02) Exploitation de données en Java (JDBC)

420-109-GG

LOTFI DERBALI

Pourquoi JDBC ?

Besoin d'une API Java pour :

- interagir avec des BD relationnelles :
  - exécuter des requêtes SQL
  - récupérer les résultats

- permettant de standardiser l'accès aux BD

Réponse de SUN à la pression des développeurs :

- JDBC (Java Java Data Base Connectivity)

Qu’est ce que JDBC ?

JDBC: Java DataBase Connectivity

- Une API (Application Programming Interface) Java permettant à un programme d’application en Java d’exécuter des instructions SQL pour accéder à une base de données relationnelle

Les étapes principales

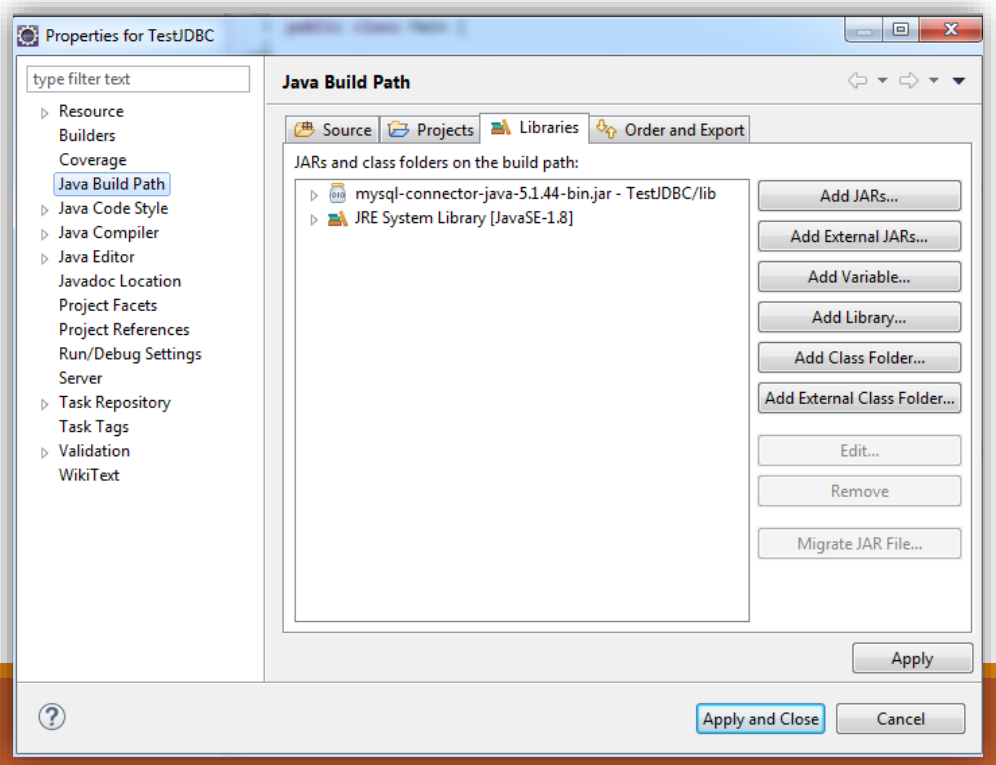
- Charger le pilote
- Établir une connexion à une base de données
- Envoyer une requête SQL
- Manipuler le résultat

JDBC: un driver (pilote) fournissant des outils pour ces fonctions

JDBC

Préparatif

- Télécharger le pilote Connector/J
  - https://dev.mysql.com/downloads/connector/j/
- Ajouter le pilote Connector/J à votre projet Eclipe



Base de données MySQL

Elle utilise un SGBD (Système de Gestion de Base de Données) pour manipuler les données

- Utilisation en ligne de commande
  - MySQL est un serveur qui utilise un système de connexion par login et mot de passe

- mysql –h <<host_name>> –u <<user_name>> –p
  - -h (host) spécifie le nom du serveur
  - -u (user) nom de l'utilisateur
  - -p (password) il faut saisir un mot de passe

MySQL: commandes

Utiliser d'une base de données

- `mysql> use <<database_name>>;`

Donner la liste des tables

- `mysql> show tables;`

Description d'une table

- `mysql> describe <<table_name>>;`

```
mysql> describe person;
+-------+----------+------+-----+---------+----------------+
| Field | Type     | Null | Key | Default | Extra          |
+-------+----------+------+-----+---------+----------------+
| id    | int(11)  | NO   | PRI | NULL    | auto_increment |
| name  | char(30) | YES  |     | NULL    |                |
| age   | int(11)  | YES  |     | NULL    |                |
+-------+----------+------+-----+---------+----------------+
3 rows in set (0.02 sec)
```

Étape 1: Charger le pilote

Charger le pilote (driver)

Pilote: contient toutes les classes nécessaire pour communiquer avec une base de données

- Il faut utiliser la méthode forName de la classe Class
- Par exemple (MySQL):

```
// registering MYSQL driver class
 try {
     Class.forName("com.mysql.jdbc.Driver").newInstance();
 } catch (ClassNotFoundException e) {
     // TODO Auto-generated catch block
     e.printStackTrace();
 } catch (IllegalAccessException e) {
     // TODO Auto-generated catch block
     e.printStackTrace();
 } catch (InstantiationException e) {
     // TODO Auto-generated catch block
     e.printStackTrace();
 }
```

Étape 2: Établir une connexion

Pour établir la connexion avec MySQL, il faut préciser

- le nom de la machine (ou son numéro IP),
- le port où le service MySQL est démarré (quasiment toujours 3306),
- le nom de la base de données,
- le login utilisé ainsi que son mot de passe.

```
localhost:3306/test_db?useSSL=false", "root", "root"
```

```
String protocole =  "jdbc:mysql:" ;
String ip =  "localhost" ;
String port =  "3306" ;
String nomBase =  "test_db" ;
String conString = protocole +  "//" + ip +  ":" + port +  "/" + nomBase ;
String nomConnexion =  "root" ;
String motDePasse =  "root" ;

con = DriverManager.getConnection(conString, nomConnexion, motDePasse) ;
```

Étape 3: Requête SQL

L’exécution d’une requête SQL passe par l'utilisation d'une classe, spécifique au pilote utilisé, implémentant l'interface Statement

Un objet de type Statement se doit d'être adapté à la base manipulée. JDBC ne fournit que l'interface Statement, qui est implantée par différentes classes dans un pilote

Obtenir un objet Statement avec la méthode createStatement.

```
Statement myStmt = myConn.createStatement();
```

Étape 3: Requête SQL

Exécuter une requête SELECT

- L'ordre SQL "SELECT * FROM table [WHERE …]"
- L'appel à "executeQuery" renvoie au final un objet de type ResultSet

```
ResultSet myRs = myStmt.executeQuery("SELECT * FROM person");
```

- La structure des ResultSet est très semblable à celle d'une Table dans une base de données relationnelle.

Étape 3: Requête SQL

Exécuter une requête INSERT / DELETE / UPDATE

- l'ordre SQL « INSERT INTO table VALUES …"
- l'ordre SQL « DELETE FROM table WHERE …"
- l'ordre SQL « UPDATE table SET … [WHERE …]"
- L'appel à "executeUpdate" retourne le nombre de lignes mises à jour / insérées

```
//execute update query
int numberOfRowsUpdated = myStmt.executeUpdate("UPDATE person set name='amy' where id=2");
```

Étape 4: Manipuler le résultat

En règle général:

- Copier les données dans une structure de données (Set, Vector, ...)
- Parcourir les résultats dans une boucle avec l'appel de la méthode next() de la classe ResultSet
- Extraire les colonnes avec des méthodes getXXX:
  - getInt, getFloat, getDouble pour Integer, Float et Double
  - getString pour char et varchar
  - getDate pour dates

```
while (myRs.next()) {
    System.out.println(myRs.getInt("id") + "   " + myRs.getString("name") + "   " + myRs.getInt("age"));
}
```