JDBC Tutorial

JDBC اختصار ل Java Database Connectivity وهى بتقدملك طريقة موحدة لل Database access سواء خاصة ال RDBMS

اولا اعمل import لل java.sql
بتنقسم العملية لكذا خطوة
1- عمل Load للدريفر المناسب

```
Class.forName(somedriver)
```

2- الحصول على اتصال (Connection Object) بإستخدام Driver.getConnection()

ال getConnection بتاخد معاملات

1- url: مسار قاعدة البيانات
2- ال username: اسم المستخدم
3- ال password: كلمة السر الخاصة بال username

3- التعامل مع قاعدة البيانات من خلال Statement او PreparedStatement
للحصول على Statement object استخدم

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

وفيها عدة ميثودز زى

```
executeQuery();
```

لإرسال استعلامات مثل select

```
int executeUpdate();
```

ملحوظة ليها ريترن بال affected rows
لإرسال update, delete, insert مثلا..

```
int [] executeBatch();
```

لتنفيذ batch (مجموعة) من ال statements ورا بعض ودى بنحددها بإضافة

```
stmt.addBatch(sqlStmt)
```

للحصول على PreparedStatement استخدم

```
PreparedStatement pStmt=connection.prepareStatement(sqlStmt);
```

حيث ال sqlStmt دى هى اللى هتقوم بتعديل متغيراتها لاحقا
--مثال

```
String sqlStmt = "UPDATE users SET name=?, email=? WHERE id=?";
PreparedStatement pStmt = this.dbConnection.prepareStatement(sqlStmt);
pStmt.setString(1, newName);
pStmt.setString(2, newEmail);
pStmt.setInt(3, id);
pStmt.executeUpdate();
```

هنا انشأنا String بإسم sqlStmt

```
String sqlStmt = "UPDATE users SET name=?, email=? WHERE id=?";
```

لجملة ابديت زى ماشايفين لكن فيها 3 متغيرات مش حددنا ليهم قيمة ولكن عبرنا عنهم بعلامة استفهام
انشأنا PreparedStatement object مرتبطة بال sqlStmt اللى جهزناه

```
PreparedStatement pStmt = this.dbConnection.prepareStatement(sqlStmt);
```

ناقص نحدد المتغيرات دى تقدر تستخدم بعض الميثودز المساعدة مثلا setInt , setString وهى ميثودز

بتستبدل ال ? بقيمة ما سواء String او int ومعرفة كالتالى
.setString(idx, val);
لاحظ اننا بنبدأ اول index ب 1 مش 0 وبعد ماتستوفى كل المتغيرات تقدر تقوم بالتنفيذ

وال Stored Procedures ؟ الموضوع دا مرتبط اكتر ب SQL اغلب ال RDBMs بيدعمو ال StoredProcedure
على كل حال تقدر تستدعيه بإستخدام ال CallableStatement
و معاها execute مناسبة

```
CallableStatement cs = con.prepareCall("{call SHOW_USERS}");
ResultSet res= cs.executeQuery();
```

تحدد ال Auto Commit بإستخدام
setAutoCommit(bool);
فى حال انك عاملها false يبقة بعد اى تعديل تستدعى
connection.commit();
ال SavePoint
اتقدم مفهوم ال SavePoint مع ال JDBC 3.0 بحيث انك تحط زى mark (علامة) تقدر تعمل rollback ليها
بحيث تأمن على شغلك
كيفية الإستخدام

```
Savepoint sp1=connection.setSavepoint(savepointname);
connection.rollback(sp1)
```

تقدر تستخدم ال
connection.rollback()
لإلغاء كل مافى ال transaction الحالى

الحصول على النواتج؟النواتج او ال rows (الصفوف الناتجة) بترجع على صورة ResultSet Object
ايه رأيكم نتعرض لشرح برنامج بيتعامل مع داتابيز بإسم jdbctut فيها جدول بإسم users
معرف كالتالى

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL auto_increment,
  `name` varchar(40) NOT NULL,
  `email` varchar(40) NOT NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `name` (`name`,`email`)
)
```

ننشئ DBMgr class كالتالى

```
class DBMgr{
        private Connection dbConnection;
        private DatabaseMetaData dmd;
        private final String CATALOG="jdbctut";
        //Code omitted
}
```

هنا انشأنا كلاس بإسم DBMgr فيه 3 data members الأول dbConnection وهو اللى هيكون المسئول
عن الإتصال بقاعدة البيانات وال dmd فيه الميتاداتا الخاصة بالداتابيز وال CATALOG هو final بيعبر عن
اسم ال catalog وهنا jdbctut

```
   private static Connection getConnection(){
      try{
```

```
        Class.forName("com.mysql.jdbc.Driver");
        System.out.println("Loaded...");

    }catch(ClassNotFoundException cnfex){
        cnfex.printStackTrace();
    }
    try {
            String url="jdbc:mysql://localhost:3306/";
            return DriverManager.getConnection(url,"root", "");
    } catch (SQLException ex) {
            Logger.getLogger(DBMgr.class.getName()).log(Level.SEVERE, null, ex);
    }
    return null;
}
```

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

هنا بنعمل Load للدريفر ويمكن يتعمل throw ل ClassNotFoundException
فبنهندله فى ال catch block

```
            String url="jdbc:mysql://localhost:3306/";
            return DriverManager.getConnection(url,"root", "");
```

هنا بنحصل على اتصال بإسم ال url الخاص بالإنجن وهنا معناها اننا هنستخدم mysql الموجودة على localhost وبتشتغل على البورت 3306 من خلال ال JDBC

نجهز ال Constructor الخاص بال DBMgr

```
public DBMgr(){
    try {
        this.dbConnection = getConnection();
        this.dbConnection.setCatalog(this.CATALOG); //Setting the catalog..
        this.dmd = this.dbConnection.getMetaData();
        System.out.println("Connected..");
        this.inspectMeta();
    } catch (SQLException ex) {
        Logger.getLogger(DBMgr.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

بنحدد ال catalog اللى هيتعامل معاه ال dbConnection من خلال setCatalog

```
.setCatalog(catalog);
```

لاحظ اننا بنستدعى ال inspectMeta وهى ميثود صغيرة كتبناها لعرض معلومات عن الدرايفر

```
public void inspectMeta(){
    try {

        System.out.println("Driver: " + dmd.getDriverName());
        System.out.println("Driver Ver:" + dmd.getDriverVersion());
    } catch (SQLException ex) {
        Logger.getLogger(DBMgr.class.getName()).log(Level.SEVERE, null, ex);
```

```
    }
}
```

بتعرض اسم الدرايفر وإصداره.. فى اكتر من كدا طبعا او تحديدا كل مايتعلق بالميتاداتا!

عايزين نعمل implement لعمليات CRUD الأساسية فى ال DBMgr

CRUD: Create Read Update Delete

insert -1

```
public void insertRecord(String name, String email){
    try {
        //No validations for sake of simplicity.
        String sqlStmt="INSERT into users (name, email) VALUES (?, ?)";
        PreparedStatement pStmt = this.dbConnection.prepareStatement(sqlStmt);
        //1-based indexing.
        pStmt.setString(1, name);
        pStmt.setString(2, email);
        pStmt.executeUpdate();


    } catch (SQLException ex) {
        Logger.getLogger(DBMgr.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

عندنا هنا بإسم insertRecord بتاخد معاملات name, email

جهزنا ال Insert statement

```
        String sqlStmt="INSERT into users (name, email) VALUES (?, ?)";
        PreparedStatement pStmt = this.dbConnection.prepareStatement(sqlStmt);
```

جهزنا متغيراتها

```
        pStmt.setString(1, name);
        pStmt.setString(2, email);
```

نفذنا ال update بإستخدام

```
        pStmt.executeUpdate();
```

delete -2

عندنا هنا ميثودين للحذف واحدة بإستخدام ال id والتانية بإستخدام ال name

الحذف بإستخدام ال id

```
public void deleteRecordByID(int id){

    try {
        String sqlStmt = "DELETE FROM users WHERE id=?";
        PreparedStatement pStmt = this.dbConnection.prepareStatement(sqlStmt);
        pStmt.setInt(1, id);
        pStmt.executeUpdate();
    } catch (SQLException ex) {
        Logger.getLogger(DBMgr.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

نفس الفكرة بنجهز ال statement

```
String sqlStmt = "DELETE FROM users WHERE id=?";
PreparedStatement pStmt = this.dbConnection.prepareStatement(sqlStmt);
```

ونجهز متغيراتها

```
pStmt.setInt(1, id);
```

ننفذ ال update

```
pStmt.executeUpdate();
```

الحذف بإستخدام ال name

```
public void deleteRecordByName(String name){
   //UNIQUE..
   try {
      String sqlStmt = "DELETE FROM users WHERE name=?";
      PreparedStatement pStmt = this.dbConnection.prepareStatement(sqlStmt);
      pStmt.setString(1, name);
      pStmt.executeUpdate();
   } catch (SQLException ex) {
      Logger.getLogger(DBMgr.class.getName()).log(Level.SEVERE, null, ex);
   }

}
```

هنا مش هنواجه مشكلة بما ان الإسم معرف على انه UNIQUE

3- ال update

```
public void updateRecord(int id, String newName, String newEmail){
   try {
      String sqlStmt = "UPDATE users SET name=?, email=? WHERE id=?";
      PreparedStatement pStmt = this.dbConnection.prepareStatement(sqlStmt);
      pStmt.setString(1, newName);
      pStmt.setString(2, newEmail);
      pStmt.setInt(3, id);
      pStmt.executeUpdate();
   } catch (SQLException ex) {
      Logger.getLogger(DBMgr.class.getName()).log(Level.SEVERE, null, ex);
   }
}
```

هنا بيتم تعديل record معين بإستخدام ال id

نجهز ال statement

```
String sqlStmt = "UPDATE users SET name=?, email=? WHERE id=?";
PreparedStatement pStmt = this.dbConnection.prepareStatement(sqlStmt);
```

نجهز متغيراتها

```
pStmt.setString(1, newName);
pStmt.setString(2, newEmail);
pStmt.setInt(3, id);
```

ننفذ ال update

```
pStmt.executeUpdate();
```

4- read

عايزين نبقة نعرض كل ال records اللى فى ال users catalog

```
public void viewRows(){
    try {
        String sqlStmt = "SELECT * FROM users ORDER BY id ASC"; //ASC is the default.
        //Ordered
        Statement stmt = dbConnection.createStatement();
        ResultSet res = stmt.executeQuery(sqlStmt);

        //First print the columns.

        ResultSetMetaData rsmd=res.getMetaData();
        int nOfCols=rsmd.getColumnCount();
        for(int i=1; i<=nOfCols; i++){
            if(i==1){
                System.out.print(String.format("%5s",rsmd.getColumnName(i)));
                continue;
            }
            System.out.print(String.format("%15s",rsmd.getColumnName(i)));
        }
        System.out.println();
        while(res.next()){
            //ID Name Email ...
            //It's better to fetch by the column name..
            int id=res.getInt("id");
            String name=String.format("%20s",res.getString("name"));
            String email=String.format("%20s",res.getString("email"));

            System.out.println(id +name +email);
        }
    } catch (SQLException ex) {
        Logger.getLogger(DBMgr.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

اوكى مش تتخض

1- ال SQL statement للإستعلام عن ال records

```
        String sqlStmt = "SELECT * FROM users ORDER BY id ASC"; //ASC is the default.
```

2- ننشئ statement object وننفذ ال sqlStmt بإستخدام executeQuery واللى ناتجها هيكون ResultSet

```
        Statement stmt = dbConnection.createStatement();
        ResultSet res = stmt.executeQuery(sqlStmt);
```

3- محتاجين نطبع اسامى ال columns اللى اتعمل عليها الإستعلام .. نقدر نحصل عليها من خلال ال ResultSetMetaData object اللى تقدر تحصل عليه من خلال

```
res.getMetaData();
```

فالفكرة هنا اننا نجيب عددهم ودا بإستخدام ال getColumnCount الموجودة بال metadata الخاصة ب res

```
        int nOfCols=rsmd.getColumnCount();
```

نعمل لوب صغيرة نحصل بيها على اسم كل column ودا بإستخدام getColumnName(idx)

```
        for(int i=1; i<=nOfCols; i++){
```

```
            if(i==1){ // ID
                System.out.print(String.format("%5s",rsmd.getColumnName(i)));
                continue;
            }
            System.out.print(String.format("%15s",rsmd.getColumnName(i)));
        }
```

مش تشغل نفسك بال 1 وغيرها دى لمجرد ال formating

نرجع لل ResultSet

ال res هنا بتعبر عن كل الصفوف الناتجة وليها عدة ميثودز مفيدة مثلا first, last و next وهكذا

احنا اللى يهمنا هنا اننا نعمل لوب على كل الصفوف ونعرضها

```
        while(res.next()){
            //ID Name Email ...
            //It's better to fetch by the column name..
            int id=res.getInt("id");
            String name=String.format("%20s",res.getString("name"));
            String email=String.format("%20s",res.getString("email"));

            System.out.println(id +name +email);
        }
    } catch (SQLException ex) {
        Logger.getLogger(DBMgr.class.getName()).log(Level.SEVERE, null, ex);
    }
```

هنا كل صف فيه التالى

id, name, email

ال id عبارة عن int وال name, email عبارة عن varcharS

فبنحصل على القيمة المناسبة من كل عمود مثلا ال id بنحصل عليه ب getInt

وال name, email بنحصل عليهم ب getString

لاحظ ان كل منهم بتاخد معامل بإسم -ال label -الخاص بال عمود او بال index الخاص بيه انا شايف ان استخدام الإسم اسهل !؟

فى غيرهم كتير مثل getFloat و getBLOB و getBooleanو getDoubleو getDate و getArray و getByte و getTime و getTimestamp و.. إلخ وهكذا بنفس الفكرة

لل result set عدة ميثودز زى ماقلنا منها

next()

بتنقلنا للصف الجديد

first()

بتنقل ال cursor للصف الأول

beforeFirst()

بتنقل ال cursor للبداية

afterLast()

بتنقل ال cursor للنهاية

last()

بتنقل ال cursor للصف الأخير

وليهم ال checks مثل

isFirst(), isLast(), isBeforeFirst(), isAfterLast();

تقدر تحدد بعض خصائص ال ResultSet Object

TYPE_FORWARD_ONLY

للأمام فقط

TYPE_SCROLL_INSENSITIVE

TYPE_SCROLL_SENSITIVE

الإتنين بيسمحولك تتحرك forward, backword لكن الفرق فى حساسيتهم ناحية اى تغيير وهما opened

CONCUR_READ_ONLY

بتحدد هل هى للقراءة فقط ؟

CONCUR_UPDATABLE

ولا قابلة للتحديث؟

تحديد الخصائص دى من خلال ال Statement اللى بيتم انشاءها

[sourcecode lang='java']Statement stmt = con.createStatement(
ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);[/sourcecode]

وفى حال عدم تحديدها هيكون

TYPE_FORWARD_ONLY, CONCUR_READ_ONLY

Refs:

JDBC API Tutorial/Reference 3rd Edition