digitalent
FORMATION

JDBC

Sommaire

- JDBC
- Requêtes Préparées
- Pilotage manuel de la transaction

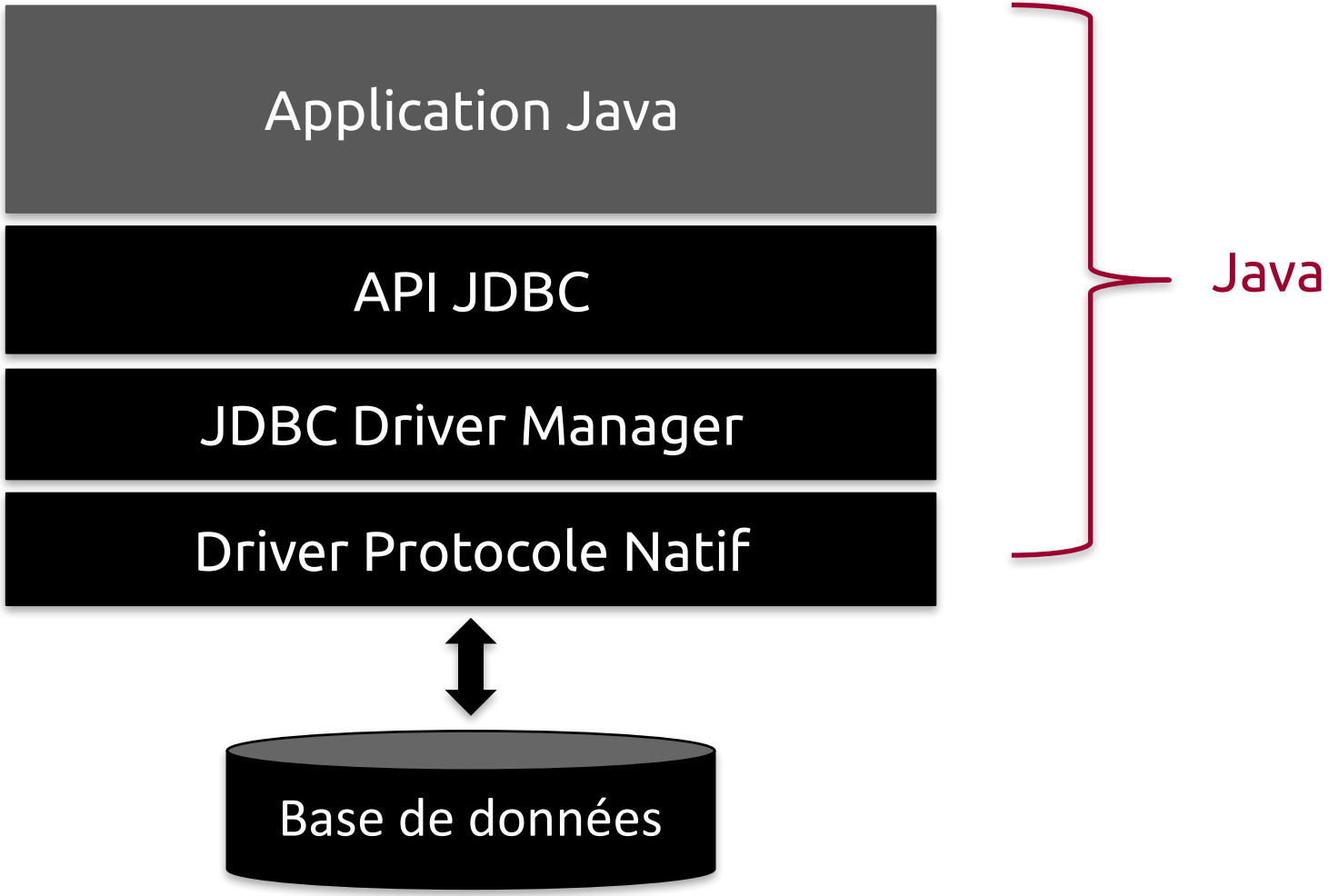
JDBC

JDBC ?

digitalent FORMATION

- Java DataBase Connectivity
- API Java permettant de communiquer avec une base de données relationnelles :
  - indépendamment du type de base de données utilisée (MySQL, Oracle, Postgres, ...)
- Packages → java.sql, javax.sql

JDBC ?



Pilote JDBC ?

digitalent FORMATION

- Un pilote JDBC est un composant logiciel qui permet à une application Java de communiquer avec une base de données relationnelle.

- Un pilote JDBC est une classe Java qui implémente l'interface java.sql.Driver
  - Exemple de driver : com.mysql.jdbc.Driver

java.sql.DriverManager

digitalent FORMATION

- Prend en charge le chargement des pilotes et permet de créer de nouvelles connexions à des bases de données.

- Tient à jour la liste principale des pilotes JDBC recensés du système.

Charger un pilote

digitalent FORMATION

Se fait par la méthode :

java.sql.DriverManager.registerDriver(**driver**)

manuellement

Usuellement : 1) Charger le driver

Class.*forName*(**"com.mysql.jdbc.Driver"**)

soit par un enregistrement

URL d'accès à une base de données

Respect un certains format

- Format

  jdbc:**[SOUS PROTOCOLE]**:**[COMPLEMENTS]**

  – jdbc : le protocole
  – [SOUS PROTOCOLE] : permet de distinguer le type de pilote
  – [COMPLEMENTS] : information de connexion à la base de données

- Exemple

2)
  – jdbc:mysql://localhost:8889/pizzadb
  – jdbc:postgres://localhost:8090/pizzadb

Se connecter à la base de données

3) Une fois que mon URL est donnée alors on peut se connecter

```
// Créer une connexion
Connection connection = DriverManager.getConnection(url);
Connection connection = DriverManager.getConnection(url,user,password);
```

Dialoguer avec la BDD

digitalent FORMATION

- Statement

```
Statement statement = connection.createStatement();
```

Instructions SQL	Méthode	Type retour	Valeur retour
SELECT	executeQuery	ResultSet	Lignes de résultat
UPDATE, INSERT, DELETE	executeUpdate	int	Nombre de lignes
Autres	execute	boolean	true si OK

INSERT, UPDATE, DELETE

digitalent FORMATION

Exemple:

```
// Supprimer toutes les pizzas
int nbPizzasSuppr = statement.executeUpdate("DELETE FROM PIZZA");
System.out.println(nbPizzasSuppr + " pizzas supprimés");

// Insérer une pizza
int nbPizzaInsere = statement.executeUpdate("INSERT INTO PIZZA(ID,NAME,PRICE)
VALUES(1,'Regina',12.0)");
System.out.println(nbPizzaInsere + " pizza inséré");

// Mettre à jour le prix d'un pizza
int nbPizzasMisAJour = statement.executeUpdate("UPDATE PIZZA SET PRICE=20.0 WHERE
ID=8");
System.out.println(nbPizzasMisAJour + " pizzas mis à jour");
```

creer une table s'est avec Exécute

SELECT

```
ResultSet resultats = statement.executeQuery("SELECT * FROM
PIZZA");

while(resultats.next()) {
    Integer id = resultats.getInt("ID");
    String name = resultats.getString("NAME");
    BigDecimal price = resultats.getBigDecimal("PRICE");
    System.out.println("[id=" + id + " name=" + name + " price=" + price
+ "]");
}
```

Travaux Pratiques

Requêtes Préparées

PreparedStatement

digitalent FORMATION

- Se créé à partir d'une connexion
- Contient une instruction SQL déjà compilée => gain de performance si plusieurs exécutions
- Peut contenir des paramètres (mis à jour avant l'exécution de la requête)

Exemple d'utilisation

```
PreparedStatement updatePizzaSt = conn.prepareStatement("UPDATE
PIZZA SET PRICE=20.0 WHERE ID=? AND PIZZA_NAME=?");
updatePizzaSt.setInt(1,1);
updatePizzaSt.setString(2, "Regina");
updatePizzaSt.executeUpdate();

PreparedStatement selectPizzaSt = conn.prepareStatement("SELECT *
FROM PIZZA WHERE ID=?");
selectPizzaSt.setInt(1,1);
ResultSet resultats = selectPizzaSt.executeQuery();
```

Voir exemple de Astrid

Transaction

Mode auto-commit

digitalent FORMATION

- Quand une connexion est créée, elle est par défaut en mode « auto-commit »
  - Chaque requête est traitée comme une transaction qui est validée à la fin de la requête.

- Pour désactiver le mode « auto-commit »
  - connection.setAutoCommit(false)

Gérer une transaction

digitalent FORMATION

- Pour valider une transaction
  - connection.commit()
- Pour annuler une transaction
  - connection.rollback()

Travaux Pratiques