Base de données avec JDBC

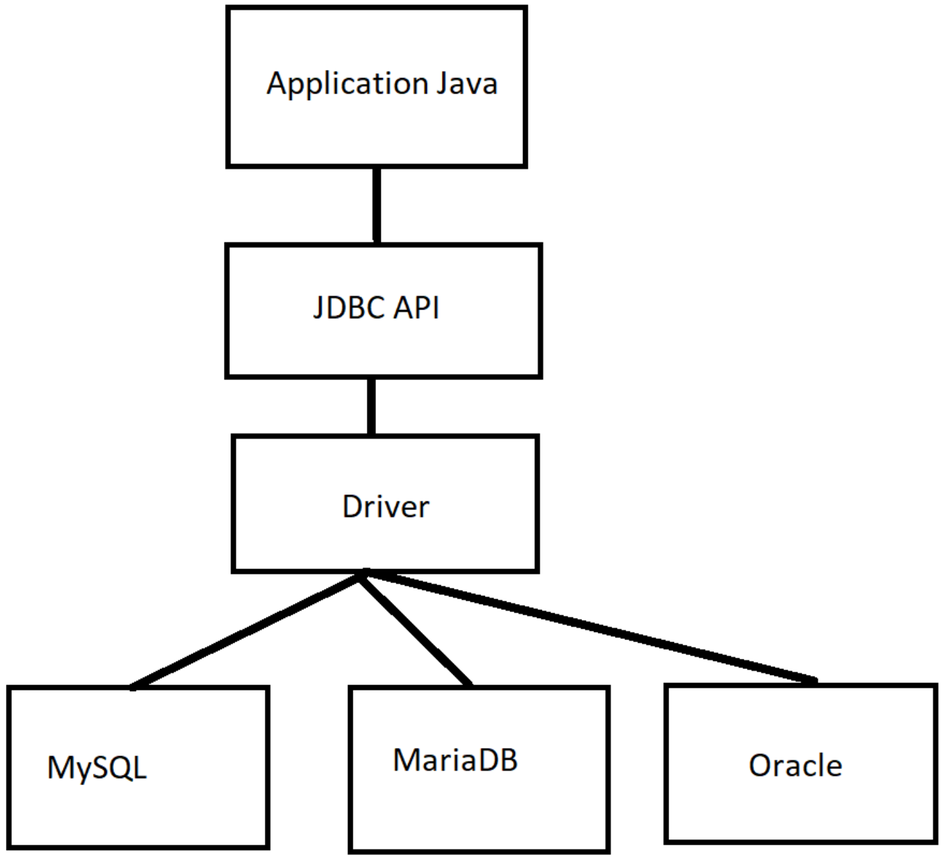
Dany Siriphol – CODA 1 ère année - 2025

JDBC

JDBC (Java Database Connectivity) est une librairie Java pour interagir avec les bases de données relationnelles.

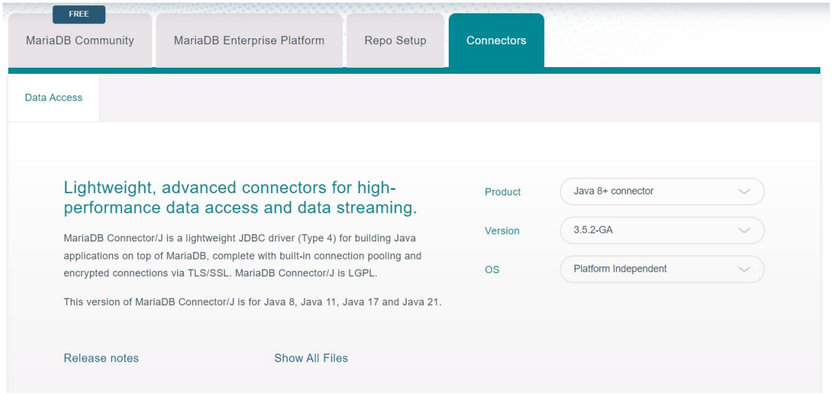
Elle fournit une API générique qui permet de faire des requete SQL indépendamment de la SGBD utilisée (MySQL, PostgreSQL, Oracle, etc.)

JDBC



Utilisation de JDBC

- Télécharger le driver de votre SGBD au format .jar
- Exemple MariaDB



- Ajouter le .jar à votre projet Java

Structure d’un programme JDBC

- Connexion à une base de données
- Création des requêtes SQL
- Exécution des requêtes SQL
- Récupération et traitement des données

Ouverture d’une connexion avec JDBC

```java
final String DB_URL = "jdbc:mariadb://localhost:3307/classicmodels";
final String USER = "root";
final String PASS = "";

try {
    Connection connection = DriverManager.getConnection(DB_URL, USER, PASS);

    connection.close();
} catch (SQLException e) {
    e.printStackTrace();
}
```

Exécuter une requête SQL

```
Connection connection = DriverManager.getConnection(DB_URL, USER, PASS);

Statement stmt = connection.createStatement();
String query = "SELECT * FROM employees";
ResultSet rs = stmt.executeQuery(query);

connection.close();
```

Récupération des résultats

```
Statement stmt = conn.createStatement();
String query = "SELECT * FROM employees";
ResultSet rs = stmt.executeQuery(query);

// Parcours toutes les lignes
while(rs.next()) {
    // traitement des lignes
}
```

Récupération des résultats

employeeNumber	lastName	firstName	extension	email
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com
1143	Bow	Anthony	x5428	abow@classicmodelcars.com
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com

```
while(rs.next()) {
    System.out.println(rs.getInt("employeeNumber"));
    System.out.println(rs.getString("lastName"));
}
```

Types Java vs SQL

SQL	JDBC/Java	setXXX
VARCHAR	java.lang.String	setString
CHAR	java.lang.String	setString
LONGVARCHAR	java.lang.String	setString
BIT	boolean	setBoolean
NUMERIC	java.math.BigDecimal	setBigDecimal
TINYINT	byte	setByte
SMALLINT	short	setShort
INTEGER	int	setInt
BIGINT	long	setLong
REAL	float	setFloat
FLOAT	float	setFloat
DOUBLE	double	setDouble
VARBINARY	byte[ ]	setBytes
BINARY	byte[ ]	setBytes

Requêtes paramétrées

Dany Siriphol – CODA 1 ère année - 2025

Requêtes préparées

Il est courant de créer des requete SQL dymaniques mais il ne faut pas les créer avec des contatenations

```
String name = "John";
String query = "SELECT * FROM employees WHERE name='" + name +"'";
```

Requêtes préparées

Une requête préparée est une requête qui contient des paramètres à fournir pour son exécution Elle sert en général à effectuer des concaténations dans une requête SQL

```
String query = "SELECT * FROM employees INNER JOIN"
        + " offices ON employees.officeCode = offices.officeCode "
        + " WHERE offices.country = ? "
        + " AND employees.lastName = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, "USA");
statement.setString(2, "Murphy");
ResultSet rs = statement.executeQuery();
```

Notions de securités : Critère du diplôme RNCP

Bloc(s) de compétences concernés	Code et intitulé de la certification professionnelle reconnue en correspondance partielle	Bloc(s) de compétences en correspondance partielle
RNCP37873BC01 - Développer une application sécurisée	RNCP31678 - TP - Concepteur développeur d'applications	RNCP31678BC01 - Concevoir et développer des composants d'interface utilisateur en intégrant les recommandations de sécurité

Exercices