4 种获取前台应用的方法（肯定有你不知道的）

阅读 798 收藏 70 2017-01-09
原文链接：www.jianshu.com

我目前已知，并且尝试过的获取当前前台应用的方法有如下几种：

1. Android5.0以前，使用ActivityManager的getRunningTasks()方法，可以得到应用包名和Activity；
2. Android5.0以后，通过使用量统计功能来实现，只能得到应用包名；
3. 通过辅助服务来实现，可以得到包名和Activity；
4. Android5.0以后，可以通过设备辅助应用程序来实现，能得到包名和Activity，不过这种方式必须用户主动触发（长按Home键）

一、ActivityManager的getRunningTasks方法

这是大家可能都知道的方法。在Android5.0以前，只要以下代码就可以获得前台应用：

```
ActivityManager activityManager = (ActivityManager)context.getApplicationContext().getSy
ComponentName runningTopActivity = activityManager.getRunningTasks(1).get(0).topActivity
```

还需要声明权限：

这种方法不止能获取包名，还能获取Activity名。但是在Android 5.0以后，系统就不再对第三方应用提供这种方式来获取前台应用了，虽然调用这个方法还是能够返回结果，但是结果只包含你自己的Activity和Launcher了。

二、通过使用量统计功能获取前台应用

stackoverflow will find a way，getRunningTasks方法失效以后，基本上搜索到的方案都是通过使用量统计功能来获取，也就是下面这种方式：

```
UsageStatsManager mUsageStatsManager = (UsageStatsManager)context.getApplicationContext()
long time = System.currentTimeMillis();
List<UsageStats> stats ;
if (isFirst){
    stats = mUsageStatsManager.queryUsageStats(UsageStatsManager.INTERVAL_DAILY, time - 
}else {
    stats = mUsageStatsManager.queryUsageStats(UsageStatsManager.INTERVAL_DAILY, time - 
}
// Sort the stats by the last time used
if(stats != null) {
    TreeMap<Long,UsageStats> mySortedMap = new TreeMap<Long,UsageStats>();
    start=System.currentTimeMillis();
    for (UsageStats usageStats : stats) {
        mySortedMap.put(usageStats.getLastTimeUsed(),usageStats);
    }
    LogUtil.e(TAG,"isFirst="+isFirst+",mySortedMap cost:"+ (System.currentTimeMillis()-st
    if(mySortedMap != null && !mySortedMap.isEmpty()) {

        topPackageName =  mySortedMap.get(mySortedMap.lastKey()).getPackageName();
```

```
    }
}
```

代码的功能是通过UsageStatsManager 来获取用户使用的程序的列表，然后按照最近使用时间排序，就得到了当前的前台应用，这种方式只能拿到包名，无法精确了Activity了。
使用这种方发之前，首先要引导用户开启使用量功能：

```
Intent intent = new Intent(Settings.ACTION_USAGE_ACCESS_SETTINGS);
startActivity(intent);
```

还要申明权限：

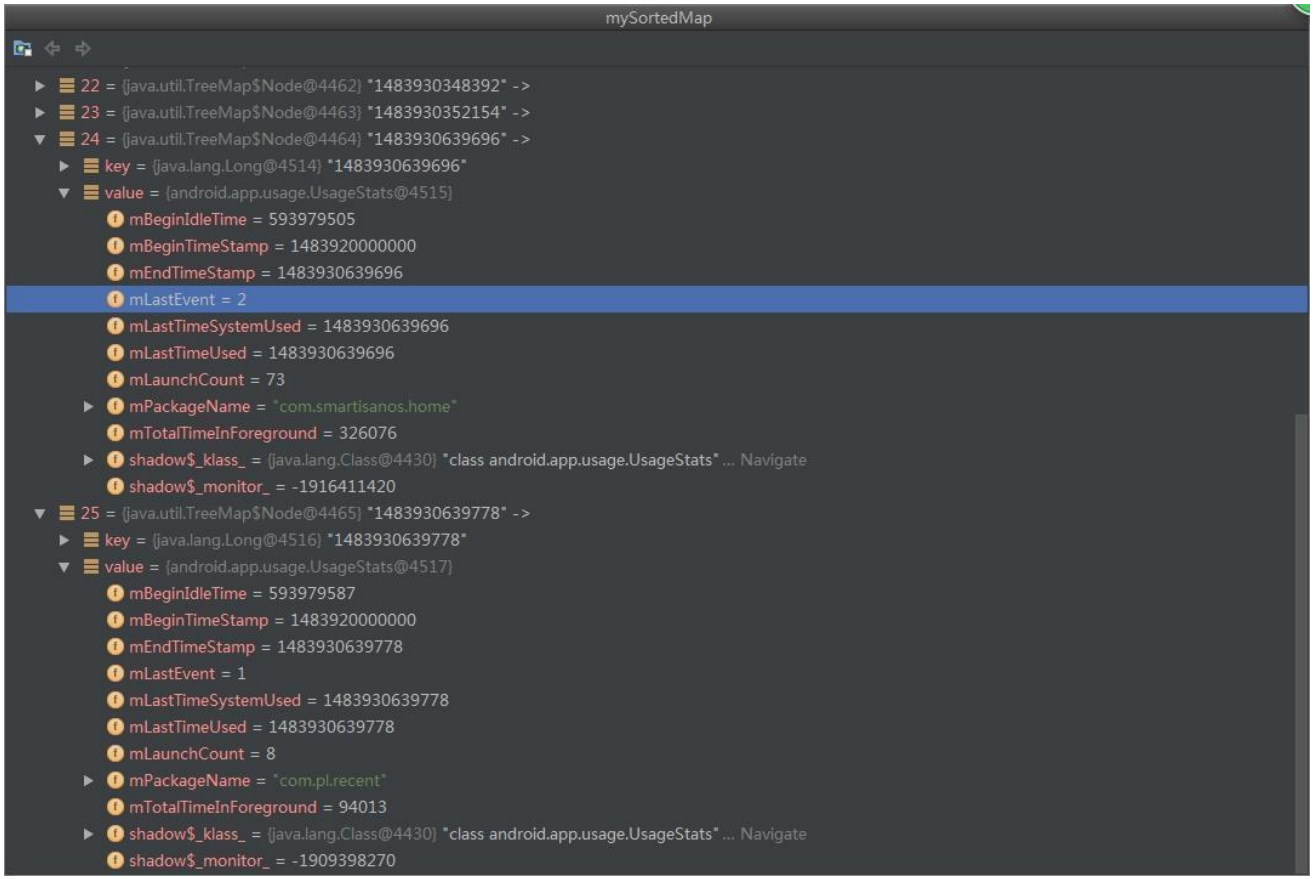
```
<uses-permission android:name="android.permission.PACKAGE_USAGE_STATS" />
```

要注意的是，只是这样并不够！因为在一些手机上，应用发起通知栏消息的时候，或者是下拉通知栏，也会被记录到使用量中，就会导致按最近时间排序出现混乱。而且收起通知栏以后，这种混乱并不会被修正，而是必须重新开启一个应用才行。
比如下图:

打开通知栏导致前台应用判断错误.gif

栏，悬浮窗就又出现了。

那怎么办呢？万能的StackOverflow也不能了，我只好自己研究，通过仔细对比，我发现UsageStats有一个hide的字段似乎有些蹊跷——mLastEvent，如下图。



新建位图图像_看图王.jpg

我发现对于打开的在前台的应用mLastEvent=1，而对于通知栏收到消息的应用，则mLastEvent！=1，有时为2，有时为0。看看UsageStats的源码，没发现有用信息，但是

义：

```
/**
 * An event type denoting that a component moved to the foreground.
 */
public static final int MOVE_TO_FOREGROUND = 1;

/**
 * An event type denoting that a component moved to the background.
 */
public static final int MOVE_TO_BACKGROUND = 2;
```

此时我们不妨乐观的假设，**这两个值分别就是UsageStats.mLastEvent中的1和2**，从名字上就能看得出含义，正是我们需要的值。
带着假设去源码中寻找答案，会发现源码中充斥着类似下面的代码：

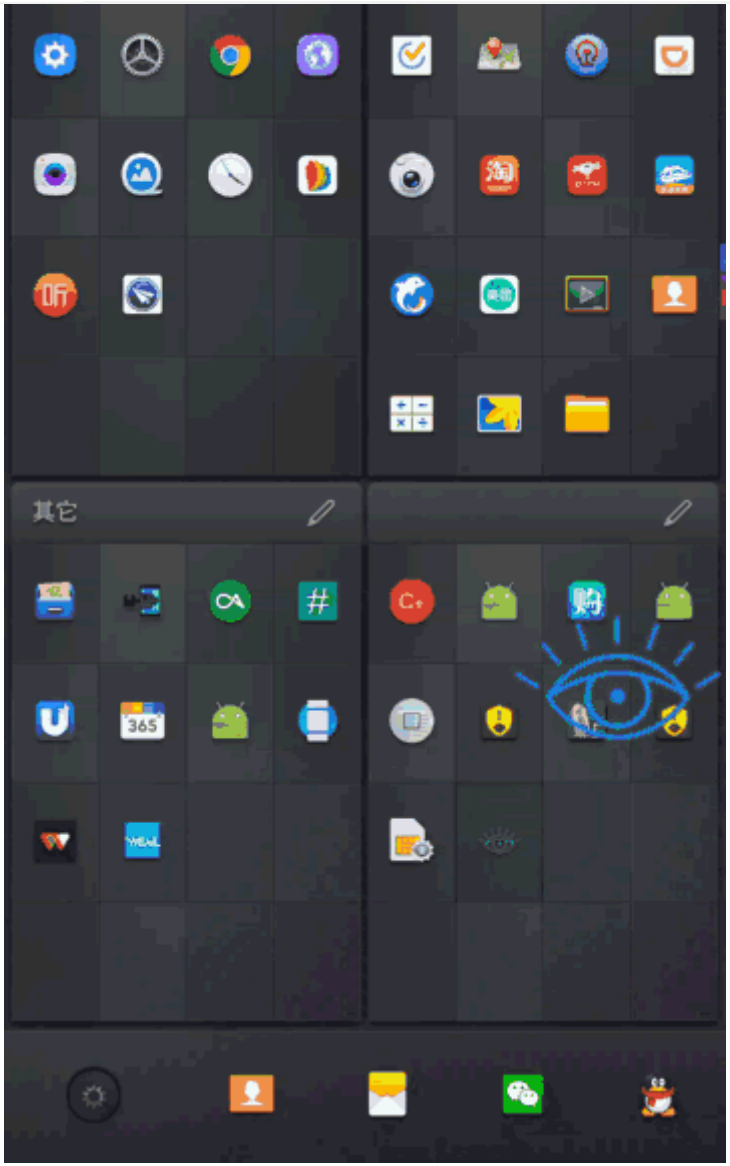
```
//下面代码来自com.android.server.usage.IntervalStats
private boolean isStatefulEvent(int eventType) {
   switch (eventType) {
       case UsageEvents.Event.MOVE_TO_FOREGROUND:
       case UsageEvents.Event.MOVE_TO_BACKGROUND:
       case UsageEvents.Event.END_OF_DAY:
       case UsageEvents.Event.CONTINUE_PREVIOUS_DAY:
           return true;
   }
   return false;
}
```

```
            // TODO(adamlesinski): Ensure that we recover from incorrect event sequences
             // like double MOVE_TO_BACKGROUND, etc.
             if (eventType == UsageEvents.Event.MOVE_TO_BACKGROUND ||
                     eventType == UsageEvents.Event.END_OF_DAY) {
                 if (usageStats.mLastEvent == UsageEvents.Event.MOVE_TO_FOREGROUND ||
                         usageStats.mLastEvent == UsageEvents.Event.CONTINUE_PREVIOUS_DAY) {
                     usageStats.mTotalTimeInForeground += timeStamp - usageStats.mLastTimeUsed;
                 }
             }

             if (isStatefulEvent(eventType)) {
                 usageStats.mLastEvent = eventType;
             }

             if (eventType != UsageEvents.Event.SYSTEM_INTERACTION) {
                 usageStats.mLastTimeUsed = timeStamp;
             }
             usageStats.mLastTimeSystemUsed = timeStamp;
             usageStats.mEndTimeStamp = timeStamp;

             if (eventType == UsageEvents.Event.MOVE_TO_FOREGROUND) {
                 usageStats.mLaunchCount += 1;
             }

             endTime = timeStamp;
         }
```

mLastEvent ==1的元素即可。代码和效果图如下：

```
//改进版本的通过使用量统计功能获取前台应用
UsageStatsManager mUsageStatsManager = (UsageStatsManager)context.getApplicationContext()
long time = System.currentTimeMillis();
List<UsageStats> stats ;
if (isFirst){
    stats = mUsageStatsManager.queryUsageStats(UsageStatsManager.INTERVAL_DAILY, time - T
}else {
    stats = mUsageStatsManager.queryUsageStats(UsageStatsManager.INTERVAL_DAILY, time - T
}
// Sort the stats by the last time used
if(stats != null) {
    TreeMap<Long,UsageStats> mySortedMap = new TreeMap<Long,UsageStats>();
    start=System.currentTimeMillis();
    for (UsageStats usageStats : stats) {
        mySortedMap.put(usageStats.getLastTimeUsed(),usageStats);
    }
    LogUtil.e(TAG,"isFirst="+isFirst+",mySortedMap cost:"+ (System.currentTimeMillis()-st
    if(mySortedMap != null && !mySortedMap.isEmpty()) {

        NavigableSet<Long> keySet=mySortedMap.navigableKeySet();
        Iterator iterator=keySet.descendingIterator();
        while(iterator.hasNext()){
            UsageStats usageStats = mySortedMap.get(iterator.next());
            if (mLastEventField==null) {
                try {
                    mLastEventField = UsageStats.class.getField("mLastEvent");
```

```
                }
                if (mLastEventField!=null) {
                    int lastEvent = 0;
                    try {
                        lastEvent = mLastEventField.getInt(usageStats);
                    } catch (IllegalAccessException e) {
                        break;
                    }
                    if (lastEvent==1){
                        topPackageName=usageStats.getPackageName();
                        break;
                    }
                }else {
                    break;
                }
            }
            if (topPackageName==null){
                topPackageName =  mySortedMap.get(mySortedMap.lastKey()).getPackageName();
            }
            runningTopActivity=new ComponentName(topPackageName,"");
            if (LogUtil.isDebug())LogUtil.d(TAG,topPackageName);
        }
    }
```



改进后不会受通知栏影响了

辅助服务(AccessibilityService)有很多神奇的妙用，比如辅助点击，比如页面抓取，还有就是获取前台应用。

这里简单介绍一下如何使用辅助服务，首先要在AndroidManifest.xml中声明：

```
<service
    android:name=".service.AccessibilityMonitorService"
    android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE"
    >
    <intent-filter>
        <action android:name="android.accessibilityservice.AccessibilityService" />
    </intent-filter>

    <meta-data
        android:name="android.accessibilityservice"
        android:resource="@xml/accessibility" />
</service>
```

然后在res/xml/文件夹下新建文件accessibility.xml，内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<accessibility-service
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:accessibilityEventTypes="typeViewClicked|typeViewLongClicked|typeWindowState(
    android:accessibilityFeedbackType="feedbackGeneric"
    android:accessibilityFlags="flagRetrieveInteractiveWindows"
    android:canRetrieveWindowContent="true"
    android:canRequestFilterKeyEvents ="true"
    android:notificationTimeout="10"
```

```
/>
```

至于其中每一项的内容，还是去看API文档吧，我这里一一解释的话，文章就太长了。关键是typeWindowStateChanged。
新建AccessibilityMonitorService，主要内容如下：

```
public class AccessibilityMonitorService extends AccessibilityService {
    private CharSequence mWindowClassName;
    private String mCurrentPackage;
    @Override
    public void onAccessibilityEvent(AccessibilityEvent event) {
        int type=event.getEventType();
        switch (type){
            case AccessibilityEvent.TYPE_WINDOW_STATE_CHANGED:
                mWindowClassName = event.getClassName();
                mCurrentPackage = event.getPackageName()==null?"":event.getPackageName()
                break;
            case TYPE_VIEW_CLICKED:
            case TYPE_VIEW_LONG_CLICKED:
                break;
        }
    }
}
```

清理后台，就会被关闭。。。

想要了解辅助服务如何监控点击和抓取页面的，可以参考Bigbang项目的BigBangMonitorService。

四、通过设备辅助应用程序获取前台应用（比较鸡肋）

所谓设备辅助应用程序，是在一些接近原生的系统上，长按Home键就会触发的应用，默认是会触发Google搜索。设备辅助应用程序有点像是需要主动触发的辅助服务，因为应用中是无法主动去触发其功能的，所以说比较鸡肋，鉴于篇幅，这里就不详细介绍了。
感兴趣的朋友可以看Demo源码中的简单应用，也可以看看Bigbang项目的BBVoiceInteractionService、BBVoiceInteractionSession和BBVoiceInteractionSessionService

Demo源码

Github

Android