Compléments

À partir de la version 3 du JDBC:

On utilise un objet OracleDataSource qui définit un ensemble de méthodes permettant l'accès à la base de données Oracle. (utilisé dans le JNDI- Java Naming and Directory Interface).

1. Importer les packages

```
import java.sql.*;
import oracle.jdbc.*;
import oracle.jdbc.pool.*;
```

2. Créer un objet OracleDataSource : ods
3. Faire le «setting » des propriétés pour le ods avec les méthodes setURL, setUser..
4. Appeler la méthode getConnection de OracleDataSource pour ods

L'exception renvoyée est de type SQLException.

Compléments

getConnection()	Obtient une connexion à la base de données
setServerName	Définit le nom du serveur de la base de données utilisée. (ou son adresse IP)
setPortNumber	Définit le port du serveur de base s de données
setPassword	Définit le mot de passe avec lequel la connexion est obtenue.
setUser	Définit le username de l'usager
setServiceName	Définit le nom du service de la base de donnée : orcl
setDatabaseName	Définit le nom de la base de donnée : orcl
setURL	Définit la chaîne de connexion qui sera utilisée pour se connecter à la base de données. Elle est de la forme String url="jdbc:oracle:thin:@111.111.111.111:1521:orcl". Au lieu de définir l'IP , le protocole, le service name séparément, il est préférable d'utiliser l'URL.
getServerName	obtient le nom du serveur de la base de données utilisée. (ou son adresse IP)
getPortNumber	obtient le port du serveur de base s de données
getPassword	obtientt le mot de passe avec lequel la connexion est obtenue.
getUser	Obtient le username de l'usager
getServiceName	Obtient le nom du service de la base de donnée : orcl
getDatabaseName	Obtient le nom de la base de donnée : orcl

Compléments

Avec la version 4 du JDBC

Exemple:

```
String user1 ="user1";
String mdep ="oracle1";
String url="jdbc:oracle:thin:@205.237.244.251:1521:orcl";
Connection conn = null;
```

Dans le try

```
OracleDataSource ods =  new OracleDataSource();
ods.setURL(url);
ods.setUser(user1);
ods.setPassword(mdep);
conn = ods.getConnection();
```

catch(SQLException se)

Compléments

Avantages par rapport au DriverManager:

- Pas besoin d'enregistrer explicitement le driver
- Les setting de connexion sont beaucoup plus facile
- On produit un pool de connexions :
  - Au lieu d'obtenir une connexion physique à chaque fois que la methode getConnection() est appelée, on vérifie d'abord dans le pool de connexions s'il n' y a pas une connexion correspondante. Mécanisme plus rapide.

Definition:

- Un pool de connexions est un mécanisme permettant de réutiliser les connexions créées. En effet, la création systématique de nouvelles instances de *Connection* peut parfois devenir très lourd en consommation de ressources. Pour éviter cela, un pool de connexions ne ferme pas les connexions lors de l'appel à la méthode *close()*. Au lieu de fermer directement la connexion, celle-ci est « retournée » au pool et peut être utilisée ultérieurement.

Compléments

Types de ResultSet

1. **Lecture du ResultSet** :

Il existe 3 types de ResultSet:

- TYPE_FORWARD_ONLY: c'est le type par défaut. Le ResultSet n'est pas «Scrollable». La lecture se fait du premier au dernier enregistrement.
- TYPE_SCROLL_INSENSITIVE: le ResultSet est «Scrollable». On le parcours du début à la fin ou de la fin vers le début. On peut aussi se positionner sur une ligne en particulier avec les méthodes absolute ou relative. Le ResultSet n'est pas sensible aux changements effectués dans la base de données lorsque celui-ci est ouvert.
- TYPE_SCROLL_SENSITIVE: identique au précèdent. Et en plus lorsqu'il est ouvert et que la BD est mise à jour, les changements apparaissent dans le ResultSet.

Compléments

2. Modification du ResultSet

- CONCUR_READ_ONLY: c'est le type par défaut. Cela veut dire que le contenu du ResultSet ne peut être modifié par programmation
- CONCUR_UPDATABLE: des mises à jours peuvent se faire par le ResultSet.
- Les méthodes du ResultSet qui permettent de le modifier sont:
  - deleteRow();
  - updateRow();
  - insertRow(); dans ce cas, il faut se positionner à un emplacement vide du resultSet (en général au début) avec la méthode MoveToInsertRow()

Compléments

Dans le cas d'une modification (updatable), voici les opérations pour une mise à jour ou une insertion

Modifier la valeur du type et de la colonne donnée (par indice ou par nom) de l'enregistrement actuellement **pointé** :

- public void updateString(int indiceCol, String value);
- public void updateString(String nomCol, String value);
- public void updateInt(int indiceCol, Int value);
- public void updateInt(String nomCol, Int value);
- public void updateDouble(int indiceCol, double value);
- Etc… (selon les type de colonnes voir le tableau TYPE de données JDBC à la page 21).

Compléments

Exemple1:-- modification—

```
String sql2 = "select nom, prenom from employes";

Statement stm
=connexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
ResultSet rst4=stm.executeQuery(sql2);
rst.next();
rst.updateString("nom", "Coluche");
rst.updateString("prenom", "Moses");
rst.updateRow();
connexion.commit();
```

Compléments

Exemple 2

String sql = "select numemp, nomemp, prenomemp from employesbidon";

Statement stm1=connexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);

```
ResultSet Resultat = stm1.executeQuery(sql);
Resultat.moveToInsertRow();
Resultat.updateInt(" numemp ", 14);
Resultat.updateString(" nom ", "Fafard");
Resultat.updateString(3,  "Chantal");
Resultat.insertRow();
connexion.commit();
Resultat.first();
// affichage.
```

Compléments

Exemple 3

String sql = "select numemp, nomemp, prenomemp from employesbidon";

```
Statement stm1=connexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
  ResultSet Resultat = stm1.executeQuery(sql);
  Resultat.absolute(3);
  Resultat.deleteRow();
// suite du code
```

Compléments

Autres informations du ResultSet:

Le ResultsSet présente une interface qui permet d'avoir des informations concernant la structure des données qu'il contient. Ces informations sont appelées des méta-données.

L'interface qui obtient les méta-données du ResultSet est appelée: ResultSetMetaData.

Pour obtenir les «metadata» on utilise la methode getMetaData() du ResultSet.

Compléments

```
String sql2="select * from employesclg";


stm = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATAB
rst = stm.executeQuery(sql2);

ResultSetMetaData structure = rst.getMetaData();
int nbcolones = structure.getColumnCount();
System.out.println("Nbre de colonnes" + " " + nbcolones);

for(int i=1; i<=nbcolones; i++)

{
    System.out.println(structure.getColumnName(i)+ "\t\t" +
structure.getColumnTypeName(i));

}
```

Compléments

Les transactions:

Le traitement de transactions est une condition obligatoire pour toutes les applications qui doivent garantir la cohérence de leurs données permanentes. Le pilote JDBC d'Oracle, travaille en mode Autocommit. Il est possible de changer ce mode grâce à la méthode SetAutocomite et un boolean (true ou false) de l'objet connection .

Exemple:

```
connexion.setAutoCommit(false);
PreparedStatement stmupdate = connexion.prepareStatement(SQL);
stmupdate.setString(int, String);
stmupdate.setString(int, String);
stmupdate.executeUpdate();
connexion.commit();
```

- Pour annuler une transaction, il faut utiliser la méthode rollback() de l'objet connection connexion.rollback();

Parfois, il est souhaitable e de faire un rollback() jusqu'à un point de sauvegarde :

savepoint :

Compléments

```
String sqlup ="update employesclg set salaireemp = ? where numemp =?";
   PreparedStatement stmupdate = conn.prepareStatement(sqlup);
                stmupdate.setFloat(1,200);
                stmupdate.setInt(2,2 );
                stmupdate.executeUpdate();
                stmupdate.clearParameters();
                conn.setAutoCommit(false);
                Savepoint save1 = conn.setSavepoint();
                stmupdate.setFloat(1,10);
                stmupdate.setInt(2,3 );
                stmupdate.executeUpdate();
                conn.rollback(save1);
                stmupdate.clearParameters();
               conn.commit();
            stmupdate.close();
```

Compléments

L'interface RowSet:

- Contenu dans le package oracle.jdbc.rowset;
- Elle dérive de l'interface du ResultSet (hérite des mêmes méthodes).
- RowSet est updatable et scrollable. (peut régler le problème du ResultSet non scrollable avec appel de procédures)
- Peut fonctionner en mode connecté ou en mode déconnecté.
- Parfois, plus simple à utiliser qu'un ResultSet.
- Il existe plusieurs type de RowSet: OracleJDBCRowSet, OracleCachedRowSet
- Puiseurs constructeurs sont possibles.

Compléments

L'interface RowSet:

- OracleJDBCRowSet: toujours en mode connecté.

1- En utilisant le constructeur par défaut:

- La connexion est obtenue au moment de l'exécution de la commande, grâce au méthodes SetUrl, SetUsername, SetPassword
- La requête est envoyée grâce à la méthode SetCommand(string SQL)
- La Requête est exécutée grâce à la méthode Execute().

– Exemple:

```
OracleJDBCRowSet rowset = new OracleJDBCRowSet();
 rowset.setUrl(bd);
  rowset.setUsername(user);
  rowset.setPassword(mpasse);
  System.out.println("vous êtes connectés");
rowset.setCommand("SELECT numemp, nomemp, prenomemp FROM  employesclg");
  rowset.execute();
// Lecture comme le ResultSet.
```

Compléments

L'interface RowSet:

OracleJDBCRowSet:

2- La connexion est passée en paramètre.

- La connexion est faîte avec ods ou DriverManager . Elle est faîte avant la création de l'objet OracleJDBCRowSet
- La requête est passée par la méthode SetCommand (string SQL)
- La requête est exécutée par la méthode Excecute()

– Exemple :

```
OracleDataSource ods = new OracleDataSource();
//suite pour se connecter
conn = ods.getConnection();
OracleJDBCRowSet rowset = new OracleJDBCRowSet(conn);
rowset.setCommand("SELECT numemp, nomemp, prenomemp FROM employesclg where codedep = ?");
rowset.setString(1, "INF");
rowset.execute();
rowset.absolute(3);

// suite du code.
```

Compléments

L'interface RowSet:

- OracleCachedRowSet: fonctionne comme le OracleJDBCRowset sauf qu'il peut être en mode déconnecté
  - On peut avoir les deux modes de fonctionnement présentés pour le OracleJDBCRowSet.
  - on peut utiliser un ResultSet (avec un Statement, PreparedStatement ou CallableStatemnt).
  - Le CachedRowSet est «peuplé» par le ResultSet avec la méthode populate(ResultSet rst)
  - Le CachedRowset obtenu est Scrollable quelque soit la nature de ResultSet qui l'a peuplé.

Compléments

```
try {
            CallableStatement Callist = conn.prepareCall(" { ?= call empgest.lister(?)}");
            Callist.setString(2,"INF");
            Callist.registerOutParameter(1, OracleTypes.CURSOR);
            Callist.execute();

          ResultSet rstlist = (ResultSet) Callist.getObject(1);
          OracleCachedRowSet orarowset = new OracleCachedRowSet ();
          orarowset.populate(rstlist);
          System.out.println("_____ du début_____");
           while(orarowset.next())
                        {
                String nomemp = orarowset.getString(2);
                 String prenomemp =orarowset.getString(3);
                 System.out.println(nomemp + prenomemp);
                          }
```

Compléments

```
System.out.println("_____ De la fin _____");
 orarowset.afterLast();
            while(orarowset.previous())
             {
                 String nomemp = orarowset.getString(2);
                   String prenomemp =orarowset.getString(3);
                   System.out.println(nomemp + prenomemp);
               }

             Callist.clearParameters();
             Callist.close();
             rstlist.close();
             orarowset.close();
             System.out.println("Liste complété ");
```

Compléments

Exemple utilisant le mode deconnécté.

```
Statement stm = conn.createStatement();
   rst=stm.executeQuery(sql);

      OracleCachedRowSet rowset = new OracleCachedRowSet ();
      rowset.populate(rst);
      conn.close();
      System.out.println("----Le troisieme---");
      rowset.absolute(3);
      System.out.println(rowset.getString(2));
     stm.close();
     rst.close();
     rowset.close();
```

Compléments

Extraire les clés générées automatiquement :

Certaines tables, utilisent des séquences pour générer automatiquement les clés primaire (ou autre) lors des insertions. Il est possible de récupérer ces clés grâce à la méthode getGeneratedKeys() de l'objet Statement ou du PreparedStatement. il faudra alors indiquer au Statement (Ou au PreparedStatement) la colonne qui utilise les clés générées.

Limitations :

- On ne peut accéder aux informations du ResultSet obtenu que par l'index de colonne.

Compléments

```
String sqli = "insert into employesclg (numemp,nomemp ,salaireemp) values(SEQ1.nextval,?,?)";
ResultSet rst = null;
try {
String colonne[] = {"numemp"};
  // on indique la colonne qui utilise la séquence.
    PreparedStatement stm= conn.prepareStatement(sqli,colonne);
          int i =0;
          while (i<=10) // On insère 10 enregistrements.
             {
              stm.setString(1, "'Patoche");
             stm.setFloat(2,21);
             stm.executeUpdate();
             i++;
             rst = stm.getGeneratedKeys(); // A chaque insertion on obtient la clé généré automatiquement
                        while( rst.next())
                         {   System.out.println(rst.getInt(1));       }
              }
    rst.close();
   stm.close();
  }
             catch(SQLException seup)
        {
           System.out.println(seup.getMessage());
        }
```