TP Android *(sugar-orm)*

Souleymane ATTO DEBA (FA)
16/11/2017

Résumé

Aujourd'hui les base de données sont devenus indispensable pour stocker les données dans les applications Android. Il existe plusieurs technologies permettant de stocker et d'accéder les données des applications. La librairie SQLite est fournie par défaut dans les applications android, mais l'utilisation de cette dernière est souvent compliquée. Il faut avoir des connaissance en SQL et des notions en base de données.
Il existe des mécanismes de persistance de données tels que Data Access Objet (dao), Object Relationnal Mapping(ORM).
Dans ce TP, nous allons voir comment configurer et utiliser la librairie sugar-ORM

Pré-requis

- *Savoir programmer une application Android avec **plusieurs activités**.*
- *Savoir utiliser les layouts*
- *La notion Base données (facultatif, mais utile pour mieux comprendre le tp)*

Code Source

*Code source **initial** : https://github.com/AttoDeba1/Developpement_Mobile/tree/master/TP_dao_final*
Code source **final** : https://github.com/AttoDeba1/Developpement_Mobile/tree/master/TP_dao_final

Explications du TP

Sugar ORM permet de persister des données faire des opérations spécifiques sur les données sans exposer les détails de la base de données.
Dans ce TP nous allons réaliser une application Android pour une mini-Bibliothèque en utilisant la librairie sugar ORM pour persistance de nos données.

Fonctionnalités de l'application

1- Enregistrer Auteur dans la base de donnée
2- Afficher la liste des auteurs
3- Modifier & Supprimer un auteur
4- Enregistrer un livre avec son auteur
5- Afficher la liste des livre
6-Supprimer un livre

Etape 1: Installation et configuration de SugarORM

Installation:

Pour installer la bibliothèque SugarORM au sein d'un projet android , il suffit juste d'ajouter la bibliothèque dans le script build.gradle(Module:app) en utilisant la commande suivante:

implementation 'com.github.satyan:sugar:1.5'

Si vous utilisez la version 2.3 d'android studio , il faut remplacer «implementation» par «compile».

Configuration

Une fois la bibliothèque bien installée dans le fichier gradle, nous allons configurer sugar ORM dans le fichier androidManifest.xml en ajoutant de meta-données comme ceci:

```
        <meta-data
            android:name="DATABASE"
            android:value="db_books.db" />
        <meta-data
            android:name="VERSION"
            android:value="3" />
        <meta-data
            android:name="QUERY_LOG"
            android:value="true" />
        <meta-data
            android:name="DOMAIN_PACKAGE_NAME"
            android:value="com.attodeba.ads" />
```

Il faut décommenter le code de la ligne 12 à la ligne 18 .

NB:

DATABASE : nom de la base de donnée(le fichier sqlite qui sera généré)

VERSION: la version du schéma relationnel de la base de données.

QUERY_LOG: paramètre facultatif d'afficher les logs dans la console.

DOMAIN_PACKAGE_NAME: il permet de spécifier le package où tous les classes représentant nos entités sont présents.

Etape 2: Connexion à la base de donnée

La connexion à la base de donnée est gérée automatiquement par la bibliothèque SugarORM, mais il faut d'initier le context de l'application. permettant à sugar de créer la base de donnée lors de la création de l'application. Pour cela il faut décommenter la ligne 13 et 19 de la classe AppConfig qui se trouve dans le package appconfig

```
1   package com.attodeba.ads.tp_dao.appconfig;
2
3   import ...
9   public class AppConfig extends Application {
10      @Override
11      public void onCreate(){
12          super.onCreate();
13          SugarContext.init(getApplicationContext());
14      }
15
16      @Override
17      public void onTerminate() {
18          super.onTerminate();
19         SugarContext.terminate();
20      }
21  }
```

Il faut faire Le reBuild du projet.

Etape 3: création des classes d'entités (ou modèle des données)

Pour ce TP nous allons utiliser deux entités Author et Book.
Pour définir ces entités, il suffit de créer des classes java qui héritent de la classe ***SugarRecord****.*
La classe sugarRecord permet de faire le mapping entre l'entité et la table relationnel correspondante dans la base de données.
Elle permet de faire les opérations CRUD sur la base de donnée

classe Author

Allez dans le package models/:
La classe author hérite de la classe SugarRecord
par défaut sugar va nous créer la table author et la table book dans la base de données.
Mais on peut toutefois modifier le nom de la table , ainsi que les différentes propriétés de la table en utilisant des annotations.
Pour renommer la table author et ses propriétés il faut décommenter la ligne 10 ,12, et 13.
Attention -> il faut écrire le constructeur par défaut pour chaque classe.

```
@Table(name="Author")
public class Author extends SugarRecord {
    @Column(name="name")
    private String name;
    @Column(name="firstName")
    private String firstName;

    public Author() {
```

classe Book

Pour la classe Book on n'utilisera pas les annotations. il faut juste hériter la classe Book de la classe SugarRecord.

Etape 4: Utilisation des méthodes CRUD sur l'entité Author et Book

Chaque méthode CRUD correspond à une commande SQLite : Create à INSERT, Read à SELECT, update à UPDATE, delete à DELETE.
Grâce à la bibliothèque SugarORM l'ensemble de ces commandes est simplifié par des fonction de la classe SugarRecord . (save(), listAll(), update(), delete() , findbyId()..)

Etape 4-1: Ajout d'une entité Author dans la base de donnée

Pour ajouter une entité dans la base de donnée, il faut appliquer la méthode save() sur l'entité.
exemple d'enregistrement d'une entité author dans la base de donnée:

```
Author author = new Author(name,firstname);
author.save();
```

Il faut aller dans l'activité AuthorActivity à la ligne 136 et complète le code de la méthode **saveAuthor(String name, String firstname)**.

Etape 4-2: Affichage de la liste des entités Author depuis la base de donnée.

Pour récupérer la liste de tous les entités Author enregistrés dans la base de données, il suffit juste d'appliquer la méthode listAll() sur SugarRecord avec comme paramètre la classe **Author.class.**
syntaxe:

```
List<Author> list = SugarRecord.listAll(Author.class);
ou                = Author.listAll(Author.class);
```

Pour afficher la liste de tous les auteurs enregistré dans la base de donnée. Il faut décommenter les lignes 37, 38 et 39 de l'activité AuthorList.

Etape 4-3: Modification & Suppression d'une entités Author

- Update Author

Pour mettre à jour une entité author, il faut d'abord récupérer l'enregistrement de l'entité dans la base de donnée par la méthode ***findById(Author.class, id)***. Une fois l'enregistrement est récupéré, on applique les modifications sur l'entité author et le sauvegarder

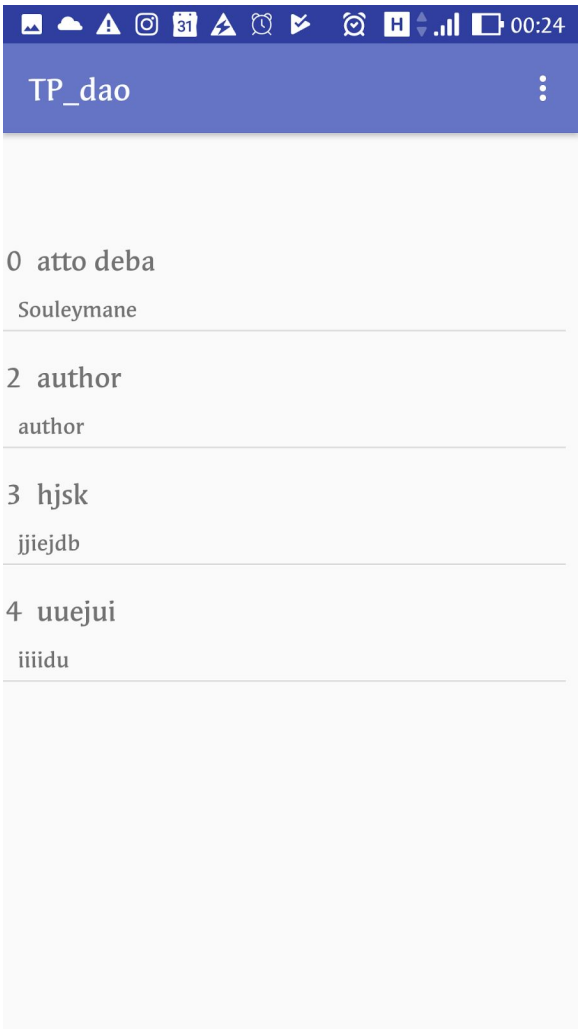
Allez dans le fichier AuthorActivity.java à la ligne 130 compléter le code de la méthode **updateAuthor**(**long** authorId, String firstname, String name);

```
125
126     /****methode permettant de mettre à jour un auteur *****/
127     public void updateAuthor(long authorId, String firstname, String name){
128         author= SugarRecord.findById(Author.class,authorId);
129         author.setFirstName(firstname);
130         author.setName(name);
131         SugarRecord.save(author);
132         //author.save();
133         saveButton.setText(SAVE_ACTION);
134     }
135     /****methode permettant d'ajouter un nouveau auteur *****/
136     public void saveAuthor(String name, String firstname){
137         author = new Author(name,firstname);
138         SugarRecord.save(author);
139     }
```

Etape 5: Ajout d'une entités Book.

Etape 5-1: Affichage la liste de tous les livres

Etape 5-2: Suppression d'un livre


00:24
TP_dao
0 atto deba
Souleymane
2 author
author
3 hjsk
jjiejdb
4 uuejui
iiiidu

Informations complémentaires

SugarRecord Java doc
http://www.atetric.com/atetric/javadoc/com.github.satyan/sugar/1.5/com/orm/SugarRecord.html

SugarORM : http://satyan.github.io/sugar/
Wikipédia ORM Définition : https://fr.wikipedia.org/wiki/Mapping_objet-relationnel