JSP et JavaBeans

Dans le cadre du développement d'applications Web interactive, certaines tâches répétitives sont systématiquement réécrite dans les pages de scripts. Notamment, toute la partie correspondant à la connexion à une base de données (utilisation du pont JDBC::ODBC, connexion au pilote ODBC, ouverture de la base...) est d'une écriture lourde et peu lisible.

Du fait de la grande modularité des langages à objet, Java va permettre de créer des classes spécialement dédiées à cette utilisation ponctuelle et répétitive, permettant de rejeter le code dans la classe plutôt que d'avoir à l'inclure dans la page JSP. Les Java Beans sont ainsi nés.

PRINCIPE

Dans sa description, un Java Bean est juste un fichier .class ne comportant qu'une classe unique, pouvant hériter ou implémenter d'autres classes.

Le Java Bean sera instancié par le serveur d'application (servlet Tomcat par exemple) lorsqu'il est appelé par une action <jsp:useBean>. Ce n'est qu'à cet instant qu'il devient réellement un bean, par le fait qu'il est appelé par du code JSP. Aussi, n'importe quelle classe java peut devenir un bean (sous réserve que le fichier ne contienne qu'une seule classe ayant le même nom que le fichier .class).

Tout le reste n'est que Java !

L'ACTION USE BEAN

Elle autorise des manipulations riches des java bean, avec passage de paramètre ou instanciations d'objets non définis. Les paramètres sont les suivants :

Paramètre	Explication	Exemple
id	Donne le nom de l'objet qui sera instancié à partir de la classe et qu'on utilisera dans le code JSP	id=“monBean”
beanName	À utiliser si id n'est pas valorisé. Permet de définir un objet complexe sans faire référence à une classe particulière (on utilisera alors setProperty)	beanName=“unBean”
class	Donne le nom de la classe à instancier.	class=“paktage.AccesBase” –< on doit préciser le chemin complet de la classe, soit dans le classes du JDK, soit dans les classes du serveur d'applications (Tomcat)
scope	Indique la persistance de l'objet (page, request, session, application)	scope=“session” l'objet sera reconnu dans toutes les pages et pourra être repris grâce à la commande session.getParameter(“monBean”)
type	Complète l'attribut class	(?)

Les propriétés d'un Bean

En outre, useBean peut être complété par la définition d'un certain nombre de propriétés réutilisables tout au long de la persistance du Bean.

On écrira ainsi pour créer une propriété de nom numCli ayant pour valeur une information récupérée d'une autre page :

```
<jsp:useBean id="monClient" class="gestion.Client" scope="session">
    <jsp:setProperty name="numCli" value="<%
request.getParameter("numero")%>" />
</jsp:useBean>
```

Ou bien, on pourra créer un objet complexe sans faire référence à une classe (la classe java.bean.Bean sera alors utilisée).

```
<jsp:useBean beanName="identité">
    <jsp:setProperty name="nom" value=""/>
    <jsp:setProperty name="prenom" value=""/>
    <jsp:setProperty name="DateNais" value="" />
</jsp:useBean>
```

Il sera alors possible de redéfinir les valeurs des propriétés à tout instant, ou d'en obtenir les valeurs avec la clause jsp:getProperty.

Par exemple

```
Jsp:useBean beanName="identité">
   <jsp:getProperty name="nom"/>
</jsp:useBean>
```

EXEMPLE SIMPLE : LA CONNEXION A UNE BASE

Code du Bean

```
package dbAccess;
import java.sql.*;
import java.io.*;

public class DBconnexion {
  String chaineConnect    = "jdbc:odbc:";
  String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
  Connection db    = null;
  Statement stmt   = null;
  ResultSet result = null;

  public DBconnexion() {}

  public void connect(String dsn, String util, String pwd) throws Exception
```

```
{
    try {
      Class.forName(driver);
      db = DriverManager.getConnection(chaineConnect + dsn, util, pwd);
    }
      catch (Exception e) {e.printStackTrace();}
  }
  public ResultSet getResults(String sproc) throws Exception {
    stmt = db.createStatement();
    result = stmt.executeQuery(sproc);
    return result;
  }
public String recup(String col,ResultSet rs) {
    String error = "";
    String ret = "";
    try {ret  = rs.getString(col) ;}
    catch (SQLException e) {error = e.toString();}
    if (!error.equals(""))
      ret = "Erreur SQL : " + error ;
    return ret ;
  }
  public void execute(String sproc) throws Exception {
    stmt = db.createStatement();
    stmt.execute(sproc);  }
  public void close() throws Exception {
    result.close();
    result = null;
    stmt.close();
    stmt = null;
    db.close();
    db = null;
  }
public int getNombreLigne(String chaineSql) throws Exception {
    stmt = db.createStatement();
    result = stmt.executeQuery(chaineSql);
    int i = 0 ;
    while (result.next()) { i++;}
    return i;
  }
  public ResultSetMetaData getMetaDonnees(String sproc) throws Exception {
    stmt = db.createStatement();
    result = stmt.executeQuery(sproc);
    ResultSetMetaData rsmd = result.getMetaData();
    return rsmd;
  }
}
```

Code de la page JSP

```
<%@ page LANGUAGE="java" IMPORT="java.sql.*" %>
<jsp:useBean
```

```
  ID="db"
  SCOPE="page"
  CLASS="dbAcces.DBConnexion" />
<%
  db.connect(CNX_Services,"","") ;
  String req = "select * from SERVICES order by TXT_NOM_SERVICE" ;
  ResultSet RS = db.getResults(req) ;
  if (RS != null)
    {
      %>
      <TABLE BORDER="0">
      <%
        while (RS.next())
          {
            String id  = db.recup("ID_SERVICE",RS) ;
            String txt = db.recup("TXT_NOM_SERVICE",RS) ;
            %>
            <TR>
              <TD><B><%=id%></B></TD>
              <TD> > </TD>
              <TD><%=txt%></TD>
            </TR>
            <%
          } // fin while
      %>
      </TABLE>
      <%
    } // fin if
  db.close() ;
%>
```

EXEMPLE AVEC HERITAGE

```
//Classe qui construit une requête à partir de données fournies (table,
champs, question)
package dbAccess;
import java.io.*;
import java.sql.*;
public class BeanCreeRequete extends dbAccess.DBconnexion {
// propriétés privées
  private String reqSql ="";
  private String clauseSelect = "" ;
  private String clauseFrom = "";
  private String clauseWhere = "";
  private String clauseGroupBy = "" ;
  private String clauseHaving= "" ;
  private String clauseOrderBy = "";
public void ajoutWhere(String champ1, String champ2, String operation,
String synchro) {
    //on ajoute une nouvelle condition dans la clause WHERE
```

```
  if (clauseWhere.equals("")) {
        clauseWhere += " WHERE " + champ1 + " " + operation + " " + champ2 ;
}
    else {
        clauseWhere += " " + synchro + " " + champ1 + " " + operation + " "
+ champ2 ;    }
  }
  public void ajoutSelect(String colonne) {
    //on ajoute une nouvelle colonne dans la clause SELECT
  if (clauseSelect.equals("")) {
    clauseSelect += "SELECT " + colonne ;  }
  else  {
    clauseSelect += ", " + colonne ;  }
  }
  public void ajoutFrom(String nomTable) {
  //on ajoute une nouvelle table à la clause FROM
    if (clauseFrom.equals("")) {
  clauseFrom += " FROM " + nomTable ;    }
   else  {
    clauseFrom += ", " + nomTable ;}
  }
  public void ajoutGroupBy(String colonne) {
    //ajoute une colonne dans la clause Group By
    if (clauseGroupBy.equals("")) {
    clauseGroupBy += " Group by " + colonne ;  }
  else  {
    clauseGroupBy += ", " + colonne ;}
  }     public BeanCreeRequete() {  }

  private String codeSQL() {
    //on construit le contenu de la variable reqSql
    //Choix entre GROUP BY et ORDER BY
    if (clauseGroupBy.equals(""))
      {reqSql = clauseSelect + clauseFrom + clauseWhere + clauseOrderBy  ;}
    else
      {reqSql = clauseSelect + clauseFrom + clauseWhere + clauseGroupBy +
clauseHaving ;}
    return reqSql;
  };


public void ajoutOrderBy(String colonne) {
  //ajoute une colonne dans la clause Group By
  if (clauseOrderBy.equals("")) {
  clauseOrderBy += " Order by " + colonne ;   }
   else  {
  clauseOrderBy += ", " + colonne ;}
  }
  public void ajoutHaving(String champ, String valeur, String operation) {
    //ajoute une condition dans la clause Having
    if (clauseHaving.equals("")) {
```

```
        clauseHaving += " HAVING " + champ + " " + operation + " " + valeur
;          }
    else {
        clauseHaving += " AND " + champ + " " + operation + " " + valeur ;
}
  }
  public ResultSet getCurseur() throws Exception {
    //retourne la liste des données correspondants à la requête.
          if (reqSql.equals("")) {codeSQL();};
          //utilise la méthode getResults de la classe mère avec le code de
la requête constitué
      Statement stmt= db.createStatement();
      ResultSet rsTemp = stmt.executeQuery(reqSql);
       return rsTemp;
      }
  public String getSQL() {
    //retourne le texte de la requête SQL
    if (reqSql.equals("")) {codeSQL();};
  return reqSql;}
}
```

Le code JSP utilisant ce bean aura la forme

```
<%@ page language="java" import="java.sql.*" %>
<jsp:useBean
  id="reqBDD"
  scope="page"
  class="lvsn.BeanCreeRequete"
 />

<%   reqBDD.connect("cnx_etud","","");  //méthode définie dans la surclasse
DBConnexion
   reqBDD.ajoutFrom("Etudiant");
   reqBDD.ajoutSelect("E_groupe_NT");
   reqBDD.ajoutSelect("E_Nom, E_prenom, E_DateNais, E_caractere ");
   reqBDD.ajoutOrderBy("E_nom");
   reqBDD.ajoutOrderBy("E_groupe_nt");…
%>
```