# DEZVOLTAREA APLICATIILOR WEB

# CURS 6

*Lect. Univ. Dr. Mihai Stancu*

DEPARTAMENTUL DE INFORMATICĂ

- suport (The Java EE 5Tutorial)
  - Capitolul 8 – CustomTags in JSP Pages

## Ce este un Tag Custom?

- un element al limbajului JSP definit de utilizator
  - cand pagina JSP ce contine un tag custom se transforma in servlet, acesta se transforma in clasa de tip TagHandler ce poate realiza anumite operatii
  - La executia servletului, containerul web invoca aceste operatii

## Ce este un Tag Custom?

- Tagurile custom au diverse intrebuintari
  - particularizare prin pasarea de atribute de catre pagina apelanta
  - transmiterea de variabile inapoi catre pagina apelanta
  - accesarea obiectelor disponibile in pagina JSP
  - comunicarea intre diverse taguri: crearea si initializarea componentelor JavaBeans, crearea unei variabile publice EL ce refera un bean intr-un tag si folosirea ei in alt tag
  - imbricarea tagurilor si comunicarea prin variabile private

# Tipuri de taguri

- Taguri cu atribute
  - Atribute simple
    - atributele simple sunt evaluate de catre container inainte de a fi pasate catre tag handler

```
<sc:catalog bookDB ="${bookDB}" color="#cccccc">
```

  - Atribute fragment
    - un fragment JSP este o portiune de cod JSP pasat catre tag handler care poate fi invocat de cate ori este nevoie

```
<sc:catalog bookDB ="${bookDB}" color="#cccccc">
  <jsp:attribute name="normalPrice">
    <fmt:formatNumber value="${price}" type="currency"/>
  </jsp:attribute>
  <jsp:attribute name="onSale">
    <strike><fmt:formatNumber value="${price}" type="currency"/></strike><br/>
    <font color="red"><fmt:formatNumber value="${salePrice}" type="currency"/></font>
  </jsp:attribute>
</sc:catalog>
```

  - Atribute dinamice
    - Un atribut dinamic este un atribut care nu este specificat in definitia unui tag

```
<colored:colored color1="red" color2="yellow" color3="blue"/>
```

## Tipuri de taguri

- Taguri cu continut
  - pot contine: taguri core sau custom, text HTML si continut dependent de tag cuprins intre tagul de start si tagul de end

```
<c:if test="${param.Clear}">
     <font color="#ff0000"
     size="+2"><strong>
     You just cleared your shopping cart!
     </strong><br> <br></font>
</c:if>
```

- Taguri care definesc variabile
  - definesc o variabila EL care poate fi folosita in pagina apelanta

```
<tlt:iterator var="departmentName" type="java.lang.String"
group="${myorg.departmentNames}">
     <tr>
     <td><a href="list.jsp?deptName=${departmentName}">
     ${departmentName}</a></td>
     </tr>
</tlt:iterator>
```

- Comunicarea intre taguri
  - comunica intre ele prin intermediul obiectelor partajate
    - publice

```
<c:set var="aVariable" value="aValue" />
<tt:anotherTag attr1="${aVariable}" />
```

    - private

```
<tt:outerTag>
          <tt:innerTag />
</tt:outerTag>
```

## Reutilizarea continutului prin fisiere de taguri

- Un fisier de tag este un fisier sursa care contine un fragment de cod JSP ce poate fi refolosit ca tag custom
- fisier cu extensia .tag – transformat in tag handler si apoi compilat

```
<%@ attribute name="greeting" required="true" %>
<%@ attribute name="name" required="true" %>
<h2><font color="black">${greeting}, ${name}!</font></h2>
```

```
<%@ taglib tagdir="/WEB-INF/tags" prefix="h" %>
...
<html><head><title>Hello</title></head>
<body bgcolor="white">
     <c:set var="greeting" value="Hello" />
     <h2>${greeting}, my name is Duke. What's yours?</h2>
     <form method="get">
          <input type="text" name="username" size="25">
          <p></p>
          <input type="submit" value="Submit">
          <input type="reset" value="Reset">
     </form>
     <c:if test="${fn:length(param.username) > 0}" >
          <h:response greeting="${greeting}" name="${param.username}"/>
     </c:if>
</body></html>
```

- Directive de fisiere de taguri
  - folosite pentru a controla aspecte ale transformarii tagurilor custom in TagHandlere, aspecte legate de taguri, atribute sau variabile expuse
    - declararea tagurilor
    - atributul body-content
    - declararea atributelor de taguri in fisiere de taguri
    - declararea variabilelor de taguri in fisiere de taguri

## DESCRIPTORUL LIBRARIEI DE TAGURI

- Descriptorul unei librarii de taguri (TLD) este un document XML ce contine informatii despre o librarie si despre fiecare tag continut
  - folosite la validarea tagurilor si de catre unelte de dezvoltare
  - trebuie sa aiba extensia ".tld" si trebuie sa existe in
    - directorul /WEB-INF/ sau
    - subdirector din fisierul WAR sau
    - in directorul /META-INF/ sau
    - subdirectorul unei librarii de taguri dintr-un JAR

```
<taglib xsi:schemaLocation="http://java.sun.com/xml/ns/javaee webjsptaglibrary_2_1.xsd"
xmlns="http://java.sun.com/xml/ns/javaee"|
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="2.1">
```

## Programarea unui TagHandler simplu

- pachetul javax.servlet.jsp.tagext
  - interfata SimpleTag
    - clasa SimpleTagSupport
  - metoda doTag() invocata la inchiderea tagului
  - acces la API de comunicare cu pagina JSP (javax.servlet.jsp.JspContext)
- includere TagHandler in aplicatii web
  - /WEB-INF/classes/
  - /WEB-INF/lib/

# Cum este invocat un TagHandler simplu?

- protocol de baza intre servletul generat al paginii JSP si TagHandler

```
ATag t = new ATag();
t.setJSPContext(...);
t.setParent(...);
t.setAttribute1(value1);
t.setAttribute2(value2);
...
t.setJspBody(new JspFragment(...))
t.doTag();
```

- interfata SimpleTag
  - metoda doTag()

```
public HelloWorldSimpleTag extends SimpleTagSupport {
    public void doTag() throws JspException, IOException {
        getJspContext().getOut().write("Hello, world.");
    }
}
```

- Definirea atributelor in TagHandler
  - pentru fiecare atribut, o metoda de tip setter in clasa de tip TagHandler

```
<c:if test="${Clear}">
```

```
public void setTest(boolean test) {
    this.test = test;
}
```

## TagHandler pentru taguri cu atribute

- Validarea atributelor
  - metoda validate() derivata din TagExtraInfo
  - informatiile legate de atribute pasate intr-un obiect de tip TagData

```
<attribute>
    <name>attr1</name>
    <required>true</required>
    <rtexprvalue>true</rtexprvalue>
</attribute>
```

```
public class TwaTEI extends TagExtraInfo {
    public ValidationMessage[] validate(TagData data) {
        Object o = data.getAttribute("attr1");
        if (o != null && o != TagData.REQUEST_TIME_VALUE) {
            if (((String)o).toLowerCase().equals("true") ||
                        ((String)o).toLowerCase().equals("false") )
                return null;
            else
                return new ValidationMessage(data.getId(),
                        "Invalid boolean value.");
        } else
            return null;
    }
}
```

- Definirea atributelor dinamice
  - implementarea metodei setDynamicAttribute () din interfata DynamicAttributes

```
private ArrayList keys = new ArrayList();
private ArrayList values = new ArrayList();

public void setDynamicAttribute(String uri, String localName,
                                Object value ) throws JspException {
    keys.add( localName );
    values.add( value );
}

public void doTag() throws JspException, IOException {
    JspWriter out = getJspContext().getOut();
    for( int i = 0; i < keys.size(); i++ ) {
        String key = (String)keys.get( i );
        Object value = values.get( i );
        out.println( "<li>" + key + " = " + value + "</li>" );
    }
}
```

## TagHandler pentru taguri cu continut

- manipularea continutului prin captarea lui intr-un StringWriter
  - continutul modificat este scris in JspWritter, obiect obtinut prin metoda getOut() din JspContext.

```
public class SimpleWriter extends SimpleTagSupport {
    public void doTag() throws JspException, IOException {
        StringWriter sw = new StringWriter();
        jspBody.invoke(sw);
        jspContext().
            getOut().println(sw.toString().toUpperCase());
    }
}
```

## TagHandler pentru taguri ce definesc variabile

- parametrii IN = atributele tagului
- parametrii OUT, in functie de vizibilitatea in pagina apelanta
  - AT_BEGIN – de la tagul de start pana la primul tag de end sau, daca nu exista, pana la sfarsitul paginii
  - AT_END – de la tagul de end pana la primul tag de end sau, daca nu exista, pana la sfarsitul paginii
  - NESTED – doar intre tagul de start si tagul de end al tagului curent

## TagHandler pentru taguri ce definesc variabile

- variabila definita trebuie setata pe contextul dorit prin metoda JspContext().setAttribute(name, value) sau JspContext.setAttribute(name,value,scope)

```
public void doTag() throws JspException, IOException {
     if (iterator == null)
          return;
     while (iterator.hasNext()) {
          getJspContext().setAttribute(var, iterator.next());
          getJspBody().invoke(null);
     }
}
public void setVar(String var) {
     this.var = var;
}
public void setGroup(Collection group) {
     this.group = group;
     if(group.size() > 0)
          iterator = group.iterator();
}
```

## TagHandler pentru taguri ce definesc variabile

- clasa TagExtraInfo ofera informatii catre TLD despre variabilele definite de taguri
  - extindere: implementarea metodei getVariableInfo() care
    - returneaza un array de obiecte de tipul VariableInfo
    - VariableInfo: numele variabilei, clasa variabilei, daca refera un obiect nou, vizibilitatea variaibilei
    - apelata de containerul web si primeste un parametru de tip javax.servlet.jsp.tagext.TagData
    - TagData: tupluri de atribut-valoare pentru fiecare dintre atributele tagului

- exemplu TagExtraInfo

```
package iterator;
public class IteratorTEI extends TagExtraInfo {
    public VariableInfo[] getVariableInfo(TagData data) {
        String type = data.getAttributeString("type");
        if (type == null)
            type = "java.lang.Object";
        return new VariableInfo[] {
            new VariableInfo(data.getAttributeString("var"),
            type,
            true,
            VariableInfo.NESTED)
        };
    }
}
```

## COOPERAREA INTRE TAGURI

- contextul paginii JSP
  - pageContext.getAttribute(name,scope)
- imbricarea tagurilor
  - SimpleTagSupport.findAncestorWithClass(from,class)

sau

  - SimpleTagSupport.getParent()

- TagHandler ce ilustreaza cele doua metode de cooperare

```
public class QueryTag extends SimpleTagSupport {
    public int doTag() throws JspException {
        String cid = getConnectionId();
        Connection connection;
        if (cid != null) {
            // there is a connection id, use it
            connection =(Connection)pageContext.getAttribute(cid);
        } else {
            ConnectionTag ancestorTag =
                        (ConnectionTag)findAncestorWithClass(this,
                        ConnectionTag.class);
            if (ancestorTag == null) {
                throw new JspTagException("A query without
                            a connection attribute must be nested
                            within a connection tag.");
            }
            connection = ancestorTag.getConnection();
            ...
        }
    }
}
```

- tagul query implementat anterior

```
<tt:connection cid="con01" ... >
    ...
</tt:connection>
<tt:query id="balances" connectionId="con01">
    SELECT account, balance FROM acct_table
        where customer_number = ?
    <tt:param value="${requestScope.custNumber}" />
</tt:query>
<tt:connection ... >
    <tt:query cid="balances">
        SELECT account, balance FROM acct_table
            where customer_number = ?
    <tt:param value="${requestScope.custNumber}" />
    </tt:query>
</tt:connection>
```

- TLD

```
<tag>
    ...
    <attribute>
        <name>connectionId</name>
        <required>false</required>
    </attribute>
</tag>
```

## Cuvinte cheie

- custom
- TLD
- atribut
- TagHandler
- NESTED

## Resurse utile

- http://www.codeproject.com/Articles/31614/JSP-JSTL-Custom-Tag-Library
- http://www.journaldev.com/2099/jsp-custom-tags-example-tutorial