Android 键盘篇

工程师 李洪江

一 键盘控制类

InputMethodManager

二 输入法内存泄漏

Android 输入法的一个 bug，在 15<=API<=23 中都存在。

参考 1：http://www.jianshu.com/p/aa2555628b17

参考 2：http://blog.csdn.net/a06_kassadin/article/details/51960273

三 窗口键盘模式

使用的注意点:

1、使用对象通常是 TextView 或者他的子类

2、通过设置 Activity 的 windowSoftInputMode 属性来处理当键盘显示的时候被隐藏的空间的处理方式以及软键盘是否默认显示。

android:windowSoftInputMode="stateAlwaysVisible|adjustPan"

- stateUnspecified：软键盘状态并没有指定，系统将选择一个合适的状态或依赖于主题的设置
- stateUnchanged：当这个Activity 出现时，软键盘将一直保持在上一个Activity 里的状态
- stateHidden：用户选择Activity 时，软键盘总是被隐藏
- stateAlwaysHidden：当该Activity 主窗口获取焦点时，软键盘也总是被隐藏
- stateVisible：软键盘通常是可见的
- stateAlwaysVisible：用户选择了该Activity 后软键盘总是显示的状态
- adjustUnspecified：默认设置，通常由系统自行决定是隐藏还是显示
- adjustResize：该Activity 总是调整屏幕的大小以便留出软键盘的空间
- adjustPan：当前窗口自动移动，以便当前焦点不被键盘覆盖，用户总能看见输入内容的部分

四 常用方法

1 控制键盘显示

```
public static void showSoftInputMethod(Context context, View view) {
    InputMethodManager iMM = (InputMethodManager) context.
                    getSystemService(Context.INPUT_METHOD_SERVICE);
    iMM.showSoftInput(view, 0);
}
```

2 控制键盘隐藏

```
public static void hideSoftInputMethod(Context context, View view) {
    InputMethodManager imm = (InputMethodManager) context.
                        getSystemService(Context.INPUT_METHOD_SERVICE)
    if (imm.isActive()) {
        imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
    }
}
```

3 建议不要用此法

```
public static void toggleSoftInputMethod(Context context) {
     try {
         InputMethodManager inputMethodManager = (InputMethodManager) context
                 .getSystemService(Context.INPUT_METHOD_SERVICE);
         inputMethodManager.toggleSoftInput(0, InputMethodManager.HIDE_NOT_ALWAYS);
     } catch (Exception e) {
         e.printStackTrace();
     }
 }
```

使用这个相当使用了键盘总是可见或者总是隐藏了。
当键盘没关闭时候，页面关闭了，键盘不会关闭。
效果类似 stateAlwaysVisible

五 常见使用场景分析

1 一个页面有 EditText 的时候，默认软键盘状态是 stateUnspecified。不指定的话，获取了焦点会弹出键盘。

方案一:配置 windowSoftInputMode 属性

不想第一次进来出现键盘可以指定：stateHidden
想第一次进来出现键盘可以指定：stateVisible

方案二：可以使用代码控制。

```
context.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_VISIBLE);
context.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);
```

2 默认软键盘状态是 stateUnspecified。当 EditText 在靠底部时候，系统会自动空闲出键盘布局来解决覆盖问题。
策略一：当页面不是滚动的时候，页面整体上移，留出键盘空间。

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

  <com.supets.commons.widget.CommonHeader
      android:id="@+id/header"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" />

  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:background="@android:color/white"
      android:orientation="vertical">

      <EditText
          android:layout_width="match_parent"
          android:layout_height="100dp"
          android:text="2" />

      <EditText
          android:layout_width="match_parent"
          android:layout_height="100dp"
          android:text="2" />

  </LinearLayout>

</LinearLayout>
```

策略二：当页面含有滚动容器时候，滚动容器会调整上移，留出键盘空间。

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

  <com.supets.commons.widget.CommonHeader
      android:id="@+id/header"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" />

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

      <LinearLayout
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:background="@android:color/white"
          android:orientation="vertical">

          <EditText
              android:layout_width="match_parent"
              android:layout_height="100dp"
              android:text="2" />

          <EditText
              android:layout_width="match_parent"
              android:layout_height="100dp"
              android:text="2" />

      </LinearLayout>

  </ScrollView>

</LinearLayout>
```

3 键盘未关闭引起上一个页面发生布局抖动，主要是 adjustpan 或者 adjustResize 引起的

```
android:windowSoftInputMode="adjustNothing"
```

4 针对下面这种布局

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.
    android:layout_width="match_parent"
    android:layout_height="match_parent"
 >

    <com.supets.commons.widget.CommonHeader
        android:id="@+id/header"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true" />

    <ScrollView
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@+id/bottom"
        android:layout_below="@+id/header">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@android:color/white"
            android:orientation="vertical">
            <EditText
                android:layout_width="match_parent"
                android:layout_height="100dp"
                android:singleLine="true"
                android:text="2" />
        </LinearLayout>
    </ScrollView>

    <Button
        android:id="@+id/bottom"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_alignParentBottom="true"
        android:gravity="center"
        android:text="一直在底部" />

</RelativeLayout>
```

底部悬停中间含有编辑框

如果是 adjustpan，跟布局是 ReleativeLayout 整个布局会根据键盘大小推动上移，键盘会盖住编辑下面。适合底部悬停页面。

如果是 adjustResize,跟布局是 ReleativeLayout 滚动布局和底部向上推动。 适合底部编辑框页面，比如聊天列表。配合键盘监听布局。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/a
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <com.supets.commons.widget.CommonHeader
        android:id="@+id/header"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true" />
    <ScrollView
        android:id="@+id/content"
        android:layout_weight="1"
        android:background="#ff0000"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_below="@+id/header">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">
            <EditText
                android:layout_width="match_parent"
                android:layout_height="100dp"
                android:singleLine="true"
                android:text="2" />
        </LinearLayout>
    </ScrollView>
    <Button
        android:id="@+id/bottom"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:gravity="center"
        android:text="一直在底部" />
</LinearLayout>
```

底部悬停中间含有编辑框

如果是 adjustpan，跟布局是 LinearLayout 整个布局会根据键盘大小推动上移，键盘会盖住编辑下面。 适合底部悬停页面。

如果是 adjustResize,跟布局是 LinearLayout 滚动布局和底部向上推动。适合底部编辑框页面，比如聊天列表。

5 被软键盘遮盖处理

原因：如果 EditText 设置了 gravity="center|right"其中之一且同时设置了 singleLine="true"，就会导致屏幕底部的 EditText 连续点击弹出键盘时，从第二次开会一直遮挡住 EditText。

解决办法：

```
android:lines="1"
android:gravity="center"
```

或者外面包裹布局，让 EditText 居中，适合宽度由内容决定。

6 布局监听器使用

```
public class KeyboardListenRelativeLayout extends RelativeLayout {
    public static final byte KEYBOARD_STATE_SHOW = -3;
    public static final byte KEYBOARD_STATE_HIDE = -2;
    public static final byte KEYBOARD_STATE_INIT = -1;
    private boolean mHasInit = false;
    private boolean mHasKeyboard = false;
    private int mHeight;
    private KeyboardListenRelativeLayout.IOnKeyboardStateChangedListener onKeyboardStateChangedListener;

    public KeyboardListenRelativeLayout(Context context) {
        super(context);
    }

    public KeyboardListenRelativeLayout(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public KeyboardListenRelativeLayout(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
    }

    public void setOnKeyboardStateChangedListener(KeyboardListenRelativeLayout.
              IOnKeyboardStateChangedListener onKeyboardStateChangedListener) {
        this.onKeyboardStateChangedListener = onKeyboardStateChangedListener;
    }

    protected void onLayout(boolean changed, int l, int t, int r, int b) {
        super.onLayout(changed, l, t, r, b);
        if(!this.mHasInit) {
            this.mHasInit = true;
            this.mHeight = b;
            if(this.onKeyboardStateChangedListener != null) {
                this.onKeyboardStateChangedListener.onKeyboardStateChanged(-1);
                Log.v("key", "KEYBOARD_STATE_INIT");
            }
        } else {
            this.mHeight = this.mHeight < b?b:this.mHeight;
        }


        if(this.mHasInit && this.mHeight > b && !this.mHasKeyboard) {
            this.mHasKeyboard = true;
            if(this.onKeyboardStateChangedListener != null) {
                this.onKeyboardStateChangedListener.onKeyboardStateChanged(-3);
                Log.v("key", "KEYBOARD_STATE_SHOW");
            }
        }

        if(this.mHasInit && this.mHasKeyboard && this.mHeight == b) {
            this.mHasKeyboard = false;
            if(this.onKeyboardStateChangedListener != null) {
                this.onKeyboardStateChangedListener.onKeyboardStateChanged(-2);
                Log.v("key", "KEYBOARD_STATE_HIDE");
            }
        }

    }

    public interface IOnKeyboardStateChangedListener {
        void onKeyboardStateChanged(int var1);
    }
}
```

在有时候，底部跟随一起滚动，键盘出来，整个底部显示出来。

```
@Override
public void onKeyboardStateChanged(int i) {
    if (i== KeyboardListenRelativeLayout.KEYBOARD_STATE_SHOW){
        mScrollView.fullScroll(ScrollView.FOCUS_DOWN);
    }
}
```

7 终极键盘是否显示方案

```
public static boolean isKeyboardShown(View rootView) {
      final int softKeyboardHeight = 100;
      Rect r = new Rect();
      rootView.getWindowVisibleDisplayFrame(r);
      DisplayMetrics dm = rootView.getResources().getDisplayMetrics();
      int heightDiff = rootView.getBottom() - r.bottom;
      return heightDiff > softKeyboardHeight * dm.density;
 }
```

```
@Override
protected void onDestroy() {
    super.onDestroy();
    removeListenerToRootView();
}
//onCreate
private void setListenerToRootView() {
    View rootView = getWindow().getDecorView().findViewById(android.R.id.content);
    rootView.getViewTreeObserver().addOnGlobalLayoutListener(mListener);
}

private void removeListenerToRootView() {
    View rootView = getWindow().getDecorView().findViewById(android.R.id.content);
    if (mListener != null) {
        rootView.getViewTreeObserver().removeOnGlobalLayoutListener(mListener);
    }
}

private ViewTreeObserver.OnGlobalLayoutListener mListener = new ViewTreeObserver.OnGlobalLayoutListener() {
    @Override
    public void onGlobalLayout() {
        View rootView = getWindow().getDecorView().findViewById(android.R.id.content);
        if (rootView != null && !InputMethodUtils.isKeyboardShown(rootView)) {
            etSecKillStocks.clearFocus();
            etSecKillDiscount.clearFocus();
            etSecKillPrice.clearFocus();
            etKeyWord.clearFocus();
        }
    }
};
```