

JDBC API

- Présentation
- Etapes dans le traitement d'une requête d'accès aux données JDBC.
- Exemple

Présentation

- L'API JDBC est composée de deux Packages `java.sql` et `javax.sql`
 - `java.sql`: Package de base de l'API.
 - `javax.sql` : package optionnel étend les fonctionnalités de `java.sql` et permet le développement 3-tiers (l'API fonctionne côté serveur)
- JDBC 1.0 (1997: `java.sql`)
- JDBC 4.0 (2005: Draft)

Etapes dans une transaction d'accès aux données JDBC

1. Chargement du pilote JDBC
2. Connexion
3. Création d'une instruction
4. Exécution de la requête
5. Traitement des résultats
6. Fermeture de la connexion

Pilotes JDBC: Présentation

- Types de pilotes:
 - Type 1: passerelle JDBC-ODBC ("sun.jdbc.odbc.JdbcOdbcDriver")
 - Type 2: Pilote JDBC partiellement écrit en Java et repose sur une API propriétaire pour accéder aux données.
 - Type 3: Pilote JDBC 100% écrit en JAVA et utilise des protocoles réseau standards pour accéder aux données.
 - Type 4: similaire au type 3 sauf que les protocoles utilisés sont spécifiques au SGBD

Pour Télécharger un pilote JDBC:

<http://developers.sun.com/product/jdbc/drivers>

(Critères de recherche: Version JDBC, Version JDK, Editeur SGBD...)

Pilotes JDBC: Chargement

- Chargement du pilote: `Class.forName(« NomduPilote »)`
 - La méthode `forName` crée une instance de la classe du pilote JDBC et l'enregistre auprès du gestionnaire de pilotes JDBC.
 - Pour un pilote de type 1:
`Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`
 - Pour les autres types il faut utiliser le nom fourni avec le pilote, exemple:
`Class.forName("com.atinav.access.jdbc2.Driver"); .`
- Remarque: pour les pilotes JDBC version 4.0 Le chargement est automatique, donc il n'est pas nécessaire de faire appel à la méthode `Class.forName()`

Connexion

- La classe Gestionnaire de pilotes JDBC DriverManager contient la méthode getConnection(url,login,pass).

- Exemple: Accès ODBC à une base de données.

```
Connection con=DriverManager.getConnection("jdbc:odbc:galerie");
```

Création de l'instruction

- L'interface Statement permet la définition et l'exécution d'une instruction SQL
- Définition de l'instruction: `Statement inst=con.createStatement();`
- Remarque:
- L'objet ResultSet créé par défaut est de type Forward only
- `createStatement` Déclenche les exception suivantes `SQLException` et `SQLFeatureNotSupportedException`
- `createStatement(int type, int modif[, int maintien])`
 - Type de ResultSet :
 - `ResultSet.TYPE_FORWARD_ONLY`: déplacement en avant uniquement (par défaut)
 - `ResultSet.TYPE_SCROLL_INSENSITIVE` : déplacement arbitraire, le ResultSet présente une vue statique de la base de données (insensible aux modifications)
 - `ResultSet.TYPE_SCROLL_SENSITIVE`
 - Modif:
 - `ResultSet.CONCUR_READ_ONLY`: Le resultset est fermé après une validation
 - `ResultSet.CONCUR_UPDATABLE`: : les données de la base peuvent être modifiées via le resultset
 - Maintien du curseur:
 - `ResultSet.HOLD_CURSORS_OVER_COMMIT` : Le ResultSet reste ouvert après une validation implicite ou explicite.
 - `ResultSet.CLOSE_CURSORS_AT_COMMIT` : Les objets ResultSet sont fermés lorsqu'une validation est effectuée implicitement ou explicitement.

Exécution de la requête

- La méthode `executeQuery` d'un objet de type `Statement` exécute une requête SQL et retourne un tableau de données de type `ResultSet`, Exemple:

```
String sql="select * from articles";  
ResultSet rec=inst.executeQuery(sql);
```
- La méthode `executeUpdate`: Exécute une instruction SQL de type `INSERT`, `UPDATE` ou `DELETE` et retourne le nombre d'enregistrements affectés: `int executeUpdate(String sql)`

Traitement des résultats

Les enregistrements de ResultSet sont numérotés à partir de 1, Un objet de type ResultSet possède aussi un pointeur sur la ligne en cours.

Méthodes de l'interface ResultSet:

- boolean first(), next(), previous(), last()
- boolean absolute(int ligne)
- Array getArray(int i) : retourne une colonne dans un tableau
- Array getArray(String colName)
- void afterLast()
- void beforeFirst()
- cancelRowUpdates()
- void deleteRow()
- int findColumn(String NomColonne) : renvoie le numéro de la colonne.
- double getDouble(int col) : retourne le champ de la colonne col dans une variable de type Double.
- void updateDouble(String columnName, double x)

Fermeture de la connexion

Con.close()

Exemple(1)

```
/*Exemple de connexion à une base de données en utilisant un
*pilote ODBC
* * */

import java.sql. *;

public class Db1 {
    Db1() {
        try{
            //1- Chargement du pilote.
            Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver" );

            //2- Connexion
            Connection con =DriverManager .getConnection ("jdbc:odbc:galerie"
            );

            //3- Création de l'instruction
            Statement inst =con. createStatement ();

            //4- Exécution de la requête SQL
            String sql ="select * from articles" ;
            ResultSet rec =inst. executeQuery (sql);
```

Exemple(2)

```
// 5- Traitement des résultats
while (rec. next ()) {
    System .out. println (rec. getString (1) + " " + rec. getString
(2));
}

// 6- Fermeture de la connexion
rec. close ();
}

// Gestion des exceptions
catch (ClassNotFoundException e ) {
    System .out. println ("Pilote introuvable:" + e.getMessage () );
}

catch (SQLException e) {
    System .out. println ("Erreur de données:" + e.getMessage () );
} }

public static void main (String [] args ) {
    //Création d'une instance de la classe.
    new Db1 (); } }
```

Exemple(1)

```
/*Exemple de connexion à une base de données en utilisant un
 *pilote JDBC pour les bases de données de type MS ACCESS
 *Lien de téléchargement:
http://www.aveconnect.com/downloads.htm#1022
 * * */

import java.sql. *;

public class Db2 {
    Db1() {
        try{
            //1- Chargement du pilote.
            Class.forName("com.atinav.access.jdbc2.Driver");

            //2- Connexion
            Connection con =DriverManager .getConnection
            ("jdbc:atinav:localhost:7227:H:\\Atelier2\\Galerie.mdb");

            //3- Création de l'instruction
            Statement inst =con. createStatement ();

            //4- Exécution de la requête SQL
            String sql ="select * from articles" ;
            ResultSet rec =inst. executeQuery (sql);
```