Chapitre 4 : Accès aux bases de données avec JDBC

Introduction

- JDBC (Java DataBase Connector) est une API chargée de communiquer avec les bases de données en Java.
- Les classes et interfaces de l'API JDBC figurent dans le package java.sql :

```
import java.sql.*;
```

- JDBC peut être utilisé pour accéder à n'importe quelle base de données à partir de:
  - Simple application Java
  - Une servlet
  - Page JSP, …

Travail avec une base de données

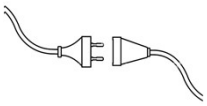
- JDBC permet de travailler avec les base de données de la même façon quelque soit leur fournisseur (Oracle, SQL Server, MySQL, PostgreSQL,…).
- Il suffit de télécharger la bibliothèque qui assure la communication entre Java et cette base de données.
- Cette bibliothèque s'appelle Driver ou Pilote ou Connecteur.
- Elle figure sur le site du fournisseur du SGBDR utilisé.
- Exemple de pilote MYSQL :
  - Le connecteur MySQL pour Java se nomme comme cet exemple : " **mysql-connector-java-8.0.27.jar**"

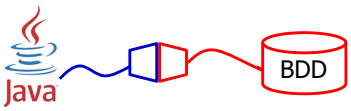
Etapes d'interaction avec une BDD

1. Chargement du pilote
2. Etablissement de la connexion
3. Création des objets encapsulant les requêtes
4. Exécution des requêtes
5. Parcours des résultats dans le cas d'une requête de sélection
6. Fermeture des objets résultats, requêtes et connexion

Chargement du pilote

- Pour se connecter à une base de données il faut charger son pilote.
- Le chargement se fait comme suit : Class.forName("nom_classe_acces_bdd");
- La documentation de la Bdd utilisée fournit le nom de la classe à utiliser.
- Exemple :
  - Dans le cas de la Bdd MySQL, ce chargement est comme suit : Class.forName(com.mysql.cj.jdbc.**Driver**) ou Class.forName(com.mysql.jdbc.**Driver**) selon les versions
- Une fois chargée, la classe JDBC qui se nomme **DriverManager** prend en charge le driver pour communiquer avec la base de donnée.





Classes de l'API JDBC

- Les classes et interfaces les plus usuelles sont les suivantes:
  - **DriverManager** (classe): charge et configure le driver de la base de données.
  - **Connection** (interface): réalise la connexion et l'authentification à la base de données.
  - **Statement** (interface): contient la requête SQL et la transmet à la base de données.
  - **PreparedStatement** (interface): représente une requête paramétrée
  - **ResultSet** (interface): représente les résultat d'une requête de sélection.

Etablissement de la connexion

- Pour se connecter à une base de données, il faut disposer d'un objet **Connection** créé grâce au DriverManager en lui passant :
  - l'URL de la base à accéder , Le login, Le mot de passe
- Exemples:
  - String url="jdbc:mysql://localhost/mydb"; // exemple URL BDD MySQL
  - String login="root";
  - String password="motdepasse";
  - **Connection** con=**DriverManager**.**getConnection**(url, login, password);

Exécution de requêtes SQL

- L'interface **Statement** permet d'envoyer des requêtes SQL à la base de données.
- Un objet Statement est créé grâce à un objet Connection de la façon suivante :

  Statement st = **con**.createStatement();
- Il possède deux méthodes :
  - **executeUpdate()** : Insertion, suppression, mise à jour.
    - int n= st.executeUpdate("**INSERT INTO Etudiant VALUES (3452,'Taha','Ali')**");
  - **executeQuery()** : Selection.
    - **ResultSet** res= st.**executeQuery**("**SELECT * FROM Etudiant**");

Requêtes avec paramètres

- L'interface **preparedStatement** permet d'envoyer des requêtes SQL à la base de données en prenant des paramètres.
- Ces paramètres sont représentés par des points d'interrogation(**?**) et doivent être spécifiés avant l'exécution.
- Exemple :
  - **PreparedStatement** p= con.**prepareStatement**("select* from Etudiant where cne=? And nom= ? ");

    p.**setInt**(1, 3452345); // remplissage 1er paramètre

    p.**setString**(2, "Alaoui"); // remplissage 2ième paramètre

    ResultSet resultats = p.**executeQuery()**; // exécution de la requête

Résultat d'une requête de sélection

- Une requête de sélection retourne un **ResultSet** qui est un ensemble d'enregistrements constitués de colonnes qui contiennent les données.
- Les principales méthodes :
  - **next()** : se déplace sur le prochain enregistrement : retourne false si la fin est atteinte. Le curseur pointe initialement juste avant le premier enregistrement.
  - **getInt(**int/String**)** : retourne le contenu de la colonne dont le numéro (resp. le nom) est passé en paramètre sous forme d'entier.
  - **getFloat(int/String)** : retourne le contenu de la colonne sous forme de nombre flottant.
  - **getDate(int/String)** : retourne le contenu de la colonne sous forme de date.
  - **Close()** : ferme le ResultSet

Résultat d'une requête de sélection

- Exemple :

```
ResultSet res= st.executeQuery("SELECT * FROM Etudiant");
while (res.next()) {
    System.out.println("CNE= "+res.getString(1)+" Nom= " +  res.getString(2)
                                    +" Prénom= "+res.getString(3));
}
res.close();
```


Enoncé TP

- Création de la base de données :
  - Utiliser MYSQL pour créer une base de données nommée « etudiantsDB » ayant une table ETUDIANT composée des champs :
    - CNE (Auto_incerement)
    - NOM
    - PRENOM
  - Remarque : Ne pas oublier de télécharger le connecteur MYSQL pour java et de le référencer dans votre projet.

Enoncé TP

- Création de l'application Web :
  - L'application web démarre avec une page d'accueil « index.jsp » qui se connecte à la base de données et affiche sous forme d'un tableau HTML le « cne », « nom » et « prénom » d'un étudiant ainsi que des liens d'ajout, de suppression de modification et de recherche d'un étudiant

Liste des Etudiants Inscrits :

CNE	NOM	PRENOM	operations	
14	Samiri	Hassan	supprimer	modifier
15	Tantawi	Taha	supprimer	modifier
16	Youssfi	Youssef	supprimer	modifier
17	Tahiri	Adil	supprimer	modifier

Ajouter Etudiant

Rechercher Etudiant : Rechercher

Enoncé TP

- La page de la recherche affiche les résultats sous forme d'un tableau HTML et affiche un lien de retour à la page d'accueil.

Liste des Etudiants Inscrits :

CNE	NOM	PRENOM	operations	
1	Samiri	Hassan	supprimer	modifier
2	Tantawi	Taha	supprimer	modifier
3	Youssfi	Youssef	supprimer	modifier
4	Tahiri	Adil	supprimer	modifier

Ajouter Etudiant

Rechercher Etudiant : ta Rechercher

Resultats de la recherche:

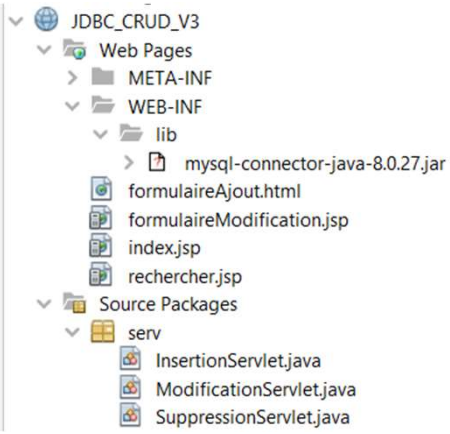
CNE	NOM	PRENOM	operations	
2	Tantawi	Taha	supprimer	modifier
4	Tahiri	Adil	supprimer	modifier

retour à la page d'accueil

- L'insertion, la modification et la suppression est confiée à des Servlets
- Le reste est confié à des pages JSP

Enoncé TP

- Le projet est réalisé avec **Netbeans** et dispose de la structure ci-dessous.



Solution : index.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1" import="java.sql.*"%>
<!DOCTYPE html>
<%
   String url="jdbc:mysql://localhost/etudiantsDB";
   String driver = "com.mysql.cj.jdbc.Driver";
   Class.forName(driver);
   Connection con;
   con = DriverManager.getConnection(url, "root", "");
   Statement stmt = con.createStatement();
   ResultSet rs = stmt.executeQuery("select * from ETUDIANT");
%>
<html>
   <head>
      <title>Affichage JSP</title>
   </head>
   <body>
      <h2> Liste des Etudiants Inscrits : </h2>
      <table border="1">
         <thead>
            <tr> <th>CNE</th> <th>NOM</th> <th>PRENOM</th> <th colspan="2">operations</th> </tr>
         </thead>
```

Solution : index.jsp

```
<tbody>
            <% while (rs.next()) {%>
            <tr> <td><%=rs.getString(1)%></td>
               <td><%=rs.getString(2)%></td>
               <td><%=rs.getString(3)%></td>
               <td><a href="SuppressionServlet?cne=<%=rs.getString(1)%>">supprimer</a></td>
               <td><a href="formulaireModification.jsp?cne=<%=rs.getString(1)%>">modifier</a></td>
            </tr> <%}%>
          </tbody>
       </table>
       <% rs.close();
          stmt.close();
          con.close();
       %>
       <a href="formulaireAjout.html"> Ajouter Etudiant </a>
       <form method="GET" action="rechercher.jsp">
        Rechercher Etudiant : <input type="text" name="rech">
          <input type="submit" value="Rechercher">
       </form>
    </body>
</html>
```


Solution : formulaireAjout.html

```
<html>
   <head> <title>formulaire</title> </head>
   <body>
      <form method="POST" action="InsertionServlet">
         Enregistrement d'un Nouvel Etudiant : <br>
         CNE : <input type="text" name="cne" value="AutoIncrement" disabled><br>
         Nom : <input type="text" name="nom"><br>
         Prénom : <input type="text" name="prenom"><br> <br>
         <input type="submit" value="Envoyer">
         <input type="reset" value="Effacer">
      </form>
   </body>
</html>
```

Solution : InsertionServlet.java

```
package serv;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;
import java.sql.*;
@WebServlet("/InsertionServlet")
public class InsertionServlet extends HttpServlet {
@Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      try {
            String fnom = request.getParameter("nom");
            String fprenom = request.getParameter("prenom");
```


Solution : InsertionServlet.java

```
      String url="jdbc:mysql://localhost/etudiantsDB";
      String driver = "com.mysql.cj.jdbc.Driver";
      Class.forName(driver);
       Connection con = DriverManager.getConnection(url, "root", "");
      Statement stmt = con.createStatement();
      stmt.executeUpdate("insert into ETUDIANT(NOM, PRENOM) values ('"+fnom+"','"+fprenom+"')");
      stmt.close();
      con.close();
      response.sendRedirect("index.jsp");
    }
   catch (ClassNotFoundException | SQLException e) {
         out.println("Erreur : " + e.getMessage() );
   }
 }
}
```

Solution : SuppressionServlet.java

```
package serv;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;
import java.sql.*;
@WebServlet("/SuppressionServlet")
public class SuppressionServlet extends HttpServlet {
@Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
   try{
   String fcne=request.getParameter("cne");
   String url="jdbc:mysql://localhost/etudiantsDB";
   String driver = "com.mysql.cj.jdbc.Driver";
   Class.forName(driver);
   Connection con = DriverManager.getConnection(url, "root", "");
```


Solution : SuppressionServlet.java

```
   Statement stmt = con.createStatement();
   int n = stmt.executeUpdate("delete from ETUDIANT where cne='"+fcne+"'");
   stmt.close();
   con.close();
   response.sendRedirect("index.jsp");
   }
   catch (SQLException|ClassNotFoundException e) {
         System.out.println("Erreur : " + e.getMessage());
   }
  }
}
```

Solution : formulaireModification.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1" import="java.sql.*"%>
<%  String fcne=request.getParameter("cne");
    String url="jdbc:mysql://localhost/etudiantsDB";
    String driver = "com.mysql.cj.jdbc.Driver";
    Class.forName(driver);
    Connection con = DriverManager.getConnection(url, "root", "");
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("select * from ETUDIANT  where cne='"+fcne+"'");
    rs.next();
%>
<html>
   <head> <title>formulaire Modification</title> </head>
   <body>
      <u>Formulaire Modification Etudiant</u>
      <form method="post" action="ModificationServlet?cne=<%=rs.getString(1)%>">
         CNE: <input type="text" name="cne" value="<%=rs.getString(1)%>" disabled><br>
         Nom : <input type="text" name="nom" value="<%=rs.getString(2)%>"><br>
         Prénom : <input type="text" name="prenom" value="<%=rs.getString(3)%>"><br> <br>
         <input type="submit" value="Envoyer">
      </form>
      <% rs.close();  stmt.close(); con.close();  %>
   </body>
</html>
```


Solution : ModificationServlet.java

```
package serv;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;
import java.sql.*;

@WebServlet("/ModificationServlet")
public class ModificationServlet extends HttpServlet {
@Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
     response.setContentType("text/html;charset=ISO-8859-1");
      try ( PrintWriter out = response.getWriter()) {
         String fcne = request.getParameter("cne");
         String fnom = request.getParameter("nom");
         String fprenom = request.getParameter("prenom");
         String url = "jdbc:mysql://localhost/etudiantsDB";
         String driver = "com.mysql.cj.jdbc.Driver";
         Class.forName(driver);
```

Solution : ModificationServlet.java

```
            Connection con = DriverManager.getConnection(url, "root", "");
            Statement stmt = con.createStatement();
            stmt.executeUpdate("UPDATE ETUDIANT SET NOM='"+fnom+"',PRENOM='"+fprenom+"'WHERE CNE='"+fcne+"'");
            stmt.close();
            con.close();
            response.sendRedirect("index.jsp");
        } catch (SQLException|ClassNotFoundException e) {
            System.out.println("Erreur : " + e.getMessage());
        }

    }
}
```


Solution: rechercher.jsp

```
<%@page contentType="text/html" pageEncoding="ISO-8859-1" import="java.sql.*"%>
<!DOCTYPE html>
<%  String rech=request.getParameter("rech");
    String url="jdbc:mysql://localhost/etudiantsDB";
    String driver = "com.mysql.cj.jdbc.Driver";
    Class.forName(driver);
    Connection con;
    con = DriverManager.getConnection(url, "root", "");
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("select * from ETUDIANT where cne like '%"+rech+"%' or nom like '%"+rech+"%' or prenom like
'%"+rech+"%' ");
%>
<html>
    <head>
        <title>Resultats de la recherche</title>
    </head>
    <body>
        <h2> Resultats de la recherche: </h2>
```

Solution: rechercher.jsp

```
<table border="1">
        <thead>
          <tr>
            <th>CNE</th> <th>NOM</th> <th>PRENOM</th> <th colspan="2">operations</th>
          </tr>
        </thead>
        <tbody>
          <% while (rs.next()) {%>
          <tr> <td><%=rs.getString(1)%></td>
            <td><%=rs.getString(2)%></td>
            <td><%=rs.getString(3)%></td>
            <td><a href="suppressionServlet?cne=<%=rs.getString(1)%>">supprimer</a></td>
            <td><a href="formulaireModification.jsp?cne=<%=rs.getString(1)%>">modifier</a></td>
          </tr> <%}%>
        </tbody>
      </table>
      <% rs.close();   stmt.close(); con.close(); %>
      <a href="index.jsp"> retour à la page d'accueil </a>
```


Métadonnées sur la base de données

- Classes pour obtenir des métadonnées:
  - **DatabaseMetaData** : informations à propos de la base de données : nom des tables, index, version ...
  - **ResultSetMetaData** : informations sur les colonnes (nom et type) d'un ResultSet
- Exemple :

```
ResultSetMetaData meta = res.getMetaData();
int nbCols = meta.getColumnCount();
while (res.next()) {
  for (int i = 1; i <= nbCols; i++) {
     System.out.print(meta.getColumnName(i)+" = "+res.getString(i) + " ");
  }
  System.out.println();
}
```

Transactions

- Une transaction est un ensemble de requêtes qui doivent s'exécuter d'un seul bloc.
- Si une requête de cet ensemble échoue alors toutes les autres sont annulées
- Par contre si toutes les requêtes réussissent alors l'ensemble est validé.

Validation automatique

- Par défaut, les connexions sont en mode de validation automatique (auto-commit) où chaque instruction SQL est considérée comme une transaction.
- La validation automatique peut être désactivée grâce à la méthode setAutoCommit()
- Exemple :
  - conn.setAutoCommit(false);// conn est un objet Connection

Commit & Rollback

- Une fois les modifications sont terminées, ils sont validées grâce à la méthode **commit ()** sur l'objet de connection
- Exemple :
  - conn.commit( );
- Si un problème se produit dans la suite des requêtes exécutées, alors l'ensemble peut être annulée grâce à la méthode rollback()
- Exemple :
  - Conn.rollback()

Commit & Rollback

- Exemple :

```
try{
   conn.setAutoCommit(false);
   Statement stmt = conn.createStatement();
  //requête correcte
   String requete1 = "INSERT INTO Etudiant VALUES (26, 'Alaoui', 'Ali')";
   stmt.executeUpdate(requete1);
   //requête fausse
   String requete2 = "INSERT INTOO Etudiant VALUES (27,'Omari', 'Omar')";
   stmt.executeUpdate(requete2);
   conn.commit(); // si il n y a pas d'erreur
}
catch(SQLException se){
   conn.rollback(); // si il  y a des erreurs
}
```

Points de sauvegarde

- Un point de sauvegarde, est un point de restauration logique dans une transaction.
- Si une erreur se produit après un point de sauvegarde, la méthode de restauration peut:
  - Annuler soit toutes les modifications,
  - Annuler uniquement les modifications apportées après le point de sauvegarde.

Points de sauvegarde

- L'objet Connection a deux nouvelles méthodes qui vous aident à gérer les points de sauvegarde :
  - setSavepoint(String savepointName) - Définit un nouveau point de sauvegarde et renvoie également un objet Savepoint.
  - releaseSavepoint(Savepoint savepointName) - Supprime un point de sauvegarde.

Points de sauvegarde

- Exemple :

```
try{
   conn.setAutoCommit(false);
   Statement stmt = conn.createStatement();
//requête correcte
   String requete1 = "INSERT INTO Etudiant VALUES (26, 'Alaoui', 'Ali')";
   stmt.executeUpdate(requete1);
// définition du point de sauvegarde
   Savepoint savepoint1 = conn.setSavepoint("Savepoint1");
   //requête fausse
   String requete2 = "INSERT INTOO Etudiant VALUES (27,'Omari', 'Omar')";
   stmt.executeUpdate(requete2);
   conn.commit(); // si il n y a pas d'erreur
}
catch(SQLException se){
   conn.rollback(savepoint1); // retour au point de sauvegarde en cas
d'erreur
}
```


Clés générées

- Lors de l'insertion des données dans deux tables en relation, la clé principale d'une table est insérée comme clé étrangère dans l'autre table.
- Si cette clé est de type Auto-Increment alors il est possible de la récupérer par la méthode getGeneratedKeys() de l'interface Statement.
- Ces clés gnénérées sont renvoyées sous forme d'un ResultSet :
  - ResultSet resultSet = statement.getGeneratedKeys();

Clés générées : Exemple

```
Connection connection = null;
try {
   Class.forName("com.mysql.cj.jdbc.Driver");
   String url = "jdbc:mysql://localhost:3306/your_database_name";
   String username = "your_username";
   String password = "your_password";
   connection = DriverManager.getConnection(url, username, password);
   connection.setAutoCommit(false);
   try {
            Statement statement = connection.createStatement();
            String table1Query = "INSERT INTO table1 (column1) VALUES ('value1')";
            statement.executeUpdate(table1Query, Statement.RETURN_GENERATED_KEYS);
            ResultSet resultSet = statement.getGeneratedKeys();
            int generatedKey = 0;
            if (resultSet.next()) {
               generatedKey = resultSet.getInt(1);
            }
            table2Query = "INSERT INTO table2 (column1, column2) VALUES ('value2', " + generatedKey + ")";
            statement.executeUpdate(table2Query);
            connection.commit();
   }
```


Clés générées : Exemple

```
            catch (SQLException e) {
                  connection.rollback();
                  e.printStackTrace();
            }
   } catch (ClassNotFoundException | SQLException e) {
      e.printStackTrace();
   } finally {
            try {
                     if (connection != null) {
                        connection.setAutoCommit(true);
                        connection.close();
                     }
            } catch (SQLException e) {
                      e.printStackTrace();
            }
   }
```