# Internet Anwendungen unter OS/390

**Dr. rer. nat. Paul Herrmannn**
**Prof. Dr.-Ing. Udo Kebschull**
**Prof. Dr.-Ing. Wilhelm G. Spruth**

**WS 2000/2001**

## Teil 6

## Web Application Server

# 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)**

# Internet Adressen

**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**

# Java Ausprägungen

**Java**

**Java Script**

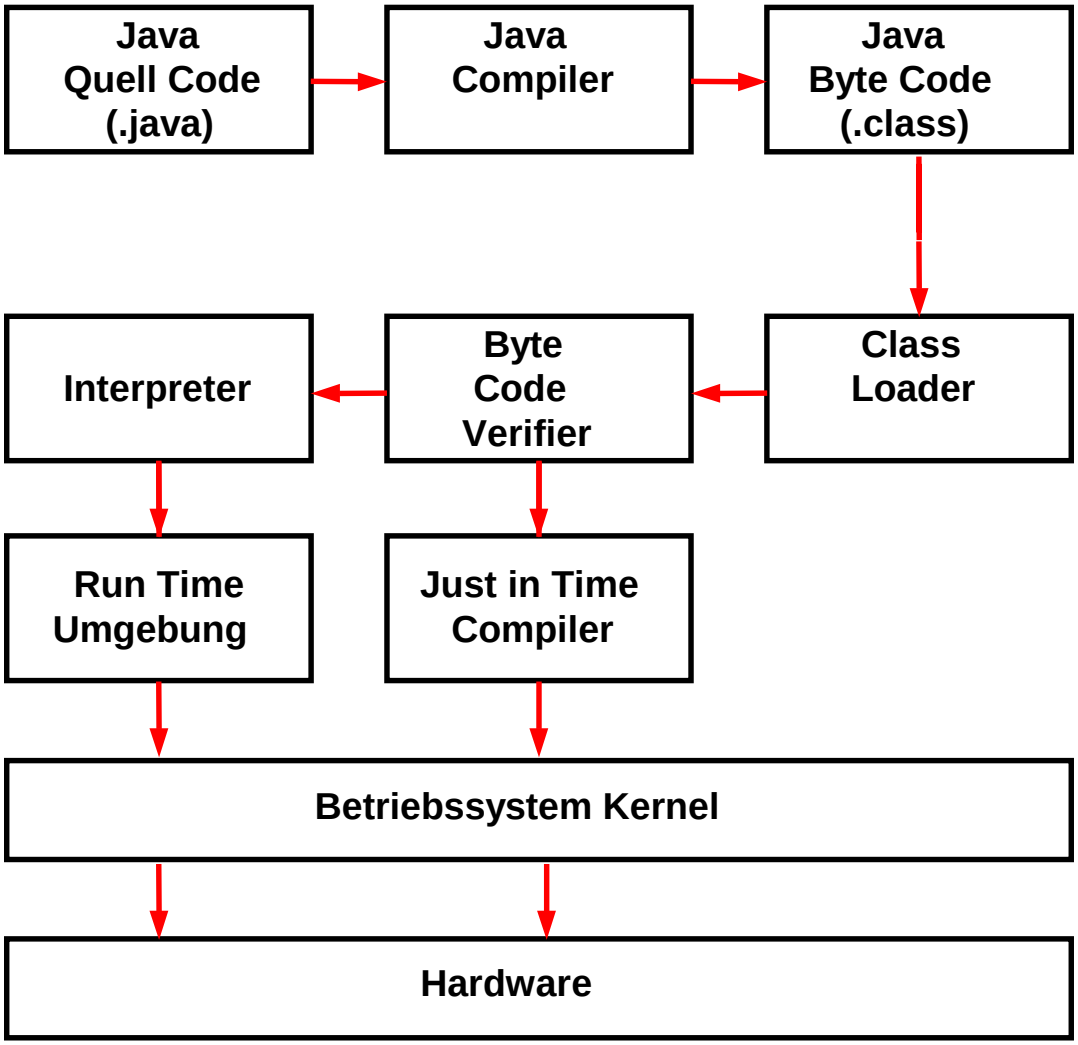
**Java Applet**

**Java Servlet**

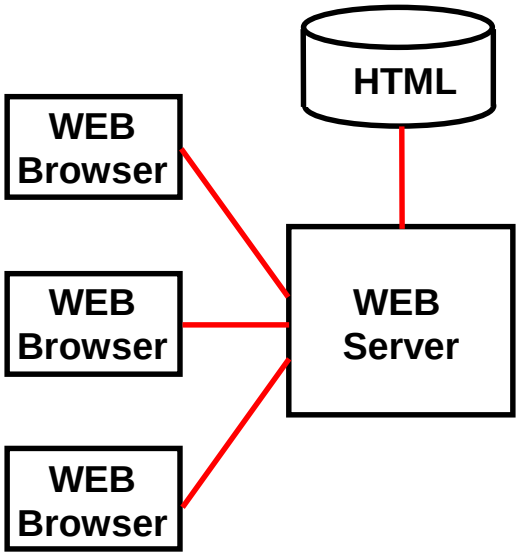
**Java Server Pages**

**Java Bean**

**Enterprise Java Bean**




# Java Virtuelle Maschine




**(WEB) Application Server** **Datenbank Server**

# 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.**

# 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**

# 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).**

2. **Klient sendet eine Request an den Server.**

   **Wird z.B. das URL**

   **www.qrx.de**

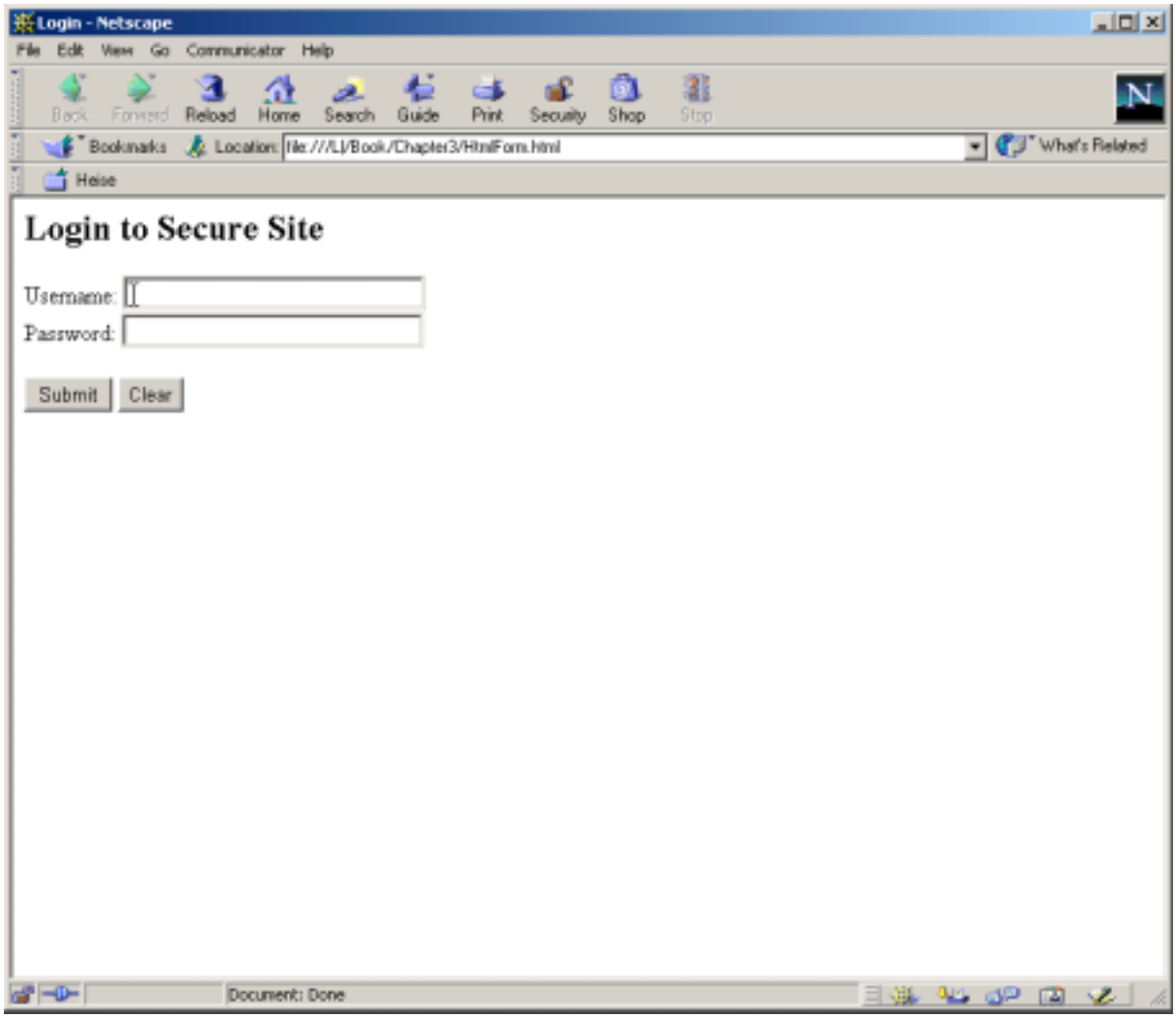
   **eingegeben, so sendet der Klient (Browser) die HTTP Nachricht**

   **GET /index.html  HTTP/1.0**

   **an den Server.**

3. **Der Server sendet eine HTML Seite als Antwort.**

4. **Die TCP Verbindung wird geschlossen.**



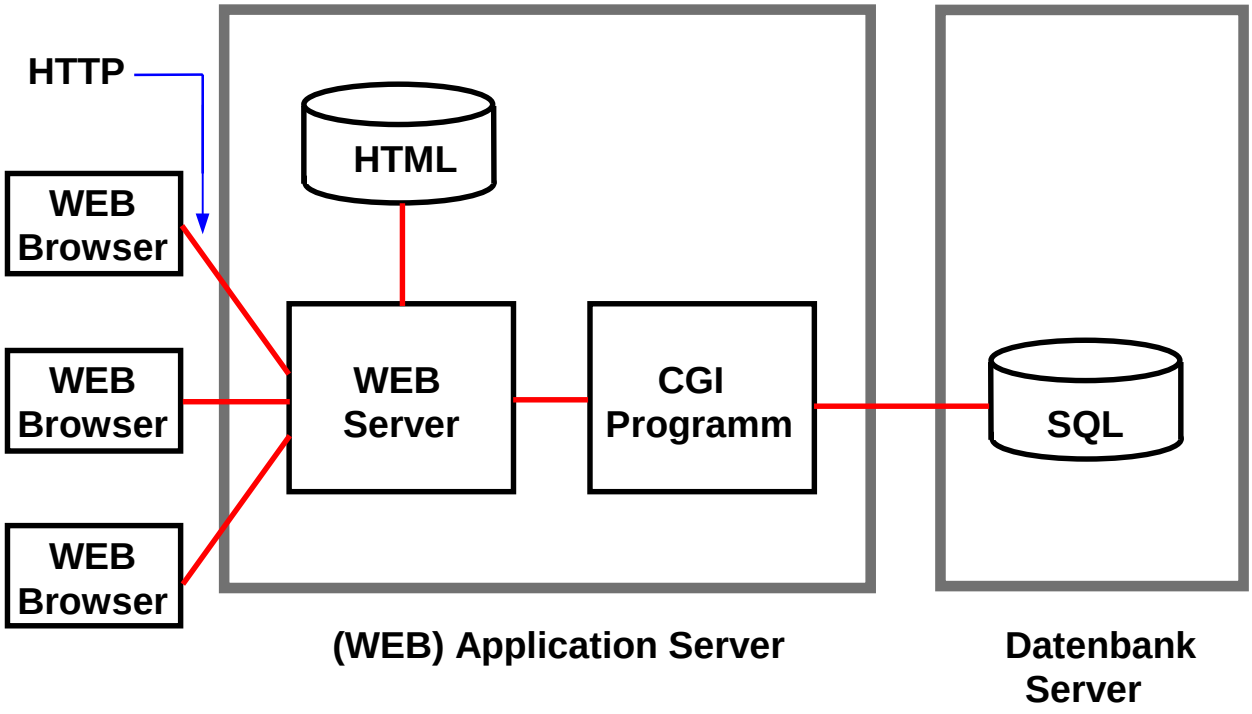

**GET /login.html HTTP/1.0**
**User-Agent: Mozilla/4.7 [en] (Win2000; I)**
**Accept: image/gif, image/jpeg, */***

**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**




# 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.**

-

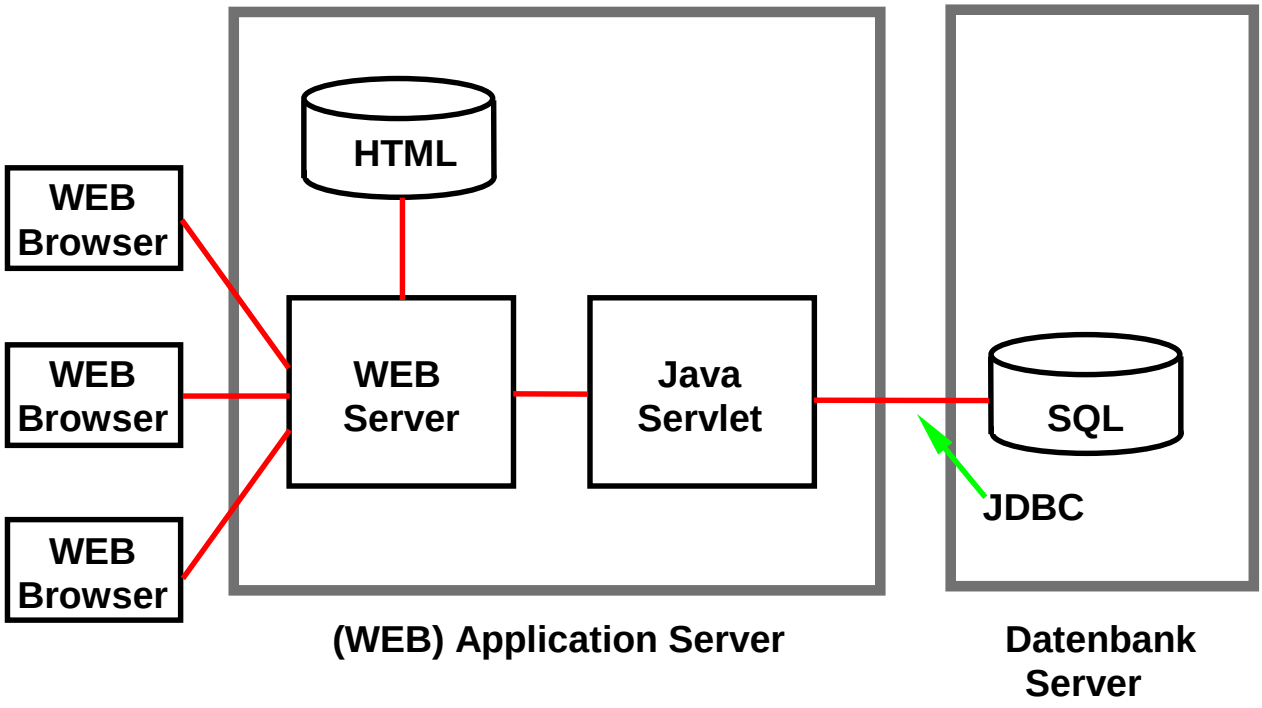
# Common Gateway Interface (CGI)

**Mit Hilfe einer URL greift ein Browser Klient auf einen Web Server zu. Der Web Server holt die angeforderte Web Seite aus seinem Plattenspeicher und sendet sie an den Klienten.**

**Im einfachsten Fall ist diese Seite *statisch,* d.h. ihr Inhalt ändert sich nie.**

**Wird ein CGI Programm durch eine URL aufgerufen, so generiert dieses *dynamische* Seiten in Echtzeit, z.B. indem ein Teil der wiedergegebenen Information aus einer SQL Datenbank abgefragt wird. Die Ausgabe geht in der Regel direkt an den Klienten.**

**CGI Programme können in allen auf dem Server verfügbaren Sprachen implementiert werden, z.B. C/C++, PERL, TCL, Unix Shell Scrip, Visual Basic, andere.**




# 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 Scriplets.**

# 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 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.**

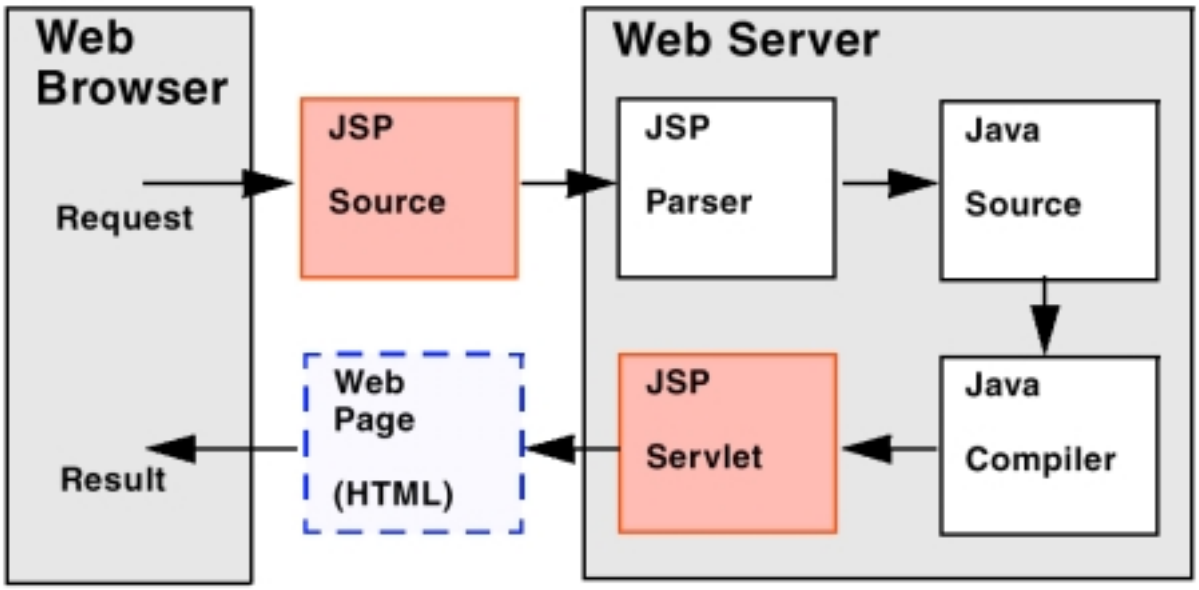
JSP files are similar in some ways to server-side includes in static HTML because both embed servlet functionality into the Web page. However, in a server-side include, a call to a servlet is embedded within a special servlet tag; in JSP, Java servlet code (or other Java code) is embedded directly into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

# Java Server Pages (JSP)

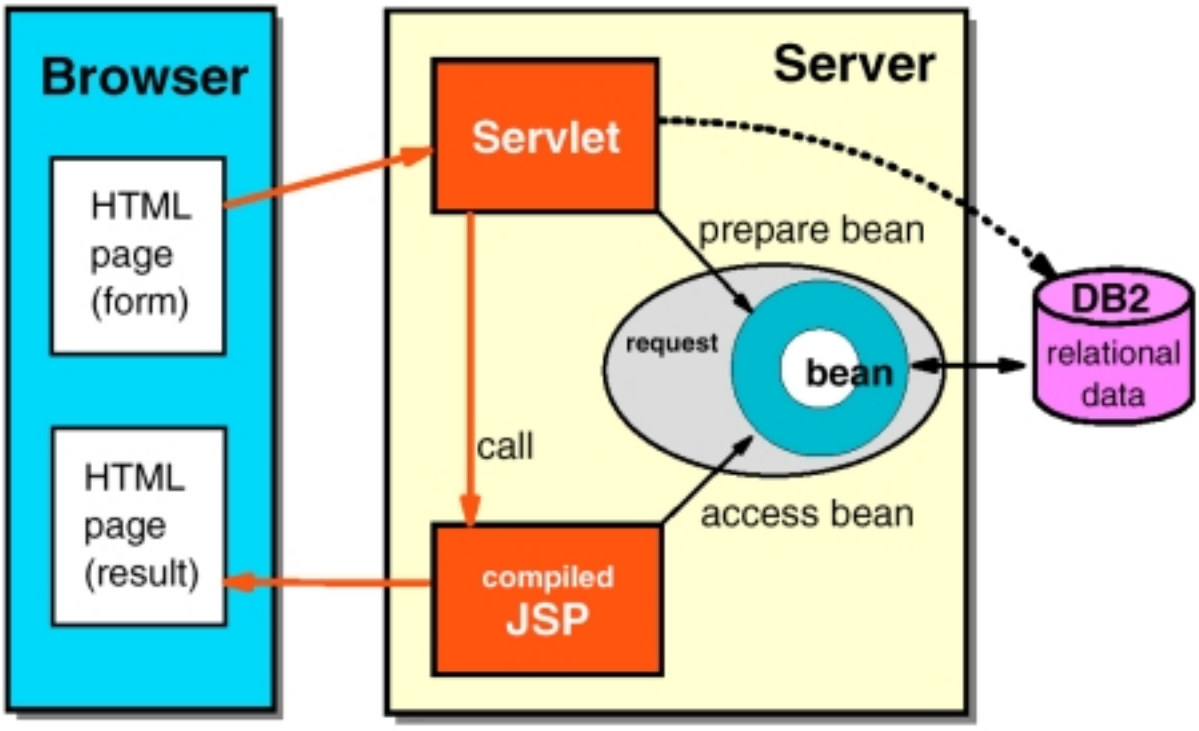
**Ein Servlet ist ein Java Programm, das Bildschirm Output in der Form einer HTML Datei produziert.**

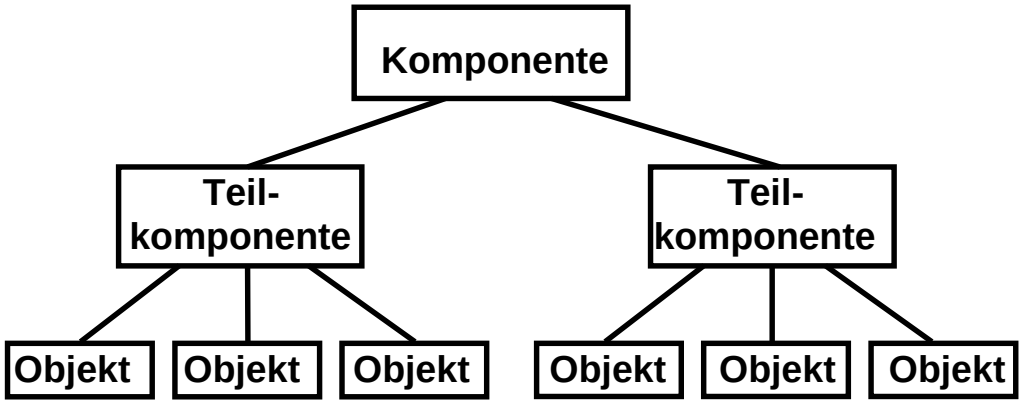
**Eine JAVAServerPage ist eine HTML Seite mit zusätzlichen JSP Tags.**

**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.**






# 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 Komponete 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.**

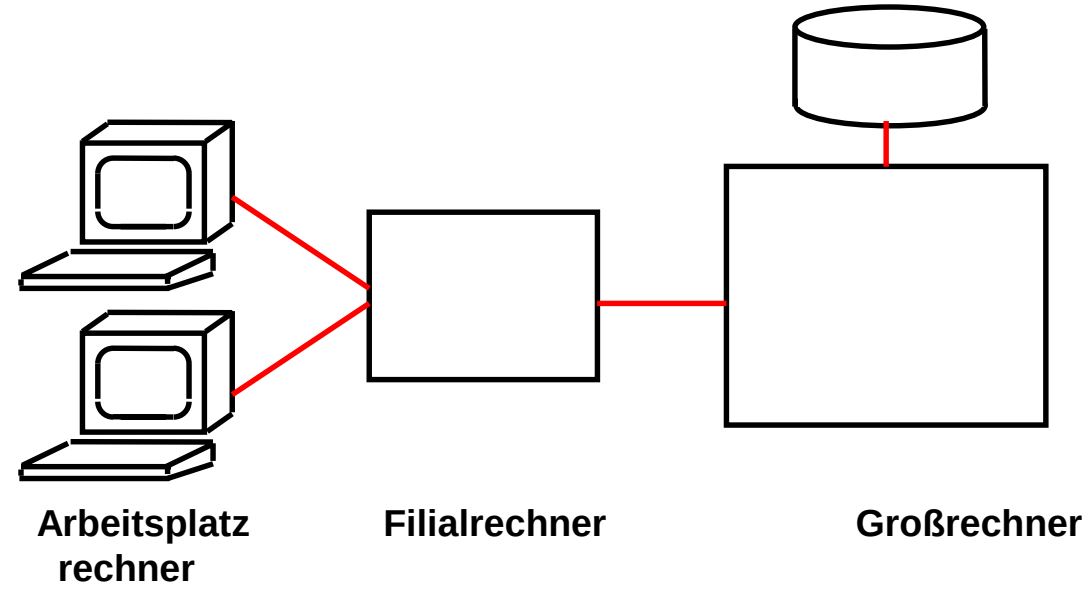
**Ein Business Object repräsentiert auf eine erkennbare, nachvollziehbare Art ein Objekt (eine Entity) des täglichen Lebens.**

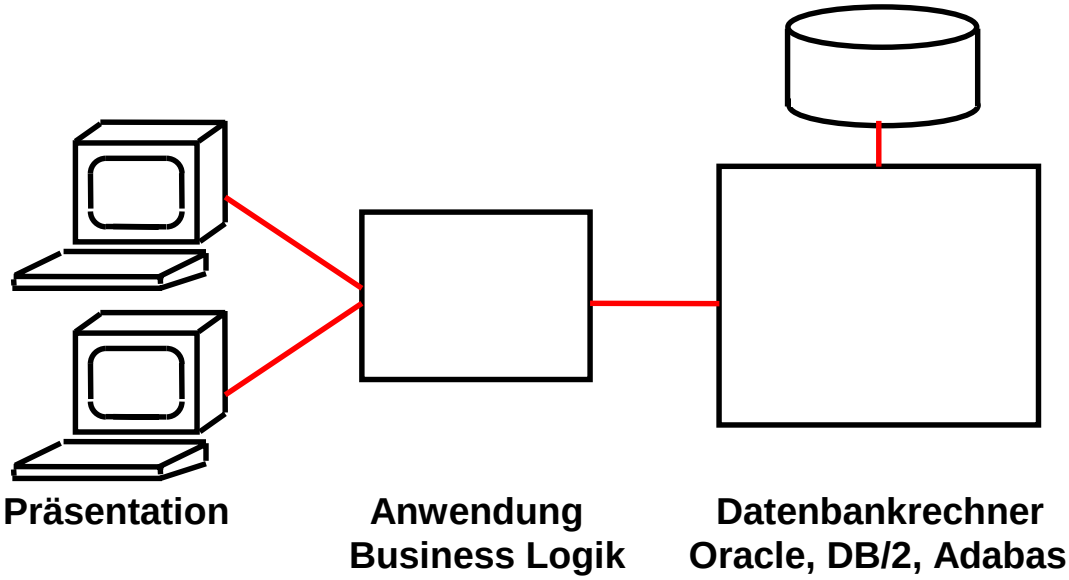
**Das Konzept eines Business Objektes steht dem betriebswirtschaftlich motivierten Anwendungskontext sehr viel näher als dem technisch motivierten Konstrukt eines programmiersprachlichen Objektes.**

**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.**

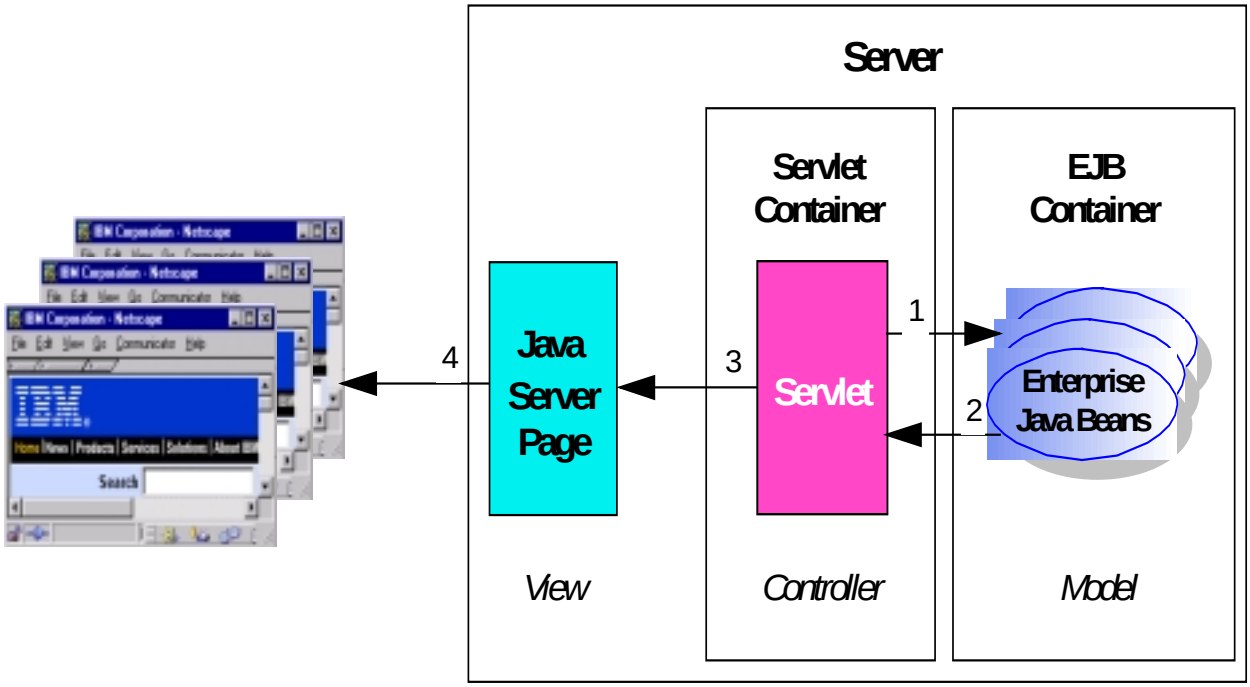
**Metadaten sind sich selbst beschreibende Information, welche die dynamische Struktur eines Systems definieren. Im Falle einer Datenbank beschreiben Metadaten die Datenbank Struktur und werden im „Repository“ verwaltet.**




# 3-Tier Architecture
# Dreistufige Architektur

**Es wird klar zwischen den Ebenen Präsentation, Anwendung und Datenbank unterschieden.**




# Model/View/Controller Triade (MVC)

**Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: „Design Patterns“. Addison Wesley Longman, USA, 1997**

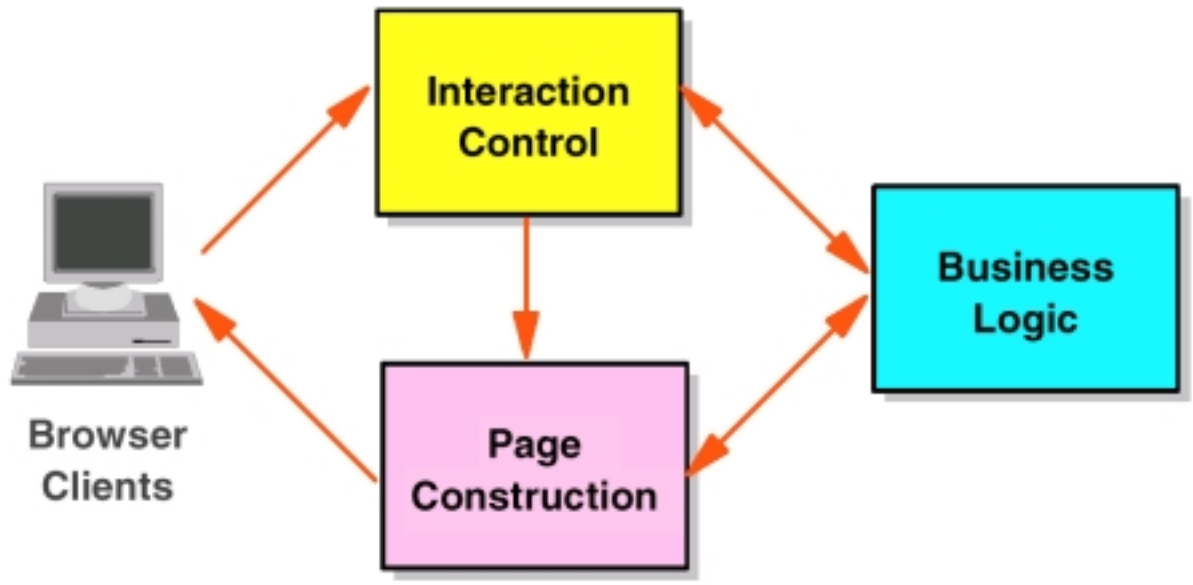
**Das „Modell“ (hier eine EJB) ist ein Anwendungsobjekt und kapselt die Business Logik. Der „View“ ist die Screen Darstellung dieses Objektes. Der „Controller“ definiert, wie die Benutzerschnittstelle auf Benutzereingaben reagiert.**

**MVC entkoppelt Modell und View zur Verbesserung von Flexibilität und Re-Use.**

**Der Entwickler der Seite 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.**

**Alternative: Die „View“ Funktion als Applet auslagern.**

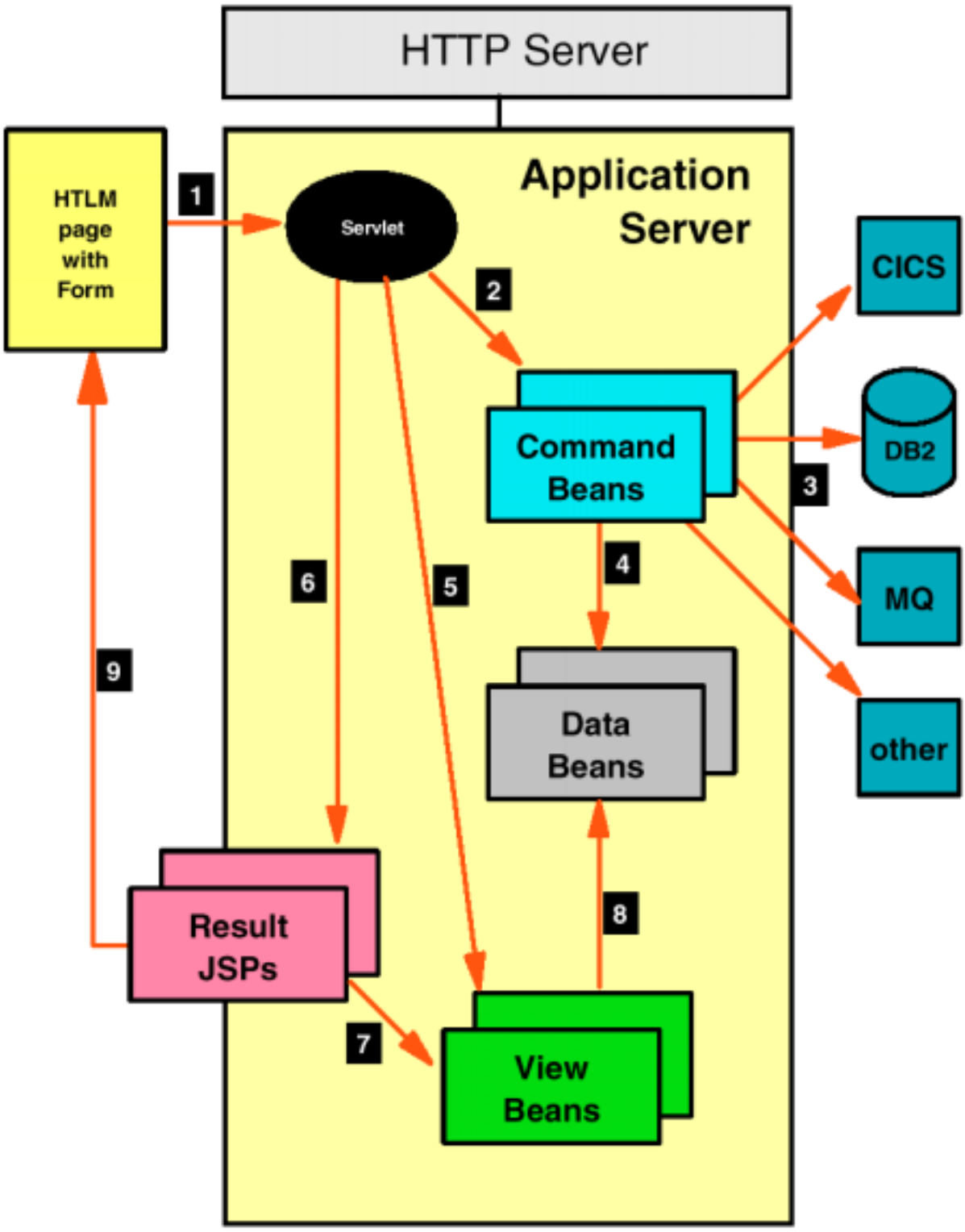



# Model - View - Controller (MVC) Ansatz

**Command- und Data Beans (plus eventuelle CICS oder MQSeries Programme, oder Stored Procedures) sind das „Modell“ (=Business Logik).**

**JSP´s und View Beans sind der „View“**

**Das Servlet ist der „Controller“**




**Architektur einer JSP Web Anwendung**

# 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)**

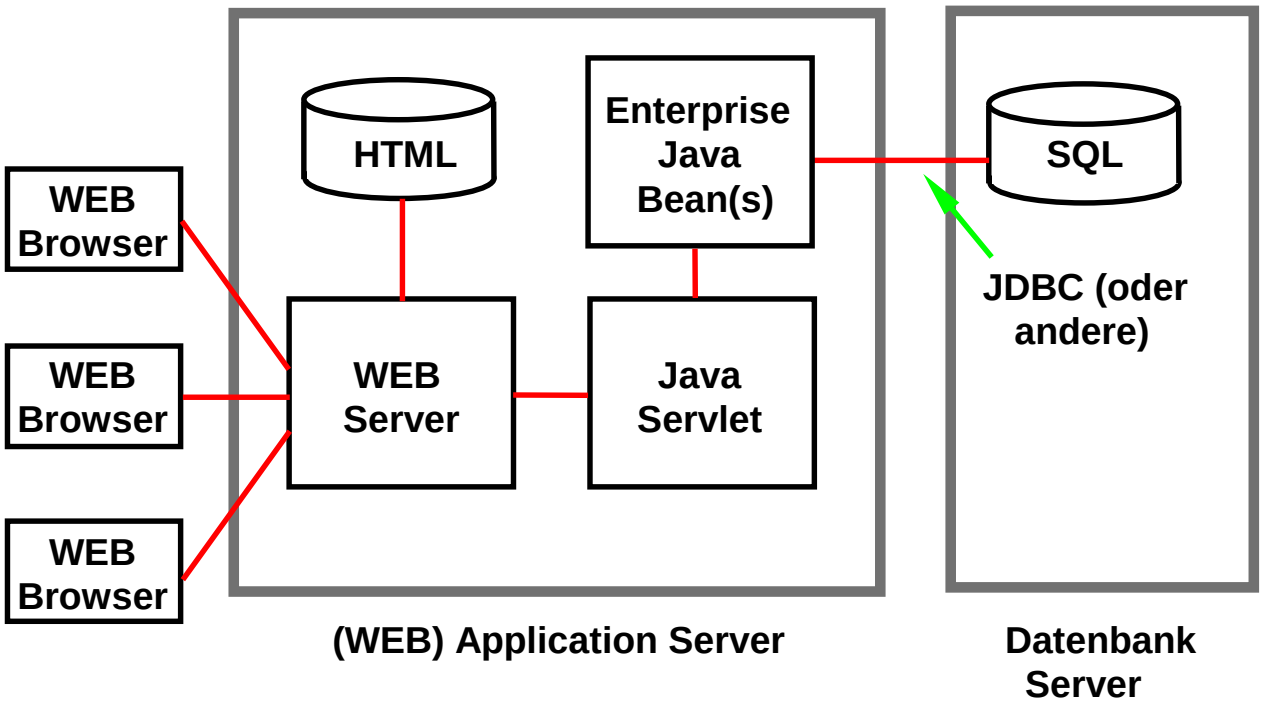
**Namens Konventionen**

**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.**




# 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 98**

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

**EJB Spezifikation legt fest:**

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

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

**Interoperabilität von EJBeans 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)**



# 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.**

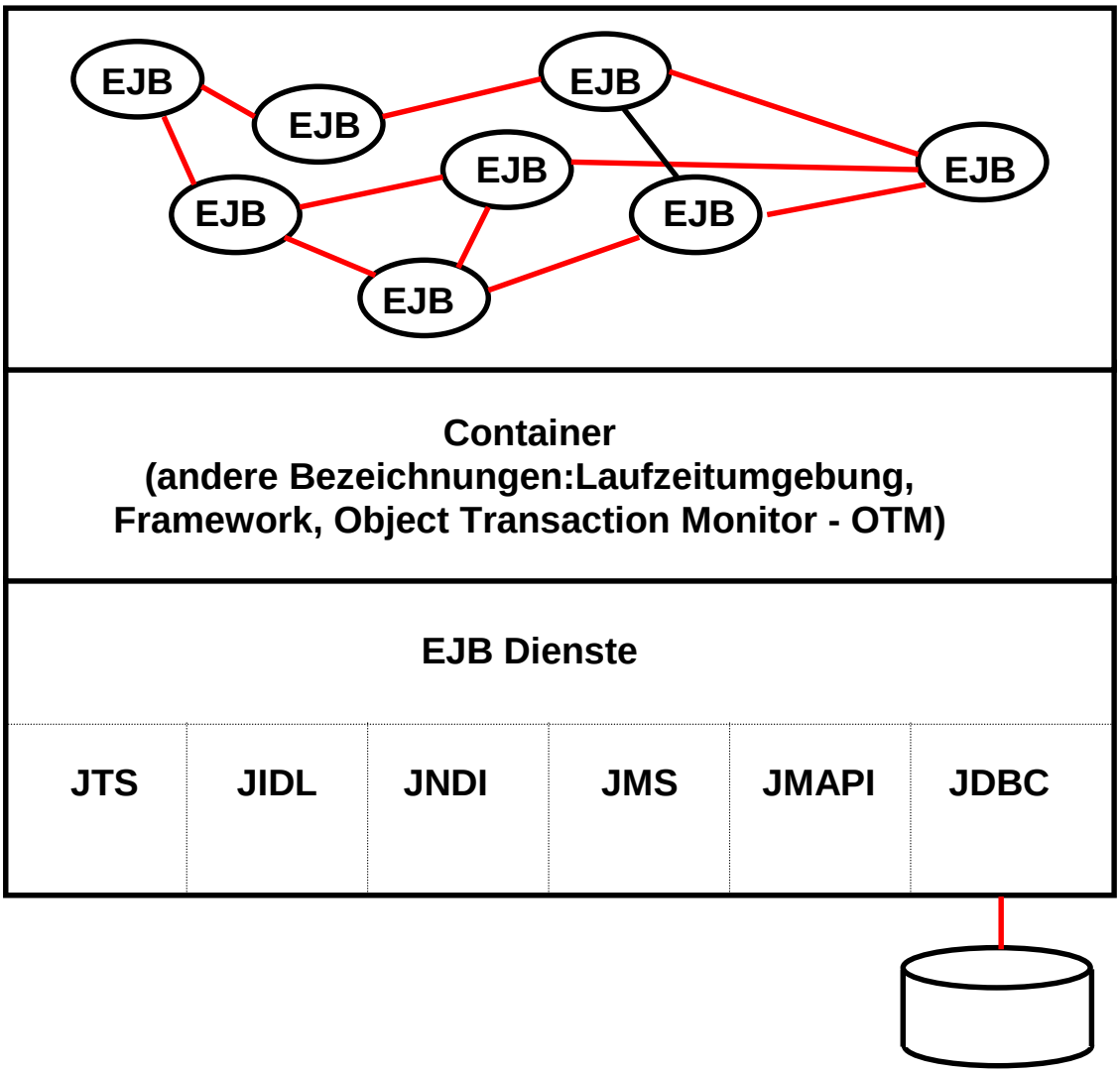
**Die permanente Speicherung eines Objektes auf einem Plattenspeicher wird als Persistenz bezeichnet.**

# Schlüsseleigenschaften 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, scalierbar 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**

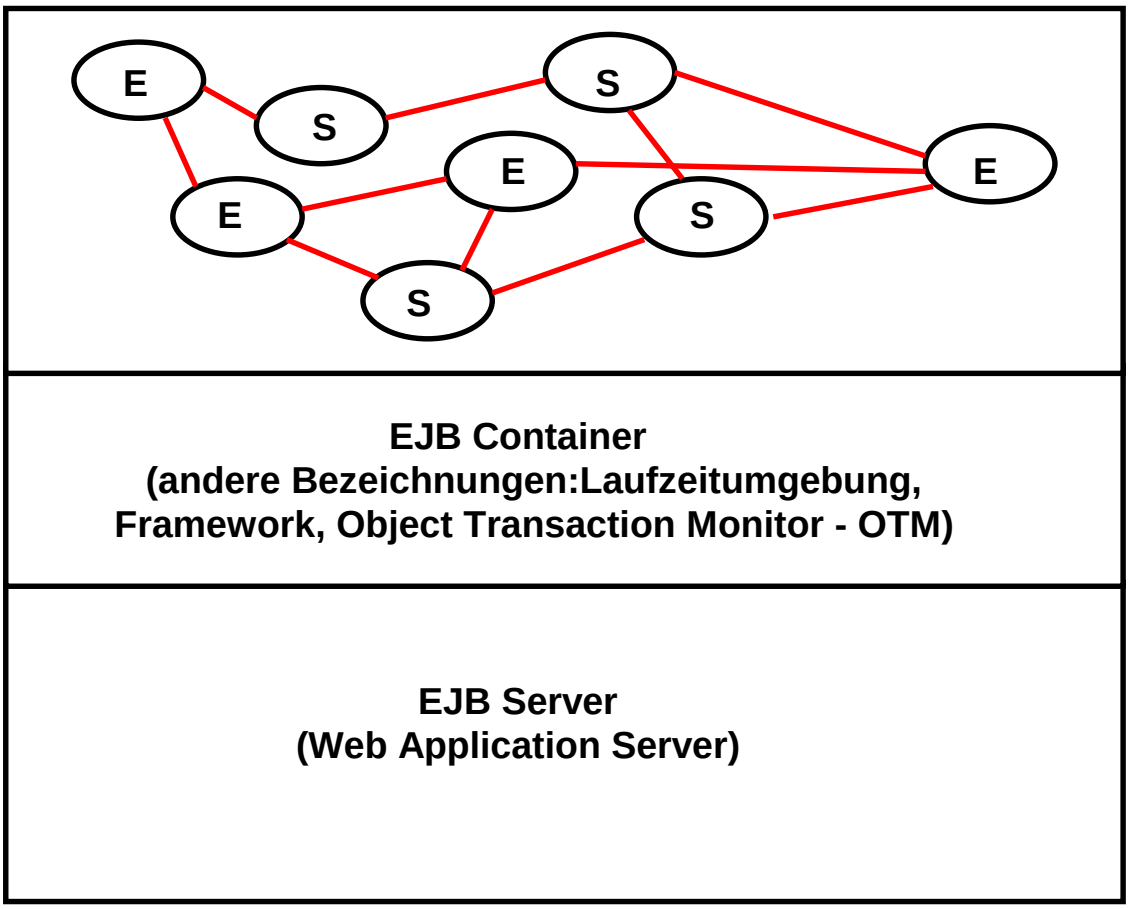
**Final Release der Version 1.1 der EJB Specifikation erfolgte im Dezember 1999**

# 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 Servics
- **JDBC** Java Data Base Connectivity
- **JMAPI** Java Management API
- **JIDL** Java interface definition language




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

# Arten von EJBs

## Entity Beans

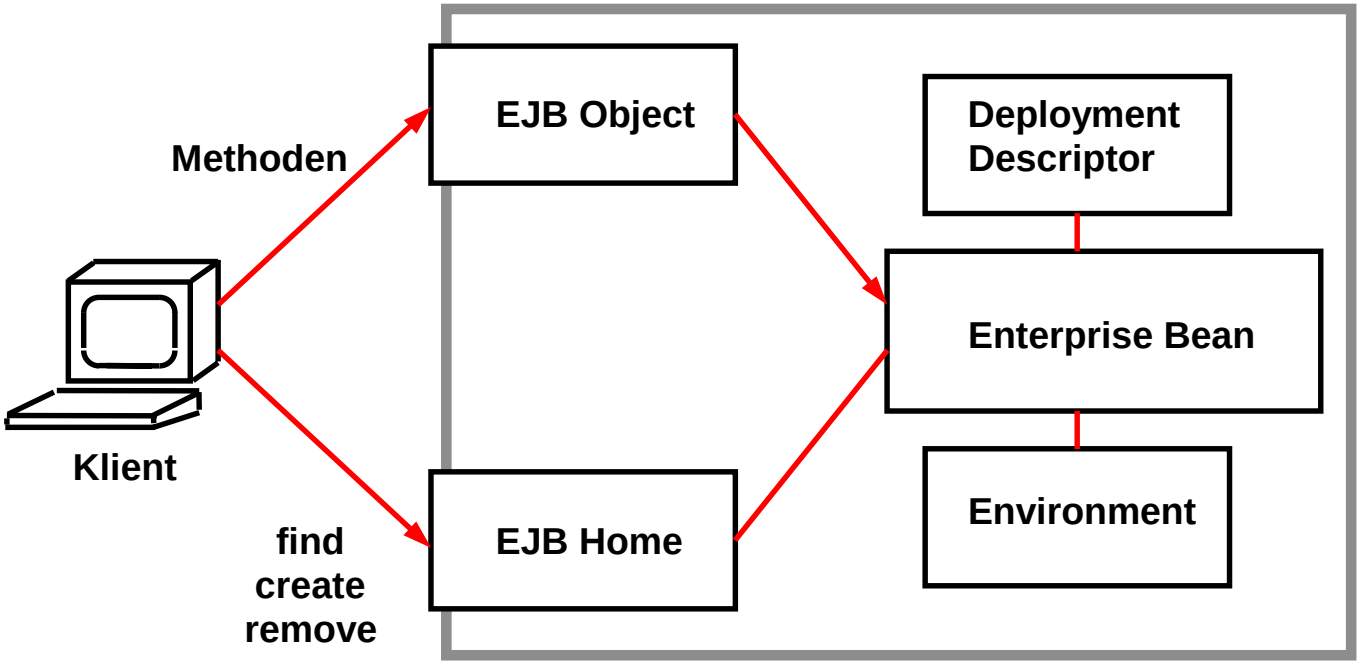
- repräsentieren spezifische Daten (z.B. eine Reihe in einer SQL Datenbank)
- Methoden ermöglichen die Manipulation der Daten, welche die Bean repräsentiert
- überleben, so lange die Daten in der Datenbank überleben.

## 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.

# 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**

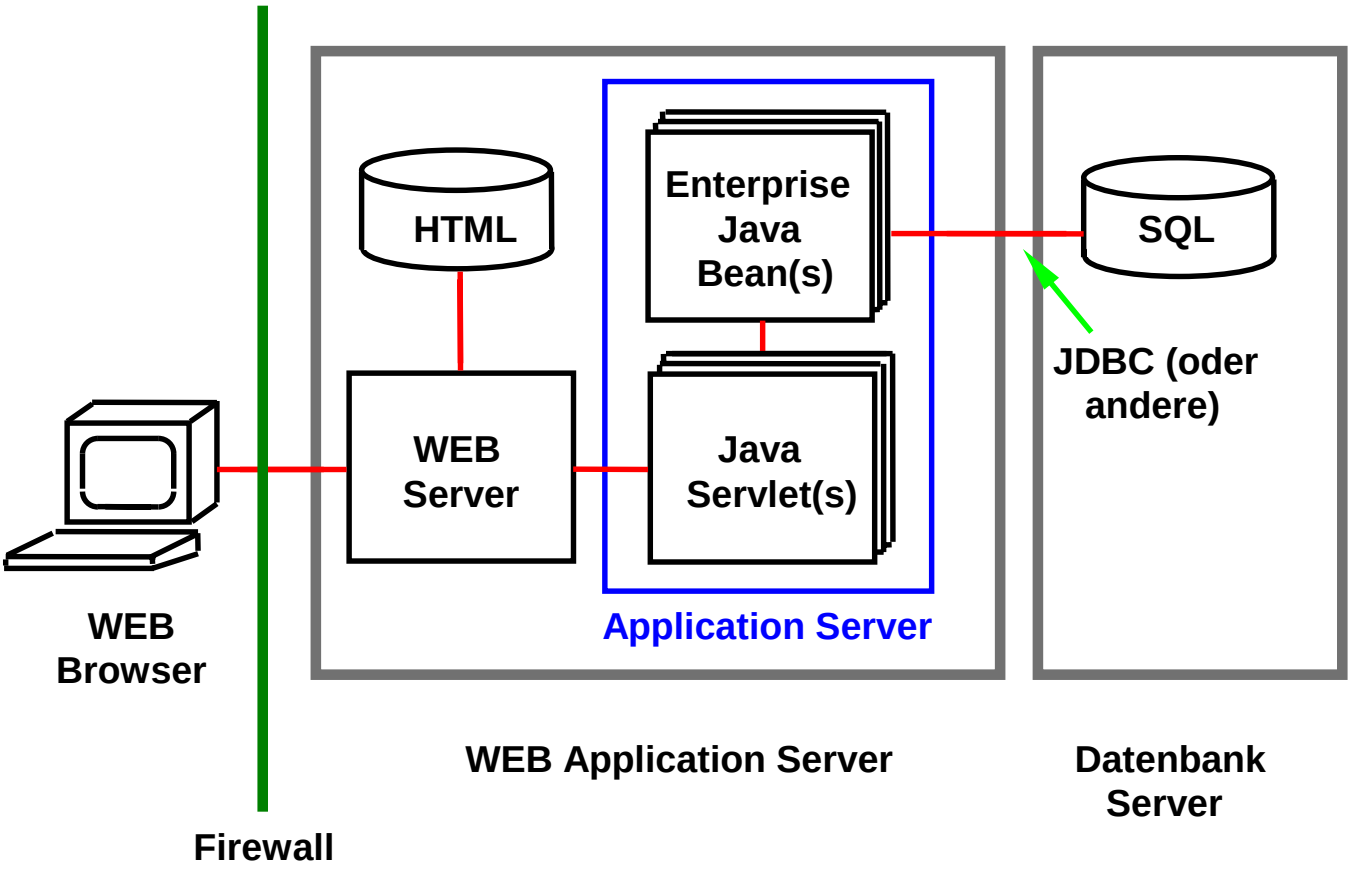
**Parameter können statisch zur Assembly Zeit oder dynamisch zur Laufzeit festgelegt werden**

**Beispiele für Parameter:**

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

**typischerweise durch den EJB Entwickler angelegt**

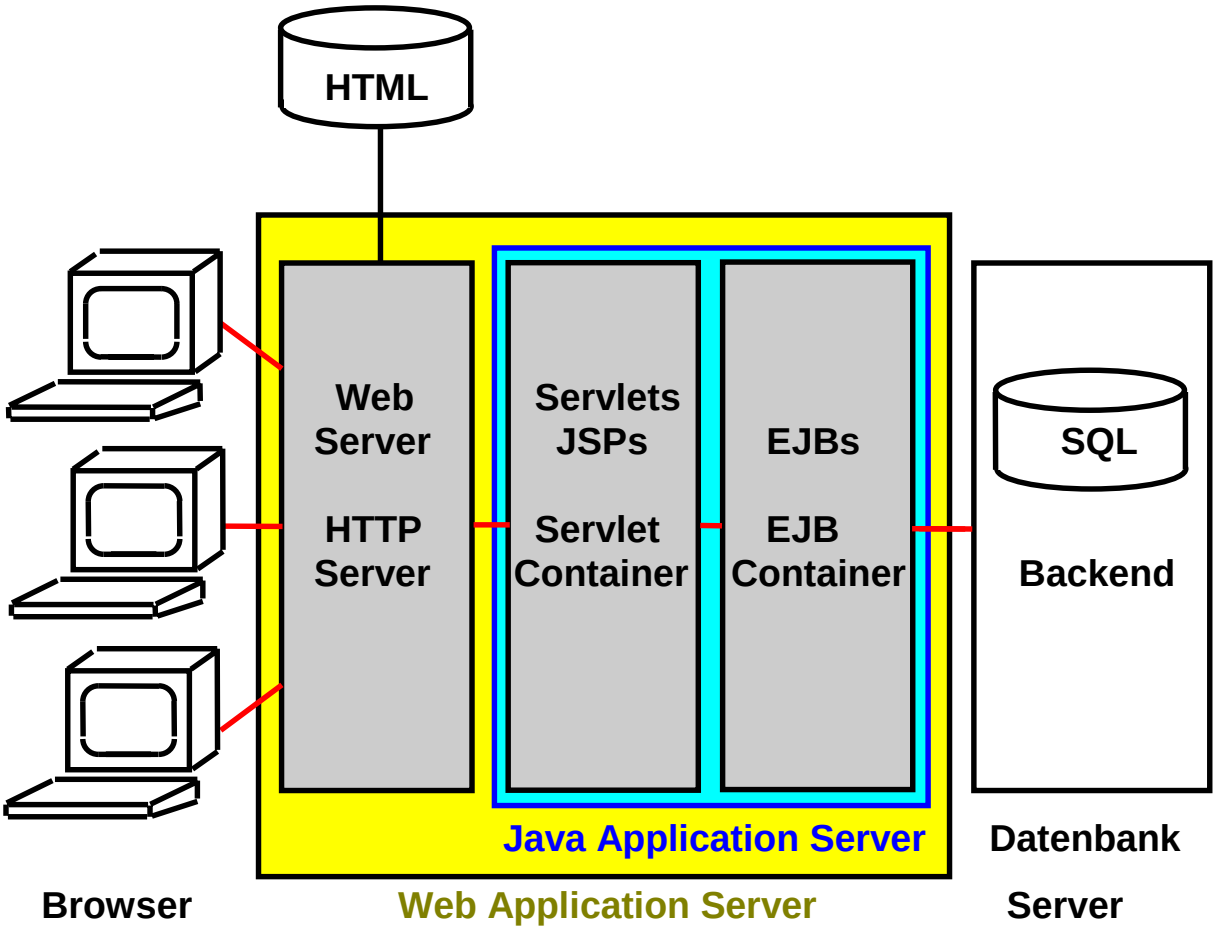
**kann durch einen Administrator mit Hilfe eines Tools abgeändert werden**

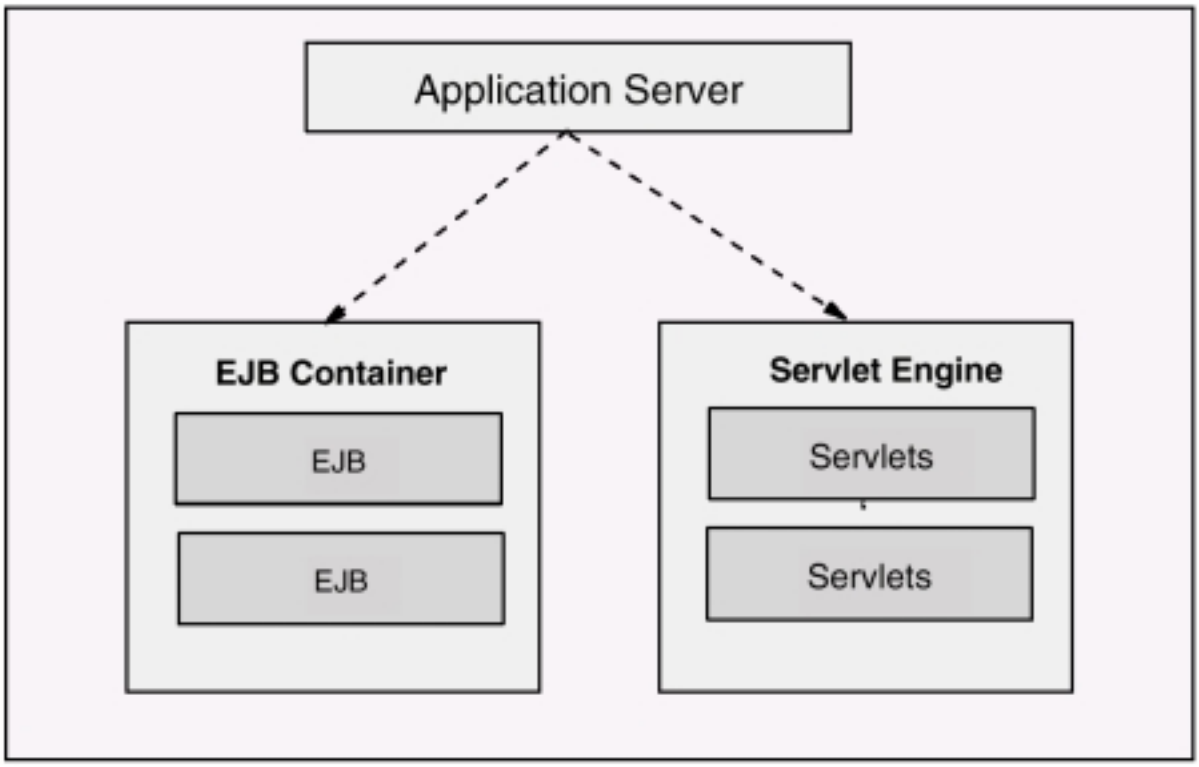



# Web Application Server

**Hersteller**

- **BEA Weblogic**
- **Brokat Twister**
- **IBM WebSphere**
- **Inprise**
- **Iona Orbix**
- **Silverstream**
- **Sun iPlanet**

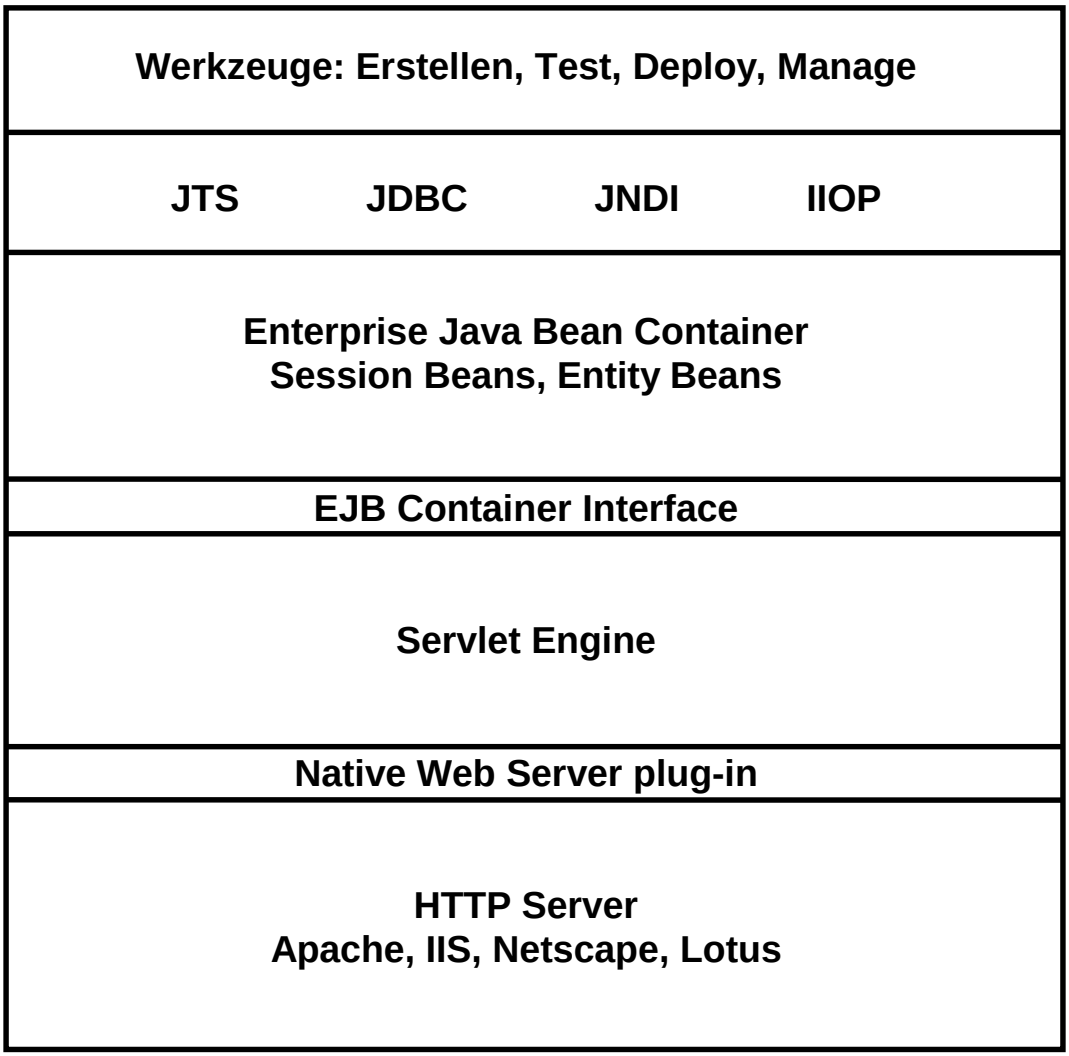

HTML
Web
Server
HTTP
Server
Servlets
JSPs
Servlet
Container
EJBs
EJB
Container
Java Application Server
SQL
Backend
Browser
Web Application Server
Datenbank
Server




# Application Server hierarchy

**Server is the process that is used to run your servlet and EJB applications, providing you both the servlet runtime component (servlet engine) and EJB runtime component (EJB container). You can define multiple application servers, each of which has its own Java Virtual Machine (JVM).**

**The Application Server is organized as a EJB server, servlet engine and its corresponding Web applications. EJB server contains the deployed EJBs where servlet engine in turn contains its Web applications.**




# Struktur eines Web Application Servers

**Aufgaben:**

**Der Web Server betreut statische HTML Seiten, CGI und proprietäre Plug-ins**

**Die Servlet Engine behandeln Java Servlet Anforderungen und dynamische Seitenanforderungen (JSP)**

**Die Servlets behandeln HTTP Anforderungen, verwaltet HTTP Sessions mit dem Klienten, erzeugt Presentation Logic via HTML, bewältigt nicht-transaktionale (Business) Anwendungen**

**Session und Entity Beans bewältigen (Business Logic) Anwendungen mit transaktionaler Integrität**

# Skalierbarkeit

**Leistungsverhalten eines Windows NT Web Servers**

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

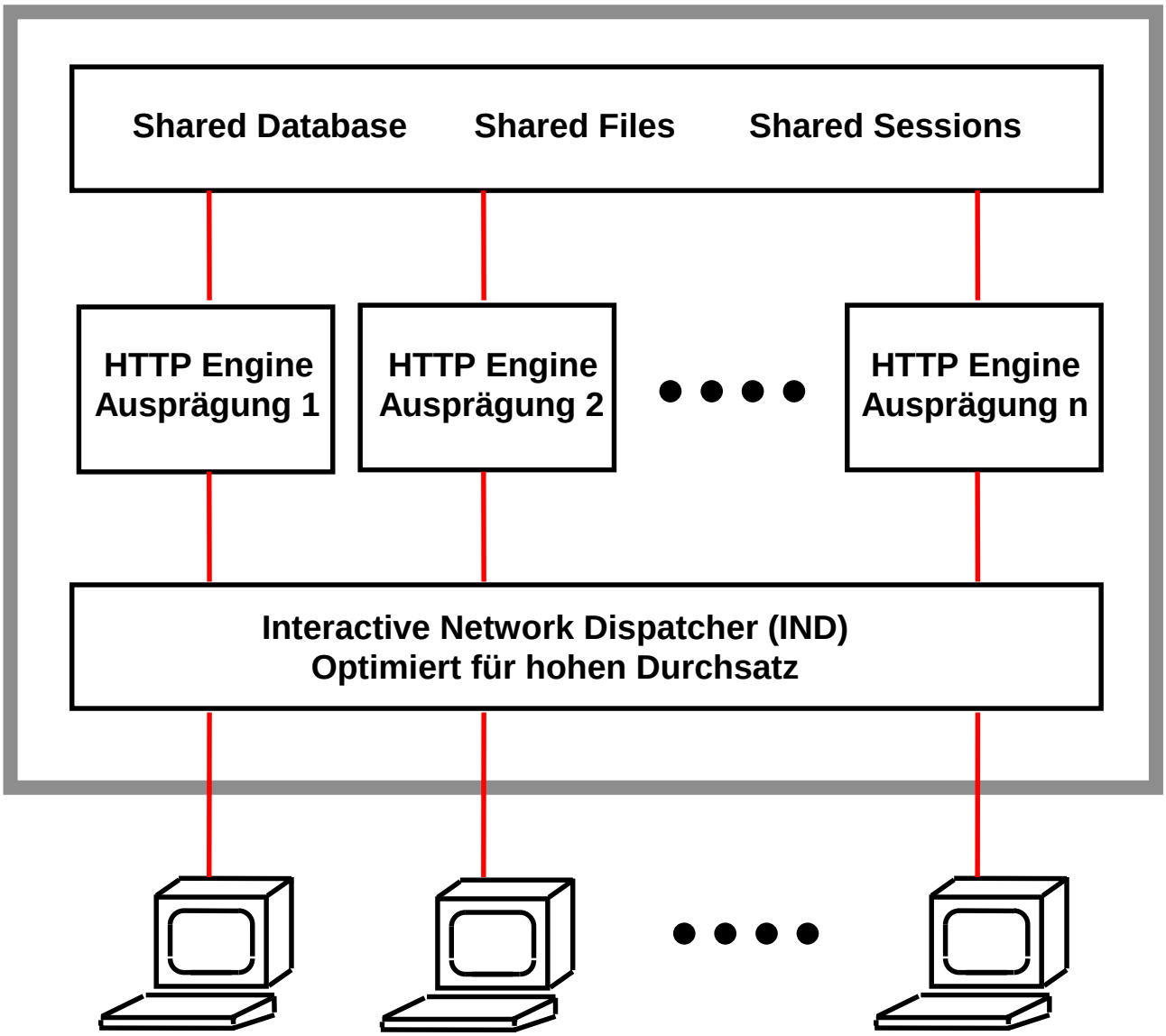
**Anforderungen eines größeren Unternehmens in 1999**

- **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**



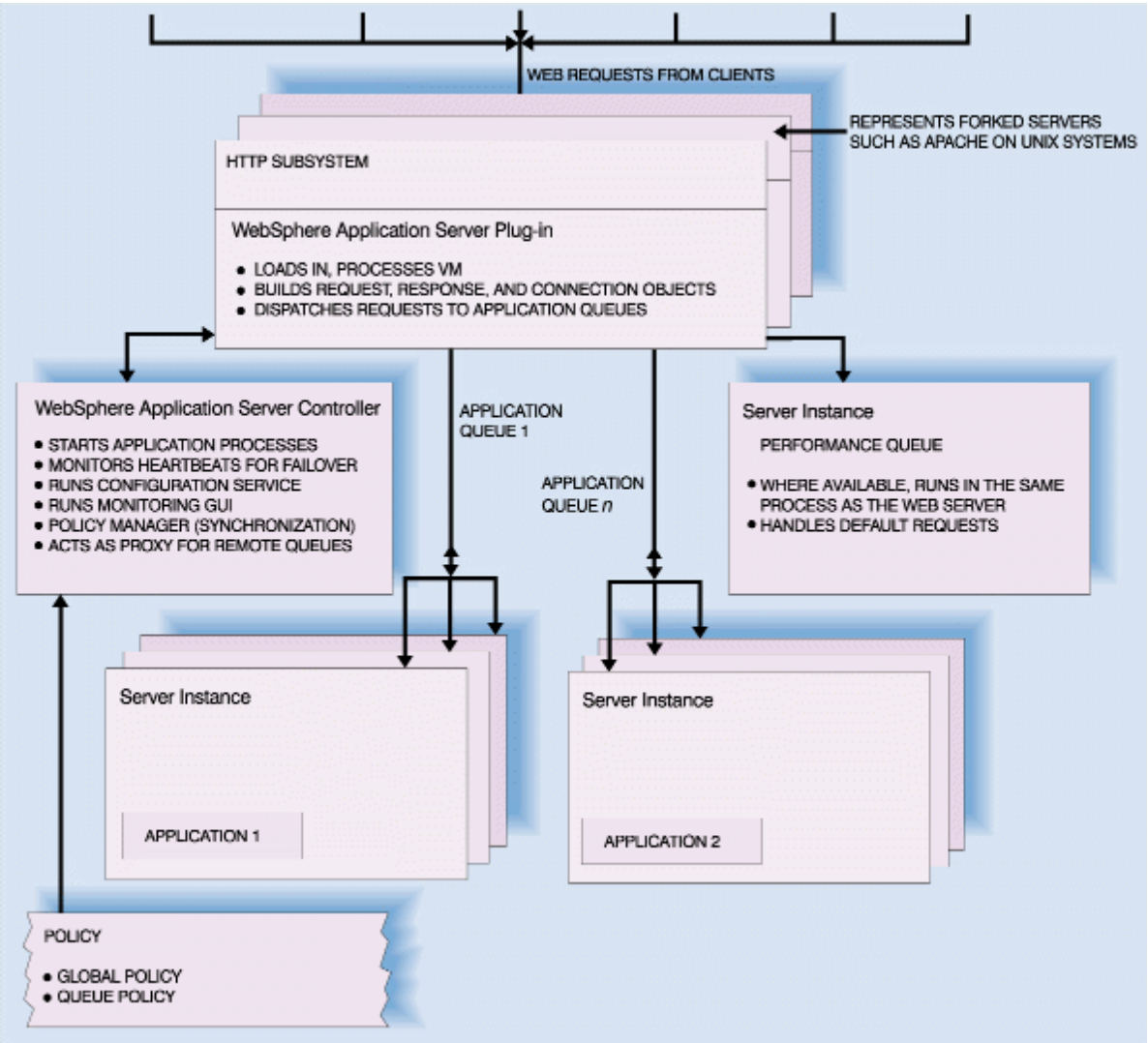

# HTTP Server

**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**

**E. Bayeh : "The WebSphere Application Server architecture and programming model". IBM Systems Journal Vol 37, No. 3, IBM Form No. G321-5680**



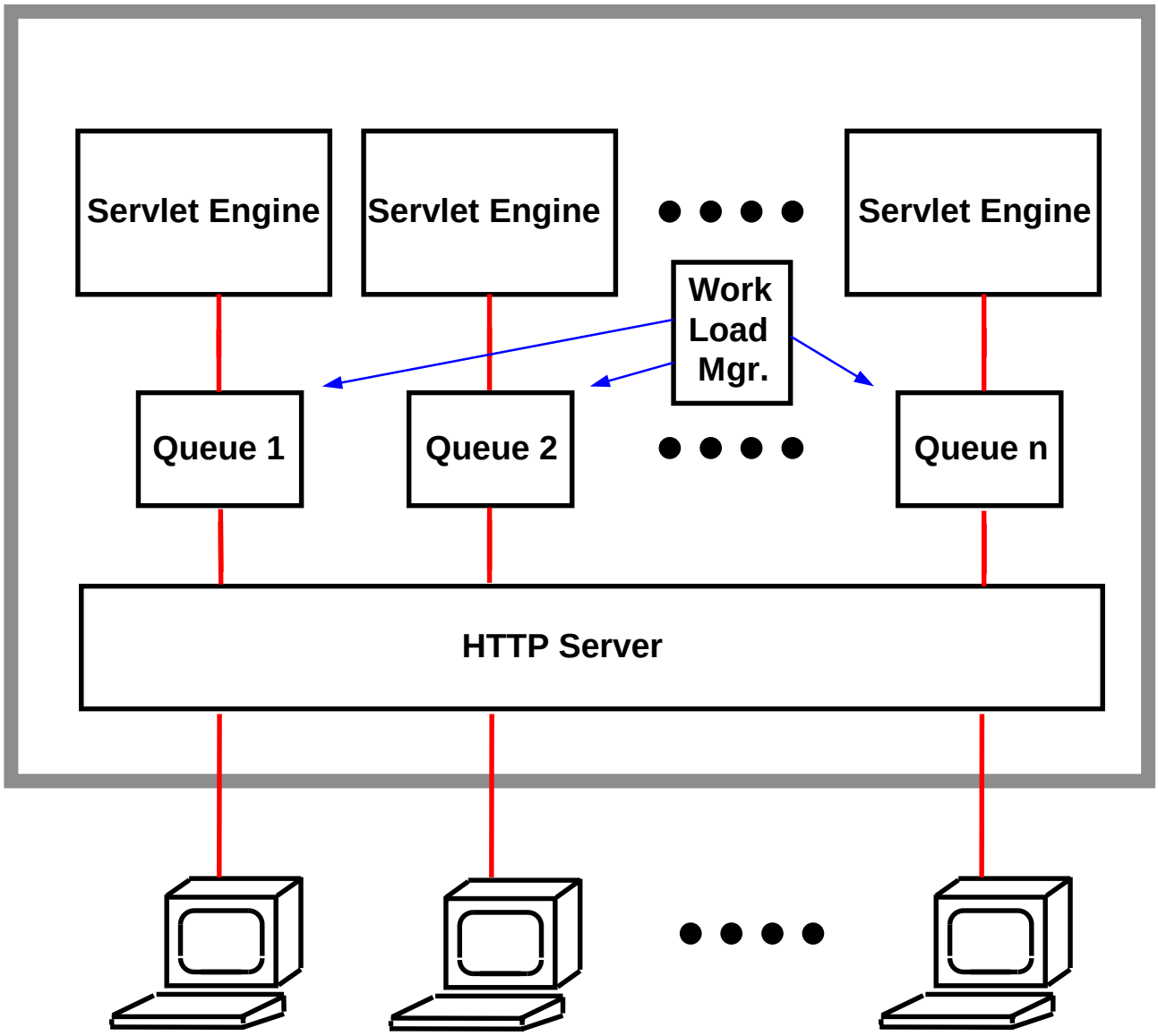

# Anwendungs-Queues und mehrfache Prozesse

**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**




# 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 OS/390

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