JSTL : Java Standard Tag Library

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

Plan

Introduction

JSTL : Java Standard Tag Library

- est un composant de la plate-forme JEE
- permet de mieux respecter le modèle MVC
- a comme objectif de ne plus utiliser le langage JAVA dans des pages JSP
- utilise des balises + EL pour remplacer le code JAVA

Introduction

Avantages

- Simplification du code
- Meilleure lisibilité
- Que des balises dans le code
- Maintenance et réutilisation plus facile
- Se protéger des failles XSS

Mise en place

Utiliser une directive JSP

```
<%@ page language="java" contentType="text/html;␣
   charset=UTF-8"     pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
   prefix="c" %>
<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/
       html;␣charset=UTF-8">
    <title>Projet JEE</title>
  </head>
  <body>
    <c:out value="Hello␣World" />
  </body>
</html>
```

Mise en place

Inclure la bibliothèque

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
   prefix="c" %>
```

Afficher un premier message Hello World

```
    <c:out value="Hello␣World" />
```

Mise en place

Inclure la bibliothèque

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
   prefix="c" %>
```

Afficher un premier message Hello World

```
    <c:out value="Hello␣World" />
```

- `prefix="c"` : indique le préfixe à utiliser pour la bibliothèque `core`
- `c:out` : utilisation de ce préfixe pour afficher un message

Mise en place

Inclure la bibliothèque

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
   prefix="c" %>
```

Afficher un premier message Hello World

```
    <c:out value="Hello␣World" />
```

- `prefix="c"` : indique le préfixe à utiliser pour la bibliothèque `core`
- `c:out` : utilisation de ce préfixe pour afficher un message

L'inclusion de la bibliothèque JSTL est toujours signalée en rouge

Mise en place

Solution

- Télécharger la bibliothèque à partir du lien suivant `https://course.oc-static.com/ftp-tutos/cours/java-ee/jstl-1.2.jar`
- Placer la dans le répertoire `lib` situé dans `WEB-INF`

Mise en place

Question

Faudrait-il inclure la bibliothèque JSTL dans nos pages JSP ?

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
   prefix="c" %>
```

Réponse

Non, on peut faire l'auto-chargement.

Mise en place

Démarche

- Créer un fichier JSP dans `WEB-INF` que nous appellerons par exemple `jstlLib.jsp`
- Déplacer les directives JSP dans `jstlLib.jsp`

```
<%@ page language="java" contentType="text/html;
   ␣charset=UTF-8"
     pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/
   core" prefix="c" %>
```

- Configurer l'auto-chargement (autoload) depuis `web.xml`

Mise en place

Configurer l'auto-chargement depuis `web.xml`

```
<jsp-config>
    <jsp-property-group>
        <url-pattern> *.jsp </url-pattern>
        <include-prelude>/WEB-INF/jstlLib.jsp</
           include-prelude>
    </jsp-property-group>
</jsp-config>
```

Mise en place

Configurer l'auto-chargement depuis `web.xml`

```
<jsp-config>
    <jsp-property-group>
        <url-pattern> *.jsp </url-pattern>
        <include-prelude>/WEB-INF/jstlLib.jsp</
          include-prelude>
    </jsp-property-group>
</jsp-config>
```

Explication

- `<url-pattern> *.jsp </url-pattern>` : pour indiquer les fichiers ciblés par l'auto-chargement
- `<include-prelude> /WEB-INF/jstlLib.jsp </include-prelude>` : le chemin du fichier à auto-charger
- Cela nous évite de faire `<%@ include file="/WEB-INF/taglibs.jsp" %>` dans chaque JSP

La bibliothèque JSTL

La JSTL est composée de 5 sous-bibliothèques

- `Core` : pour les principaux de l'algorithmique (déclaration et gestion de variables, les structures conditionnelles et itératives...)
- `Format` : pour le formatage de données et l'internationalisation
- `XML` : pour la manipulation des fichiers XML
- `Function` : pour le traitement des chaînes de caractères
- `SQL` : pour les requêtes SQL

La sous-bibliothèque Core

Afficher une valeur

```
<c:out value="JEE" /> <%-- Affiche JEE --%>
```

La sous-bibliothèque Core

Afficher une valeur

```
<c:out value="JEE" /> <%-- Affiche JEE --%>
```

Afficher une valeur en utilisant EL

```
<c:out value="${ 1 lt 3 and 2 > 1 }" /> <%-- Affiche
    true --%>
```

La sous-bibliothèque Core

Afficher une valeur

```
<c:out value="JEE" /> <%-- Affiche JEE --%>
```

Afficher une valeur en utilisant EL

```
<c:out value="${ 1 lt 3 and 2 > 1 }" /> <%-- Affiche
    true --%>
```

Afficher le contenu d'une variable avec utilisation de valeur par défaut

```
<c:out value="${JEE}" default = "JSTL"/> <%--
   Affiche le contenu de la variable JEE si elle
   existe, sinon affiche JSTL --%>
```

La sous-bibliothèque Core

Afficher une valeur

```
<c:out value="JEE" /> <%-- Affiche JEE --%>
```

Afficher une valeur en utilisant EL

```
<c:out value="${ 1 lt 3 and 2 > 1 }" /> <%-- Affiche
    true --%>
```

Afficher le contenu d'une variable avec utilisation de valeur par défaut

```
<c:out value="${JEE}" default = "JSTL"/> <%--
   Affiche le contenu de la variable JEE si elle
   existe, sinon affiche JSTL --%>
```

Une deuxième utilisation de la valeur par défaut

```
<c:out value="${JEE}" > JSTL </c:out>
```

La sous-bibliothèque `Core`

Pourquoi écrire autant pour afficher une variable ?

- permet d'échapper les caractères spéciaux
- se protéger des failles XSS

La sous-bibliothèque Core

Pourquoi écrire autant pour afficher une variable ?

- permet d'échapper les caractères spéciaux
- se protéger des failles XSS

Exemple

```
<c:out value="<p> Bonjour 'John Wick'. </p>" />
<%-- affiche <p> Bonjour 'John Wick'. </p> --%>
```

La sous-bibliothèque Core

Pourquoi écrire autant pour afficher une variable ?

- permet d'échapper les caractères spéciaux
- se protéger des failles XSS

Exemple

```
<c:out value="<p>␣Bonjour␣'John␣Wick'.␣</p>" />
<%-- affiche <p> Bonjour 'John Wick'. </p> --%>
```

Pour désactiver cette option (excapeXml)

```
<c:out value="<p>␣Bonjour␣'John␣Wick'.␣</p>"
   escapeXml="false" />
<%-- affiche Bonjour 'John Wick'. --%>
```

La sous-bibliothèque Core

Déclarer une variable

```
<c:set var="JEE" value="J'aime la plateforme JEE"
   scope="request" />
```

Explication

- On a déclaré une variable `JEE`
- On l'initialise avec la valeur `J'aime la plateforme JEE`
- On lui affecte la porté `request`

La sous-bibliothèque `Core`

La sous-bibliothèque Core

Créer un objet de type Personne à partir de l'objet `perso` défini dans la servlet et ajouté comme attribut de requête

```
<c:set scope="session" var="p" value="${perso}" />
```

La sous-bibliothèque `Core`

Deux autres attributs sont possibles

- `target` : le nom de l'objet à modifier
- `property` : le nom de la propriété de cet objet qui sera modifié

La sous-bibliothèque Core

Deux autres attributs sont possibles

- `target` : le nom de l'objet à modifier
- `property` : le nom de la propriété de cet objet qui sera modifié

La modification d'un attribut de l'objet

```
<c:set target="${perso}" property="nom" value="
   Travolta" />
<%-- l'objet p aura comme nouveau nom Travolta --%>
<c:out value="${p.nom} ${p.prenom}" />
<%-- affiche Travolta John --%>
```

La sous-bibliothèque Core

Deux autres attributs sont possibles

- `target` : le nom de l'objet à modifier
- `property` : le nom de la propriété de cet objet qui sera modifié

La modification d'un attribut de l'objet

```
<c:set target="${perso}" property="nom" value="
   Travolta" />
<%-- l'objet p aura comme nouveau nom Travolta --%>
<c:out value="${p.nom} ${p.prenom}" />
<%-- affiche Travolta John --%>
```

La suppression d'une variable

```
<c:remove var="JEE" />
<%-- supprime la variable JEE --%>
```

La sous-bibliothèque Core

Les structures conditionnelles sans sinon (else)

```
<c:if test="${ 3 > 2 and 2 > 1 }" >
        c'est facile
</c:if>
<%-- affiche c'est facile car la condition est vraie
    --%>
```

La sous-bibliothèque Core

Les structures conditionnelles sans sinon (else)

```
<c:if test="${ 3 > 2 and 2 > 1 }" >
        c'est facile
</c:if>
<%-- affiche c'est facile car la condition est vraie
    --%>
```

Explication

- L'attribut `test` est obligatoire
- On peut ajouter deux autres attributs optionnels `scope` et `var`
  - `var` : pour stocker le résultat du test
  - `porté` : pour définir la porté de cette variable

La sous-bibliothèque Core

Exemple avec `var` et `scope`

```
<c:if test = "${ 3 > 2 and 2 > 1 }" var = "result"
   scope = "session">
     <c:out value = "${ result }" />
</c:if>
<%-- affiche true --%>
```

La sous-bibliothèque Core

Les structures conditionnelles avec un ou plusieurs sinon (else (if))

```
<c:choose>
    <c:when test = "${ condition }"> resultat </c:
       when>
    ...
    <c:otherwise> resultat par default</c:otherwise>
</c:choose>
```

La sous-bibliothèque Core

Les structures conditionnelles avec un ou plusieurs sinon (else (if))

```
<c:choose>
    <c:when test = "${ condition }"> resultat </c:
       when>
    ...
    <c:otherwise> resultat par default</c:otherwise>
</c:choose>
```

Explication

- c:choose : équivalent de switch
- c:when : équivalent de case dans le switch
- c:otherwise : équivalent de default dans le switch

La sous-bibliothèque Core

Les structures itératives

```
<c:forEach var = "i" begin = "0" end = "10" step = "
   1">
        <c:out value = "${ i }"/>
</c:forEach>
<%-- affiche 0 1 2 3 4 5 6 7 8 9 10 --%>
```

La sous-bibliothèque Core

Les structures itératives

```
<c:forEach var = "i" begin = "0" end = "10" step = "
   1">
        <c:out value = "${ i }"/>
</c:forEach>
<%-- affiche 0 1 2 3 4 5 6 7 8 9 10 --%>
```

Explication

- `var` : n'est pas obligatoire. On l'ajoute quand on a besoin d'utiliser la valeur du compteur
- `begin` : valeur initiale du compteur
- `end` : valeur finale de notre compteur
- `step` : le pas à ajouter au compteur après chaque itération

La sous-bibliothèque Core

Pour parcourir une collection

```
<c:forEach items="${list}" var="element">
    <c:out value="${element['nom']}" />
</c:forEach>
```

La sous-bibliothèque Core

Pour parcourir une collection

```
<c:forEach items="${list}" var="element">
    <c:out value="${element['nom']}" />
</c:forEach>
```

Explication

- `items` : pour définir la liste à parcourir
- `var` : pour récupérer l'élément courant de la liste
- On peut aussi ajouter un attribut `varStatus` pour récupérer des informations sur l'itération courante

La sous-bibliothèque Core

Pour parcourir une collection

```
<c:forEach items="${list}" var="element" varStatus="
   status">
     Element n : <c:out value="${status.count}"/>
        valeur : <c:out value="${element['nom']}" />
</c:forEach>
```

Les différentes propriétés de varStatus

- first : contient true si c'est la première itération
- last : contient true si c'est la dernière itération
- step : contient la valeur de l'attribut step
- count : contient l'indice de l'itération courante (commence de 1) (sinon index commence de 0)

La sous-bibliothèque Core

Pour parcourir une chaîne de caractère

```
<c:forTokens var="sousChaine" items="bonjour, c'est
   John;Wick" delims="; ">
     ${sousChaine}<br/>
</c:forTokens>
```

Explication

- On parcourt une chaîne de caractère par token
- On peut définir un ou plusieurs séparateurs

La sous-bibliothèque Core

Pour ajouter un lien

```
<c:url value="/tapage" var="monLien" />
<a href="${monLien}">lien</a>
```

La sous-bibliothèque Core

Pour ajouter un lien

```
<c:url value="/tapage" var="monLien" />
<a href="${monLien}">lien</a>
```

/tapage est la route d'une servlet définie dans `web.xml`

La sous-bibliothèque Core

Pour ajouter un lien

```
<c:url value="/tapage" var="monLien" />
<a href="${monLien}">lien</a>
```

/tapage est la route d'une servlet définie dans `web.xml`

Pour ajouter un lien avec paramètre

```
<c:url value="/tapage" var="monLien">
  <c:param name="nom" value="Wick"/>
  <c:param name="prenom" value="John"/>
</c:url>
<a href="${monLien}">lien</a>
```

La sous-bibliothèque Core

Pour ajouter un lien

```
<c:url value="/tapage" var="monLien" />
<a href="${monLien}">lien</a>
```

/tapage est la route d'une servlet définie dans `web.xml`

Pour ajouter un lien avec paramètre

```
<c:url value="/tapage" var="monLien">
  <c:param name="nom" value="Wick"/>
  <c:param name="prenom" value="John"/>
</c:url>
<a href="${monLien}">lien</a>
```

Remarque

Pour faire la redirection, le **forwarding** ou l'import, il faut remplacer `c:url` respectivement par `c:redirect`, `c:forward` ou `c:import`.

La sous-bibliothèque Function

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/
   functions" prefix="fn" %>
```

La sous-bibliothèque Function

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/
   functions" prefix="fn" %>
```

Pour récupérer la longueur d'une chaîne de caractère (ou liste)

```
fn:length("chaine")
<%-- Retourne 6 --%>
```

La sous-bibliothèque Function

Pour inclure cette bibliothèque dans le jstlLib.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/
   functions" prefix="fn" %>
```

Pour récupérer la longueur d'une chaîne de caractère (ou liste)

```
fn:length("chaine")
<%-- Retourne 6 --%>
```

Pour tester si une chaîne contient une autre sous-chaîne de caractère

```
fn:contains("Bonjour", "Bon")
<%-- Retourne true --%>
```

La sous-bibliothèque Function

Pour inclure cette bibliothèque dans le jstlLib.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/
   functions" prefix="fn" %>
```

Pour récupérer la longueur d'une chaîne de caractère (ou liste)

```
fn:length("chaine")
<%-- Retourne 6 --%>
```

Pour tester si une chaîne contient une autre sous-chaîne de caractère

```
fn:contains("Bonjour", "Bon")
<%-- Retourne true --%>
```

Pour extraire une sous-chaîne

```
fn:substring("John Wick", 5, 8)
<%-- Retourne Wick --%>
```

La sous-bibliothèque `Function`

Autres fonctions

- `fn:trim(String)` : élimine les espaces au début et à la fin de la chaîne
- `fn:toUpperCase(String)` : retourne la chaîne passée en paramètre en majuscule
- `fn:toLowerCase(String)` : retourne la chaîne passée en paramètre en minuscule
- `fn:escapeXml(String)` : élimine les caractères spéciaux en les remplaçant par leur code HTML (Exemple : `${ fn:escapeXml("Les balises <p> & <b>") }` retourne `"Les balises < p > & < b >"`
- ...

La sous-bibliothèque Format

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt"
   prefix="fmt" %>
```

La sous-bibliothèque Format

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt"
   prefix="fmt" %>
```

Pour convertir une valeur en monnaie

```
<c:set var = "montant" value = "112233.44" />
montant = <fmt:formatNumber value = "${ montant }"
   type="currency"/>
```

```
<%-- Affiche montant = 112 233,44 € --%>
```

La sous-bibliothèque Format

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt"
   prefix="fmt" %>
```

Pour convertir une valeur en monnaie

```
<c:set var = "montant" value = "112233.44" />
montant = <fmt:formatNumber value = "${ montant }"
   type="currency"/>
```

```
<%-- Affiche montant = 112 233,44 € --%>
```

L'attribut `type` peut prendre d'autres valeurs telles que `percent` et `number`

La sous-bibliothèque Format

Quelques autres attributs

```
<c:set var = "montant" value = "112233.44" />
montant = <fmt:formatNumber value ="${ montant }"
   type = "currency"  currencySymbol = "$"
   maxIntegerDigits = "3"/>
<%-- Affiche montant = 233,44 $     --%>
```

La sous-bibliothèque `Format`

Autres attributs

- `groupingUsed` : prend `true` pour préciser si les nombres doivent être groupés, `false` sinon.
- `maxFractionDigits` : indique le nombre maximum de chiffres dans la partie décimale
- `var` : contient le nom de la variable reçoit le résultat
- `scope` : précise la portée de cette variable
- `minIntegerDigits`, `minFractionDigits`...

La sous-bibliothèque Format

Pour convertir en nombre

```
<fmt:parseNumber value = "${ param.id }" var = "id"/
   >
```

Autres attributs

- `integerOnly` : prend `true` pour un résultat de type entier, float si `false`.
- `scope` : précise la portée de cette variable
- ...

La sous-bibliothèque Format

Pour formatter une date

```
<jsp:useBean id = "now" class="java.util.Date" />
Aujourd'hui, c'est le <fmt:formatDate value = "${
   now }" type="date" dateStyle="short"/>
```

Autres valeurs de l'attribut dateStyle

- `long` : remplace l'indice du mois par son nom (janvier, février...)
- `full` : même chose que `long` + le nom du jour (lundi, mardi...)
- Autres valeurs : `medium` et `default`

La sous-bibliothèque `Format`

Autres attributs de `formatDate`

- `timeStyle` : permet de formatter l'heure et prend les mêmes valeurs que `dateStyle`
- `type` : prend une des valeurs suivantes : `date`, `time` ou `both`
- `var` : contient le nom de la variable reçoit le résultat
- `scope` : précise la portée de cette variable
- ...

La sous-bibliothèque `Format`

Autres attributs de `formatDate`

- `timeStyle` : permet de formatter l'heure et prend les mêmes valeurs que `dateStyle`
- `type` : prend une des valeurs suivantes : `date`, `time` ou `both`
- `var` : contient le nom de la variable reçoit le résultat
- `scope` : précise la portée de cette variable
- ...

Il existe également une balise `parseDate` qui permet de convertir en date et qui prend les mêmes attributs que `formatDate`

La sous-bibliothèque Xml

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml"
   prefix="x" %>
```

La sous-bibliothèque Xml

Pour inclure cette bibliothèque dans le `jstlLib.jsp`

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml"
   prefix="x" %>
```

Considérant le fichier XML `personnes.xml` situé dans la racine du projet

```
<personnes>
        <personne id="1">
                <nom>wick</nom>
                <prenom>john</prenom>
        </personne>
        <personne id="2">
                <nom>white</nom>
                <prenom>alain</prenom>
        </personne>
</personnes>
```

La sous-bibliothèque Xml

Pour importer le fichier Xml

```
<c:import url="file:/C:/.../eclipse-workspace/
   TestJstlAtos/personnes.xml" var="personnes" />
```

La sous-bibliothèque Xml

Pour importer le fichier Xml

```
<c:import url="file:/C:/.../eclipse-workspace/
   TestJstlAtos/personnes.xml" var="personnes" />
```

Pour parser le contenu du fichier et l'affecter à une variable

```
<x:parse xml="${personnes}" var="list"  />
```

La sous-bibliothèque Xml

Pour importer le fichier Xml

```
<c:import url="file:/C:/.../eclipse-workspace/
   TestJstlAtos/personnes.xml" var="personnes" />
```

Pour parser le contenu du fichier et l'affecter à une variable

```
<x:parse xml="${personnes}" var="list"  />
```

Pour récupérer une personne de la liste des personnes

```
<x:set var="personne" select="$list/personnes/
   personne[@id=1]" />
```

La sous-bibliothèque Xml

Pour importer le fichier Xml

```
<c:import url="file:/C:/.../eclipse-workspace/
   TestJstlAtos/personnes.xml" var="personnes" />
```

Pour parser le contenu du fichier et l'affecter à une variable

```
<x:parse xml="${personnes}" var="list"  />
```

Pour récupérer une personne de la liste des personnes

```
<x:set var="personne" select="$list/personnes/
   personne[@id=1]" />
```

Pour afficher le contenu de la balise nom

```
<br/> nom = <x:out select="$personne/nom"/>
```

La sous-bibliothèque `Xml`

Remarques

Comme la sous-bibliothèque `core`, la sous-bibliothèque `xml` dispose de balises

- `set` pour déclarer une variable,
- `out` pour afficher,
- `if` et `choose` : pour tester
- `forEach` : pour itérer
- ...

Dépendance JSTL sous Maven

Voici la dépendance `JSTL` à ajouter dans le cas d'un `Maven Project`

```
<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

Dépendance JSTL sous Maven

Voici la dépendance `JSTL` à ajouter dans le cas d'un `Maven Project`

```
<!-- https://mvnrepository.com/artifact/javax.servlet/jstl -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

N'oublions pas de définir un préfixe dans la page JSP

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <body>
    <h2>Hello World!</h2>
    <c:out value="Bonjour" />
  </body>
</html>
```

Et d'activer les `Expression Language`

```
<%@ page isELIgnored="false" %>
```