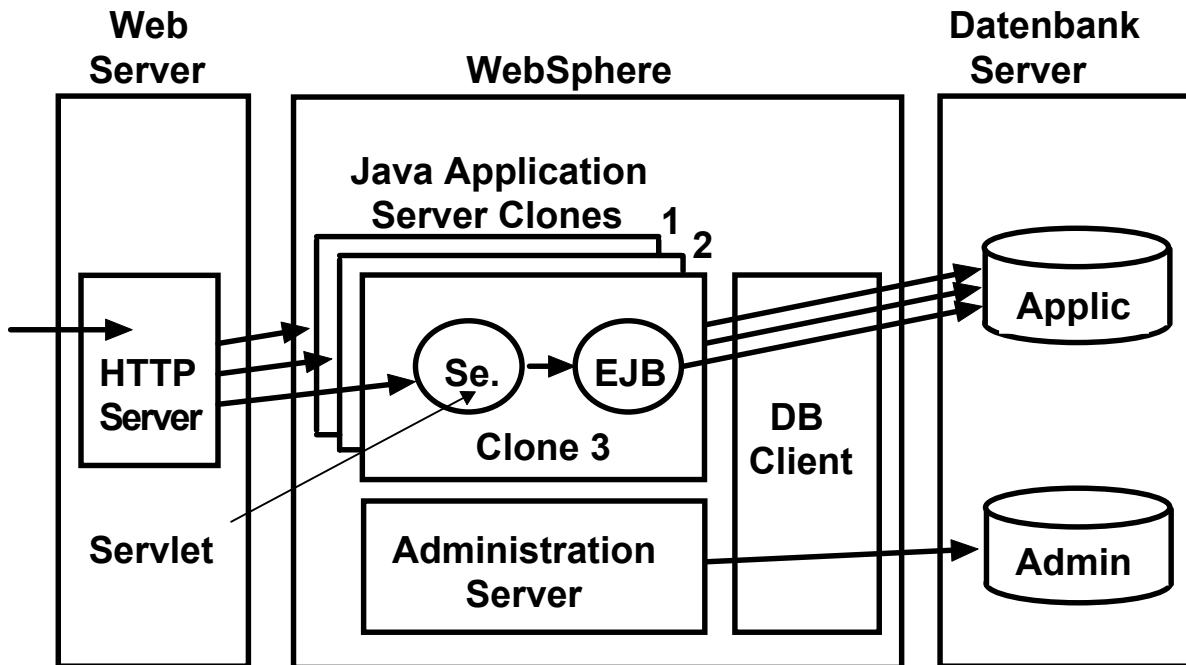
**Web Application Server
als Tranaktionsmonitor
Front End**





Web Application Server

Gemeinsame JVM für Servlet Engine und EJB Container
kein RMI für Servlet - EJB Kommunikation



Mehrfache Clones des Java Application Servers

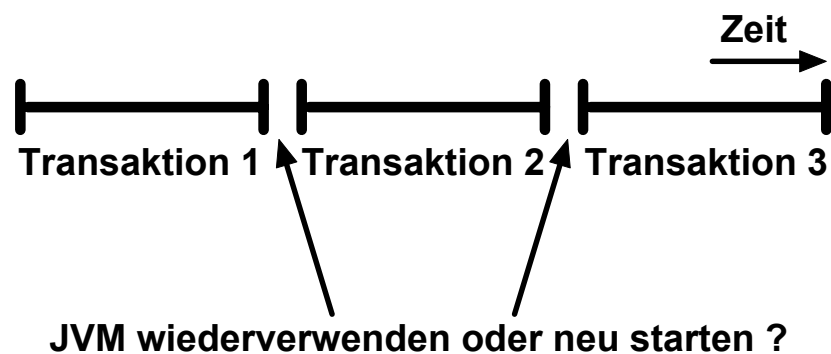
jeder Clone mit einer eigenen Java Virtuellen Maschine

Ausführung mehrerer Transaktionen hintereinander

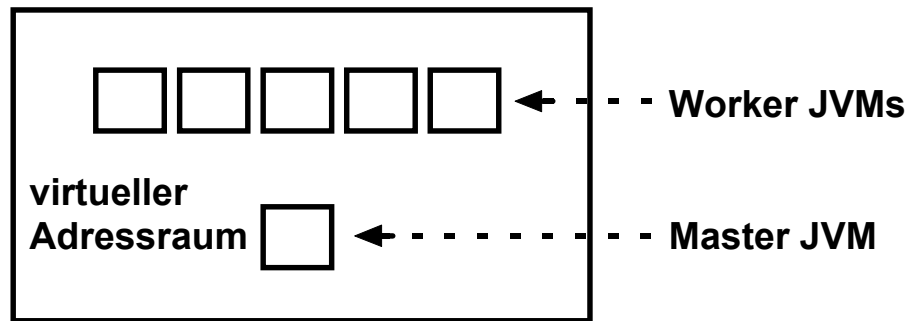
Wird dieselbe JVM für mehrere in sich abgeschlossene Transaktionen hintereinander benutzt, ist es grundsätzlich möglich, dass eine Transaktion die Folge-Transaktion beeinflusst, indem sie den Zustand (State) der JVM ändert.

Beispiele:

- überschriebene statische Variablen
- geladene native Bibliotheken.



Lösung: JVM um zusätzliche Reset Methode anreichern
Persistent Reusable Java Virtuelle Maschine (PRJVM)



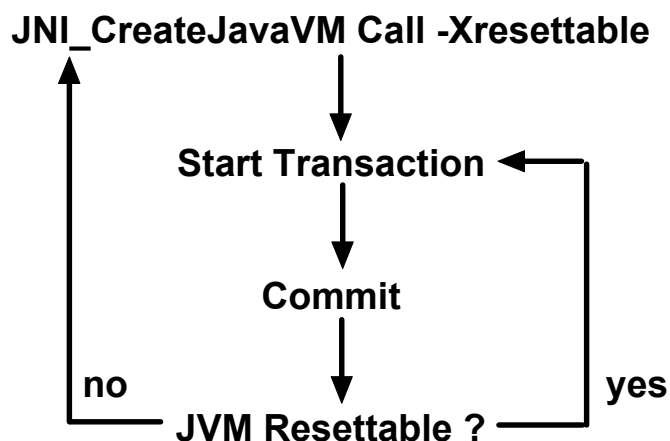
Master JVM startet Worker JVM mit dem Aufruf
JNI_CreateJavaVM mit der Option **-Xresettable**

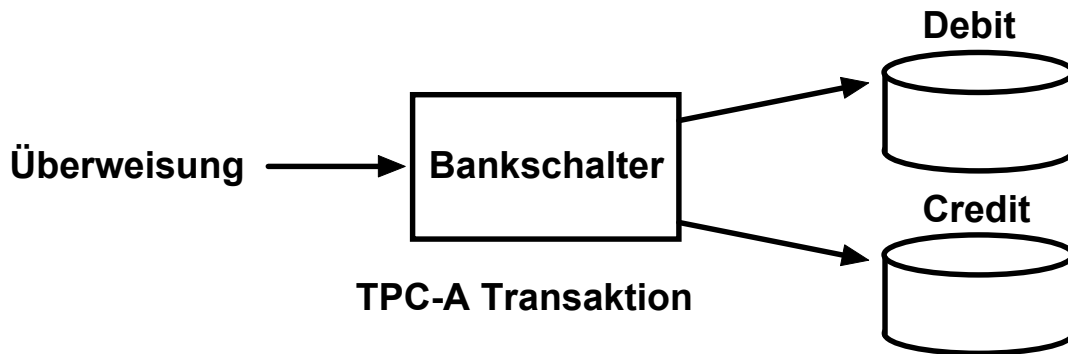
Es ist möglich, dass die Ausführung einer Transaktion die PRJVM in einen Zustand *unresettable* versetzt. Dies kann auf einem asozialen, jedoch legalen Verhalten der Tansaktion beruhen.

Beispiele

- übrig gebliebene geöffnete Files,
- noch existierende „user threads“
- Referenzen in den Transient Heap durch lokale Variablen.

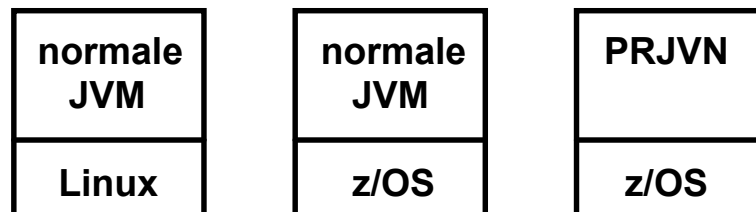
Wenn eine dieser Bedingungen auftritt, erzeugt der Aufruf **ResetJavaVM** den Rückgabewert *false*.





Diplomarbeit Marc Beyerle, Nov. 2003

Vergleichsmessungen, IBM zSeries Rechner, 1 CPU



Testumgebung

Ergebnisse

Normale JVM zLinux	normale JVM z/OS	Verhältnis zLinux zu z/OS	PRJVM z/OS	Verhältnis PRJVM zu normale JVM
Tx/s 1,223	Tx/s 0,796	1,54	Tx/s 261,04	327,94

Bei strikter Einhaltung der ACID Regeln brauchen EJBs die PRJVM