# **[Android探索之ContentProvider熟悉而又陌生的组件](http://www.cnblogs.com/whoislcj/p/5507928.html)**

### **前言：**

    总结这篇文章之前我们先来回顾一下Android Sqlite数据库，参考文章：http://www.cnblogs.com/whoislcj/p/5506294.html，Android程序内部数据存储如果使用Sqlite数据库，那么Android 如何实现程序间数据共享？Android 提供了一种机制可以实现程序间的数据共享，它就是Android 四大组件之一ContentProvider,Android为存储和获取数据提供统一的接口，用于实现程序间数据共享，不要将其理解为数据库。

     为什么说是熟悉又陌生呢？因为我们经常使用到，Android内置的许多数据都是采用ContentProvider，比如图片，视频，音频，手机联系人等，至于陌生那是因为我很少自己去实现一个ContentProvider，今天我们重点是来实现一个自定义ContentProvider。

#### **ContentProvider类简介：**

##### **1.) 我们一般要继承ContentProvider，那么要实现那些函数呢？**

* ContentProvider()   构造函数
* onCreate()    创建数据时调用的回调函数
* insert()      插入数据
* delete()     删除数据
* update()    更新数据
* query()      查询数据
* getType()  得到数据类型

##### **2.)URI简介：**

     ContentProvider通过URI来访问数据执行增删改查的操作，一个完整的URI有 content://自定义ContentProvider/xxx数据库名称

     我们先声明一个作用域：

//访问URI作用域

public static final String CONTENT\_URI="com.whoislcj.testsqlite.personprovider";

    对应URI举例说明一下：

* content://com.whoislcj.testsqlite.personprovider/person   返回person所以数据
* content://com.whoislcj.testsqlite.personprovider/person/10 返回id为10的person数据

##### **3.）UriMatcher简介**

       主要用于匹配Uri，为什么要匹配Uri呢？通过上面的Uri举例可以看出操作域不一样，对于执行一个delete、update、query来说我们要获取参数参数执行不能对应操作。

   使用：

[IMG_256](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

//定义一个UriMatcher类对象，用来匹配Uri的。

private static final UriMatcher uriMatcher;

//集合操作

public static final int INCOMING\_COLLECTION = 1;

//单个ID操作

public static final int INCOMING\_SIGNAL = 2;

static {

//常量UriMatcher.NO\_MATCH表示不匹配任何路径的返回码

uriMatcher = new UriMatcher(UriMatcher.NO\_MATCH);

//如果match()方法匹配com.whoislcj.testsqlite.personprovider/person路径，返回匹配码为1

uriMatcher.addURI(CONTENT\_URI, "person", INCOMING\_COLLECTION);//添加需要匹配uri，如果匹配就会返回匹配码

//如果match()方法匹配content://com.ljq.provider.personprovider/person/230路径，返回匹配码为2

uriMatcher.addURI(CONTENT\_URI, "person/#", INCOMING\_SIGNAL);//#号为通配符

}

[IMG_257](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

##### **4.）ContentUris简介**

      ContentUris是对URI的操作类，比如获取URI路径里的参数，或者给URI拼接一个参数

    举例说明：

* long id = ContentUris.parseId(uri);//从uri中获取id
* Uri rowUri = ContentUris.withAppendedId(uri, rowId);//uri追加id 生成该条数据完整的URI地址

##### **5.）**ContentResolver简介****

       ContentResolver主要用于为外部程序提供增删改查的操作函数，也可以注册观察者来监听数据的变化。

##### **6.）自定义ContentProvider具体实现：**

IMG_258 View Code

##### **7.）外部如何访问**

IMG_259 View Code

##### **8.）如何监听数据变化**

需要注册一个自定义的观察者，当时如下

[IMG_260](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

// 为uri的数据改变注册监听器 getContentResolver().registerContentObserver(

Uri.parse("content://com.whoislcj.testsqlite.personprovider/person"), true,

new Observer(new Handler()));

// 提供方自定义的ContentOberver监听器

private final class Observer extends ContentObserver {

public Observer(Handler handler) {

super(handler);

}

@Override

public void onChange(boolean selfChange, Uri uri) {

// 查询发送邮箱中的短息(处于正在发送状态的短信放在发送箱)

Log.e("MainActivity", "onChange--->uri ：" + uri.toString());

}

}

[IMG_261](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

同样数据操作位置也需要执行如下代码

getContext().getContentResolver().notifyChange(uri, null);

##### **9.）访问权限控制**

声明读写自定义权限

<permission android:name="com.whoislcj.testsqlite.personprovider.read" />

<permission android:name="com.whoislcj.testsqlite.personprovider.write" />

<uses-permission android:name="com.whoislcj.testsqlite.personprovider.read" />

<uses-permission android:name="com.whoislcj.testsqlite.personprovider.write" />

ContentProvider注册声明：

[IMG_262](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

<provider

android:name=".PersonProvider"

android:authorities="com.whoislcj.testsqlite.personprovider"

android:enabled="true"

android:exported="true"

android:readPermission="com.whoislcj.testsqlite.personprovider.read"

android:writePermission="com.whoislcj.testsqlite.personprovider.write">

</provider>

[IMG_263](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

#### **10.）关于getTpye**

        ContentProvider里面一个getType ()函数很多人不知道 这个干嘛的，接下来介绍一下，

[IMG_264](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

// 数据集的MIME类型字符串则应该以vnd.android.cursor.dir/开头

public static final String CONTENT\_TYPE = "vnd.android.cursor.dir/person";

// 单一数据的MIME类型字符串应该以vnd.android.cursor.item/开头

public static final String CONTENT\_TYPE\_ITME = "vnd.android.cursor.item/person";

/\*\*

\* 该方法用于返回当前Url所代表数据的MIME类型。

\*

\* @param uri

\* @return

\*/

@Override

public String getType(Uri uri) {

switch (uriMatcher.match(uri)) {

case INCOMING\_COLLECTION:

return CONTENT\_TYPE;

case INCOMING\_SIGNAL:

return CONTENT\_TYPE\_ITME;

default:

throw new IllegalArgumentException("Unknown URI " + uri);

}

}

[IMG_265](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

假设我们在项目搞了一个联系人列表Activity，我们需要外面来访问这个Activity，首先看下这个Activity的注册声明：

[IMG_266](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

<activity android:name=".TestActivity" android:icon="@mipmap/ic\_launcher">

<intent-filter>

<action android:name="com.whoislcj.testsqlite.personprovider" />

<category android:name="android.intent.category.DEFAULT"/>

<data android:mimeType="vnd.android.cursor.dir/person" />

</intent-filter>

</activity>

[IMG_267](http://www.cnblogs.com/whoislcj/p/javascript:void(0);)

看到上面的mimeType:vnd.android.cursor.dir/person

外部如何启动呢：

Uri uri = Uri.parse("content://com.whoislcj.testsqlite.personprovider/person");

Intent intent = new Intent();

intent.setAction("com.whoislcj.testsqlite.personprovider");

intent.setData(uri);

startActivity(intent);

这样以来系统会去调用你定义的ContentProvider中的getType，去匹配出相应的Activity来实现跳转。