12/11/2018

Formateur : Med LAMNAOUR

ISMO TETOUAN

SQLITE sous Android

Développement mobile

Table des matières

[I. SQLite et Android 2](#_Toc529879381)

[1) Présentation 2](#_Toc529879382)

[2) SQLite sous android 2](#_Toc529879383)

[3) Création et mise à jour d’une base de données 2](#_Toc529879384)

[4) Insertion des données 4](#_Toc529879385)

[1) Afficher les données 7](#_Toc529879386)

[2) Mise à jour des données 9](#_Toc529879387)

[3) Suppression des données 9](#_Toc529879388)

[II. Exercice d’application 9](#_Toc529879389)

# SQLite et Android

## Présentation

**SQLite** est un système de gestion de base des données.

SQLite est le SGBD le plus distribué au monde, grâce à son utilisation dans de nombreux logiciels grand public comme Firefox, Skype et dans certains produits d'Apple, d'Adobe et de McAfee et dans les bibliothèques standards de nombreux langages comme PHP ou Python.

Une base de données sous Android nécessite peu de mémoire lors de l'exécution (env. 250 ko), ce qui en fait un bon candidat pour être intégré dans d'autres environnements d'exécution.

## SQLite sous android

**SQLite est intégrée dans chaque appareil Android.** L'utilisation d'une base de données SQLite sous Android ne nécessite pas de configuration ou d'administration de la base de données.

Vous devez uniquement définir les instructions SQL pour créer et mettre à jour la base de données. Ensuite, celle-ci est gérée automatiquement par la plate-forme Android.

Le chemin de sauvegarde des base de base de données sous android est le répertoire **DATA /data/ APP\_NAME/databases/FILENAME (APP\_NAME est le nom du package de l’application)**.

## Création et mise à jour d’une base de données

Les étapes de création d’une base de données SQLite sous android sont :

1. Créer une classe qui hérite de **SQLiteOpenHelper**.
2. Ajouter un constructeur a la classe que vous avez créé qui accepte comme paramètre le Contexte de l’application. Dans le code du constructeur, appeler la méthode **super() de SQLiteOpenHelper**, en précisant le nom de la base de données et sa version actuelle.
3. **SQLiteOpenHelper** est une classe abstraite qui demande en cas de l’héritage l’implémentation des méthodes suivantes :
   1. **onCreate()** : est appelée pour accéder à une base de données qui n'est pas encore créée.
   2. **onUpgrade()** : Cette méthode vous permet de mettre à jour un schéma de base de données existant ou de supprimer la base de données existante et la recréer par la méthode onCreate().

**Exemple:**

**public class MyDataBase extends SQLiteOpenHelper {  
  
 public static final String *DATABASE\_NAME* = "Student.db";  
 public static final String *TABLE\_NAME* = "Student\_table";  
  
 public static final String *COL\_1* = "ID";  
 public static final String *COL\_2* = "NOM";  
 public static final String *COL\_3* = "PRENOM";  
 public static final String *COL\_4* = "NOTE";  
  
  
 public MyDataBase(Context context) {  
 super(context, *DATABASE\_NAME*, null, 1);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase sqLiteDatabase) {  
 sqLiteDatabase.execSQL("CREATE TABLE " + *TABLE\_NAME* + " ("+*COL\_1* + " INTEGER PRIMARY KEY AUTOINCREMENT, "+ *COL\_2* + " TEXT, "+*COL\_3* + " TEXT, "+*COL\_4*+" REAL)");  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {  
 sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + *TABLE\_NAME*);  
 onCreate(sqLiteDatabase);  
 }**

**}**

Dans le fichier MainActivity.java :

**public class MainActivity extends AppCompatActivity {  
  
 MyDataBase db;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);**

**// La ligne qui créer la base de données  
 db = new MyDataBase(this);  
   
 }  
}**

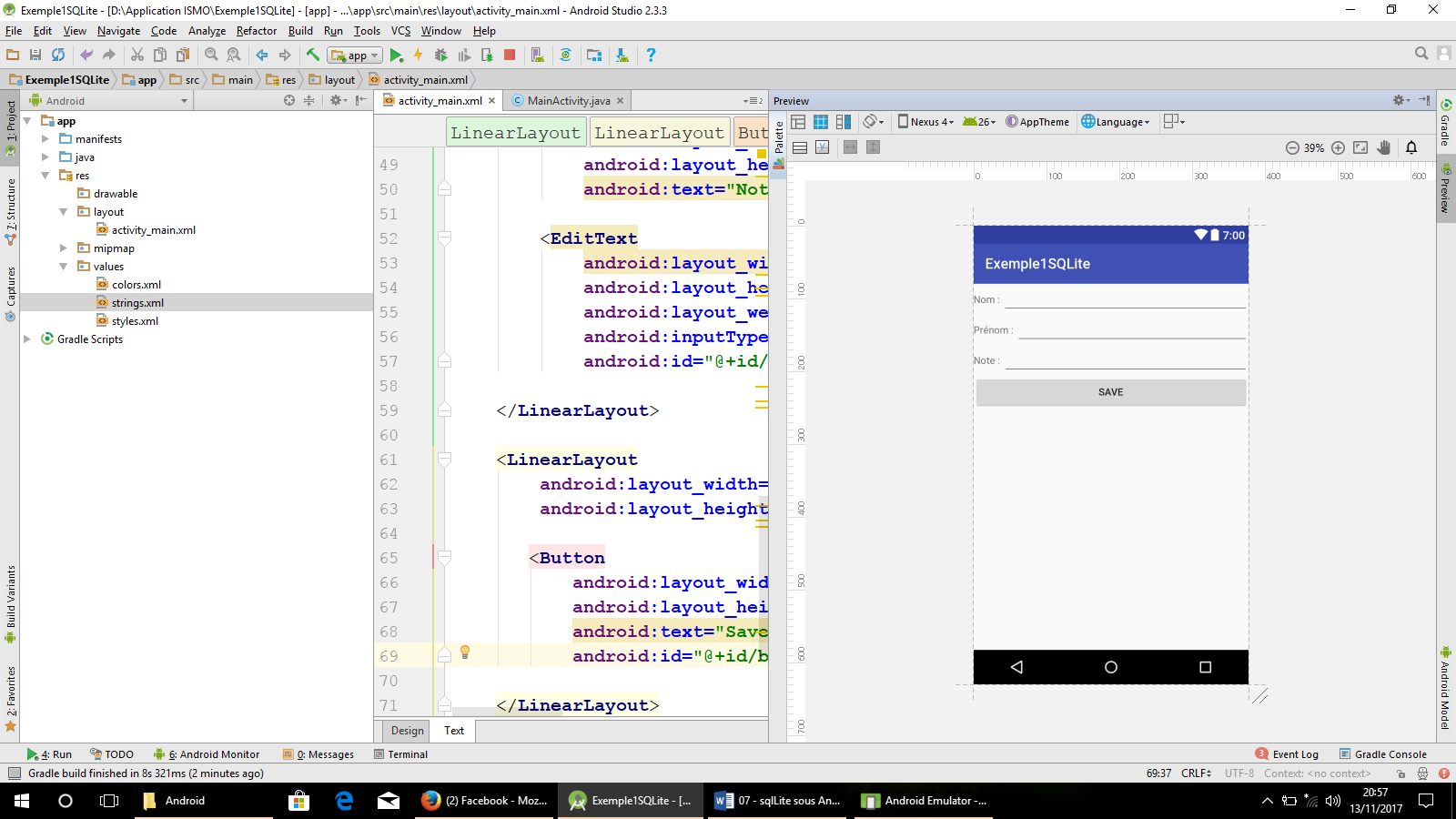
Pour chaque table, il faut créer une classe qui contient les mêmes champs que la table :

**public class Student {  
 private int id;  
 private String nom;  
 private String prenom;  
 private int note;  
  
 public Student(){  
  
 }**

**public Student(int id, String nom, String prenom, int note) {  
 this.id = id;  
 this.nom = nom;  
 this.prenom = prenom;  
 this.note = note;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getNom() {  
 return nom;  
 }  
  
 public void setNom(String nom) {  
 this.nom = nom;  
 }  
  
 public String getPrenom() {  
 return prenom;  
 }  
  
 public void setPrenom(String prenom) {  
 this.prenom = prenom;  
 }  
  
 public int getNote() {  
 return note;  
 }  
  
 public void setNote(int note) {  
 this.note = note;  
 }  
}**

## Insertion des données

Voici un exemple d’insertion dans une base de données SQLite :



Le fichier XML de cette activité est :

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context="mobile.tetouan.ofppt.ismo.exemple1sqlite.MainActivity"**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**>  
  
 <**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Nom : "**/>  
  
 <**EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:id="@+id/t\_nom"**/>  
  
 </**LinearLayout**>  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**>  
  
 <**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Prénom : "**/>  
  
 <**EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:id="@+id/t\_prenom"**/>  
  
 </**LinearLayout**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**>  
  
 <**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Note : "**/>  
  
 <**EditText  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:inputType="numberDecimal"  
 android:id="@+id/t\_note"**/>  
  
 </**LinearLayout**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**>  
  
 <**Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Save"  
 android:id="@+id/btn\_save"**/>  
  
 </**LinearLayout**>  
  
</**LinearLayout**>

Le fichier mainActivity.java est :

**public class MainActivity extends AppCompatActivity {  
  
 MyDataBase db;  
 EditText t1,t2,t3;  
 Button b1;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 db = new MyDataBase(this);  
  
 t1 = (EditText) findViewById(R.id.*t\_nom*);  
 t2 = (EditText) findViewById(R.id.*t\_prenom*);  
 t3 = (EditText) findViewById(R.id.*t\_note*);  
  
 b1 = (Button) findViewById(R.id.*btn\_save*);  
  
 b1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Student s = new Student();  
  
 s.setNom(t1.getText().toString());  
 s.setPrenom(t2.getText().toString());  
 s.setNote(Integer.*parseInt*(t3.getText().toString()));  
  
 long r=MyDataBase.*insertStudent*(db.getWritableDatabase(),s);  
  
 if(r!=-1){  
 Toast.*makeText*(getApplicationContext(),"Insertion Reussie",Toast.*LENGTH\_LONG*).show();  
 }else{  
 Toast.*makeText*(getApplicationContext(),"Insertion Echoue",Toast.*LENGTH\_LONG*).show();  
 }  
 }  
 });  
 }  
}**

Il faut ajouter la méthode **insertStudent** dans la classe MyDataBase comme suit :

**public static long insertStudent(SQLiteDatabase db, Student s){  
 ContentValues insertValues = new ContentValues();  
 insertValues.put(MyDataBase.*COL\_2*,s.getNom());  
 insertValues.put(MyDataBase.*COL\_3*,s.getPrenom());  
 insertValues.put(MyDataBase.*COL\_4*,s.getNote());  
 long result = db.insert(MyDataBase.*TABLE\_NAME*,null,insertValues);  
 db.close();  
  
 return result;  
}**

## Afficher les données

Nous allons afficher les Etudiants enregistrés dans la base de données dans une ListView.

Le fichier MainActivity.xml :

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context="mobile.tetouan.ofppt.ismo.exemple1sqlite.MainActivity"**>  
<**ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/list"**/>  
   
</**LinearLayout**>

Le fichier MainActivity.java :

**public class** MainActivity **extends** AppCompatActivity {  
  
 MyDataBase **db**;ListView **lst**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **db** = **new** MyDataBase(**this**);  
  
 ListView ls = (ListView) findViewById(R.id.***list***);  
  
 ArrayList<Student> sts = MyDataBase.*getAllStudents*(**db**.getReadableDatabase());  
  
 String[] noms = **new** String[sts.size()];  
  
 **int** i=0;  
 **for**(Student s:sts) {  
 noms[i] = s.getNom();  
 i++;  
 }  
  
 ArrayAdapter ad = **new** ArrayAdapter(**this**,android.R.layout. ***simple\_list\_item***, noms);  
 ls.setAdapter(ad);}  
}

Il faut ajouter la méthode ***getAllStudents***dans la classe *MyDataBase* comme suit :

**public static ArrayList<Student> getAllStudents(SQLiteDatabase db) {**

**ArrayList<Student> stds = new ArrayList<>();  
  
 Cursor cur = db.rawQuery("SELECT \* FROM "+MyDataBase.*TABLE\_NAME*,null);  
  
 while(cur.moveToNext()){  
 Student s = new Student();**

**s.setId(cur.getInt(0));  
 s.setNom(cur.getString(1));  
 s.setPrenom(cur.getString(2));  
 s.setNote(cur.getInt(3));  
  
 stds.add(s);  
 }  
  
 return stds;  
}**

## Mise à jour des données

Pour mettre à jour les informations d’un étudiant, on ajoute la méthode **updateStudent** suivante dans la classe ***MyDataBase*** :

**public static long** updateStudent(SQLiteDatabase db,Student s){  
 ContentValues insertValues = **new** ContentValues();  
 insertValues.put(MyDataBase.***COL\_2***,s.getNom());  
 insertValues.put(MyDataBase.***COL\_3***,s.getPrenom());  
 insertValues.put(MyDataBase.***COL\_4***,s.getNote());

**long** result = db.update(MyDataBase.***TABLE\_NAME***,insertValues,**"ID = "+s.getId(),null**);

db.close();  
  
 **return** result;  
}

## Suppression des données

Pour supprimer un étudiant on ajouter à la classe ***MyDataBase***la méthode **deleteStudent***:*

**public static long** deleteStudent(SQLiteDatabase db,**int** id){  
 **long** result = db.delete(MyDataBase.***TABLE\_NAME***,**"id = "**+id,**null**);  
 db.close();  
  
 **return** result;  
}

# Exercice d’application

1. Créer un nouveau projet
2. Ajouter à votre projet la classe **Produit** :
   1. Id produit (int – auto\_increment)
   2. Libelle (chaine de caractères)
   3. Famille (chaine de caractères)
   4. Prix achat (reel)
   5. Prix vente (reel)
3. Créer la classe **MyDBProduit** qui va permettre la création de la base de données **DBProduit.db**
4. Modifier la classe **MyDBProduit** pour permettre la création de la table **Produit** (cette table possède les mêmes champs que la classe produit).
5. Ajouter à cette classe des méthodes qui permettent de réaliser les actions suivantes :
   1. Insérer un produit : qui permet d’insérer un produit dans la base de données
   2. Modifier un produit : qui permet de modifier un produit par son ID.
   3. Supprimer un produit : qui permet de modifier un produit par son ID.
   4. Retourner la liste des produits : Pour retourner tous les produits enregistrés dans la table **PRODUIT**.
   5. Retourner les informations d'un produit : Pour retourner les informations d'un produit en fournissant comme paramètre son ID.
6. Modifier le code XML de démarrage de notre application pour afficher l'interface suivante (Créer un style pour les boutons de cette activité et un style pour le titre de l'activité) :



1. Dans le code Java de l'activité de démarrage, Insérer 5 produits dans la table produit.
2. Créer une activité pour afficher tous les produits dans une ListView, Les informations à afficher dans chaque item est ID et Libelle produit. La clique sur le bouton Liste des produits de l'activité de démarrage permet l'accès a cette activité.
3. Créer une activité qui demande à l’utilisateur de saisir les informations d’un produit et puis il va enregistrer ce dernier dans la table produit. Cette activité est accessible en cliquant sur le bouton Nouveau produit de l'activité de démarrage.
4. Créer une activité qui va permettre :
   1. Afficher dans une liste déroulante tous les ID des produits.
   2. La sélection d’un ID permet d’afficher les autres informations du produit dans des TextView.
5. Créer une activité qui va permettre :
   1. Afficher dans une liste déroulante tous les ID des produits.
   2. La sélection d’un ID permet d’afficher les autres informations du produit dans des EditText.
   3. Ajouter deux buttons a votre activité :
      1. **« Modifier »** : pour modifier les informations du contact. Un message de confirmation est affiché avant de procéder à la modification.
      2. **« Supprimer »**: pour supprimer les informations du produit. Un message de confirmation est affiche avant de procéder à la suppression.