Java Server Pages : JSP

Eléments syntaxiques d'une JSP

- Une page JSP peut être formée par les éléments suivants :
  - Les expressions
  - Les déclarations
  - Les directives
  - Les scriptlets
  - Les actions
  - Les JSTL

Expressions

- Les expressions JSP sont, des expressions Java qui vont être évaluées à l'intérieur d'un appel de méthode *print*.
- Une expression commence par les caractères <%= et se termine par les caractères %>.
- Comme l'expression est placée dans un appel de méthode, il est interdit de terminer l'expression via un point-virgule.
- Syntaxe: <%=expression%>
- Equivalent à: out.println(expression) ;
- Exemple : <%=new Date()%>
- Equivalent à: out.println(new Date()) ;

Déclarations

- Dans certains cas, un peu complexe, il est nécessaire d'ajouter des méthodes et des attributs à la servlet qui va être générée (en dehors de la méthode de service).
- Une construction JSP particulière permet de répondre à ces besoins. Elle commence par les caractères *<%!* et se termine, par les caractères *%>*. Voici un petit exemple d'utilisation.
- Exemple:

```
<%@ page language="java" %>
<HTML>
  <%! private int userCounter = 0; %>
<BODY>
Vous êtes le <%= ++userCounter %><SUP>ième</SUP> client du
      site</P>
</BODY><HTML>
```

Directives

- Une directive permet de spécifier des informations qui vont servir à configurer et à influer sur le code de la servlet générée.
- Ce type de construction se repère facilement étant donné qu'une directive commence par les trois caractères *<%@*.
- Notons principalement deux directives :
  - *<%@ page ... %>* et
  - *<%@ include ... %>*
- Voyons de plus près quelques unes des possibilités qui vous sont offertes.

Directive <%@ page .. %>

- La directive <%@ page .. %> permet de pouvoir spécifier des informations utiles pour la génération et la compilation de la servlet.
- En fait, cette directive accepte de nombreux paramètres dont les principaux sont:
  -
  - <%@ page import="package|classe %>
  - <%@ page session="true|false" %>
  - <%@ page extends="classe" %>
  - <%@ page errorPage="url" %>
  - <%@ page isErrorPage="true|false" %>

Directive <%@ include .. %>

- La directive <%@ *include ... %*> est très utile si plusieurs pages se doivent de partager une même ensemble d'information.
- C'est souvent le cas avec les entêtes et les pieds de pages. Dans ce cas, codez ces parties dans des fichiers séparés et injectez les, via cette directive, dans tous les autre fichiers qui en ont besoin.
- Voici un petit exemple d'utilisation de cette directive:

```
<%@ page language="java" %>
<HTML>
<BODY>
<%@ include file="header.jsp" %>
 <!-- Contenu de la page à générer -->
<%@ include file="footer.jsp" %>
</BODY><HTML>
```

Scriptlets

- Les scriptlets correspondent aux blocs de code introduit par le caractères <% et se terminant par %>.
- Ils servent à ajouter du code dans la méthode de service.
- Le code Java du scriptlet est inséré tel quel dans la servlet générée : la vérification, par le compilateur, du code aura lieu au moment de la compilation totale de la servlet équivalent.
- L'exemple complet de JSP présenté précédemment, comportait quelques scriptlets :

```
<%
for(int i=1; i<=6; i++) {
 out.println("<H" + i + " align=\"center\">Heading " +i+
    "</H" + i + ">");
 }
%>
```

Les actions

- Les actions constituent une autre façon de générer du code Java à partir d'une page JSP.
- Les actions se reconnaissent facilement, syntaxiquement parlant : il s'agit de tag XML ressemblant à <jsp:tagName ... />.
- Cela permet d'indiquer que le tag fait partie du namespace (espace de noms) *jsp*. Le nom du tag est préétabli.
- Enfin, le tag peut, bien entendu comporter plusieurs attributs.
- Il existe plusieurs actions différentes. Les principales sont les suivantes

Les actions

- <jsp:include> : Inclusion coté serveur
  - Exemple : <jsp:include page= "entete.jsp" />
- <jsp:forward> : Redirection vers une page
  - Exemple : <jsp:forward page="affiche.jsp" />
- <jsp:useBean> : Instanciation d'un objet java (java bean)
  - Exemple :
    - <jsp:useBean id="jbName" class="TheClass" scope="session" />
- <jsp:setProperty> : Cette action, permet de modifier une propriété sur un objet créé via l'action *<jsp:useBean ...>*
  - Exemple :
    - <jsp:setProperty name="jbName" property="XXX" value="<%= javaExpression %>" />
- <jsp:getProperty> : cette action est l'inverse de la précédente : elle permet de retourner dans le flux HTML, la valeur de la propriété considérée.
  - Exemple :
    - <jsp:getProperty name="jbName" property="XXX" />

JSTL :

- **Sun** a proposé une spécification pour une librairie de tags standard : la **Java Standard Tag Library** (**JSTL**).
- La **JSTL** est une implémentation de Sun qui décrit plusieurs actions basiques pour les applications web **J2EE**. Elle propose ainsi un ensemble de librairies de tags pour le développement de pages **JSP**.
- Le but de la **JSTL** est de simplifier le travail des auteurs de page JSP, c'est à dire la personne responsable de la couche présentation d'une application web J2EE.
- En effet, un web designer peut avoir des problèmes pour la conception de pages JSP du fait qu'il est confronté à un langage de script complexe qu'il ne maîtrise pas forcément.

Librairies de la JSTL1.2

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

Exemple de déclaration au début d'une JSP :

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

Les fichiers jars à inclure au classpath de votre projets :

- jstl-1.2.jar
- standard-1.2.3.jar

<c:...../> : Librairie de base

- Déclaration:

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

- Gestion des variables de scope :
  - Cette section comporte les actions de base pour la gestion des variables de scope d'une application web :
    - L'affichage de variable
    - La création/modification/suppression de variable de scope
    - La gestion des exceptions

<c:...../> : Librairie de base

- **Afficher une expression <c:out …./>**

```
<!-- Afficher l'entête user-agent du navigateur ou
   "Inconnu" si il est absent : -->
```

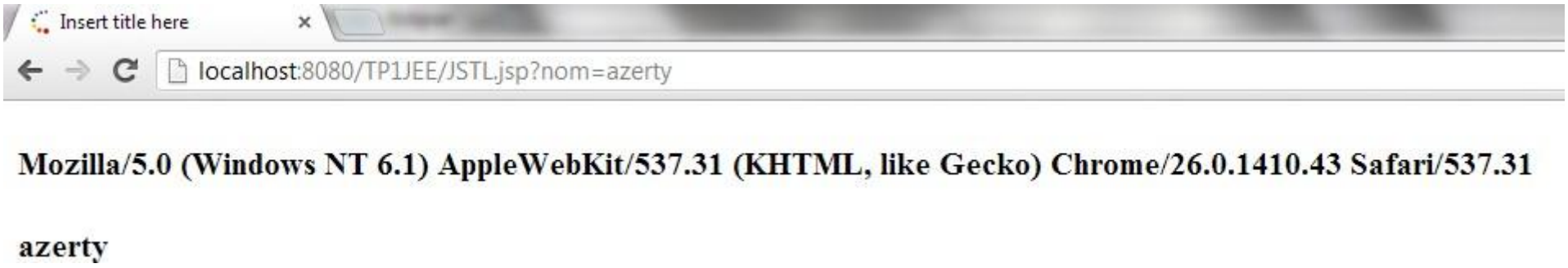
- `<c:out value="${header['user-agent']}" default="Inconnu"/>`

```
<!- ou encore : -->
```

- `<c:out value="${header['user-agent']}"> Inconnu </c:out>`

```
<!-- Afficher le paramètre nom de la req http : -->
```

- `<c:out value="${param['nom']}" default="Inconnu"/>`



<c:...../> : Librairie de base

<c:if/> : Traitement conditionnel

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

<c:chose/> : Traitement conditionnel exclusif

```
<c:choose>
  <c:when test="${param['x']==1}">Premier</c:when>
  <c:when test="${param['x']==2}">Deuxième</c:when>
  <c:otherwise>Aucun</c:otherwise>
</c:choose>
```

<c:...../> : Librairie de base

- **<c:forEach/> : Itérer sur une collection :**
  - Permet d'effectuer simplement des itérations sur plusieurs types de collections de données.
  - L'attribut **items** accepte les éléments suivant comme collection :
    - Les tableaux d'objets ou de types primaires
    - Une implémentation de **java.util.Collection** en utilisant la méthode **iterator()**.
    - Une implémentation de **java.util.Iterator**.
    - Une implémentation de **java.util.Enumeration**.
    - Une implémentation de **java.util.Map**, en utilisant les méthodes **entrySet().iterator()**.
    - Une **String** dont les différents éléments sont séparés par des virgules (mais il est préférable d'utiliser <c:forTokens/> à sa place).
    - Une valeur **null** sera considérée comme une collection vide (pas d'itération).
    - Si l'attribut **items** est absent, les attributs **begin** et **end** permettent d'effectuer une itération entre deux nombres entiers.
  -

<c:...../> : Librairie de base

- **<c:forEach/> : Itérer sur une collection :**
  - Exemples :

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

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

<!-- Afficher les nombres de 1 à 10 -->
<c:forEach var="entry" begin="1" end="10">
   ${entry},
</c:forEach>
<!-- Afficher tous les paramètres de la requête et leurs valeurs  -->

<c:forEach var="p" items="${param}">
  Le paramètre ${p.key} vaut ${p.value}<br/>
</c:forEach>
```

<c:...../> : Librairie de base

- **<c:forTokens/> : Itérer sur des éléments d'une String :**
  - Permet de découper des chaînes de caractères selon un ou plusieurs délimiteurs. Chaque marqueur ainsi obtenu sera traité dans une boucle de l'itération.
  - Exemple :

```
<!-- Afficher séparément des mots séparés par un point-
   virgule -->
<c:forTokens var="p" items="mot1;mot2;mot3;mot4" delims=";">
    ${p}<br/>
</c:forTokens>
```

<c:...../> : Librairie de base

- **<c:param/> : Ajouter un paramètre à une URL**
  - Permet d'ajouter simplement un paramètre à une URL représentée par le tag parent.
  - Cette balise doit avoir comme balise parent une balise **<c:url/>**, **<c:import/>** ou **<c:redirect/>** (mais pas forcement comme parent direct).
  - Exemples:

```
<!-- La forme suivante : -->
<c:url value="/mapage.jsp?paramName=paramValue"/>

<!-- est equivalente à : -->
<c:url value="/mapage.jsp">
  <c:param name="paramName" value="paramValue"/>
</c:url>
```

<c:...../> : Librairie de base

- **<c:url/> : Créer une URL**
  - Permet de créer des URLs absolues, relatives au contexte, ou relatives à un autre contexte.
  - Exemple :

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

<c:...../> : Librairie de base

- **<c:redirect/> : Redirection**
  - Envoi une commande de redirection HTTP au client.
  - Exemple :

```
<!-- Redirection vers le portail de developpez.com : -->
<c:redirect url="http://www.developpez.com"/>

<!-- Redirection vers une page d'erreur avec des paramètres:
   -->
<c:redirect url="/error.jsp">
   <c:param name="from"
   value="${pageContext.request.requestURI}"/>
</c:redirect>
```

<c:...../> : Librairie de base

- **<c:import/> : Importer des ressources**
  - Permet d'importer une ressource selon son URL.
  - Contrairement à **<jsp:include/>**, la ressource peut appartenir à un autre contexte ou être hébergée sur un autre serveur...
  - Exemples :

```
<!-- Importer un fichier de l'application (comme <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"/>
<!-- Importe une ressource distante dans un Reader -->
<c:url value="http://www.server.com/file.jsp" var="fileUrl">
   <c:param name="file" value="filename"/>
   <c:param name="page" value="1"/>
</c:url>
<!-- Ouverte d'un flux avec un Reader -->
<c:import url="${fileUrl}" varReader="reader">
</c:import>
```