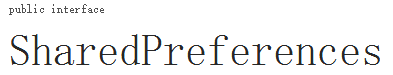
Android数据存储有5中方式：

1. 使用SharedPreferences
2. 文件存储数据
3. SQLite数据库存储数据
4. 使用ContentProvider存储数据
5. 网络存储数据
6. SharePreperences是轻量级的数据存储



它的本质是基于XML文件存储key-value键值对数据，通常用来存储一些简单的配置信息。其存储位置在**/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**Editor**对象实现。

实现SharedPreferences存储的步骤如下：

1. 根据Context获取SharedPreferences对象

Context ctx = MainActivity.this;

SharedPreferences sp = ctx.getSharedPreferences(String name, int mode);

参数name：若该文件不存在，则会在创建Editor时创建。

参数mode：三种形式

[MODE\_PRIVATE](http://android.xsoftlab.net/reference/android/content/Context.html#MODE_PRIVATE): 指定该SharedPreferences数据只能被本应用程序读、写。

[MODE\_WORLD\_READABLE](http://android.xsoftlab.net/reference/android/content/Context.html#MODE_WORLD_READABLE): 指定该SharedPreferences数据能被其他应用程序读，但不能写。

[MODE\_WORLD\_WRITEABLE](http://android.xsoftlab.net/reference/android/content/Context.html#MODE_WORLD_WRITEABLE): 指定该SharedPreferences数据能被其他应用程序读，写。

1. 利用edit()方法获取Editor对象。

Editor editor = sp.edit();

1. 通过Editor对象存储key-value键值对数据。

editor.putString("STRING\_KEY", "string");

editor.putInt("INT\_KEY", 0);

editor.putBoolean("BOOLEAN\_KEY", true);

4、通过commit()方法提交数据。

editor.commit();

上述是在本程序中进行数据存储，若需要操作另外一个应用中的Sharepreference。

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(String packname, Context.CONTEXT\_IGNORE\_SECURITY);这里的packname就是其他程序的包

1. 使用其他程序的Context获取对应的SharedPreferences

  SharedPreferences read = pvCount**.**getSharedPreferences("lock", Context.MODE\_WORLD\_READABLE);

1. 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。

获得SharePreferences中的数据，用SharePreferences对象的getString(String key , String default);

若没有对应的键，则返回值为default。

但是Shareferences中能存储boolean、float、int、String、long这五种基本数据类型

1. 文件存储数据

文件存储和j2SE环境中保存数据到文件是一样的。可以用来存放大量的数据，如文本图片等。

其默认的存储位置： **/data/data/< >/files/\*\*\*.\*\*\***

1. Context提供了2个方法用来打开数据文件里的文件IO流:

**FileInputStream openFileInput(Stirng name);**

**和FileOutputStream openFileOutput(String name,int MODE);**

其中name用于指定文件名称，但是不能包含路径分隔符；

Mode表示指定打开文件的模式。

Mode有以下四中模式：

[MODE\_APPEND](http://android.xsoftlab.net/reference/android/content/Context.html#MODE_APPEND)（32768）：该模式会检查文件是否存在，存在就往文件种追加内容，否则就新创建文件.

[MODE\_PRIVATE](http://android.xsoftlab.net/reference/android/content/Context.html#MODE_PRIVATE)（0）：默认操作模式，表示该文件是私有的数据，只能被应用本身访问，在该模式下，写入的内同会覆盖原文件的内容，如果想把新的内容追加到原文件的后面，选择MODE\_APPEND.

[MODE\_WORLD\_READABLE](http://android.xsoftlab.net/reference/android/content/Context.html#MODE_WORLD_READABLE)（1）和[MODE\_WORLD\_WRITEABLE](http://android.xsoftlab.net/reference/android/content/Context.html" \l "MODE_WORLD_WRITEABLE)（2）：用来控制其他应用是否有权限读写该文件。（1）表示当前文件可以被其他的文件读取.(2)表示可以被其他的应用写入.

如果希望文件被其他的文件度和写，可以将模式设置为：MODE\_WORLD\_READABLE + MODE\_WORLD\_WRITEABLE.

**一些常用的函数：**

**getDir(String name , int mode)**:在应用程序的数据文件夹下获取或者创建name对应的子目录

**File getFilesDir()**:获取该应用程序的数据文件夹得绝对路径

**String[] fileList()**:返回该应用数据文件夹的全部文件

利用openFileOutput()方法保存文件，文件是保存在手机空间上的，一般手机的存储空间不是很大，存放些小文件还行，但是要存储大文件，我们可以把文件存在SDCard中。

1. 除了文件流，Activity还提供了读取SDCard的操作



读写SDCard卡上的文件，需要用到Environment类去获取SDCard的目录，

Environment中的一些静态常量，用于标识外部存储的状态：

|  |  |  |
| --- | --- | --- |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_BAD\_REMOVAL](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_BAD_REMOVAL) | 在没有挂载前存储媒体已经被移除 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_CHECKING](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_CHECKING) | 正在检查存储媒体 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_EJECTING](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_EJECTING) | 存储媒体在访问过程中被弹出 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_MOUNTED](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_MOUNTED) | 存储媒体已经挂载，并且挂载点可读/写 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_MOUNTED\_READ\_ONLY](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_MOUNTED_READ_ONLY) | 存储媒体已经挂载，挂载点只读 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_NOFS](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_NOFS) | 存储媒体是空白或是不支持的文件系统 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_REMOVED](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_REMOVED) | 存储媒体被移除 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_SHARED](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_SHARED) | 存储媒体正在通过USB共享 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_UNKNOWN](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_UNKNOWN) | 不支持的媒体等，无法识别 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_UNMOUNTABLE](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_UNMOUNTABLE) | 存储媒体无法挂载 |
| [String](http://android.xsoftlab.net/reference/java/lang/String.html) | [MEDIA\_UNMOUNTED](http://android.xsoftlab.net/reference/android/os/Environment.html#MEDIA_UNMOUNTED) | 存储媒体没有挂载 |

步骤：

（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File file = 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file.

注意：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

<uses-permission android:name="android.permission.**MOUNT\_UNMOUNT\_FILESYSTEMS**"/>  
<uses-permission android:name="android.permission.**WRITE\_EXTERNAL\_STORAGE**"/>

1. **SQLite数据库数据存储**

SQLite是一个轻量级、嵌入式的关系型数据库。

1、 Android中数据库操作的核心类及其接口

1）android.database.sqlite.SQLiteDatabase（类）：完成数据的CRUD(增删改查)操作及事务处理。

2）android.database.sqlite.SQLiteOpenHelper（类）：定义数据库的创建以及更新操作类

3）android.database.Cursor（接口）：保存所有的查询结果

4）android.content.ContentValues（类）：对传递的数值进行封装

2、 SQLite3中存放的数据类型类有5种：

**NULL、INTEGER、REAL、TEXT、BLOB**

在SQLite 3 数据库中，任何列，除了INTEGER PRIMARY KEY列，都可以存储任何存储类的值。

为了最大化SQLite和其他数据库引擎之间的兼容性，SQLite支持列的”类型相像“的概念。

列相像是由列声明的类型确定的，规则是按照下面的顺序：

1. 如果声明的类型包含字符串“INT”那么它被赋予INTEGER相像。

2. 如果列声明的类型包含任何字符串“CHAR”，“CLOB”，或“TEXT”，那么此列拥有TEXT相像。注意类型VARCHAR包含“CHAR”，因此也会赋予TEXT相像。

3. 如果列声明的类型包含“BLOB”或没有指定类型，那么此列拥有NONE相像。

4. 如果列声明的类型包含任何“REAL”，“FLOA”，或“DOUB”，那么此列拥有REAL相像。

5. 其他情况，相像是NUMERIC。

注意规则的顺序是重要的。声明类型为“CHARINT”的列同时匹配规则1和规则2，但第一个规则会优先采用，因此此列的相像是INTEGER。

http://www.sqlite.org/datatype3.html（SQLite3列相像的示例）

|  |  |  |
| --- | --- | --- |
| **Example Typenames From The CREATE TABLE Statement or CAST Expression** | **Resulting Affinity** | **Rule Used To Determine Affinity** |
| INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 INT8 | INTEGER | 1 |
| CHARACTER(20) VARCHAR(255) VARYING CHARACTER(255) NCHAR(55) NATIVE CHARACTER(70) NVARCHAR(100) TEXT CLOB | TEXT | 2 |
| BLOB *no datatype specified* | BLOB | 3 |
| REAL DOUBLE DOUBLE PRECISION FLOAT | REAL | 4 |
| NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME | NUMERIC | 5 |

3、在SQLiteDataBase类中提供了对数据进行操作的各种方法，对于增删改，可以使用：

1）db.exectue(String sql);

2) db.exectue(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

在SQLiteDataBase中也提供了各自的操作方法：

1. [insert](http://android.xsoftlab.net/reference/android/database/sqlite/SQLiteDatabase.html#insert(java.lang.String, java.lang.String, android.content.ContentValues))([String](http://android.xsoftlab.net/reference/java/lang/String.html) table, [String](http://android.xsoftlab.net/reference/java/lang/String.html) nullColumnHack, [ContentValues](http://android.xsoftlab.net/reference/android/content/ContentValues.html) values)
2. [update](http://android.xsoftlab.net/reference/android/database/sqlite/SQLiteDatabase.html#update(java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String[]))([String](http://android.xsoftlab.net/reference/java/lang/String.html) table, [ContentValues](http://android.xsoftlab.net/reference/android/content/ContentValues.html) values, [String](http://android.xsoftlab.net/reference/java/lang/String.html) whereClause, [String[]](http://android.xsoftlab.net/reference/java/lang/String.html) whereArgs)
3. [delete](http://android.xsoftlab.net/reference/android/database/sqlite/SQLiteDatabase.html#delete(java.lang.String, java.lang.String, java.lang.String[]))([String](http://android.xsoftlab.net/reference/java/lang/String.html) table, [String](http://android.xsoftlab.net/reference/java/lang/String.html) whereClause, [String[]](http://android.xsoftlab.net/reference/java/lang/String.html) whereArgs)

参数table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，这样就不会错误；insert的第三个参数ContentValues类型的变量，是键值对组成的Map，key表示列名，value表示该列插入的值；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。

对于数据查询操作，有以下几种查询方式：

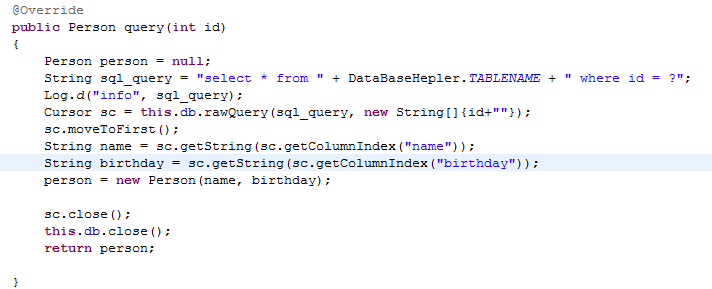
1. db.rawQuery(String sql, String[] selectionArgs);
2. db.query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy);
3. db.query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit);
4. db.query(String distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit);

各方法中的参数说明：

* table：表名称
* colums：表示要查询的列所有名称集
* selection：表示WHERE之后的条件语句，可以使用占位符
* selectionArgs：条件语句的参数数组
* groupBy：指定分组的列名
* having：指定分组条件,配合groupBy使用
* orderBy：y指定排序的列名
* limit：指定分页参数
* distinct：指定“true”或“false”表示要不要过滤重复值
* Cursor：返回值，相当于结果集ResultSet

上述各种数据查询方法都返回一个游标： Cursor对象，代表数据集的游标。

* c.move(int offset); //以当前位置为参考,移动到指定行
* c.moveToFirst(); //移动到第一行
* c.moveToLast(); //移动到最后一行
* c.moveToPosition(int position); //移动到指定行
* c.moveToPrevious(); //移动到前一行
* c.moveToNext(); //移动到下一行
* c.isFirst(); //是否指向第一条
* c.isLast(); //是否指向最后一条
* c.isBeforeFirst(); //是否指向第一条之前
* c.isAfterLast(); //是否指向最后一条之后
* c.isNull(int columnIndex); //指定列是否为空(列基数为0)
* c.isClosed(); //游标是否已关闭
* c.getCount(); //总数据项数
* c.getPosition(); //返回当前游标所指向的行数
* c.getColumnIndex(String columnName);//返回某列名对应的列索引值
* c.getString(int columnIndex); //返回当前行指定列的值



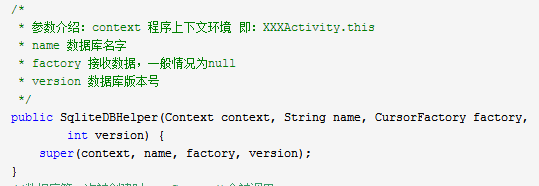
该方法是以唯一ID查询的，所有只有一个返回集，在用Cursor游标进行查询遍历时，**先要把游标指向返回集合的第一个：sc.moveToFirst()**

当我们完成了对数据库的操作后，记得调用SQLiteDatabase的close()方法释放数据库连接，否则容易出现SQLiteException。

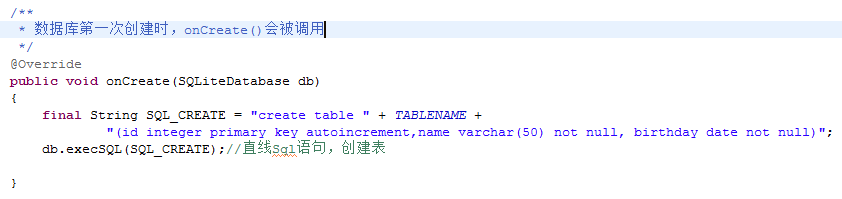
4、在开发中，我们需要封装一个继承与SQLiteOpenHelper类的数据库操作类，然后以这个类为基础，再封装我们的业务逻辑方法。

继承SQLiteOpenHelper会重新2个方法和创建一个构造函数：

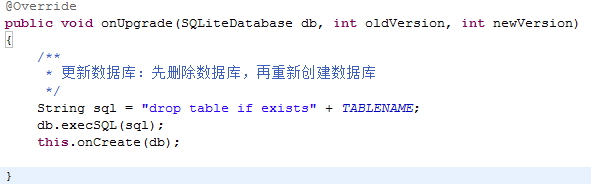
1). 编写构造函数：用于创建



2). 重新onCreate(SQLiteDatabase db)

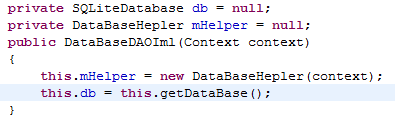


3). 重新onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)



5、创建一个DAO类，封装程序所需要的逻辑业务；或者创建一个接口，其中包含所有业务所需要的方法，再创建一个方法实现该接口，并编写实现相应的业务逻辑。

在Dao构造方法中实例化sqliteDBHelper并获取一个SQLiteDatabase对象，作为整个应用的数据库实例；



6、Android SDK提供了一个工具，用于管理sqlite3数据库。

Android SDK的tools目录下提供了一个sqlite3.exe工具，这是一个简单的sqlite数据库管理工具。开发者可以方便的使用其对sqlite数据库进行命令行的操作。

程序运行生成的\*.db文件一般位于"/data/data/项目名(包括所处包名)/databases/\*.db"，因此要对数据库文件进行操作需要先找到数据库文件：

1、进入shell 命令

adb shell

2、找到数据库文件

#cd data/data

#ls --列出所有项目

#cd project\_name --进入所需项目名

#cd databases

#ls --列出现寸的数据库文件

3、进入数据库

#sqlite3 test\_db --进入所需数据库

会出现类似如下字样：

SQLite version 3.6.22

Enter ".help" for instructions

Enter SQL statements terminated with a ";"

sqlite>

至此，可对数据库进行sql操作。

4、sqlite常用命令

>.databases --产看当前数据库

>.tables --查看当前数据库中的表

>.help --sqlite3帮助

>.schema --各个表的生成语句

1. 用ContentProvider存储数据

一个程序可以通过实现一个ContentProvider的抽象接口将自己的数据完全暴露出去，而且ContentProviders是以类似数据库中表的方式将数据暴露，也就是说ContentProvider就像一个“数据库”。那么外界获取其提供的数据，也就应该与从数据库中获取数据的操作基本一样，只不过是采用URI来表示外界需要访问的“数据库”。

ContentProvider也提供了类似于数据库增删改查的几个数据操作方法。主要区别就在于，每个方法的第一个参数都是一个URI，标识的是该操作应该对哪一个应用程序中的数据进行操作，这个URI由共享数据的应用提供。根据此URI就可以对获得数据并进行相应的操作。

具体见：android四大组件之ContentProvider

1. 网络存储数据

可以调用WebService返回的数据或是解析HTTP协议实现网络数据交互。具体需要熟悉java.net.\*，Android.net.\*这两个包的内容