



# www.dirasats.com

هذا الغلاف لا يعبر عن حقوق الملكية او فحوى الكتاب, فهو مجرد واجهة للموقع المحمل منه



شكرا لك على ثقتك بنا وعلى اختيار موقعنا

www.dirasats.com



من اجل تواصل معنا المرجو زيارة الموقع ستجد جميع المعلومات

www.dirasats.com

# Chapitre 6

---

## 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 donnée.
- ❑ Cette bibliothèque s'appelle Driver ou Pilote ou Connecteur.
- ❑ Elle figure sur le site du fournisseur du SGBDR utilisé.

## 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

- ❑ Dans ce cours nous allons prendre MySQL comme exemple.
- ❑ Le connecteur MySQL pour Java se nomme comme cet exemple : " **mysql-connector-java-5.1.23-bin.jar**"
- ❑ Pour se connecter à une base de données il faut charger son pilote.
- ❑ La documentation de la Bdd utilisée fournit le nom de la classe à utiliser.

## Chargement du pilote

- ❑ Le chargement se fait comme suit :  
`Class.forName("nom_classe_acces_bdd");`  
ou  
`Class.forName("nom_classe_acces_bdd").newInstance();`
- ❑ Exemple :
  - Dans le cas de la Bdd MySQL, ce chargement est comme suit : `Class.forName(com.mysql.jdbc.Driver)`
- ❑ 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 résultats d'une requête de sélection.

22/11/2019

cours JEE - Dr. Abdessamad Belangour

354

## 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
- Exemple avec MySQL:
  - `String url="jdbc:mysql://localhost/mydb";`
  - `String login="root";`
  - `String password="motdepasse";`
  - `Connection con=DriverManager.getConnection(url, login, password);`

22/11/2019

cours JEE - Dr. Abdessamad Belangour

355

## 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= stm.**executeQuery("SELECT \* FROM Etudiant")**;

22/11/2019

cours JEE - Dr. Abdessamad Belangour

356

## 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= ? ");**

Java - Dr A. Belangour

357

## Requêtes avec paramètres

```
p.setInt(1, 3452345);  
p.setString(2, "Alaoui");  
ResultSet resultats = p.executeQuery();
```

## Résultat d'une requête de sélection

- ❑ Une requête de sélection retourne un **ResultSet**
- ❑ ResultSet 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.

## Résultat d'une requête de sélection

---

- **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();
```



## Exemple complet

---

```
import java.sql.*;

public class Main {

    public static void main(String[] args) {

        String url="jdbc:mysql://localhost/etudiantsDB";
        String driver = "com.mysql.jdbc.Driver";
        try {
            Class.forName(driver).newInstance();
            Connection con=DriverManager.getConnection(url,"root","");
            Statement st = con.createStatement();
```

## Exemple complet

---

```
        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();
        st.close();
        con.close();
    }
```

## Exemple complet

---

```
catch (Exception e) {  
    System.out.println("Erreur : " + e.getMessage() + " source : " +  
        e.getStackTrace());  
}  
}  
}
```

## Métadonnées sur la base de données

---

- Il est possible aussi d'obtenir des informations sur la base de données grâce aux objets suivants :
  - **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

## Métadonnées sur la base de données

---

□ Exemple :

```
try {  
    ResultSetMetaData meta= resultats.getMetaData();  
    int nbCols = meta.getColumnCount();  
    while (resultats.next()) {  
        for (int i = 1; i <= nbCols; i++)  
            System.out.println(resultats.getString(i) + " ");  
    }  
    resultats.close();  
}  
catch (SQLException e) { //traitement de l'exception }
```