La technologie JSP (Java Server Page)

Table des matières

1 Introduction

Constat :

- La production de pages HTML à l'aide de Servlet est une opération fastidieuse.
- Le respect d'une charte graphique est difficile.
- Les graphistes ne peuvent travailler sur des servlets.

Solution :

- Introduire du code Java dans une page HTML (ou XML).
- Exécuter ce code à la volée et le remplacer par le résultat de son exécution.

Version courante de la technologie JSP : 3.0 (JEE 9)

2 Principe des pages JSP

- Dans une page JSP on trouve du code HTML (ou XML), des **directives**, des **actions** et du code Java.

```
<html><body>
<p><%
    for(int i=0; i<10; i++) out.print(i + " ");
%></p>
</body></html>
```

- La requête « `GET /compter.jsp` », est traitée comme suit :
  - ▷ Produire et compiler le code source Java de la servlet (si nécessaire)
  - ▷ Instancier la classe et appeler la méthode `init` (si nécessaire)
  - ▷ Appeler la méthode `service`

3 Introduire du code Java

Page JSP avec du code Java (à éviter)

```
<!-- Les déclarations -->
<%!
  int i = 0;
  int plus10(int j) { return j+10; }
%>

<!-- Les scriplets -->
<%
  i = plus10(i); out.println(i);
%>

<!-- Les expressions -->
<p>Le compteur plus 10 est <%= i + 10 %></p>
```

Attention : les injections JavaScript sont possibles

```
<%!
String text = "<script src='http://myserver/myscript.js'></script>";
%>
<%= text %>
```

3.1 Utilisation du code Java

- Un exemple de test :

```
<% if (age > 30) {
    %>
    <p>L'age est supérieur à 30</p>
    <%
 } else {
    %>
    <p>L'age est inférieur à 30</p>
    <%
 } %>
```

La version ci-dessous est préférable :

```
<%
   if (age > 30) {
      out.println("<p>L'age est supérieur à 30</p>");
   } else {
      out.println("<p>L'age est inférieur à 30</p>");
   }
%>
```

- Un exemple de boucle :

```
<% for(int i=0; i<10; i++) {
    %>
    <p>i = <%= i %>.</p>
    <%
    }
%>
```

La version ci-dessous est préférable :

```
<%
   for(int i=0; i<10; i++) {
      out.println("<p>i = ");
      out.println(i);
      out.println(".</p>");
   }
%>
```

4 La directive page

- Forme générale :

```
<%@ page
    import="java.io.*;java.sql.*"
    session="true"
    isThreadSafe="true"
    errorPage="erreur.jsp"
    isErrorPage="false"
    contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"
    language="java"
    %>
```

- il peut y avoir plusieurs directives `page` dans une JSP :

```
<%@ page import="java.io.*" %>
<%@ page import="java.sql.*" %>
<%@ page isThreadSafe="false" %>
...
```

5 Les directives `include` et `taglib`

- La directive `include` (inclusion à la compilation) :

```
<%@ include file="banniere.jsp" %>
```

- La directive `taglib` :

```
<%@ taglib uri="monTag" prefix="jlm" %>
...
<jlm:debut> ... </jlm:debut>
```

5.1 Les variables implicites

- Requête et réponse :

  `HttpServletRequest request`

  Tout sur la requête, le navigateur, le protocole et le client.

  `HttpServletResponse response`

  Tout sur la réponse, le codage, les erreurs, le tampon de sortie.

  `javax.servlet.jsp.JspWriter out`

  Accès à l'impression formattée.

- Accès aux données :

  `javax.servlet.http.HttpSession session`

  Tout sur la session courante.

  `javax.servlet.ServletContext application`

  Tout sur l'application.

- Contexte et paramètres :

  `javax.servlet.jsp.PageContext pageContext`

  Tout sur la page : requête, réponse, session, sortie.

  `javax.servlet.ServletConfig config`

  Tout sur la configuration de la servlet : paramètres, nom.

6 Les actions JSP

- Les actions JSP sont des balises qui agissent sur le déroulement de l'exécution.
- Forme générale :

```
<jsp:action
      attribut1="valeur1"
      attribut2="valeur2"
      ... />
```

6.1 Inclusion dynamique de JSP

Inclusion dynamique de contenu

```
<jsp:include page="url_de_la_page" flush="true" />
```

Exemple :

Page banniere.jsp

```
<h2>Titre : <%= request.getParameter("titre") %></h2>
```

Inclusion dynamique de banniere.jsp

```
<html><body>

    <h1>Essai de jsp:include</h1>

    <jsp:include page="banniere.jsp" flush="true">
        <jsp:param name="titre" value="Titre par jsp:include"/>
    </jsp:include>

    <!-- une deuxième utilisation est possible -->

</body></html>
```

6.2 Chaînage de JSP

Chaînage de JSP/Servlet :

```
<jsp:forward page="relative_url" />
```

Un exemple :

```
<jsp:forward page="banniere.jsp">
  <jsp:param name="titre" value="Premier titre"/>
</jsp:forward>

<p>paragraphe ignoré !</p>
```

6.3 Utiliser des JavaBeans

Soit le JavaBean `myapp.Product` :

```java
package myapp;

import lombok.Data;

@Data
public class Product {

    String name;
    String price;
    String description;

}
```

La page `showProduct.jsp` réalise son affichage :

```jsp
<jsp:useBean id="product" scope="session" class="myapp.Product" >
    <p>Nouveau produit !</p>
</jsp:useBean>

<p>Nom: <%= product.getName() %></p>
<p>Prix: <%= product.getPrice() %></p>
<p>Description: <%= product.getDescription() %></p>
```

Le code JSP placé dans l'action `<jsp:useBean>` est exécuté si le bean est créé. Le **scope** peut être `page`, `request` ou `application`.

6.4 Utilisation du langage d'expressions

Ne plus utiliser de code Java pour l'affichage d'un **bean** :

```jsp
<jsp:useBean id="product" scope="session" class="myapp.Product" >
    <p>Nouveau produit !</p>
</jsp:useBean>

<p>Nom: ${product.name}</p>
<p>Prix: ${product.price}</p>
<p>Description: ${product.description}</p>
```

Les formes possibles d'une EL (le `useBean` est optionel) :

`${variable}`	accès à une variable (page, requête, session, app.)
`${variable.property}`	accès à une propriété
`${variable[ index ]}`	accès à un élément d'une table
`${variable[ key ]}`	accès à un élément d'une Map
`${variable.key}`	accès à un élément d'une Map

```
${person.products[3].name}
```

6.5 Modifier des JavaBeans

- Il existe trois façons de modifier un JavaBean dans une page JSP :

```jsp
<jsp:setProperty name="product" property="name" value="Voiture" />
```

En récupérant la valeur d'un paramètre :

```
<jsp:setProperty name="product" property="price" param="price" />
```

En récupérant tous les paramètres :

```
<jsp:setProperty name="product" property="*" />
```

6.6 Un exemple complet

- Un exemple d'affectation des JavaBeans (`changeProduct.jsp`) :

```
<jsp:useBean
    id="product"
    scope="session"
    class="myapp.Product" />

<jsp:setProperty name="product" property="*"/>

<jsp:forward page="showProduct.jsp" />
```

Quelques essais :

```
changeProduct.jsp
changeProduct.jsp?name=Voiture&price=200
changeProduct.jsp
changeProduct.jsp?desc=blablabla
```

7 Gestion des erreurs

- Une page de gestion des erreurs peut ressembler à ceci :

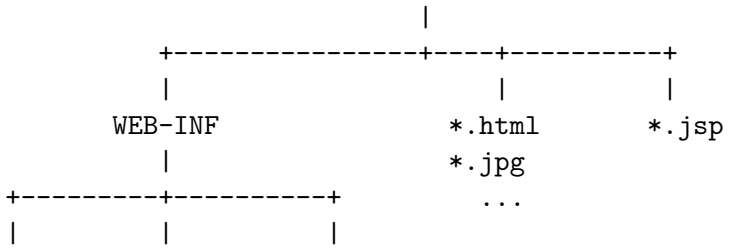
```
<html>

<%@ page
   language="java"
   isErrorPage="true"
%>

<body>
erreur : <%= exception.getMessage() %>
...
```

8 Une application JSP

- Structure des fichiers d'une application WEB JSP/Servlet :



```
web.xml    classes      lib

config.    *.class      *.jar
```

- ces fichiers peuvent être rangés dans une WAR (**Web Application aRchive**) en fait une archive jar.

9 Configuration d'une application JSP

Le fichier `web.xml`

```
<web-app ...>

    ... premières déclaration ...
    ... déclaration des servlets ...
    ... configuration des sessions ...

    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <error-page>
        <exception-type>java.lang.Exception</exception-type>
        <location>/erreurs.jsp</location>
    </error-page>

    <jsp-config> ... </jsp-config>

</web-app>
```

- Le détail de la configuration des pages JSP :

```
<jsp-config>

    ... déclaration des taglib ...

    <jsp-property-group>
        <description>Toutes les pages</description>

        <!-- pages concernées par ces propriétés -->
        <url-pattern>*.jsp</url-pattern>

        <!-- encodage de sortie de ces pages -->
        <page-encoding>UTF-8</page-encoding>

        <!-- page(s) à inclure avant -->
        <include-prelude>/prelude.jsp</include-prelude>

        <!-- page(s) à inclure après -->
        <include-coda>/coda.jsp</include-coda>
    </jsp-property-group>

</jsp-config>
```

10 La JSTL (JSP Standard Tag Lib)

La JSTL 1.1 offre une multitude de balises pour traiter :

- l'internationalisation,

- les tests, les études de cas, les boucles.

Mais également (moins utile) :

- la lecture, le traitement de documents XML,
- le traitement de requêtes SQL.

Sans EL et la JSTL

```
<jsp:useBean id="person" scope="session" class="myapp.Person" />

<p><%= person.getName() %></p>

<% if (person.getCity().equals("Marseille")) { %>
    <p>de Marseille</p>
<% } %>
```

Avec EL et la JSTL

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

<p><c:out value ="${person.name}" /></p>

<c:if test="${person.city == 'Marseille'}">
    <p>de Marseille</p>
</c:if>
```

11 Les librairies de balises

Principes :

- étendre le language JSP,
- réutilisation de balises standards,
- améliorer l'approche déclarative et limiter la présence du code Java,
- réutilisation de code JSP.

11.1 Installation d'une librairie de balises

- Un exemple `datetime` :

```
taglibs-datetime.jar
taglibs-datetime.tld
```

- Copier `taglibs-datetime.jar` dans le répertoire `/WEB-INF/lib`
- Copier `taglibs-datetime.tld` dans le répertoire `/WEB-INF`
- Ajouter dans `/WEB-INF/web.xml` la déclaration

```
<jsp-config>
  <taglib>
    <taglib-uri>http://jakarta.apache.org/taglibs/datetime</taglib-uri>
    <taglib-location>/WEB-INF/datetime.tld</taglib-location>
  </taglib>
  ...
</jsp-config>
```

11.2 Utilisation d'une librairie de balises

- Dans chaque page JSP qui a besoin d'utiliser les balises de `datetime` il faut ajouter :

```
<%@ taglib
    uri="http://jakarta.apache.org/taglibs/datetime"
    prefix="dt" %>
```

- On peut maintenant l'utiliser :

```
<p>La date en millisecondes est <dt:currentTime/></p>

<p>En francais
    <dt:format pattern="MM/dd/yyyy hh:mm">
        <dt:currentTime/>
    </dt:format>
</p>
```