

4.4.1 ContentProvider初探

分类 [Android 基础入门教程](#)

Android 基础入门教程(Q群号 : 153836263)

本节引言：

本节给大家带来的是Android四大组件中的最后一个——ContentProvider(内容提供者)，可能部分读者有疑问了，“Android不是有五大组件的吗？还有个Intent呢？”对的，Intent也是很重要的，但是他只是维系这几个组件间的纽带！Intent我们下一章会讲解！说会这个ContentProvider，我们什么时候会用到他呢？有下面这两种：

- 1.我们想在自己的应用中访问别的应用，或者说一些ContentProvider暴露给我们的一些数据，比如手机联系人，短信等！我们想对这些数据进行读取或者修改，这就需要用到ContentProvider了！
- 2.我们自己的应用，想把自己的一些数据暴露出来，给其他的应用进行读取或操作，我们也可以用到ContentProvider，另外我们可以选择要暴露的数据，就避免了我们隐私数据的泄露！

好像好流弊的样子，其实用起来也很简单，下面我们来对ContentProvider进行学习~ **官方文档**：[ContentProvider](#) 本节我们来讲解下ContentProvider的概念，给大家写几个常用的使用系统ContentProvider的示例，以及自定义ContentProvider！

1.ContentProvider概念讲解：

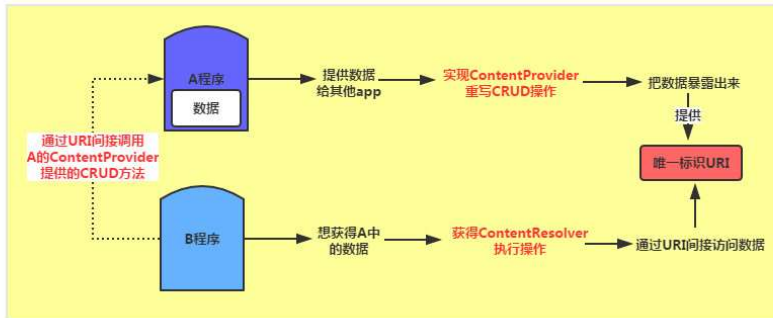
- 1.0 Android基础入门教程
 - 1.0.1 2015年最新Android基...
 - 1.1 背景相关与系统架构分析
 - 1.2 开发环境搭建
 - 1.2.1 使用Eclipse + ADT + S...
 - 1.2.2 使用Android Studio开...
 - 1.3 SDK更新不了问题解决
 - 1.4 Genymotion模拟器安装
 - 1.5.1 Git使用教程之本地仓...
 - 1.5.2 Git之使用GitHub搭建...
 - 1.6 .9(九妹)图片怎么玩
 - 1.7 界面原型设计
 - 1.8 工程相关解析(各种文件...
 - 1.9 Android程序签名打包
 - 1.11 反编译APK获取代码&...
- 2.1 View与ViewGroup的概念
 - 2.2.1 LinearLayout(线性布局)
 - 2.2.2 RelativeLayout(相对布...
 - 2.2.3 TableLayout(表格布局)
 - 2.2.4 FrameLayout(帧布局)
 - 2.2.5 GridLayout(网格布局)
 - 2.2.6 AbsoluteLayout(绝对...
- 2.3.1 TextView(文本框)详解
- 2.3.2 EditText(输入框)详解
- 2.3.3 Button(按钮)与ImageB...
- 2.3.4 ImageView(图像视图)
- 2.3.5.RadioButton(单选按钮...
- 2.3.6 开关按钮ToggleButton...
- 2.3.7 ProgressBar(进度条)
- 2.3.8 SeekBar(拖动条)
- 2.3.9 RatingBar(星级评分条)
- 2.4.1 ScrollView(滚动条)

ContentProvider(内容提供者)

ContentProvider的概述:

当我们想允许自己的应用的数据允许别的应用进行读取操作,我们可以让我们的App实现ContentProvider类,同时注册一个Uri,然后其他应用只要使用ContentResolver根据Uri就可以操作我们的app中的数据了!而数据不一定是数据库,也可能是文件,xml或者其他,但是SharedPreferences使用基于数据库模型的简单表格来提供其中的数据!

ContentProvider的执行原理



URI简介:

专业名词叫做:通用资源标识符,而你也可以类比为网页的域名,我们暂且就把他叫做资源定位符吧,就是定位资源所在路径的而在本书ContentProvider中,Uri非常重要,我们分析一个简单的例子吧:
content://com.jay.example.providers.myprovider/word/2

分析:

content:协议头,这个是规定的,就像http,ftp等一样,规定的,而ContentProvider规定的是content开头的接着是provider所在的全限定类名
word:代表资源部分,如果想访问word所有资源,后面的2就不用写了,直接写word
2:访问的是word资源中id为2的记录

附加

当然,上面也说过数据不仅仅来自于数据库,有时也来源于文件,xml或者网络等其他存储方式,但是依旧可以使用上面这种URI定义方式:
比如:当表示的xml文件时:~/word/detail表示word节点下的detail结点
另外:Uri还提供了一个parse()方法将字符串转换为URI
eg:Uri uri = Uri.parse("Content://~");

2.4.2 Date & Time组件(上)

2.4.3 Date & Time组件(下)

2.4.4 Adapter基础讲解

2.4.5 ListView简单实用

2.4.6 BaseAdapter优化

2.4.7 ListView的焦点问题

2.4.8 ListView之checkbox错...

2.4.9 ListView的数据更新问题

2.5.0 构建一个可复用的自定...

2.5.1 ListView Item多布局的...

2.5.2 GridView(网格视图)的...

2.5.3 Spinner(列表选项框)...

2.5.4 AutoCompleteTextVie...

2.5.5 ExpandableListView(...

2.5.6 ViewPager(翻转视图)...

2.5.7 Toast(吐司)的基本使用

2.5.8 Notification(状态栏通...

2.5.9 AlertDialog(对话框)详解

2.6.0 其他几种常用对话框基...

2.6.1 PopupWindow(悬浮框...

2.6.2 菜单(Menu)

2.6.3 ViewPager的简单使用

2.6.4 DrawerLayout(官方侧...

3.1.1 基于监听的事件处理机制

3.2 基于回调的事件处理机制

3.3 Handler消息传递机制浅析

3.4 TouchListener PK OnTo...

3.5 监听EditText的内容变化

3.6 响应系统设置的事件(Co...

3.7 AsyncTask异步任务

3.8 Gestures(手势)

4.1.1 Activity初学乍练

4.1.2 Activity初窥门径

4.1.3 Activity登堂入室

4.2.1 Service初涉

4.2.2 Service进阶

4.2.3 Service精通

4.3.1 BroadcastReceiver牛...

4.3.2 BroadcastReceiver庖...

4.4.1 ContentProvider初探

2.使用系统提供的ContentProvider

其实很多时候我们用到ContentProvider并不是自己暴露自己的数据,更多的时候通过 **ContentResolver**来读取其他应用的信息,最常用的莫过于读取系统APP,信息,联系人,多媒体信息等!如果你想来调用这些ContentProvider就需要自行查阅相关的API资料了!另外,不同的版本,可能对应着不同的URL!这里给出如何获取URL与对应的数据库表的字段,这里以最常用的联系人为例,其他自行google~

①来到系统源码文件下:all-src.rar -> TeleponeProvider ->

AndroidManifest.xml查找对应API

②打开模拟器的file

exploer/data/data/com.android.providers.contacts/databases/contact2

导出后使用SQLite图形工具查看,三个核心的表:raw_contact表, data表, mimetypes表!

下面演示一些基本的操作示例:

1) 简单的读取收件箱信息:

核心代码:

```
private void getMsgs() {
    Uri uri = Uri.parse("content://sms/");
```

```
ContentResolver resolver = getContentResolver();  
//获取的是哪些列的信息  
Cursor cursor = resolver.query(uri, new String[]{"address",  
"date", "type", "body"}, null, null, null);  
while (cursor.moveToNext())  
{  
    String address = cursor.getString(0);  
    String date = cursor.getString(1);  
    String type = cursor.getString(2);  
    String body = cursor.getString(3);  
    System.out.println("地址:" + address);  
    System.out.println("时间:" + date);  
    System.out.println("类型:" + type);  
    System.out.println("内容:" + body);  
    System.out.println("=====");  
}  
cursor.close();  
}
```

别忘了，往AndroidManifest.xml加入读取收件箱的权限：

```
<uses-permission android:name="android.permission.READ_SMS"/>
```

运行结果：

部分运行结果如下：

```
=====  
地址:1065866983  
时间:1413527884228  
类型:1  
内容:欢迎登录WLAN，如非本人使用可发CZWLANMM到10086重置密码。温馨提醒：2014年4月30日起，非套餐客户资费按0.1元/MB收取  
=====  
地址:1065866983  
时间:1413527867346  
类型:1  
内容:您申请的动态密码为：141296  
=====  
地址:10658000  
时间:1413517039306  
类型:1  
内容:新闻早晚快报讯：原铁道部运输局局长张曙光受贿案一审宣判，张曙光被判处死刑，缓期二年执行。详见：http://isjh.cn/IVnEoVY  
=====
```

2) 简单的往收件箱里插入一条信息

核心代码：

```
private void insertMsg() {  
    ContentResolver resolver = getContentResolver();  
    Uri uri = Uri.parse("content://sms/");  
    ContentValues conValues = new ContentValues();  
    conValues.put("address", "123456789");  
    conValues.put("type", 1);  
    conValues.put("date", System.currentTimeMillis());  
    conValues.put("body", "no zuo no die why you try!");  
    resolver.insert(uri, conValues);  
    Log.e("HeHe", "短信插入完毕~");  
}
```

4.4.2 ContentProvider再探...

4.5.1 Intent的基本使用

4.5.2 Intent之复杂数据的传递

5.1 Fragment基本概述

5.2.1 Fragment实例精讲—...

5.2.2 Fragment实例精讲—...

5.2.3 Fragment实例精讲—...

5.2.4 Fragment实例精讲—...

5.2.5 Fragment实例精讲—...

6.1 数据存储与访问之——文...

6.2 数据存储与访问之——S...

6.3.1 数据存储与访问之——...

6.3.2 数据存储与访问之——...

7.1.1 Android网络编程要学...

7.1.2 Android Http请求头与...

7.1.3 Android HTTP请求方...

7.1.4 Android HTTP请求方...

7.2.1 Android XML数据解析

7.2.2 Android JSON数据解析

7.3.1 Android 文件上传

7.3.2 Android 文件下载 (1)

7.3.3 Android 文件下载 (2)

7.4 Android 调用 WebService

7.5.1 WebView(网页视图)基...

7.5.2 WebView和JavaScrip...

7.5.3 Android 4.4后WebVie...

7.5.4 WebView文件下载

7.5.5 WebView缓存问题

7.5.6 WebView处理网页返...

7.6.1 Socket学习网络基础准备

7.6.2 基于TCP协议的Socket...

7.6.3 基于TCP协议的Socket...

7.6.4 基于UDP协议的Socke...

8.1.1 Android中的13种Draw...

8.1.2 Android中的13种Draw...

8.1.3 Android中的13种Draw...

8.2.1 Bitmap(位图)全解析 P...

8.2.2 Bitmap引起的OOM问题

8.3.1 三个绘图工具类详解

8.3.2 绘图类实战示例

运行结果：



注意事项：

上述代码在4.4以下都可以实现写入短信的功能，而5.0上就无法写入，原因是：从5.0开始，默认短信应用外的软件不能以写入短信数据库的形式发短信！

3) 简单的读取手机联系人

核心代码：

```
private void getContacts() {  
    //①查询raw_contacts表获得联系人的id  
    ContentResolver resolver = getContentResolver();  
    Uri uri = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;  
    //查询联系人数据  
    cursor = resolver.query(uri, null, null, null, null);  
    while (cursor.moveToNext())  
    {  
        //获取联系人姓名,手机号码  
        String cName = cursor.getString(cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME));  
        String cNum = cursor.getString(cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER));  
        System.out.println("姓名:" + cName);  
    }  
}
```

- 8.3.3 Paint API之—— Mask...
- 8.3.4 Paint API之—— Xferm...
- 8.3.5 Paint API之—— Xferm...
- 8.3.6 Paint API之—— Xferm...
- 8.3.7 Paint API之—— Xferm...
- 8.3.8 Paint API之—— Xferm...
- 8.3.9 Paint API之—— Color...
- 8.3.10 Paint API之—— Colo...
- 8.3.11 Paint API之—— Colo...
- 8.3.12 Paint API之—— Path...
- 8.3.13 Paint API之—— Sha...
- 8.3.14 Paint几个枚举/常量值...
- 8.3.15 Paint API之——Type...
- 8.3.16 Canvas API详解(Part 1)
- 8.3.17 Canvas API详解(Part...
- 8.3.18 Canvas API详解(Part...
- 8.4.1 Android动画合集之帧...
- 8.4.2 Android动画合集之补...
- 8.4.3 Android动画合集之属...
- 8.4.4 Android动画合集之属...
- 9.1 使用SoundPool播放音...
- 9.2 MediaPlayer播放音频与...
- 9.3 使用Camera拍照
- 9.4 使用MediaRecord录音
- 10.1 TelephonyManager(电...
- 10.2 SmsManager(短信管理...
- 10.3 AudioManager(音频管...
- 10.4 Vibrator(振动器)
- 10.5 AlarmManager(闹钟服务)
- 10.6 PowerManager(电源服...
- 10.7 WindowManager(窗口...
- 10.8 LayoutInflater(布局服务)
- 10.9 WallpaperManager(壁...
- 10.10 传感器专题(1)——相...
- 10.11 传感器专题(2)——方...
- 10.12 传感器专题(3)——加...
- 10.12 传感器专题(4)——其...
- 10.14 Android GPS初涉
- 11.0 《2015最新Android基...

```
        System.out.println("号码:" + cNum);
        System.out.println("=====");
    }
    cursor.close();
}
```

别忘了加读联系人的权限：

```
<uses-permission android:name="android.permission.READ_CONTACTS"
"/>
```

运行结果：

部分运行结果如下：

```
=====
姓名:啊菊
号码:6303 65
=====
姓名:傻机
号码:633229
=====
姓名:爽
号码:633290
=====
```

4) 查询指定电话的联系人信息

核心代码：

```
private void queryContact(String number) {
    Uri uri = Uri.parse("content://com.android.contacts/data/phones/filter/" + number);
    ContentResolver resolver = getContentResolver();
    Cursor cursor = resolver.query(uri, new String[]{"display_name"}, null, null, null);
    if (cursor.moveToFirst()) {
        String name = cursor.getString(0);
        System.out.println(number + "对应的联系人名称: " + name);
    }
    cursor.close();
}
```

运行结果：

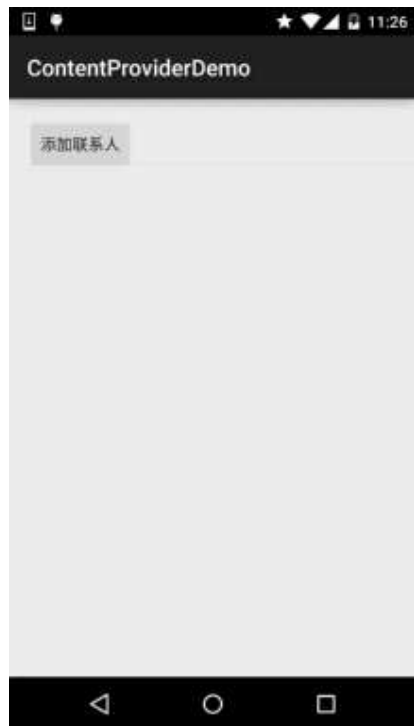
```
I/System.out : 633290对应的联系人名称: 爽
```

5) 添加一个新的联系人

核心代码：

```
private void AddContact() throws RemoteException, OperationAppl  
icationException {  
    //使用事务添加联系人  
    Uri uri = Uri.parse("content://com.android.contacts/raw_con  
tacts");  
    Uri dataUri = Uri.parse("content://com.android.contacts/da  
ta");  
  
    ContentResolver resolver = getContentResolver();  
    ArrayList<ContentProviderOperation> operations = new ArrayL  
ist<ContentProviderOperation>();  
    ContentProviderOperation op1 = ContentProviderOperation.new  
Insert(uri)  
        .withValue("account_name", null)  
        .build();  
    operations.add(op1);  
  
    //依次是姓名，号码，邮编  
    ContentProviderOperation op2 = ContentProviderOperation.new  
Insert(dataUri)  
        .withValueBackReference("raw_contact_id", 0)  
        .withValue("mimetype", "vnd.android.cursor.item/nam  
e")  
        .withValue("data2", "Coder-pig")  
        .build();  
    operations.add(op2);  
  
    ContentProviderOperation op3 = ContentProviderOperation.new  
Insert(dataUri)  
        .withValueBackReference("raw_contact_id", 0)  
        .withValue("mimetype", "vnd.android.cursor.item/pho  
ne_v2")  
        .withValue("data1", "13798988888")  
        .withValue("data2", "2")  
        .build();  
    operations.add(op3);  
  
    ContentProviderOperation op4 = ContentProviderOperation.new  
Insert(dataUri)  
        .withValueBackReference("raw_contact_id", 0)  
        .withValue("mimetype", "vnd.android.cursor.item/ema  
il_v2")  
        .withValue("data1", "779878443@qq.com")  
        .withValue("data2", "2")  
        .build();  
    operations.add(op4);  
    //将上述内容添加到手机联系人中~  
    resolver.applyBatch("com.android.contacts", operations);  
    Toast.makeText(getApplicationContext(), "添加成功", Toast.LE  
NGTH_SHORT).show();  
}
```

运行结果:



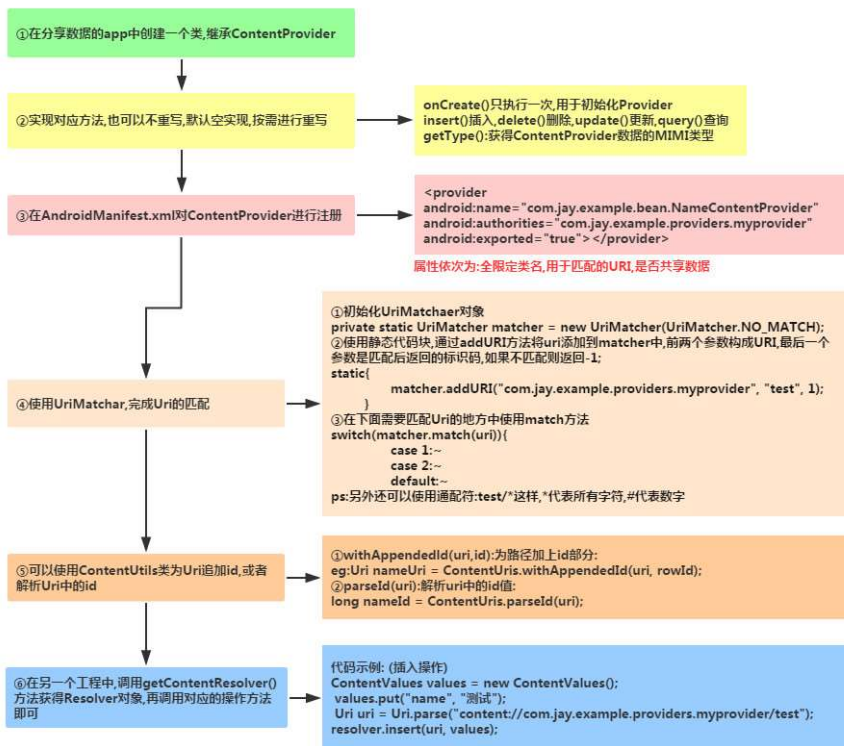
别忘了权限：

```
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_PROFILE"/>
```

3. 自定义ContentProvider

我们很少会自己来定义ContentProvider，因为我们很多时候都不希望自己应用的数据暴露给 其他应用，虽然这样，学习如何ContentProvider还是有必要的，多一种数据传输的方式，是吧~ 这是之前画的一个流程图：

自定义ContentProvider流程解析



接下来我们就来一步步实现：

在开始之前我们先要创建一个数据库创建类(数据库内容后面会讲~)：

DBOpenHelper.java

```
public class DBOpenHelper extends SQLiteOpenHelper {

    final String CREATE_SQL = "CREATE TABLE test(_id INTEGER PRIMARY KEY AUTOINCREMENT,name) ";

    public DBOpenHelper(Context context, String name, CursorFactory factory,
        int version) {
        super(context, name, null, 1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_SQL);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // TODO Auto-generated method stub
    }
}
```


Step 1 : 自定义ContentProvider类，实现onCreate()，getType()，根据需求重写对应的增删改查方法：

NameContentProvider.java

```
public class NameContentProvider extends ContentProvider {

    //初始化一些常量
    private static UriMatcher matcher = new UriMatcher(UriMatc
her.NO_MATCH);
    private DBHelper dbHelper;

    //为了方便直接使用UriMatcher,这里addURI,下面再调用Matcher进行匹
配

    static{
        matcher.addURI("com.jay.example.providers.myprovider",
"test", 1);
    }

    @Override
    public boolean onCreate() {
        dbHelper = new DBHelper(this.getContext(), "tes
t.db", null, 1);
        return true;
    }

    @Override
    public Cursor query(Uri uri, String[] projection, String se
lection,
        String[] selectionArgs, String sortOrder) {
        return null;
    }

    @Override
    public String getType(Uri uri) {
        return null;
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {

        switch(matcher.match(uri))
        {
            //把数据库打开放到里面是想证明uri匹配完成
            case 1:
                SQLiteDatabase db = dbHelper.getReadableDatabas
e();

                long rowId = db.insert("test", null, values);
                if(rowId > 0)
                {
                    //在前面已有的Uri后面追加ID
                    Uri nameUri = ContentUris.withAppendedId(uri, r
```

```

        owId);

        //通知数据已经发生改变
        getContext().getContentResolver().notifyChange(
nameUri, null);

        return nameUri;
    }

}

return null;
}

@Override
public int delete(Uri uri, String selection, String[] selectionArgs) {
    return 0;
}

@Override
public int update(Uri uri, ContentValues values, String selection,
        String[] selectionArgs) {
    return 0;
}
}

```

Step 2 : AndroidManifest.xml中为ContentProvider进行注册：

```

<!--属性依次为：全限定类名,用于匹配的URI,是否共享数据 -->
<provider android:name="com.jay.example.bean.NameContentProvider"
        android:authorities="com.jay.example.providers.myprovider"
        android:exported="true" />

```

好的，作为ContentProvider的部分就完成了！

接下来，创建一个新的项目，我们来实现ContentResolver的部分，我们直接通过按钮点击插入一条数据：

MainActivity.java

```

public class MainActivity extends Activity {

    private Button btninsert;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        btninsert = (Button) findViewById(R.id.btninsert);
    }
}

```

```
//读取contentprovider 数据
final ContentResolver resolver = this.getContentResolver();

btninsert.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        ContentValues values = new ContentValues();
        values.put("name", "测试");
        Uri uri = Uri.parse("content://com.jay.example.providers.myprovider/test");
        resolver.insert(uri, values);
        Toast.makeText(getApplicationContext(), "数据插入成功", Toast.LENGTH_SHORT).show();
    }
});
}
```

如何使用？ 好吧，代码还是蛮简单的，先运行作为ContentProvider的项目，接着再运行ContentResolver的项目， 点击按钮插入一条数据，然后打开file explorer将ContentProvider的db数据库取出，用图形查看工具 查看即可发现插入数据，时间关系，就不演示结果了~

4.通过ContentObserver监听ContentProvider的数据变化

使用ContentObserver监听ContentProvider的数据变化



代码示例:使用ContentObserver监听用户发送短信

```
package com.jay.example.smsobserver;

import android.app.Activity;
import android.database.ContentObserver;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //①为content://sms的数据改变注册监听器
        getContentResolver().registerContentObserver(Uri.parse("content://sms"), true,
            new MyObserver(new Handler()));
    }

    private final class MyObserver extends ContentObserver {
    {

        public MyObserver(Handler handler) {
            super(handler);
        }

        @Override
        public void onChange(boolean selfChange) {
            //查询发送短信的短信(处于正在发送状态的信息放在发送箱)
            Cursor cursor = getContentResolver().query(Uri.parse("content://sms/outbox"),
                null, null, null, null);
            //遍历查询得到的结果集,就可以获得用户正在发送的短信了
            while(cursor.moveToNext())
            {
                StringBuilder sb = new StringBuilder();
                //发送地址
                sb.append("address=").append(cursor.getString(cursor.getColumnIndex("address")));
                //短信标题
                sb.append(";subject").append(cursor.getString(cursor.getColumnIndex("subject")));
                //短信内容
                sb.append(";body").append(cursor.getString(cursor.getColumnIndex("body")));
                //短信标题
                sb.append(";time").append(cursor.getLong(cursor.getColumnIndex("date")));
                System.out.println("用户发送出去的信息:" + sb.toString());
            }
        }
    }
}
```

使用指南：

运行程序后，晾一边，收到短信后，可以在logcat上看到该条信息的内容，可以根据自己的需求 将Activitiy改做Service，而在后台做这种事情~

本节小结：

好的，关于ContentProvider的初探就到这里，本节我们学习了：ContentProvider的概念以及流程，使用系统提供的一些ContentProvider，以及定制自己的ContentProvider，最后还讲解了通过ContentObserver监听ContentProvider的数据变化，ContentProvider的内容就掌握得差不多 了，下一节我们来走走文档看下有什么不知道的~谢谢

^

QR

反馈

坚持一个月, 看美剧不用字幕

每天只要 5分钟, 坚持30天, 效果真的不一样!

立即行动

<div>在线实例</div> <div><ul style="list-style-type: none">· HTML 实例· CSS 实例· JavaScript 实例· Ajax 实例· jQuery 实例· XML 实例· Java 实例</div>	<div>字符集&工具</div> <div><ul style="list-style-type: none">· HTML 字符集设置· HTML ASCII 字符集· HTML ISO-8859-1· HTML 实体符号· HTML 拾色器· JSON 格式化工具</div>	<div>最新更新</div> <div><ul style="list-style-type: none">· Swift 正式开源· PHP 7 正式发布· Shell 编程快速入门· Shell 文件包含· Shell 输入/输出...· Shell printf 命令· Shell 基本运算符</div>	<div>站点信息</div> <div><ul style="list-style-type: none">· 意见反馈· 免责声明· 关于我们· 文章归档</div>
			<div>关注微信</div> <div></div> <div>Copyright © 2013-2015 菜鸟教程 runoob.com All Rights Reserved. 备案号：闽ICP备15012807号-1</div>