Android – TP3 : BD avec la librairie Room
Projet - Gestion de comptes

- **Ce TP permet de continuer ou débuter le mini-projet**
- Vous trouverez sur le web toute la documentation nécessaire pour ce module
  - Documentation Java : http://docs.oracle.com/javase/8/docs/api
  - Guide User Interface Android : http://developer.android.com/guide/topics/ui/index.html
  - Android référence : http://developer.android.com/reference/android/package-summary.html

Comprendre la librairie Room en analysant une application TodoList

Récupérez l'archive TodoList.zip sur Chamilo. Cette archive contient un projet Android proposant une application pour réaliser une liste de tâche. Cette application vous servira d'exemple pour réaliser la base de données BD de votre projet.

Lancez et analysez l'application de gestion de tâches TodoList.

Fichiers importants du projet TodoList :

`fr.iut2.todolist.db/DatabaseClient.java` : cette classe permet de réaliser la première création de BD sur le device et, par la suite, de faire le lien entre l'application et cette BD à l'aide de l'objet de type `AppDatabase`.

`fr.iut2.todolist.db/AppDatabase.java` : cette classe décrit les entités (tables) disponibles, le numéro de version de la BD (ces informations sont contenues dans l'annotation `@Database`) et précise les DAO (data access object – interface précisant les méthodes insert, delete, etc. pour une table).

`fr.iut2.todolist.db/Task.java` : cette classe représente l'entité tâche. Les annotations (`@Entity`, `@PrimaryKey`, `@ColumnInfo`, etc.) décrivent les informations utiles à la librairie Room pour faire le lien avec la BD.

`fr.iut2.todolist.db/TaskDao.java` : cette interface précise les méthodes méthodes insert, delete, etc. pour une table. Les annotations (`@Dao`, `@Query`, `@Insert`, `@Delete`, etc.) décrivent les informations utiles à la librairie Room pour faire le lien avec la BD.

`fr.iut2.todolist/MainActicity.java` : cette activité va récupérer la liste des tâches et l'afficher à l'aide de l'objet de Type `ListView` et son adapter (pour plus d'informations, reportez-vous au document M4104C-2.TP2.exercice5). Pour récupérer la liste de tâches, une classe asynchrone `GetTasks` est créé (qui hérite d'`AsyncTask`). Cette classe décrit la demande à la BD dans la méthode `doInBackground(…)` et décrit le traitement lorsqu'on obtient la réponse de la BD dans la méthode `onPostExecute(…)`.

`fr.iut2.todolist.db/AddTaskActivity.java` : cette activité va ajouter une tâche à la BD. Pour ce faire, une classe asynchrone `SaveTask` est créé (qui hérite d'`AsyncTask`). Cette classe décrit la demande à la BD dans la méthode `doInBackground(…)` et décrit le traitement lorsqu'on obtient la réponse de la BD dans la méthode `onPostExecute(…)`.

`Gradle Scripts : build.graddle (Module : app) et libs.version.toml` contiennent les dépendances pour la librairie Room.

Les URLs utiles pour utiliser la librairie Room :

Guide Room : https://developer.android.com/training/data-storage/room/

Guide Entity : https://developer.android.com/training/data-storage/room/defining-data

Guide DAO : https://developer.android.com/training/data-storage/room/accessing-data

COMMENT lire une base de données SQLITE sur un device avec Android Studio ?

Ouvrir la vue View > Tool Windows > « App Inspection »

Vous pourrez vérifier les tables et faire des requêtes SQL dans l'onglet « Database Inspector »

Installation de la librairie Room sur votre projet

Pour installer la librairie Room dans votre projet, ajoutez les dépendances suivantes :

- dans le fichier `build.grappdle (Module : app)` – Gradle Scripts

```
dependencies {
    …
    annotationProcessor libs.room.compiler
    implementation libs.room.runtime
```

- dans le fichier `libs.versions.toml` – Gradle Scripts

```
[versions]
…
room = "2.6.1"

[libraries]
…
room-compiler = { group = "androidx.room", name = "room-compiler", version.ref = "room" }
room-runtime = { group = "androidx.room", name = "room-runtime", version.ref = "room" }
```

Créer `DatabaseClient` et `AppDatabase`

En prenant exemple sur le projet TodoList, créez les classes `DatabaseClient` et `AppDatabse` dans votre projet. Créez un package `db` et placez les classes dedans. Dans la classe `DatabaseClient`, modifiez le nom de la BD. Remplacez « MyToDos » par « EcoleDesLoustics » par exemple. Dans la classe `AppDatabase`, modifiez l'annotation `entities` et la méthode retournant le DAO pour prendre en compte l'entité `User` et son DAO que nous allons créer dans la suite du document.

Créer une entité `User` et son DAO

En prenant exemple sur le projet TodoList, créer une entité `User` et son DAO associé `UserDAO`. Pour votre projet, un utilisateur n'aura qu'un prénom et un nom (élève de primaire).

IMPORTANT : quand on développe avec une BD sous Android, si vous faites des modifications sur l'entité (ajouter, modifier ou supprimer un champ), vous devez absolument supprimer la BD présente sur le device (pour prendre en compte les dernières modifications). Pour ce faire, soit vous supprimer l'application soit vous effacer les données de l'application sur le device. Attention on ne parle pas ici de migration vers une nouvelle version de l'application (c'est un autre mécanisme), on parle simplement de la phase de développement.

Si vous rencontrez un problème « **Room cannot verify the data integrity.** » lors de l'installation de votre application, dans le fichier AndroidManifest modifiez la propriété suivante : `android:allowBackup="false"`

Ajouter un utilisateur

En prenant exemple sur le projet TodoList et la classe `AddTaskActivity`, créer une activité qui va ajouter un utilisateur. Dans cette activité, créez une classe asynchrone `SaveUser` (qui hérite d'`AsyncTask`). Cette classe décrit la demande à la BD dans la méthode `doInBackground(…)` et décrit le traitement lorsque qu'on obtient la réponse de la BD dans la méthode `onPostExecute(…)`. Pensez bien à créer un objet de Type `SaveUser` et à l'exécuter pour faire l'ajout.

Vous pourrez voir si l'ajout s'est fait correctement soit en regardant directement dans la BD de votre device (reportez vous à l'encadré ci avant **COMMENT lire une base de données SQLITE sur un device avec Android Studio 3 ?**) soit en réalisant l'étape suivante « lister les utilisateurs ».

Lister les utilisateurs

En prenant exemple sur le projet TodoList et la classe `MainActivity`, créer une activité qui va lister les utilisateurs. Dans cette activité, créez une classe asynchrone `GetUsers` (qui hérite d'`AsyncTask`). Cette classe décrit la demande à la BD dans la méthode `doInBackground(…)` et décrit le traitement lorsque qu'on obtient la réponse de la BD dans la méthode `onPostExecute(…)`. Pensez bien à créer un objet de Type `GetUsers` et à l'exécuter pour faire la demande d'information.

Pré-remplir la BD à la création

Dans la suite, nous présentons un mécanisme pour pré-remplir la BD, nous vous conseillons pour votre projet de créer des utilisateurs fictifs pour vos tests.

Si vous souhaitez pré-remplir la BD à la création, utilisez la méthode `addCallback(…)` du `databaseBuilder` dans la classe `DatabaseClient`. Par exemple, dans le projet TodoList, commenter l'affection à `appDatabase` et décommenter l'instruction ci-dessous (**ATTENTION** comme précisé dans l'étape « créer une entité `User` et son DAO », vous devez soit supprimer l'application soit effacer les données de l'application sur le device pour prendre en compte l'insertion de tâches à la première création de la BD).

```
// Ajout de la méthode addCallback permettant de populate (remplir) la base de données à sa création
appDatabase = Room.databaseBuilder(context, AppDatabase.class, "MyToDos").addCallback(roomDatabaseCallback).build();
```

L'objet `roomDatabaseCallback` est un objet qui permet à la création de la BD d'agir directement dessus. Par exemple, dans le projet TodoList, nous insérons 2 tâches sous forme SQL :

```
// Objet permettant de populate (remplir) la base de données à sa création
RoomDatabase.Callback roomDatabaseCallback = new RoomDatabase.Callback() {

    // Called when the database is created for the first time.
    @Override
    public void onCreate(@NonNull SupportSQLiteDatabase db) {
        super.onCreate(db);

        db.execSQL("INSERT INTO task (task, description) VALUES(\"tâche 1\", \"Installer la librairie Room\");");
        db.execSQL("INSERT INTO task (task, description) VALUES(\"tâche 2\", \"Créer DatabaseClient\");");
}};
```

Comment faire pour conserver l'utilisateur courant ?

Dans la suite, nous présentons un mécanisme permettant de conserver une information en contexte d'une application (en gros, avoir une ou plusieurs variables globales à l'application). Comme indiqué dans le titre de cette section, nous souhaitons par exemple conserver l'utilisateur courant en cours d'exécution pour, par exemple, enregistrer les meilleurs résultats sur les différents exercices (fonctionnalités supplémentaires possibles).

A venir, *Shared Preferences*

https://developer.android.com/training/data-storage/shared-preferences?hl=fr

Pour aller plus loin

Des exemples, codelabs, blogs :

https://developer.android.com/topic/libraries/architecture/room

Présentation de Room sur la chaîne Android Developers :

https://www.youtube.com/watch?v=SKWh4ckvFPM

Le projet TodoList fourni se base sur l'article de Belal Khan :

https://www.simplifiedcoding.net/android-room-database-example/

Versions et dépendances :

https://developer.android.com/jetpack/androidx/releases/room