Persistance des données sous android avec SQLite

Plan

I. Qu'est-ce que SQLite

SQLite est une bibliothèque open source écrite en C qui propose un moteur de base de données relationnelle accessible par le langage SQL. Android inclus une implémentation de SQLite.

II. SQLiteOpenHelper

SQLiteOpenHelper est une classe abstraite du package android.database.sqlite qui permet de gérer la création de la base et ses versions, généralement on utilise une sous classes qui implémente les deux méthodes abstraites onCreate et onUpgrade.

II.1. Quelques méthodes

- *SQLiteOpenHelper(Context context, String name, CursorFactory factory, int version)*
- *SQLiteDatabase getWritableDatabase ()*
- *SQLiteDatabase getReadableDatabase ()*

II.2. Exemple

```
package com.bib;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
public class SQLiteBib extends SQLiteOpenHelper{
  public SQLiteBib(Context context, String name, CursorFactory factory,
      int version) {
    super(context, name, factory, version);
  }
  @Override
  public void onCreate(SQLiteDatabase db) {
    String Sql="create table Livre (id INTEGER PRIMARY KEY AUTOINCREMENT, titre text
                NOT  NULL,nbpage INTEGER );";
    db.execSQL(Sql);
  }
  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
  }
}
```

III. ContentValues

ConentValues est une classe du package android.content qui permet de stocker un ensemble de paires (clé, valeur).

III.1. Quelques méthodes

- *ContentValues ( )*
- *void put(String key, Integer value)*
- *void put(String key, Float value)*
- *void put(String key, String value)*

III.2. Exemple

```
ContentValues v = new ContentValues();
v.put("titre", "Android2");
v.put("nbpage", 170);
```

IV. SQLiteDatabase

SQLiteDatabase est une classe du package android.database.sqlite qui permet de gérer une base de données SQLite.

IV.1. Quelques méthodes

- *void close ( )*
- *long insert (String table, String nullColumnHack, ContentValues values)*
- *int update (String table, ContentValues values, String whereClause, String[]whereArgs)*
- *int delete (String table, String whereClause, String[] whereArgs)*
- *Cursor query (String table, String[] columns, String selection, String[]selectionArgs, String groupBy, String having, String orderBy, String limit)*

- *Cursor rawQuery (String sql, String[] selectionArgs)*
- *void execSQL (String sql)*

IV.2. Exemples

IV.2.1. insert

```
String titre="Android2";
int nbPage=170;
SQLiteBib b = new SQLiteBib(this, "bib.db", null, 1);
SQLiteDatabase db = b.getWritableDatabase();
ContentValues v = new ContentValues();
v.put("titre", titre);
v.put("nbpage", nbPage);
db.insert("Livre", null, v);
db.close();
```

IV.2.2. delete

```
int id = 1 ;
SQLiteBib b = new SQLiteBib(this, "bib.db", null, 1);
SQLiteDatabase db = b.getWritableDatabase();
db.delete("Livre", "id=" + id, null);
db.close();
```

IV.2.3. update

```
String titre="Android II";
int nbPage=180;
ContentValues v = new ContentValues();
v.put("titre", titre);
v.put("nbpage", nbPage);
SQLiteDatabase db = b.getWritableDatabase();
db.update("Livre", v, "id=" + id, null);
db.close();
```

IV.2.4. query

```
SQLiteBib b = new SQLiteBib(this, "bib.db", null, 1);
SQLiteDatabase db = b.getWritableDatabase();
Cursor c = db.query("Livre", new String[] { "id", "titre", "nbpage" },
    null, null, null, null, null);
```

IV.2.5. rawQuery

```
SQLiteBib b = new SQLiteBib(this, "bib.db", null, 1);
SQLiteDatabase db = b.getWritableDatabase();
String sql="Select * From Livre";
Cursor c = db.rawQuery(sql, null);
```

V. Cursor

Cursor est une interface du package android.database.sqlite qui permet l'accès à un résultat d'une requête sur une base SQLite.

V.1. Quelques méthodes

- *abstract void close ()*
- *abstract float getFloat (int columnIndex)*
- *abstract int getInt (int columnIndex)*
- *abstract String getString (int columnIndex)*
- *abstract byte[] getBlob (int columnIndex)*
- *abstract boolean moveToNext ()*
- *abstract boolean moveToPosition (int position)*
- *abstract boolean moveToPrevious ()*

V.2. Exemple

```
while (c.moveToNext()) {
      int id = c.getInt(0);
      String titre = c.getString(1);
      int nbpage = c.getInt(2);
      Livre l = new Livre(id, titre, nbpage);
}
```

VI. Utilisation du ? dans les requêtes

VI.1. Utilité

Dans la clause Where des requêtes update, delete, query et rawQuery on peut utiliser des ? dans les emplacement des valeurs de tests des conditions. Par la suite les valeurs sont passées dans un tableau de String. L'utilisation du ? permet d'obtenir une écriture simple des requêtes pour minimiser les erreurs et simplifier leurs correction.

VI.2. Exemples

VI.2.1. ? dans delete

```
int id = 1 ;
SQLiteBib b = new SQLiteBib(this, "bib.db", null, 1);
SQLiteDatabase db = b.getWritableDatabase();
db.delete("Livre", "id=?", new String[]{id+""});
db.close();
```

VI.2.2. ? dans update

```
String titre="Android II";
int nbPage=180;
ContentValues v = new ContentValues();
v.put("titre", titre);
v.put("nbpage", nbPage);
SQLiteDatabase db = b.getWritableDatabase();
db.update("Livre", v, "id= ?", new String[]{id+""});
```

VI.2.3. ? dans query

Exécuter la requête : « SELECT * From Livre Where titre like %android% And nbpage>50 ».

```
String titre="android";
int minNbPage=50 ;
SQLiteBib b = new SQLiteBib(this, "bib.db", null, 1);
SQLiteDatabase db = b.getWritableDatabase();
Cursor c = db.query("Livre", new String[] { "id", "titre", "nbpage" },
    "titre like ? And nbpage>?",new String[]{"%"+titre+"%",minNbPage+""}, null,
null, null);
```

VI.2.4. ? dans rawQuery

Exécuter la requête : "Select * From Livre Where titre like %android% And nbpage>50".

```
String titre="android";
int minNbPage=50 ;
SQLiteBib b = new SQLiteBib(this, "bib.db", null, 1);
SQLiteDatabase db = b.getWritableDatabase();
String sql="Select * From Livre Where titre like %android% And nbpage>50";
Cursor c = db.rawQuery(sql, new String[]{"%"+titre+"%",minNbPage+""});
```

VII. Etapes à suivre

Lors de la programmation d'une application android qui utilise une base de données SQLite, suivre les étapes suivantes :

- Déclarer une classe équivalente à chaque table de la base SQLite. Le nombre d'attributs de chaque classe est égal à celui des champs de la table.
- Déclarer une sous classe de la classe SQLiteOpenHelper et redéfinir le constructeur, la méthode onCreate et la méthode onUpgrade.
- Utiliser les méthodes de la classe SQLiteDatabase pour exécuter les requêtes.
- Utiliser la classe ContentValues pour remplir les paires clé/valeur des requêtes INSERT et UPDATE.
- Utiliser la classe Cursor pour parcourir les résultats des requêtes de type SELECT.