
Persistance de données Android

BD SQLite



Contenu

Contexte.....	1
Exemple d'utilisation.....	2
Une classe à persister	2
La classe de connexion à la base SQLite	2
La classe et la création de la base de données	3
Ajouter et récupérer des objets.....	3
Mise à jour et suppression d'objets.....	4

Contexte

Afin de stocker des données dans une application Android, nous utiliserons le SGBDR SQLite intégré à Android.

Exemple d'utilisation

Une classe à persister







Soit une classe quelconque appelée ici `Objet`, dont voici la définition.

```
public class Objet {  
    private long id ;  
    private String nomChamp1 ;  
    private String nomChamp2 ;  
  
    public Objet(String nomChamp1, String nomChamp2) {...}  
    public void setId(long id) { this.id = id; }  
    public long getId() { return id; }  
    public String getNomChamp1() { return nomChamp1; }  
    public String getNomChamp2() { return nomChamp2; }  
}
```

La classe de connexion à la base SQLite

Comme bien souvent lorsque l'on utilise un objet pour connecter une base de données, nous utiliserons de préférence un Singleton afin d'assurer l'unicité de l'instance. L'instance de la classe de connexion sera donc attribut statique de la classe en question.

Quelques points d'explication :

-  La classe de connexion et manipulation de la bd doit étendre `SQLiteOpenHelper`.
-  La surcharge de la méthode `onCreate` permet de créer les tables de la base de données si elle n'existe pas déjà.
-  La base de données SQLite permet de gérer le versionning et ainsi l'attribution d'une nouvelle version appellera la surcharge de la méthode `onUpgrade` permettant éventuellement d'appliquer une modification de la structure des tables tout en réutilisant les données déjà présentes.
-  Les mises à jour et ajout d'occurrences dans la base de données se fait à partir d'une structure `ContentValues` fonctionnant sur le principe de clé/valeur. La clé correspond alors au nom du champ et la valeur à son contenu.
-  La sélection de données, comme dans beaucoup de langage de programmation, nous retourne un curseur à parcourir pour en récupérer toutes les données.
-  Le filtrage des données à la sélection, suppression ou mise à jour se fait grâce à une substitution du « ? » dans le String de sélection par les éléments d'un tableau de String.

La classe et la création de la base de données

```
public class BaseExemple extends SQLiteOpenHelper {
    private static BaseExemple base = null;

    private BaseExemple(Context context) {
        super(context, "nomDeLabase", null, 1);
    }

    public static BaseExemple getBase(Context context) {
        if (base == null) {
            base = new BaseExemple(context);
        }
        return base;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table nomDeTable ( " +
            " id integer primary key autoincrement " +
            ", nomChamp1 text " +
            ", nomChamp2 text ) ");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("drop table nomDeTable");
        onCreate(db);
    }
}
```

Ajouter et récupérer des objets

```
public long ajouterObjet(Objet o) {
    ContentValues cv = new ContentValues();
    cv.put("nomChamp1", o.getNomChamp1());
    cv.put("nomChamp2", o.getNomChamp2());
    long res = this.getWritableDatabase().insert("nomDeTable", null, cv);
    o.setId(res);
    this.getWritableDatabase().close();
    return res;
}

public ArrayList<Objet> getLesObjets() {
    ArrayList<Objet> lesObjets = new ArrayList<>();
    Cursor c = this.getReadableDatabase().query("nomDeTable", null, null
        , null, null, null, null);
    while (c.moveToNext()) {
        Objet o = new Objet(
            c.getString(c.getColumnIndex("nomChamp1")),
            c.getString(c.getColumnIndex("nomChamp2")));
        o.setId(c.getLong(c.getColumnIndex("id")));
        lesObjets.add(o);
    }
    this.getReadableDatabase().close();
    return lesObjets;
}
```

Mise à jour et suppression d'objets

```
public void majObjet(Objet o) {
    ContentValues cv = new ContentValues();
    cv.put("nomChamp1", o.getNomChamp1());
    cv.put("nomChamp2", o.getNomChamp2());
    this.getWritableDatabase().update("nomDeTable", cv, "id=?",
        new String[]{String.valueOf(o.getId())});
    this.getWritableDatabase().close();
}

public void supprimerObjet(Objet aSupp) {
    this.getWritableDatabase().delete("nomDeTable", "id=?",
        new String[]{String.valueOf(aSupp.getId())});
    this.getWritableDatabase().close();
}
```