Les apports de JSP 2.0

1 Introduction

Nous avons déjà vu quelques nouveautés de la version 2.0 dans les travaux pratiques précédents (JSTL et langage d'expressions). Il nous reste deux améliorations, l'écriture de balises personnalisées et la production de document XML.

2 Paramétrage des pages jsp

Pour affiner le paramétrage des pages JSP, vous pouvez ajouter au fichier `web.xml` les éléments suivants :

À ajouter au fichier `web.xml`

```
...
<jsp-config>
    <jsp-property-group>
        <description>Toutes les pages</description>
        <url-pattern>*.jsp</url-pattern>
        <page-encoding>UTF-8</page-encoding>
        <include-prelude>/prelude.jsp</include-prelude>
        <include-prelude>/copyright.jsp</include-prelude>
        <include-coda>/coda.jsp</include-coda>
    </jsp-property-group>
</jsp-config>
...
```

Cette configuration va inclure les pages `/prelude.jsp` et `/copyright.jsp` au début et `/coda.jsp` à la fin de chaque page JSP. Le code de ces pages est ci-dessous :

Page `/prelude.jsp`

```
<!DOCTYPE html>

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

<html>
<head>
    <title>Mon application</title>
    <link rel="stylesheet" type="text/css" href="${charte}" />
</head>
<body>
```

Page `/copyright.jsp`

```
<!-- (c) JL Massat 2017 -->
```

Page `/coda.jsp`

```
</body>
</html>
```

Les pages JSP standards deviennent allégées et se concentrent sur le contenu de l'élément `body`.

3 Balises personnalisées

Depuis les JSP 2.0 nous pouvons facilement créer de nouvelles balises JSP définies à partir d'un fichier JSP. Pour ce faire, suivez les étapes ci-dessous :

1. Placez vous dans l'application WEB de test définie dans les TP précédents.
2. Créez les répertoires `/WEB-INF/tags/mestags/`
3. Créez le fichier `/WEB-INF/tags/mestags/message.tag`

Fichier `message.tag`

```
<%@ tag language="java" pageEncoding="UTF-8" %>

<p>Bonjour</p>
```

4. Testez l'utilisation de votre nouvelle balise `message` avec la page JSP ci-dessous :

```
<%@ taglib prefix="mm" tagdir="/WEB-INF/tags/mestags" %>

<html>
<body>
<h1>Tester ma balise</h1>

<mm:message />

</body>
</html>
```

5. Les balises personnalisées peuvent également **avoir un contenu** et utiliser ce contenu. Modifiez le fichier `message.tag` avec

Nouvelle version de `message.tag`

```
<%@ tag language="java" pageEncoding="UTF-8"%>

<p>Votre contenu : <jsp:doBody /></p>
```

6. Testez cette nouvelle version avec :

```
<%@ taglib prefix="mm" tagdir="/WEB-INF/tags/mestags" %>

<html>
<body>
<h1>Tester ma balise</h1>

<mm:message>
    <b>Bonjour</b>
</mm:message>

</body>
</html>
```

7. Nous pouvons aussi **déclarer des attributs** et les utiliser dans le corps de notre nouvelle balise. Modifiez le fichier `message.tag` avec

Nouvelle version avec attribut de `message.tag`

```
<%@ tag language="java" pageEncoding="UTF-8"%>

<%@ attribute name="type" required="true" description="Type du message" %>

<p>Votre contenu (type ${type}) : <jsp:doBody /></p>
```

8. A ce stade la page JSP de test doit avoir une erreur de validation : l'attribut `type` est manquant. Utilisez maintenant cette nouvelle version :

```
<%@ taglib prefix="mm" tagdir="/WEB-INF/tags/mestags" %>
<html>

<body>
<h1>Ajouter une balise</h1>

<mm:message type="Important">Mon contenu</mm:message>

</body>
</html>
```

9. Nous pouvons aussi **ajouter un type pour les attributs**. Nous obtenons la nouvelle version ci-dessous :

Nouvelle version avec attribut typé de `message.tag`

```
<%@ tag language="java" pageEncoding="UTF-8"%>

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

<%@ attribute name="type" required="true" description="Type du message" %>
<%@ attribute name="textes" required="false" type="java.util.List" %>

<p>Votre contenu (type ${type}) : <jsp:doBody /></p>

<c:if test="${textes != null}">
  <p>Vos textes :</p>
  <ul>
  <c:forEach var="texte" items="${textes}">
    <li><c:out value="${texte}"/></li>
  </c:forEach>
  </ul>
</c:if>
```

10. Le nouvelle attribut étant **facultatif**, vous pouvez continuer d'utiliser l'ancienne version de la page de test.
11. Utilisez maintenant la version ci-dessous pour tester l'utilisation d'un **attribut typé** :

```
<%@page import="java.util.Arrays"%>
<%@page import="java.util.List"%>
<%@ taglib prefix="mm" tagdir="/WEB-INF/tags/mestags" %>
<html>

<body>
<h1>Ajouter une balise</h1>

<mm:message type="Important">Mon contenu</mm:message>

<%
List<String> mesTextes = Arrays.asList("Hello", "Salut");
%>

<mm:message type="Alerte" textes="<%= mesTextes %>"/>

</body>
</html>
```

12. Comparez avec la nouvelle version ci-dessous :

```
<%
List<String> mesTextes = Arrays.asList("Hello", "Salut");
pageContext.setAttribute("mesTextes", mesTextes);
%>

<mm:message type="Alerte" textes="${mesTextes}"/>
```

Travail à faire : vous devez construire une nouvelle balise qui facilite la création d'un élément HTML `select`. Prenez soin d'utiliser un attribut facultatif (`label`) qui associe un titre (élément HTML `label`) au `select` :

```
<mm:select
    name="statut"
    value="${personne.statut}"
    options="${status}"
    label="Situation :"
    />
```

4 Balises personnalisées en Java

Il est également possible de définir de nouvelles balises en fournissant l'implantation sous la forme **d'une classe Java**. Vous trouverez plus d'information dans ces deux leçons en ligne

- Comment créer un TAG simple en Java˜ ? [1]
- Comment créer un TAG avec contenu en Java˜ ? [2]

Travail à faire : faites fonctionner les exemples proposés dans ces documentations.

Travail à faire : utilisez ce mécanisme pour créer deux balises qui réalisent l'affichage d'un tableau (les lignes paires et impaires sont colorées de manière différente) :

```
<mm:tableau centrer="oui">
    <mm:ligne>première ligne</mm:ligne>
    <mm:ligne>deuxième ligne</mm:ligne>
</mm:tableau>
```

1. http://www.avajava.com/tutorials/lessons/how-do-i-create-a-tag-using-simpletagsupport.html
2. http://www.avajava.com/tutorials/lessons/how-do-i-create-a-tag-using-simpletagsupport-that-uses-the-tag-body.html

5 Production de documents XML

Vous trouverez ci-dessous un exemple de page JSPX (codage XML des pages JSP). La syntaxe est légèrement différente mais le mode de fonctionnement reste le même. L'idée est de faciliter la production de page XML. Cette approche est moins utilisée aujourd'hui. Essayez la page ci-dessous (avec l'extension `.jspx`) :

Fichier `test-xml.jspx`

```
<jsp:root
    xmlns:jsp="http://java.sun.com/JSP/Page"
    xmlns:c="http://java.sun.com/jsp/jstl/core"
    version="2.0"
    >

<!-- fixer l'encodage de sortie     -->
<!-- syntaxe XML de la directive @ page -->
<jsp:directive.page contentType="text/xml; charset=iso-8859-1" />

<!-- fixer le format de sortie (comme XSL) -->
<jsp:output
    omit-xml-declaration="false"
    doctype-root-element="messages"
    doctype-system="http://monserveur.fr/messages.dtd"/>

<messages>

    <!-- un message en dur -->
    <message>
    Hello
    </message>

    <!-- évaluation java comme <%= %> -->
    <!-- syntaxe XML des expressions -->
    <message>
    <jsp:expression>10 + 20</jsp:expression>
    </message>

    <!-- un message dynamique -->
    <jsp:element name="message">
        <jsp:attribute name="class">important</jsp:attribute>
        <jsp:body>
          Bizarre
        </jsp:body>
    </jsp:element>

    <!-- JSTL + EL -->
    <message>
    <c:out value="${11 + 22}" />
    </message>

</messages>

</jsp:root>
```

Ce document XML est traité par le moteur de servlet et les éléments de l'espace de nom `xmlns:jsp` sont évalués et remplacés par le résultat de l'évaluation. Les pages JSP sont des documents de texte contrairement aux pages JSPX qui sont des documents XML.

Travail à faire : Faites en sorte de prendre une (**et une seule**) de vos anciennes page JSP et de la transformer en JSPX pour qu'elle produise du XHTML valide. Elle doit ressembler à ceci :

```
<jsp:root
   version="2.0"
   xmlns:jsp="http://java.sun.com/JSP/Page"
   xmlns:fmt="http://java.sun.com/jsp/jstl/fmt"
   xmlns="http://www.w3.org/1999/xhtml">
<jsp:output
    omit-xml-declaration="false"
    doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"/>
<html>
  <jsp:directive.page contentType="text/html" />
  <head>
    <title>XHTML en JSPX</title>
  </head>
  <body>
    <p>Un exemple de XHTML en JSPX
    <jsp:useBean id="now" class="java.util.Date" />
    <fmt:formatDate value="${now}" pattern="MMMM d, yyyy, H:mm:ss"/>
    </p>
  </body>
</html>
</jsp:root>
```