JavaServer Pages

Seminar Webprogrammierung WS04/05

Timothy Burk

6. Juni 2005

Einleitung

Ansätze für dynamische Websites

- Verschiedene CGI-Lösungen (Perl, PHP, ...)
- Eigener Prozess für jede einzelne Anfrage nötig
- Skripte müssen bei jeder Anfrage neu interpretiert werden

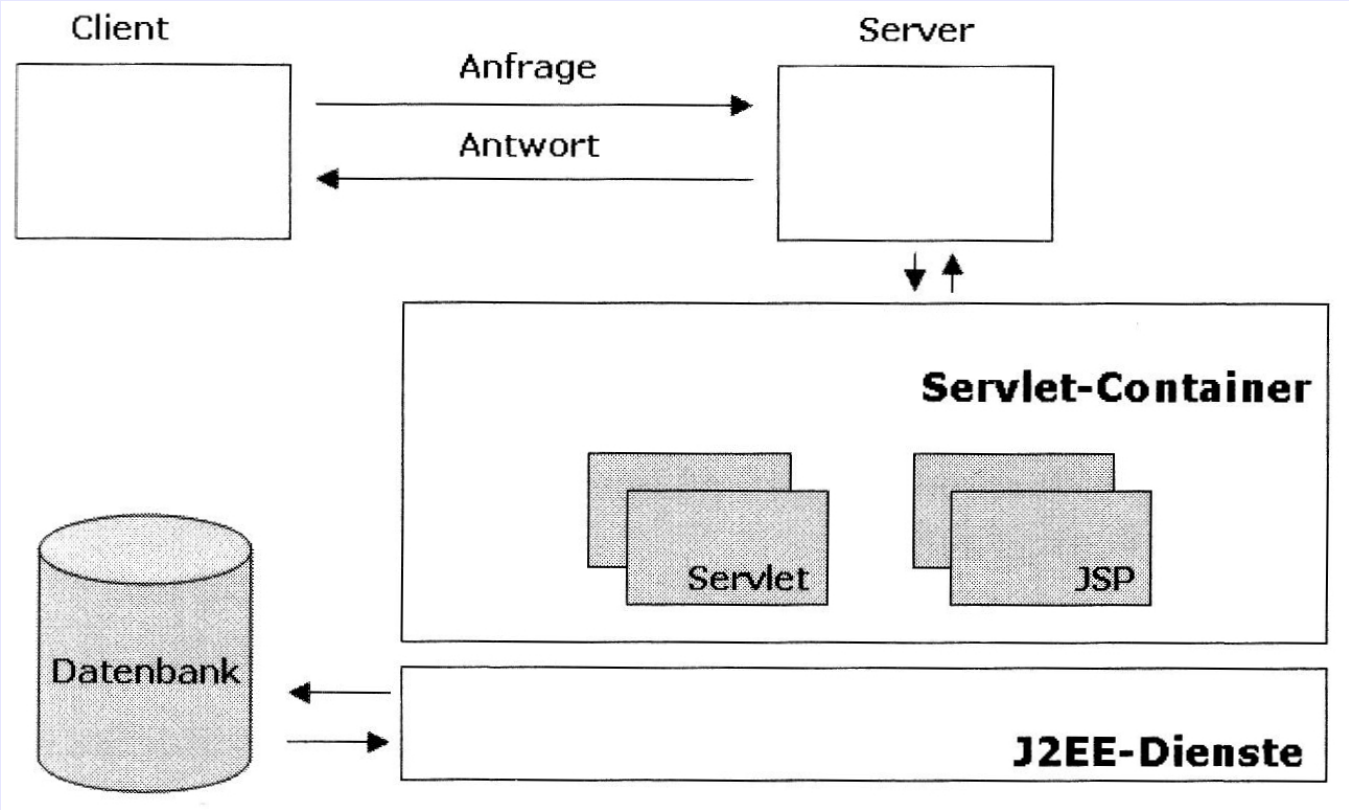
$\Rightarrow$ Die Lösung: Webserver-APIs
  - z.B. ISAPI (MS IIS) oder Apache-API
  - Vorteil: Webanwendungen laufen im Prozess des Webservers ab
  - Nachteil: plattformabhängig

Java-Servlets

- Sun Microsystems führt 1996 eine plattformunabhängige Lösung auf Java-Basis ein: **Java-Servlets**
- Java-Klassen, die im Kontext des Webservers auf einer JVM verarbeitet werden
- JVM arbeitet Anfragen multithreaded ab
- Servlets müssen von `GenericServlet` oder `HTTPServlet` abgeleitet werden

Servlet-Container und Prozessraum

- Servlets werden innerhalb eines Servlet-Containers ausgeführt



Problem: Ausgabe von Webseiten mit Servlets

- HTML-Tags werden mit `out.print(...)` an den Ausgabestrom geschickt
  $\longrightarrow$ keine Trennung von Seitendesign und Anwendungslogik
  $\longrightarrow$ unübersichtlicher Quellcode

- Lösungsansatz: Einfache Skriptelemente und HTML-Code mischen (z.B. PHP, ASP)

$\Rightarrow$ JavaServer Pages

JavaServer Pages

Was sind JavaServer Pages?

- „Technologie zur Erzeugung von HTML- und XML-Ausgaben eines Webservers“ (Wikipedia)
- Sun Microsystems, 1998
- Statische Inhalte und Java-/JSP- Codefragmente in einer Datei
- Von Java gestützte Erzeugung dynamischer Webseiten

Format

Eine JSP ist eine Quelltextdatei für Webseiten

- Standard- oder JSP-Syntax
- XML-Syntax
  (Beispiele folgen)

Eigenschaften von JavaServer Pages

- Enthält JSP-Anweisungen und HTML-Tags
- HTML-Tags werden unverändert an den Client geschickt
- JSP-Anweisungen geben dynamische Inhalte zurück
- Anwendungslogik ist in Tag-Bibliotheken oder JavaBeans implementiert
- Gleichzeitig kann Java-Code auch direkt in der JSP notiert werden

Grundgerüst einer JSP

Diese JSP gibt die aktuelle Uhrzeit aus.

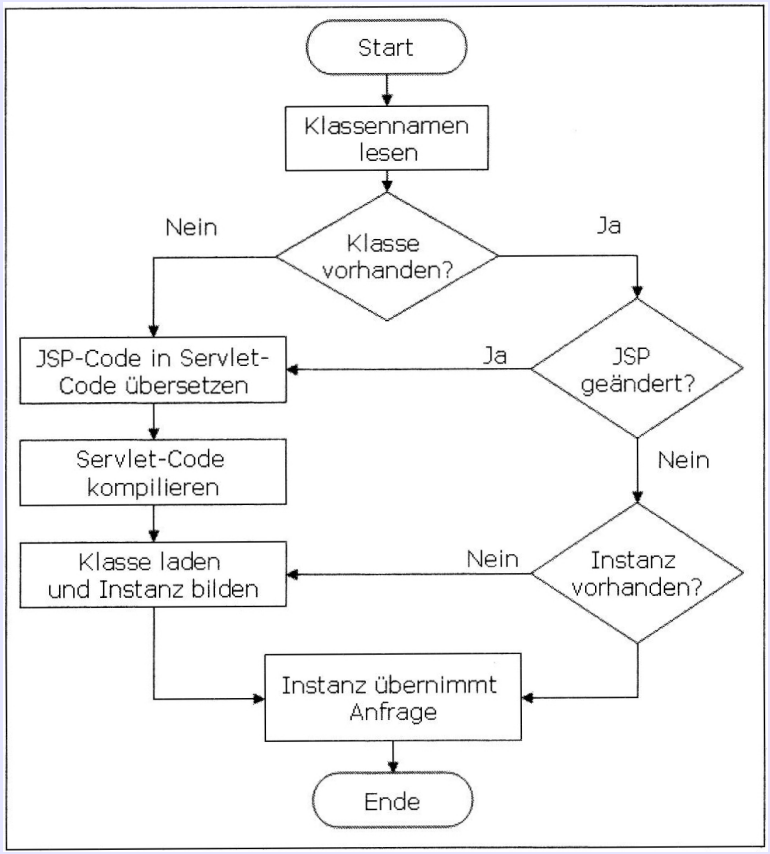
```
<%@ page contentType="text/html" %>
<%@ page import="java.util.*" %>
<html>
  <head>
    <title>Datum und Uhrzeit</title>
  </head>
  <body>
    <jsp:useBean id="now" class="GregorianCalendar"/>
    Hallo! Wir haben heute
    <%= now.time %>
  </body>
</html>
```

Übersetzungs- und Ausführungsphase (1)

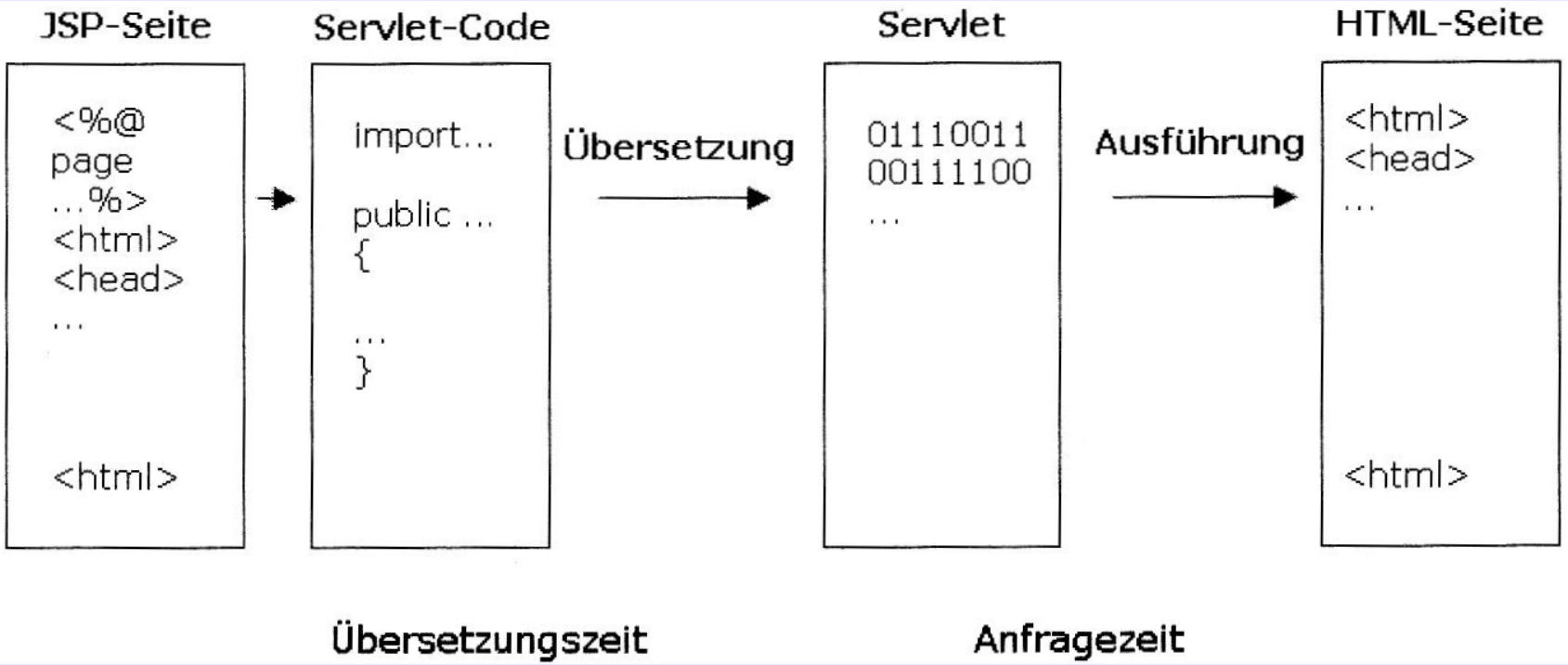
Was passiert genau, wenn ein Client eine JSP von einem Webserver anfragt?

- Der Server reicht die Anfrage an den Servlet-/JSP-Container weiter
- Der Container arbeitet mit der sogenannten JSP-Implementierungsklasse

Übersetzungs- und Ausführungsphase (2)



Übersetzungs- und Ausführungsphase (3)



JSP-Elemente

- **Direktiven:** Anweisungen an den JSP-Container
  `<%@ page contentType="..." session="true" %>`
- **Skriptelemente:** Java-Code
  → Drei Subkategorien
- **Aktionen:** Vordefinierte Funktionen
  `<jsp:useBean id="age" class="AgeBean"/>`
- **Expression Language:** EL-Ausdrücke haben die gleiche Funktion wie Skriptausdrücke, aber eine vereinfachte Syntax
  `${name}`

Skriptelemente

- **Deklarationen** legen Variablen, Methoden oder innere Java-Klassen fest

```
<%! int x = 0; %>
```

- **Skriptlets** enthalten Java-Code, der eine Funktionalität implementiert

```
<jsp:scriptlet>
String myname = (String)session.getAttribute("myname");
String myemail = (String)session.getAttribute("myemail"
</jsp:scriptlet>
```

- **Ausdrücke** werten Variablen und Operationen zu Zeichenketten aus

```
<%= java.util.Date() %>
```

Erweiterungsmechanismen

Tagbibliotheken (Taglibs)

- Ersetzen von Skriptlets durch vordefinierte Aktionen, die in Taglibs zusammengefasst sind
- JSP Standard Tag Library (JSTL) hält Aktionen für häufig benötigte Aufgabe bereit
- Benutzerdefinierte Tags und Taglibs ermöglichen vielfältige Ergänzungen zur JSTL
- Definition erfolgt zeitgemäß in XML

JavaBeans

- JavaBeans können über JSP-Aktionen geladen und benutzt werden
- Beans stellen den mächtigsten Erweiterungsmechanismus für JSP dar

Beispiel

index.jsp

```
<jsp:root version="2.0" xmlns:jsp="http://java.sun.com/JSP/Page">
<jsp:directive.page contentType="text/html; ISO-8859-1" session="true"/>
  <html>
...
    <body>
      Datum: <jsp:expression>new java.util.Date()</jsp:expression>
...
      <h1>Registrierung:</h1>
      <p>Bitte melden Sie sich hier an:</p>
        <form action="employeeList.jsp" method="get">
          Name: <input type="Text" name="myname" value="" size="40"/>
          E-Mail: <input type="Text" name="myemail" value="" size="40"/>
          <input type="Submit" value="Anmelden..."/>
        </form>
    </body>
  </html>
</jsp:root>
```

employeeList.jsp

```
<%@ page contentType="text/html; ISO-8859-1" session="true" %>
<%@ taglib prefix="mysqlquery" uri="/WEB-INF/tlds/mysqlqueries.tld" %>
...
    Sie sind angemeldet als: ${param.myname}, ${param.myemail}
    <jsp:scriptlet>
      String myname = request.getParameter("myname");
      String myemail = request.getParameter("myemail");
      session.setAttribute("myname", myname);
      session.setAttribute("myemail", myemail);
    </jsp:scriptlet>
    <h1>Liste aller Mitarbeiter</h1>
...
      <mysqlquery:employeelist>
      <tr>
        <td>${name}</td><td>${vorname}</td>
        <td><a href="employeeDetails.jsp?mnr=${mnr}">Details anzeigen</a></td>
      </tr>
      </mysqlquery:employeelist>
...
```

mysqlqueries.tld

```
<?xml version="1.0" encoding="UTF-8" ?>
<taglib 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/web-jsptaglibrary_2_0.xsd"
    version="2.0">
  <description>Bibliothek MySQL-Abfragen</description>
    <tlib-version>1.0</tlib-version>
    <short-name>mysqlquery</short-name>
    <uri>/MySQLQueryLibrary</uri>

    <tag>
     <description>Liste aller Mitarbeiter ausgeben</description>
        <name>employeelist</name>
     <tag-class>com.tburk.studium.EmployeeList</tag-class>
     <body-content>scriptless</body-content>
        <variable>
         <name-given>mnr</name-given>
        </variable>
        <variable>
       <name-given>name</name-given>
    </variable>
    <variable>
     <name-given>vorname</name-given>
    </variable>
    </tag>
</taglib>
```

EmployeeList.java

```
public class EmployeeList extends SimpleTagSupport {
  private MysqlConnHandler mch;

  public void doTag() throws JspException, IOException {
    this.mch = new MysqlConnHandler("org.gjt.mm.mysql.Driver", "localhost", "dbproject");
    if(mch.connect()) {
      String sql = "SELECT mnr, mname, vorname FROM Mitarbeiter ORDER BY mname ASC";
      ResultSet rs = mch.mysqlQuery(sql);
      try {
        while(rs.next()) {
          String mnr = rs.getString(1);
          String name = rs.getString(2);
          String vorname = rs.getString(3);
          JspContext context = getJspContext();
          context.setAttribute("mnr", mnr);
          context.setAttribute("name", name);
          context.setAttribute("vorname", vorname);
          JspFragment fragment = getJspBody();
          fragment.invoke(null);
        }
        rs.close();
        mch.close();
      } catch (SQLException e) {
        e.printStackTrace();
      }}}}
```

Fazit

Vor- und Nachteile von JSP

+ JSP ist effizient
+ Java eröffnet als OO-Programmiersprache mehr Möglichkeiten als Skriptsprachen
+ plattformunabhängig
– Servlet-/JSP-Container ist notwendig: Tomcat
– Komplexes Deployment von Webanwendungen