FH Bielefeld
University of
Applied Sciences

Campus Minden

Webbasierte Anwendungen
SS 2015

Java Server Pages

Dozentin: Grit Behrens
mailto:grit.behrens@fh-bielefeld.de

Studiengang Informatik Fachbereich Technik

Lehrinhaltsübersicht der Vorlesungen zu WBA

1. Einführung in WBA
2. Wiederholung: Grundlagen des WWW, HTML und HTTP
3. Clientseitige Implementierungstechnologien: Javascript, DOM, Ajax, (Java-Applet)
4. **Serverseitige Implementierungstechnologien: JSP**, Java-Servlet
5. Anbindung von Datenbanken
6. WEB-Frameworks: JSF

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. **Einführung in JSP**
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

Einführung in Java Server Pages (JSP)

- Entwickelt von Sun Microsystems für die Realisierung einer **Präsentationsschicht** von Webanwendungen
- **Web-Scriptsprache** auf Grundlage der zahlreichen Java - APIs
- Ermöglichen **Integration** von Businesskomponenten und existierenden Systemen ins Web
- Basiert auf der **Java-Servlet-API (**Standard Tags und erweiterbare Taglibs)
- Dient im Wesentlichen zur einfachen **Erzeugung von HTML- und XML-Seiten** auf WebServern
- **Mischen von HTML (XML) mit JSP** in einer Seite möglich (ähnlich wie z.B. PHP und Perl)
- Ermöglicht konsequente **Aufgabentrennung in Web-Entwicklerteams** (bei steigender Komplexität der Webanwendungen sehr bedeutsam):
  - Präsentationsschicht incl. Design in JSP (möglich ohne Java-Code)
  - Logik – Implementierung in Java
- Daten werden mithilfe von **Java - Beans** in die Webseiten transportiert

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

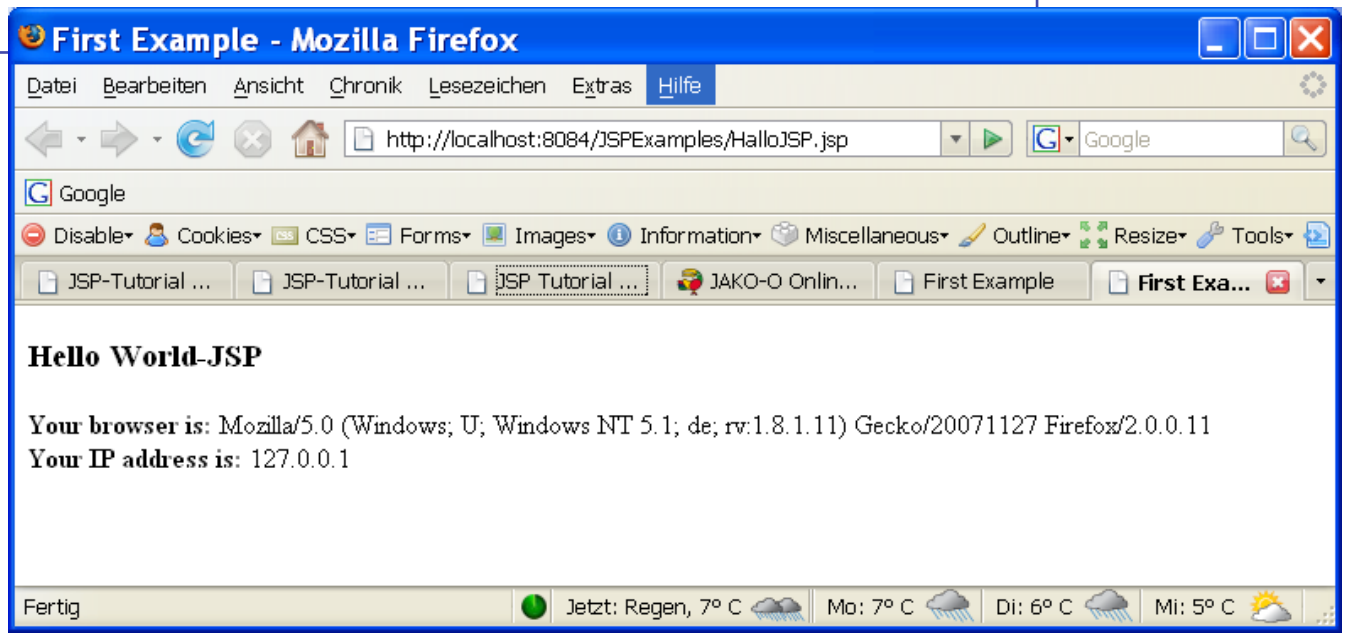
1. Einführung in JSP
2. **Erstes Beispiel „Hello World“**
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

Erstes Beispiel: Hallo.jsp

```
<html>
    <head><title>First Example</title></head>
    <body>
    <h3>Hello World-JSP</h3>

    Your browser is: <%= request.getHeader("User-Agent") %><br>
    Your IP address is: <%= request.getRemoteAddr() %><br>

    </body>
</html>
```



Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

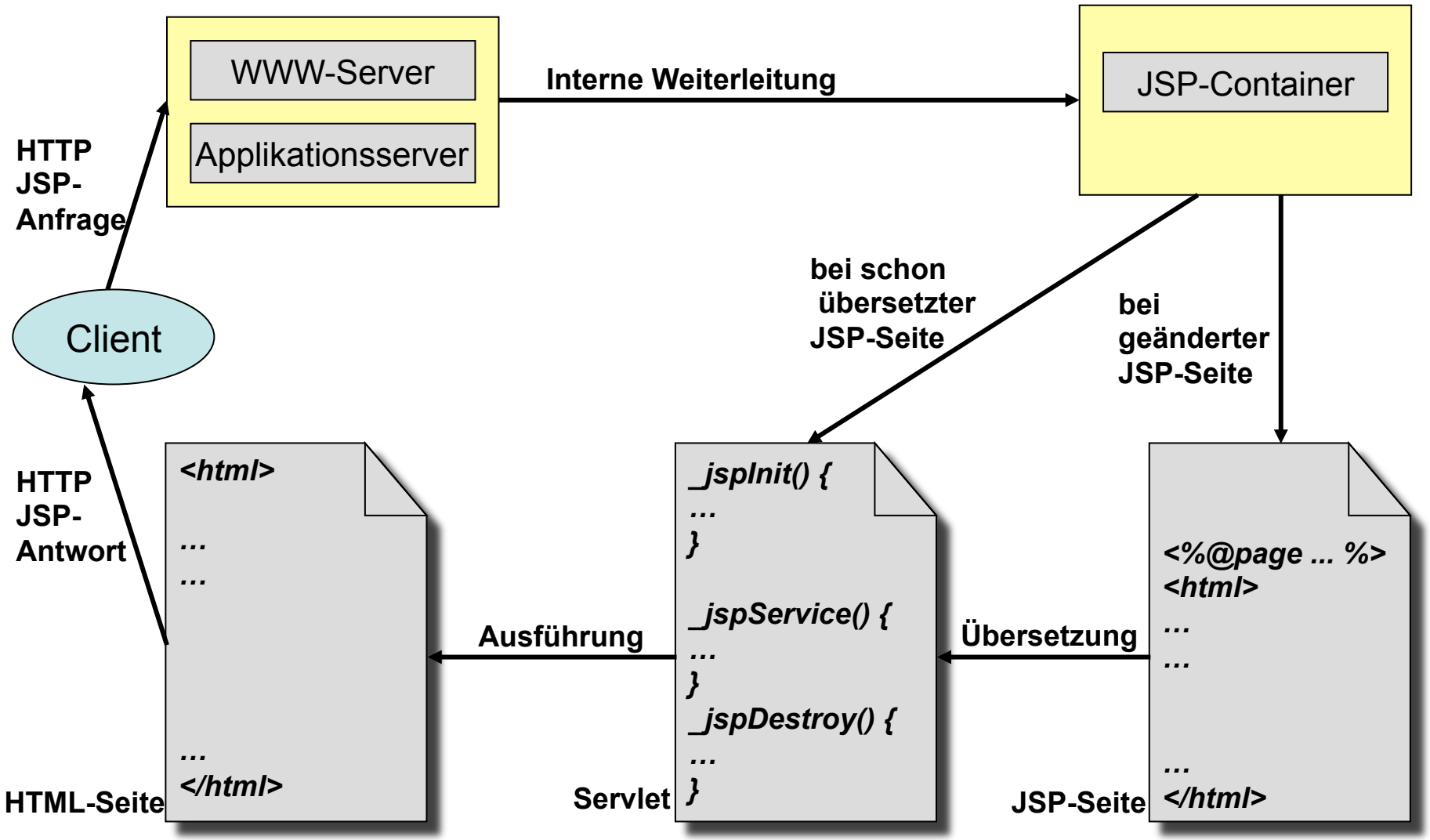
1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. **Lebenszyklus einer JSP**
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

Lebenszyklus einer JSP - Seite

Jede JSP wird im Verlauf Ihres Lebenszyklus vom Container zunächst in ein Servlet transformiert. Zur Ausführung kommen immer nur Servlets.

1. Servlet Container erhält HTTP-Anfrage
2. Servlet Container identifiziert JSP-Anfrage (URL Mapping)
3. Servlet Container ruft JSP Engine auf
4. JSP Engine prüft, ob für Anfrage bereits eine aktuelle kompilierte JSP existiert
5. Falls nicht, wird die entsprechende JSP-Seite aus Dateisystem geladen und ein Servlet erzeugt
6. Übersetzen des Servlets und Bereitstellen für ClassLoader
   - Initialisierung der JSP mit Methode *jspInit()*
   - Bearbeiten der Anfrage in *jspService()* oder auch*: "jspProcessRequest()"*
   - Vernichten mit *jspDestroy()* (falls aktuellere JSP vorliegt oder Servlets heruntergefahren werden sollen)

Lebenszyklus einer JSP-Seite bei Anfrage



Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. **Syntaktische Konstrukte in JSP (Scriptlets)**
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

Scriptlets

Durch Scriptlets wird die Mächtigkeit der Sprache Java in JSP ‘s genutzt.

- Java kann in JSPs als Code eingebettet werden.
- Java-Komponenten können von JSPs heraus genutzt werden.
- Technische Vorraussetzungen:
  - verwendete Bibliotheken liegen im Classpath der Web-Anwendung
  - benötigte Packages sind importiert (Direktiven).

- **Vorteil:**
  - einfache Möglichkeit, beliebigen Java-Code in den JSPs zu verwenden
- **Nachteil:**
  - erzeugter Code schwer lesbar
  - gewünschte Trennung von Präsentation und Logik aufgehoben.

  Scriptlets sollten in modernen großen Webanwendungen nicht eingesetzt werden.

- **Verständnis der Scriptlets wichtig wegen:**
  - Wartung älterer Projekte (sehr häufiger Einsatz)
  - Hilfreich beim Erstellen kleiner Webanwendungen
  - werden entgegen der Warnungen in vielen Fällen auch in größeren Projekten eingesetzt

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. **Vereinbarungen**
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

JSP – Seiten: Konstrukte

Def: JSP-Seiten mit Scriplets *sind Textdokumente und ähneln HTML-Dokumenten, in denen zusätzlicher Java-Code eingepflegt (Skripte genant) ist, mit folgender Syntax:*

<% Java-Code %>

vereinfachte Sichtweise:

1. *Java-Code innnerhalb der JSP-Seite wird vom Container verarbeitet*
2. *der Inhalt der Antwort erzeugt*
3. *Seite an Client gesendet*

JSP-Konstrukte:

- Vereinbarungen
- Anweisungsfragmente
- Ausdrücke
- Interpreter - Anweisungen

JSP - Konstrukte: Vereinbarungen

Vereinbarungen *werden verwendet, um Variablen, Methoden oder innere Klassen zu vereinbaren.*

Syntax: `<% ! Vereinbarung(en) %>`

Beachte:
- mehrere Vereinbarungen in einem Skript möglich
- Initialisierung von Variablen möglich
- Vereinbarungen stehen allen Skripten dieser einen JSP-Seite zur Verfügung

Beispiel:

```
<%! int k = 1; %>
<%!  float meth_f; %>

<%!  String meth_f (int i){
        if ( (k+i) % 2 == 0)
              return “gerade";
        else
              return “ungerade";
     } %>
```

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. **Anweisungsfragmente**
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

JSP - Konstrukte: Anweisungsfragmente

Anweisungsfragmente *werden verwendet, um Java-Anweisungen in JSP-Seiten einzubetten.*

Syntax: <% Anweisungsfragment(e) %>

Beachte:
- zur Ausgabe steht die implizite Variable *out* zur Verfügung
- vereinbarte Variablen dürfen nur in Anweisungsfragmenten verwendet werden, nicht in Methoden von Vereinbarungen

Beispiel:

```
<%
int i=3;
String erg;

erg = meth_f(i);
out.println("Die Summe von "+k+" und "+i+" ergibt eine "+erg+" Zahl.");
%>
```

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
   1. Vereinbarungen
   2. Anweisungsfragmente
   3. **Ausdrücke**
   4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

JSP-Konstrukte: Ausdrücke und Kommentare

Ausdrücke *dienen dazu, den Wert eines Java-Ausdruckes in das Ausgabedokument zu schreiben (auch* ***Scriptlets*** *genannt).*

Syntax: ***<%= Ausdruck %>***

Beachte:

- Schreibweise ohne Semikolon
- Skript der Form: *<%= ausdruck %>*
  ist äquivalent zu dem Skript: *<% out.print(ausdruck);%>*

Beispiel:
Die Summe von <%= i %> und <%= k %> ergibt eine <%= erg %> Zahl.

Kommentare

1. HTML-Kommentar: ***<!-- Kommentar -->*** wird in die HTML-Seite übernommen
2. JSP-Kommentar : ***<%-- Kommentar --%>***
3. Java-Kommentar: ***<% /* Kommentar */ %>***

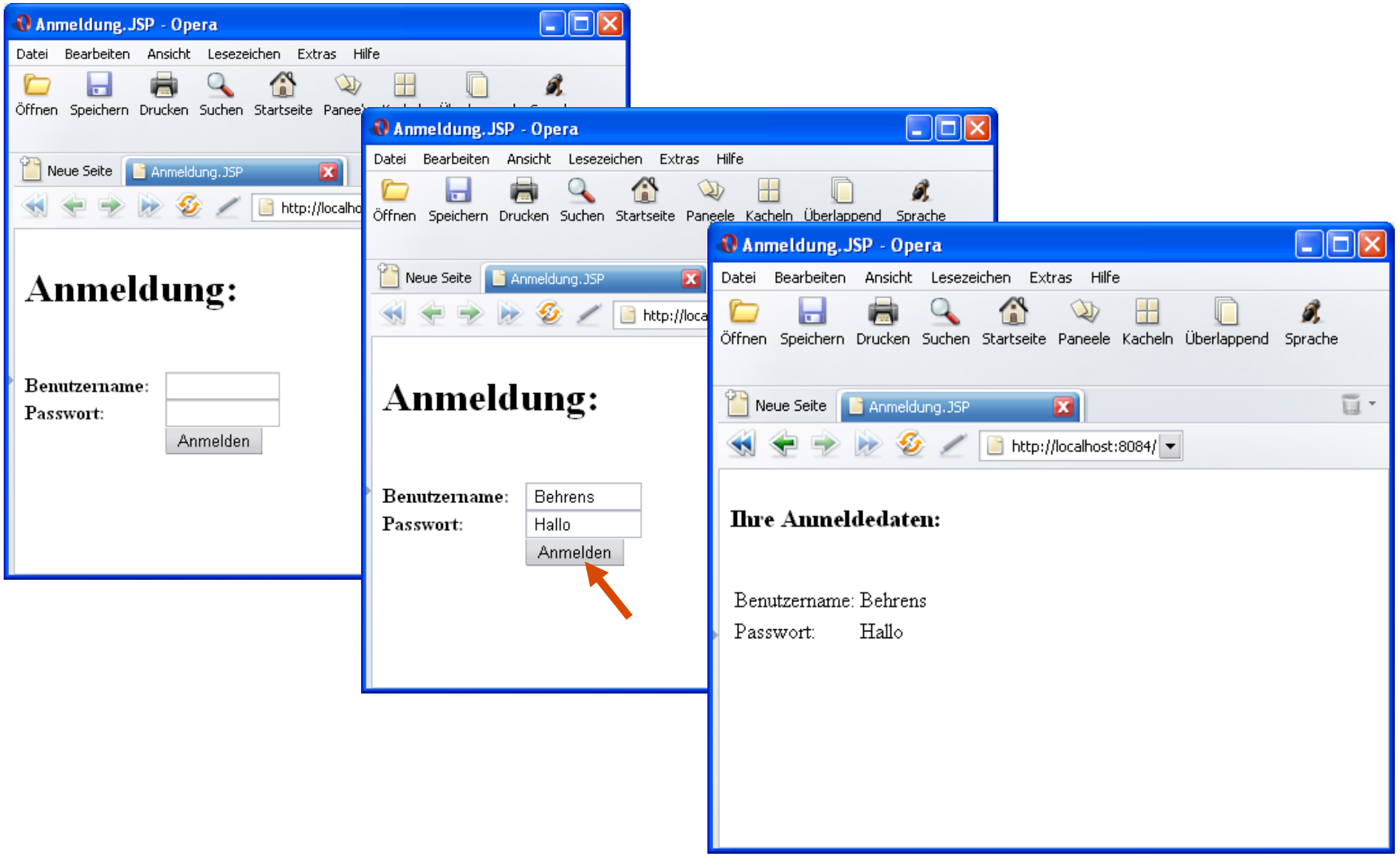
Beispiel für Scriptlets : Anmeldung

```
<html>
<body>
<% //Benutzerdaten einlesen
   String us = request.getParameter("user");
   String pw = request.getParameter("password");

   // falls Angaben unvollst., Form. noch mal  senden...
   if (us ==null || "".equals (us)
|| pw == null || "".equals (pw)) {
%>
<h1>Anmeldung:</h1>
<br>
    <form method = "get">
    <table border="0" cellspacing=0 cellpadding=0>
    <tr>
    <td><b>Benutzername: </b>   </td>
    <td><input type ="text" name = "user" size=12
        maxlength=50 value=
       "<%out.print( us != null? us: " ");%>“>
    </td></tr>
    <tr>
    <td><b>Passwort: </b>   </td>
    <td><input type ="text" name = "password" size=12
        maxlength=50 value=
“<%out.print( pw != null? pw: “ ");%> ">
</td></tr>
weiter…
```

```
weiter …
<tr>
    <td>  </td>
    <td><input type="submit"
      value="Anmelden"></td>
    </tr>
    </table>
    </form>
    <br>
<%
  }
  else {
%>
<h3>Ihre Anmeldedaten:</h3>
       <br>
       <table>
       <tr><td>Benutzername: </td><td> <
      %out.print(us);%> </td></tr>
       <tr><td>Passwort: </td><td> <
      %out.print(pw);%> </td></tr>
       </table>
<%
   }
%>
    </body>
</html>
```

Beispiel für Scriptlets : Anmeldung



Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. **Interpreter - Anweisungen**
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

JSP- Interpreteranweisungen

JSP- Interpreteranweisungen (auch JSP-Direktiven genannt) geben dem JSP- Compiler Hinweise darüber, wie er mit der JSP-Seite und ihren Inhalten umgehen soll.

Syntax:

```
<%@ direktive param1="value1" param2="value2" %>
```

- **bis zur JSP Version 2.0 drei Direktiven:**
  - `include`
  - `page`
  - `taglib`
- **seit JSP-Version 2.0 :**
  - `attribute`
  - `tag`

JSP-Konstrukte: Interpreteranweisungen

Interpreteranweisung **`include`**:

- ermöglicht das Einbinden anderer Dateien in eine JSP-Seite

Syntax:

<%@ include file="relativer_url" %> *(Einfügen einer Text-, HTML- oder JSP-Datei)*

Interpreteranweisung **`page`**:

- ermöglicht die Servletkonfiguration u.a.
  - das Einbinden anderer Java-Klassen (Attribut **`import`**)
  - das Setzen des Zeichensatzes (Attribut **`pageencoding`**)

Syntax:

<%@ page attributename="attributwert" %> *(ein Attribut)*

<%@ page attributname1="attributwert1" *(mehrere Attribute)*
page attributname2="attributwert2"%>

Attribute für Interpreteranweisung page (Auswahl)

Attribut	Bedeutung
import	Import von Java-Klassen in den aktuellen Namensraum
errorPage	Registrierung einer anderen JSP als Error – Handler. Bei Auftreten einer Ausnahme erfolgt eine automatische Weiterleitung an die registrierte Seite.
isErrorPage	Konfiguration einer JSP als Error - Handler. Der Zugriff auf die Ausnahme erfolgt mittels vordefiniertem Objekt namens *exception.*
contentType	Content-Type des Dokumentes
session	Festlegung, ob die JSP an einer Session automatisch beteiligt ist (default = on)
buffer	Definition der Puffergröße für den *PrintWriter* in KB*, oder none* bei ausgeschalteter Pufferung.
autoflush	Festlegung, ob Puffer (*buffer)* automatisch geflusht werden soll, falls er voll ist (default = on).
pageEncoding	Festlegung der Zeichenkodierung (wichtig für internationalisierte Anwendungen)

JSP-Konstrukte: Interpreteranweisungen

***Interpreteranweisung* `taglib`:**

- ermöglicht das Einbinden und Nutzen von Tag-Libraries z.B. Java-Standard Tag Library (JSTL)
- Attribut **`prefix`** für eindeutige Zuordnung von Tags zu Tag-Bibliotheken (gleiche Namen möglich)
- Attribute **`uri`** und **`tagdir`** zur Lokalisierung der Taglibs durch den Container

Syntax:

<%@ taglib uri="tag-library.tld" prefix="pre" [tagdir="WEB-INF/tags"]%>

Beispiel für Einbinden und Nutzung der JSTL:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
…
<ol>
   <c:forEach var="fehler" items="${fehlermeldungen}">
      <li>${fehler}</li>
   </c:forEach>
</ol>
```

…vorausgesetzt, es existiert eine Liste mit dem Namen „fehlermeldungen"

Beispiel für Interpreteranweisung include

Aufgabe: ein einheitlicher Footer aus der Datei „IncludeBeispiel.jsp“ soll in eine JSP „Include.JSP“ eingefügt werden.

Vorgehen:

- Erstellen der Dateien „includeBeispiel.jsp“(wird eingebunden) und „include.jsp“
- Editieren der web.xml (bei Netbeans automatisch erstellt und zeigt auf index.jsp)

includeBeispiel.jsp

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>IncludeBeipiel.JSP</title>
   </head>
   <body>
   <hr>
   <%@ page import = "java.util.Date" %> <!– Import der Java-Klasse -->
   <p><%=new Date() %>
   <p>Designed by Behrens
   </body>
</html>
```

Beispiel für Interpreteranweisung include

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Include JSP</title>
   </head>
   <body>
   <h1>Include JSP</h1>
   <p> Hier steht ein spezifischer Seiteninhalt.
   <br> <br> <br> <br> <br> <br> <br>
   <%@ include file = "includeBeispiel.jsp" %>
   </body>
</html>
```

include.jsp

Beispiel für Interpreteranweisung include

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/
     ns/j2ee/web-app_2_4.xsd"
      version="2.4">
   <session-config>
      <session-timeout>
         30
      </session-timeout>
   </session-config>
   <welcome-file-list>
     <welcome-file>
        include.jsp
     </welcome-file>
   </welcome-file-list>
</web-app>
```

web.xml

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. **JSP und Java Beans**
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

Nutzen von Java - Beans im JSP-Kontext

Bei java - basierten Webapplikationen Nutzung von Beans zur Bereitstellung von Daten für die JSP- basierte Präsentationsschicht :

1. Aufbereiten der Daten für die View und Bereitstellung durch Beans
2. Ausgabe der Daten aus den Beans mit Standard-Aktionen wie
   - jsp:useBean
   - jsp:getProperty
   - jsp:setProperty

- ***Bei reinen JSP-Anwendungen (Modell 1) bequeme Speicherung von Formulardaten in JSP ‘s***

- ***Mit JSP-Version 2.0 kaum noch von Bedeutung***
  - Verdrängung durch Expression Language und den Standard Tags der JSTL

- Es wird aktuell noch sehr zahlreich von Beans Gebrauch gemacht und auch Altprojekte müssen gepflegt werden

JSP und Java Beans

Inhalt von Java- Beans:

- Komponenten - Modell der Sprache Java
- besitzen einen parameterlosen Konstruktor
- stellen getter/setter – Methoden bereit (Zugriff auf die Attribute)

Properties – Attribute der JSPs

Methoden für den Zugriff auf die *Properties*:

public propertyType ***getPropertyName();*** // get-Methode
public void ***setPropertyName(****PropertyType value);* // set-Methode

public boolean ***isPropertyName();*** // is-Property für Boolean-Type, statt get-Methode

Zugriff auf Java-Beans JSP - Aktionen

JSP – Aktionen:

- ***deklarative*** Aktionen für den Zugriff auf Java-Beans aus JSP-Seiten
- in XML notierte standardisierte JSP – Tags
- werden von der JSP – Engine in Java – Code für Servlets übersetzt

Typen von JSP – Aktionen:

- Objekt erzeugen

  < jsp:useBean id="Bezeichner" class="Klassenname" />

- Properties lesen

  < jsp:getProperty name="Bezeichner" property="propertyName" />

weiter… ->

Zugriff auf Java Beans mit JSP - Aktionen

Typen von JSP – Aktionen (weiter):

- Properties setzen

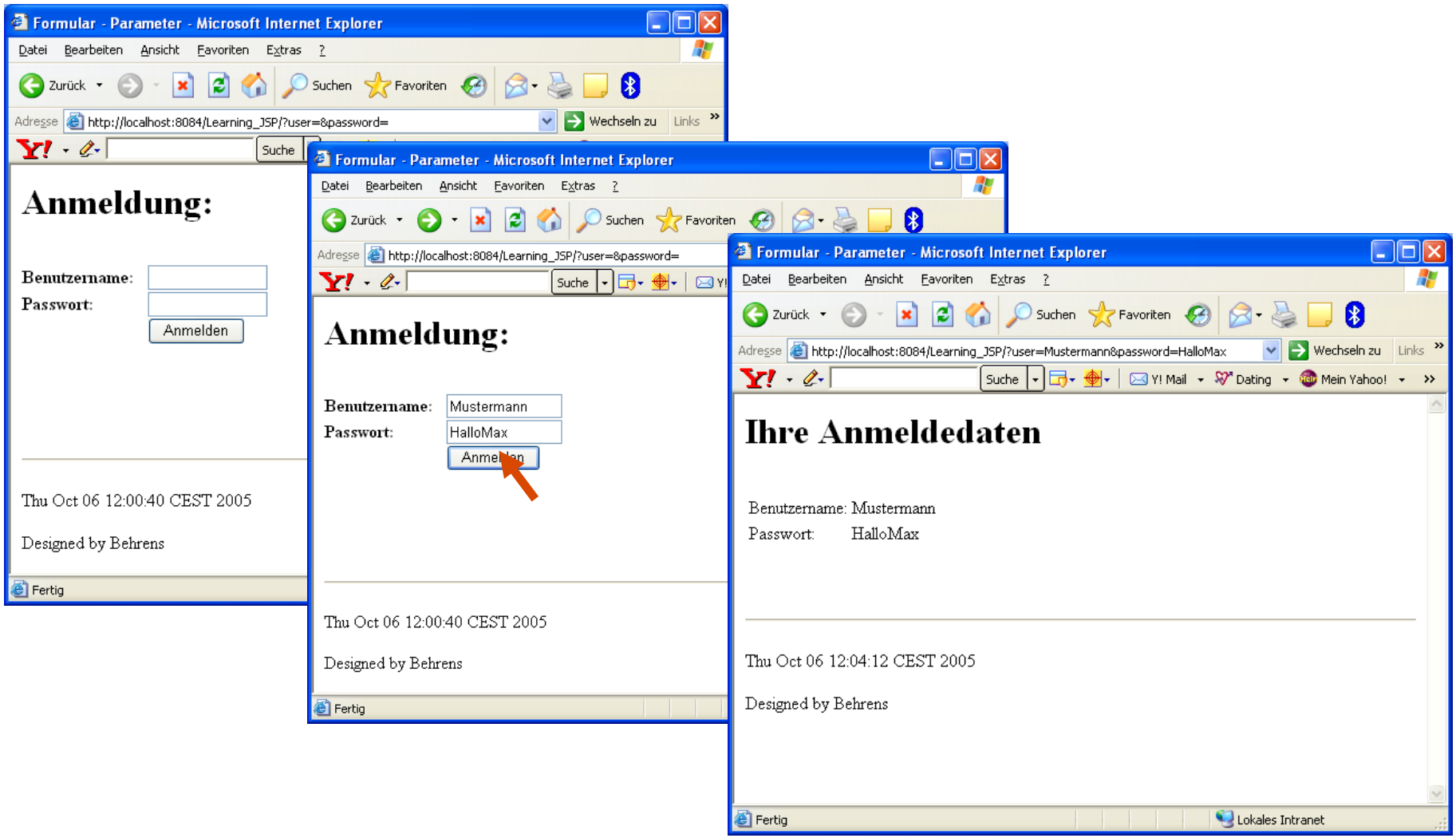
  <jsp:setProperty name="Bezeichner"
  property="propertyName"
  value ="Wert" />

- Alle Properties setzen

  <jsp:setProperty name=„Bezeichner" property="*" />

  Schreiben aller Parameter innerhalb des aktuellen Requests in die gleichnamigen Attribute der JavaBean-Instanz mit der *id* des angegebenen ***Bezeichners***.

Beispiel für JSP – Aktionen: Lesen und schreiben in Java-Bean Klasse Customer



Beispiel : Java Bean Klasse „Customer“

```
public class Customer {
    private String user     = new String();
    private String password = new String();

    //parameterloser Konstruktor,  Instanz von Customer
    public Customer(){}
    public Customer( String user, String password){
        this.user = user;
        this.password = password;
    }

    // getter/setter  - Methoden
    public String getUser(){return user;}
    public void setUser( String user){
        this.user = user;
    }
    public String getPassword() {return password;}
    public void setPassword( String password){
        this.password = password;
    }
}
```

Costumer.java

Wiederholung

Beispiel für Scriptlets : Anmeldung

```
<html>
<body>
<% //Benutzerdaten einlesen
   String us = request.getParameter("user");
   String pw = request.getParameter("password");

  // falls Angaben unvollst., Form. noch mal  senden...
  if (us ==null || "".equals (us)
|| pw == null || "".equals (pw)) {
%>
<h1>Anmeldung:</h1>
<br>
   <form method = "get">
   <table border="0" cellspacing=0 cellpadding=0>
   <tr>
   <td><b>Benutzername: </b>   </td>
   <td><input type ="text" name = "user" size=12
      maxlength=50 value=
      "<%out.print( us != null? us: " ");%> "></td>
   </tr>
   <tr>
   <td><b>Passwort: </b>   </td>
   <td><input type ="text" name = "password" size=12
      maxlength=50 value= “<%out.print( pw != null?
pw: “ ");%> "></td>
   </tr>

weiter…
```

```
weiter …

<tr>
   <td>  </td>
   <td><input type="submit" value="Anmelden"></td>
   </tr>
   </table>
   </form>
   <br>

<%
  }
  else {
%>
<h3>Ihre Anmeldedaten:</h3>
      <br>
      <table>
      <tr><td>Benutzername: </td><td> <
      %out.print(us);%> </td></tr>
      <tr><td>Passwort: </td><td> <
      %out.print(pw);%> </td></tr>
      </table>
<%
  }
%>
   </body>
</html>
```

Beispiel für JSP – Aktionen: Lesen und schreiben in Java-Bean Klasse Customer

```
 …..
else {
%>
   <h1>Ihre Anmeldedaten</h1>
   <jsp:useBean id="customer" class="shop.book.data.Customer"/>
    <!– alle Properties setzen→
   <jsp:setProperty name="customer" property="*" />
   <br>
   <table>
   <tr><td>Benutzername:</td><td><%=customer.getUser()%></td></tr>
   <tr><td>Passwort:</td><td><%=customer.getPassword()%></td></tr>
   </table>
   <%
       }
   %>
   <br> <br> <br>
   <%@ include file = "includeBeispiel.jsp" %>
   </body>
</html>
```

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. **Implizite Objekte**
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

Implizite Objekte

Def: Implizite Objekte *ermöglichen innerhalb von Skripten in JSP-Seiten Zugang zu wichtigen Ressourcen des Servers (z.B. über die derzeit gültige Session, über die Anwendung, den Request, den zu erzeugenden Response,…). Sie* ***werden vom JSP-Server*** *automatisch* ***erzeugt.***

• ***request***	javax.servlet.http.HttpServletRequest
• ***response***	javax.servlet.http.HttpServletResponse
• ***pageContext***	javax.servlet.jsp.PageContext
• ***session***	javax.servlet.http.HttpSession
• ***application***	javax.servlet.ServletContext
• ***out***	javax.servlet.jsp.JspWriter
• ***config***	javax.servlet.ServletConfig
• ***page***	javax.servlet.jsp.HttpJspPage
• ***exception***	java.lang.Throwable

Das implizite Objekt Request

Def: Das Implizite Objekt Request

kapselt alle Informationen über die ***Anfrage an die aktuelle JSP-Seite*** und ist Instanz einer Klasse, welche das Interface **`HttpServletRequest`** aus dem Paket **`javax.servlet.http`** implementiert. Es ermöglicht den Zugang zu den ***http-Headern*** der Anfrage, zu Cookies und Anfrageparametern.

Beispiel:

```
<h4> HTTP-Header</h4>
User-Agent:          <%= request.getHeader("User-Agent")%> <br>
Host      :          <%= request.getHeader("Host")%> <br>
Accept    :          <%= request.getHeader("Accept")%> <br>
Accept-Language:     <%= request.getHeader("Accept-Language")%> <br>
```

Das implizite Objekt Request : Methoden

Methode	Rückgabewert	Kurze Beschreibung
getCookies()	Cookie[]	Liefert alle Cookies des Nutzers als ein Cookie-Array (Vorsicht bei mehreren Anwendungen !)
getHeader(String)	String	Gibt den benannten Header zurück. (z.B. „user-agent“)
getHeaders(String)	Enumeration	Werden mehrere Header verwendet, können diese mittels dieser Methode als Enumeration geholt werden
getHeaderNames()	Enumeration	liefert alle Header-Namen in Form einer Enumeration
getMethod()	String	Gibt die HTTP-Methode (POST oder GET) zurück.
getQueryString()	String	Der Anfrage-String des Clients in der URL (z.B. Browsereingabe oder Formular mit GET-Methode)
getRequestedSessionId()	String	Gibt die Session-ID zurück, die der Client anfragt.
getRequestURL()	String	Die URL die für die Anfrage genutzt wurde ohne Parameter
getSession()	HttpSession	Gibt die Session des Nutzers zurück.

Das implizite Objekt Response

Def: Das Implizite Objekt Response

kapselt alle Informationen über die ***Antwort an die aktuelle JSP-Seite*** und ist Instanz einer Klasse, welche das Interface `HttpServletResponse` aus dem Paket `javax.servlet.http` implementiert. Es enthält alle Informationen für die HTTP-Response und wird vor allem genutzt, um den *Content-Type* festzulegen und *Response-Header* oder *Cookies* zu setzen.

Häufig verwendete Methode:

setContentType()

Hinweis:

Will man bspw. mit JSPs XML-Code erzeugen, so muss der Content-Type mittels response.setContentType("text/xml") auf "text/xml" gesetzt werden.
für wml: „vnd.wap.wml“

Das implizite Objekt Response : Methoden (I)

Methode	Rückgabewert	Kurze Beschreibung
addCookie(*Cookie cookie)*	void	Fügt den übergebenen Cookie der Response hinzu
addDateHeader(*String name, long date)*	void	Formatiert einen Datums-String und fügt diesen unter dem angegebenen Namen den Http-Headern hinzu
addHeader(*String name, String value)*	void	Fügt einen Header unter dem angegebenen Namen hinzu
addIntHeader(String name, int value)	void	Unter dem angegebenen Header-Namen wird ein Ganzzahl-Wert hinzugefügt
encodeURL(*String url)*	String	Bereinigt die URL von Sonderzeichen, Leerzeichen und dergleichen in einer URL nicht zugelassenen Zeichen

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. **Aktionen (JSP – Tags)**
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

Aktionen oder : JSP- TAGS

Aktionen sind spezielle XML- ähnliche Elemente zur Erzeugung und Bearbeitung von Objekten zur Steuerung der Verarbeitung des Contents.

- Aktionen können stammen aus:
  - Standard – Aktion (stehen immer zur Verfügung)
  - Selbstentwicklung (Costum Tags, müssen in JSP eingebunden sein)
  - Zukauf kommerzieller Tags (auch Costum Tags,-“-)
  - Standard Tag Library

- Beispiele für Tags:
  - Steuerungskonstrukte wie "if", "for", "try/catch" (Verfügbarkeit ohne Skriptlets)
  - Formatierung und Internationalisierung
  - Formatieren der Fehlerausgaben
  - beliebige andere Tags denkbar wie komplett formatierte Grafiken, Reports oder vom Desktop her geläufige Controls wie bspw. Ausklappmenüs

- Tags werden in Tag-Libraries zusammengefasst
- zu jeder Library gehört ein XML-Deployment-Deskriptor.

Aktionen oder : JSP- TAGS

Syntax von Tags (ähnlich zu XML):

```
<pre:aktion attribut1="wert1" attribut2="wert2"...>
      möglicher Body des Tags.
</pre:aktion>
```

`pre:` steht für Prefix
- wird im Kopf der JSP - Seite definiert
- `jsp:` für Standard Tag Library
- beliebig für Costum Tag Libraries

Standard Aktionen (Auswahl)

<jsp:include>

- Alternative zur Direktive **<%@ include file="relativer_url" %>**

<jsp:forward>

- Weiterleitung des Requests an andere JSP-Seite
- attribute **page** legt Seite fest (relativer Pfad), am die HTTP-Request gehen soll

<jsp:param>

- Übergabe von Parametern an Aktionen wie z.B. bei **<jsp:forward>**
- **Beispiel:**

```
<jsp:forward page="tutorial.jsp" >
   <jsp:param name="chapter" value="includes" />
</jsp:forward>
```

<jsp:element>

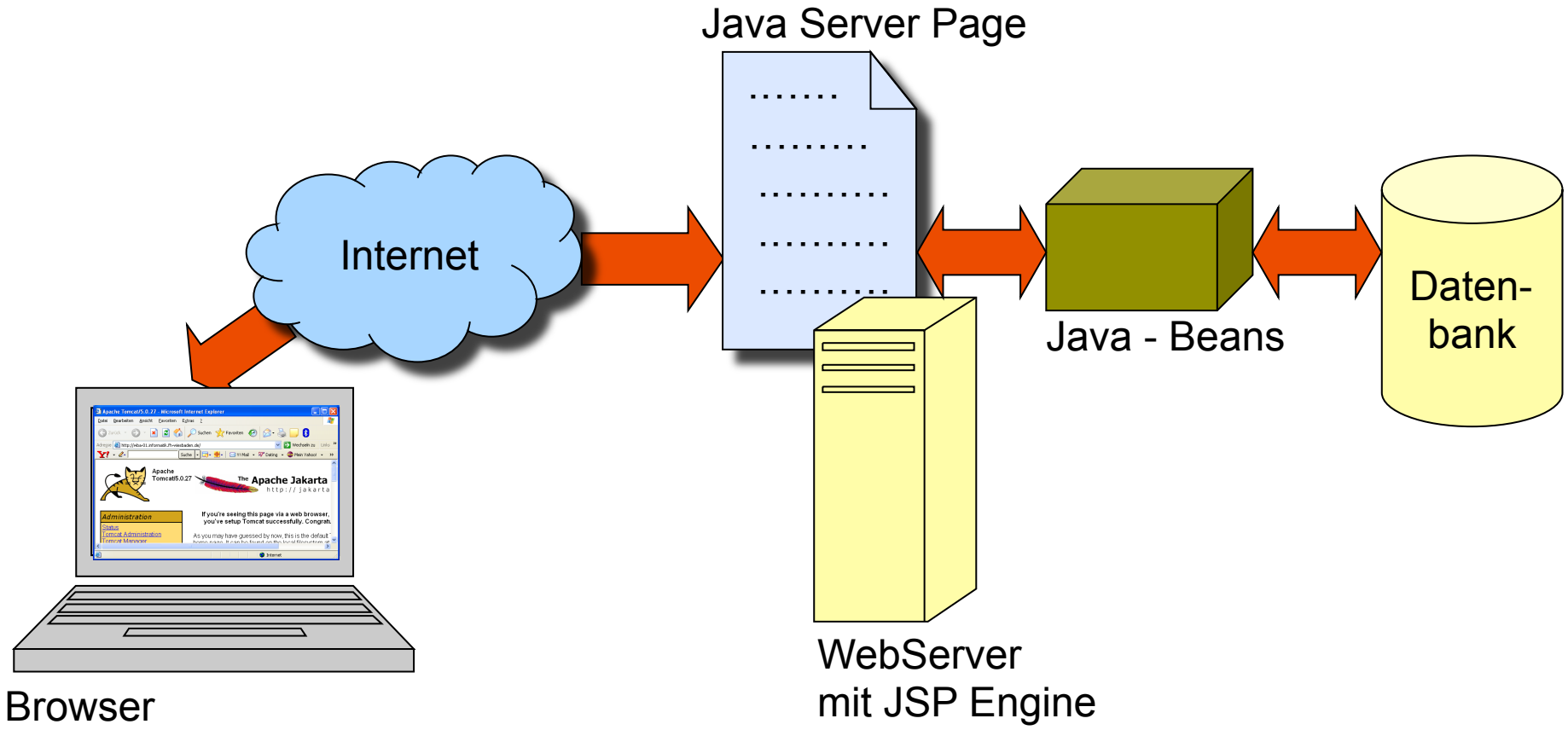
- ermöglicht das Einbinden beliebiger eigener Tags mit Attributen und Body
- **Beispiel:**

```
<jsp:element name="meinDynamischerTag">
   <jsp:attribute name="isXml">true</jsp:attribute>
   <jsp:attribute name="isHtml">false</jsp:attribute>
   <jsp:body>irgendein body für das Tag</jsp:body>
</jsp:element>
```

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. **JSP - Architekturen**
9. Webserver
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

JSP Model 1: Architekturschema



Daten werden mithilfe von Java-Beans in die Webseiten transportiert

JSP Architektur - Model 1

Merkmale:

- Für jede Webseite gibt es eine eigene JSP
- alle Links auf andere Seiten sind in den JSP ‘s angegeben
- Geschäftslogik liegt gewöhnlich außerhalb der JSP
  -> saubere Trennung von Präsentation und Geschäftslogik möglich
- es gibt aber auch oft innerhalb der JSP ‘s Geschäftslogik in Form von Scriptlets (z.T. unübersichtlich)

Für kleine Webanwendungen in gut funktionierenden Teams kann JSP-Model 1 ausreichend sein.

JSP Architektur - Model 2

Merkmale:
- Übertragung des MVC- Musters in die Web- Anwendungen
- keine seitenzentrierte Sichtweise mehr
- Anwendung im Struts – Framework und im JSF – Framework (erste Versionen mit jsp später Lösungen mit xml-seiten)
- Inhalt der weiteren Veranstaltung

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. **Webserver**
10. Zusammenfassung
11. Literatur
12. Hinweise zum Praktikum

Organisation von Web - Anwendungen

Motivation:

- Web-Anwendung besteht aus mehreren Komponenten:
  - Servlets,
  - JSP-Seiten,
  - HTML-Dokumenten,
  - Bilder,
  - Applets,
  - Java Beans,
  - …
- einheitliche Portierung auf verschiedene Server notwendig!

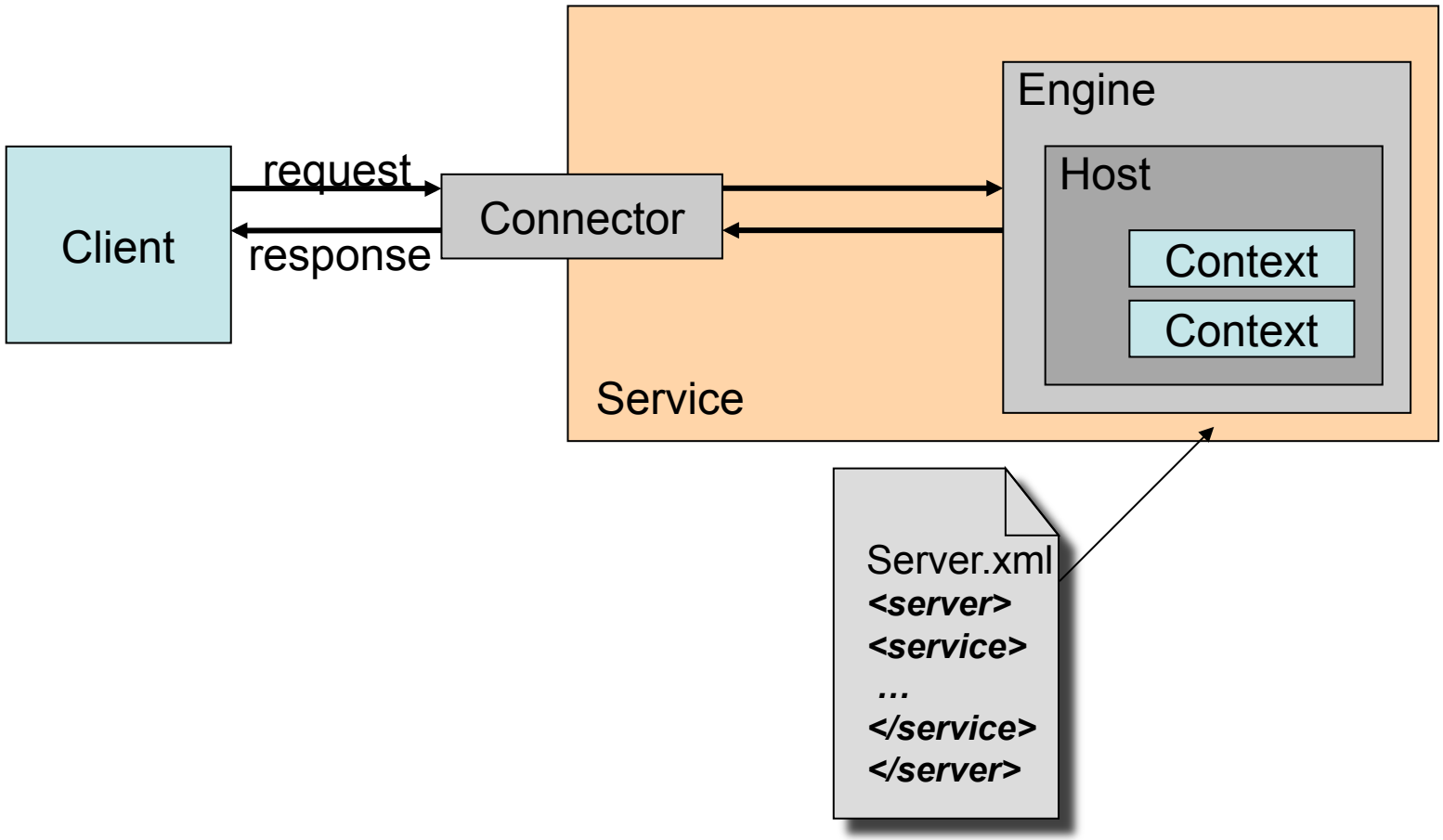
Lösung:

→ Servlet Spezifikation 2.2 enthält Konzept der ***Web-Applikation.***

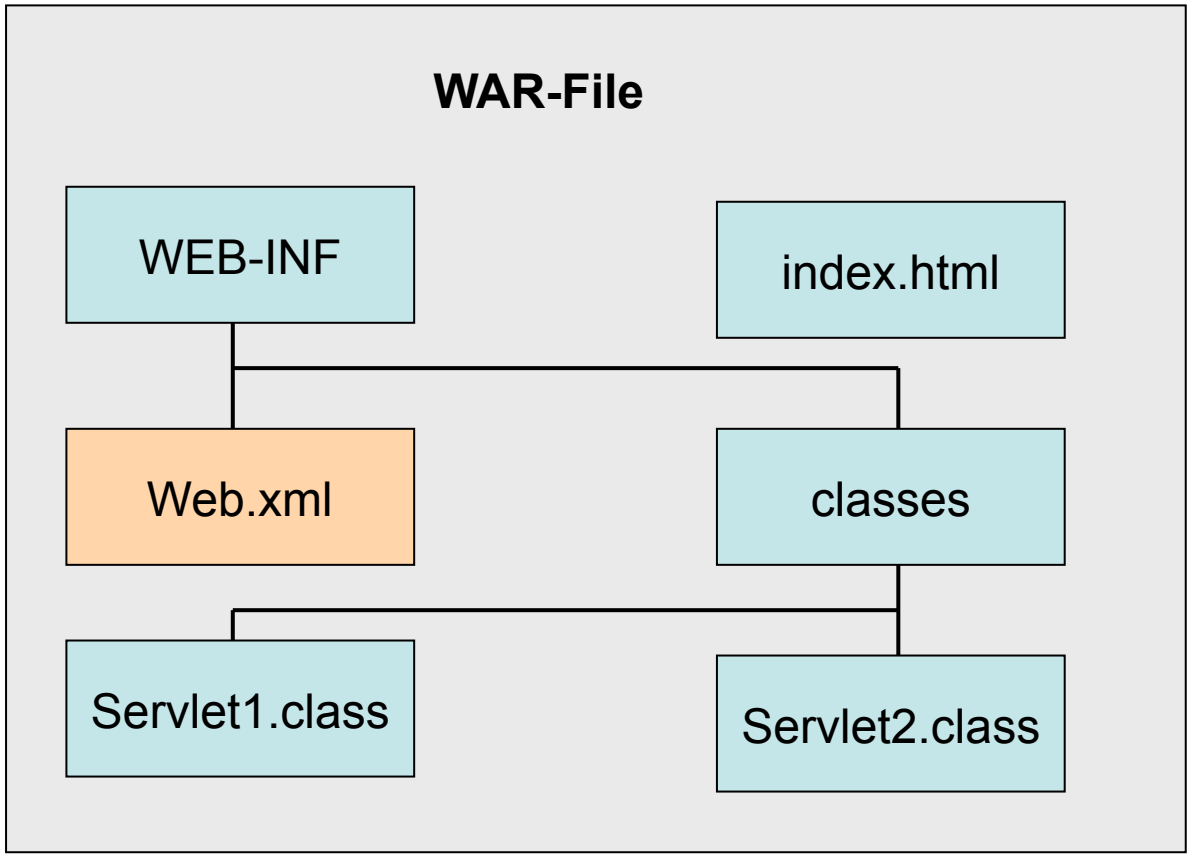
1. ***Web Application Archives (WAR-Archive)***
   fassen alle Komponenten einer Web-Anwendung zusammen
2. ***Deployment-Deskriptoren (xml- Dateien)***
   enthalten alle Abhängigkeiten von externen Ressourcen und einstellbare Parameter

WebServer

TomCat (Catalina Container):Komponenten



Aufbau des Web - Archives



TomCat - Verzeichnisse

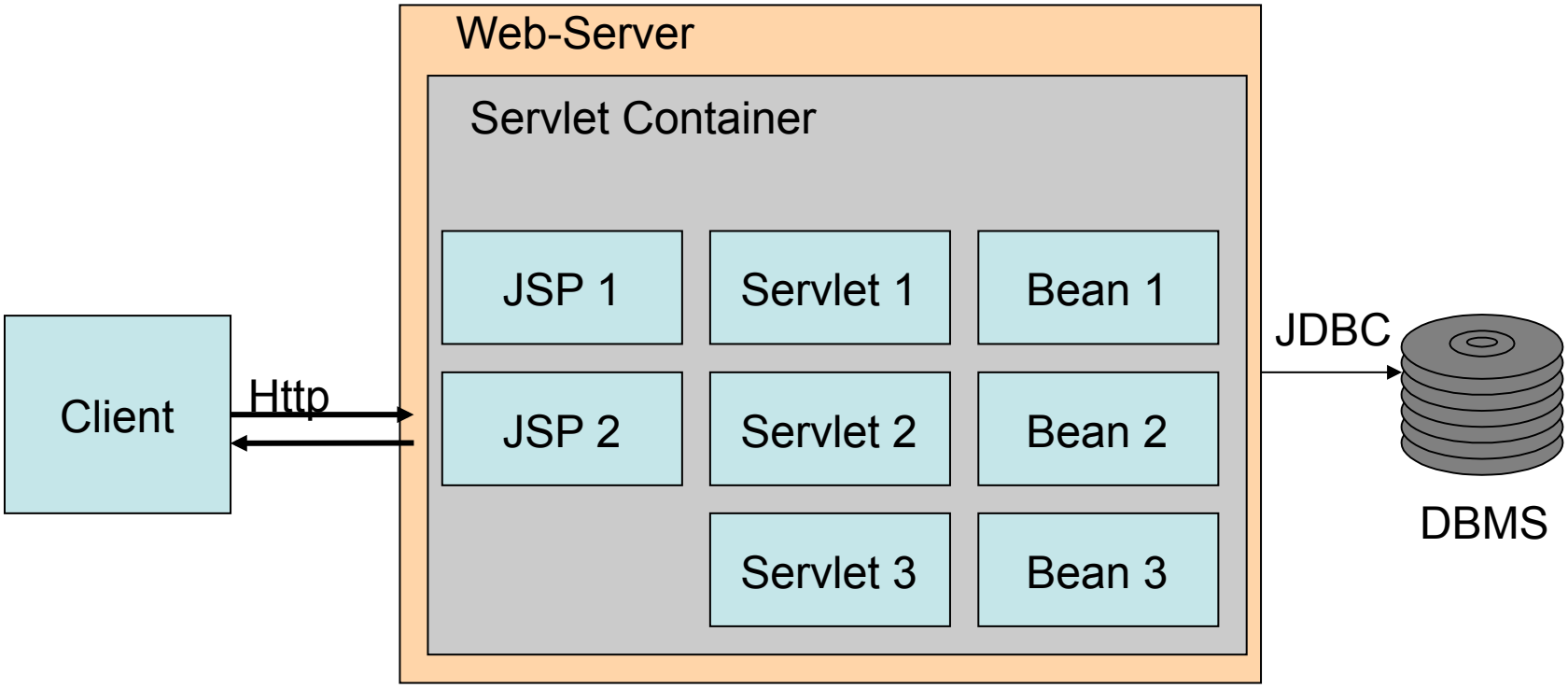
- **Bin** Skripte für den Betrieb (*shutdown, startup*)
- **Conf** Konfigurationsdateien (*server.xml, web.xml*)
- **Doc** TomCat Dokumentation
- **Lib** JAR-Archive für TomCat. Hier können auch eigene Archive hineinkopiert werden, das Startscript nimmt sie automatisch in den CLASSPATH auf

- **Logs** Logdateien von TomCat und Jasper
- **Src** TomCat und Servlet-API-Quellcode (optional)
- **Webapps** Web-Applikationen enthält auch Beispiele

- **Work** erstellt Tomcat automatisch für temporäre Dateien während der Laufzeit

- **Classes** optionales Verzeichnis für eigene Klassen, die automatisch in den Classpath aufgenommen werden

Technologien: Java–Servlet und JSP



Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. **Zusammenfassung**
11. Literatur
12. Hinweise zum Praktikum

Zusammenfassung

- Syntax und Funktionalität von Java Server Pages an Beispielen zu u.a. Anmeldeformularen gelernt.
- Zugriff von JSPs auf Java-Beans mit Beispielen
- Häufig verwendete implizite Objekte Request und Response
- Aktionen in Tag Libraries vorgestellt
- JSPs als Bestandteil von Webanwendungen

Serverseitige Implementierungstechnologien (I): Java Server Pages (JSP)

1. Einführung in JSP
2. Erstes Beispiel „Hello World“
3. Lebenszyklus einer JSP
4. Syntaktische Konstrukte in JSP (Scriptlets)
    1. Vereinbarungen
    2. Anweisungsfragmente
    3. Ausdrücke
    4. Interpreter - Anweisungen
5. JSP und Java Beans
6. Implizite Objekte
7. Aktionen (JSP – Tags)
8. JSP - Architekturen
9. Webserver
10. Zusammenfassung
11. **Literatur**

Literatur für Java Server Pages

- Heiko Wöhr „Webtechnologien“ , dpunkt.verlag, Heidelberg 2004
- Volker Turau: „Java Server Pages und J2EE – Unternehmensweite Web-basierte Anwendungen“, dpunkt.verlag GmbH Heidelberg 2001
- Volker Turau, Krister Saleck, Christopher Lenz: Web – basierte Anwendungen entwickeln mit JSP 2.0“, dpunkt.verlag 2004
- http://www.heise.de/ix/artikel/2000/07/152/ (3 Artikel zu JSP-Tutorial)
- http://www.jsptutorial.org

Ausblick:

Serverseitige Implementierungstechnologien:

Java-Servlet