Développement web avancé – Java EE (SUITE)

Génie Informatique – Semestre S6

Les JSP (Java Server Pages)

1. Définition et fonctionnement de base

JSP : Java Server Pages est une *extension standard aux Servlets*
http://www.oracle.com/technetwork/java/javaee/jsp/index.html

L'utilisation des JSP est basée sur la technique server side include (SSI). Une page ssi (shtml) est demandée par un client web, le serveur Web passe la main au programme adéquat qui traite la partie de la page le concernant.

Ce programme génère la partie dynamique la page HTML créée dans son ensemble est retournée au serveur puis au client Web.

JSP est ainsi la technique des ssi en Java càd une page HTML contenant du code Java. Elle permet une meilleure division des tâches :

- présentation générale par les graphistes
- coté dynamique par des programmeurs (Java)

Concrêtement, toute la page HTML est convertie en une servlet. Cette servlet est traitée par le moteur Java intégré au serveur Web (technologie des servlets) et retourne la page HTML construite.

2. Principe de base et exemples

Pour résumer :

- Servlet = du code Java contenant de l'HTML
- JSP = une page HTML contenant du code Java

Avec les JSP, les parties statiques de la page HTML sont écrites en HTML et les parties dynamiques de la page HTML sont écrites en Java.

2.1. Un premier JSP

Considérant la fichier suivant nommé MaDate .jsp

```
<html><head><title>Obtenu par une JSP</title></head>
<body>
<h3>Bonjour de ma part </h3> <hr>
La date courante est :  <%= new java.util.Date() %>
</body>
</html>
```

Le fichier est traité quand le client demande l'URL de la JSP :

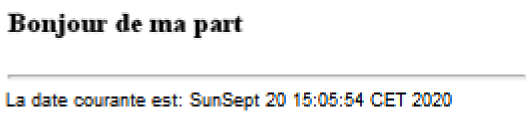
```
http://serveurWeb:<port>/.../MaDate.jsp
```

NB : Un Des exemples de JSP (code + liens pour l'exécution) sont disponibles à partir de REP_INSTALL_TOMCAT/webapps/examples/jsp sachant que REP_INSTALL_TOMCAT est le répertoire dans lequel Tomcat est installé ;

2.2. Exécution de JSP

Il faut mettre les pages JSP dans un endroit particulier du serveur Web. Cet endroit dépend du serveur Web et de sa configuration. Pour tomcat par exemple en configuration standard, on execute la page JSP stocké dans le répertoire standard **(REP_INSTAL_TOMCAT\webapps\examples\jsp\MaDate.jsp)** en utilisant l'url : **http://localhost:8080/examples/jsp/MaDate.jsp**

Le résultat de MaDate.jsp



Une autre exécution donne une autre date. Le résultat est alors dynamique.

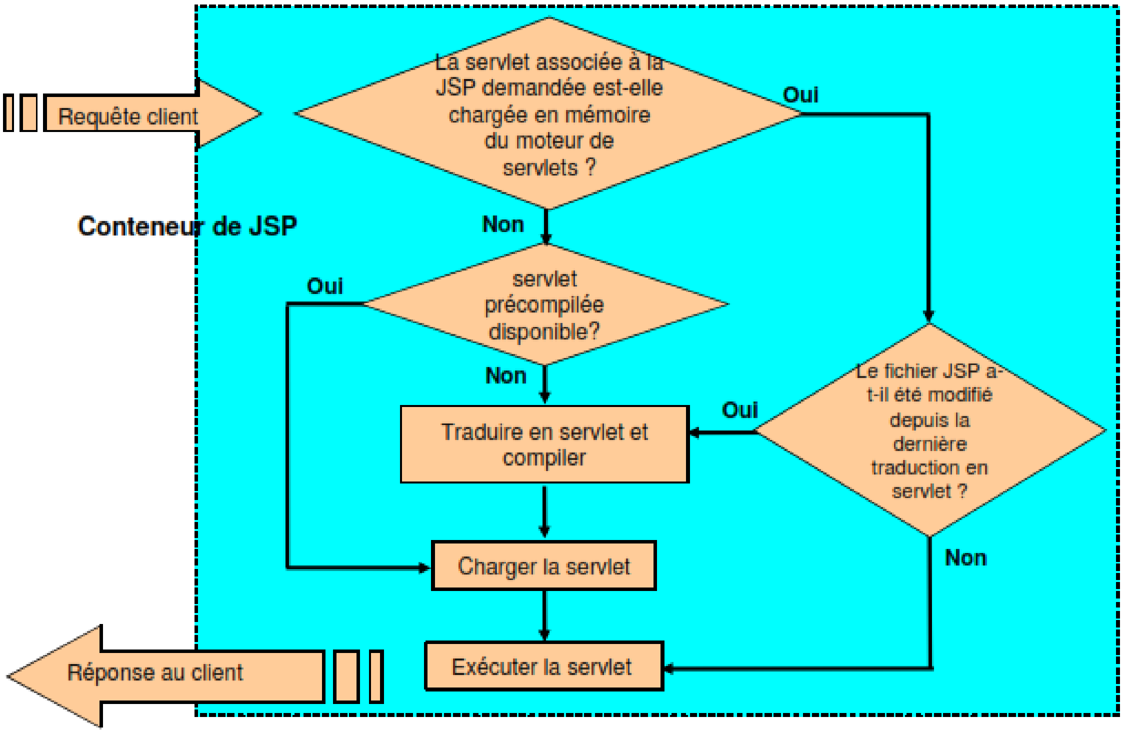
Le moteur de JSP a construit une servlet (MaDate_jsp.java sous l'arborescence work) Cette phase est parfois appelée la traduction de la JSP (en servlet). Le moteur a ensuite compilé et exécuté la servlet.

La Servlet construite ressemble à la suivante :

```
package org.apache.jsp;
...
public class MaData_jsp extends HttpJspBase {
...
    public void _jspService(HttpServletRequest request, HttpServletResponse
response) throws IOException, ServletException {
  ...
            pageContext = _jspxFactory.getPageContext(...);
            session = pageContext.getSession();
            out = pageContext.getOut();
            // HTML
   // begin [file="C:\\...\\examples\\jsp\\MaDate.jsp";from=(0,0);to=(4,24)]
         out.write("<html><head><title>Obtenu par une JSP</title></head>\r\n
<body>\r\n\r\n<h3>Bonjour de ma part</h3> <hr>\r\n
La date courante est :  ");
     // end
//begin [file="C:\\...\\examples\\jsp\\MaDate.jsp";from=(4,27)to=(4,49)]
                out.print( new java.util.Date() );
            // end
            // HTML
// begin [file="C:\\...\\examples\\jsp\\date.jsp";from=(4,51);to=(6,7)]
                out.write("\r\n</body>\r\n</html>"); // end
            ...
    }
}
```

2.3. Enchaînement d'exécution

L'algorithme d'execution d'une JSP ressemble au processus suivant :



3. Composantes d'une JSP

Une servlet peut contenir trois principales parties :

- **Les scriptlets <% %>**
- **déclarations <%! %>**
- **expressions <%= %>**
-

3.1. Scriptlets <% %>

contient du code Java insérer dans _jspService() de la servlet, donc peut utiliser out, request, response, etc.

Exemple :

```
<%
    String[] langages = {"Java", "C++", "Smalltalk", "Simula 67"};
    out.println("<h3>Principaux langages orientés objets : </h3>");
    for (int i=0; i < langages.length; i++) {
            out.println("<p>" + langages[i] + "</p>");
    }
%>
```

3.2. Déclarations <%! %>

Les déclarations Sont des déclarations Java. Ils Seront insérées comme des membres de la Servlet et permet de définir des méthodes ou des données membres.

Exemples :

```
<%!
      int random4() {
             return (int)(Math.random() * 4);
      }
%>
```

```
<%!
      int nombreFetiche = 2;
%>
```

3.3. Expressions <%= %>

En fait expression Java qui renvoie un objet String ou un type primitif. C'est Un raccourci pour <% out.println(…); %>

`<%= XXX %>` **est équivalent à** **`<% out.println(XXX); %>`**

Attention au ;

Il est donc converti en out.println(…) dans la méthode _jspService(…) de la servlet.

```
La somme est:  <%= (195 + 9 + 273) %>
Je vous réponds à l'adresse :  <%= request.getParameter("email_address") %>
```

4. Objets prédéfinis dans une JSP

Il existe plusieurs objets qui peuvent être immédiatement utilisés dans une expression ou une scriptlet d'une JSP. Les plus utilisés sont:

- **out : le canal de sortie**
- **request (HttpServletRequest) : l'objet requête**
- **response (HttpServletResponse) : l'objet réponse**

Il y en a d'autres (Cf. mêmes objets dans une servlet)

Un exemple complet : **complet.jsp**

```
<html><head><title>JSP complet</title></head>
<body>
<%! String[] langages = {"Java", "C++", "PHP", ".net"};
    int random4() {
            return (int) (Math.random() * 4);
}
%>
<p>Parmi tous les langages orientés objets :</p>
<ol>
<%
     for (int i=0; i < langages.length; i++) { out.println("<li>" + langages[i]
+ "</li>");
    }
  %>
  <p>Le prochain que j'apprends est <b><%= langages[random4()] %> </b></p>
  </body>
```

5. Déboguer les JSP

La fenêtre de lancement du serveur Web donne des indications. Suivant les serveurs, une page HTML est retournée avec des indications.

Ces éléments sont très souvent relatifs à la servlet et pas à la page JSP.

Ils s'agit des directives `<%@ page errorPage= ...%>` et `<%@ page isErrorPage="true"%>`

Un page JSP peut référencer une page erreur par `<%@ page errorPage="page.jsp"%>`. La page erreur est indiquée par l'entête `<%@ page isErrorPage="true"%>`

Si une exception est levée le traitement est dérouté vers la page erreur qui connaît la référence exception qui repère l'exception.

Exemple :

Prenons l'exemple de la page `langages.jsp`

```
<%@ page errorPage="erreur.jsp"%>
<%! String[] langages = {"Java", "C++", "Smalltalk", "Simula 67"};
%>
<p>Parmi tous les langages orientés objets :</p>
<ol>
<%
    // levée d'une ArrayIndexOutOfBoundsException
    for (int i=0; i < 7; i++) { out.println("<li>" + langages[i] + "</li>");
}
%>
```

La page erreur.jsp peut être écrite comme suite :

```
<%@ page isErrorPage="true"%>
<html><body>
exception levée <b> <%= exception %> </b>
<hr>
<h3>trace de la pile</h3>
<pre>
<%
java.io.PrintWriter myWriter = new java.io.PrintWriter(out);
     exception.printStackTrace(myWriter);
%>
</pre>
</body></html>
```

Charger la page langages.jsp amène à une page d’erreur :

exception levée **java.lang.ArrayIndexOutOfBoundsException**

trace de la pile

```
java.lang.ArrayIndexOutOfBoundsException
        at jsp._0002fjsp_0002flangages_0002ejsplangages_jsp_0._jspServi
        at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.ja
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        at org.apache.jasper.servlet.JspServlet$JspCountedServlet.servi
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        at org.apache.jasper.servlet.JspServlet$JspServletWrapper.servi
        at org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServl
```

6. Enchaînement des pages JSP

Un page JSP peut en appeler une autre par

la directive : `<jsp:forward>`

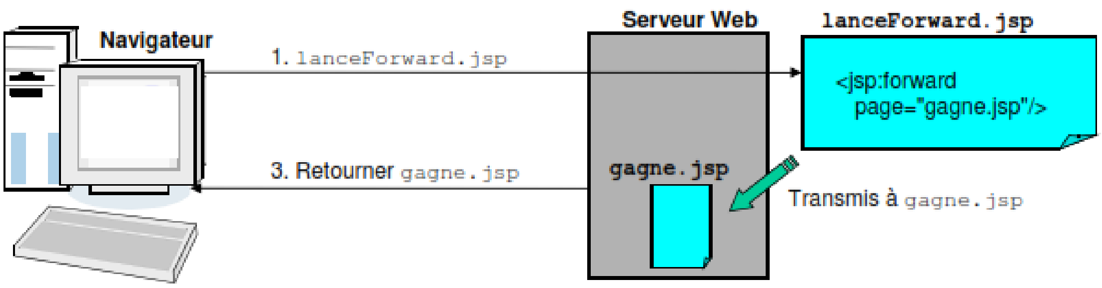
Syntaxe :

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

Exemple : une page JSP nommée lanceForward.jsp

```
<% String repUtilisateur = request.getParameter("repTextField");
   int rep = Integer.parseInt(repUtilisateur);
if ((rep % 2) == 0) {
%>
<jsp:forward page="gagne.jsp"/>
<% } else { %>
<jsp:forward page="perdu.jsp"/>
<% } %>
On n'affiche jamais cela
```

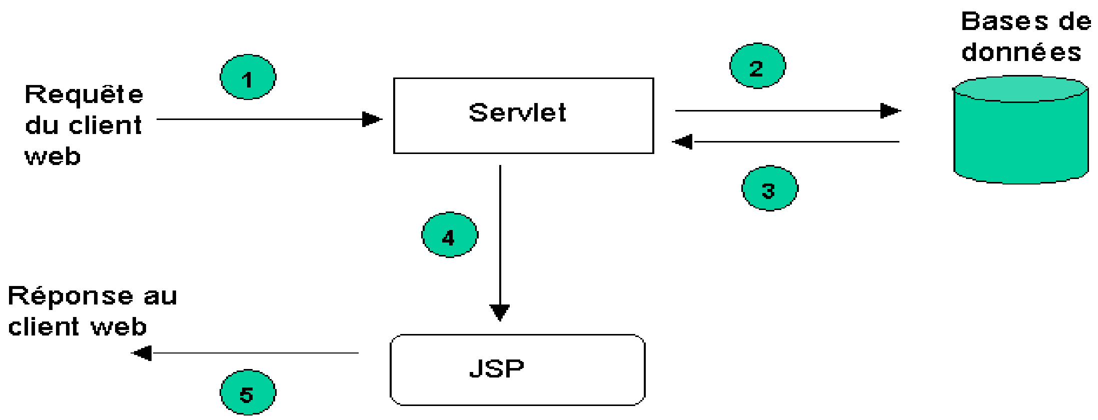
Après un <jsp:forward>, le traitement est entièrement pris en charge par nouvelle page.



Architecture MVC

Au niveau d'une application web Java EE la composition en une architecture dite MVC (Model, View, Controller) peut se faire comme suite :

- modèle = les données accédées par un code Java (JDBC, RMI, EJB, etc.)
- vues = JSP
- contrôleur = servlets



Ci-dessous un exemple de la syntaxe d'une servlet pour lancer la JSP :

```
public void doPost(HttpServletRequest request, HttpServletResponse response){
   ServletContext context = getServletContext();  // héritée de GenericServlet
   RequestDispatcher dispatcher =
context.getRequestDispatcher("/maPageMiseEnForme.jsp");
   dispatcher.forward(request, response);
}
```

- La servlet peut passer des valeurs à la JSP appelé grâce à `setAttribute()`.

```
public void doPost(HttpServletRequest request, HttpServletResponse response) {
     // appelle les méthodes sur les objets métiers
     ArrayList theList = // un objet à passer
     // ajoute à la requête
     request.setAttribute("nomDelObjet", theList);
     ServletContext context = getServletContext();
     RequestDispatcher dispatcher =
context.getRequestDispatcher("/jspAAppeler.jsp");
     dispatcher.forward(request, response);
}
```

- La JSP extrait les objets de `request` grâce à `getAttribute()`

```
<%  ArrayList theList = (ArrayList)
       request.getAttribute("nomDelObjet");
// maintenant, utiliser l'ArrayList
%>
```

Référence:

1. JavaServer Pages. Hans Bergsten; ed O'Reilly. ISBN 1-56592-746-X
2. Introduction aux Serlets et les JSP, Philippe Genoud, UGA