Développement en JAVA

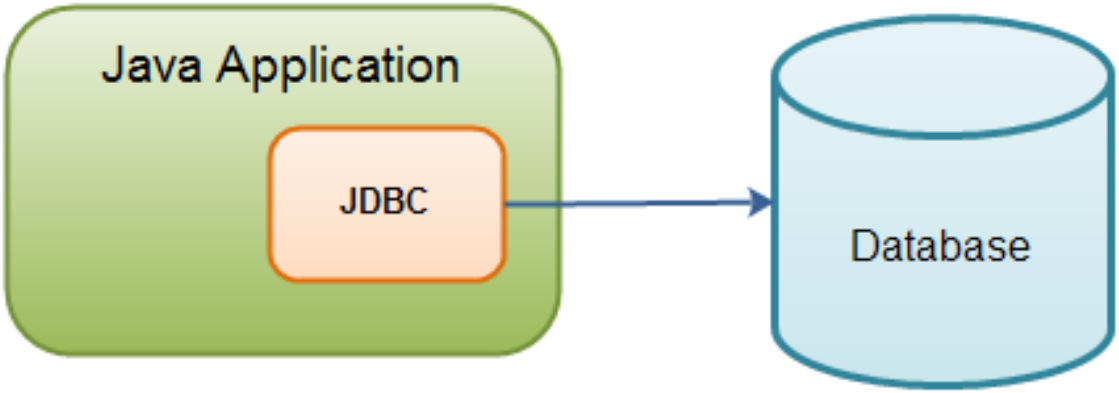
Connexion aux base de données

Agenda

- Le principe de JDBC
- Les classes de base
- Le pooling
- Les transactions

JDBC

- **J**ava **D**ata **B**ase **C**onnection
- **API** qui standardise l'accès à une base de données
- Une implémentation (Driver) existe pour chaque produit du marché (MySQL, PostgreSQL, Oracle…). L'interface est donc standard.



Configuration de JDBC

- JDBC est distribué sous la forme d'une library externe qu'il faut ajouter dans le buildPath du projet
- Afin de vérifier que le jar JDBC est bien présent dans notre projet, on peut forcer le chargement du Driver de la manière suivante:

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
System.out.println("JDBC driver successfuly loaded");
```

La classe DriverManager

- La classe `java.sql.DriverManager` s'occupe de gérer la configuration du driver JDBC
- La méthode `getConnection` permet d'ouvrir une connexion avec la BDD, elle attend en paramètre:
  - L'URL de la base
  - L'utilisateur
  - Le mot de passe
- L'object `Connection` obtenu doit **toujours être fermé** lorsque la connexion n'est plus nécessaire avec la méthode `close()`

La classe DriverManager

Exemple

```
Connection conn = null;
try{
 conn = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test", "root", "");
}catch(Exception e){
  // Gestion des exceptions
}finally{
   // Fermeture de la connexion dans tous les cas
  if(conn != null){
    conn.close();
  }
}
```

La classe Statement

- Un objet de classe `java.sql.Statement` est capable d'exécuter un requête sur la base de données
- Un objet de type `Statement` est obtenu auprès de la base de données
- Cette classe possède notamment une méthode `executeQuery(String query)` qui permet de lancer directement une commande SQL sur la BDD

```
Statement stmt = conn.createStatement();
ResultSet result = stmt.executeQuery("SELECT * FROM items");
```

La classe ResultSet

- La classe `java.sql.ResultSet` contient les résultats de la requête
- Elle se parcours à l'aide comme un Iterator à l'aide de la méthode `next()` qui permet d'avancer à la ligne suivante
- On peut ensuite récupérer les valeurs en précisant le nom de la colonne ou son indice dans la réponse (cf. documentation d'Oracle)
- Une exception de type `SQLException` peut être levée

La classe ResultSet

Exemple

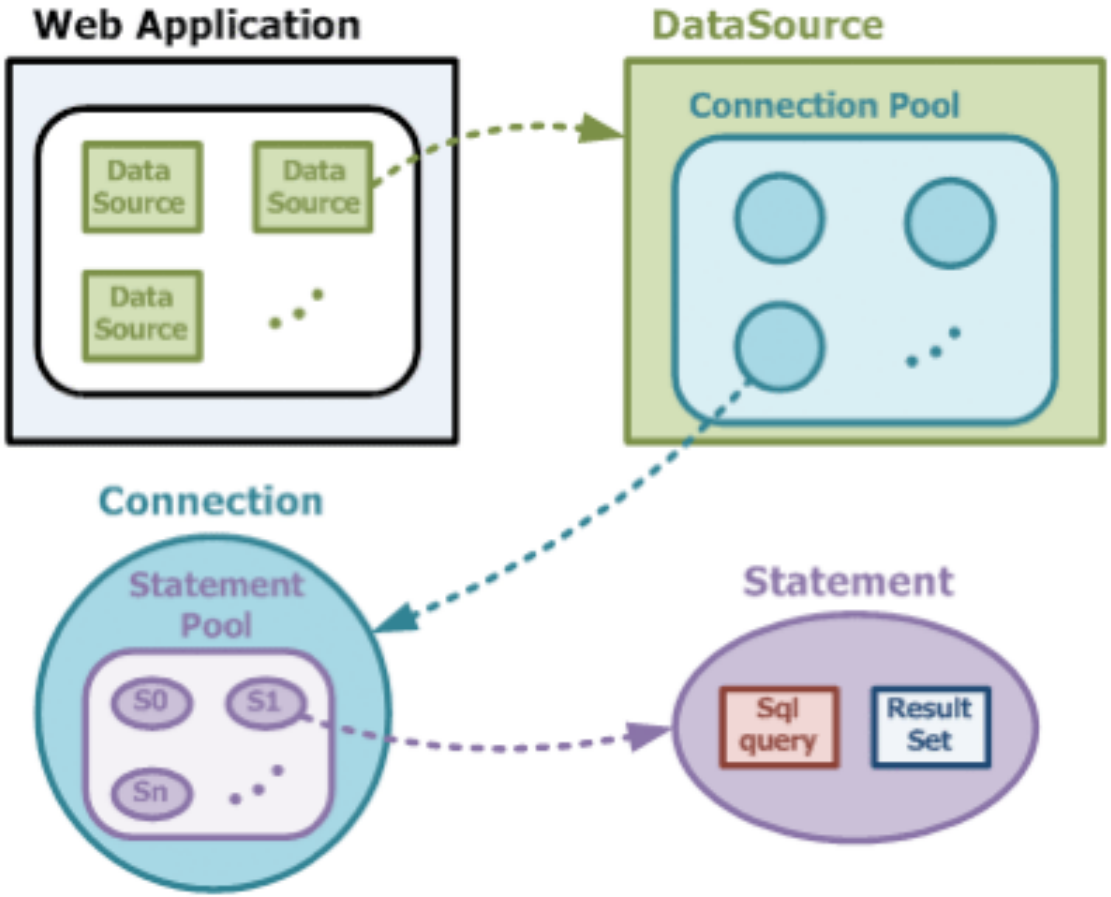
```
try{
ResultSet result = stmt.executeQuery("SELECT * FROM items");

// Affichage de toutes les lignes de la table items
while(result.next()){
  System.out.print(result.getInt(1) + result.getString(3) );
  System.out.print(result.getInt("id") + result.getString("name"));
}
}catch(SQLException e){
// Peut être par exemple déclenché si la requête est incorrecte
}
```

Pooling

- L'ouverture d'une connexion à la base de données est une opération qui peut s'avérer couteuse pour une application
- D'autre part une base de données n'autorise souvent qu'un faible nombre de connexion en parallèle par application afin de garantir qu'un programme n'utilise pas toute les connexions
- Afin de gérer ces deux aspects, il est conseillé de mettre un mécanisme de pooling
- La plus part des framework modernes (JEE, Spring) proposent de mécanisme de pooling

Pooling


Web Application
Data Source
Data Source
Data Source
DataSource
Connection Pool
Connection
Statement Pool
S0
S1
Sn
Statement
Sql query
Result Set

Les transactions

- La plus part des base relationnels sont transactionnels
- Une transaction est un ou plusieurs ordres envoyés à la BDD ensembles. Si un des ordres échoue, l'ensemble de la transaction est annulée
- Une transaction est considérée comme fiable si elle respecte les 4 propriétés ACID:
  - Atomicité
  - Cohérence
  - Isolation
  - Durabilité

Les transactions

- Par défaut JDBC effectue un commit (envoi de l'ordre en base) à chaque exécution de Statement
- Ce comportement doit être désactivé pour pouvoir bénéficier de la puissance des transactions

```
con.setAutoCommit(false);
```

- Lorsque l'ensemble des Statement constituant la transaction ont été exécuté, on effectue un commit

```
con.commit();
```