Android软键盘的全面解析，让你不再怕控件被遮盖

注明

本文已授权CSDN官方**微信**公众号《mobilehub》原创首发，转载请务必注明出处。

修正

1.修正了flagNoFullscreen和flagNoExtractUi的说明，以及区别。

背景

1.**Android**软键盘这块从我入职到现在，是一个一直纠缠我的问题。

2.从布局挤压，到EditText显示不全，在到弹出时卡顿，在**android**软键盘面前我无数次跌倒。

3.因为网上大多数的知识点比较分散而且很杂，所以本篇做一个整合篇。

4.Android软键盘这块知识点比较密集，了解过一次之后，差不多什么情况都可以找到原因了。

5.感谢Android软键盘的问题，从我入职陪伴我到现在，让我一个一个不停的解决。

前言

(1)InputMethodService的源码解析，从源码解读中告诉你为什么软键盘弹出的是一个**Dialog**

(2)Android软键盘显示时，设置windowSoftInputMode的作用

(3)EditText设置imeOptions属性对软键盘的影响

(4)软键盘上面的按键监听

(5)横屏状态下，不希望软键盘显示全屏怎么处理

(6)控制软键盘的弹出和关闭的方法

(7)EditText在软键盘弹出的时候显示不全，怎么获取软键盘弹出和关闭的监听

(8)软键盘弹出的时候，造成页面卡顿，这时候如何发现问题并解决问题

(9)Android键盘面板冲突，布局闪动的解决方法

Android软键盘的显示原理

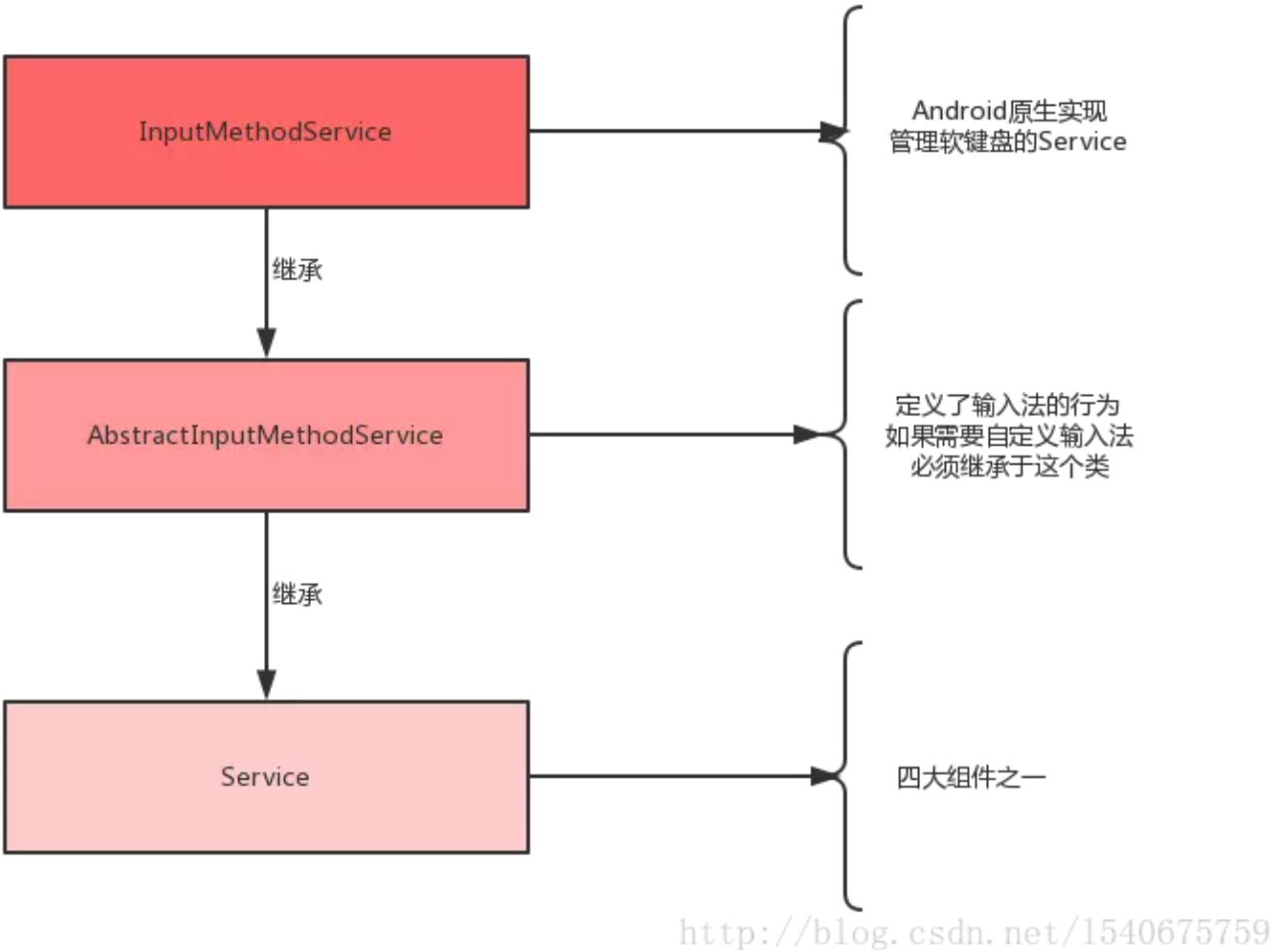
软键盘其实是一个Dialog

InputMethodService为我们的输入法创建了一个Dialog，并且对某些参数进行了设置，使之能够在底部或者全屏显示。当我们点击输入框时，系统会对当前的主窗口进行调整，以便留出相应的空间来显示该Dialog在底部，或者全屏。

其实这段话我们经常在各种软键盘博客所看到，但是大家并不知道Android是怎么为我们创建的这个Dialog，所以我先带大家来看下软键盘生成这块的源码，了解软键盘的生成流程。

InputMethodService的源码解析

我们先来看一下InputMethodService的继承关系：



因为InputMethodService属于服务，接下来我们先看一下服务的入口onCreate()方法：

```
@Override
public void onCreate() {
    //设置主题与xml里面设置theme是一样的道理
    mTheme = Resources.selectSystemTheme(mTheme,
            getApplicationInfo().targetSdkVersion,
            android.R.style.Theme_InputMethod,
            android.R.style.Theme_Holo_InputMethod,
            android.R.style.Theme_DeviceDefault_InputMethod,
            android.R.style.Theme_DeviceDefault_InputMethod);
    super.setTheme(mTheme);
    //创建InputMethodMananger
    mImm = (InputMethodManager)getSystemService(INPUT_METHOD_SERVICE);
    mSettingsObserver = SettingsObserver.createAndRegister(this);
    mShouldClearInsetOfPreviousIme = (mImm.getInputMethodWindowVisibleHeight() > 0);
```

```
         * 这里注意一下，首先这里的命名属于Window，然后我们发现了Gravity.BOTTOM，就更加确定了这个就是
         * 软键盘所创建的Dialog对象
         */
        mWindow = new SoftInputWindow(this, "InputMethod", mTheme, null, null, mDispatcherState,
                WindowManager.LayoutParams.TYPE_INPUT_METHOD, Gravity.BOTTOM, false);
        if (mHardwareAccelerated) {
            mWindow.getWindow().addFlags(WindowManager.LayoutParams.FLAG_HARDWARE_ACCELERATED);
        }
        initViews();
        mWindow.getWindow().setLayout(MATCH_PARENT, WRAP_CONTENT);
    }
```

通过上面的分析，我们怀疑这里的SoftInputWindow是软键盘弹出创建的Dialog对象，下面我们看下SoftInputWindow的源码。

```
public class SoftInputWindow extends Dialog{
    ....
}
```

看到这里大家就能明白了，为什么说软键盘就是一个Dialog。而且这里通过设置Gravity.BOTTOM来控制当前Dialog在Window中的位置。

软键盘的显示调整(windowSoftInputMode)

在Android中，可以通过给Activity设置windowSoftInputMode这个属性来控制软键盘与Activity的主窗口的交互方式。

Activity 的主窗口与包含屏幕软键盘的窗口的交互方式，该属性的设置影响两个方面：

1. 当Activity成为用户注意的焦点时软键盘的状态 - 隐藏还是可见。
2. 对Activity主窗口所做的调整 - 意思是是否将其尺寸调小为软键盘腾出空间，或者当窗口部分被软键盘遮挡时是否平移其内容以使当前焦点可见。

该设置必须是下面所列的值之一，或者是一个“state...”值加上一个“adjust...”值的组合，在任一组中设置多个值（例如，多个“state...”值）都会产生未定义结果。各值之间使用垂直条 (|) 分隔

stateUnspecified-不指定软键盘的状态（隐藏还是可见） 将由系统选择合适的状态，或依赖主题中的设置，这是对软键盘行为的默认设置

stateUnchanged-保留状态 当 Activity 转至前台时保留软键盘最后所处的任何状态，无论是可见还是隐藏

stateHidden-隐藏软键盘 当用户确实是向前导航到 Activity，而不是因离开另一Activity 而返回时隐藏软键盘

stateAlwaysHidden-始终隐藏软键盘 当 Activity 的主窗口有输入焦点时始终隐藏软键盘

stateVisible-显示软键盘 在正常的适宜情况下（当用户向前导航到 Activity 的主窗口时）显示软键盘

stateAlwaysVisible-显示软键盘 当用户确实是向前导航到 Activity，而不是因离开另一Activity 而返回时.

(2)在软键盘弹出时，是否需要Activity对此进行调整

adjustUnspecified 主窗口的默认行为，不指定 Activity 的主窗口是否调整尺寸以为软键盘腾出空间，或者窗口内容是否进行平移以在屏幕上显露当前焦点。 系统会根据窗口的内容是否存在任何可滚动其内容的布局视图来自动选择其中一种模式。 如果存在这样的视图，窗口将进行尺寸调整，前提是可通过滚动在较小区域内看到窗口的所有内容。

adjustResize 始终调整 Activity 主窗口的尺寸来为屏幕上的软键盘腾出空间。

adjustPan 不调整 Activity 主窗口的尺寸来为软键盘腾出空间， 而是自动平移窗口的内容，使当前焦点永远不被键盘遮盖，让用户始终都能看到其输入的内容。 这通常不如尺寸调整可取，因为用户可能需要关闭软键盘以到达被遮盖的窗口部分或与这些部分进行交互。

adjustNoting 软键盘弹出时，主窗口Activity不会做出任何响应。

windowSoftInputMode 应用场景

下面将通过例子来介绍adjustNoting、adjustUnspecified、adjustResize、adjustPan在软键盘弹出的区别：

adjustPan ： 当软键盘弹出时，会将主窗口的平移（translateY），来适应软键盘的显示。

adjustResize ： 当软键盘弹出时，会让布局重新绘制，这种一般适应于带有滑动性质的控制，让其向下滚动，然后适应软键盘的显示。

adjustNoting： 软键盘弹出时，主窗口不会做出任何反应。

非滚动布局xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

            <EditText
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="请输入您要输入的内容1" />

            <EditText
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="请输入您要输入的内容2" />

                    ..........<中间包含无数的EditText>

            <EditText
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="请输入您要输入的内容12" />

            <EditText
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="请输入您要输入的内容13" />

</LinearLayout>
```

（1）设置windowSoftInputMode为adjustNoting

从上图发现，当点击EditText12时，弹出软键盘将主窗口下半部分给遮盖，并且主窗口没有做出任何反应。

（2）设置windowSoftInputMode为adjustPan

当设置其属性为adjustPan时，当软键盘弹出时，主窗口布局会上移至直到显示EditText12。

（3）设置windowSoftInputMode为adjustUnspecified

当设置其属性为默认属性adjustUnspecified时，发现当点击EditText12时，主窗口上移来保持EditText12在软键盘之上，这时adjustUnspecified的表现形式与adjustPan相同，所以在无滑动的控件上，默认的指定形式为adjustPan。

（4）设置windowSoftInputMode为adjustResize

设置其属性为adjustResize时，发现软键盘弹出的状态与adjustNoting表现一致，当设置adjustResize时，布局会为了软键盘弹出而重新绘制给软键盘留出空间，而由于控件无法滑动，所以表现的形式与adjustNoting一致。

滚动布局xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
```

```
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <EditText
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="请输入您要输入的内容1" />

            <EditText
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="请输入您要输入的内容2" />

              ..........<中间有很多了EditText>

            <EditText
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="请输入您要输入的内容12" />

            <EditText
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="请输入您要输入的内容13" />

        </LinearLayout>

    </ScrollView>

</LinearLayout>
```

还是同样的操作，点击最下面的EditText13

（1）设置windowSoftInputMode为adjustNoting

我们可以看出，当点击EditText12时，弹出软键盘将主窗口下半部分给遮盖，并且主窗口没有做出任何反应，和不加ScrollView是一样的情况。

（2）设置windowSoftInputMode为adjustResize

我们可以发现，当设置其属性为adjustResize时，当软键盘弹出时，ScrollView会重新绘制，然后滚动EditText13位置，使其显示在软键盘之上。

（3）设置windowSoftInputMode为adjustUnspecified

当设置其属性为默认属性adjustUnspecified时，可以发现在添加了ScrollView控件时，布局的窗口并不会上移（这个观察Toolbar就可以发现），而通过重绘ScrollView，让其滚动到最低端，并且给软键盘流出控件，而这个表现即和adjustResize完全一致。

（4）设置windowSoftInputMode为adjustPan

可以发现，在滑动空间下，设置属性adjustPan时，依旧会将主窗口上移，来使EditText13显示在软键盘之上，可以通过观察Toolbar得知。

windowSoftInputMode总结

通过上面的例子，我们可以完全理解adjust系列的各个参数的作用。而软键盘的显示和隐藏这里面需要并不多，而且内容并不算复杂，大家回去自己尝试下就可以。

EditText设置imeOptions属性对软键盘的影响

在日常开发中，如果需要将软键盘的Enter键更改为其他键，可以设置其android:imeOptions 属性，这个属性可以控制软键盘的Enter键，以及横屏情况下的软键盘显示状态。

该设置必须是下面所列的值之一，或者是一个“action...”值加上一个“flag...”值的组合，在action...组中设置多个值（例如，多个“action...”值）都会产生未定义结果，而flag....可以设置多个。各值之间使用垂直条 (|) 分隔

(1)控制软键盘上的Enter键

android:imeOptions=”normal”

当android:singleLine=”true”
输入框后面还有输入控件的时候会显示next，没有时会显示done(完成)
当android:singleLine=”false”
输入框会进行换行操作

android:imeOptions=”actionUnspecified”

该属性为默认属性，一般情况下为**“normal”**的使用情形。

android:imeOptions=”actionNone”

显示回车键，当singleLine为true的时候，会跳到下个可输出的控件，否则软键盘消失，输入完毕。

android:imeOptions=”actionGo”

显示为Go(前往)按钮，需要配合android:singleLine使用，否则为回车键起换行作用，并且需要自己写事件。

android:imeOptions=”actionSearch”

显示搜索（Search）按钮，需要配合android:singleLine使用，否则为回车键起换行作用，并且需要自己写事件。

android:imeOptions=”actionSend”

显示send(发送)按钮，需要配合android:singleLine使用，否则为回车键起换行作用，并且需要自己写事件。

android:imeOptions=”actionNext”

显示next(下一步)按钮，作用是跳到下一个可输入的控件，需要配合android:singleLine使用，否则为回车键起换行作用。

android:imeOptions=”actionDone”

显示done(完成)按钮，作用编辑完成，让软键盘消失.需要配合android:singleLine使用，否则为回车键起换行作用。

android:imeOptions=”actionPrevious”

显示上一步按钮，如果前面有输入控件，点击后会回到前一个控件获取焦点，.需要配合android:singleLine使用，否则为回车键起换行作用。

可能各个输入法的显示图标不一样，但是效果是一样的，这里用的是搜狗输入法。

(2)横屏下控制软键盘

android:imeOptions=”flagNoFullscreen”

在横屏下,当设置这个标志时，软键盘在弹出的时候，永远不会变成全屏状态，但是这个属性在API中说并不一定所有输入法都支持这个属性。

android:imeOptions=”flagNoExtractUi”

这个属性也有意思，它的表现形式和**flagNoFullscreen**比较像。

因为在横屏下，这两个属性单独设置都让软键盘半屏显示，但是这两个属性还是有所不同的。

这里可以参阅下API，flagNoExtractUI显示的半屏软键盘本身软键盘显示还是全屏的，但是将之前的全屏输入框给隐藏掉，所以给你显示半屏的效果。而且在单独使用的时候，可能你会发现软键盘是先从全屏然后过渡到半屏的。

所以要去掉全屏到半屏过渡效果，在横屏状态下，需要和flagNoFullscreen一块使用，来达到更好的体验。

android:imeOptions=”flagNavigatePrevious”

横屏下设置输入法全屏，设置输入框上的按钮为(previous)上一个的作用。

android:imeOptions=”flagNavigateNext”

横屏下设置输入法全屏，设置输入框上的按钮为(Next)下一个作用。

android:imeOptions=”flagNoAccessoryAction”

横屏下设置输入法全屏，并且使其输入框上的按钮隐藏。

android:imeOptions=”flagNoEnterAction”

横屏下设置输入法全屏，输入框内的按钮为完成(Done)状态.编辑完毕，点完成，软键盘消失。

总结一下:

这里大部分的属性，已经介绍完毕，如果英语好的同学，可以去看下官方文档，可以更好的理解，并且本文以搜狗输入法为实践，可能其他的输入法与其显示的不同，但是功能应该都是一样的。

而如果要在横屏状态不希望软键盘全屏显示最好是将flagNoFullscreen和flagNoExtractUi结合使用，这样体验上会更好

android:imeOptions=” flagNoFullscreen|flagNoExtractUi”

Android官网的Api

Android软键盘上的按键监听

```
mMainEt = (EditText) findViewById(R.id.main_et);
mMainEt.setOnEditorActionListener(new TextView.OnEditorActionListener() {
    @Override
    public boolean onEditorAction(TextView v,  int actionId,  KeyEvent event) {
        switch (actionId) {
            //点击GO键
            case EditorInfo.IME_ACTION_GO:
                return true;
            //点击Done
            case EditorInfo.IME_ACTION_DONE:
                return true;
            //点击Next
            case EditorInfo.IME_ACTION_NEXT:
                return true;
            //点击Previous
            case EditorInfo.IME_ACTION_PREVIOUS:
                return true;
            //点击None
            case EditorInfo.IME_ACTION_NONE:
                return true;
            //点击Send
            case EditorInfo.IME_ACTION_SEND:
                return true;
        }
        return false;
    }
});
```

上面的方式，只是展示了如何监听各个按键的方法，如果需要消费事件，则需要return true。

实际案例1：横屏下软键盘不全屏显示

其实已经在第二方面介绍了，只要设置EditText 的android:imeOptions属性为 flagNoFullScreen 和 flagNoExtraUI即可。

实际案例2：控制软键盘弹出和关闭

一般来说，在实际项目中，都会使用工具类来控制软键盘的显示或者关闭。

```
InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
if (imm != null) {
    view.requestFocus();
    imm.showSoftInput(view,  0);
}
```

(2)关闭软键盘

```
InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT_METHOD_SERVICE);
if (imm != null) {
    imm.hideSoftInputFromWindow(view.getWindowToken(),  0);
}
```

而如果对于这一块有什么不明白的，可以参考这篇博客。

Android手动显示和隐藏软键盘方法总结

实际案例3：EditText显示不全，并且需要监听软键盘弹出和关闭

现在有一个常见的需求，EditText被布局包裹，然后需要将原生的EditText背景给替换掉或者直接设置为null（或者其他），然后和布局上下存在间距，然后整体与底部对齐。

这时候弹出软键盘，请看图：

从图中可以发现，当原生的EditText背景被替换之后，软键盘会遮盖掉自定义区域，并且直接显示在EditText之下，正常情况下，我们是希望软键盘显示在整个外层的Layout之下的。

当时对于这个问题，我没有头绪好一阵子，现在来看，那时候挺年轻的。

而这个问题，就属于软键盘遮挡布局的问题：

> 引用块内容

(1)软键盘遮盖焦点：

当软键盘弹出的时候，将EditText等输入类的控件的焦点遮盖时，这时候可以设置adjustPan或者adjustResize可以很好的解决。

(2)软键盘遮盖没有遮盖焦点，但是遮盖了需要显示的控件：

这时候设置通过设置属性adjustPan或者adjustResize还不足以解决问题，因为软键盘并没有遮盖了EditText的焦点，所以单独设置这两个属性是对软键盘或者界面是无法产生改变的。

这时我们必须从另外一个角度来考虑这个问题，就是来监听软键盘的弹出和关闭来操作布局，来解决这个问题。

这时候普遍会有以下的几种解决方案：

(1)设置adjustResize属性，当软键盘弹出的时候会重绘布局，然后设置根布局的OnLayoutChangeListener的监听，来监听布局的变化。

```
mScrollView.addOnLayoutChangeListener(new View.OnLayoutChangeListener() {
@Override
public void onLayoutChange(View v,  int left,  int top,  int right,  int bottom,  int oldLeft,  int old
            Log.d("new change ",  "left : " + left + "top : " + top + "right : " + right + "bottom
            Log.d("old change ",  "oldLeft : " + oldLeft + "oldTop : " + oldTop + "oldRight : " + o
        }
    });

//当软键盘弹出
07-09 21:16:22.911 23653-23653/com.xiucai.softdemo D/new change: left : 0top : 0right : 1080bottom : 81
07-09 21:16:22.911 23653-23653/com.xiucai.softdemo D/old change: oldLeft : 0oldTop : 0oldRight : 1080ol

//当软键盘关闭
07-09 21:16:44.457 23653-23653/com.xiucai.softdemo D/new change: left : 0top : 0right : 1080bottom : 16
07-09 21:16:44.457 23653-23653/com.xiucai.softdemo D/old change: oldLeft : 0oldTop : 0oldRight : 1080ol
```

我们可以通过上述打印结果发现，根布局的bottom发生变化了，从1692变化到817。我们可以通过这种方式来监听软键盘的弹出和收回，然后实现平移根布局或者其他的操作来保证软键盘不会遮盖你所需要的显示的控件。

(2)通过getViewTreeObserver().addOnGlobalLayoutListener()监听窗体的可见区域，来判断软键盘是否弹出。

```
/**
```

```
private void controlKeyboardLayout(final View root,  final View scrollToView) {
    root.getViewTreeObserver().addOnGlobalLayoutListener( new OnGlobalLayoutListener() {
        @Override
        public void onGlobalLayout() {
            Rect rect = new Rect();
            //获取root在窗体的可视区域
            root.getWindowVisibleDisplayFrame(rect);
            //获取root在窗体的不可视区域高度(被其他View遮挡的区域高度)
            int rootInvisibleHeight = root.getRootView().getHeight() - rect.bottom;
            //若不可视区域高度大于100，则键盘显示
            if (rootInvisibleHeight > 100) {
                int[] location = new int[2];
                //获取scrollToView在窗体的坐标
                scrollToView.getLocationInWindow(location);
                //计算root滚动高度，使scrollToView在可见区域
                int srollHeight = (location[1] + scrollToView.getHeight()) - rect.bottom;
                root.scrollTo(0,  srollHeight);
            } else {
                //键盘隐藏
                root.scrollTo(0,  0);
            }
        }
    });
}
```

上面代码是借鉴android 解决输入法键盘遮盖布局问题 这篇文章而来，目的让大家了解这种方式是怎么判断软键盘的弹出和隐藏。

里面需要平移的View，不一定是ScrollView，其他View也可，一般来说作为XML的根布局即可。

平常一般来说，我会使用这个工具类

```
package com.xiucai.common.manager;

import android.graphics.Rect;
import android.util.Log;
import android.view.View;
import android.view.ViewTreeObserver;

import java.util.LinkedList;
import java.util.List;
```

```
 */

public class SoftKeyBroadManager  implements ViewTreeObserver.OnGlobalLayoutListener{

    public interface SoftKeyboardStateListener {
        void onSoftKeyboardOpened(int keyboardHeightInPx);

        void onSoftKeyboardClosed();
    }

    private final List<SoftKeyboardStateListener> listeners = new LinkedList<SoftKeyboardStateListener>
    private final View activityRootView;
    private int lastSoftKeyboardHeightInPx;
    private boolean isSoftKeyboardOpened;

    public SoftKeyBroadManager(View activityRootView) {
        this(activityRootView,  false);
    }

    public SoftKeyBroadManager(View activityRootView,  boolean isSoftKeyboardOpened) {
        this.activityRootView = activityRootView;
        this.isSoftKeyboardOpened = isSoftKeyboardOpened;
        activityRootView.getViewTreeObserver().addOnGlobalLayoutListener(this);
    }

    @Override
    public void onGlobalLayout() {
        final Rect r = new Rect();
        //r will be populated with the coordinates of your view that area still visible.
        activityRootView.getWindowVisibleDisplayFrame(r);

        final int heightDiff = activityRootView.getRootView().getHeight() - (r.bottom - r.top);
        Log.d("SoftKeyboardStateHelper",  "heightDiff:" + heightDiff);
        if (!isSoftKeyboardOpened && heightDiff > 500) { // if more than 100 pixels,  its probably a ke
            isSoftKeyboardOpened = true;
            notifyOnSoftKeyboardOpened(heightDiff);
            //if (isSoftKeyboardOpened && heightDiff < 100)
        } else if (isSoftKeyboardOpened && heightDiff < 500) {
            isSoftKeyboardOpened = false;
            notifyOnSoftKeyboardClosed();
        }
    }

    public void setIsSoftKeyboardOpened(boolean isSoftKeyboardOpened) {
```

```
    public boolean isSoftKeyboardOpened() {
        return isSoftKeyboardOpened;
    }

    /**
     * Default value is zero (0)
     *
     * @return last saved keyboard height in px
     */
    public int getLastSoftKeyboardHeightInPx() {
        return lastSoftKeyboardHeightInPx;
    }

    public void addSoftKeyboardStateListener(SoftKeyboardStateListener listener) {
        listeners.add(listener);
    }

    public void removeSoftKeyboardStateListener(SoftKeyboardStateListener listener) {
        listeners.remove(listener);
    }

    private void notifyOnSoftKeyboardOpened(int keyboardHeightInPx) {
        this.lastSoftKeyboardHeightInPx = keyboardHeightInPx;

        for (SoftKeyboardStateListener listener : listeners) {
            if (listener != null) {
                listener.onSoftKeyboardOpened(keyboardHeightInPx);
            }
        }
    }

    private void notifyOnSoftKeyboardClosed() {
        for (SoftKeyboardStateListener listener : listeners) {
            if (listener != null) {
                listener.onSoftKeyboardClosed();
            }
        }
    }
}
```

不管是全屏或者其他情况，使用这个工具类来监听软键盘就可以了，具体的用法：

```
SoftKeyBroadManager mManager =new SoftKeyBroadManager ("根布局");
//添加软键盘的监听，然后和上面一样的操作即可.
mSoftKeyBroadManager.addSoftKeyboardStateListener();
//注意销毁时，得移除监听
mSoftKeyBroadManager.removeSoftKeyboardStateListener();
```

最后如果感觉上面的方案不可行，Github也有一个现成方案，但是博主本人没试过，原理都是一样的。大家可以自行取舍。

Github软键盘监听的工具类

知乎上讨论软键盘的文章

实际案例4： 软键盘弹出来卡顿，找不到原因?

正常来说博文实差不多到这里就应该结束了，但是博主在实际开发中，也会遇到一些诡异的现象，例如软键盘弹出卡顿，但是这种情况下，根本无法定位到卡顿原因。

博主遇到这个问题时，怀疑了设置属性错误，怀疑了线程XX没关，怀疑了布局太过于复杂，总之该想的博主都想了，但是无论怎么试都是徒劳的。

因为博主犯了一个大错

在没找到原因之前，胡乱猜测，可能是这块?是不是那个的问题，而不确定问题的来源这个问题我感觉大家都会遇到，不从事情的本质上下手，这样会多花很多时间用在无用的地方，使自己的开发效率很低。

推荐一款检测卡顿的神器BlockCanary

还原下我当时遇到的问题：

当时我在做一个直播间的功能，直播间从主页跳转进入的，给大家截一张图，大家就懂了。

当时做直播间其他功能的时，发现直播间软键盘在弹出和关闭的时候卡顿。

当时怀疑：

(1)什么动画没停，什么线程没关。

(2)软键盘 弹出的时候是不是加载的布局太多.

(3)直播间的布局太过于复杂，导致软键盘弹出时绘制卡顿。

在长期的测试发现一个现象，就是在高端机型上这种状态不明显，而在底端机型问题比较严重，有时候弹出软键盘卡顿很长时间。

过了半个月博主思路换了，想想软键盘弹出卡顿，能不能从卡顿原因下手，来解决问题，后来找到了BlockCanary，接入使用后发现：

```
原因：
竟然是 主页在软键盘每次弹出或者关闭的时候重新绘制.因为当时BlockCanary当时指向主页的RecyclerView重绘，我当时想是
```

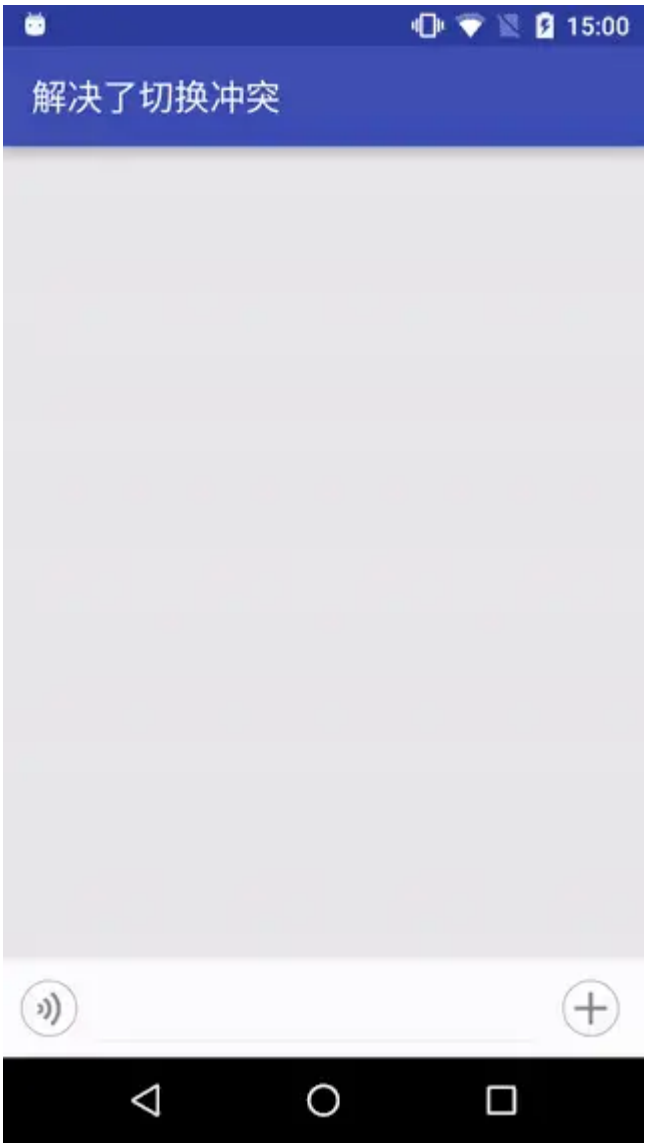
最后，博主凭直觉认为和主页SingleTask有关，因为没有确切的理由，这里只是提出自己的观点，而最后问题也解决了。

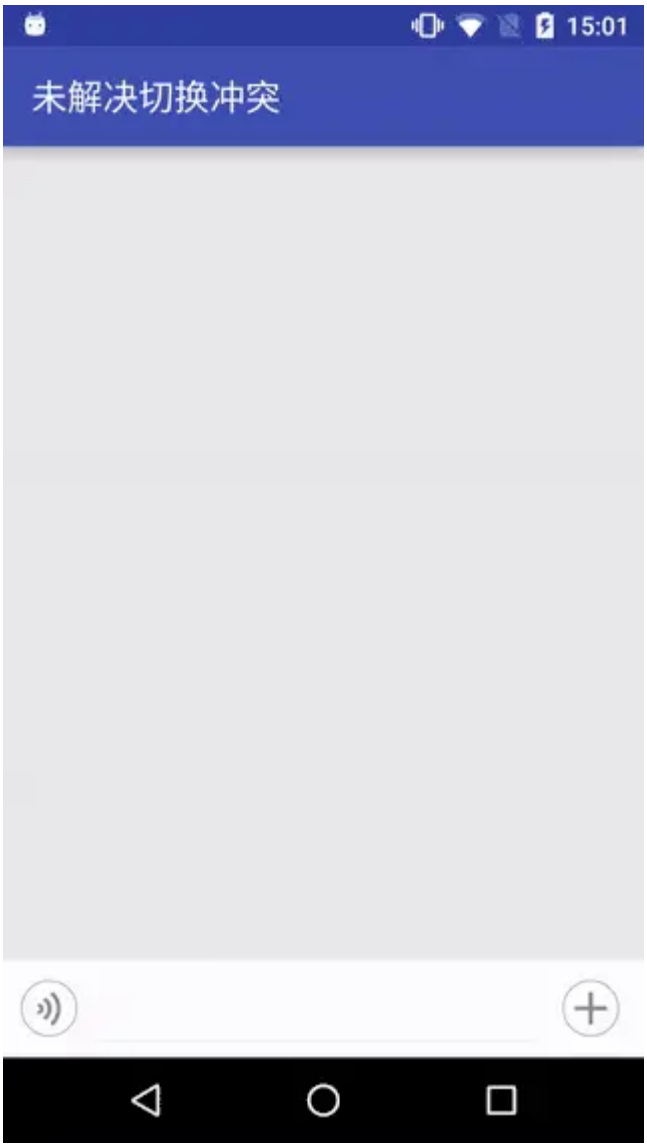
如果没有BlockCanary我永远发现不了，卡顿的原因是在**看不见的主页。**

后来我给主页设置成adjustNothing因为主页不需要弹出软键盘。

实际案例5：Android键盘面板冲突，布局闪动的解决方法

这个问题是我在找Android软键盘相关的问题的时候发现的，这块我也没遇到这个问题，所以给大家两个相关的介绍的地址，希望能对大家有帮助。





解决Android软键盘，布局闪动的相关博文

解决Android软键盘布局闪动的Demo

开源的Android软键盘布局闪动的解决方案

总结

(1)第一次写这么长的博客，感觉会有一些不足，各位看官如果有不合理的地方，或者有误的地方请直接指出。

(3)写完这篇博客之后，感觉博客干货还是不多，所以定位这篇文章算是总结性质加上实际案例性质的博客。

(4)Android软键盘的总结就差不多到这里，希望各位看官，如果看到这里有收获，就点点赞，灌灌水，顶一波，这样博主才有写下去的动力。

(5)感谢小辉同学的校验，调整了文章中不通顺的地方。

参考文档

1.彻底搞定Android开发中软键盘的常见问题
http://blog.csdn.net/mynameishuangshuai/article/details/51567357

2.Android UI(EditText)详解
http://blog.csdn.net/qq_28057577/article/details/51919965?locationNum=12&fps=1

3.微信软键盘布局闪动问题
https://blog.dreamtobe.cn/2015/09/01/keyboard-panel-switch/

Android

相关热门文章

Android编译期插桩，让程序自己写代码(一)

年糕妈妈技术团队 ♥ 9 3

从996.icu来谈一谈如何高效支配时间

Gityuan ♥ 15 15

声网 Agora SDK 使用体验征文大赛

稀土君 ♥ 16 1

Android--关闭某个指定activity

程序yang ♥ 21 3

评论

输入评论...

栗子酱油饼 Android @ O

用了SoftKeyBroadManager，回调的软键盘高度，好像含有 状态栏的高度，我把数值减了一下就正确了。。。请问一下大佬，为什么我这Activity里面返回的软键盘会有状态栏高度？

2月前 1 回复

Vander (作者) Android开发 @ ...

回复 栗子酱油饼: 一年前的文章哈。记得不清楚，别怪我哈。传入的根布局，然后去拿到activity的根布局也就是包含状态栏和标题栏那一层。所以你得到的结果是包含状态栏的高度。

2月前

栗子酱油饼 Android @ O

回复 Vander (作者): 哇，大佬秒回！好的，我想也是这样！谢谢！

2月前

Rox

很棒，开发这么多年android，第一次静下来看看关于键盘的文章。相形见绌啊- -

5月前 1 回复

Vander (作者) Android开发 @ ...

回复 Rox: 客气了.感觉开发Android时间长的话,都有自己擅长的领域.

5月前

beforenights 喵喵喵 @ 喵呜不停