

Java DataBase Connectivity

Emmanuel Fernandez
Juillet 2022

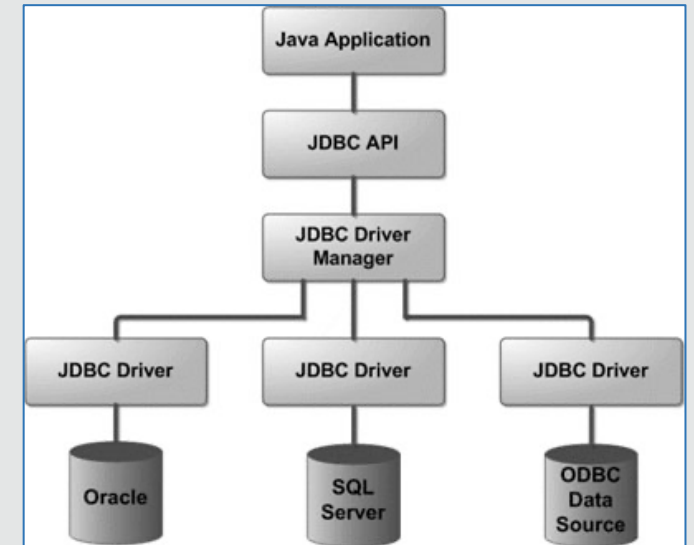
Objectifs

- Présentation de l'Api JDBC
- Notion de pilote/fournisseur JDBC
- L'utilisation des classes d'accès aux données
 - La connexion
 - Les commandes
 - Les jeux d'enregistrements
 - Les requêtes paramétrées
 - Les procédures stockées
- Les transactions
- Externaliser la chaine de connexion dans un fichier de configuration

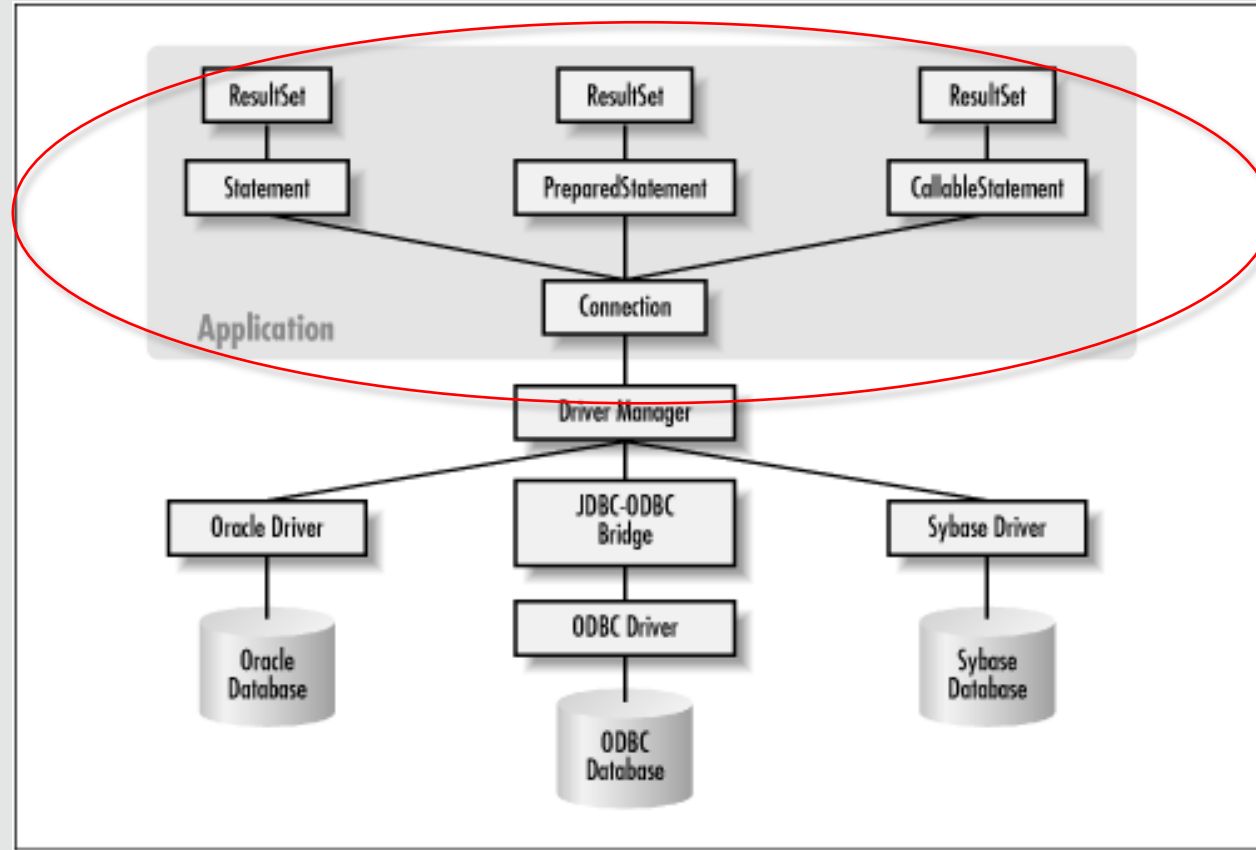
Java DataBase Connectivity

API JDBC

- 2 parties :
 - Interfaces du package java.sql
 - Un pilote JDBC
 - fournit généralement par le concepteur de la base de données
 - Implémente les interfaces du package java.sql



Java DataBase Connectivity Architecture



Pour démarrer

- Installer JDBC sur votre Machine
 - L'API JDBC est incluse dans le JDK
- Installer un pilote JDBC
 - Le pont JDBC - ODBC est installé automatiquement sous Windows
 - Télécharger le pilote JDBC si besoin
- Installer une base de données si besoin

Etablir une connexion

- Charger le pilote

```
Class.forName("oracle.jdbc.OracleDriver");  
  
//ou bien (recommandé par Oracle)  
  
DriverManager.registerDriver(new OracleDriver());
```

- Mettre en place la connexion

```
String url="jdbc:oracle:thin:@demeter:1521:orcl";  
  
Connection conn = DriverManager.getConnection(url,"admin","secret");
```

- Exécuter un ordre SQL

```
Statement stmt=conn.createStatement();  
  
stmt.executeUpdate("Insert into Coffees values ('Colombie',101,7,0,0)");
```

- Fermer la connexion

```
conn.close();
```

JDBC Statement (Commande)

- Création

- A partir d'une connexion valide

```
Statement stmt=con.createStatement();
```

- Récupération de données

```
stmt.executeQuery(requête)
```

- Modification des données

```
stmt.executeUpdate(requête)
```

Retrouver les valeurs depuis un SELECT

- Objet ResultSet

```
ResultSet rs=stmt.executeQuery("Select  cof_name,price from coffees");
```

- Parcours du résultat

```
while (rs.next()){  
    System.out.println(rs.getString("cof_name")+" "+rs.getFloat("price"));}
```


Les méthodes getXxx

- getByte
- getShort
- getInt
- getLong
- getFloat
- getDouble
- getBigDecimal
- getBoolean
- getString
- getDate
- getTime
- getTimestamp
- getAsciiStream
- getUnicodeStream
- getBinaryStream
- getObject

Se déplacer dans le ResultSet

■ Types de déplacement

- `ResultSet.TYPE_FORWARD_ONLY`
- `ResultSet.TYPE_SCROLL_INSENSITIVE`
- `ResultSet.TYPE_SCROLL_SENSITIVE`

■ Mode d'ouverture

- `ResultSet.CONCUR_READ_ONLY`
- `ResultSet.CONCUR_UPDATEABLE`

```
createStatement(type, mode);
```

Les déplacements possibles

Soit le resultSet rs:

```
rs.next()  
rs.previous()  
rs.afterLast(), rs.beforeFirst()  
rs.last(), rs.first()  
rs.absolute(numéroEnregistrement)  
rs.relative(nombreEnregistrements)
```

Les commandes paramétrées

- Créer un objet PreparedStatement

```
PreparedStatement cmdp= conn.prepareStatement ("update article set prix=?  
where refart like ?");
```

- Fixer les valeurs des paramètres

```
cmdp.setInt(1,10);  
cmdp.setString(2,"Velo");
```

Index du premier paramètre = 1

- Exécuter

```
cmdp.executeUpdate();  
cmdp.executeQuery();
```

Appel à une procédure stockée

- Créer un objet CallableStatement

```
CallableStatment cmdp= conn.prepareCall("{ ?=call insertArticle(?,?)");
```

- Fixer les valeurs des paramètres in

```
cmdp.setString(2,"Velo");  
cmdp.setInt(3,199.99);
```

- Définir les paramètres out

```
cmdp.registerOutputParameter(1,java.sql.Types.INTEGER);
```

- Exécuter

```
cmdp.execute();  
cmdp.executeQuery();
```

Les transactions

- Activer/ Désactiver le mode auto-commit

```
con.setAutoCommit(false);
```

- Gestion manuelle des transactions

- Début implicite
- Fin en succès ou échec

```
con.commit();
```

```
con.rollback();
```