

Java DataBase Connectivity

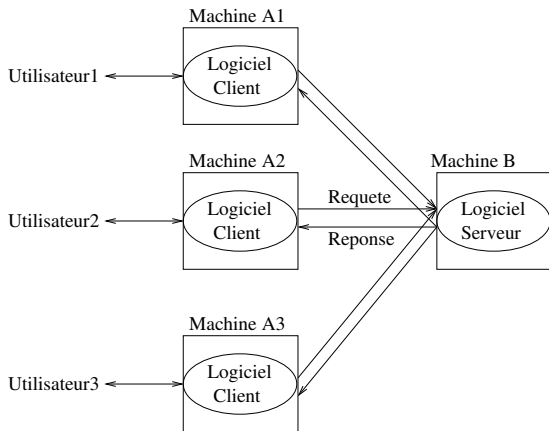


Philippe Mathieu & Guillaume Dufrene

IUT-A Lille

<http://www.iut-a.univ-lille.fr>

prenom.nom@univ-lille.fr



Un des axes les plus importants de l'informatique !
JDBC, comme le WEB, fonctionnent en mode Client-Serveur

Interfaces	Classes	Exceptions
Array Blob CallableStatement Clob Connection DatabaseMetaData Driver PreparedStatement Ref ResultSet ResultSetMetaData Statement ...	Date DriverManager DriverPropertyInfo Time Timestamp Types	BatchUpdateException DataTruncation SQLException SQLWarning

Décrite dans le paquetage `java.sql` avec java SE

- Toujours fourni sous la forme d'une archive jar
- Il y a un pilote spécifique pour chaque SGBD
- Enregistrer le pilote.

```
Class.forName("nom du driver");
```

```
Class.forName("org.sqlite.JDBC");
```

```
Class.forName("com.mysql.jdbc.Driver");
```

```
Class.forName("org.postgresql.Driver");
```

- C'est l'éditeur du serveur qui fournit le driver, mais le driver est installé sur le poste client !

Avant toute requête il faut qu'une connexion soit établie avec le SGBD et la base de données

- Etablir la connexion

```
Connection con =  
    DriverManager.getConnection(URL, Util, Mdp);
```

- L'URL de connexion dépend du Driver !

```
url = "jdbc:sqlite:cheminFichier";  
url = "jdbc:mysql://localhost:3306/mabase";  
url = "jdbc:postgresql://localhost:5432/mabase";
```

- Créer une requête pré-compilée

```
PreparedStatement ps=con.prepareStatement ("...");
```

- Exécuter la ou les requêtes

```
ResultSet rs= ps.executeQuery();
```

```
int nb = ps.executeUpdate();
```

- Attention à toujours bien fermer la connexion !

`con.close()` entraîne la fermeture des autres objets qui en dépendent.

Un premier exemple JDBC

Vider totalement la table `clients` !

```
import java.sql.*;

public class Efface
{
    public static void main(String args[]) throws Exception
    {
        // enregistrement du driver
        Class.forName("org.postgresql.Driver");

        // connexion à la base
        String url = "jdbc:postgresql://localhost/mabase";
        String nom = "paul";
        String mdp = "xxx";
        Connection con = DriverManager.getConnection(url,nom,mdp);

        // execution de la requete
        String query = "delete from clients";
        PreparedStatement ps = con.prepareStatement(query);
        ps.executeUpdate();

        // fermeture des espaces
        con.close();
    }
}
```

- La compilation n'a pas besoin de référence au driver

```
javac MaClasse.java
```

- L'exécution en a physiquement besoin

```
java -cp .:driver.jar MaClasse
```

ou

Mettre à jour la variable d'environnement `CLASSPATH`

```
export CLASSPATH=.:driver.jar
```

```
java MaClasse
```

Plus il y a de jars , plus la version `CLASSPATH` est pertinente.

- Chaque paramètre est identifié par un ? numéroté à partir de 1
- Des “setters” existent pour chaque type de donnée :
setInt, setString, setDate, setBlob, ...
- Ils simplifient et unifient la gestion des différents types

```
String query;  
query = "select * from personnes where age > ?";  
PreparedStatement ps=con.prepareStatement(query);  
ps.setInt(1,18);  
ResultSet rs = ps.executeQuery();
```

La méthode `executeQuery` renvoie un objet `ResultSet`

- `next()` permet de parcourir le `resultSet`.
Chaque appel avance d'une ligne. `true` tant qu'il y a des lignes.
- On lit les colonnes d'une ligne par des "getters" adaptés à chaque type de données : `getString`, `getInt`, `getDate`, ...
- Le paramètre peut être au choix :
 - ▶ Le nom de la colonne
 - ▶ le numéro de la colonne
- ```
String pds = rs.getString("poids");
```

# Un second exemple

Afficher la table Clients

```
import java.sql.*;

public class Select
{
 public static void main(String args[]) throws Exception
 {
 Class.forName("org.postgresql.Driver");

 String url = "jdbc:postgresql://localhost/mabase";
 String nom = "admin";
 String mdp = "xxx";
 Connection con = DriverManager.getConnection(url,nom,mdp);
 String query = "select NOM,PRENOM,AGE from CLIENTS";
 PreparedStatement ps = con.prepareStatement(query);
 ResultSet rs = stmt.executeQuery(query);

 System.out.println("Liste des clients:");
 while (rs.next())
 {
 String n = rs.getString("nom");
 String p = rs.getString("prenom");
 int a = rs.getInt("age");
 System.out.println(n + " " + p + " " + a);
 }
 con.close();
 }
}
```

La fermeture des connexions est capitale, que le programme se passe bien ou mal

```
Connection con=null;
try
{
 con = DriverManager.getConnection(url,nom,mdp);
 ...
}
catch (Exception e)
{
 // affichage ou traitement de l'erreur impératif
}
finally
{
 try {con.close();} catch(Exception e2) {}
}
```

- JDBC est une API définissant les accès aux SGBD
- Ce sont les drivers qui implémentent les interfaces de JDBC
- Le driver est fourni par l'éditeur du SGBD via un `jar`
- Pour exécuter une requête il faut :
  - ▶ Charger les classes du driver
  - ▶ Etablir la connexion
  - ▶ Définir la requête
  - ▶ Eventuellement, parcourir le `ResultSet`
  - ▶ Fermer la connexion