

Android 数据库技术

- [Android SQLite 解析](#) 
- [ContentResolver](#)
- [ContentProvider](#)
- [Android 数据存储（总结篇）](#) 

Android SQLite 解析

说到 SQLite，无论 C++、Java 程序员还是其他的非主流程程序员，应该都听说过它，可见其非常流行。SQLite 是轻量级的、嵌入式的、关系型数据库，目前已经在 iPhone、Android 等手机系统中使用，而且被其他的公司广泛使用，比如说：Adobe，具体 SQLite 的介绍可以到其[官方网站](#)浏览。

在学习 Android SQLite 前，必须对 SQL 语句有很深入的了解（如果忘记了，利用这次机会好好复习下）。关于数据库、表的创建等基础知识，由于篇幅有限就不在此详细说明，主要说明数据库的 4 大基本操作：添加(insert)、删除(delete)、查询(query)、修改(update)，这是在学习 Android SQLite 的过程中最为关注的部分。除了这 4 大操作以外，我们还需要注意的地方就是：获取查询结果的记录集（Recordset）。

Android SQLite 分析

等有了这些基本概念，再来学习 Android SQLite，从 Android SDK 中摘要如下：

Interfaces

SQLiteCursorDriver	A driver for SQLiteCursors that is used to create them and gets notified by the cursors on significant events in their lifetimes.
SQLiteDatabase.CursorFactory	Used to allow returning sub-classes of Cursor when calling query.

Classes

SQLiteClosable	An object create from a SQLiteDatabase that can be closed.
SQLiteCursor	A Cursor implementation that exposes results from a query on a SQLiteDatabase .
SQLiteDatabase	Exposes methods to manage a SQLite database.
SQLiteOpenHelper	A helper class to manage database creation and version management.
SQLiteProgram	A base class for compiled SQLite programs.
SQLiteQuery	A SQLite program that represents a query that reads the resulting rows into a CursorWindow .
SQLiteQueryBuilder	This is a convenience class that helps build SQL queries to be sent to SQLiteDatabase objects.
SQLiteStatement	A pre-compiled statement against a SQLiteDatabase that can be reused.

Exceptions

首先关注到的是 [SQLiteDatabase](#) 类，在 Android SDK 中看其详细说明，其主要接口如下：

返回值	函数原型
long	insert (String table, String nullColumnHack, ContentValues values) Convenience method for inserting a row into the database.
Int	delete (String table, String whereClause, String[] whereArgs) Convenience method for deleting rows in the database.
Cursor	query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit) Query the given table, returning a Cursor over the result set.
Int	update (String table, ContentValues values, String whereClause, String[] whereArgs) Convenience method for updating rows in the database.

看到这些是不是感到似曾相识了？不错，这就是数据库的 4 大基本操作：添加 (insert)、删除 (delete)、查询 (query)、修改 (update)。感觉轻松很多了，再仔细看下这些函数的参数，发现很多参数都是按照 SQL 语句定义的，

例如：select * from p_employee where id <> '9999' order by id asc

需要额外说明 2 个数据结构：ContentValues, Cursor;

ContentValues 就相当于 C++ 中的 map[(String key, Integer value)]，其主要接口包括 put(), get()。再结合 insert 操作的主要目的：按照表中数据段将对

应的数据项写入到表中，就可以看出来 ContentValues 主要是存放表中每个表的数据段，以及其对应的值。

Cursor 也就是前面说的：查询结果的记录集。从记录集的特征可以想到其包含的操作应该有：MoveFirst()、MoveLast()、MoveNext()、Move()、IsLast()、GetColumns() 等，而且它还是个抽象类[abstract class]，SQLiteCursor 就是其具体的实现。

关于 SQLiteDatabase 类，其他值得关注的函数是：Create()、execSQL()。至于其他的函数，比如 beginTransaction()，endTransaction() 等关于数据库同步操作的函数，就只有在使用的过程中深入了解。

额外补充说明

最后值得一提的是：一个很好的辅助类 SQLiteOpenHelper，其简化了数据库的操作。按照 Android SDK 文档中的说明，也可以通过继承此类、改写其接口的方法来实现对数据库的操作，SQLiteOpenHelper，其主要接口如下：

Public Constructors	
	<code>SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)</code> Create a helper object to create, open, and/or manage a database.

Public Methods	
synchronized void	<code>close()</code> Close any open database object.
synchronized <code>SQLiteDatabase</code>	<code>getReadableDatabase()</code> Create and/or open a database.
synchronized <code>SQLiteDatabase</code>	<code>getWritableDatabase()</code> Create and/or open a database that will be used for reading and writing.
abstract void	<code>onCreate(SQLiteDatabase db)</code> Called when the database is created for the first time.
void	<code>onOpen(SQLiteDatabase db)</code> Called when the database has been opened.

仔细分析其说明，发现这个类主要是创建一个数据库，并对数据库的版本进行管理。当在程序中调用这个类的方法 `getWritableDatabase()` 或者 `getReadableDatabase()` 方法的时候，如果当时没有数据库，那么 Android 系统就会自动创建一个数据数据库。

其他辅助说明

- 所有的数据库文件存放在手机中的/data/data/package_name/databases 路径下。最为重要的一点是：在 Android 中，所有的应用软件的数据（包括文件、数据库）为该应用软件所私有的，如果需要在不同的应用中共享数据，必须使用 **ContentProvider** 实现，一个 **ContentProvider** 类实现了一组标准的方法接口，从而能够让其他的应用保存或读取此 **ContentProvider** 的各种数据类型。
- 为了方便测试，如何查看 SQLite 数据库的内容？使用 Android SDK 提供的工具：sqlite3.exe。这是一个命令行实用工具，列举主要使用的命令如下：[查看表结构](#) sqlite3 "path" .dump
例如 G:\android\android-sdk-window\tools>sqlite3 30m.db3 .bump,
运行结构如下：

```
BEGGIN TRANSACTION;  
CREATE TABLE [TmProgramm] ([index] INTEGER,[nID] INTEGER, [nChannelID]  
INTEGER,[strTitle] CHAR(255),  
[dwDuration] INTEGER, [strThumbNail] CHAR(1024),[strURL] CHAR(255),[strPathName]  
CHAR(255),  
[strAuthor] CHAR(255),[dwSize] INTEGER,[State] INTEGER,[Percent] INTEGER,  
[strConverted] CHAR(255),[dwVideoType] INTEGER,[dwDownloadedBytes]  
INTEGER,[strTime] CHAR(255),  
[strHTTPVersion] CHAR(255),[strUserAgent] CHAR(255),[strReferer] CHAR(255));  
COMMIT;
```

这就是当初创建数据库的语句

[查看表中的内容](#) sqlite3 "path" .schema

具体运行结果可以[下载数据库文件 30m.db3](#)，在命令行中运行就可以看到了
其他帮助命令查询 sqlite3 .help

总结说明

通过对以上的学习，简单写了个数据库的例子（[moandroid.database.zip](#)），大家可以参考下。

相关文章

- [Activity 、Intent 深入解析](#)
- [Android 实现联网（一）——package 说明](#)
- [Android 画图学习总结（一）——类的简介](#)
- [Cocos2d Android 移植手记（一）——Opengl ES 创建流程](#)
- [Android SDK 深入解析— Application Components](#)

Android 应用程序之间数据共享—ContentResolver

Android 是如何实现应用程序之间数据共享的？一个应用程序可以将自己的数据完全暴露出去，外界更本看不到，也不用看到这个应用程序暴露的数据 是如何存储的，或者是使用数据库还是使用文件，还是通过网上获得，这些一切都不重要，重要的是外界可以通过这一套标准及统一的接口和这个程序里的数据打交道，例如：添加(insert)、删除(delete)、查询(query)、修改(update)，当然需要一定的权限才可以。

如何将应用程序的数据暴露出去？ Android 提供了 ContentProvider，一个程序可以通过实现一个 Content provider 的抽象接口将自己的数据完全暴露出去，而且 Content providers 是以类似数据库中表的方式将数据暴露。Content providers 存储和检索数据，通过它可以让所有的应用程序访问到，这也是应用程序之间唯一共享数据的方法。要想使应用程序的数据公开化，可通过 2 种 方法：创建一个属于你自己的 Content provider 或者将你的数据添加到一个已经存在的 Content provider 中，前提是有相同数据类型并且有写入 Content provider 的权限。

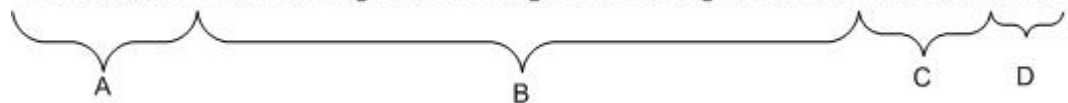
如何通过一套标准及统一的接口获取其他应用程序暴露的数据？ Android 提供了 ContentResolver，外界的程序可以通过 ContentResolver 接口访问 ContentProvider 提供的数据。

当前篇主要说明，如何获取其它应用程序共享的数据，比如获取 Android 手机电话簿中的信息。

什么是 URI？

在学习如何获取 ContentResolver 前，有个名词是必须了解的：URI。URI 是网络资源的定义，在 Android 中赋予其更广阔的含义，先看个例子，如下：

`content://com.example.transportationprovider/trains/122`



将其分为 A, B, C, D 4 个部分：

A：标准前缀，用来说明一个 Content Provider 控制这些数据，无法改变的；

B：URI 的标识，它定义了是哪个 Content Provider 提供这些数据。对于第三方应用程序，为了保证 URI 标识的唯一性，它必须是一个完整的、小写的 类名。这个标识在<provider> 元素的 authorities 属性中说明：

```
<provider
```

```
name=".TransportationProvider" authorities="com.example.transportationprovider" . . . >
```

C：路径，Content Provider 使用这些路径来确定当前需要生什么类型的数据，

URI 中可能不包括路径，也可能包括多个；

D: 如果 URI 中包含，表示需要获取的记录的 ID；如果没有 ID，就表示返回全部；由于 URI 通常比较长，而且有时候容易出错，切难以理解。所以，在 Android 当中定义了一些辅助类，并且定义了一些常量来代替这些长字符串，例如：

People.CONTENT_URI

ContentResolver 介绍说明

看完这些介绍，大家一定就明白了，ContentResolver 是通过 URI 来查询 ContentProvider 中提供的数据。除了 URI 以外，还必须知道需要获取的数据段的名称，以及此数据段的数据类型。如果你需要获取一个特定的记录，你就必须知道当前记录的 ID，也就是 URI 中 D 部分。

前面也提到了 Content providers 是以类似数据库中表的方式将数据暴露出去，那么 ContentResolver 也将采用类似数据库的操作来从 Content providers 中获取数据。现在简要介绍 ContentResolver 的主要接口，如下：

返回值	函数声明
final Uri	<code>insert(Uri url, ContentValues values)</code> Inserts a row into a table at the given URL.
final int	<code>delete(Uri url, String where, String[] selectionArgs)</code> Deletes row(s) specified by a content URI.
final Cursor	<code>query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)</code> Query the given URI, returning a Cursor over the result set.
final int	<code>update(Uri uri, ContentValues values, String where, String[] selectionArgs)</code> Update row(s) in a content URI.

看到这里，是否感觉与数据库的操作基本一样的？就是这样的，详细解析请参考 [Android SQLite 解析](#) 篇中的说明，不在此详细说明。

最后一个问题：如何获取 ContentResolver？调用 `getContentResolver()`，例如：`ContentResolver cr = getContentResolver();`

制作 ContentResolver 实例

以上就完全介绍了如何获取、使用 ContentResolver，启动 Eclipses，制作一个完整的实例如下：



打开 showcontent.java, 修改如下:

```
package moandroid.showcontact;

import android.app.ListActivity;

import android.database.Cursor;

import android.os.Bundle;

import android.provider.Contacts.Phones;

import android.widget.ListAdapter;

import android.widget.SimpleCursorAdapter;
```

```
public class showcontact extends ListActivity {

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        Cursor c = getContentResolver().query(Phones.CONTENT_URI,
        null, null, null, null);

        startManagingCursor(c);

        ListAdapter adapter = new SimpleCursorAdapter(this,
        android.R.layout.simple_list_item_2, c,
        new String[] { Phones.NAME, Phones.NUMBER },
        new int[] { android.R.id.text1, android.R.id.text2 });

        setListAdapter(adapter);

    }

}
```

然后在 AndroidManifest.XML 中<application>元素前增加如下许可：
<uses-permission android:name="android.permission.READ_CONTACTS" />
最后运行程序，在模拟器启动后，单击 Menu 返回到 Home 界面，打开 Contacts
选择 Contacts 标签页，添加 2 个联系人信息。返回到 Home，选择

moandroid.showcontact 运行，刚添加的 2 个联系人信息将显示在界面上，如下：



总结说明

ContentResolver 的使用极大的方便了应用程序之间共享数据，如何将应用程序的数据完全暴露给其他应用程序使用了，将在下篇文章 Android 应用程序之间数据共享——ContentProvider 中说明。

相关文章

- [Android 应用程序之间数据共享—ContentProvider](#)

[mo-Android 感受 Android 带给我们的新体验](#)

- [Activity 、Intent 深入解析](#)
- [Android 实现联网（一）——package 说明](#)
- [Android 画图学习总结（一）——类的简介](#)
- [Cocos2d Android 移植手记（一）——Opengl ES 创建流程](#)

Android 应用程序之间数据共享—ContentProvider

在 [Android 应用程序之间数据共享—ContentResolver](#) 中，已经说明了 Android 是如何实现应用程序之间数据共享的，并详细解析了如何获取其他应用程序共享的数据。ContentProviders 存储和检索 数据，通过它可以让所有的应用程序访问到，这也是应用程序之间唯一共享数据的方法。那么如何将应用程序的数据暴露出去？

通过以前文章的学习，知道 ContentResolver 是通过 ContentProvider 来获取其他与应用程序共享的数据，那么 ContentResolver 与 ContentProvider 的接口应该差不多的。

其中 ContentProvider 负责

- 组织应用程序的数据；
- 向其他应用程序提供数据；

ContentResolver 则负责

- 获取 ContentProvider 提供的数据；
- 修改/添加/删除更新数据等；

ContentProvider 是如何向外界提供数据的？

Android 提供了 ContentProvider，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”。至于如何从 URI 中识别出外界需要的是哪个“数据库”，这就是 Android 底层需要做的事情了，不在此详细说。简要分析下 ContentProvider 向外界提供数据操作的接口：

[query\(Uri, String\[\], String, String\[\], String\)](#)

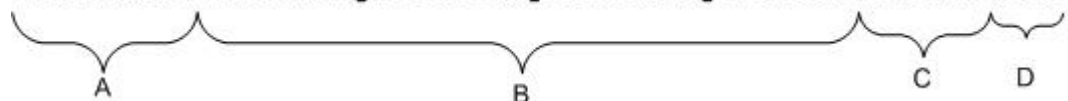
[insert\(Uri, ContentValues\)](#)

[update\(Uri, ContentValues, String, String\[\]\)](#)

[delete\(Uri, String, String\[\]\)](#)

这些操作与数据库的操作基本上完全一样，在此不详细说，具体的解析可以参考 [Android Sqlite 解析](#) 篇中的详细说明。需要特殊说明的地方是 URI：

`content://com.example.transportationprovider/trains/122`



在 URI 的 D 部分可能包含一个 ID，这个应该出现在 SQL 语句中的，可以以种特殊的方式出现，这就要求我们在提供数据的时候，需要来额外关注这个特殊的信息。Android SDK 推荐的方法是：在提供数据表字段中包含一个 ID，在创建表时 INTEGER PRIMARY KEY AUTOINCREMENT 标识此 ID 字段。

ContentProvider 是如何组织数据的？

组织数据主要包括：存储数据，读取数据，以数据库的方式暴露数据。数据的存储需要根据设计的需求，选择合适的存储结构，首选数据库，当然也可以选择本地其他文件，甚至可以是网络上的数据。数据的读取，以数据库的方式暴露数据这就要求，无论数据是如何存储的，数据最后必须以数据的方式访问。

可能还有 2 个问题，是需要关注的。

1. **ContentProvider** 是什么时候创建的，是谁创建的？访问某个应用程序共享的数据，是否需要启动这个应用程序？这个问题在 **Android SDK** 中没有明确说明，但是从数据共享的角度出发，**ContentProvider** 应该是 **Android** 在系统启动时就创建了，否则就谈不上数据共享了。这就要求在 **AndroidManifest.XML** 中使用 `<provider>` 元素明确定义。
2. 可能会有多个程序同时通过 **ContentResolver** 访问一个 **ContentProvider**，会不会导致像数据库那样的“脏数据”？这个问题一方面需要数据库访问的同步，尤其是数据写入的同步，在 **AndroidManifest.XML** 中定义 **ContentProvider** 的时候，需要考虑是 `<provider>` 元素 `multiprocess` 属性的值；另外一方面 **Android** 在 **ContentResolver** 中提供了 `notifyChange()` 接口，在数据改变时会通知其他 **ContentObserver**，这个地方应该使用了观察者模式，在 **ContentResolver** 中应该有一些类似 `register`，`unregister` 的接口。

至此，已经对 **ContentProvider** 提供了比较全面的分析，至于如何创建 **ContentProvider**，可通过 2 种方法：创建一个属于你自己的 **ContentProvider** 或者将你的数据添加到一个已经存在的 **ContentProvider** 中，当然前提是有相同数据类型并且有写入 **Content provider** 的权限。在 **Android SDK** 的 sample 中提供的 [Notepad 具体实例](#) 中去看源代码！

相关文章

[mo-Android 感受 Android 带给我们的新体验](#)

- [Android 应用程序之间数据共享—ContentResolver](#)
- [Activity 、 Intent 深入解析](#)
- [Android 实现联网（一）——package 说明](#)
- [Android 画图学习总结（一）——类的简介](#)
- [Cocos2d Android 移植手记（一）——Opengl ES 创建流程](#)

Android 数据存储（总结篇）

在前面的 2 篇文章：[Android SQLite 解析](#)、[Android 应用程序之间数据共享](#)中分别详细说明了，如何使用数据库存储信息，以及如何通过 ContentProvider 获取其他应用程序共享的数据，现将 Android 数据存储做下总结，在以后的开发过程中根据需求选择合适的数据存储方式。

Android 提供了 5 种方式存储数据：

1. 使用 SharedPreferences 存储数据；
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；

其中 3, 4 已经在 Android SQLite 解析、Android 应用程序之间数据共享篇幅中详细说明，不在此重复说明，现将其他 3 种方式详细介绍。

使用 SharedPreferences 存储数据

首先说明 SharedPreferences 存储方式，它是 Android 提供的用来存储一些简单配置信息的一种机制，例如：登录用户的用户名与密码。其采用了 Map 数据结构来存储数据，以键值的方式存储，可以简单的读取与写入，具体实例如下：

```
void ReadSharedPreferences()
{
String  strName,strPassword;
SharedPreferences  user = getSharedPreferences( "user_info",0);
strName = user.getString( "NAME", " " );
strPassword = user getString( "PASSWORD", " " );
}

void WriteSharedPreferences(String  strName,String strPassword)
{
SharedPreferences  user = getSharedPreferences( "user_info",0);
uer.edit();
```

```
user.putString( "NAME" , strName);
user.putString( "PASSWORD" , strPassword);
user.commit();
}
```

数据读取与写入的方法都非常简单，只是在写入的时候有些区别：先调用 `edit()` 使其处于编辑状态，然后才能修改数据，最后使用 `commit()` 提交修改的数据。实际上 `SharedPreferences` 是采用了 XML 格式将数据存储到设备中，在 DDMS 中的 File Explorer 中的 `/data/data/<package name>/shares_prefs` 下。以上面的数据存储结果为例，打开后可以看到一个 `user_info.xml` 的文件，打开后可以看到：

```
<?xml version=" 1.0" encoding=" UTF-8" ?>
<map>
<string name=" NAME" >moandroid</string>
<string name=" PASSWORD" >SharedPreferences</string>
</map>
```

使用 `SharedPreferences` 是有些限制的：只能在同一个包内使用，不能在不同的包之间使用。

文件存储数据

文件存储方式是一种较常用的方法，在 Android 中读取/写入文件的方法，与 Java 中实现 I/O 的程序是完全一样的，提供了 `openFileInput()` 和 `openFileOutput()` 方法来读取设备上的文件。[FilterInputStream](#)，[FilterOutputStream](#) 等可以到 [Java io package](#) 说明中去详细学习，不再此详细说明，具体实例如下：

```
String fn = "moandroid.log" ;
FileInputStream fis = openFileInput(fn);
FileOutputStream fos = openFileOutput(fn, Context.MODE_PRIVATE);
除此之外,Android还提供了其他函数来操作文件,详细说明请阅读 Android SDK。
```

网络存储数据

网络存储方式，需要与 Android 网络数据包打交道，关于 Android 网络数据包的详细说明，请阅读 Android [SDK 引用了 Java SDK 的哪些 package?](#)。

总结说明

以上 5 中存储方式，在以后的开发过程中，根据设计目标、性能需求、空间需求等找到合适的数据存储方式。Android 中的数据存储都是私有的，其他应用程序都是无法访问的，除非[通过 ContentResolver 获取其他程序共享的数据](#)。

相关文章

- [Activity 、Intent 深入解析](#)
- [Android 实现联网（一）——package 说明](#)
- [Android 画图学习总结（一）——类的简介](#)
- [Cocos2d Android 移植手记（一）——Opengl ES 创建流程](#)
- [Android SDK 深入解析— Application Components](#)