


# Java Server Pages und Tag Libraries

Praktikum Internet-Werkzeuge  
SoSe 2006


Iryna Kozlova





## Wo stehen wir?





blackboard







Redaktions-  
system






Chatsystem





JSPs und TagLibs 10.05.2006 


## Wie geht es weiter?





JSPs und TagLibs 10.05.2006 

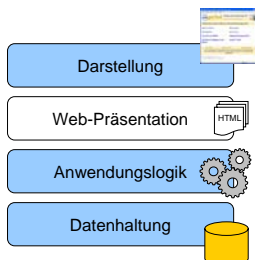
## Was sind Java Server Pages?

- Serverseitige Sprache zur Integration von Java-Code
- Mischen von dynamischem und statischem Inhalt einer HTML-Seite
- Auswertung durch eine JSP-Engine
- Erlaubt die einfache Nutzung von
  - Datenbanken
  - Sessions
  - Cookies
  - XML
  - ...





JSPs und TagLibs 10.05.2006 

## Wozu Java Server Pages?




- In JSPs können beliebige Java-Objekte eingebunden werden (OO-Konzept)
- Java-Anwendungen können so über das WWW verfügbar gemacht werden
- Damit sind JSPs wesentlich ausdrucksmächtiger als PHP-Skripte




JSPs und TagLibs 10.05.2006 

## Wie funktionieren Java Server Pages?

- Textdatei (HTML) mit der Endung **.jsp**
- JSP-Engine bzw. Web-Application-Server (bei uns Tomcat)
- Kompilierung bei erstem Aufruf
- Wird keine "Ausführung" der Seite gewünscht, übergibt man der URL den Parameter **?jsp\_precompile="true"**
- Neu-Kompilierung bei Änderung



JSPs und TagLibs 10.05.2006 

## JSP-Seiten bestehen aus:

- normalem HTML
- Direktiven  
`<%@ Direktive %>`
- Deklarationen (Vereinbarungen)  
`<%! String name = null; %>`
- Anweisungen (Scriptlets)  
`<% if (name != null) { %>`  
    Guten Tag,  
    `<% out.print(name); } %>`
- Ausdrücken  
`<%= "Guten Tag, " + name %>`

prInt

JSPs und TagLibs

10.05.2006



Kein Semikolon!

## JSP-Klasse

- JSP wird kompiliert zu einer Implementation von `javax.servlet.jsp.HttpJspPage`
- Definiert folgende Methoden:
  - `jspInit()` (wird beim ersten Aufruf ausgeführt)
    - kann überschrieben werden, z.B. für Datenbank-Verbindungen
  - `_jspService()` (wird bei jedem Aufruf ausgeführt)
    - darf nicht überschrieben werden, hier landet die ganze Seite!
  - `jspDestroy()` (wird vorm Löschen der JSP ausgeführt)
    - kann überschrieben werden, z.B. zum Lösen der DB-Verbindung oder für andere "Aufräumarbeiten"

prInt

JSPs und TagLibs

10.05.2006



## Implizite Objekte

- Implizite Objekte stehen jeder JSP zur Verfügung, ohne dass sie deklariert werden müssen.
- `request`
- `response`
- `out`
- `session`
- ...

prInt

JSPs und TagLibs

10.05.2006



## Das `request`-Objekt

- `String getParameter(String name)`
- `Enumeration getParameterNames()`
- `String getHeader(String name)`
- `Enumeration getHeaderNames()`
- `Cookies[] getCookies()`
- `HttpSession getSession([boolean create])`

prInt

JSPs und TagLibs

10.05.2006



## Beispiel: request

- Der Aufruf
- `http://www.xyz.de/seite.jsp?name=peter`
- Auf eine Seite `seite.jsp` mit:
  - `<%= request.getParameter("name") %>`
- ergibt
- `peter`

prInt

JSPs und TagLibs

10.05.2006



## Das `response`-Objekt

- `setHeader(String name, String wert)`
- `addCookie(Cookie c)`
- Cookie-Objekte kann man wie folgt erstellen:
  - `Cookie c = new Cookie("Username", "Peter");`
  - `c.setMaxAge(int sekunden)`

prInt

JSPs und TagLibs

10.05.2006



## Direktiven

- Include-Direktive zum Einbinden einer weiteren Datei:  
`<%@ include file="relativerPfad" %>`
- Page-Direktive zum Setzen von Eigenschaften der JSP  
`<%@ page extends="com.Oberklasse"
import="java.lang.*; java.sql.*"
isErrorPage="true"
%>`
- Taglib-Direktive zum Einbinden einer Tag-Bibliothek  
`<%@ taglib
uri="http://www.bla.de/myTags.tld"
prefix="myTags"
%>`

prInt

JSPs und TagLibs

10.05.2006



## Kommentare

- 3 verschiedene Arten:
- Normale HTML-Kommentare (dürfen Ausdrücke enthalten), beim Client sichtbar  
`<!-- Kommentar von <%= a.getName() %> -->`
- JSP-Kommentare, beim Client unsichtbar:  
`<%-- Kommentar --%>`
- Java-Kommentare im JSP-Code, beim Client unsichtbar:  
`<% ... /* Kommentar */ ... %>`

prInt

JSPs und TagLibs

10.05.2006



## Tag Libraries

- Definition eigener Tags (Marken) für HTML, die durch die JSP-Engine verarbeitet werden
- Trennung von Code und Design
- Integration in HTML-Werkzeuge

```
<html>
<body>
  <mopo:header rubrik="sport" />
  ...
  <mopo:footer />
</body>
</html>
```

prInt

JSPs und TagLibs

10.05.2006



## Wie binde ich TagLibs ein?

- TagLibs in vier Schritten:

  1. Einbinden der TagLib-Direktive in der JSP
  2. Anpassen des Application-Deskriptors web.xml
  3. In WEB-INF/ den TagLib-Deskriptor einfügen myTags.tld
  4. In WEB-INF/classes die entsprechenden Klassen bereitstellen

prInt

JSPs und TagLibs

10.05.2006



## Schritt 1: Direktive

- Taglib-Direktive: Taglib  $\longrightarrow$  Namespace

```
<%@ taglib
uri="http://jakarta.apache.org/taglibs/
myTags" prefix="myPrefix" %>
```

- uri als Identifier
- prefix gibt den Namespace an

```
<myPrefix>Hello surname="Baier">
</myPrefix>Hello>
```

prInt

JSPs und TagLibs

10.05.2006



## Schritt 2: web.xml

- Der uri wird ein Deskriptor zugewiesen
- Konfiguration der Tags nicht in web.xml
- extra Deskriptor-Datei .tld

```
<taglib>
  <taglib-uri>
    http://jakarta.apache.org/taglibs/myTags
  </taglib-uri>
  <taglib-location>/WEB-INF/myTags.tld
</taglib-location>
</taglib>
```

prInt

JSPs und TagLibs

10.05.2006



## Schritt 3: Deskriptor-Datei .tld

- Zuweisung von Tags zu Tag-Handler-Klassen
- Definition der Attribute
 

```
<tag>
  <name>Hello</name>
  <tagclass>bsp.PersonalHello</tagclass>
  <bodycontent>empty</bodycontent>
  <attribute>
    <name>surname</name>
    <required>yes</required>
  </attribute>
</tag>
```

prInt

JSPs und TagLibs

10.05.2006



## Schritt 4: Tag-Handler-Klassen

- implementieren das Interface
 

```
javax.servlet.jsp.tagext.Tag
```
- entweder in /WEB-INF/classes oder als Java-Archive (.jar) in /WEB-INF/lib
 

```
public int doStartTag() throws JspException {
  try {
    pageContext.getOut().print("Hello World");
  } catch (IOException ex) {
    throw new JspException("IO problems");
  }
  return SKIP_BODY;
}
```

prInt

JSPs und TagLibs

10.05.2006



## JavaBeans

- Beans sind Komponenten
- Programmierkonvention
- Leerer Konstruktor
- Alle Attribute private
- Public `getMyAttribute()` und `setMyAttribute()`
- automatisch "implizites Objekt"



prInt

JSPs und TagLibs

10.05.2006



## Beans in JSPs

```
<jsp:useBean
  id="zaehler"
  scope="session"
  class="myCounter" />
▪ Zugriff mit
<jsp:getProperty name="zaehler"
  property="anzahl" />
<jsp:setProperty name="zaehler"
  property="anzahl" param="2" />
▪ oder
<% zaehler.getZaehler(); %>
<% zaehler.setZaehler(2); %>
```

prInt

JSPs und TagLibs

10.05.2006



## CLASSPATH einer JSP

- um Beans oder andere Klassen in einer JSP benutzen zu können:
  - `<%@ page import="paketname.*" %>`
  - `jsp/WEB-INF/classes/paketname...`

prInt

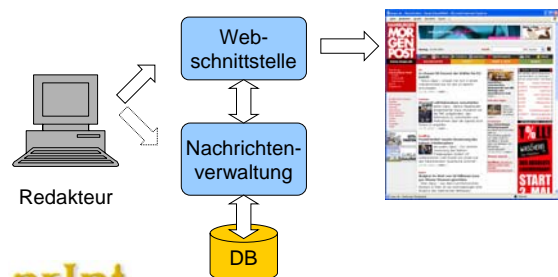
JSPs und TagLibs

10.05.2006



## Was ist ein Redaktionssystem?

- System zum Erstellen, Verwalten und Präsentieren von Nachrichten / Artikeln



prInt

JSPs und TagLibs

10.05.2006



## Eure Aufgabe

- Schreibt eine JavaBean für einen einzelnen Artikel und eine JavaBean für die Artikelliste.
- Benutzt diese Beans in einer JSP, die einen Artikel in eine Liste einfügt.
- Schreibt eine Anzeigeseite, welche die Liste der Artikel anzeigt
- Bonusaufgaben:
  - Leitet die Nutzer nach dem Einfügen auf die Listenseite (1 Punkt)
  - Schreibt eine eigene TagLib für Euren Seitenkopf (2 Punkte)

print

JSPs und TagLibs

10.05.2006

