Les balises JSTL

Fred Hémery

IUT Béthune
Département
Réseaux &
Télécommunications

IC3 Application Web — 07/08

Présentation

- La bibliothèque `JSTL` (**Javaserver page Standard Tag Library**) contient un ensemble de balises qui sont généralement utilisées dans les pages jsp par les concepteurs.
- Elle contient un langage d'expression propre (**EL : Expression Language**) qui facilite la manipulation des variables utilisées dans la page JSP
- Elle propose des balises de contrôle de flux (itération, conditionnelles)
- Elle facilite l'internationalisation des pages
- Elle facilite l'accès aux données stockées dans une base de données

Les composants

Fonctions	URI	Préfix
Core	`http://java.sun.com/jstl/core`	c
Traitement XML	`http://java.sun.com/jstl/xml`	x
I18N	`http://java.sun.com/jstl/fmt`	fmt
Accès Base de données (SQL)	`http://java.sun.com/jstl/sql`	sql
Fonctions	`http://java.sun.com/jsp/jstl/functions`	fn

Le langage d'expression (EL)

- Les expressions sont disponibles pour définir des valeurs d'attributs dans les balises.
- Une expression se situe dans

```
${expr}
```

- par exemple

```
<c:if test="${livre.prix <= user.preferences.limiteDepenses}">
 Le Livre ${livre.titre} rentre dans le budget!
</c:if>
```

- un autre exemple

```
<c:forEach var= "produit" items= "${produits}" >
  <c:out value="Prix de ${produit.name} est ${produit.prix}" />
</c:forEach>
```


Le langage d'expression (EL)

- Les expressions peuvent faire référence aux variables qui sont accessibles dans le contexte de la page du jsp (variables accessible dans la méthode service).
- Ces variables peuvent avoir la portée (scope) page, request, session, ou application.
- Le langage d'expression définit des variables
  - `pagescope` : les variables dont la visibilité est la page
  - `requestscope` : les variables dont la visibilité est la requête
  - `sessionScope` : les variables dont la visibilité est la session
  - `applicationScope` : les variables dont la visibilité est l'application
- De même que les paramètres de la requêtes
  - `param` (1ère valeur du paramètre), `paramValues`(un tableau des valeurs d'un paramètre)

Le langage d'expression (EL)

- L'exemple qui suit visualise l'ensemble des valeurs des paramètres d'une requête :

```
<c:forEach var="aParam" items="${paramValues}">
  param: ${aParam.key}" values:
  <c:forEach var="aValue" items="${aParam.value}">
    ${aValue}
  </c:forEach> <br>
</c:forEach>
```

- De la même façon on peut obtenir les caractéristiques de l'en-tête de la requête en utilisant les variables `header` et `headerValues`.
- `initParam` contient les paramètres initiaux de l'application (définis dans le fichier `web.xml`)
- `cookie` contient la liste des cookies associés à la requête

Le langage d'expression (EL)

- L'opérateur "." permet l'accès à une propriété d'un objet, par exemple :

```
Cher ${user.firstName}
De ${user.address.city},
merci pour votre visite de notre site internet.
```

- L'opérateur "`[]`" permet de préciser la clé d'accès à un objet de type `Map`, par exemple :

```
<%--  produitDir  est un objet Map qui contient une description
      des produits, préférences  est un objet Map qui contient
      les préférences d'un user --%>
produit:
  ${produitDir[produit.custId]}
préférence:
  ${user.preferences[couleur]}
```

- Si la propriété d'un objet n'est pas définie, il est possible d'utiliser l'opérateur `default` qui évite de lever une exception.

```
Ville: <c:out value="${user.adresse.ville}" default= "N/A" />
```


Les balises du groupe Core

- `<c:out ...>` : permet d'afficher des données dans une page jsp, équivalent à `<%= ... %>` et de `${el-expression}` par exemple :

```
Vous avez
<c:out value="${sessionScope.user.articleCount}"/> articles.
<c:out value="${customer.adresse.ville}" default="unknown"/>
```

- `<c:set var="essai" value="elexprvalue"/>` : permet de définir une variable dans une page jsp, par exemple :

```
<c:set var="att1" >
  <acme:foo>coucou</acme:foo>
</c:set>
<acme:atag att1="${att1}"/>

<!-- set property in JavaBeans object -->
<c:set target="${cust.address}"
       property="city" value="${city}"/>

<!-- set/add element in Map object -->
<c:set target="${preferences}"
       property="color"value="${param.color}"/>
```

Les balises du groupe Core

- `<c:remove ...>` : permet de supprimer une variable définie dans la page jsp auparavant, par exemple :

```
<c:remove var="cachedResult" scope="application"/>
```

- `<c:catch ...>` : permet de gérer les exceptions dans la page jsp, par exemple :

```
<c:catch var="exception" >
  <!-- Execution we can recover from if exception occurs -->
  ...
</c:catch>

<c:if test= ${exception != null} >
  Sorry. Processing could not be performed because...
</c:if>
```


Les balises du groupe Core

- `<c:if test= ...>` : permet de mettre en place des instructions conditionnelles, par exemple :

```
<c:if test="${user.visitCount == 1}">
This is your first visit. Welcome to the site!
</c:if>
```

- `<c:choose>` : permet de mettre en place des instructions conditionnelles sous forme de choix multiples, par exemple :

```
<c:choose>
 <c:when test="${user.category == 'trial'} >
   ...
 </c:when>
 <c:when test="${user.category == 'member'} >
   ...
 </c:when>
 <c:otherwise>
   ...
 </c:otherwise>
</c:choose>
```


Les balises du groupe Core

- La mise en place d'un `if/then/else`

```
<c:choose>
 <c:when test="${count == 0}" >
   No records matched your selection.
 </c:when>
 <c:otherwise>
  <c:out value="${count}"/>
    records matched your selection.
 </c:otherwise>
</c:choose>
```


Les balises du groupe Core

- Une balise qui réalise une itération : `<c:forEach ...>`, dont voici un exemple :

```
<table>
 <c:forEach var="customer"  items="${customers}" >
  <tr><td><c:out value="${customer}"/></td></tr>
 </c:forEach>
</table>
```

- La valeur de l'attribut `items` peut être un objet de type
  - `Collection` (`List`, `LinkedList`, `ArrayList`, `Vector`, `Stack`, `Set`)
  - `Map` (`HashMap`, `Hashtable`, `Properties`, `Provider`, `Attributes`)
  - `Array`
  - `Iterator`, `Enumeration`
  - Une chaîne de caractères contenant des éléments séparés par une virgule, par exemple :
    `"Monday,Tuesday,Wednesday,Thursday,Friday"`

Les balises du groupe Core

- Si le type de `items` est `Map` alors l'élément obtenu au cours de l'itération doit préciser :
  - `key` : pour obtenir la clé ;
  - `value` : pour obtenir la valeur associée

```
<c:forEach var="entry" items="${myHashtable}">
  Next element is <c:out value="${entry.value}"/>
</c:forEach>
```

- Il est possible d'obtenir l'état de l'itération en utilisant l'attribut `varStatus`.
  - `current` : renvoie l'élément courant
  - `Index` : la valeur de l'index
  - `first` : boolean qui indique si on se trouve sur le premier
  - `count` : qui indique le nombre d'éléments déjà parcourus

Les balises du groupe Core

- Dans l'itération on peut préciser un intervalle de valeurs à parcourir, par exemple :

```
<c:forEach var="i"  begin="100"  end="110" >
 <c:out value="${i}" />
</c:forEach>
```

- On peut faire coopérer des balises, par exemple :

```
<c:forEach var="product" items="${products}" varStatus="status">
  <c:choose>
    <c:when test="${status.count % 2 == 0}">
      item pair
    </c:when>
    <c:otherwise>
      item impair
    </c:otherwise>
  </c:choose>
</c:forEach>
```


Les balises du groupe Core

- `<c:url ...>` est une balise qui permet de spécifier des url encodés, par exemple :

```
<c:url value="http://acme.com/exec/register" var="myUrl">
  <c:param name="name" value="${param.name}"/>
  <c:param name="country" value="${param.country}"/>
</c:url>
<a href='<c:out value="${myUrl}"/>' >Register</a>
```

- `<c:import ...>` est une balise qui permet d'inclure des parties de page en provenance d'autres urls, comme peut le faire `<jsp:include>` par exemple :

```
<c:import url="ftp://ftp.acme.com/README" />
```

- `<c:redirect ...>` est une balise qui permet de rediriger(`forward`) la construction de la réponse vers une autre page par exemple :

```
<c:redirect url="http://acme.com/register"/>
```


Les balises du groupe SQL

- Les balises SQL de JSTL permettent
  - De faire des requêtes SQL
  - D'accéder facilement aux résultats d'une requête
  - De faire des requêtes de modification (update, delete, insert)
  - De grouper plusieurs requêtes dans une transaction (validation des requêtes en fin de transaction uniquement)
- Les balises SQL utilisent pour dialoguer avec la base de données une instance de la classe `DataSource`.
  - La classe `DataSource` permet de gérer un ensemble de connexions en attente d'utilisation. Ce mécanisme permet de réduire le temps de connexion. Si plus aucune connexion n'est disponible, alors la demande est mise en attente.

Les balises du groupe SQL

- Voici un exemple d'utilisation des balises SQL

```
<sql:query var="customers" dataSource="${dataSource}">
  SELECT * FROM customers WHERE country ='China' ORDER BY
  lastname
</sql:query>

<table>
<c:forEach var="row" items="${customers.rows}">
 <tr>
  <td><c:out value="${row.lastName}"/></td>
  <td><c:out value="${row.firstName}"/></td>
  <td><c:out value="${row.address}"/></td>
 </tr>
</c:forEach>
</table>
```


Les balises du groupe SQL

- Voici un autre exemple d'utilisation des balises SQL

```
<table>
 <!-- nom des colonnes -->
<tr>
<c:forEach var= columnName  items= ${result.columnNames} >
  <th><c:out value="${columnName}"/></th>
</c:forEach>

</tr>
<!-- les données  -->
<c:forEach var="row" items="${result.rowsByIndex}">
  <tr>
    <c:forEach var="column" items="${row}">
      <td><c:out value="${column}"/></td>
    </c:forEach>
  </tr>
</c:forEach>
</table>
```


Les balises du groupe SQL

- Les modifications des données d'une base de données peuvent être validées en utilisant une transaction.

```
<sql:transaction dataSource="${dataSource}">
 <sql:update>
    UPDATE account SET Balance = Balance - ?
    WHERE accountNo = ?
   <sql:param value="${transferAmount}"/>
   <sql:param value="${accountFrom}"/>
 </sql:update>

 <sql:update>
    UPDATE account SET Balance = Balance + ?
    WHERE accountNo = ?
 <sql:param value="${transferAmount}"/>
 <sql:param value="${accountTo}"/>
</sql:update>
</sql:transaction>
```


Les balises du groupe SQL

- La balise `<sql:setDataSource ...>` permet de modifier le lien entre l'application web et la base de données utilisée
- Syntaxe

```
<sql:setDataSource
   {dataSource="dataSource" |
    url="jdbcUrl"
    [driver="driverClassName"]
    [user="userName"]
    [password="password"]}
  [var="varName"]
  [scope= {page|request|session|application} ]/>
```

- Un exemple

```
<sql:setDataSource
    url="jdbc:postgresql://iut-gtr2/stock"
    driver="org.postgresql.Driver"
    user="martin"
    password="bdgtr"
  var="stock"
  scope= "application" />
```


Les balises du groupe FMT

- Les balises du groupe FMT permettent d'internationaliser facilement le contenu des pages jsp.
  - Choix du message en fonction du langage demandé par le client
  - Adaptation de l'affichage des formats numériques
  - Adaptation de l'affichage des dates
  - Etc ...

```
<fmt:formatNumber value="123456.7891" pattern="#,#00.0#"/>
```

Par exemple

```
<jsp:useBean id="now" class="java.util.Date" />
 <fmt:formatDate value= ${now}  timeStyle="long" dateStyle="long"/>
```

donnera

```
October 22, 2001 4:05:53 PM PDT
```

aux U.S. et

```
22 octobre 2001 16:05:53 GMT-07:0
```

en France.
Autre exemple

```
<fmt:message key="Welcome" />
```


Les balises du groupe XML

- Les balises du groupe XML sont adaptées au traitement des documents au format XML qui deviennent de plus en plus souvent utilisés dans les applications web et notamment lors d'échanges entre entreprises.
- Voici une liste non exhaustive des balises présentes dans ce groupe
  - `<x:parse>` analyse un document xml
  - `<x:out>` analyse une expression XML(xpath) et l'affiche
  - `<x:set>` fixe une variable
  - `<x:if>`, `<x:choose>`, `<x:when>`, `<x:otherwise>`, `<x:forEach>` des balises de contrôle de flux d'exécution
  - `<x:transform>` permet de fixer des règles de transformation

Les balises du groupe Functions

- Les balises du groupe Functions vont permettre de manipuler des chaînes de caractères.
  - `length` : renvoie la longueur d'une chaîne de caractères
  - `toUpperCase`, `toLowerCase` : transforme une chaîne de caractères en majuscules (minuscules)
  - `substring substringAfter substringBefore` : renvoie une sous-chaîne de caractères
  - `trim` : supprime les espaces inutiles
  - `contains containsIgnoreCase startsWith endsWith indexOf` : recherche d'une sous-chaîne de caractères
  - `split join` : découpe, concatène des chaînes de caractères
  - ...

Utilisation de la bibliothèque de balises standards

- Il faut installer les paquetages `standard.jar` et `jstl.jar` dans le répertoire `lib` de votre application web.
- Il faut déclarer l'utilisation de la librairie de balises standard dans le fichier `web.xml`.

```
<jsp-config>
<taglib>
<taglib-uri>http://java.sun.com/jstl/core</taglib-uri>
<taglib-location>/WEB-INF/tld/c.tld</taglib-location>
</taglib>
</jsp-config>
```

- Il faut ajouter une directive `taglib` dans le fichier `jsp`.

```
<%@ taglib prefix="c" uri="http ://java.sun.com/jstl/core" %>
<html>
<head>
<title>Simple Example</title>
</head>
<body>
<c :set var="browser" value="$header['User-Agent']"/>
<c :out value="$browser"/>
</body>
</html>
```