ContentObserver

一开始认为，监听短信的接收只可以用广播接收者(BroadcastReceiver)来接收。后来发现了一个新的方法，就是用 ContentObserver 来监听。

在使用 ContentObserver 的过程中，发现了一个问题，问题描述如下：当手机接收到短信的时候， onChange 方法调用了一次，当打开短信App后，onChange方法又调用了一次。这样一来就调用了两次 onChange。

我们一般都不希望它调用两次，比如目前市场上的手环类App一般都具有短信提醒功能，只要手机接收到短信，手环App就会发送某些指令到手环上，手环收到指令进行振动提醒。实现的思路一般是在ContentObserver的 onChange方法中写发送数据到手环的代码。假如出现了上述的问题，出现的结果是，手机接到短信后，手环振动一下，打开短信App，手环又振动一下。

在 android 中经常会出现改变数据库内容后再去使用数据库更新的内容，很多人会重新去 query 一遍，但是这样的问题就是程序会特别占内存，而且有可能会忘关 cursor 而导致程序内存未释放等等。

android 内部提供了一种 ContentObserver 来监听数据库内容的变化。作用是观察（捕捉）特定 Uri 引起的数据库的变化，继而做一些相应的处理。

public final void registerContentObserver(Uri uri, boolean notifyForDescendants, ContentObserver observer)

使用方法：

// 获取 ContentResolver

mContentResolver = getContentResolver();

// 注册ContentObserver ,第一个参数是 Uri，第二个参数如果为 true ，则该Uri的派 // 生 Uri（比如 “content://sms/ inbox”）也可以监听，第三个参数是一个

// ContentObserver。

mContentResover.registerContentObserver(Uri.parse(“content://sms”),true,new SmsContentObserverr(new Handler()));

SmsReceiver 继承 ContentObserver：

class SmsContentObserverr extends ContentObserver{  
  
 private Uri mUri;  
  
 public SmssReciever(Handler handler) {  
  
 super(handler);  
  
 }

// 只要 “content://sms” 里面的数据发生了变化就会调用该方法

public void onChange(boolean selfChange,Uri uri){

super.onChange(selfChange,uri);  
 // selfChange = false, Uri = content://sms/2750 收到短信后调用的  
 // selfChange = false, Uri = content://sms/inbox 打开了短信App后调用的  
 Log.e("onChange","selfChange = "+selfChange+", Uri = "+uri.toString());  
  
 // 第一遍 先执行content://sms/raw 第二遍则 uri.toString :content://sms/inbox  
 if (uri.toString().equals("content://sms/inbox")) {  
 return;  
 }

// 发送数据到手环的代码

……

}

}