JDBC

JDBC pour Java DataBase Connectivity est une spécification d'API pour accèder à une base de donnée relationnelle depuis une application Java. Pour se connecter à une base de donnée, il faut un Driver, qui implémente JDBC pour un moteur de base de donnée relationnelle donné. Par exemple il en existe un pour Oracle, un pour MySQL, pour PostgreSQL, etc ...

Installation du Driver

Pour installer le Driver JDBC pour MySQL, ajoutez la dépendance suivante dans votre `pom.xml` :

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.22</version>
</dependency>
```

Pour Oracle utilisez cette dépendance :

```
<dependency>
  <groupId>com.oracle.database.jdbc</groupId>
  <artifactId>ojdbc8</artifactId>
  <version>21.1.0.0</version>
</dependency>
```

N'oubliez pas refresh vos dépendances Maven après avoir modifié votre `pom.xml` !

Connexion à la base de donnée

Pour se connecter à une base de donnée utilisez la méthode static `DriverManager.getConnexion()` en passant en paramètre la chaine de caractère de connexion ainsi que les identifiants.

Pour MySQL :

```
Class.forName("com.mysql.cj.jdbc.Driver")
Connection connexion =
DriverManager.getConnection("jdbc:mysql://localhost:3306/nomDeLaBase","root", "mot de passe");
```

Si MySQL vous lance une erreur de TimeZone, ajoutez ceci à la fin de l'URL JDBC : `?useLegacyDatetimeCode=false?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC`.

Pour Oracle :

```
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection connexion =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","root", "mot de passe");
```

Il est recommandé de faire un singleton de l'instance de la connexion à la base de donnée.

Interactions avec la base de donnée

Requêtes

L'objet `Statement` permet d'exécuter des requêtes SQL :

```
Statement statement = connexion.createStatement();
ResultSet resultSet = statement.executeQuery("SELECT * FROM Utilisateurs");
```

Le `ResultSet` est un itérateur sur les enregistrements contenues dans la réponse à la requête. On peut l'exploiter en itérant dessus grâce à la méthode `next()` :

```
while (rs.next()) {
  Utilisateur utilisateur = new Utilisateur();
  utilisateur.setPrenom(rs.getString("PRENOM"));
  utilisateur.setNom(rs.getString("NOM"));
  listeUtilisateur.add(utilisateur);
}
```

On peut en ensuite récupérer le contenu de l'enregistrement courant avec des méthodes comme `getString()`, `getInteger()`, etc ... en leur le passant le nom de la colonne dans la base.

Les classes `Statement` et `ResultSet` doivent toutes les deux êtres fermées après leur utilisation car il s'agit de ressources non managées. Pour une syntaxe concise on peut utiliser le "Try With Resource" :

```
try (Statement statement = this.connection.createStatement();
     ResultSet result = statement.executeQuery("SELECT * FROM Utilisateurs");) {
```

```
      while (result.next()) {
          Utilisateur utilisateur = new Utilisateur();
          utilisateur.setPrenom(rs.getString("PRENOM"));
          utilisateur.setNom(rs.getString("NOM"));
          listeUtilisateur.add(utilisateur);
      }
} catch (SQLException e) {
     System.err.println("Erreur à l'exécution de la requête SQL");
}
```

Requêtes DML

Pour modifier des données dans la base avec des commandes `INSERT`, `UPDATE` ou `DELETE` on utilise la méthode `executeUpdate()` de la classe `Statement` :

```
int modifiedRows = statement.executeUpdate("DELETE FROM Utilisateurs WHERE NOM = 'Shepard'");
```

`executeUpdate()` retourne le nombre de lignes modifiées par la requête.

SQL Dynamique

Il est possible de faire des requêtes préparées en utilisant la classe `PreparedStatement` :

```
PreparedStatement preparedStatement = connexion.prepareStatement("DELETE FROM Utilisateurs
WHERE NOM = ?");
```

On peut ensuite affecter des paramètres :

```
preparedStatement.setString(1,"Shepard");
```

Enfin, on exécute le `PreparedStatement` en utilise soit `executeUpdate()` soit `executeQuery()` selon qu'il s'agisse d'une commande DML ou d'une requête.

Revision #9
Created 2020-11-21 23:23:37 UTC by Arsène Lapostolet
Updated 2021-05-23 08:42:35 UTC by Arsène Lapostolet