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 : 3.1 (JEE 10)

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>
    <%
 } %>
```

Une boucle

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

L'utilisation est **très délicate** et doit être évité autant que possible.

3.2 Les variables implicites

- Requête et réponse :

  `HttpServletRequest request`

  `HttpServletResponse response`

  `jakarta.servlet.jsp.JspWriter out`

- Accès aux données :

  `jakarta.servlet.http.HttpSession session` La session courante.

  `jakarta.servlet.ServletContext application` Tout sur l'application.

- Contexte et paramètres :

  `jakarta.servlet.jsp.PageContext pageContext`

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

  `jakarta.servlet.ServletConfig config`

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

4 Les directives

La directive page

```
<%@ 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"
    %>
```

La directive include

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

La directive taglib

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

5 Les actions JSP

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

Inclusion dynamique de contenu

```
<jsp:include page="url_externe_ou_interne" flush="true" >
    <jsp:param name="nom_paramètre" value="valeur" />
    ...
</jsp:include>
```

Chaînage de JSP

```
<jsp:forward page="relative_url">
    <jsp:param name="nom_paramètre" value="valeur" />
    ...
</jsp:forward>

texte ignoré.
```

5.1 Utiliser des JavaBeans

Un javabean

```
package myapp;

@lombok.Data
public class Product {
    String name;
    String price;
    String description;
}
```

La page showProduct.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`.

5.2 Utilisation du langage d'expressions

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

```
<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}
```

5.3 Modifier des JavaBeans

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

Affectation d'une valeur constante

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

Affectation avec la valeur d'un paramètre

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

Affectation toutes les propriétés avec les paramètres

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

5.4 Un exemple complet

Un exemple d'affectation des JavaBeans :

Page 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?description=blablabla
```

6 Gestion des erreurs

Une page de gestion des erreurs peut ressembler à ceci :

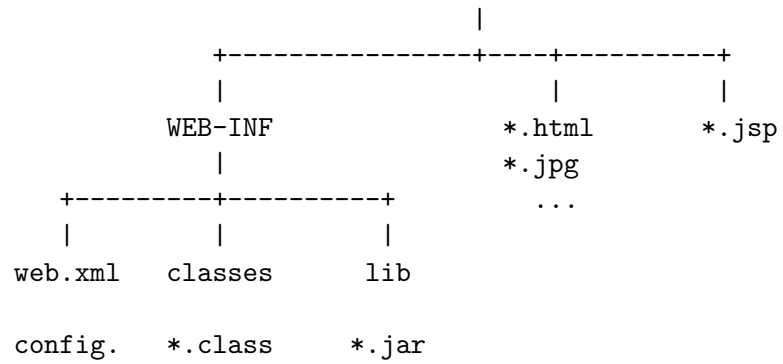
```
<html>

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

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

7 Une application JSP

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



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

8 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>
```

9 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.

9.1 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.
- la lecture, le traitement de documents XML (moins utile),
- le traitement de requêtes SQL (moins utile).

Sans EL et la JSTL

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

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

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

Avec EL et la JSTL

```
<%@ taglib prefix="c"
     uri="jakarta.tags.core" %>


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

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