Praktikum DBAE / WI

Tag Libraries (TagLibs)

Pascal Reuss, M.Sc.

Raum A08b Spl.
Email: reusspa@uni-hildesheim.de

Intelligente
Informationssysteme

Stiftung Universität Hildesheim
2003

Tag Libraries Übersicht

- JSP (und auch Servlets)
  - Verwendung von Markup (Tags)
  - Kenntnisse in Java nötig
  - Portierung des Codes oft nur schwer möglich
- Tag-Bibliothek
  - Definition eigener Tags
  - Kapselung der oft benötigten Funktionalitäten
  - Einfache und verständliche Tags in JSPs
  - Wiederverwendung von Code
  - Leichte Portierung
  - Importieren bestehender Bibliotheken

Tag Libraries – Keine Tags

```
<html>
<head><title>Database Connection direct in JSP</title></head>
<body>
<table>
<%@ page import="java.util.*;" %>
<%@ page import="javax.sql.*;" %>
<%
   String url = „jdbc:postgresql://localhost:5432/database“;
   String user = „testuser“;
   String pw = „testpw“;
   try {
            Class.forName(„org.postgresql.Driver“);
            Connection con = DriverManager.getConnection(url, user, password);
            String sql = „SELECT * FROM testtable“;
            PreparedStatement pstmt = con.prepareStatement(sql);
            ResultSet result = pstmt.executeQuery();

   } catch (ClassNotFoundException ce, SQLException se) {
            System.out.println(ce.getMessage() + se.getMessage());
   }
            //Hier würde noch die Ausgabe des Resultsets hinkommen
%></table></body></html>
```

Tag Libraries – Vordefinierte Tags

- Vordefinierte Tag Libraries
  - JSTL, Regexp, Apache Jakarta Taglibs
- Liefern vordefinierte Funktionalität für JSPs in Tags
  - Schleifen, Fallunterscheidung
  - SQL
  - Mailversand
  - Eingabeprüfungen
  - Spracheinstellungen
  - ……

Tag Libraries - JSTL

- Standard Bibliothek mit definierten Tags
  - Behandeln von Bedingungen
  - Schleifen
  - Manipulation von XML-Dokumenten
  - DB-Zugriffe
  - Gebräuchliche Funktionen
  - Formatierung von Zahlen und Daten

Tag Libraries - JSTL

- Verschiedene Bibliotheken für die Aufgaben:
  - Core: uri → http://java.sun.com/jsp/jstl/core
    - Variablen setzen und wiedergeben
    - Schleifen
  - XML: uri → http://java.sun.com/jsp/jstl/xml
  - Internationalization: uri → http://java.sun.com/jsp/jstl/fmt
  - SQL: uri → http://java.sun.com/jsp/jstl/sql
    - DB-Verbindung
    - Anfragen stellen
  - Functions: uri → http://java.sun.com/jsp/jstl/functions

Tag Libraries – Vordefinierte Tags

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head><title>Database Connection with JSTL in JSP</title></head>
<body>
<table>
<sql:setDataSource var=„db“ driver=„org.postgresql.Driver“
  url=„jdbc:postgresql://localhost:5432/testdatabase“ user=„testuser“ password=„testpw“ />
```

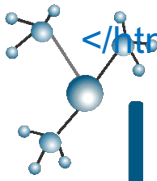
Variante 1:

```
<sql:query dataSource=„${db}“ sql=„SELECT * FROM testtable“ var=„result“ />
```

Variante 2:

```
<sql:query dataSource=„${db}“ var=„result“ >SELECT * FROM testtable</sql:query>

<c:foreach var=„row“ items=„${result.rows}“>
            <tr><td>…..</td></tr>
</c:foreach>
</table>
</body>
</html>
```



Tag Libraries – Eigene Tags

- Eigene Tags (custom tags)
  - Für spezielle Funktionalität können eigene Tags definiert werden
  - TagHandler: Java Klasse mit Funktionalität
  - Tag Library Descriptor (definiert den Tag und verbindet ihn mit der Funktionalität)

Tag Libraries – Eigene Tags

- Tag-Handler
  - Java-Klasse für die Funktionalität des Tags
  - Benötigtes Package: `javax.servlet.jsp.tagext`
  - Typischerweise eine Erweiterung von `TagSupport` oder `BodyTagSupport`
    - Beide Klassen enthalten die zu implementierenden Methoden
    - In der Klasse steht dann die Variable `pageContext` (wie in einer JSP-Datei) zur Verfügung
  - Diese Klasse muss in einem Package (nicht den default Package) definiert werden

- Tag Library Descriptor (TLD)
  - Konfiguration des Tag-Handlers
  - Analog zu dem Web Deployment Descriptor (`web.xml`) für Servlets

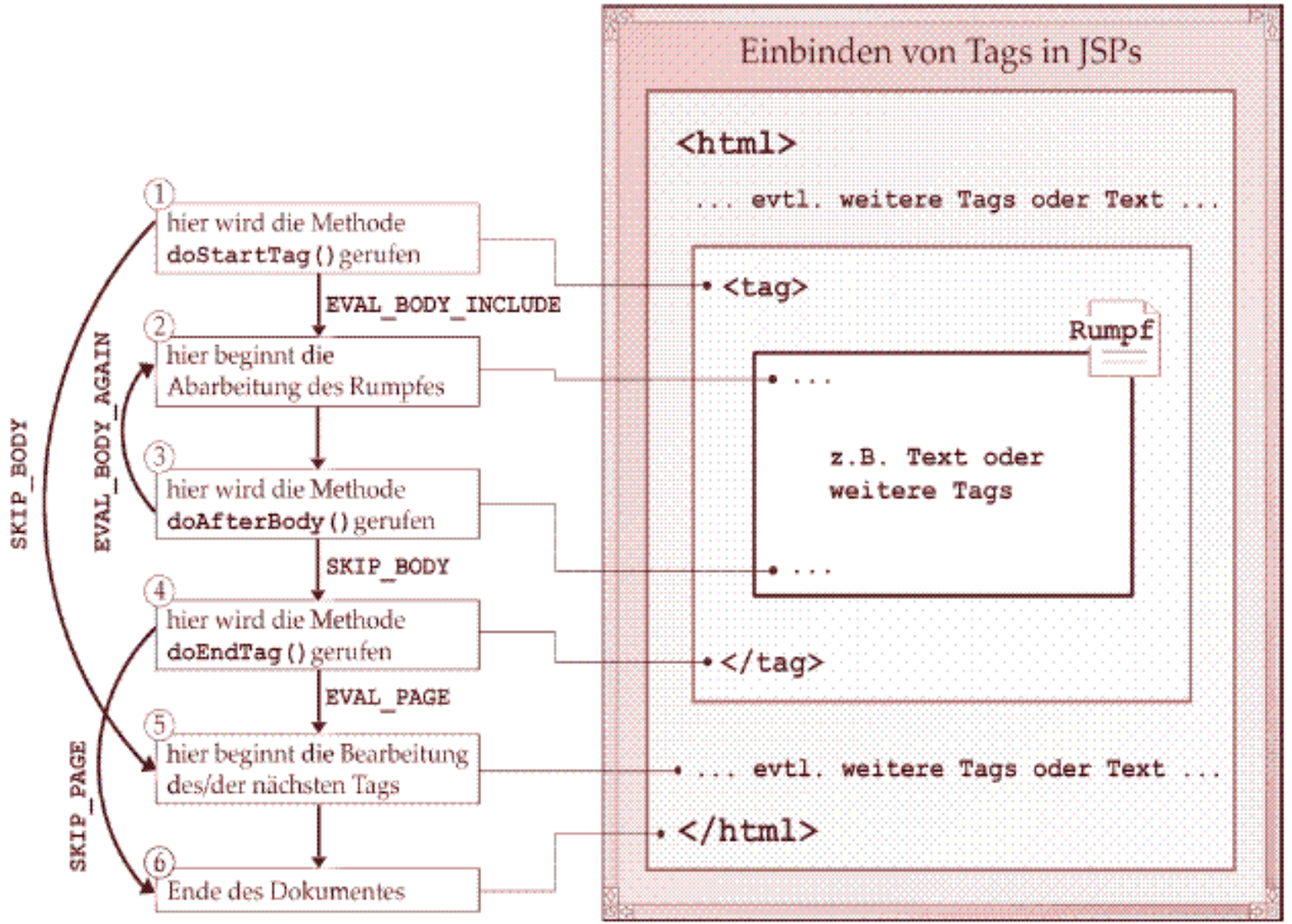
Tag Libraries – Eigene Tags

- Lebenszyklus eines Tags vom Typ `TagSupport` (analog zu dem Lebenszyklus eines Servlets)
  - `doStartTag()`
    - Aufruf der Methode bei einem öffnenden Tag `<meinTag>`
    - Initialisierung und Überprüfungen
    - Übergabe eines Parameters am Ende der Methode, der entscheidet, ob der Rumpf bearbeitet werden soll.
  - doAfterBody()
    - Aufruf der Methode unmittelbar vor dem schließenden Tag
    - Modifizierungen des Rumpfs
    - Mehrmalige Bearbeitung des Tags (nach `doStartTag()`)
    - Wird nur ausgeführt, wenn der Tag einen Rumpf besitzt

Tag Libraries – Eigene Tags

- Lebenszyklus
  - `doEndTag()`
    - Abschluss des Tags (`</meinTag>`), falls die Bearbeitung des Tags nicht am Ende von `doStartTag()` abgebrochen wurde
    - Freigabe eventuell benötigter Ressourcen
    - Speicherung des Zustands des Tags

Tag Libraries – Eigene Tags



Quelle: Das J2EE Codebook

Tag Library – Eigene Tags

- Erforderliche Eingaben
  - Namensraum (namespace) der Bibliothek
    - notwendig, weil verschiedene Bibliotheken mit den gleichen Namen für ein Tag verwenden können
    - Reservierte Namensräume (nicht verwendbar)
      - java, javax, jsp, jspx, servlet, sun, sunx
  - Tag-Mapping
    - Name des Tags
    - Implementierende Klasse (Tag Handler)
    - Liste der Attributen
      - Name
      - Angabe, ob das Attribut notwendig oder nicht

Tag Libraries – Eigene Tags

- Attribute für eigene Tags
  - Name: eindeutige Bezeichnung
  - Required: Optional oder nicht
  - Type: Java-Klassen-Typ (String, Int, …), String ist Standard
  - Description: Beschreibung
  - Fragement: Attribut soll wie ein JSP Fragment behandelt werden

Tag Libraries - Eigene Tags

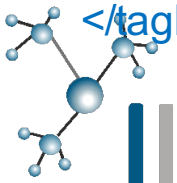
```
public class DatabaseConnection extends SimpleTagSupport {
        String url = "jdbc:postgresql://localhost:5432/testdatabase“;
        String user = „testuser";
        String password = „testpw";

        public int doStartTag() {
                try {
                  Class.forName(„org.postgresql.Driver“);
                  Connection con = DriverManager.getConnection( url, user, password );
                  String sql = „SELECT * FROM testtable“;
                  PreparedStatement stmt = con.prepareStatement(sql);
                  ResultSet rs=pstmt.executeQuery();

                  JSPWriter out = pageContext().getOut();
                  while(rs.next()){
                                  out.println(„<tr><td>“ + rs.getString(1) + „</td></tr>“);
                  }
                } catch (IOException ie) {
                }
    }
}
```

Tag Libraries – Eigene Tags

```
<taglib>
  <tlib-version>1.0</tlib-version>
  <short-name>Database TLDs</short-name>
  <tag>
        <name>dbquery</name>
        <tag-class>customtags.DatabaseConnection</tag-class>
        <body-content>empty</body-content>
  </tag>
  <tag>
        <name>dbselect</name>
        <tag-class>customtags.DatabaseSelection</tag-class>
        <body-content>empty</body-content>
        <attribute>
              <name>database</name>
              <required>true</required>
        </attribute
  </tag>
</taglib>
```



Tag Libraries – Eigene Tags

```
<%@ taglib prefix:db uri= "WEB-INF/custom.tld" %>
<html>
<head><title>Database Connection with JSTL in JSP</title></head>
<body>
<table>
<db:dbquery />
<db:dbselect database=„testdb2“ />
</table>
</body>
</html>
```

Referenzen

K. Samaschke, T. Stark
Das J2EE Codebook