Android 数据存储的五种方式

1. sharedPreferences

适用范围：保存少量的数据，且这些数据的格式非常简单：字符串型、基本类型的值。比如应用程序的各种配置信息（如是否打开音效、是否使用震动效果、小游戏的玩家积分等），解锁口 令密码等。

核心原理：保存基于XML文件存储的key-value键值对数据，通常用来存储一些简单的配置信息。通过DDMS的File Explorer面板，展开文件浏览树,很明显SharedPreferences数据总是存储在/data/data//shared_prefs目录下。SharedPreferences对象本身只能获取数据而不支持存储和修改，存储修改是通过SharedPreferences.edit()获取的内部接口Editor对象实现。 SharedPreferences本身是一 个接口，程序无法直接创建SharedPreferences实例，只能通过Context提供的getSharedPreferences(String name, int mode)方法来获取SharedPreferences实例，该方法中name表示要操作的xml文件名，第二个参数具体如下：

- Context.MODE_PRIVATE: 指定该SharedPreferences数据只能被本应用程序读、写。
- Context.MODE_WORLD_READABLE: 指定该SharedPreferences数据能被其他应用程序读，但不能写。
- Context.MODE_WORLD_WRITEABLE: 指定该SharedPreferences数据能被其他应用程序读，写。
  Editor有如下主要重要方法：
- SharedPreferences.Editor clear():清空SharedPreferences里所有数据
- SharedPreferences.Editor putXxx(String key , xxx value): 向SharedPreferences存入指定key对应的数据，其中xxx 可以是boolean,float,int等各种基本类型据
- SharedPreferences.Editor remove(): 删除SharedPreferences中指定key对应的数据项
- boolean commit(): 当Editor编辑完成后，使用该方法提交修改。

读写其他应用的SharedPreferences: 步骤如下

```
1、在创建SharedPreferences时，指定MODE_WORLD_READABLE模式，表明该SharedPreferences数据可以被其他程序读取

2、创建其他应用程序对应的Context:
Context pvCount =
createPackageContext("com.tony.app",Context.CONTEXT_IGNORE_SECURITY);这里的com.tony.app就是其他程序的包名
3、使用其他程序的Context获取对应的SharedPreferences
SharedPreferences read =
pvCount.getSharedPreferences("lock",Context.MODE_WORLD_READABLE);
4、如果是写入数据，使用Editor接口即可，所有其他操作均和前面一致。
```

SharedPreferences对象与SQLite数据库相比，免去了创建数据库，创建表，写SQL语句等诸多操作，相对而言更加方便，简洁。但是SharedPreferences也有其自身缺陷，比如其职能存储boolean，int，float，long和String五种简单的数据类型，比如其无法进行条件查询等。所以不论SharedPreferences的数据存储操作是如何简单，它也只能是存储方式的一种补充，而无法完全替代如SQLite数据库这样的其他数据存储方式。

2. 文件存储数据

核心原理: Context提供了两个方法来打开数据文件里的文件IO流 FileInputStream openFileInput(String name); FileOutputStream(String name , int mode),这两个方法第一个参数 用于指定文件名，第二个参数指定打开文件的模式。具体有以下值可选：
　MODE_PRIVATE：为默认操作模式，代表该文件是私有数据，只能被应用本身访问，在该模式下，写入的内容会覆盖原文件的内容，如果想把新写入的内容追加到原文件中。可　以使用Context.MODE_APPEND
　MODE_APPEND：模式会检查文件是否存在，存在就往文件追加内容，否则就创建新文件。
　MODE_WORLD_READABLE：表示当前文件可以被其他应用读取；
　MODE_WORLD_WRITEABLE：表示当前文件可以被其他应用写入。
读写sdcard上的文件

其中读写步骤按如下进行:

1、调用Environment的getExternalStorageState()方法判断手机上是否插了sd卡,且应用程序具有读写SD卡的权限，如下代码将返回true

Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)

2、调用Environment.getExternalStorageDirectory()方法来获取外部存储器，也就是SD卡的目录,或者使用"/mnt/sdcard/"目录

3、使用IO流操作SD卡上的文件

注意点：手机应该已插入SD卡，对于模拟器而言，可通过mksdcard命令来创建虚拟存储卡

> 必须在`AndroidManifest.xml`上配置读写`SD`卡的权限

3. SQLite数据库存储数据

SQLiteDatabase类为我们提供了很多种方法，上面的代码中基本上囊括了大部分的数据库操作；对于添加、更新和删除来说，我们都可以使用

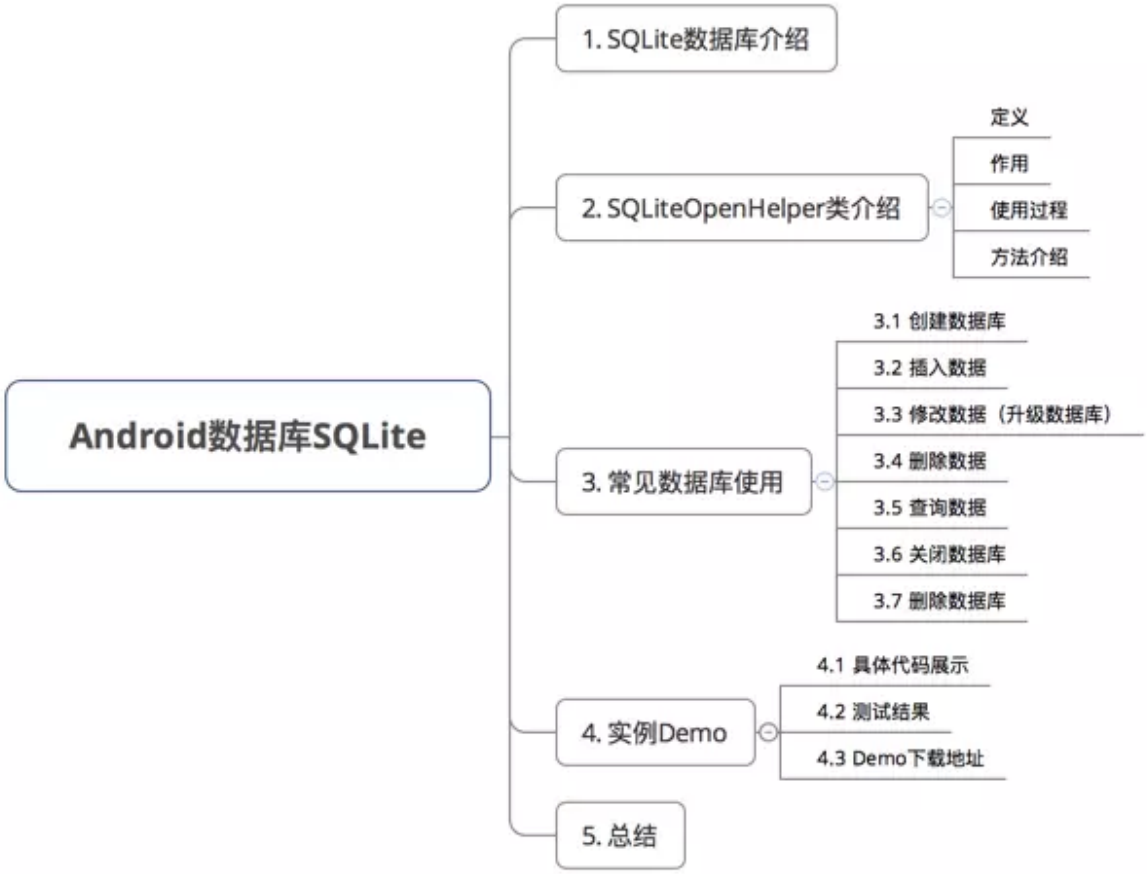
1. db.executeSQL(String sql);
2. db.executeSQL(String sql, Object[] bindArgs);//sql语句中使用占位符，然后第二个参数是实际的参数集
   除了统一的形式之外，他们还有各自的操作方法：
3. db.insert(String table, String nullColumnHack, ContentValues values);
4. db.update(String table, Contentvalues values, String whereClause, String whereArgs);
5. db.delete(String table, String whereClause, String whereArgs);
   以上三个方法的第一个参数都是表示要操作的表名；insert中的第二个参数表示如果插入的数据每一列都为空的话，需要指定此行中某一列的名称，系统将此列设置为NULL，不至于出现错误；insert中的第三个参数是ContentValues类型的变量，是键值对组成的Map，key代表列名，value代表该列要插入的值；update的第二个参数也很类似，只不过它是更新该字段key为最新的value值，第三个参数whereClause表示WHERE表达式，比如“age > ? and age < ?”等，最后的whereArgs参数是占位符的实际参数值；delete方法的参数也是一样

数据的添加

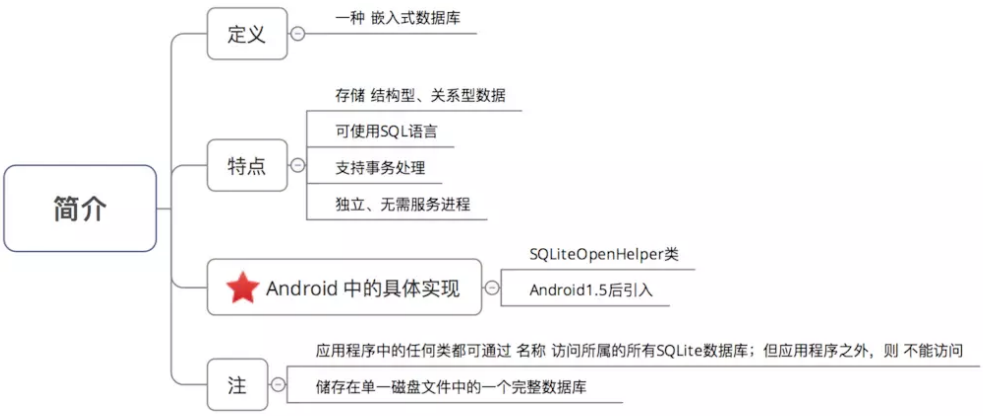
使用insert方法

```
ContentValues cv = new ContentValues();//实例化一个ContentValues用来装载待插入的数据
cv.put("title","you are beautiful");//添加title
cv.put("weather","sun"); //添加weather
cv.put("context","xxxx"); //添加context
String publish = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")
                        .format(new Date());
cv.put("publish ",publish); //添加publish
db.insert("diary",null,cv);//执行插入操作
```

SQLlite数据库

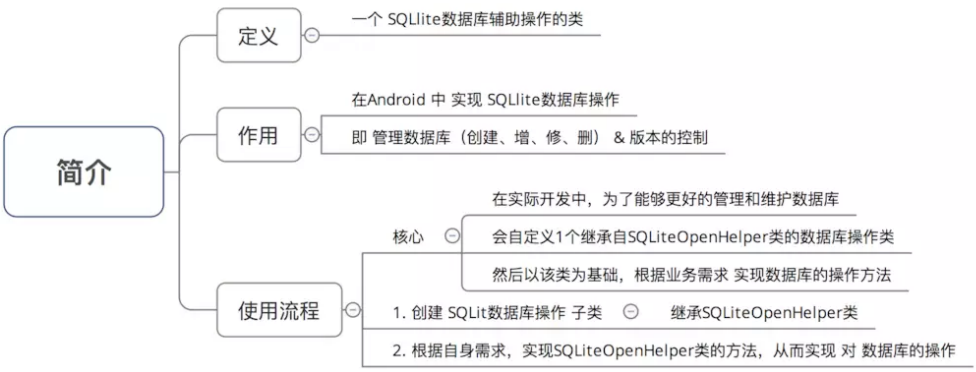


1. SQLlite数据库 简介



2. SQLiteOpenHelper类

2.1 简介



2.2 SQLiteOpenHelper类 常用方法

```
/**
  *  创建数据库
  */
 // 1. 创建 or 打开 可读/写的数据库（通过 返回的SQLiteDatabase对象 进行操作）
 getWritableDatabase（）

 // 2. 创建 or 打开 可读的数据库（通过 返回的SQLiteDatabase对象 进行操作）
 getReadableDatabase（）

 // 3. 数据库第1次创建时 则会调用，即 第1次调用 getWritableDatabase（） /
getReadableDatabase（）时调用
 // 在继承SQLiteOpenHelper类的子类中复写
 onCreate(SQLiteDatabase db)

 // 4. 数据库升级时自动调用
 // 在继承SQLiteOpenHelper类的子类中复写
 onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)

 // 5. 关闭数据库
 close（）

 /**
  *  数据库操作（增、删、减、查）
  */
 // 1. 查询数据
 (Cursor) query(String table, String[] columns, String selection, String[]
selectionArgs, String groupBy, String having, String orderBy, String limit)
 // 查询指定的数据表返回一个带游标的数据集。
 // 各参数说明：
 // table：表名称
 // colums：列名称数组
 // selection：条件子句，相当于where
 // selectionArgs：条件语句的参数数组
 // groupBy：分组
 // having：分组条件
 // orderBy：排序类
```

```
 // limit：分页查询的限制
 // Cursor：返回值，相当于结果集ResultSet

 (Cursor) rawQuery(String sql, String[] selectionArgs)
 //运行一个预置的SQL语句，返回带游标的数据集（与上面的语句最大的区别 = 防止SQL注入）

 // 2. 删除数据行
 (int) delete(String table,String whereClause,String[] whereArgs)

 // 3. 添加数据行
 (long) insert(String table,String nullColumnHack,ContentValues values)

 // 4. 更新数据行
(int) update(String table, ContentValues values, String whereClause, String[]
whereArgs)

 // 5. 执行一个SQL语句，可以是一个select or 其他sql语句
 // 即 直接使用String类型传入sql语句 & 执行
 (void) execSQL(String sql)
```

3. 具体使用

1. 自定义数据库子类

```
/**
  * 创建数据库子类，继承自SQLiteOpenHelper类
  * 需 复写 onCreat（）、onUpgrade（）
  */
public class DatabaseHelper extends SQLiteOpenHelper {

    // 数据库版本号
    private static Integer Version = 1;

    /**
     * 构造函数
     * 在SQLiteOpenHelper的子类中，必须有该构造函数
     */
    public DatabaseHelper(Context context, String name,
SQLiteDatabase.CursorFactory factory,
                          int version) {
        // 参数说明
        // context：上下文对象
        // name：数据库名称
        // param：一个可选的游标工厂（通常是 Null）
        // version：当前数据库的版本，值必须是整数并且是递增的状态

        // 必须通过super调用父类的构造函数
        super(context, name, factory, version);
    }

    /**
     * 复写onCreate（）
     * 调用时刻：当数据库第1次创建时调用
     * 作用：创建数据库 表 & 初始化数据
     * SQLite数据库创建支持的数据类型： 整型数据、字符串类型、日期类型、二进制
     */
    @Override
```

```
    public void onCreate(SQLiteDatabase db) {
                // 创建数据库1张表
                // 通过execSQL（）执行SQL语句（此处创建了1个名为person的表）
                String sql = "create table person(id integer primary key
autoincrement,name varchar(64),address varchar(64))";
                db.execSQL(sql);

                // 注：数据库实际上是没被创建 / 打开的（因该方法还没调用）
                // 直到getWritableDatabase() / getReadableDatabase() 第一次被调用时才
会进行创建 / 打开
    }

    /**
     * 复写onUpgrade（）
     * 调用时刻：当数据库升级时则自动调用（即 数据库版本 发生变化时）
     * 作用：更新数据库表结构
     * 注：创建SQLiteOpenHelper子类对象时,必须传入一个version参数，该参数 = 当前数据库版
本，若该版本高于之前版本，就调用onUpgrade()
     */

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // 参数说明：
        // db ： 数据库
        // oldVersion ： 旧版本数据库
        // newVersion ： 新版本数据库

        // 使用 SQL的ALTER语句
        String sql = "alter table person add sex varchar(8)";
        db.execSQL(sql);
    }

}
```

2. 创建数据库：getWritableDatabase（）、getReadableDatabase（）

```
// 步骤1：创建DatabaseHelper对象
 // 注：此时还未创建数据库
 SQLiteOpenHelper dbHelper = new
DatabaseHelper(SQLiteActivity.this,"test_carson");

// 步骤2：真正创建 / 打开数据库
 SQLiteDatabase sqliteDatabase = dbHelper.getWritableDatabase(); // 创建 or 打开
可读/写的数据库
 SQLiteDatabase sqliteDatabase = dbHelper.getReadableDatabase(); // 创建 or 打开
可读的数据库
```

注：当需操作数据库时，都必须先创建数据库对象 & 创建 / 打开数据库。

- 对于操作 = “增、删、改（更新）”，需获得 可"读 / 写"的权限：getWritableDatabase()
- 对于操作 = “查询”，需获得 可"读 "的权限getReadableDatabase()

3. 操作数据库（增、删、查、改）

4. 使用ContentProvider存储数据

5. 网络存储数据

小结

- Android中本地存储的数据都只能是少量的数据，实际开发中，大量的数据还是存储在服务器端；
- 之前不清楚文件存储还可以存储在内存中，经过学习懂了文件存储的两种方式；
- 数据库这边还需要部分练习来强化映像；
- ContentProvider存储数据和网络存储，就先暂时没有过多的了解；