Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

M2 MIAGE FC / FA

MIAGE LILLE

Nouvelles Technologies et Systèmes d'Informations

JSTL - EL

Loïc Nogues

Octobre 2012

People matter, results count.

Présentation

- Un framework = une librairie de balises de présentation
- JSTL = une librairie standard pour la plupart des fonctionnalités de base d'une application J2EE.
- Sun spécifie les bases de la librairie et laisse l'implémentation libre (en TP on va utiliser l'implémentation de Jakarta de la JSTL)

Les librairies

Librairie	URI	Préfixe
Core	http://java.sun.com/jsp/jstl/core	c
Format	http://java.sun.com/jsp/jstl/fmt	fmt
XML	http://java.sun.com/jsp/jstl/xml	x
SQL	http://java.sun.com/jsp/jstl/sql	sql
Fonctions	http://java.sun.com/jsp/jstl/functions	fn

EL (Expressions Languages)

- accès simple aux beans des différents scopes de l'application web (page, request, session et application)

- Utilisé conjointement avec des librairies de tags, elles permettent de se passer totalement des scriptlets.

- Une expression EL est de la forme suivante : ${ expression }

  ${ object.property }

  ${ object["index property"] }

  ${ object[index] }

  ${ object[0] }

EL – les objets implicites

- pageContext : Accès à l'objet PageContext de la page JSP.
- pageScope : Map permettant d'accéder aux différents attributs du scope 'page'.
- requestScope : Map permettant d'accéder aux différents attributs du scope 'request'.
- sessionScope : Map permettant d'accéder aux différents attributs du scope 'session'.
- applicationScope : Map permettant d'accéder aux différents attributs du scope 'application'.
- param : Map permettant d'accéder aux paramètres de la requête HTTP sous forme de String.
- paramValues : Map permettant d'accéder aux paramètres de la requête HTTP sous forme de tableau de String.
- header : Map permettant d'accéder aux valeurs du Header HTTP sous forme de String.
- headerValues : Map permettant d'accéder aux valeurs du Header HTTP sous forme de tableau de String.
- cookie : Map permettant d'accéder aux différents Cookies.
- initParam : Map permettant d'accéder aux init-params du web.xml.

EL – les objets implicites

- Exemple :

  ${sessionScope ["name"] } affiche l'attribut "name" de la session

  ${ header["user-agent"] } affiche l'header "user-agent" envoyé par le navigateur.

- Attributs des différents scope

- Lors de l'évaluation d'un terme, si celui ci n'est ni un type primaire, ni un objet implicite, le conteneur JSP recherchera alors un attribut du même nom dans les différents scopes de l'application.

- L'expression suivante : ${ name } entraine la recherche de l'attribut "name" successivement dans les différents scopes, dans l'ordre
  page, request, session, application (pageScope["name"], requestScope["name"], sessionScope["name"], et applicationScope["name"]

EL – les opérateurs

Opérateurs arithmétiques		Opérateurs relationnels	
+	Addition	**==** ou **eq**	Egalité
-	Soustraction	**!=** ou **ne**	Inégalité
*****	Multiplication	**<** ou **lt**	Plus petit que
/ ou **div**	Division	**>** ou **gt**	Plus grand que
% ou **mod**	Modulo	**<=** ou **le**	Plus petit ou égal
		>= ou **ge**	Plus grand ou égal

- Les opérateurs arithmétiques ne peuvent être utilisés que sur des données numériques.
- Les opérateurs relationnels d'égalité et d'inégalité utilisent la méthode equals() d'Object. Il est donc conseillé de la redéfinir si on se sert de ces opérateurs.
- Les opérateurs relationnels de comparaison ne s'appliquent que si l'interface Comparable est implémentée La méthode compareTo() est alors utilisée.

EL – les opérateurs

Opérateurs logiques :		Autres :	
&& ou **and**	ET logique (**true** si les deux opérandes valent **true**, **false** sinon)	**empty**	**true** si l'opérande est **null**, une chaîne vide, un tableau vide, une Map vide ou une List vide. **false** sinon.
|| ou **or**	OU logique (**true** si au moins une des deux opérandes vaut **true**, **false** sinon)	**( )**	Modifie l'ordre des opérateurs.
! ou **not**	NON logique (**true** si l'opérande vaut **false**, et inversement)	**? :**	Opérateur conditionnel en ligne, format particulier : *condition* **?** *valeur_si_true* **:** *valeur_si_false*

- Les opérateurs logiques ne s'appliquent que sur des booléens.

EL – Accès aux propriété des objets

- Propriété d'un bean standard (opérateur point)

  ${ bean.name}

  Récupère dans le scope l'attribut bean et appelle sur cette objet la méthode getName ()

  Les appels à des propriétés peuvent être chainés

  ${person.adress.city}

EL – Accès aux propriété des objets

- Propriétés indexées

  Permet d'accéder à un élément d'un tableau ou d'une List

  ${list[0]}
  ${list["0"]}
  ${ list[config.value] }

EL – Accès aux propriété des objets

- Propriétés mappés

  Permet d'accéder à un élément d'une Map

  ${ map["clef1"]}
  ${ map['clef2']}
  ${ map[config.key]}

EL – Accès aux propriété des objets

- Gestion des exceptions
  - NullPointeurException (et les valeurs null) :
    Les différentes propriétés d'un élément ne sont pas forcément renseignées et peuvent très bien être null.

    Exemple, dans l'expression

    ${ sessionScope['data'].information.date },

    si un élément est null l'expression prendra la valeur null mais aucune exception ne sera lancée.

    De plus, lors de l'affichage dans la page JSP, toutes les valeurs null sont remplacées par des chaînes vides, afin de ne pas afficher le texte null à l'utilisateur...

EL – Accès aux propriété des objets

- Gestion des exceptions
  - **ArrayOutOfBoundException** :
    De la même manière, l'accès à un index incorrect d'un élément d'un tableau ou d'une List ne provoquera pas d'exception mais renverra une valeur null.

Core

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

- Gestion des variables de scope (out, set, remove, catch)
  - Afficher une expression

```
<!-- Afficher l'user-agent du navigateur ou "Inconnu" si il est absent : -->
<c:out value="${header['user-agent']}" default="Inconnu"/>
```

  - Définir une variable de scope ou une propriété

```
<!-- Mettre ${expression} dans l'attribut "varName" de la session : -->
<c:set scope="session" var="varName" value="${expression}" />

<!-- Changer la propriété "name" de l'attribut "varName" de la session : -->
<c:set target="${session['varName']" property="name" value="new value"/>
```

  - Supprimer une variable du scope

```
<!-- Supprime l'attribut "varName" de la session -->
<c:remove var="varName" scope="session"/>
```

  - Intercepter des exceptions

```
<!-- Stocker dans le scope page l'exception intercepté : -->
<c:catch var="varName">
            <c:set target="beans" property="prop" value="1"/>
</c:cath>
```

Core

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

- Actions conditionnels (if, choose, when, otherwise)
  - Test simple

```
<!-- Afficher un message si le paramètre "page" de la requête HTTP est absent-->
<c:if test="${empty param['page']}">
            Le paramètre page est absent !
</c:if>
```

  - Test complexe

```
<!-- Afficher un message différent selon la valeur du bean 'value' : -->
<c:choose>
            <c:when test="${value==1}"> value vaut 1 (Un) </c:when>
            <c:when test="${value==2}"> value vaut 2 (Deux) </c:when>
            <c:when test="${value==3}"> value vaut 3 (Trois) </c:when>
            <c:otherwise>
                        value vaut ${value} (?)
            </c:otherwise>
</c:choose>
```

Core

- Itérations (forEach, forTokens)
  - Iteration sur une collection

```
<!-- Afficher tous les éléments d'une collection dans le request-->
<c:forEach var="entry" items="${requestScope['myCollection']}" >
     ${entry}<br/>
</c:forEach>
```

  - Iteration sur une collection avec index

```
<!-- Afficher seulement les 10 premiers éléments -->
<c:forEach var="entry" items="${requestScope['myCollection']}" begin="0"
end="9">
            ${entry}<br/>
</c:forEach>
```

  - Boucle sur des id

```
<!-- Afficher les nombres de 1 à 10 -->
<c:forEach var="entry" begin="1" end="10">
           ${entry},
</c:forEach>
```

  - Iteration sur une map

```
<!-- Afficher tous les paramètres de la requête HTTP (param est une Map)-->
<c:forEach var="entry" items="${param}" >
           Le paramètre "${entry.key}" vaut "${entry.value}".<br/>
</c:forEach>
```

  - Iteration sur une chaine de caracttère

```
<c:forTokens var="p" items="mot1;mot2;mot3;mot4" delims=";">
           ${p}<br/>
</c:forTokens>
```

Core

- Les URLs (param, url, redirect, import)

```
<!-- Création d'un lien dont les paramètres viennent d'une MAP -->
<c:url value="/index.jsp" var="variableURL">
          <c:forEach items="${parameterMap}" var="entry">
                    <c:param name="${entry.key}" value="${entry.value}"/>
          </c:forEach>
</c:url>
<a href="${variableURL}">Mon Lien</a>
```

- Importer des ressources

```
<!-- Importer un fichier de l'application (similaire à <jsp:include/>) -->
<c:import url="/file.jsp">
          <c:param name="page" value="1"/>
</c:import>

<!-- Importer une ressource distante FTP dans une variable -->
<c:import url="ftp://server.com/path/file.ext" var="file" scope="page"/>
```

Format

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

- Affichage de message en fonction de la locale
  - Création d'un fichier properties nommé
    - Message_fr_FR.properties (langue française pays France)
    - Message_fr.properties (uniquement langue française)
    - Message.properties (par défaut)
  - Ce fichier contient des couples clé/valeurs :

    message.hello = Bienvenue

    message.title = Titre de la page
  - Utilisation de la balise mesage pour afficher le contenu

    `<fmt:message key="message.hello"/>` → Bienvenu pour les utilisateurs français
  - Possibilité de paramétrer des messages

```
<!-- Affichage d'un message avec deux paramètres -->
<fmt:message key="message.key">
            <fmt:param value="${mailbox.userName}"/>
            <fmt:param value="${mailbox.messageCount}"/>
</fmt:message>
```

    Message.key = Bienvenue {0}, votre boîte de réception {1,choice, 0#ne comporte aucun message | 1#comporte un message | 1<comporte {1} messages}
- Gestion de la locale
  - Par défaut récupérée en fonction du header HTTP
  - En forçant une locale en utilisant `<fmt:setLocale value="fr_FR" scope="session"/>`

Format

- Formatage des données
  - Des dates

```
<!-- Créer une date initialisé au premier janvier 2005 : -->
<fmt:parseDate value="01/01/2005" pattern="dd/MM/yyyy" var="date"/>

<!-- Afficher une date selon un format spécifique : -->
<fmt:formatDate value="${dateBeans}" pattern="dd/MM/yyyy"/>

<!-- Afficher une date selon un format standard : -->
<fmt:formatDate value="${dateBeans}" style="full"/>
```

  - Des nombres

```
<c:set var="val" value="200.51" />
<fmt:setLocale value="en_US"/>
<fmt:formatNumber value="${val}" />    => affiche 200.51
<fmt:setLocale value="fr_FR"/>
<fmt:formatNumber value="${val}" /> => affiche 200,51

<!-- Afficher une somme en Euros -->
<ftm:formatNumber value="15" type="currency" currencySymbol="&euro;"/>

<!-- Afficher un pourcentage --> <
<fmt:formatNumber value="0.25" type="percent"/>
```

Fonctions

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

- ${fn:contains()}
- ${fn:containsIgnoreCase()}
- ${fn:endsWith()}
- ${fn:escapeXml()}
- ${fn:indexOf()}
- ${fn:join()}
- ${fn:length()}
- ${fn:replace()}
- ${fn:split()}
- ${fn:startsWith()}
- ${fn:substring()}
- ${fn:substringAfter()}
- ${fn:substringBefore()}
- ${fn:toLowerCase()}
- ${fn:toUpperCase()}
- ${fn:trim()}

Exemple :

${ fn:endsWith("Il était une fois", "fois") } → true

${ fn:indexOf("Il était une fois", "une") } → 10

Permet d'écrire des test plus complexe comme

<c:if test="${fn.length(requestScope["listeLivre"]) > 10)"> ….</c:if>

Autres librairies

- XML

Permet des manipulation de XML directement dans la JSP (utilisation de Xpath et XSLT notamment)

- SQL

Permet des actions SQL directement dans les JSP.

Scriptlets - exemple

```
<%@ page import="packageNames.Book" %>
<%@ page import="java.util.*" %>
<ul>
<% List bookList = (List) request.getAttribute
  ("books-list");
    if (bookList!=null) {
        Iterator iterator = bookList.iterator();
        int i = 0;
        while ( iterator.hasNext() ) {
            Book b = (Book) iterator.next();
            out.print ("<li class='" + (i%2==0?"pair":"impair") + "'>");
            out.print (b.getName() + " (" + b.getPrice() + " &euro;)");
            if ( b.getPrice() < 30.0 )
                out.print (" <img src='hot.png' alt='Moins de 30 &euro;'/>");
            out.println ("</li>");
            i++;
        }
}
%>
</ul>
```

- Jakarta Struts - par la pratique (37.05 €)
- Swing : la synthèse (37.91 €)
- Jakarta Struts - précis & concis (O'Reilly) (8.55 €)
- Java, conception et déploiement J2EE (23.75 €)
- Programmation Orienté Aspect pour Java / J2EE (42.75 €)
- Cahiers du Programmeur Java 1 (21.85 €)
- Java & XSLT (40.85 €)
- Java & XML 2nd Edition (47.5 €)
- Enterprise JavaBeans (45.6 €)
- Le Livre de Java premier langage (27.5 €)
- Java en action (55.1 €)
- Java in a Nutshell (51.3 €)
- Au coeur de Java 2 ; tome 1 (38.0 €)
- Total Java (10.0 €)

JSTL - EL - exemple

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<ul>
    <c:forEach items="${requestScope['books-list']}" var="book" varStatus="status">
        <li class="${status.index%2==0?'pair':'impair'}">${book.name} (${book.price}
  &euro;)
            <c:if test="${book.price < 30.0}">
                <img src='hot.png' alt='Moins de 30 &euro;'/>
            </c:if>
        </li>
    </c:forEach>
</ul>
```

- Jakarta Struts - par la pratique (37.05 €)
- Swing : la synthèse (37.91 €)
- Jakarta Struts - précis & concis (O'Reilly) (8.55 €)
- Java, conception et déploiement J2EE (23.75 €)
- Programmation Orienté Aspect pour Java / J2EE (42.75 €)
- Cahiers du Programmeur Java 1 (21.85 €)
- Java & XSLT (40.85 €)
- Java & XML 2nd Edition (47.5 €)
- Enterprise JavaBeans (45.6 €)
- Le Livre de Java premier langage (27.5 €)
- Java en action (55.1 €)
- Java in a Nutshell (51.3 €)
- Au coeur de Java 2); tome 1 (38.0 €)
- Total Java (10.0 €)

Documentation

- La page officielle chez Sun :
  http://java.sun.com/products/jsp/jstl/
- La page officielle de la spécification de la JSTL :
  https://jstl-spec-public.dev.java.net/
- L'implémentation du projet Jakarta de la JSTL 1.1 :
  http://jakarta.apache.org/taglibs/doc/standard-doc/intro.html
- L'API des interfaces et classes de bases de la JSTL :
  http://java.sun.com/products/jsp/jstl/1.1/docs/api/index.html
- La documentation des différents tags :
  http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/

Questions

?