B7. Web-Programmierung mit Java

B7.1 Applets
B7.2 Servlets
B7.3 Java Server Pages (JSP) 

Literatur:
Volker Turau/Ronald Pfeiffer: Java Server Pages, dpunkt 2000
Bruce Perry: Java Servlet & JSP Cookbook, O'Reilly 2004
http://java.sun.com/j2ee/1.4/docs/tutorial/doc/ (chapter 12)

Server-seitige Lösungen: Überblick (1)

- Common Gateway Interface (CGI)
  - einfach zu verwenden
    - » Parameter über Umgebungsvariablen
    - » Ergebnis Text auf Standardausgabe
  - CGI-Anwendungen in jeder Programmiersprache realisierbar
  - Nachteile:
    - » Schlechte Performance, keine Unterstützung von "Sitzungen"
- Web-Server APIs
  - Beispiele: NSAPI (Netscape), ISAPI (Microsoft), Java Servlets (Sun)
  - Dynamisches Laden von Programmteilen in den Server
  - Vorteile:
    - » Bessere Performance, Realisierbarkeit von Transaktionen, ...
  - Nachteile:
    - » teilweise proprietär; schlecht portabel
    - » schlechte Trennung von Anwendungslogik und Präsentation

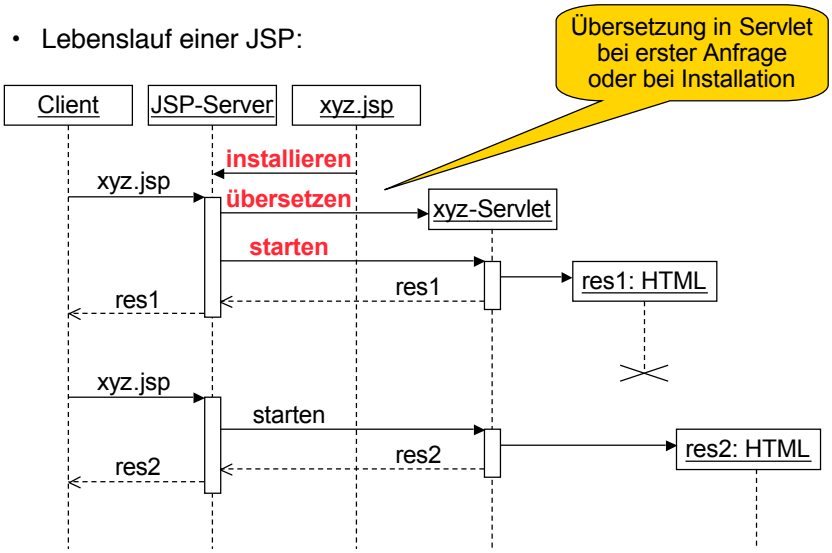
Server-seitige Lösungen: Überblick (2)

- Server-Side Includes
  - Erstmals im NCSA Web-Server realisiert: "umgekehrte Einbettung"
  - Eingeschränkte Anweisungen; keine volle Programmiersprache

- Server-seitige Skripte (Aktive Server-Seiten)
  - Benutzung vollwertiger Programmier- oder Skriptsprache
  - Beliebte Sprache für Server-Skripte: PHP (Personal Home Page Toolkit)
  - Microsoft Active Server Pages (ASP)
    - » Verwendung verschiedener Skriptsprachen (JScript, VBScript)
    - » Einsatz von Komponenten (Active/X und DCOM)
  - Java als Skript-Sprache: Java Server Pages (JSP)
    - » Einsatz von Komponenten (JavaBeans und Enterprise Java Beans)

Java Server Pages und Servlets

- Lebenslauf einer JSP:



JSP-Sprachelemente

- Skript-Elemente
  - Einbettung von Java-Code
- Implizite Objekte
  - Einfacher Zugriff auf wichtige Servlet-Bestandteile
- Direktiven
  - Globale Anweisungen an Übersetzungsvorgang
- Aktionen
  - Standardelemente für Laufzeitverhalten

- Prinzipiell kann JSP zur Generierung beliebiger Texte verwendet werden.
  - Neben HTML zunehmend wichtige Zielsprache: XML

Einbettung von Java-Code in HTML

- Möglichkeiten zur Einbettung:
  - Spezielle Tags (z.B. <script> für JavaScript)
    - » Gefahr der Inkompatibilität mit HTML-Weiterentwicklung
  - Tags aus Sonderzeichen
    - » Unelegant, aber bequem manuell zu handhaben
    - » JSP: <%, <%!, <%=, <%@, %>, <%--, --%>
  - XML-Syntax mit *Namespaces*
    - » "Namespace" (xmlns) durch URL definiert, z.B. "jsp"
    - » Tags der Form <jsp: xyz>

- JSP benutzt ***zwei*** Varianten der Einbettung
  - Sonderzeichen (JSP-Syntax)
  - XML-Syntax prinzipiell immer möglich, aber vor allem für Aktionen verbreitet

JSP-Skript-Elemente

- Vereinbarungen
  - Syntax: `<%! Vereinbarungen %>`
    `<jsp:declaration> Vereinbarungen </jsp:declaration>`
  - Beispiel: `<%! String title = "Date JSP"; %>`
  - Wird in Instanzvariable der generierten Klasse übersetzt, d.h. Werte bleiben über einzelne Requests hinaus erhalten!
- Anweisungen (*Scriptlets*)
  - Syntax: `<% Anweisungen %>`
    `<jsp:scriptlet> Anweisungen </jsp:scriptlet>`
  - Beispiel:
    ```
    <% java.util.Date now = new
         GregorianCalendar().getTime(); %>
    ```
  - Lokale Variablen: in anderen Anweisungen sichtbar, nicht in Methoden
- Ausdrücke
  - Syntax: `<%= Ausdruck %>`
    `<jsp:expression> Ausdruck </jsp:expression>`
  - Beispiel: `<%= now %>`
  - Äquivalent zu `<% out.print(now); %>`

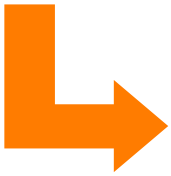
Implizite Objekte in JSP-Skripten

- Auswahl der wichtigsten impliziten Objekte:
- **`request (javax.servlet.http.HttpServletRequest)`**
  - Lesen von HTTP-Headern, Parametern, Cookies etc. der Anfrage
- **`response (javax.servlet.http.HttpServletResponse)`**
  - Ausgeben von HTTP-Headern, Cookies etc. in der Antwort
- **`session (javax.servlet.http.HttpSession)`**
  - Verfolgung von "Sitzungen" (zusammengehörigen Interaktionen)
- **`out (javax.servlet.jsp.JspWriter)`**
  - Ausgabestrom (Ergebnisseite)
  - Übliche print- und println-Operationen stehen zur Verfügung
- Beispiel:

```
<% if (request.getParameter("CountButton")!=null) {
     counter.count();
   }; %>
```

Erzeugter Servlet-Code (Auszug)

```
<html>
  <%! String title = "Date JSP"; %>
  <head>
      <title> <%=title%> </title>
  </head>
  <body>
      <h1> <%=title%> </h1>
      <p>Current time is:
            <% java.util.Date now = new GregorianCalendar().getTime(); %>
            <%=now%>
  </body>
</html>
```



```
...
out.write("\r\n");
out.write("\t<body>\n");
out.write("\t\t<h1> ");
out.print(title);
out.write(" </h1>\n");
out.write("\t\t<p>Current time is:\n");
out.write("\t\t\t");
java.util.Date now = new GregorianCalendar().getTime();
out.write("\n");
out.write("\t\t\t");
out.print(now);
out.write("\n");
```


JavaBeans in JSP: Aktion useBean

- Syntax für useBean-Aktion:

```
<jsp:useBean id=LokalerName class=KlassenName
             scope=Gültigkeitsbereich />
```

  scope: "page“ (aktuelle Seite), "request" (aktuelle Anfrage)
  "session" (aktuelle Sitzung), "application" (gesamte Anwendung)

- Lesen von Eigenschaften:

```
<jsp:getProperty   name=LokalerName
                   property=EigenschaftsName/>
```

- Setzen von Eigenschaften:

```
<jsp:setProperty   name=LokalerName
                   property=EigenschaftsName/>
                   value=WertAlsString/>
```

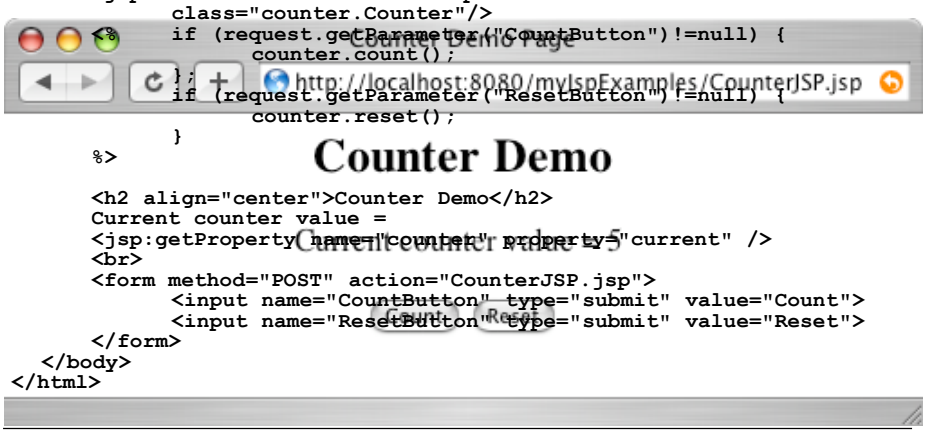
```
<jsp:getProperty   name=counter property=current/>
```

ist gleichwertig zu:

```
<%=counter.getCurrent;%>
```

Zähler mit Java Server Pages: HTML-Quelle

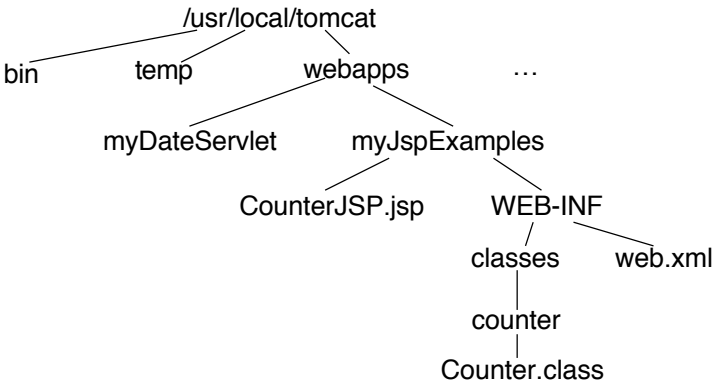
```
<%@ page contentType="text/html" session="true"%>
<%@ page language="java"%>
<html>
  <head><title>Counter Demo Page</title></head>
  <body>
      <jsp:useBean id="counter" scope="session"
            class="counter.Counter"/>
      <%
            if (request.getParameter("CountButton")!=null) {
                  counter.count();
            };
            if (request.getParameter("ResetButton")!=null) {
                  counter.reset();
            }
      %>

      <h2 align="center">Counter Demo</h2>
      Current counter value =
      <jsp:getProperty name="counter" property="current" />
      <br>
      <form method="POST" action="CounterJSP.jsp">
            <input name="CountButton" type="submit" value="Count">
            <input name="ResetButton" type="submit" value="Reset">
      </form>
  </body>
</html>
```



Installation auf JSP-Server/Servlet-Container

- Dateikonventionen über Ablageort
- Kompaktes Archiv für Web-Anwendung: "Web Archive (WAR)"

Beispiel:



Medieninformatik: Gestaltung & Systementwicklung

- Neue Medien nutzen komplexe neue Technologien
  - Z.B. Java Server Pages
- Medieninformatiker
  - ***Müssen*** gut programmieren können
  - Müssen fortgeschrittene Software-Technologien kennen … und sich in solche schnell einarbeiten können
  - Müssen Hardware und Software für Multimedia-Bearbeitung kennen … und sich in solche schnell einarbeiten können
  - Müssen Verständnis für kreative Gestaltung haben, einschließlich eigener praktischer Erfahrung