UNIVERSITÄT LEIPZIG

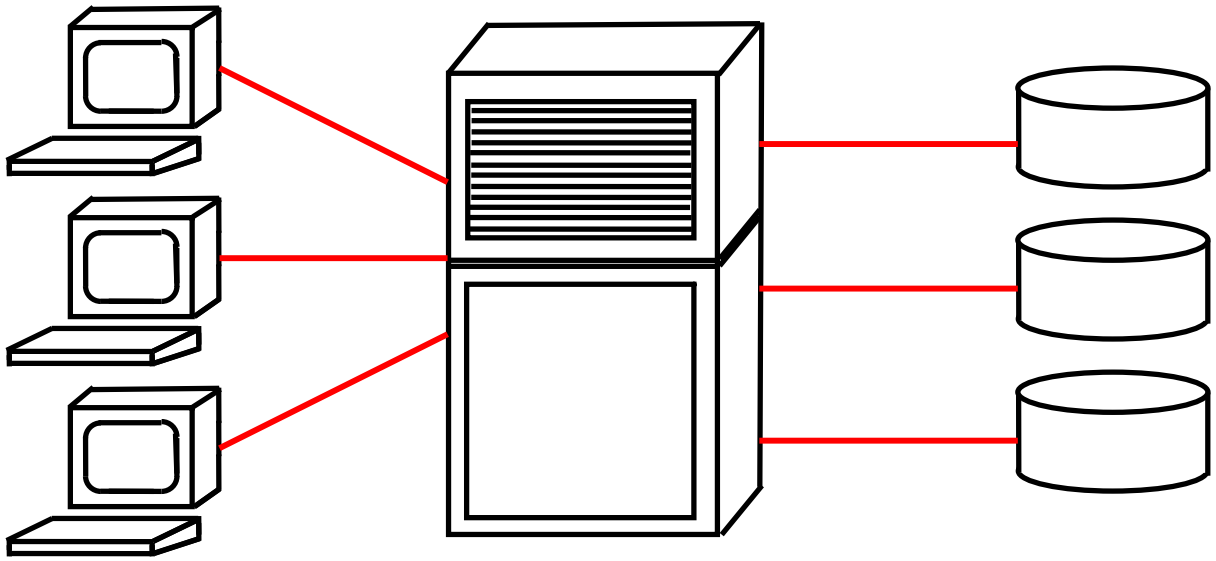
Mainframe Internet Integration

Prof. Dr. Martin Bogdan
Prof. Dr.-Ing. Wilhelm G. Spruth

SS2013

Java Enterprise Edition Teil 2

Servlets

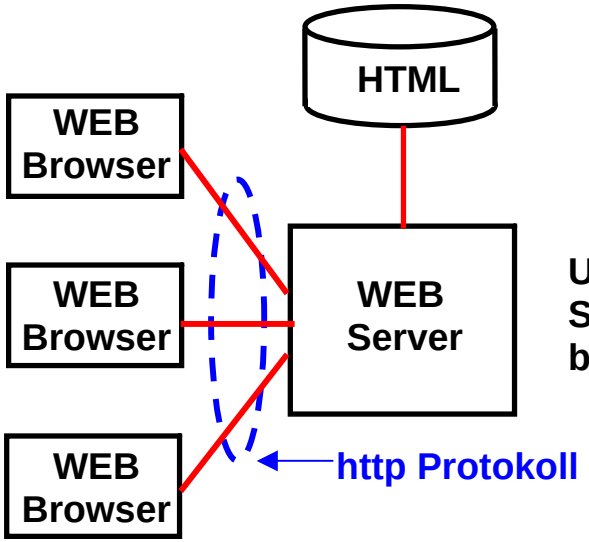


Browser **http Server z.B. Apache** **statische HTML Seiten**

Statische HTML Pages

Statische HTML Seiten sind der einfachste Fall einer Client/Server Anwendung. Der Klient (Browser) ruft eine (statische) HTML Seite auf, die unmodifiziert an den Klienten zurückgespielt wird. Die aus den statischen HTML Seiten bestehende Datensammlung wird lediglich offline geändert (z.B. durch das Hochladen mittels FTP). Der sehr große Vorteil: Es gibt keinerlei Datenintegritätsprobleme.

Es wird angenommen, dass 90% aller Web Zugriffe auf statische HTML Seiten erfolgen.



Unter z/OS wird der Web Server als HTTP Server bezeichnet,

http Protokoll

HTML Seiten werden zum/vom 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.

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

HTTP ist eines von vielen Schicht 5 Übertragungsprotokollen. Beispiele für Schicht 5 Protokolle sind:

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

Serving static Web pages unter z/OS

Ein Web-Server unter z/OS, wie der HTTP Server unter Unix System Services, arbeitet ähnlich wie ein Web-Server auf anderen Plattformen. Der Benutzer sendet eine HTTP-Anfrage an den z/OS HTTP Server, um eine bestimmte Datei zu erhalten. Der HTTP Server ruft die Datei von seinem Datei-Repository ab, und schickt sie an den Benutzer, zusammen mit Informationen über die Datei (z. B. MIME-Typ und Größe) in dem HTTP-Header.

Der z/OS HTTP Server unterscheidet sich von anderen Web-Servern. Da z/OS Dateien im EBCDIC Format codiert sind, müssen z/OS Dokumente zunächst in das ASCII-Format konvertiert werden, ehe sie über das Internet an einen Endbenutzer geschickt werden. Binäre Dokumente und Bilder müssen nicht konvertiert werden.

Der z/OS HTTP Server führt diese Konvertierungen durch und erspart damit dem Programmierer diesen Schritt.

Für das Hochladen von Dokumenten verwendet der Programmierer normalerweise FTP. Dabei gibt der Programmierer ASCII als FTP-Transport-Format an. Der z/OS http Server konvertiert das Format dann automatisch in EBCDIC. Wenn „binary“ als FTP-Transport-Format angegeben wird. wird die Datei nicht konvertiert.

Der zLinux HTTP-Server hat diese Probleme nicht, da zLinux alle Daten im ASCII-Format verarbeitet und in der Regel kein EBCDIC benutzt.

Beim Transfer von Daten zwischen zwei Rechnern kann der Java Programmierer das Encoding der Daten spezifizieren.

Es wird behauptet, dass mehr als 50% aller betriebswirtschaftlich relevanter Daten, auf die über das WWW zugegriffen wird, im EBCDIC Format auf Mainframes gespeichert sind.

HTML Forms

Beim Aufruf von statischen HTML Seiten sendet der Browser lediglich eine URL an den Web Server. Für die Nutzung des Browsers und WWW als Client Server System ist es erforderlich, weitere Daten vom Klienten an den Server zu senden. Dies geschieht mittels „Form Data“. Weiterhin muss der Server eine Anfrage beantworten können. Bei der Nutzung von Java geschieht dies in der Regel mit Hilfe von Servlets und Java Server Pages (JSP).

HTML Forms sind ein einfache Werkzeuge, mit dem ein Benutzer Daten mit Hilfe des HTTP-Protokolls an einen Server schicken kann. HTML Forms erlauben es dem Browser, „Form Data“ an den Server zu senden,

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.

Der FORM Tag spezifiziert:

- **Die zu benutzende HTTP-Methode. In den meisten Fällen 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".**

Form Data

Wenn Sie jemals eine Web-Suchmaschine verwendet haben, einen on-line Buchladen besuchten, Aktienkurse on-line verfolgten oder ein Quote für den Preis eines Flugtickets abfragten, haben Sie wahrscheinlich eine merkwürdig aussehenden URLs gesehen, wie

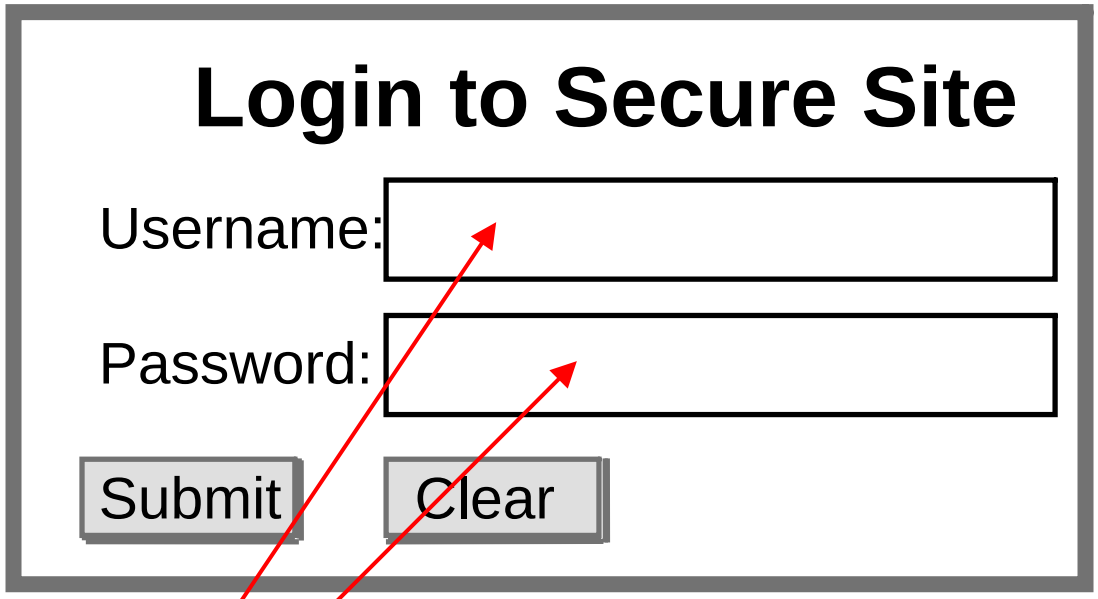
http://host/path?user=Marty&origin=bwi&dest=lax ← **Dies ist ein Beispiel für eine GET Anfrage**

Dieser Teil wird als „Form Data“ bezeichnet und ist ein häufig benutztes Verfahren, um Daten von einer gesendeten Webseite an ein Server-seitiges Programm zu übergeben.

Daten zusätzlich zu einer URL sind Form Data. Sie werden in der Regel von einem Browser an einen Webserver in einem von zwei Formaten übertragen.

- **Für GET-Anfragen werden die Form Daten an das Ende der URL nach einem Fragezeichen angebracht, siehe obiges Beispiel.**
- **Für POST-Anfragen werden die Form Daten an den Server in die http Nachricht eingebettet.**

Für alle außer sehr einfachen Anfragen wird die POST-Methode verwendet.



HTML Form

Die hier dargestellte Wiedergabe auf einer HTML Seite erwartet eine Eingabe seitens des Benutzers in den hiefür vorgesehenen Feldern.

Diese Darstellung wird mit Hilfe von Form Tags innerhalb des HTML Codes programmiert.

Folgend ist eine HTML Seite dargestellt, die vom Server an den Browser gesendet wurde. Nachdem der Benutzer die beiden Felder ausgefüllt hat und den „Submit“ Button aktiviert, wird der Inhalt der beiden Felder, zusammen mit der Action, als Form Data an den Server gesendet.

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

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

Der FORM Tag spezifiziert:

- **Die zu benutzende HTTP-Methode. Hier ist dies POST; die Daten werden innerhalb des Bodys der http 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".**

Ein weiteres Beispiel finden Sie unter http://www.informatik.uni-leipzig.de/cs/support/SampleForm.pdf

HTML Forms

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.

Wenn der Server die Nachricht mit den Form Date empfängt, wird das Programm xyz.servlet im Verzeichnis abc.de/servlet aufgerufen, wobei die beiden Variablen username und password übergeben werden.

Multipurpose Internet Mail Extensions (MIME) ist ein Standard, der die Struktur und den Aufbau von E-Mails und anderen Internetnachrichten festlegt.

Servlets

Ein Applet Tag , z.B.

```
<APPLET CODE="HelloApplet.class"></APPLET>
```

in einer HTML-Seite bewirkt das Herunterladen des Applet Byte Codes vom Server auf den Klienten, wo der Code ausgeführt wird.

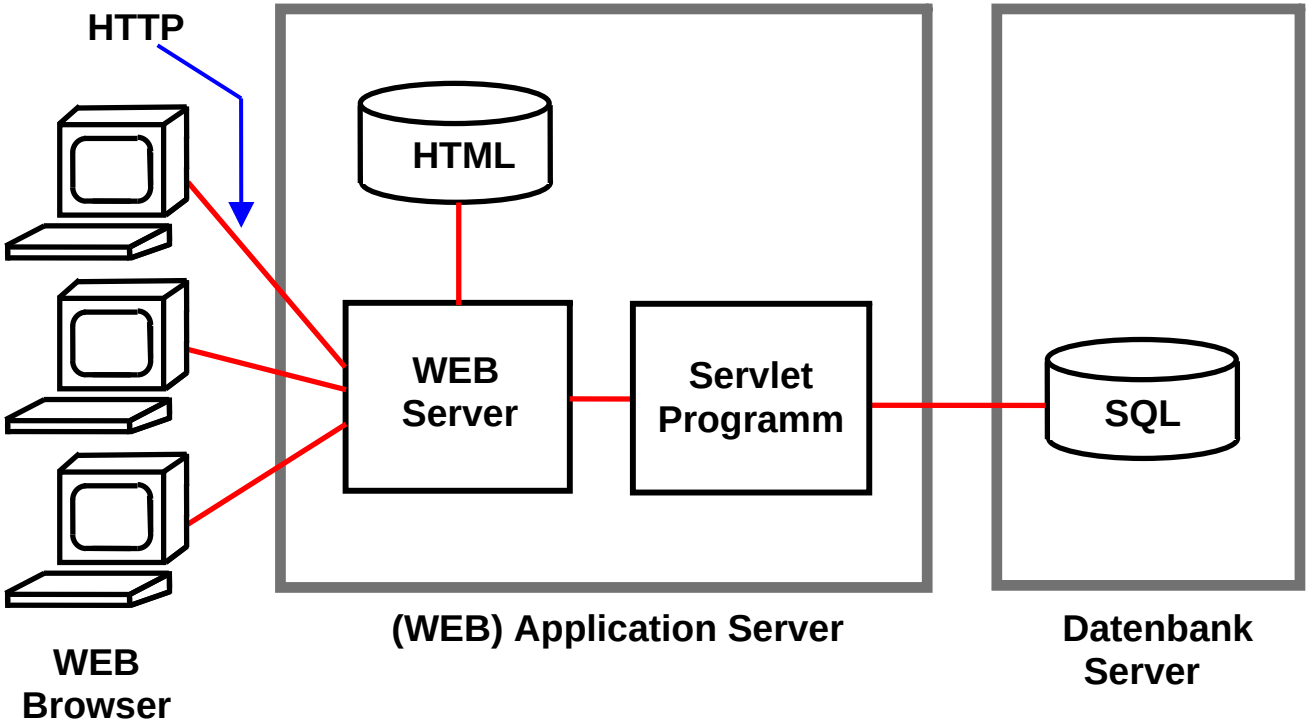
Ein Servlet Tag , z.B.

```
<SERVLET CODE="HelloServlet"></SERVLET>
```

in einer HTML-Seite bewirkt das Ausführen eines Servlets auf dem Server.

Java Servlets sind normale Java-Klassen, die auf einem Server innerhalb einer standardisierten Laufzeit-Umgebung, der "Servlet Engine" oder des "Servlet Containers", ablaufen. Der Servlet Container beinhaltet eine normale Java Virtuelle Maschine.

- **Servlets sind vollwertige Java-Programme; sie verfügen über alle Java APIs, einschließlich JDBC (Java Data Base Connectivity) und SQLJ. Ein Applet kann im Gegensatz dazu auf keine Server-seitigen Daten zugreifen.**
- **Im Gegensatz zu CGI erfordert das Java Servlet nur "Light Weight Context Switches" implementiert über Java Threads. Daraus resultiert ein deutlich besseres Leistungsverhalten.**
- **Da das Servlet im Hauptspeicher verbleibt, können Verbindungen (Connections) zur Datenbank offen gehalten werden. Ein Servlet kann einen gemeinsamen Vorrat an Datenbankverbindungen verwalten, und diesen je nach Bedarf einzelnen Concurrent-Benutzern zuordnen.**
- **Leistungsfähiges Fehler- und Type-Checking.**



Dynamischer WEB Seiten Inhalt

Damit der Server mit den übertragenen Form Data etwas anfangen kann, enthält die übertragene Nachricht die Angabe eines Programmnamens, der mit dem Parameter ACTION spezifiziert wird.

Das Programm wird aufgerufen und die übertragenen Daten werden verarbeitet. Zur Erstellung einer Antwort kann das Programm z.B. Daten aus einer z/OS DB2 Datenbank auslesen, um eine dynamische HTML Seite zu erstellen.

Der HTML Code

```
<FORM METHOD=POST ACTION="/servlet/HelloWorld">
```

ruft ein Java Servlet mit dem Namen HelloWorld auf.

Servlet Container

Servlets laufen in einer Servlet-spezifischen Laufzeitumgebung, die auch als Container oder Servlet Engine bezeichnet wird. Der Servlet Container besteht im Wesentlichen aus einer Reihe von spezifischen Java Klassen, die innerhalb der gleichen JVM wie die Servlets untergebracht sind. Eine Servlet Klasse erbt mit Hilfe von „extends HttpServlet“ die Eigenschaften dieser Container Klassen.

Der Servlet Container verbessert u.a. die Servlet-Ausführungszeit und stellt dem Programmierer vorgefertigte Strukturen zur Verfügung. 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.**

Beispiel: 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);
   }
}
```

Vererbung der Servlet Klassen

Dies ist eine Java Variable mit dem Namen „message“ .

message hat die Form eines html Programms (why not ?) .

Jedes Servlet verfügt über die Methoden init, destroy und service.

Die Methode service ist für die Bearbeitung des Servlet Aufrufs zuständig.

Der in message enthaltene HTML Code wird an den Browser gesendet

Java Server Pages (JSP)

Java Server Pages sind in der Java Programmiersprache geschrieben. Eine JSP ist in Wirklichkeit eine andere Darstellungsform eines Servlets.

JSPs 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.

Dies ermöglicht eine Trennung der HTML Seiten-Logik vom Seitenentwurf und der Seitenwiedergabe.

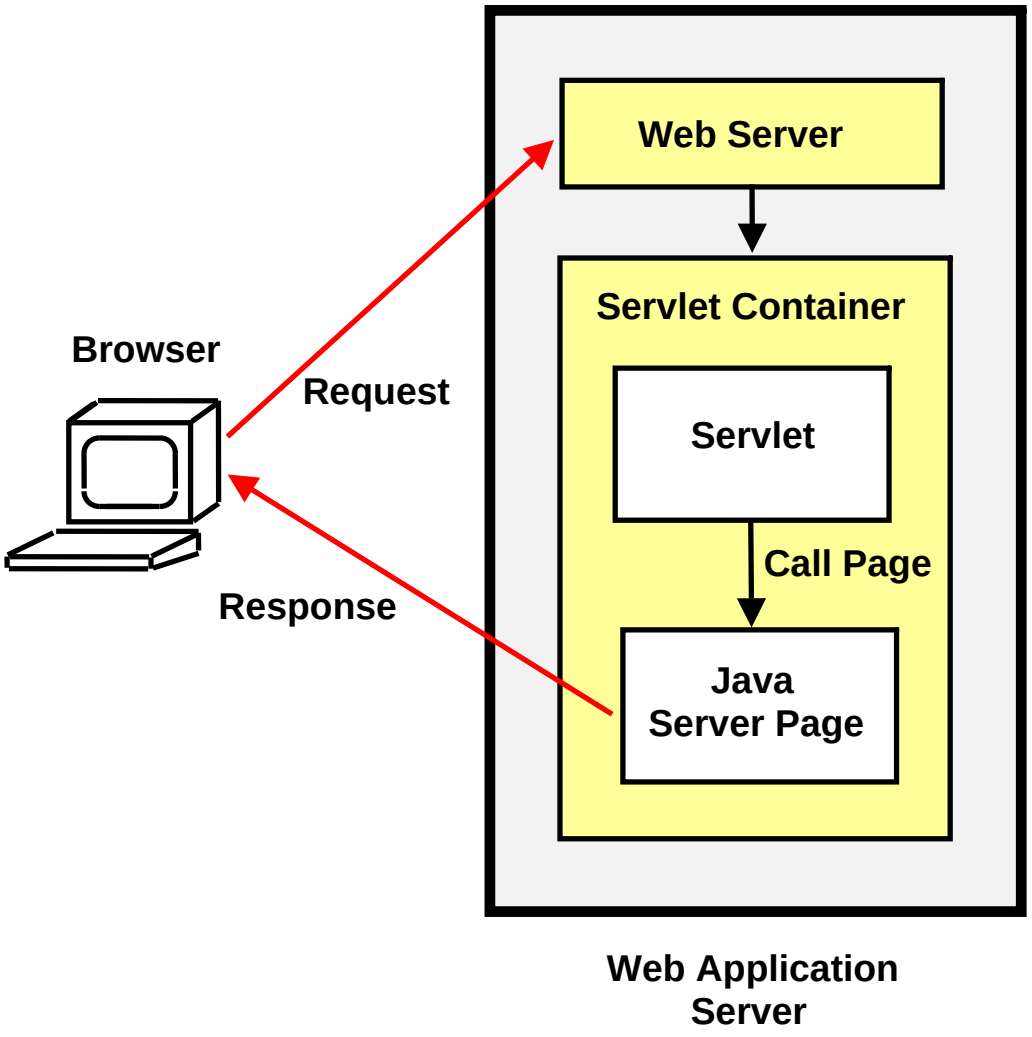
Einfache Java Server Page (1)

Den folgenden Text in einer Datei mit der .jsp extension im JSP directory speichern und mit einem Browser ansehen:

```
<html>
<head>
<title>JSP Example </title>
</head>
<body>
Hello! The time is now <%= new java.util.Date() %>
</body> </html>
```

Die Zeichenfolgen <%= und %> schließen Java Epressions ein. Diese werden zur Run Time ausgewertet. Die Klasse new java.util.Date() ist Bestandteil des JDK. (Normalerweise würde hier ein komplexeres Präsentationslogik Programm stehen).

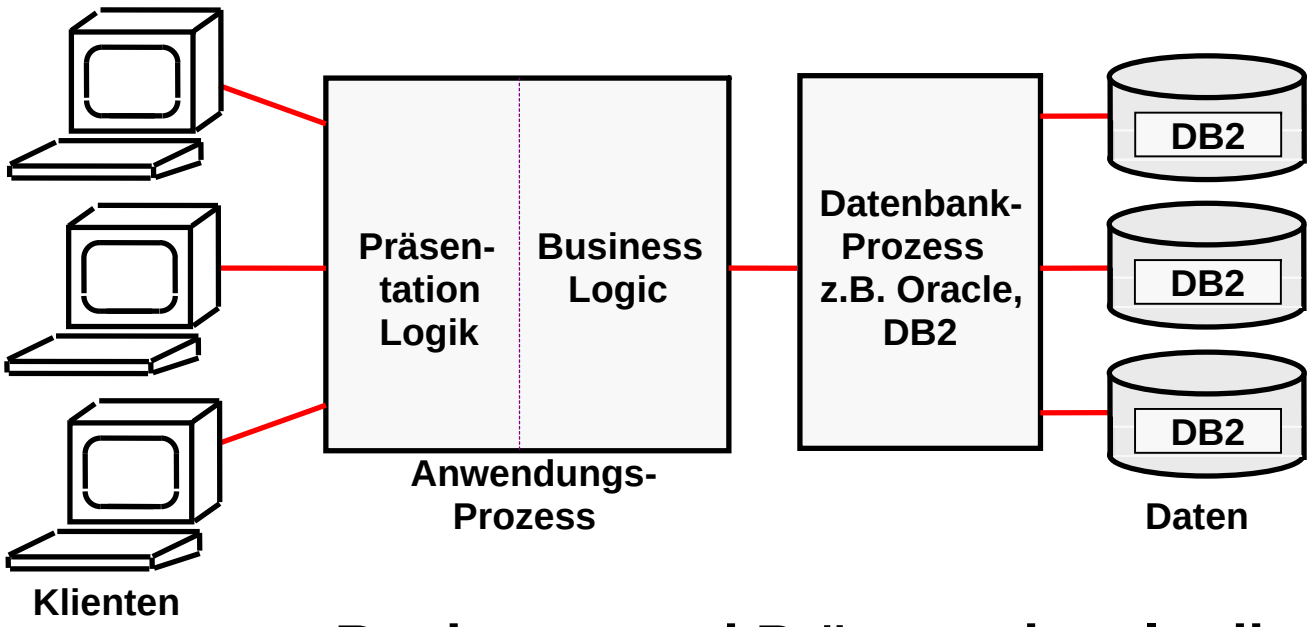
Bei jedem Reload der HTML Seite in den Browser wird die gültige Zeit wiedergegeben.



Interaktion Servlet - JSP

In der Praxis ist es eher selten, dass eine JSP direkt aufgerufen wird.

In der Regel wird ein Servlet aufgerufen, welches wiederum eine JSP aufruft.



Business- und Präsentationslogik

In einfachen Fällen (z.B. einige hundert Zeilen Code) kann ein Servlet die Business Logik und eine JSP die Präsentationslogik implementieren. In komplexeren Fällen ist es sinnvoll, beides in eine größere Anzahl von Java Klassen zu strukturieren. Hier bietet es sich an, den Großteil der Programmierlogik als getrennte Java Klassen zu implementieren, wobei die Servlets und JSPs lediglich Steuerfunktionen übernehmen.

Hierfür bietet sich ein Java Komponenten Modell an, die Java Beans.

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.

Java Beans

Unter Java Beans versteht man kleine Java-Programme (Klassen) mit festgelegten Konventionen für die Schnittstellen, die eine Wiederverwendung in mehreren Anwendungen (Applikationen, Servlets und Applets) ermöglichen, ähnlich wie bei Unterprogramm-Bibliotheken in anderen Programmiersprachen.

Dies ist vor allem im Hinblick auf das Software-Engineering von komplexen Programmsystemen interessant.

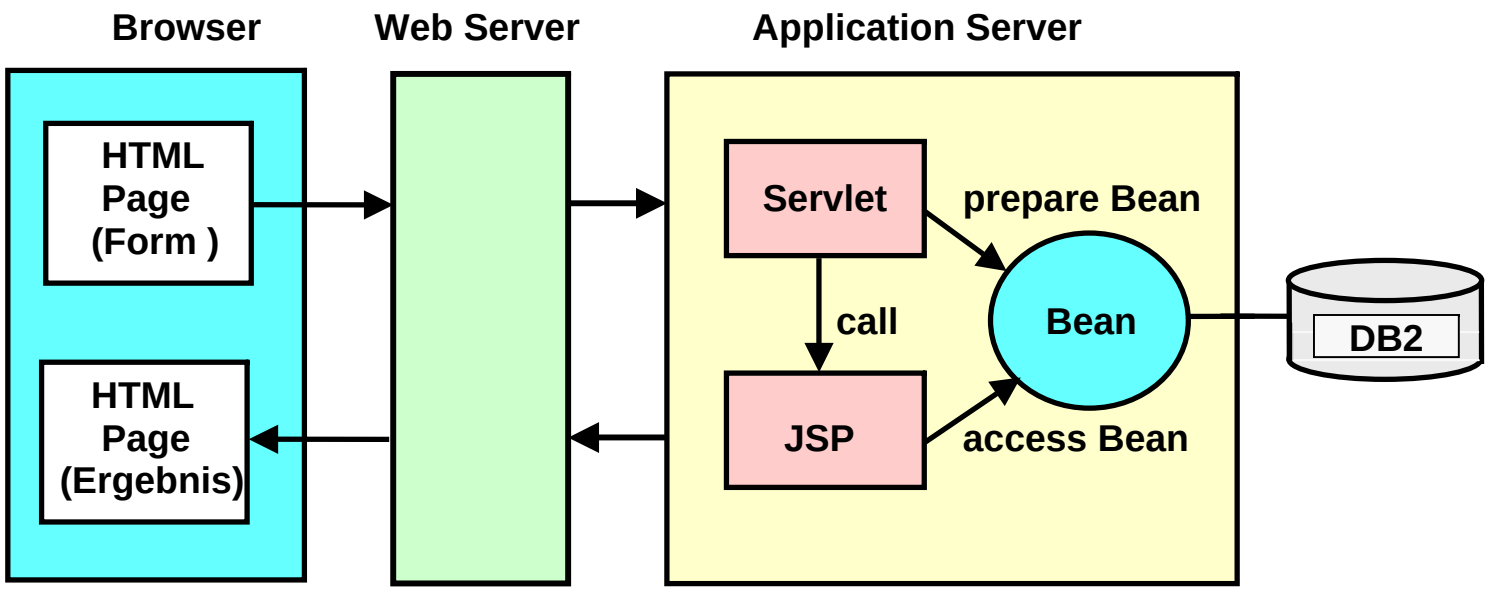
Dafür existiert ein eigenes Beans Development Kit BDK, das man zusätzlich zum JDK installieren kann, und ein Package java.beans, das ab Version 1.1 im JDK enthalten ist,

JavaBeans sind ein Objektorientiertes Java Komponenten Modell. JavaBeans sind Java binary parts. Sie werden häufig für visuelle Komponenten eingesetzt (etwa Buttons und Scrollbalken)

Hauptmerkmale der Java Beans sind:

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

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

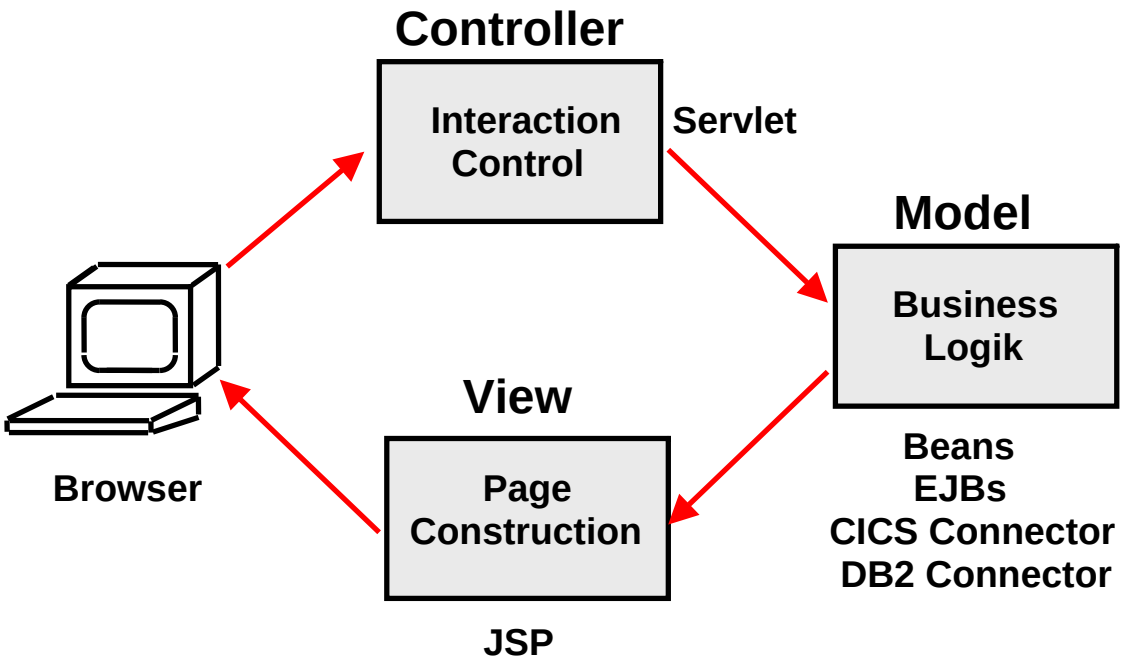
Der Web Server parsed eine HTML Seite und ruft ein Servlet auf.

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.

In der Praxis: Servlets und JSP werden von verschiedenen Programmierern 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.



Als Model/View/Controller Triade wird ein zentrisches Programmier Modell bezeichnet. Die gesamte Anwendungslogik (EJB, Servlet, JSP) läuft auf dem Server. Der Klient (*thin* client) braucht nur einen Browser.

Die Model/View/Controller Triade wird durch die oben dargestellte Kombination von Servlet, Java Beans und JSP implementiert..

Model/View/Controller Triade (MVC)

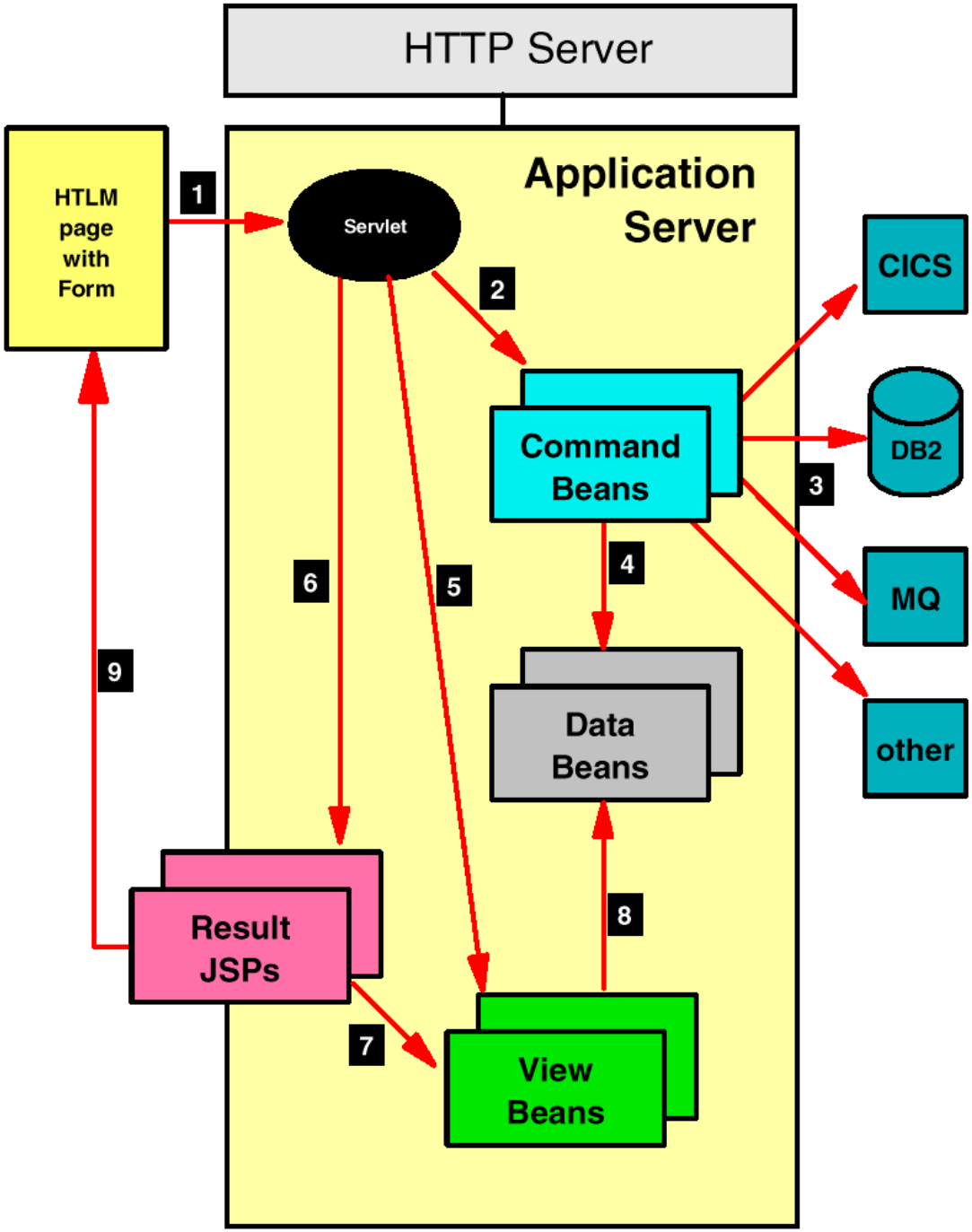
Das “Modell” ist ein Anwendungsobjekt und kapselt die Business Logik. Der “View” ist die Bildschirm Darstellung dieses Objektes. Der “Controller” definiert, wie die Benutzerschnittstelle auf Benutzereingaben reagiert.

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.



Architektur einer JSP Web Anwendung

Erläuterung siehe unten

Model - View - Controller Ansatz

1. HTML-Seite, die in einem vorherigen Schritt erstellt wurde, enthält eine oder mehrere HTML Forms, die ein Servlet für die Verarbeitung des nächsten Interaktion aufrufen.

2. Das Servlet erhält die Kontrolle aus der Application Server zur Validierung und Kontrolle des Ablaufes. Es ruft die Command Beans auf, welche die Geschäftslogik ausführen.

3. Command Beans steuern die Verarbeitung der Geschäftslogik. Die Logik kann in Command Beans eingebettet sein. Alternativ kann die Verarbeitung an Back-End-oder Enterprise-Systeme delegiert werden, wie relationale Datenbanken, MQSeries,Transaktionen Systeme (CICS, IMS) usw. Command Beans rufen Datenbank-und Transaktions-Systemen über „Connectoren" auf..

4. Ergebnisse der Command Beans (oder Back-End-Systeme) werden in Data Beans gespeichert. Data Beans können ein SQL Ergebnis oder eine CICS COMMAREA enthalten. Sie sind das Äquivalent von Unit Records.

5. View Beans definieren, wie Data Beans auf dem Bildschirm dargestellt werden sollen. Ein Servlet initialisiert die View Beans und registriert sie, so dass eine JSPs sie finden kann. View Beans enthalten die gleichen Informationen wie Data Beans, sind aber für die Verwendung durch JSP-Code optimiert.

6. Das Servlet ruft eine JSP zur Ausgabe Verarbeitung und Formatierung in Abhängigkeit von den Ergebnissen der Command Beans auf. JSPs generieren die Ausgabeinformationen für den Browser.

7. JSPs benutzen Tags zur Deklaration der View Beans, sowie um den Zugriff auf alle dynamischen Daten zu erhalten, die in der Ausgabe wiedergegeben werden müssen.

8. View Beans enthalten eine oder mehrere Data Beans und enthalten zugeschnitte Methoden, so dass die JSP Zugriff auf die Daten hat, die in den Data Beans gespeichert sind. Data Beans enthalten nicht notwendigerweise die erforderlichen Methoden, damit eine JSP auf die Daten zugreifen kann.

9. JSP assembliert die Ausgabe und sendet sie als HTML-Seite mit dynamischen Daten zurück an den Browser. In vielen Fällen enthält die Ausgabe wieder Form Tags, damit der Benutzer den Dialog mit der Anwendung fortzusetzen kann.