JavaServer Pages

Seminar Medientechnik
Prof. Hußmann
Vortrag: Wilhelm Lechner
4. 7. 2003

Übersicht

1. Einführung
2. Lebenszyklus von JSP Seiten
3. JSP-Sprachelemente
4. Aufruf von Webkomponenten
5. Verwendung von JavaBeans
6. Benutzerdefinierte Tags

Was ist JSP?

- Sprache zum Entwickeln von dyn. Web-Seiten
- Mechanismen zum Erweitern der JSP-Sprache
- Erweiterung von Java Servlets
- Aktuelle Version: 1.2
- Bestandteil von J2EE 1.3
  - javax.servlet.jsp
  - javax.servlet.jsp.tagext

Woraus besteht eine JSP-Seite?

- Statische Template-Daten
  z.B. HTML, WML, XML
- JSP-Tags für dynamische Inhalte

```
<html>
  <body>
    <jsp:useBean id="clock"
                 scope="page"
                 class="java.util.GregorianCalendar" />
    Die aktuelle Zeit ist:
    <jsp:getProperty name="clock" property="time" />
    <br>
    Der Server steht in Zeitzone
    <jsp:getProperty name="clock" property="timeZone" />
  </body>
</html>
```

Vorteile von JSP

- Trennung von Präsentations- und Anwendungslogik
- Leichter zu ändern als Java Servlets
- Wiederverwendbarer Code durch Kapselung von Anwendungslogik in
  - JavaBeans Komponenten
  - benutzerdefinierten Tags

Lebenszyklus von JSP-Seiten (1)

- Vorbereitung: 2 Schritte der Übersetzung
  1. Übersetzung der JSP-Seite in eine Servlet-Klasse
     Name z.B.:
     `index.jsp` ⟶ `index$jsp.java`
  2. Kompilieren der Servlet-Klasse
- Laden und Instantiieren des Servlets
- Ausführen der Methode **`jspInit()`**
  - geeignet für alle einmaligen Aufgaben
  - kann durch eine Deklaration auf der JSP-Seite überschrieben werden

Lebenszyklus von JSP-Seiten (2)

- Bearbeiten der einzelnen Requests
  Ausführen der Methode
  `_jspService (HttpServletRequest req, HttpServletResponse resp)`
- Beendigung
  Ausführen der Methode **`jspDestroy()`**
  – Freigabe von Ressourcen

JSP-Sprachelemente - Schreibweisen

- JSP
  – kompakt, schneller zu tippen
  – Tag-Namen beginnen mit einem Sonderzeichen
- XML Namensraum: **`jsp`**
  – ausführlich
  – für spezielle JSP Editoren und automatische Generatoren

JSP-Sprachelemente - Arten

- Skripte (Scriptlets)
- Direktiven
- Aktionen

JSP-Sprachelemente - Skripte (1)

- Vereinbarungen
  - Variablen, Methoden und innere Klassen
  - Gültigkeitsbereich: ganze JSP-Seite

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

JSP-Sprachelemente - Skripte (2)

- Anweisungsfragmente (Scriptlets)
  - werden direkt in die übersetzte Klasse übernommen
  - keine vollständigen Anweisungen nötig

```
<% if (zahl % 2 == 0) { %>
       gerade Zahl
<% } else { %>
       ungerade Zahl
<% } %>
```


JSP-Sprachelemente - Skripte (3)

- Ausdrücke
  - Wert eines Java-Ausdruckes wird in das Ausgabe-dokument geschrieben

  <%= *ausdruck* %>

  ist äquivalent zu:
  <% out.print(*ausdruck*) %>

- Kommentare

  <%-- ... --%>

JSP-Sprachelemente - Arten

- Skripte
- **Direktiven**
- Aktionen

JSP-Sprachelemente - Direktiven

- Informationen von globalem Charakter
- JSP 1.2 kennt 3 verschiedene Direktiven
  - include
  - page
  - taglib
- Syntax

```
<%@ Direktive [Attribut="Wert"]... %>
```

JSP-Sprachelemente - Direktive "include"

- Einfügen von Dateien mit ausgelagerten wiederkehrenden Textteilen
- vollständiger Inhalt wird zur Übersetzungszeit in die JSP-Seite eingefügt
- Syntax:
  <%@ include file="*Pfadangabe*" %>
  - Attribut "file" muß zur Übersetzungszeit feststehen
  - Datei mit beliebigem Text möglich: JSP, HTML etc.

JSP-Sprachelemente - Direktive "page"

- Steuerung des Übersetzungsprozesses
- 12 Attribute können verwendet werden:
  - language
  - extends
  - import
  - session
  - buffer
  - autoFlush
  - isThreadSafe
  - info
  - errorpage
  - isErrorPage
  - contentType
  - pageEncoding

JSP-Sprachelemente - Aktionen

- Standardaktionen
  - include
  - forward
  - plugin

  in Verbindung mit Java-Beans:
  - useBean
  - setProperty
  - getProperty

- benutzerdefinierte Aktionen

Kommunikation zwischen JSP-Seiten

- Implizite Objekte

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

- JavaBeans

Aufruf von Webkomponenten (1)

- Aktion "include"
  `<jsp:include page="`*`Pfadangabe`*`"`
  `flush="true"/>`
- Einfügen von JSP-Seiten in die aktuelle Seite zur Laufzeit
- Kommunikation mit der aufgerufenen Seite über das Request-Objekt:
  - **`set/getAttribute()`** des Request-Objekts
  - **`<jsp:param name="`*`name`*`" value="`*`wert`*`">`**

Aufruf von Webkomponenten (2)

- Aktion "forward":
  `<jsp:forward page="`*`Pfadangabe`*`"/>`
- Weiterleiten einer Anfrage an eine andere JSP-Seite oder ein Servlet
- bricht die Ausführung der aktuellen Seite ab
- Informationen können über das Request-Objekt an die neue Seite gegeben werden

Aufruf von Webkomponenten (3)

- Aktion "plugin":
  - `<jsp:plugin type="bean|applet" code="Klasse" codebase="Basisverzeichnis"/>`
- vom Browser abhängiger HTML-Code
- 2 Arten von Einbettungen:
  - Applets
  - JavaBeans
- Ausführung in einem Java-Plugin beim Client

JavaBeans in JSP Seiten

- Trennung von visuellem Design und Anwendungslogik
- Wiederverwendbarkeit
- Design-Konventionen
  - Konstruktor ohne Argumente
  - keine öffentlichen Instanzvariablen
  - Zustand wird durch Eigenschaften festgelegt
    - nicht unbedingt Instanzvariablen
    - primitive Typen als auch Klassen
    - einfach und Arrays
    - schreibbar oder lesbar oder beides

Zugriff auf JavaBeans Komponenten

- über öffentliche Instanzmethoden
  Beispiel: Eigenschaft **e** vom Typ **T**
  - Schreiben
    - **`public void setE(T wert)`**
  - Lesen
    - **`public T getE()`**
    - **`public boolean isE()`**
- Ermittlung der Funktionalität durch Introspektion
  - **`java.lang.reflect`**
  - **`java.beans`**

Verwenden von JavaBeans in JSP (1)

```
<jsp:useBean id="beanName" class="KlassenName"
             scope="bereich"/>
```

oder

```
<jsp:useBean id="beanName" class="KlassenName"
                        scope="bereich">
   <jsp:setProperty ...>
</jsp:useBean>
```

Gültigkeitsbereich (scope):

- page
- request
- application
- session

Verwenden von JavaBeans in JSP (2)

- Setzen von Eigenschaften
  - primitive Datentypen oder entsprechende Klassen:

```
<jsp:setProperty name="beanName"
                 property="propName|*"
                [value="wert"]
                [param="paramName"]/>
```

  - andere Klassen (Beispiel):

```
<jsp:setProperty name="myBean" property="locale"
   value="<%=java.util.Locale.GERMAN%>"/>
```

    äquivalent zu:

```
<% myBean.setLocale(java.util.Locale.GERMAN); %>
```


Verwenden von JavaBeans in JSP (3)

- Lesen von Eigenschaften

```
<jsp:getProperty name="beanName"
                 property="propName"/>
```

  äquivalent zu:

```
<%= beanName.getPropName()%>
```

Benutzerdefinierte ([engl.] Custom) Tags

- Kapselung sich wiederholender Funktionalität
- Zusammenfassung in Tag-Bibliotheken
- Einbindung mit "taglib"-Direktive

```
<%@ taglib uri="tag library descriptor"
           prefix="prefix" %>
```

- XML-Syntax

```
<prefix:tagname [attr1="wert"].../>
```

oder

```
<prefix:tagname [attr1="wert"]...>
...
</prefix:tagname>
```


Merkmale von Custom Tags

- Modifizierung durch Attribute
- Zugriff auf alle Objekte einer JSP Seite
- Kommunikation untereinander
- Verschachtelung

Tag Handler Klassen

- Objekte zur Ausführung von Tags
  - Interfaces bzw. Defaultimplementierungen im Paket **`javax.servlet.jsp.tagext`**
- 2 unterschiedliche Arten von Tags
  - einfache Tags:
    - ***`Tag`*** **`(TagSupport)`**
  - Tags, die aus Anfangs- und End-Tag bestehen
    - ***`IterationTag`*** **`(TagSupport)`**
    - ***`BodyTag`*** **`(BodyTagSupport)`**

Tag-Bibliotheks-Deskriptor (TLD) - Struktur

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
    PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
    "http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">
<taglib>
  <tlib-version>1.0</tlib-version>
  <jsp-version>1.2</jsp-version>
  <short-name></short-name>
  <tag>
    ...
  </tag>
</taglib>
```

Einfache Tags

- Handler
  - Interface ***Tag*** (Klasse **TagSupport**):
    - **doStartTag()**
      Rückgabewert **SKIP_BODY**
    - **doEndTag()**
      Rückgabewert **EVAL_PAGE** **Rest der Seite wird abgearbeitet**
      SKIP_PAGE **Rest der Seite wird übersprungen**
- TLD:

```
<tag>
  <name>tagname</name>
  <tag-class>Klassenname</tag-class>
  <body-content>empty</body-content>
  ...
</tag>
```


Body Tags (1)

- keine Interaktion mit dem Body
  - einmalige Auswertung
    Interface ***Tag*** (Klasse **TagSupport**)
    - **doStartTag():**
      Rückgabewert
      - **EVAL_BODY_INCLUDE** Body wird ausgewertet
      - **SKIP_BODY** Body wird übersprungen
  - wiederholte Auswertung
    Interface ***IterationTag*** (Klasse **TagSupport**)
    - **doStartTag()** und **doAfterBody():**
      Rückgabewert **EVAL_BODY_AGAIN**

Body Tags (2)

- Interaktion mit dem Body

  Interface ***BodyTag*** (Klasse **`BodyTagSupport`**)

  – **`doStartTag():`**

  Rückgabewert **`EVAL_BODY_BUFFERED`**

  1. aktueller Ausgabestrom wird gesichert
  2. **`BodyContent`**-Objekt wird erzeugt (neuer Ausgabestrom)
  3. Wert von **`out`** wird auf das **`BodyContent`**-Objekt geändert

  – **`doInitBody():`**

  - wird vor der Auswertung des Bodys aufgerufen
  - kann zur Initialisierung des BodyContents verwendet werden

Body Tags (3)

– **`doAfterBody():`**

- wird nach der Auswertung des Bodys aufgerufen
- gibt an, ob der Body erneut ausgewertet werden soll:
  Rückgabewert **`EVAL_BODY_BUFFERED`** oder **`SKIP_BODY`**

– **`doInitBody() und doAfterBody():`**

- Zugriff auf den Ausgabestrom des Bodys:
  `getBodyContent()` der Klasse **`BodyTagSupport`**
- Kopieren in den übergeordneten Ausgabestrom:
  ***`bodyContent`*`.writeout(getPreviousOut());`**
- Löschen des Pufferinhalts:
  ***`bodyContent`*`.clearBody();`**

Body Tags (4)

- TLD:

```
<tag>
  <name>tagname</name>
  <tag-class>Klassenname</tag-class>
  <body-content>JSP|tagdependent</body-content>
  ...
</tag>
```


Tags mit Attributen

- Handler
  - für jedes Attribut ist eineEigenschaft mit get/set-Methoden (JavaBeans-Konvention) nötig
- TLD

```
<tag>
  <name>tagname</name>
  <tag-class>Klassenname</tag-class>
  <body-content>JSP</body-content>
  <attribute>
    <name>attribut1</name>
    <required>true|false|yes|no</required>
    <rtexprvalue>true|false|yes|no</rtexprvalue>
  </attribute>
</tag>
```

Tags zur Definition von Skript-Variablen (1)

- Name wird normalerweise als Wert eines Attributs angegeben
  - Beispiel:

```
<tt:lookup id="tx" type="UserTransaction">
<% tx.begin %>
```

- Handler
  - Aufruf der get-Methode des Attributs
  - erzeugt eine Variable und weist ihr einen Wert zu
  - speichert Variable und Wert als Attribut eines Scope-Objekts

```
pageContext.setAttribute
    (String name, Object wert[, int scope])
```


Tags zur Definition von Skript-Variablen (2)

- Information über die Skript-Variable
  - TLD:

```
<tag>
    ...
    <variable>
        <name-from-attribute>attributname</name-from-attribute>
        <variable-class>Klassenname</variable-class>
        <declare>true|false</declare>
        <scope>NESTED | AT_BEGIN | AT_END</scope>
    </variable>
    ...
</tag>
```

Tags zur Definition von Skript-Variablen (3)

- Unterklasse von **`TagExtraInfo`**
  - Methode **`getVariableInfo(Tagdata data):`**
    - Holen der Attributwerte des Tags
      `data.getAttributeString("attrname")`
    - Verwenden der Information für
      `new VariableInfo(Name, Klasse, Declare, Scope)`
  - TLD:

```
<tag>
  ...
  <tei-class>
    vollständiger Klassenname
  <tei-class>
  ...
</tag>
```


Kooperation zwischen Tags

- zwischen beliebigen Tags
  - Speichern im **`pageContext`**
- zwischen verschachtelten Tags
  - im untergeordneten Tag
    `static final Tag findAncestorWithClass(von,Klasse)`

Java Standard Tag Library (JSTL)

- enthält Kernfunktionen für viele JSP-Anwendungen (z.B. für Iterationen ...)
- verwendbar auf vielen JSP Containern
- nach Funktion in mehrere Namensräume aufgeteilt
- Framework zur Integration existierender Tag-Bibliotheken
- eigene Sprache zur Formulierung von Ausdrücken

Literatur

- Turau, V.; *Java Server Pages: Dynamische Generierung von Web-Dokumenten* 2. Auflage, Heidelberg: dpunkt-Verlag, 2001
- Turau V., Saleck K., Schmidt M.; *Java Server Pages und J2EE* 2. Auflage, Heidelberg: dpunkt-Verlag, 2001
- Hall, M.; *More Servlets und JavaServer Pages* München: Markt und Technik, 2002
- http://java.sun.com/webservices/docs/1.1/tutorial/doc/ (Kap. 15-17)
- http://forum.java.sun.com/forum.jsp?forum=45
- http://www.jsp-develop.de/
- http://jsptags.com