Android EditText 篇

工程师 李洪江

一 完美解决 EditText 和 ScrollView 的滚动冲突

参考：http://blog.csdn.net/z191726501/article/details/50701300

知识点：

1 触摸的是 EditText 并且当前 EditText 可以滚动则将事件交给 EditText 处理；否则将事件交由其父类处理。

2 滚动方向判断

```
public class FixScrollViewEditTextListener implements EditText.OnTouchListener {

    @Override
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if ((view instanceof EditText && canVerticalScroll((EditText) view))) {
            view.getParent().requestDisallowInterceptTouchEvent(true);
            if (motionEvent.getAction() == MotionEvent.ACTION_UP) {
                view.getParent().requestDisallowInterceptTouchEvent(false);
            }
        }
        return false;
    }

    private boolean canVerticalScroll(EditText editText) {
        if (Build.VERSION.SDK_INT >= 14) {
            //垂直方向上可以滚动
            return editText.canScrollVertically(-1) || editText.canScrollVertically(0);
        } else {
            //滚动的距离
            int scrollY = editText.getScrollY();
            //控件内容的总高度
            int scrollRange = editText.getLayout().getHeight();
            //控件实际显示的高度
            int scrollExtent = editText.getHeight() -
                        editText.getCompoundPaddingTop()
                        - editText.getCompoundPaddingBottom();
            //控件内容总高度与实际显示高度的差值
            int scrollDifference = scrollRange - scrollExtent;
            if (scrollDifference == 0) {
                return false;
            }
            return (scrollY > 0) || (scrollY < scrollDifference - 1);
        }
    }
}
```

二 过滤器使用

1 长度限制

```
mEditText.setFilters(new InputFilter[]{new InputFilter.LengthFilter(10)});
```

2 表情过滤

```
public class EmojiInputFilter implements InputFilter {

String EMOJI="[\ud83c\udc00-\ud83c\udfff]|[\ud83d\udc00-\ud83d\udfff]|[☀-➿]";

Pattern emoji = Pattern.compile(EMOJI, 66);

public EmojiInputFilter() {}

public CharSequence filter(CharSequence source, int start, int end,
        Spanned dest, int dstart, int dend) {
 Matcher emojiMatcher = this.emoji.matcher(source);
 return emojiMatcher.find()?"":null;
}

}
```

3 金额过滤器

```
public class CashierInputFilter implements InputFilter {
    Pattern mPattern;
    private float MAX_VALUE = Integer.MAX_VALUE;//输入的最大金额
    private static final int POINTER_LENGTH = 2;//小数点后的位数
    private static final String POINTER = ".";
    private static final String ZERO = "0";

    public CashierInputFilter() {
        mPattern = Pattern.compile("([0-9]|\\.)*");
    }

    public CashierInputFilter(float maxvaule) {
        this.MAX_VALUE = maxvaule;
        mPattern = Pattern.compile("([0-9]|\\.)*");
    }

    /**
     * @param source 新输入的字符串
     * @param start  新输入的字符串起始下标，一般为0
     * @param end    新输入的字符串终点下标，一般为source长度-1
     * @param dest   输入之前文本框内容
     * @param dstart 原内容起始坐标，一般为0
     * @param dend   原内容终点坐标，一般为dest长度-1
     * @return 输入内容
     */
    @Override
    public CharSequence filter(CharSequence source, int start, int end,
                               Spanned dest, int dstart, int dend) {
        String sourceText = source.toString();
        String destText = dest.toString();

        //验证删除等按键
        if (TextUtils.isEmpty(sourceText)) {
            return "";
        }

        Matcher matcher = mPattern.matcher(source);
        //已经输入小数点的情况下，只能输入数字
        if (destText.contains(POINTER)) {
            if (!matcher.matches()) {
                return "";
            }
            Matcher matcher = mPattern.matcher(source);
            //已经输入小数点的情况下，只能输入数字
            if (destText.contains(POINTER)) {
                if (!matcher.matches()) {
                    return "";
                } else {
                    if (POINTER.equals(source.toString())) {  //只能输入一个小数点
                        return "";
                    }
                }
                //验证小数点精度，保证小数点后只能输入两位
                int index = destText.indexOf(POINTER);
                int length = dend - index;
                if (length > POINTER_LENGTH) {
                    return dest.subSequence(dstart, dend);
                }
            } else {
                /**
                 * 没有输入小数点的情况下，只能输入小数点和数字
                 * 1. 首位不能输入小数点
                 * 2. 如果首位输入0，则接下来只能输入小数点了
                 */
                if (!matcher.matches()) {
                    return "";
                } else {
                    //首位不能输入小数点
                    if ((POINTER.equals(source.toString())) && TextUtils.isEmpty(destText)) {
                        return "";
                    //如果首位输入0，接下来只能输入小数点
                    } else if (!POINTER.equals(source.toString()) && ZERO.equals(destText)) {
                        return "";
                    }
                }
            }
            //验证输入金额的大小
            float sumText = Float.parseFloat(destText + sourceText);
            if (sumText > MAX_VALUE) {
                return dest.subSequence(dstart, dend);
            }
        return dest.subSequence(dstart, dend) + sourceText;
        }
    }
```

三 常用属性使用

1 输入类型

android:inputType="none"

android:inputType="text"

android:inputType="textCapCharacters" 字母大写

android:inputType="textCapWords" 首字母大写

android:inputType="textCapSentences" 仅第一个字母大写

android:inputType="textAutoCorrect" 自动完成

android:inputType="textAutoComplete" 自动完成

android:inputType="textMultiLine" 多行输入

android:inputType="textImeMultiLine" 输入法多行（如果支持）

android:inputType="textNoSuggestions" 不提示

android:inputType="textUri" 网址

android:inputType="textEmailAddress" 电子邮件地址 ‘

android:inputType="textEmailSubject" 邮件主题
android:inputType="textShortMessage" 短讯
android:inputType="textLongMessage" 长信息
android:inputType="textPersonName" 人名
android:inputType="textPostalAddress" 地址
android:inputType="textPassword" 密码-----比较常用于登录
android:inputType="textVisiblePassword" 可见密码
android:inputType="textWebEditText" 作为网页表单的文本
android:inputType="textFilter" 文本筛选过滤
android:inputType="textPhonetic" 拼音输入 //数值类型
android:inputType="number" 数字---- 数字键盘
android:inputType="numberSigned" 带符号数字格式---- 数字键盘
android:inputType="numberDecimal" 带小数点的浮点格式---- 数字键盘
android:inputType="phone" 拨号键盘
android:inputType="datetime" 时间日期
android:inputType="date" 日期键盘
android:inputType="time" 时间键盘

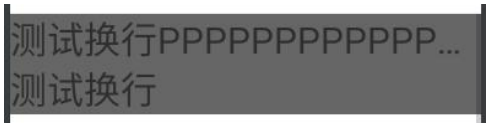
比如：想要设置一个可编辑的文本框的输入内容为只能输入数字，则就可以：
（1）xml 中定义 InputType 为 number
（2）代码中设置 InputType 为 TYPE_CLASS_NUMBER | TYPE_NUMBER_VARIATION_NORMAL

具体代码设置参考：
http://developer.android.com/reference/android/widget/TextView.html#attr_android:inputType

2 单行属性
代码和效果

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:singleLine="true"
    android:textSize="30dp"
    android:background="#666666"
    android:text="测试换行PPPPPPPPPPPP">

<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:maxLines="1"
    android:background="#666666"
    android:textSize="30dp"
    android:text="测试换行PPPPPPPPPPPP" />
```



分析：
maxLines 是在限制高度， singleLine 是强制不让换行。
具体效果有什么区别呢？
从高度来讲是一样的，两者肯定都显示一行

从换行的位置来讲就有区别了，maxLines 并不会改变其换行的位置，而 singleLine 则会。从这个角度讲，singleLine 的显示会好一些，因为如果超过一行 singleLine 会在一行内显示，后面加上"...",而 maxlines="1" 则不会，它依然会在原来换行的位置换行，所以有时候一行不满，但是却不显示剩下的部分。

android:maxLines=“1”代替 android:singleLine=”true”，需要设置 EditText 设置 android:inputType="text"才有效，但不会显示...，不设置使用 ellipsize 属性会显示...,但可能显示不满一行。

3 android 中 minLines 和 maxLines 的区别

maxLines 的 EditText 最大行数为 3 行，当输入的内容超过 3 行后，它形状的大小不会根据输入内容的多少而改变，反正它显示的内容就是 3 行。

minLines 的 EditText 是至它至少显示 3 行内容，当输入的内容超过 3 行后，它形状的大小根据输入内容的多少而改变。

Lines 的 EditText 是固定显示 3 行内容，形状不会根据输入内容的多少而改变。

4 限制文本显示的长度

android:maxLength="13"作用就是限制 TextView 只能显示 13 个文本长度。

android:maxEms="13" 作用都是设置 TextView 的字符宽度。

5 限制输入字符

android:digits="1234567890."

四 特殊按键处理

1 拦截返回键

重写 TextView 子类的 OnKeyPreIme 方法

```
public boolean onKeyPreIme(int keyCode, KeyEvent event) {
      if(keyCode == 4 && (event == null || event.getAction() == 1)
            && this.mListener != null) {
          this.mListener.onBackPress();
      }

      return super.onKeyPreIme(keyCode, event);
}
```

常用按键定义： http://www.jianshu.com/p/498a8389524c

2 回车键处理

Edittext 设置 android:imeOptions="actionSearch"2.3 及以上版本不起作用，2.3 以下好使。

键盘上的回车键本来就是换行用的，当设置单行后，回车换行就失去作用了，进而就可以用作其他用途了，例如：搜索、发送、go 等。

android:singleLine="true"

或者

android:inputType="text"

android:maxLines="1"

android:imeOptions 属性

1.actionUnspecified 未指定,对应常量 EditorInfo.IME_ACTION_UNSPECIFIED.

2.actionNone 没有动作,对应常量 EditorInfo.IME_ACTION_NONE

3.actionGo 去往,对应常量 EditorInfo.IME_ACTION_GO

4.actionSearch 搜索,对应常量 EditorInfo.IME_ACTION_SEARCH

5.actionSend 发送,对应常量 EditorInfo.IME_ACTION_SEND

6.actionNext 下一个,对应常量 EditorInfo.IME_ACTION_NEXT

7.actionDone 完成,对应常量 EditorInfo.IME_ACTION_DONE

捕捉编辑框软键盘 enter 事件:

1）setOnKeyListener

2）OnEditorActionListener

实现 android 按下回车键便隐藏输入键盘，有两种方法：

1)如果布局是多个 EditText,为每个 EditText 控件设置 android:singleLine=”true”，弹出的软盘输入法中回车键为 next，直到最后一个获取焦点后显示为 Done，点击 Done 后，软盘输入键盘便隐藏。或者将 EditText 的 imeOptions 属性设置 android:imeOptions=”actionDone”，则不管是不是最后一个 EditText，点击回车键即隐藏输入法。

2)监听 Enter 的事件,编写 Enter 的事件响应。设置文本框的 OnKeyListener,当 keyCode==KeyEvent.KEYCODE_ENTER 的时候，表明 Enter 键被按下，就可以编写自己事件响应功能了。

```
EditText password=(EditText)findViewById(R.id.password);

password.setOnKeyListener(new OnKeyListener(){

    public boolean onKey(View v, int keyCode, KeyEvent event) {

        if(keyCode == KeyEvent.KEYCODE_ENTER){
            InputMethodManager imm = (InputMethodManager)v.getContext().
                    getSystemService(Context.INPUT_METHOD_SERVICE);
            if(imm.isActive()){
              imm.hideSoftInputFromWindow(v.getApplicationWindowToken(), 0 );
             }
            return true;
        }
       return false;

    }

});
```

3 搜索按键处理

```
//设置android:imeOptions=“actionSearch ”
//代码设置监听
mSearchInput.setOnEditorActionListener(this);

@Override
 public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
     //单机键盘上搜索按键，执行搜索事件
     if (actionId == EditorInfo.IME_ACTION_SEARCH) {
         String input = v.getText().toString().trim();
         return search(input);
     }
     return false;
 }

 private boolean search(String input) {
     if (!TextUtils.isEmpty(input)) {
         if (mSearchListener != null) {
             mSearchListener.onSearch(input);
         }
         return true;
     } else {
         MYUtils.showToastMessage(MiaTextUtils.getString(R.string.m_map_search_input_none));
     }
     return false;
 }

 public interface SearBarListener {
     void onSearch(String result);

     void gotoSearch();
 }
```

五 单击事件处理

```
//设置禁止获取焦点
//编辑框单击事件
mSearchInput.setFocusable(false);
mSearchInput.setOnClickListener(this);
```

六 文本监听器

1 控制清除按钮显示

```
mSearchInput.addTextChangedListener(new MiaEditText.SimpleTextWatch() {
      @Override
      public void onTextChanged(CharSequence s, int start, int before, int count) {
          mClean.setVisibility(s.length() > 0 ? View.VISIBLE : View.GONE);
      }
  });
```

2 监控行数变化，内容变化，长度变化。

```
public class CommentEditText extends EmojiconEditText implements TextWatcher {

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

	public CommentEditText(Context context) {
		super(context);
		init();
	}

	private int MAX140 = 140;
	private static final int MAXLINES = 4;
	private EditBackListener listenr;
	private int beforeLine = 0;
	private int afterLine = 0;
	private int beforeNum = 0;

	public void setFontNum(int maxsize) {
		MAX140 = maxsize;
	}

	rivate void init() {
		addTextChangedListener(this);
		setMaxLines(MAXLINES);
		setUseSystemDefault(false);
		setEmojiconSize((int) getTextSize());
	}

	@Override
	public boolean onKeyPreIme(int keyCode, KeyEvent event) {
		if (event != null && event.getKeyCode() == KeyEvent.KEYCODE_BACK) {
			if (listenr != null && listenr.OnBackPressed()) {
				return true;
			}
		}
		return false;
	}


	@Override
	public void afterTextChanged(Editable editText) {
		afterLine = this.getLineCount();
		// 限制最大输入行数
		if (beforeLine != afterLine) {
			// 行数发生变化时候调用
			if (listenr != null) {listenr.OnLineChange(beforeLine < afterLine);}
		}
		String input = editText.toString();
		int length = input.length();
		if (length > MAX140) {
			this.setText(input.substring(0, MAX140));
			this.setSelection(MAX140);
			// 字数超过140
			if (listenr != null) {listenr.OnFontOver();}
		}

		if (length > 0) {
			if (listenr != null) {listenr.OnVisibleSend(true);}
		} else {
			if (listenr != null) {listenr.OnVisibleSend(false);}
		}

		if (length < beforeNum) {
			Log.v("CommentEditText", "delete mode");
			return;
		}
		super.onTextChanged(null, 0, 0, 0);  emoji处理
	}

	@Override
	public void beforeTextChanged(CharSequence arg0, int arg1, int arg2,
			int arg3) {
		beforeLine = this.getLineCount();
		beforeNum = arg0.length();
	}
	@Override
	public void onTextChanged(CharSequence text, int start, int lengthBefore,
			int lengthAfter) {
	}
```

```
    public void setEditBackListener(EditBackListener listenr) {
            this.listenr = listenr;
    }
    public interface EditBackListener {
             boolean OnBackPressed();

             boolean OnLineChange(boolean isup);

             boolean OnFontOver();

             void OnVisibleSend(boolean visible);
    }

    public static  class SimpleEditBackListener implements EditBackListener {

            @Override
            public boolean OnBackPressed() {
                    return false;
            }

            @Override
            public boolean OnLineChange(boolean isup) {
                    return false;
            }

            @Override
            public boolean OnFontOver() {
                    return false;
            }

            @Override
            public void OnVisibleSend(boolean visible) {

            }
    }

}
```

七 禁止复制粘贴问题解决方案

1 通过 edittext 的设置属性

```
edittext.setLongClickable(false);
edittext.setTextIsSelectable(false);
```

2 通过设置 callback 监听

```
setCustomSelectionActionModeCallback(new ActionMode.Callback() {
@Override
 public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        return false;
}

@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
        return false;
}

@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        return false;
 }

@Override
public void onDestroyActionMode(ActionMode mode) {

}
});但是用过的人知道这些对小米手机并没有给什么卵用
```

3 安卓监听剪切复制粘贴事件

参考 1 http://blog.csdn.net/zhaizu/article/details/70154301
参考 2 http://blog.csdn.net/XiFangzheng/article/details/52791508

4 一种禁止复制粘贴选择方法

```
setOnEditorActionListener(new OnEditorActionListener() {

        @Override
        public boolean onEditorAction(TextView v, int actionId,
                        KeyEvent event) {
                if (actionId==EditorInfo.IME_ACTION_SEND) {
                //只处理回车键Action
                return true;
                }
                return false;
        }
});
```