Langage JSP : PRINCIPES

Fortement imbriqué dans le code HTML, le langage JSP doit réaliser la partie dynamique d'un site web, c'est à dire :

- L'interrogation des bases de données en correspondance avec la requête envoyée, le parcours de ces bases et le remplissage des structures d'affichage (tableaux, listes, ...) en conséquence.
- Les opérations diverses de calcul, d'enregistrement des coordonnées du client...

Le code JSP est compris entre les balises <% et %>, il utilise la syntaxe du langage java (voir fiches correspondantes). Les fichiers auront l'extension .jsp et seront stockés sur le serveur. Ce dernier exécute le code correspondant lorsqu'il est sollicité par le serveur Web.

Une page JSP contient un certain nombre de balises qui seront détaillées plus loin. Notamment :

- <%= .... %> : reproduit l'information comprise entre les crochets sur la page Web construite
- <%! .....%> : déclare une variable dont la valeur existe tout au long d'une session en un seul exemplaire
- <%– .... –%> : commentaire qui ne sera pas reproduit dans la page Web construite (à distinguer du commentaire en HTML de la forme <!– ... –> qui lui sera envoyé au client)
- <%@ page .... %> : définit tout l'environnement d'exécution (quelle est la page d'erreur, quelles sont les classes java à inclure...)
- <jsp:include ...> : demande l'inclusion d'une autre page JSP (permet de n'écrire un bout de code qu'une seule fois et de le réutiliser dans plusieurs pages)
- <jsp:bean ... > : fait appel à un javaBean
- <jsp:forward ... > : redirige l'exécution vers une autre page JSP (principe du redirect)

Comme les autres langages de script, JSP permet de manipuler un certain nombre d'objets prédéfinis :

- request : correspond à la page qui a invoqué la page JSP (celle qui contenait le formulaire)
- response : correspond à la page qui est construite grâce au code JSP
- session : environnement de travail commun à toutes les pages d'une connexion d'un client.
- out : objet permettant l'écriture sur un flux de sortie (notamment la page construite par le script JSP).
- application : permet d'obtenir des informations sur l'application en cours d'exécution sur le serveur
- page : représente la page en cours d'interprétation (équivalent du this en java).

Comme le langage Java, JSP est extrêmement sensible à la casse (Majuscule / minuscule)

LA DIRECTIVE @

En début d'un fichier JSP, il est intéressant de définir le comportement de la page ou d'incorporer des instructions récurrentes à partir d'un autre fichier (par exemple un entête, un script JavaScript...)

Comportement de la page : <%@ page ...%>

Cette directive fournit de nombreuses et précieuses informations. Les options (voir un extrait dans le tableau ci-dessous) seront toutes comprises à l'intérieur de balises <%@ page ...%>.

OPTION	EXPLICATION	EXEMPLE
import	les paquetages java utilisés (certains le sont par défaut : java.lang.*, javax.servlet.*, javax.servlet.http.*, javax.servlet;jsp.*)	import="java.applet.*,java.io.*"
extends	la classe qu'étend ce JSP	extends HttpServlet
errorPage	page d'erreur vers laquelle rediriger les exceptions non interceptées dans le code JSP errorpage="CapteErreur.jsp	
isErrorPage	Indique si la page JSP est une page d'erreur (alors on peut utiliser l'objet exception sans le redéclarer) ou non	IsErrorPage=false (la valeur par défaut est false)
session	Indique que la page permet la gestion des objets session ou non (la valeur par défaut est true, on peut donc créer des session et les utiliser sans avoir à déclarer un objet de la classe HttpSession)	session = true
buffer	Indique la taille de données mise en mémoire tampon avant de les envoyer vers l'objet ServletResponse. S'exprime en kilo octets (kb)	buffer = 32 kb (valeur par défaut 8kb)
autoFlush	Indique si le buffer doit être automatiquement vidé lorsqu'il est plein ou si une exception doit être levée	autoFlush = true (valeur par défaut true)

Incorporation de fichier <%@ include ...%>

Lorsque des éléments de programmation doivent apparaître dans plusieurs pages JSP, il est possible d'écrire ce morceau de code dans un fichier JSP et de l'introduire dans un autre par l'usage de la balise

```
<%@ include file="nomfichier" %>
```

Ce fichier peut contenir toute forme de texte, aussi bien du javaScript que de l'HTML ou du JSP.

Le code ainsi inclus est analysé par le serveur d'application, contrairement à l'action <jsp:include ... > qui ne fait que recopier le texte.

LES ACTIONS

Elles consistent à faire réaliser à la page des actions de type inclusion de Bean, de fichiers, redirection de page...

QUAND CREER UNE PAGE JSP ?

- À chaque fois qu'une interaction est nécessaire avec l'utilisateur et qu'on lui demande d'envoyer des données pour produire un résultat.
  La page utilisera alors :
  - L'objet request pour récupérer les résultat de la page appelante lorsque leur nom est

connu.

```
//récupération de la valeur d'une information envoyée par la page appelante
          request.getParameter("txtnom")
```

- L'objet session pour avoir accès aux informations véhiculées de page en page ou pour définir des données à réutiliser plus tard

```
//création d'une variable de session nom en fonction d'une donnée envoyée
par la page appelante
         session.setAttribute("nom",request.getParameter("txtnom") )
//appel à une variable de session crée précédemment
                session.getAttribute("mot de passe").toString()
```

- Un objet Enumeration pour traiter les paramètres lorsque l'on ne connaît pas a priori leur nom (si cela dépend de ce qu'a validé l'utilisateur). On importera alors le package java.util.*.

```
//récupération de l'ensemble des valeurs envoyées par la page appelante
        Enumeration e = request.getParameterNames( );
//parcours des informations
      while (e.hasMoreElements()) {
        //récupération du nom  du prochain élément
        String name = (String)e.nextElement();
       //récupération de la valeur du paramètre connu sous son nom
        String value = request.getParameter(name);
 ..............
  } // fin de la boucle
```

- À chaque fois qu'un traitement intermédiaire est nécessaire (vérification de données, calculs de synthèse, choix de la page à suivre en fonction de données fournies).
  La page utilisera alors :
  - L'objet response pour définir la page vers laquelle on redirigera le traitement ou la clause <jsp:forward .../>

```
//renvoi vers une autre page
          response.setRedirect("Pageerreur.jsp")
         <jsp:forward  page="Pageerreur.jsp" />
```

COMMENT ACCEDER AUX BASES DE DONNEES

L'accès aux bases dans des pages JSP se réalise au travers d'une connexion JDBC/ODBC. Un pilote ODBC doit avoir été créé préalablement. Pour utiliser les fonctionnalités base de données de JSP, il est nécessaire d'importer le package java.sql.*. Toute manipulation sur une base de données peut lever une exception qu'il est nécessaire d'encadrer dans un ***try {… accès aux bases…} catch (…) {…}***.

```
//définit le pilote odbc à utiliser
    String url = "jdbc:odbc:cnx_etud";
//vérifie l'existence d'une connexion préalable (facultatif)
   Connection cnxion = (Connection)application.getAttribute("dbconn");
   if (cnxion == null) {
```

```
       //charge les outils d'accès aux bases
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     /établit la connexion au pilote ODBC/
      cnxion = DriverManager.getConnection(url, "", "");
     //établit que la connexion a eu lieu (facultatif)
      application.setAttribute("dbconn", cnxion);
```

- Lorsque l'on souhaite modifier le contenu d'une base (insertion, suppression, modification), on se contentera d'un objet Statement auquel on demandera d'exécuter une requête.

```
//création du texte de la requête à partir d'une donnée fournie par la page
appelante
String req = "Delete from employé where emp_nom='"
req +=  request.getParameter("txtnom") +"'";
//création d'un objet Statement sur la connexion établie précédemment
Statement stmt = cnxion.createStatement( );
// exécution de la requête
stmt.executeQuery(req) ;
```

- Lorsque l'on souhaite exploiter le contenu informatif de la base (sélection), on aura recours à un objet ResultSet que l'on parcourra.

```
//création d'un objet Statement sur la connexion établie précédemment
Statement stmt = cnxion.createStatement( );
//exécution de la requête et récupération du résultat dans un objet
ResultSet
ResultSet rs = stmt.executeQuery("Select * from employé") ;
//parcours du résultat
while (rs.next( ) ) {
    //accès à un champ par son nom
      rs.getString("emp_nom") ;
   //accès à un champ par son numéro (il faut commencer au numéro 1
    rs.getString(1) ;
} //fin de la boucle
```

- Lorsque l'on souhaite étudier ou afficher la structure de la base, on utilisera un objet ResultSetMetaData.

```
//création d'un objet Statement sur la connexion établie précédemment
Statement stmt = cnxion.createStatement( );
//exécution de la requête et récupération du résultat dans un objet
ResultSet
ResultSet rs = stmt.executeQuery("Select * from employé") ;
//récupération de la structure correspondant à la requête
ResultSetMetaData rsmd = rs.getMetaData( ) ;
//calcul du nombre de colonnes
int nbCols = rsmd.getColumnCount();
// parcours du ResultSetMetaData
<% for (int i=1; i<=nbCols ;i++) {
         //accès au nom de la colonne i
           rsmd.getColumnName(i)
```

```
        }//fin de la boucle
```

REUTILISATION AVEC JSP

Fonctionnement très pratique du langage, il offre deux possibilités pour réutiliser du code JSP ou JAVA écrit par ailleurs ou produit dans l'intention de s'en resservir.

- eSi des fonctions sont réutilisées à divers endroits, on les écrira dans une page spécifique en JSP et on inclura le code dans les pages qui doivent y faire référence par l'instruction include.

```
//permet de réutiliser le code contenu dans une autre page
           <jsp:include  page="PageFonctions.jsp" />
```

- Si des comportements plus complexes sont à réutiliser (par exemple, la connexion à une base de données, l'accès aux méthode d'un compte bancaire…) on écrira une classe java (.class) qui sera utilisée en tant que Bean.

```
//déclaration d'une instance d'un Bean
// id est le nom de l'instance, class donne le nom de la classe utilisée,
scope décrit la portée du Bean…
   <jsp:useBean  id="unCompte"   class="packageGesCompte.Compte
scope="session" />
```