20. Lesson 23/05/23

Databases for Mobile Apps

- **Local** and **Remote** Databases
- **SQLite** and **Room**
- Using an Android DB in practice

Databases for Android

In an Android app it is possible to use **two different kind od DBs.**

1. **Local:** if you want your data to be device specific (stay local, then you should go with a local DB).
   A local DB might be a preferable choice over remote DB when you have to simply save data locally and avoid the server requests.
2. **Remote:** if the app needs to synchronize data across all its users or involve live data being fed to a user's request then you need to use a remote database. (e.g. weather app)

Databases for Android (Remote)

- An Android application should not directly access to a database deployed on a remote server.
- Best practices require to implement a web service between the database and the Android application.
- Having a **web service layer:**
  - **Reduces the complexity** of the Android application.
  - **Reduces the dependency** on database specific operations.
- The problem of accessing a client/server database like MySQL from an Android application can be defined as the problem of consuming a web service hosted somewhere.
- Don't need to care about what is behind the web service:
  - It may be a MySQL database, MongoDB database, a Social Media network.
- What you need is the API endpoints exposed the web service.

Databases for Android (Local)

If all the informations should be stored on a mobile device, there are only some different options including SQLite and Realm.

- **SQLite** is a opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features.
- **Realm** is a NoSQL mobile database that runs directly inside phones, tablets or wearables.

→ Use **Room** instead of SQLite if we want a local database.

SQLite

The inbuilt SQLite core library is within the Android OS.

- It will handle CRUD (Create, Read, Update and Delete) operations required for a database.
- Java classes and interfaces for SQLite are provided by the android.database.
- SQLite maintains an effective database management system.

This conventional method has its own disadvantages:

- You have to write long repetitive code, which will be time-consuming as well as prone to mistakes.
- It is very difficult to manage SQL queries for a complex relational database.

```
private static final String SQL_CREATE_ENTRIES =
    "CREATE TABLE " + FeedEntry.TABLE_NAME + " (" +
    FeedEntry._ID + " INTEGER PRIMARY KEY," +
    FeedEntry.COLUMN_NAME_TITLE + " TEXT," +
    FeedEntry.COLUMN_NAME_SUBTITLE + " TEXT)";

private static final String SQL_DELETE_ENTRIES =
    "DROP TABLE IF EXISTS " + FeedEntry.TABLE_NAME;
```

Save data using SQLite

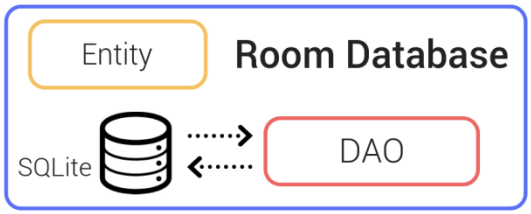
https://developer.android.com/training/data-storage/sqlite

Room Persistence Library

To overcome this, Google has introduced **Room Persistence Library.**

This acts as an **abstraction layer for the existing SQLite APIs.**

All the required packages, parameters, methods, and variables are imported into an Android project by using simple annotations.



What is the Room Persistence Library?

Room is an SQLite object mapping library. It was built to assist developers implementing local SQLite database transactions.

It helps to avoid the boilerplate code that was previously associated with interacting with the SQLite database.

Essentially the room persistence library allows developers to easily convert SQLite table data into java objects. Room provides compile time checks of SQLite statements.

Save data in a local database using Room:
https://developer.android.com/training/data-storage/room
SQLite and the Room Persistence Library:
https://codingwithmitch.com/blog/sqlite-and-the-room-persistence-library/
What to choose Realm or SQLite with Room?:
https://itnext.io/what-to-choose-realm-or-sqlite-with-room-e55c34b1675c