Android移动开发课程

实验指导书-Database部分

【实验目的】

初步了解Android 的 Notification 组件

【实验设计】

本次实验包括两个验证实验和一个自选实验。

其中验证实验为个人实验，已经提供源代码、操作步骤、实验指导视频

由于NetWork涉及网络通讯，因此本实验提供了一个Jsp编写的服务器，源代码已经给出，可以在Tomcat下运行。

要求：

按照实验步骤完成，

以个人为单位提交，提交实验报告一份，实验报告需要回答指导书中问题。

自选实验为小组实验，建议2-4人组队，提供了参考选题和参考资料。

要求：

以小组为单位提交，提交实验报告一份、源代码一份、可以执行的APK文件一个。

【实验内容】

应用不可避免的要和数据打交道，当数据规模越来越大的时候，就需要对于数据进行管理，比如采用数据库。

Android自带了一个sqlite，基于sqlite，Google提供了Room这个ORM。当然，Litepal和GreenDao数据库开源框架。

本实验包括sqlite的使用和Room的使用，并基于数据库，要求设计一个利用数据库完成的自选实验。

验证实验一：Sqlite的使用

在example下提供了本实验源码，分别是Exp1REceiver和Exp1Activity两个文件。

对于SQLite的操作应该放在子线程中，但为了方便起见，本实验放在主线程中执行。

1. 在Android开发中不可避免的和数据打交道，我们可以使用文件或者数据库的方式

保存数据

Sqlite是一款轻量型的关系数据库，相比MySQL等其他数据库，它占用的资源非常低、执行速度快、不是程序通信的独立进程而是连接到程序中成为程序一个主要部分、存储在单一磁盘文件中的一个完整数据库、可以自由复制到其他机器上、数据库文件可以在不同字节顺序的机器间自由的共享(跨平台/可移植性)、没有额外的依赖、不需要安装、源码完全开源、弱类型、Android SDK自带，简单好用

2.SQLiteOpenHelper

如果这个数据库存在，那么SQLiteOpenHelper负责管理数据库。 如果数据库不存在，那么SQLiteOpenHelper会建立一个数据库。 如果有需要的话，SQLiteOpenHelper还负责升级数据库。

SQLiteOpenHelper 类包含一组用于管理数据库的实用 API。当使用此类获取对数据库的引用时，系统仅在需要时才执行可能需要长时间运行的数据库创建和更新操作，而不是在应用启动期间执行。

该类是一个抽象类，要使用SQLiteOpenHelper类时必须创建该类的派生类。 需要实现抽象方法 OnCreate(), OnUpgrade()

SQLiteOpenHelper拥有两个构造方法可供重写，两个构造方法前四个参数是一致的：第一个参数为Context，第二个参数为数据库名，第三个参数允许我们返回自定义的Cursor，第四个参数为数据库版本号。

参考：*https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper*

3.获得一个数据库对象

获得一个数据库：getWritableDatabase()、getReadableDatabase()，两者均会返回一个可读写的数据库对象，区别在于当数据库不可写入的时候，getReadableDatabase()返回对象以只读方式打开，getWritableDatabase()抛出异常。

4. 了解ContentValues

可以填充key-value

5.插入操作

了解SQL插入语法：

INSERT INTO table_name VALUES (值1, 值2,....)

INSERT INTO table_name (column1,column2...) VALUES (value1,value2...)

了解SQLiteDatabase.insert方法

public long insert (String table_name, String nullColumnHack, ContentValues values)

由于底层数据库不允许插入空行，当values为空的时候，将nullColumnHack置为null

6. 更新操作

了解SQL更新语法：

UPDATE table_name SET column1=value1,column2=value2,...WHERE whereClause

了解SQLiteDatabase.update方法

int update (String table_name, ContentValues values, String whereClause, String[] whereArgs)

7. 删除操作

了解SQL更新语法：

DELETE FROM table_name WHERE whereClause;

了解SQLiteDatabase.delete方法

int update (String table_name, ContentValues values, String whereClause, String[] whereArgs)

8.查询操作

了解SQL查询语法

SELECT (column1,column2...)

FROM table_name

[WHERE selection]

[GROUP BY groupBy [HAVING having]]

[ORDER BY orderBy]

了解SQLiteDatabase.query方法

Cursor query(String table_name, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)

9. 了解Adapter

Adapter是用来帮助填出数据的中间桥梁。适配器将各种数据以合适的形式显示在View中，将Model转化成View。

10. 使用SQL语句操作SQLite

了解SQLiteDatabase.execSQL方法

了解SQLiteDatabase.rawQuery方法

验证实验二：Room的使用

Room持久性库提供了一个基于SQLite的抽象层，以便在利用SQLite的全部功能的同时，实现更强大的数据库访问能力。

了解Room的依赖

```
    dependencies {
  def room_version = "2.2.5"
  implementation "androidx.room:room-runtime:$room_version"
  annotationProcessor "androidx.room:room-compiler:$room_version"
  implementation "androidx.room:room-ktx:$room_version"
  implementation "androidx.room:room-rxjava2:$room_version"
  implementation "androidx.room:room-guava:$room_version"
  testImplementation "androidx.room:room-testing:$room_version"
}
```

Room 包含 3 个主要组件：

数据库：包含数据库持有者，并作为应用已保留的持久关系型数据的底层连接的主要接入点。

使用 @Database 注释的类应满足以下条件：

1.是扩展 RoomDatabase 的抽象类。

2.在注释中添加与数据库关联的实体列表。

3.包含具有 0 个参数且返回使用 @Dao 注释的类的抽象方法。

在运行时，您可以通过调用 Room.databaseBuilder() 或 Room.inMemoryDatabaseBuilder() 获取 Database 的实例。

Entity：表示数据库中的表。

DAO：包含用于访问数据库的方法。

借助java注释，完成数据库设计。

参考：https://www.jianshu.com/p/4cb6ccf09350

自主实验，从以下选题中挑选一个实验完成，或者自拟题目，二到四人组队完成。

选题一：

利用SQLite作为数据库，现有一个叫做"今天拍到XXX的猫了嘛"的app。

含有一张"cat"数据表，其表中含有int类型字段"id"，char类型字段"cat_name",char类型"cat_address"。

现在需要升级为version2.0,新添加一张表"see_cat"，含有int类型自增字段"id",char类型字段"cat_name"，date类型"time"。

允许用户向"see_cat"添加（insert）一条数据。当用户点击"btn_ins_see_cat"按钮时，触发该事件，其所需要内容来自"etv_ins_see_name"的EditText控件和"etv_ins_see_time"的EditText控件。

删去（delete）一条数据，当用户点击"btn_del_see_cat"按钮时，触发该事件，其所需要内容来自"etv_del_see_time"的EditText控件。

向"cat"更新(update)一条数据. 当用户点击"btn_upd_cat"按钮时，触发该事件，其所需要内容来自"etv_upd_cat_name"的EditText控件和"etv_upd_cat_address"的EditText控件。

获得(query)"see_cat"的数据，当用户点击"btn_que_see_cat"按钮时，触发该事件，其所需要内容来自"etv_que_see_cat"的EditText控件。

您不能使得用户app中的原来的"cat"表发生变动，而且要求app不会崩溃。 您只需要编写code部分，请使用java作为code编写的语言而非kotlin语言。