Android 中常见控件的介绍和使用

1 TextView 文本框

1.1 TextView 类的结构

TextView 是用于显示字符串的组件，对于用户来说就是屏幕中一块用于显示文本的区域。TextView 类的层次关系如下：

java.lang.Object

↳ android.view.View

↳ android.widget.TextView

直接子类：

Button, CheckedTextView, Chronometer, DigitalClock, EditText

间接子类：

AutoCompleteTextView, CheckBox, CompoundButton, ExtractEditText,MultiAutoCompleteTextView, RadioButton, ToggleButton

1.2 TextView 类的方法

主要方法	功能描述	返回值
TextView	TextView 的构造方法	Null
getDefaultMovementmethod	获取默认的箭头按键移动方式	Movementmethod
getText	获得 TextView 对象的文本	CharSquence
length	获得 TextView 中的文本长度	Int
getEditableText	取得文本的可编辑对象，通过 这个对象可对 TextView 的文本进行操作，如在光标之后插入字符	Void
getCompoundPaddingBottom	返回底部填充物	Int
setCompoundDrawables	设置图像显示的位置，在 设置该 Drawable 资源之前需要调用 setBounds（Rect）	Void
setCompoundDrawablesWithIntrinsicBounds	设置 Drawable 图像的显示位置，但其边界不变	Void
setPadding	根据位置设置填充物	Void
getAutoLinkMask	返回自动连接的掩码	Void
setTextColor	设置文本显示的颜色	Void
setHighlightColor	设置文本选中时显示的颜色	Void
setShadowLayer	设置文本显示的阴影颜色	Void
setHintTextColor	设置提示文字的颜色	Void

setLinkTextColor	设置链接文字的颜色	Void
setGravity	设置当 TextView 超出了文本本身时横向以及垂直对齐	Void
getFreezesText	设置该视图是否包含整个文本，如果包含则返回真值，否则返回假值	Boolean

1.3 TextView 标签的属性

XML 属性

属性名称	描述
android:autoLink	设置是否当文本为 URL 链接/email/电话号码/map 时，文本显示为可点击的链接。可选值(none/web/email/phone/map/all)
android:autoText	如果设置，将自动执行输入值的拼写纠正。此处无效果，在显示输入法并输入的时候起作用。
android:bufferType	指定 getText()方式取得的文本类别。选项 editable 类似于 StringBuilder 可追加字符，也就是说 getText 后可调用 append 方法设置文本内容。
android:capitalize	设置英文字母大写类型。此处无效果，需要弹出输入法才能看得到，参见 EditView 此属性说明。
android:cursorVisible	设定光标为显示/隐藏，默认显示。
android:digits	设置允许输入哪些字符。如“1234567890.+-*/%\n()”
android:drawableBottom	在 text 的下方输出一个 drawable，如图片。如果指定一个颜色的话会把 text 的背景设为该颜色，并且同时和 background 使用时覆盖后者。
android:drawableLeft	在 text 的左边输出一个 drawable，如图片。
android:drawablePadding	设置 text 与 drawable(图片)的间隔，与 drawableLeft、drawableRight、drawableTop、drawableBottom 一起使用，可设置为负数，单独使用没有效果。
android:drawableRight	在 text 的右边输出一个 drawable，如图片。
android:drawableTop	在 text 的正上方输出一个 drawable，如图片。
android:editable	设置是否可编辑。这里无效果，参见 EditView。
android:editorExtras	设置文本的额外的输入数据。在 EditView 再讨论。
android:ellipsize	设置当文字过长时,该控件该如何显示。有如下值设置：”start”——省略号显示在开头；”end”——省略号显示在结尾；”middle”——省略号显示在中间；”marquee” ——以跑马灯的方式显示(动画横向移动)
android:freezesText	设置保存文本的内容以及光标的位置。
android:gravity	设置文本位置，如设置成“center”，文本将居中显示。
android:hint	Text 为空时显示的文字提示信息，可通过 textColorHint 设置提示信息的颜色。比较奇怪的是 TextView 本来就相当于 Label，怎么会不设置 Text？！
android:imeOptions	附加功能，设置右下角 IME 动作与编辑框相关的动作，如 actionDone 右下角将显示一个“完成”，而不设置默认是一个回车符号。这个在 EditView 中再详细说明，此处无用。
android:imeActionId	设置 IME 动作 ID。在 EditView 再做说明，
android:imeActionLabel	设置 IME 动作标签。在 EditView 再做说明。
android:includeFontPadding	设置文本是否包含顶部和底部额外空白，默认为 true。
android:inputMethod	为文本指定输入法，需要完全限定名（完整的包名）。例如：com.google.android.inputmethod.pinyin，但是这里报错找不到。
android:inputType	设置文本的类型，用于帮助输入法显示合适的键盘类型。在 EditView 中再详细说明，这里无效果。
android:linksClickable	设置链接是否点击连接，即使设置了 autoLink。
android:marqueeRepeatLimit	在 ellipsize 指定 marquee 的情况下，设置重复滚动的次数，当设置为 marquee_forever 时表示无限次。

android:ems	设置 TextView 的宽度为 N 个字符的宽度。这里测试为一个汉字字符宽度，如图：
android:maxEms	设置 TextView 的宽度为最长为 N 个字符的宽度。与 ems 同时使用时覆盖 ems 选项。
android:minEms	设置 TextView 的宽度为最短为 N 个字符的宽度。与 ems 同时使用时覆盖 ems 选项。
android:maxLength	限制显示的文本长度，超出部分不显示。
android:lines	设置文本的行数，设置两行就显示两行，即使第二行没有数据。
android:maxLines	设置文本的最大显示行数，与 width 或者 layout_width 结合使用，超出部分自动换行，超出行数将不显示。
android:minLines	设置文本的最小行数，与 lines 类似。
android:lineSpacingExtra	设置行间距。
android:lineSpacingMultiplier	设置行间距的倍数。如”1.2”
android:numeric	如果被设置，该 TextView 有一个数字输入法。此处无用，设置后唯一效果是 TextView 有点击效果，此属性在 EdtiView 将详细说明。
android:password	以小点”.”显示文本
android:phoneNumber	设置为电话号码的输入方式。
android:privateImeOptions	设置输入法选项，此处无用，在 EditText 将进一步讨论。
android:scrollHorizontally	设置文本超出 TextView 的宽度的情况下，是否出现横拉条。
android:selectAllOnFocus	如果文本是可选择的，让他获取焦点而不是将光标移动为文本的开始位置或者末尾位置。TextView 中设置后无效果。
android:shadowColor	指定文本阴影的颜色，需要与 shadowRadius 一起使用。效果：
android:shadowDx	设置阴影横向坐标开始位置。
android:shadowDy	设置阴影纵向坐标开始位置。
android:shadowRadius	设置阴影的半径。设置为 0.1 就变成字体的颜色了，一般设置为 3.0 的效果比较好。
android:singleLine	设置单行显示。如果和 layout_width 一起使用，当文本不能全部显示时，后面用“…”来表示。如 android:text="test_ singleLine " android:singleLine="true" android:layout_width="20dp"将只显示“t…”。如果不设置 singleLine 或者设置为 false，文本将自动换行
android:text	设置显示文本.
android:textAppearance	设置文字外观。如“?android:attr/textAppearanceLargeInverse ”这里引用的是系统自带的一个外观，？表示系统是否有这种外观，否则使用默认的外观。可设置的值如下：textAppearanceButton/textAppearanceInverse/textAppearanceLarge/textAppearanceLargeInverse/textAppearanceMedium/textAppearanceMediumInverse/textAppearanceSmall/textAppearanceSmallInverse
android:textColor	设置文本颜色
android:textColorHighlight	被选中文字的底色，默认为蓝色
android:textColorHint	设置提示信息文字的颜色，默认为灰色。与 hint 一起使用。
android:textColorLink	文字链接的颜色.
android:textScaleX	设置文字之间间隔，默认为 1.0f。分别设置 0.5f/1.0f/1.5f/2.0f 效果如下：
android:textSize	设置文字大小，推荐度量单位”sp”，如”15sp”
android:textStyle	设置字形[bold(粗体) 0, italic(斜体) 1, bolditalic(又粗又斜) 2] 可以设置一个或多个，用“|”隔开
android:typeface	设置文本字体，必须是以下常量值之一：normal 0, sans 1, serif 2, monospace(等宽字体) 3]
android:height	设置文本区域的高度，支持度量单位：px(像素)/dp/sp/in/mm(毫米)

android:maxHeight	设置文本区域的最大高度
android:minHeight	设置文本区域的最小高度
android:width	设置文本区域的宽度，支持度量单位：px(像素)/dp/sp/in/mm(毫米)。
android:maxWidth	设置文本区域的最大宽度
android:minWidth	设置文本区域的最小宽度

1.4 TextView 的使用

我们可以在 XML 布局文件中声明及设置 TextView，也可以在代码中生成 TextView 组件。

在 xml 布局中：

```
 //显示超链接：
<TextView android:id="@+id/textView0" android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:textColor="#FF0000"
   android:textSize="18dip" android:background="#FFFFFF" android:text="拨打手机：13888888888"
  android:gravity="center_vertical|center_horizontal" android:autoLink="phone" />

<TextView android:id="@+id/textView1" android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:textColor="#FF0000"
  android:textSize="18dip" android:background="#00FF00"
  android:text="Google搜索：http://www.google.com.hk" android:gravity="center_vertical|center_horizontal"
  android:autoLink="web" />

<TextView android:id="@+id/textView2" android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:textColor="#FF0000"
  android:textSize="18dip"
  android:background="#FFFF00"
  android:text="发送邮件：1007857613@qq.com"
  android:gravity="center_vertical|center_horizontal" android:autoLink="email" />

//设置文字的滚屏
<TextView android:id="@+id/textView3" android:layout_width="fill_parent"
  android:layout_height="20dp" android:textSize="18dip"
  android:ellipsize="marquee" android:focusable="true"
  android:marqueeRepeatLimit="marquee_forever"
  android:focusableInTouchMode="true"
  android:scrollHorizontally="true"
  android:text="文字滚屏文字跑马灯效果加长加长加长加长加长加长加长加长加长加长加长"
  android:background="#FF0000"
  android:textColor="#FFFFFF">
      </TextView>

//设置字符阴影
<TextView android:id="@+id/TextView4" android:layout_width="fill_parent"
  android:layout_height="wrap_content" android:textSize="18dip"
```

```
 android:background="#69541b" android:textColor="#FFFFFF" android:text="设置字符串阴影颜色为绿色"
 android:shadowColor="#45b97c" android:shadowRadius="3.0"
 android:gravity="center_vertical|center_horizontal" />
      //设置字符的外形
<TextView android:layout_width="fill_parent"
   android:layout_height="wrap_content" android:textSize="18dip"
    android:background="#FFFFFF" android:textColor="#FF0000" android:text="设置文字外形为italic"
    android:textStyle="italic" android:gravity="center_vertical|center_horizontal" />

</LinearLayout>
```

在代码中：

```
/动态创建textView
        linearLayout = (LinearLayout)findViewById(R.id.linearLayout01);
        param    = new LinearLayout.LayoutParams(LinearLayout.LayoutParams.FILL_PARENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT);
           LinearLayout layout1 = new LinearLayout(this);
             layout1.setOrientation(LinearLayout.HORIZONTAL);
           TextView tv = new TextView(this);
           tv.setId(200);
           tv.setText("动态创建TextView");
           tv.setBackgroundColor(Color.GREEN);
           tv.setTextColor(Color.RED);
           tv.setTextSize(20);
           layout1.addView(tv, param);
           linearLayout.addView(layout1, param);
```

效果图：

2 EditText 编辑框

2.1 EditText 类的结构

EditText 和 TextView 的功能基本类似，他们之间的主要区别在于 EditText 提供了可编辑的文本框。

```
java.lang.Object
      android.view.View
```

android.widget.TextView
android.widget.EditText

直接子类：
AutoCompleteTextView, ExtractEditText

间接子类：
MultiAutoCompleteTextView

2.2 EditText 常用的方法

方法	功能描述	返回值
setImeOptions	设置软键盘的 Enter 键	void
getImeActionLable	设置 IME 动作标签	Charsequence
getDefaultEditable	获取是否默认可编辑	boolean
setEllipse	设置文件过长时控件的显示方式	void
setFreeezesText	设置保存文本内容及光标位置	void
getFreeezesText	获取保存文本内容及光标位置	boolean
setGravity	设置文本框在布局中的位置	void
getGravity	获取文本框在布局中的位置	int
setHint	设置文本框为空时，文本框默认显示的 字符	void
getHint	获取文本框为空时，文本框默认显示的 字符	Charsequence
setIncludeFontPadding	设置文本框是否包含底部和顶端的额外空白	void
setMarqueeRepeatLimit	在 ellipsize 指定 marquee 的情况下，设置重复滚动的次数， 当设置为 marquee_forever 时表示无限次	void

2.3 EditText 标签的主要属性

属性名称	描述
android:autoLink	设置是否当文本为 URL 链接/email/电话号码/map 时，文本显示为可点击的链接。可选值(none/web/email/phone/map/all)。这里只有在同时设置 text 时才自动识别链接，后来输入的无法自动识别。
android:autoText	自动拼写帮助。这里单独设置是没有效果的，可能需要其他输入法辅助才行，效果参见视频。
android:bufferType	指定 getText()方式取得的文本类别。选项 editable 类似于 StringBuilder 可追加字符， 也就是说 getText 后可调用 append 方法设置文本内容。spannable 则可在给定的字符区域使用样式，参见这里 1、这里 2。
android:capitalize	设置英文字母大写类型。设置如下值：sentences 仅第一个字母大写；words 每一个单词首字母大小，用空格区分单词；characters 每一个英文字母都大写。在模拟器上用 PC 键盘直接输入可以出效果，但是用软键盘无效果。
android:cursorVisible	设定光标为显示/隐藏，默认显示。如果设置 false，即使选中了也不显示光标栏。
android:digits	设置允许输入哪些字符。如“1234567890.+-*/%\n()”
android:drawableTop	在 text 的正上方输出一个 drawable。在 EditView 中的效果比较搞笑： ，居然在文本框里，而且删不了。
android:drawableBottom	在 text 的下方输出一个 drawable(如图片)。如果指定一个颜色的话会把 text 的背景设为该颜色，并且同时和 background 使用时覆盖后者。

android:drawableLeft	在 text 的左边输出一个 drawable(如图片)。
android:drawablePadding	设置 text 与 drawable(图片)的间隔，与 drawableLeft、drawableRight、drawableTop、drawableBottom 一起使用，可设置为负数，单独使用没有效果。
android:drawableRight	在 text 的右边输出一个 drawable，如图片。
android:editable	设置是否可编辑。仍然可以获取光标，但是无法输入。
android:editorExtras	指定特定输入法的扩展，如“com.mydomain.im.SOME_FIELD”。源码跟踪至 EditorInfo.extras，暂无相关实现代码。
android:ellipsize	设置当文字过长时,该控件该如何显示。有如下值设置：”start”——省略号显示在开头；”end”——省略号显示在结尾；”middle”—省略号显示在中间；”marquee” ——以跑马灯的方式显示(动画横向移动)
android:freezesText	设置保存文本的内容以及光标的位置。参见：这里。
android:gravity	设置文本位置，如设置成“center”，文本将居中显示。
android:hint	Text 为空时显示的文字提示信息，可通过 textColorHint 设置提示信息的颜色。
android:imeOptions	设置软键盘的 Enter 键。有如下值可设置：normal，actionUnspecified，actionNone，actionGo，actionSearch，actionSend，actionNext，actionDone，flagNoExtractUi，flagNoAccessoryAction，flagNoEnterAction。可用’|’设置多个。这里仅设置显示图标之用，参见文章末尾例子。
android:imeActionId	设置 IME 动作 ID，在 onEditorAction 中捕获判断进行逻辑操作。
android:imeActionLabel	设置 IME 动作标签。但是不能保证一定会使用，猜想在输入法扩展的时候应该有用。
android:includeFontPadding	设置文本是否包含顶部和底部额外空白，默认为 true。
android:inputMethod	为文本指定输入法，需要完全限定名（完整的包名）。例如：com.google.android.inputmethod.pinyin，但是这里报错找不到。关于自定义输入法参见这里。 sentences 仅第一个字母大写；words 每一个单词首字母大小，用空格区分单词；characters 每一个英文字母都大写
android:inputType	设置文本的类型，用于帮助输入法显示合适的键盘类型。有如下值设置：none、text、textCapCharacters 字母大小、textCapWords 单词首字母大小、textCapSentences 仅第一个字母大小、textAutoCorrect、textAutoComplete 自动完成、textMultiLine 多行输入、textImeMultiLine 输入法多行（如果支持）、textNoSuggestions 不提示、textEmailAddress 电子邮件地址、textEmailSubject 邮件主题、textShortMessage 短信息（会多一个表情按钮出来，点开如下图： ）、textLongMessage 长讯息？、textPersonName 人名、textPostalAddress 地址、textPassword 密码、textVisiblePassword 可见密码、textWebEditText 作为网页表单的文本、textFilte 文本筛选过滤、textPhonetic 拼音输入、numberSigned 有符号数字格式、numberDecimal 可带小数点的浮点格式、phone 电话号码、datetime 时间日期、date 日期、time 时间。部分参考这里。
android:marqueeRepeatLimit	在 ellipsize 指定 marquee 的情况下，设置重复滚动的次数，当设置为 marquee_forever 时表示无限次。
android:ems	设置 TextView 的宽度为 N 个字符的宽度。参见 TextView 中此属性的截图。
android:maxEms	设置 TextView 的宽度为最长为 N 个字符的宽度。与 ems 同时使用时覆盖 ems 选项。
android:minEms	设置 TextView 的宽度为最短为 N 个字符的宽度。与 ems 同时使用时覆盖 ems 选项。
android:maxLength	限制输入字符数。如设置为 5，那么仅可以输入 5 个汉字/数字/英文字母。
android:lines	设置文本的行数，设置两行就显示两行，即使第二行没有数据。

android:maxLines	设置文本的最大显示行数，与 width 或者 layout_width 结合使用，超出部分自动换行，超出行数将不显示。
android:minLines	设置文本的最小行数，与 lines 类似。
android:linksClickable	设置链接是否点击连接，即使设置了 autoLink。
android:lineSpacingExtra	设置行间距。
android:lineSpacingMultiplier	设置行间距的倍数。如”1.2”
android:numeric	如果被设置，该 TextView 有一个数字输入法。有如下值设置：integer 正整数、signed 带符号整数、decimal 带小数点浮点数。
android:password	以小点”.”显示文本
android:phoneNumber	设置为电话号码的输入方式。
android:privateImeOptions	提供额外的输入法选项(字符串格式)。依据输入法而决定是否提供，如这里所见。自定义输入法继承 InputMethodService。这篇文章也许有帮助。
android:scrollHorizontally	设置文本超出 TextView 的宽度的情况下，是否出现横拉条。
android:selectAllOnFocus	如果文本是可选择的，让他获取焦点而不是将光标移动为文本的开始位置或者末尾位置。TextView 中设置后无效果。
android:shadowColor	指定文本阴影的颜色，需要与 shadowRadius 一起使用。参见 TextView 中此属性的截图。
android:shadowDx	设置阴影横向坐标开始位置。
android:shadowDy	设置阴影纵向坐标开始位置。
android:shadowRadius	设置阴影的半径。设置为 0.1 就变成字体的颜色了，一般设置为 3.0 的效果比较好。
android:singleLine	设置单行显示。如果和 layout_width 一起使用，当文本不能全部显示时，后面用“…”来表示。如 android:text="test_ singleLine " android:singleLine="true" android:layout_width="20dp"将只显示“t…”。如果不设置 singleLine 或者设置为 false，文本将自动换行
android:text	设置显示文本.
android:textAppearance	设置文字外观。如“?android:attr/textAppearanceLargeInverse ”这里引用的是系统自带的一个外观，？表示系统是否有这种外观，否则使用默认的外观。可设置的值如下：textAppearanceButton/textAppearanceInverse/textAppearanceLarge/textAppearanceLargeInverse/textAppearanceMedium/textAppearanceMediumInverse/textAppearanceSmall/textAppearanceSmallInverse
android:textColor	设置文本颜色
android:textColorHighlight	被选中文字的底色，默认为蓝色
android:textColorHint	设置提示信息文字的颜色，默认为灰色。与 hint 一起使用。
android:textColorLink	文字链接的颜色.
android:textScaleX	设置文字缩放，默认为 1.0f。参见 TextView 的截图。
android:textSize	设置文字大小，推荐度量单位”sp”，如”15sp”
android:textStyle	设置字形[bold(粗体) 0, italic(斜体) 1, bolditalic(又粗又斜) 2] 可以设置一个或多个，用“|”隔开
android:typeface	设置文本字体，必须是以下常量值之一：normal 0, sans 1, serif 2, monospace(等宽字体) 3]
android:height	设置文本区域的高度，支持度量单位：px(像素)/dp/sp/in/mm(毫米)
android:maxHeight	设置文本区域的最大高度
android:minHeight	设置文本区域的最小高度
android:width	设置文本区域的宽度，支持度量单位：px(像素)/dp/sp/in/mm(毫米)，与 layout_width 的区别看这里。
android:maxWidth	设置文本区域的最大宽度
android:minWidth	设置文本区域的最小宽度

2.4 EditText 的使用

EditText 和 TextView 一样，既可以在 Xml 中声明实现，也可以在代码中动态的生成，关于在代码动态生成和 TextView 的类似，这里就不在赘述。下面以一个实例来说明 EditText 的简单使用：

XML 布局中：

```
<TextView android:text="TextView" android:id="@+id/textView1"
            android:layout_height="wrap_content" android:layout_width="fill_parent"></TextView>

 <!-- 用于输入数字的文本框 -->
     <EditText android:id="@+id/editText1" android:inputType="date"
            android:layout_width="fill_parent" android:layout_height="wrap_content"
            android:maxLength="40" android:hint="输入电话号码" android:textColorHint="#FF000000"
            android:phoneNumber="true" android:imeOptions="actionGo"></EditText>

     <!-- 用于输入密码的文本框 -->
     <EditText android:id="@+id/editText2" android:inputType="date"
            android:layout_width="fill_parent" android:layout_height="wrap_content"
            android:maxLength="40" android:hint="输入密码" android:textColorHint="#FF000000"
            android:password="true" android:imeOptions="actionSearch"></EditText>
```

/rec/values/string.xml 中：

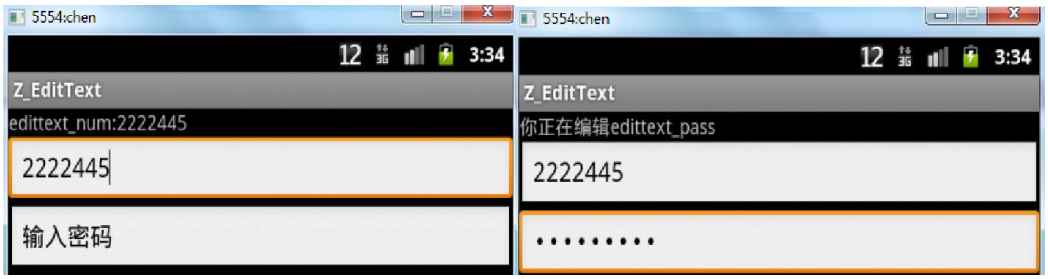
```
string   name="RadioButton1">Windows</string >
     <string   name="RadioButton2">Linux</string >
     <string   name="RadioButton3">Mocos</string >
     <string   name="RadioButton4">Java</string >
```

在 Activity 中让 editText 显示在屏幕上并实现监听：

```
/**
            * 获得TextView对象
            * 获得两个EditTextView对象
            */
           textview = (TextView)findViewById(R.id.textView1);
           edittext_num = (EditText)findViewById(R.id.editText1);
           edittext_pass= (EditText)findViewById(R.id.editText2);
           /**
            * 对EditText进行监听
            * 获得编辑框里面的内容并显示
            */
           edittext_num.addTextChangedListener(new TextWatcher() {

                @Override
                public void onTextChanged(CharSequence s, int start, int before, int count) {

                     //if(!(textview.getText().toString()==null))
                          textview.setText("edittext_num:"+edittext_num.getText().toString());
                }

                @Override
```

```
            public void beforeTextChanged(CharSequence s, int start, int count,
                    int after) {
                // TODO Auto-generated method stub

            }

            public void afterTextChanged(Editable s) {
                // TODO Auto-generated method stub

            }
        });

        edittext_pass.addTextChangedListener(new TextWatcher() {

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {

                //if(!(textview.getText().toString()==null))
                    textview.setText("你正在编辑edittext_pass");
            }
            public void beforeTextChanged(CharSequence s, int start, int count,
                    int after) {
                // TODO Auto-generated method stub
            }

            public void afterTextChanged(Editable s) {
                // TODO Auto-generated method stub
            }
        });
    }
```

示例的效果图如下：



3 Button 按钮

3.1Button 类的结构

Button 类的层次关系如下：

java.lang.Object

↳ android.view.View

↳ android.widget.TextView

↳android.widget.Button

直接子类：
CompoundButton

间接子类：
CheckBox, RadioButton, ToggleButton

3.2 Button 类的方法

方法	功能描述	返回值
Button	Button 类的构造方法	Null
onKeyDown	当用户按键时，该方法调用	Boolean
onKeyUp	当用户按键弹起后，该方法被调用	Boolean
onKeyLongPress	当用户保持按键时，该方法被调用	Boolean
onKeyMultiple	当用户多次调用时，该方法被调用	Boolean
invalidateDrawable	刷新 Drawable 对象	void
scheduleDrawable	定义动画方案的下一帧	void
unscheduleDrawable	取消 scheduleDrawable 定义的动画方案	void
onPreDraw	设置视图显示，列如在视图显示之前调整滚动轴的边界	Boolean
sendAccessibilityEvent	发送事件类型指定的 AccessibilityEvent。发送请求之前，需要检查 Accessibility 是否打开	void
sendAccessibilityEventUnchecked	发送事件类型指定的 AccessibilityEvent。发送请求之前，不需要检查 Accessibility 是否打开	void
setOnKeyListener	设置按键监听	void

3.3 Button 标签的属性

由于 Button 是继承 TextView,所以 TextView 有的属性，它都能用。

属性	描述
android：layout_height	设置控件高度。可选值：fill_parent,warp_content,px
android:layout_width	设置控件宽度，可选值：fill_parent,warp_content,px
android：text	设置控件名称，可以是任意字符
android：layout_gravity	设置控件在布局中的位置，可选项：top,left,bottom,right,center_vertical,fill_vertica,fill_horizonal，center，fill 等
android：layout_weight	设置控件在布局中的比重，可选值：任意的数字
android：textColor	设置文字的颜色
android：bufferType	设置取得的文本类别，normal、spannable、editable
android：hint	设置文本为空是所显示的字符
android：textColorHighlight	设置文本被选中时，高亮显示的颜色
android：inputType	设置文本的类型，none，text，textWords 等

3.4 Button 的使用

Button 可以在 xml 中声明，也可以在代码中动态创建。

在 xml 中：

```
<Button android:orientation="vertical"
            android:layout_height="wrap_content"   android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:textColor="#0000FF"
            android:text="这是button1，可以点击" />
            <Button android:orientation="vertical"
            android:layout_height="wrap_content"   android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:textColor="#CD0000"
            android:text="这是button2，可以点击" />
            <Button android:orientation="vertical"
            android:layout_height="wrap_content"   android:id="@+id/button3"
            android:layout_width="wrap_content"
           android:textColor="#000000"
            android:text="这是 button3，可以点击" />
```

在代码中创建 Button：

```
button4 = new Button(this);
          button4.setId(100);
          button4.setText("动态创建的Button");
          button4.setTextColor(Color.GREEN);
```

创建好了 Button 后，我们就可以对其进行监听了，其中有两种方式：

一种是继承 OnClickListenner 接口：

```
public class ButtonActivity extends Activity implements OnClickListener{
button1 = (Button)findViewById(R.id.button1);
button1.setOnClickListener(this);
public void onClick(View v) {
   switch(v.getId()){
        case R.id.button1:
          .........
        break;
    }
 }
}
```

另一种方式是：

```
public class ButtonActivity extends Activity {
   button1 = (Button)findViewById(R.id.button1);
      button1.setOnClickListener(new Button.OnClickListener(){

            public void onClick(View v) {
                  // TODO Auto-generated method stub
                  ..........
```

```
            }

        });
}
```

效果图：

4 ImageButton 图片按钮

4.1 ImageButton 类的结构

ImageButton 是带有图标的按钮，它的类层次关系如下：

```
java.lang.Object
    android.view.View
        android.widget.ImageView
            android.widget.ImageButton
```

4.1 ImageButton 类常用的方法

方法	功能描述	返回值
ImageButton	构造函数	null
setAdjustViewBounds	设置是否保持高宽比，需要与 maxWidth 和 maxHeight 结合起来一起使用	Boolean
getDrawable	获取 Drawable 对象，获取成功返回 Drawable，否则返回 null	Drawable
getScaleType	获取视图的填充方式	ScaleType
setScaleType	设置视图的填充方式，包括矩阵、拉伸等七种填充方式	void
setAlpha	设置图片的透明度	void
setMaxHeight	设置按钮的最大高度	void
setMaxWidth	设置按钮的最大宽度	void
setImageURI	设置图片的地址	void
setImageResource	设置图片资源库	void

setOnTouchListener	设置事件的监听	Boolean
setColorFilter	设置颜色过滤	void

4.2 ImageButton 标签的常用属性

属性	描述
android：adjustViewBounds	设置是否保持宽高比，true 或 false
android：cropToPadding	是否截取指定区域用空白代替。单独设置无效果，需要与 scrollY 一起使用。True 或者 false
android：maxHeight	设置图片按钮的最大高度
android：maxWidth	设置图片的最大宽度
android：scaleType	设置图片的填充方式
android：src	设置图片按钮的 drawable
android：tint	设置图片为渲染颜色

4.3ImageButton 的使用

两种实现方式，在 x m l 和代码中都可以实现，但相比较而言，在 x m l 中实现更有利于代码的改动：

在 xml 中声明 ：

```
<ImageButton
 android:id="@+id/button1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:src="@drawable/p1"   //使用自己的图片
/>
<ImageButton
 android:id="@+id/button2"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:src="@android:drawable/sym_call_incoming " //使用系统自带的图片
/>
```

在代码中创建

```
magebutton3 =   new ImageButton(this);
        imagebutton3.setId(100);
        //设置自己的图片
  imagebutton3.setBackgroundDrawable(getResources().getDrawable(R.drawable.p2));
```

接下来就可以对 imagebutton 进行监听，我们这里是在通过继承 OnClickListener 接口来实现的：

```
    imagebutton1 = (ImageButton)findViewById(R.id.button1);
        imagebutton2 = (ImageButton)findViewById(R.id.button2);
        //注册监听
        imagebutton1.setOnClickListener(this);
        imagebutton2.setOnClickListener(this);
          /注册监听
        imagebutton3.setOnClickListener(this);
        //加入布局
```

```
            layout = new LinearLayout(this);
            layout.addView(imagebutton3,param);
            linnearlayout.addView(layout,param);
public void onClick(View v) {
            // TODO Auto-generated method stub
            switch(v.getId()){
            case R.id.button1:
                    textveiw.setText("你刚才点击的是ImageButton1");
                    break;
            case R.id.button2:
                    textveiw.setText("你刚才点击的是ImageButton2");
                    break;
            case 100:

                    textveiw.setText("你刚才点击的是ImageButton3");
                    if(s){

      imagebutton3.setBackgroundDrawable(getResources().getDrawable(android.R.drawable.stat_sys_vp_phone_call_on_hold));
                      s=false;
                      break;
                    }

                    imagebutton3.setBackgroundDrawable(getResources().getDrawable(R.drawable.p2));
                    s = true;
                  break;

            }
      }
```

效果如下：

5 CheckBox 多项选择

5.1 CheckBox 类的层次关系

多项选择 CheckBox 组件也被称为复选框，该组件常用于某选项的打开或者关闭。它的层次关系如下：

```
java.lang.Object
    android.view.View
      android.widget.TextView
        android.widget.Button
          android.widget.CompoundButton
            android.widget.CheckBox
```

5.2 CheckBox 常用的方法

方法	功能描述	返回值
dispatchPopulateAccessibilityEvent	在子视图创建时，分派一个辅助事件	boolean（true：完成辅助事件分发 false：没有完成辅助事件分发）
isChecked	判断组件状态是否勾选	boolean（true：被勾选，false：未被勾选）
onRestoreInstanceState	设置视图恢复以前的状态	void
performClick	执行 click 动作，该动作会触发事件监听器	boolean（true：调用事件监听器，false：没有调用事件监听器）
setButtonDrawable	根据 Drawable 对象设置组件的背景	void
setChecked	设置组件的状态	void
setOnCheckedChangeListener	设置事件监听器	void
tooggle	改变按钮当前的状态	void
onCreateDrawableState	获取文本框为空时，文本框里面的内容	CharSequence
onCreateDrawableState	为当前视图生成新的 Drawable 状态	int[]

5.3 CheckBox 的使用

下面是一个使用 ChekBox 的实例：

在 XML 中声明定义 4 个 CheckBox：

```
<TextView
     android:id="@+id/TextView1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
<CheckBox
  android:id="@+id/CheckBox1"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="@string/CheckBox1"
>
</CheckBox>
<CheckBox
  android:id="@+id/CheckBox2"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="@string/CheckBox2"
>
</CheckBox>
```

```
<CheckBox
android:id="@+id/CheckBox3"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/CheckBox3"
>
</CheckBox>
<CheckBox
android:id="@+id/CheckBox4"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/CheckBox4"
>
</CheckBox>
    <Button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="提交"
    >
     </Button>
```

在/rec/values/string.xml 中定义四个选项：

```
 <string name="CheckBox1">篮球</string>
 <string name="CheckBox2">足球</string>
 <string name="CheckBox3">乒乓球</string>
  <string name="CheckBox4">排球</string>
```

然后在 Activity 中就可以使用并监听 CheckBox 了：

```
//用来显示题目
    TextView  m_TextView1;
    //“提交按钮”
    Button          m_Button1;
    //4个多选项
    CheckBox m_CheckBox1;
    CheckBox m_CheckBox2;
    CheckBox m_CheckBox3;
    CheckBox m_CheckBox4;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        m_TextView1 = (TextView) findViewById(R.id.TextView1);
        m_Button1 = (Button) findViewById(R.id.button1);
```

```
/* 取得每个CheckBox对象 */
m_CheckBox1 = (CheckBox) findViewById(R.id.CheckBox1);
m_CheckBox2 = (CheckBox) findViewById(R.id.CheckBox2);
m_CheckBox3 = (CheckBox) findViewById(R.id.CheckBox3);
m_CheckBox4 = (CheckBox) findViewById(R.id.CheckBox4);

//对每个选项设置事件监听
m_CheckBox1.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
    {
        if(m_CheckBox1.isChecked())
        {
            DisplayToast("你选择了："+m_CheckBox1.getText());
        }
    }
});
//////////////////
m_CheckBox2.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
    {
        if(m_CheckBox2.isChecked())
        {
            DisplayToast("你选择了："+m_CheckBox2.getText());
        }
    }
});
/////////////////
m_CheckBox3.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
    {
        if(m_CheckBox3.isChecked())
        {
            DisplayToast("你选择了："+m_CheckBox3.getText());
        }
    }
});
////////////////
m_CheckBox4.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
    {
        if(m_CheckBox4.isChecked())
        {
            DisplayToast("你选择了："+m_CheckBox4.getText());
        }
```

```
            }
        });
        //对按钮设置事件监听
        m_Button1.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v)
            {
                int num = 0;
                if(m_CheckBox1.isChecked())
                {
                    num++;
                }
                if(m_CheckBox2.isChecked())
                {
                    num++;
                }
                if(m_CheckBox3.isChecked())
                {
                    num++;
                }
                if(m_CheckBox4.isChecked())
                {
                    num++;
                }
                DisplayToast("谢谢参与！你一共选择了"+num+"项！");
            }
        });
}

/* 显示Toast  */
public void DisplayToast(String str)
{
        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        //设置toast显示的位置
        toast.setGravity(Gravity.TOP, 0, 220);
        //显示该Toast
        toast.show();
        }
```

该实例的运行效果图如下所示;

6 RadioGroup、RadioButton 单项选择

6.1 类的层次关系

RadioButton 指的是一个单选按钮，它有选中和不选中两种状态，而 RadioGroup 组件也被称为单项按钮组，它可以有多个 RadioButton。一个单选按钮组只可以勾选一个按钮，当选择一个按钮时，会取消按钮组中其他已经勾选的按钮的选中状态。

RadioButton 的类层次关系如下：

java.lang.Object
    android.view.View
        android.widget.TextView
            android.widget.Button
                android.widget.CompoundButton
                android.widget.RadioButton

而 RadioGroup 类的层次关系如下：

java.lang.Object
    android.view.View
        android.view.ViewGroup
            android.widget.LinearLayout
            android.widget.RadioGroup

6.2 常用的方法

RadioButton 使用到的公共方法：

public void toggle ()

//将单选按钮更改为与当前选中状态相反的状态。

//如果这个单选按钮已经选中，这个方法将不切换单选按钮。

该方法的源码如下：

```
@Override
public void toggle() {
    // we override to prevent toggle when the radio is already
    // checked (as opposed to check boxes widgets)
    if (!isChecked()) {
        super.toggle();
    }
}
```

RadioGroup 中使用到的公共方法：

(1)public void addView (View child, int index, ViewGroup.LayoutParams params)

功能：使用指定的布局参数添加一个子视图

参数：child　所要添加的子视图

index　将要添加子视图的位置

params　所要添加的子视图的布局参数

(2)public void check (int id)

如果传递-1 作为指定的选择标识符来清除单选按钮组的勾选状态，相当于调用 clearCheck()操作

参数：id　该组中所要勾选的单选按钮的唯一标识符（id）

参见 getCheckedRadioButtonId()　clearCheck()

(3)public void clearCheck ()

功能：清除当前的选择状态，当选择状态被清除，则单选按钮组里面的所有单选按钮将取消勾选状态，getCheckedRadioButtonId()将返回 null

参见

check(int)

getCheckedRadioButtonId()

(4)public RadioGroup.LayoutParams generateLayoutParams (AttributeSet attrs)

功能：基于提供的属性集合返回一个新的布局参数集合

参数： attrs 用于生成布局参数的属性

返回值：返回一个 ViewGroup.LayoutParams 或其子类的实例

(5)public int getCheckedRadioButtonId ()

功能：返回该单选按钮组中所选择的单选按钮的标识 ID，如果没有勾选则返回-1

返回值：返回该单选按钮组中所选择的单选按钮的标识 ID

参见

check(int)

clearCheck()

(6)public void setOnCheckedChangeListener (RadioGroup.OnCheckedChangeListener listener)

功能：注册一个当该单选按钮组中的单选按钮勾选状态发生改变时所要调用的回调函数

参数：listener 当单选按钮勾选状态发生改变时所要调用的回调函数

(7)**public void setOnHierarchyChangeListener (ViewGroup.OnHierarchyChangeListener listener)**

功能：注册一个当子内容添加到该视图或者从该视图中移除时所要调用的回调函数

参数：listener 当层次结构发生改变时所要调用的回调函数

RadioGroup 中受保护的方法：

(1)protected LinearLayout.LayoutParams generateDefaultLayoutParams ()

功能：当布局为垂直方向时，将返回一个宽度为“填充父元素”(MATCH_PARENT)，高度为“包裹内容”的布局参数集合，如果为水平方向时，将返回宽度为“包裹内容”，高度为“填充父元素”的布局参数集合

(match_parent 即为 fill_parent,public static final int FILL_PARENT/MATCH_PARENT = -1)

返回值：返回一个默认的布局参数集合

（2）protected void onFinishInflate ()

功能：当视图从 XML 中加载，且相应的子视图被添加之后，调用该方法，

即使子类重写了该方法，应该确保去调用父类的方法（通常放在方法在第一句），这样才能完成相应的调用参数

返回值： 返回一个默认的布局参数集合

6.3RadioButton 和 RadioGroup 的综合使用

XML 布局中：

```
<RadioGroup
   android:id="@+id/RadioGroup01"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="vertical"
   android:layout_x="3px"
   android:layout_y="54px"
     >
     <!--        选项要在rec/values/string.xml中定义  -->
   <RadioButton
      android:id="@+id/RadioButton1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/RadioButton1"
   />
   <RadioButton
```

```
    android:id="@+id/RadioButton2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/RadioButton2"
  />
  <RadioButton
    android:id="@+id/RadioButton3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/RadioButton3"
  />
  <RadioButton
    android:id="@+id/RadioButton4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/RadioButton4"
  />
 </RadioGroup>
```

Avtivity 中：

```
/**
        * 获得TextView对象
        * 获得RadioGroup对象
        * 获得4个RadioButton对象
        */
        m_TextView = (TextView) findViewById(R.id.TextView01);
        m_RadioGroup = (RadioGroup) findViewById(R.id.RadioGroup01);
        m_Radio1 = (RadioButton) findViewById(R.id.RadioButton1);
        m_Radio2 = (RadioButton) findViewById(R.id.RadioButton2);
        m_Radio3 = (RadioButton) findViewById(R.id.RadioButton3);
        m_Radio4 = (RadioButton) findViewById(R.id.RadioButton4);

        /* 设置事件监听  */
        m_RadioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup group, int checkedId)
            {
                // TODO Auto-generated method stub
                if (checkedId == m_Radio2.getId())
                {
                    DisplayToast("正确答案：" + m_Radio2.getText() + "，恭喜你，回答正确！");
                }
                else
                {
                    DisplayToast("请注意，回答错误！");
                }
            }
        });
    }
```

```
/*  显示Toast   */
public void DisplayToast(String str)
{
        Toast toast = Toast.makeText(this, str, Toast.LENGTH_LONG);
        //设置toast显示的位置
        toast.setGravity(Gravity.TOP, 0, 220);
        //显示该Toast
        toast.show();
        }
```

示例的效果图如下：

7 Toast 提示

7.1 Toast 类的结构

Toast 是 Android 提供的“快显讯息”类，它的用途很多，使用起来非常的简单。它是直接继承 java.lang.Object 的。因此它的类层次结构如下：

```
java.lang.Object
        android.widget.Toast
```

7.2 Toast 的常量

Toast 中有两个关于 Toast 显示时间长短的常量：

常量

int　LENGTH_LONG

持续显示视图或文本提示较长时间。该时间长度可定制。

参见

setDuration(int)

int　LENGTH_SHORT

持续显示视图或文本提示较短时间。该时间长度可定制。该值为默认值。

参见

setDuration(int)

7.3 Toast 类的方法

(1)public int cancel ()

如果视图已经显示则将其关闭，还没有显示则不再显示。一般不需要调用该方法。正常情况下，视图会在超过存续期间后消失。

(2)public int getDuration ()

返回存续期间

请参阅 setDuration(int)

(3)public int getGravity ()

取得提示信息在屏幕上显示的位置。

请参阅 Gravity setGravity()

(4)public float getHorizontalMargin ()

返回横向栏外空白。

(5)public float getVerticalMargin ()

返回纵向栏外空白。

(6)public View getView ()

返回 View 对象。

请参阅 setView(View)

(7)public int getXOffset ()

返回相对于参照位置的横向偏移像素量。

```
Toast msg = Toast.makeText(Main.this, "Message", Toast.LENGTH_LONG);
msg.setGravity(Gravity.CENTER, msg.getXOffset() / 2, msg.getYOffset() / 2);
msg.show();
```

(8)public int getYOffset ()

返回相对于参照位置的纵向偏移像素量。

(9)public static Toast makeText (Context context, int resId, int duration)

生成一个从资源中取得的包含文本视图的标准 Toast 对象。

参数:

context 使用的上下文。通常是你的 Application 或 Activity 对象。

resId 要使用的字符串资源 ID，可以是已格式化文本。

duration 该信息的存续期间。值为 LENGTH_SHORT 或 LENGTH_LONG

异常:

当资源未找到时抛异常 *Resources.NotFoundException*

(10)public static Toast makeText (Context context, CharSequence text, int duration)

生成一个包含文本视图的标准 Toast 对象。

参数:

context	使用的上下文。通常是你的 Application 或 Activity 对象。
resId	要显示的文本，可以是已格式化文本。
duration	该信息的存续期间。值为 LENGTH_SHORT 或 LENGTH_LONG

(11)public void setDuration (int duration)

设置存续期间。

请参阅

LENGTH_SHORT

LENGTH_LONG

(12)public void setGravity (int gravity, int xOffset, int yOffset)

设置提示信息在屏幕上的显示位置。

（自定义 Toast 的显示位置，toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0)可以把 Toast 定位在左上角。Toast 提示的位置 xOffset:大于 0 向右移，小于 0 向左移

请参阅

Gravity

getGravity()

(13)public void setMargin (float horizontalMargin, float verticalMargin)

设置视图的栏外空白。

参数:

horizontalMargin 容器的边缘与提示信息的横向空白（与容器宽度的比）。

verticalMargin 容器的边缘与提示信息的纵向空白（与容器高度的比）。

(14)public void setText (int resId)

更新之前通过 makeText() 方法生成的 Toast 对象的文本内容。

参数: resId 为 Toast 指定的新的字符串资源 ID。

(15)public void setText (CharSequence s)

更新之前通过 makeText() 方法生成的 Toast 对象的文本内容。

参数 s 为 Toast 指定的新的文本

(16)public void setView (View view)

设置要显示的 View 。

（注意这个方法可以显示自定义的 toast 视图，可以包含图像，文字等等。是比较常用的方法。）

请参阅

getView()

(17)public void show ()

按照指定的存续期间显示提示信息。

7.4 Toast 的使用实例

我们接下来的示例要实现的是 Toast 的直接显示以及 toast 显示 view 的内容：

首先我们在 XML 布局中声明了两个 Button 按钮：

```
<Button android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Toast显示View"
/>
```

```
<Button android:id="@+id/button2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Toast直接输出"
/>
```

然后在 Activity 中：

```
Button button1=(Button)findViewById(R.id.button1);
            button1.setOnClickListener(bt1lis);
            Button button2=(Button)findViewById(R.id.button2);
            button2.setOnClickListener(bt2lis);
        }
        OnClickListener bt1lis=new OnClickListener(){

                public void onClick(View v) {
                        showToast();
                }
        };
        OnClickListener bt2lis=new OnClickListener(){
                public void onClick(View v) {
                        Toast.makeText(Z_Toast1Activity.this,"直接输出测试", Toast.LENGTH_LONG).show();
                }
        };
        public void showToast(){
         LayoutInflater li=(LayoutInflater)getSystemService(Context.LAYOUT_INFLATER_SERVICE);
         View view=li.inflate(R.layout.toast,null);
         //把布局文件toast.xml转换成一个view
         Toast toast=new Toast(this);
         toast.setView(view);
         //载入view,即显示toast.xml的内容
         TextView tv=(TextView)view.findViewById(R.id.tv1);
         tv.setText("Toast显示View内容");
         //修改TextView里的内容
         toast.setDuration(Toast.LENGTH_SHORT);
         //设置显示时间，长时间Toast.LENGTH_LONG，短时间为Toast.LENGTH_SHORT,不可以自己编辑
         toast.show();
        }
```

我们实现的效果图如下：

8 Spinner 下拉列表

8.1 Spinner 类的层次关系

Spinner 功能类似 RadioGroup，相比 RadioGroup，Spinner 提供了体验性更强的 UI 设计模式。一个 Spinner 对象包含多个子项，每个子项只有两种状态，选择或未被选中。Spinner 类的层次关系如下：

```
java.lang.Object
    android.view.View
      android.view.ViewGroup
        android.widget.AdapterView<T extends android.widget.Adapter>
            android.widget.AbsSpinner
              android.widget.Spinner
```

8.2 Spinner 类的主要方法

（1）public int getBaseline()
返回这个控件文本基线的偏移量。如果这个控件不支持基线对齐，那么方法返回-1.
返回值：返回控件基线左边边界位置，不支持时返回-1
（这个类不知道干什么用，只找到下面的代码

（2）public CharSequence getPrompt()
返回值：　当对话框弹出的时候显示的提示（即：获得弹出视图上的标题字）

（3）public void onClick(DialogInterface dialog, int which)
当点击弹出框中的项时这个方法将被调用。
参数：dialog　　点击弹出的对话框
　　which　　点击按钮(如：Button)或者点击位置

（4）public Boolean performClick()
如果它被定义就调用此视图的 OnClickListener（译者注：例如可以在加载时默认弹出下拉列表）。
返回值 ：　为 True 一个指定的 OnClickListener 被调用，为 false 时不被调用。

（5）public void setOnItemClickListener(AdapterView.OnItemClickListener　l)
Spinner 不支持 item 的点击事件，调用此方法将引发异常。

参数：l　这个监听将被忽略

（6）public void setPromptId(CharSequence prompt)

设置对话框弹出的时候显示的提示（译者注：设置弹出视图上的标题字）

参数

prompt　设置的提示

（7）public void setPromptId(int promptId)

设置对话框弹出的时候显示的提示（译者注：设置弹出视图上的标题字）

参数：prompted 当对话框显示是显示这个资源 id 所代表的提示。

受保护的方法

（8）protected void onDetachedFromWindow ()

当这个视图从屏幕上卸载时候被调用。在这一点上不再绘制视图。

（9）protected void onLayout (boolean changed, int l, int t, int r, int b)

当 View 要为所有子对象分配大小和位置时，调用此方法。派生类与子项们应该重载这个方法和调用布局每一个子项。

参数：changed　这是这个视图的一个新的大小或位置

l　相对父空间的左位置

t　相对父空间的顶端位置

r　相对父空间的右端位置

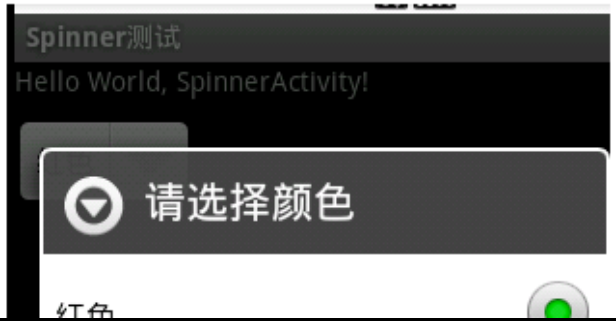
b　相对父空间的底部位置

参见

Creates and positions all views

8.3 Spinner 类的属性

XML 属性

属性名称	描述
android:prompt	该提示在下拉列表对话框显示时显示。（也就是对话框的标题： setPrompt("请选择颜色"); 

8.4 Spinner 的使用示例

首先我们在 Xml 中声明 Spinner，这里同时声明了一个 TextView 方便等下显示 Spinner 的监听结果：

```
<TextView    android:id="@+id/TextView01"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/hello"/>
<Spinner android:id="@+id/Spinner01"
      android:layout_width="300dip"
      android:layout_height="wrap_content">
</Spinner>
```

然后就可以在 Activity 中使用了：

```
final TextView textview = (TextView)findViewById(R.id.TextView01);
        Spinner spinner = (Spinner) findViewById(R.id.Spinner01);
        final List<String> list   = new ArrayList<String>();
        list .add("Spinner子项1");
        list .add("Spinner子项2");
        list .add("Spinner子项3");
        //将可选内容list与ArrayAdapter相连接
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, list );
        //设置下拉列表的风格
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
       //将Adapter添加到Spinner
        spinner.setAdapter(adapter);
```

好了，到这步以及在屏幕上可以显示出Spinner了，接下来就是添加事件监听了

```
        //添加事件监听
spinner.setOnItemSelectedListener(new Spinner.OnItemSelectedListener(){
                @Override
                public void onItemSelected(AdapterView<?> arg0, View arg1,
                        int arg2, long arg3) {
                    // TODO Auto-generated method stub
                    textview.setText("你当前选择的是：  "+list.get(arg2));
                }
                @Override
                public void onNothingSelected(AdapterView<?> arg0) {
                    // TODO Auto-generated method stub

                }
                    });
```

示例的使用效果图如下：

9 ListView 列表

9.1 ListView 类的层次关系

ListView 是用来显示一个列表的控件，以下是 ListView 类的层次关系：

java.lang.Object
  android.view.View
    android.view.ViewGroup
      android.widget.AdapterView<T extends android.widget.Adapter>
        android.widget.AbsListView
          android.widget.ListView
直接子类
        ExpandableListView （使用竖滚动条查看的两级列表视图）

9.2 ListView 的所有方法

公共方法

（1）public void addFooterView (View v)

加一个固定显示于 list 底部的视图。如果此方法被调用超过一次，所加的几个视图将按照它们加入的顺序排列。加入的视图可取得焦点。

注意：在调用 setAdapter 之前调用此方法。这样的话，可以利用点击光标来收起有 header view 和 footer view 的 ListView。

参数：v　　要加的视图

（2）public void addFooterView (View v, Object data, boolean isSelectable)

加一个固定显示于 list 底部的视图。如果此方法被调用超过一次，所加的几个视图将按照它们加入的顺序排列。加入的视图可取得焦点。

注意：在调用 setAdapter 之前调用此方法。这样的话，可以利用点击光标来收起有 header view 和 footer view 的 ListView。

参数：v　　要加的视图
　　data　　和此视图关联的数据
　　isSelectable　　设为 true 则表示 footer view 可以被选中

（3）public void addHeaderView (View v)

加一个固定显示于 list 顶部的视图。如果此方法被调用超过一次，所加的几个视图将按照它们加入的顺序排列。加入的视图可取得焦点。

注意：在调用 setAdapter 之前调用此方法。这样的话，可以利用点击光标来收起有 header view 和 footer view 的 ListView。

参数：v　　要加的视图

（4）public void addHeaderView (View v, Object data, boolean isSelectable)

加一个固定显示于 list 顶部的视图。如果此方法被调用超过一次，所加的几个视图将按照它们加入的顺序排列。加入的视图可取得焦点。

注意：在调用 setAdapter 之前调用此方法。这样的话，可以利用点击光标来收起有 header view 和 footer view 的 ListView。

参数：v　　要加的视图
　　data　　和此视图关联的数据
　　isSelectable　　表示此 header view 可选与否

（5）public void clearChoices ()

取消之前设置的任何选择

（6）public boolean dispatchKeyEvent (KeyEvent event)

按照可以获得焦点的顺序（从视图树的顶端到当前获得焦点的视图），分派一个按键事件给下一个视图。若此视图有焦点，事件将会分派给它自己。否则它将按照顺序，分派给下一个节点。此方法同时触动所有按键监听器。

参数：event　　被分派的事件

返回：若事件被处理，则返回 true；否则为 false

（7）public boolean dispatchPopulateAccessibilityEvent (AccessibilityEvent event)

在视图的子项目被构建时，分派一个辅助事件。

参数：event　　事件

返回：若事件全部完成，则返回 true

（8）public ListAdapter getAdapter ()

返回 ListView 当前用的适配器。返回的适配器不可以和传给 setAdapter(ListAdapter)的参数一样，但是可以是 WrapperListAdapter。

返回：当前用来显示 ListView 中数据的适配器
参见：setAdapter(ListAdapter)

（9）public long[] getCheckItemIds ()
此方法已经过时了。使用 getCheckedItemIds()代替。
返回被选中项目的索引集合。只有当选择模式没有被设置为 CHOICE_MODE_NONE 时才有效。

（10）public long[] getCheckedItemIds ()
返回被选中项目的索引集合。只有当选择模式没有被设置为 CHOICE_MODE_NONE，并且适配器有稳定的 ID（hasStableIds()==true） 时，结果才有效。
返回
一个新的数组，包含列表中每个被选中的索引（id）

（11）public int getCheckedItemPosition ()
返回当前被选中的项目。只有当选择模式已被设置为 CHOICE_MODE_SINGLE 时 ，结果才有效。
返回：
返回当前被选中的项目的索引；若没有项目被选中，则返回 INVALID_POSITION
参见
setChoiceMode(int)

（12）public SparseBooleanArray getCheckedItemPositions ()
返回当前被选中的项目集合。只有当选择模式没有被设置为 CHOICE_MODE_NONE 时 ，结果才有效。
返回
类型为 SparseBooleanArray 的值，其中，对每一个索引所代表的项目，若被选中，则返回 true；当选择模式被设置为 CHOICE_MODE_NONE 时 ，返回 null。

（13）public int getChoiceMode ()
返回
返回当前的选择模式
参见
setChoiceMode(int)

（14）public Drawable getDivider ()
返回
返回当前画在列表元素之间，作为分隔符的图形

（15）public int getDividerHeight ()
返回
返回分隔符的高度

（16）public int getFooterViewsCount ()
返回
列表中的页脚视图数量；缺省实现时，数量为 0

（17）public int getHeaderViewsCount ()
返回
列表中的页眉视图数量；缺省实现时，数量为 0

（18）public boolean getItemsCanFocus ()
返回
ListAdapter 所生成的视图是否可以包含能取得焦点的项目

（19）public int getMaxScrollAmount ()
返回
The maximum amount a list view will scroll in response to an arrow event.
响应箭头事件时，列表视图可以滚动的最大值。（译者注：此处翻译待改进，恐怕需要仔细查看源代码才能明白其含义，也可以用 Google Code 搜索相关的代码）

（20）public boolean isItemChecked (int position)
对于由 position 指定的项目，返回其是否被选中。只有当选择模式已被设置为 CHOICE_MODE_SINGLE 或 CHOICE_MODE_MULTIPLE 时 ，结果才有效。
参数
position 要返回选中状态的项目
返回
返回项目的选中状态；若选择模式无效，则返回 false

（21）public boolean onKeyDown (int keyCode, KeyEvent event)

KeyEvent.Callback.onKeyMultiple()的缺省实现：若视图被激活并且可以被点击，当出现 KEYCODE_DPAD_CENTER 和 KEYCODE_ENTER 代表的行为时，做点击该视图的动作。

参数

keyCode　　表示按某个按键的按键代号，参见 KeyEvent

event　　定义按键动作的按键事件对象

返回

若事件被成功处理，则返回 true；若想要下一个接收器处理该事件，则返回 false

（22）public boolean onKeyMultiple (int keyCode, int repeatCount, KeyEvent event)

KeyEvent.Callback.onKeyMultiple()的缺省实现：总是返回 false（不处理该事件）。

参数

keyCode　　表示按某个按键的按键代号，参见 KeyEvent

repeatedCount　　实现动作的次数

event　　定义按键动作的按键事件对象

返回

若事件被成功处理，则返回 true；若想要下一个接收器处理该事件，则返回 false

（23）public boolean onKeyUp (int keyCode, KeyEvent event)

KeyEvent.Callback.onKeyMultiple()的缺省实现：当出现 KEYCODE_DPAD_CENTER 和 KEYCODE_ENTER 代表的行为时，做点击该视图的动作。

参数

keyCode　　表示按某个按键的按键代号，参见 KeyEvent

event　　定义按键动作的按键事件对象

返回

若事件被成功处理，则返回 true；若想要下一个接收器处理该事件，则返回 false

（24）public void onRestoreInstanceState (Parcelable state)

重新创建并显示一个视图，此视图拥有之前 onSaveInstanceState()保存的内部状态。当 state 为 null 时，此方法不会被调用。

参数

state　　之前 onSaveInstanceState()保存的状态

（25）public Parcelable onSaveInstanceState ()

保存视图的内部状态，用于以后创建新的拥有同样状态的实例。可保存的状态只包含非持久性的，或者可重新组建的信息。比如，永远不可能保存你当前在屏幕上的位置，因为当新的实例被放置于视图层次体系中时，位置会被重新计算。

一些可以被保存的状态：文本视图（但是通常不是指文本本身，因为文本是被保存在内容提供商或其他持久性的储存体中）中当前的光标位置；列表视图中当前的选中项。

返回

返回一个包含视图当前动态状态的接口方法对象；若没有东西被保存，则返回 null。缺省情况下返回 null。

（26）public boolean onTouchEvent (MotionEvent ev)

此方法用于处理触摸屏的动作事件。

参数

ev　　动作事件

返回

若事件被成功处理，则返回 true；否则返回 false

（27）public boolean performItemClick (View view, int position, long id)

调用定义好的 OnItemClickListener。

参数

view　　AdapterView 中被点击到的视图

position　　视图在适配器中的索引

id　　被点击到的项目的行 id

返回

若有定义好的 OnItemClickListener 被成功调用，则返回 true；否则返回 false

（28）public boolean removeFooterView (View v)

删除之前加入的某个页脚视图。

参数

v　　要删除的视图

返回

若视图被成功删除，则返回 true；若此视图不是页脚视图，则返回 false

（29）public boolean removeHeaderView (View v)

删除之前加入的某个页眉视图。

参数

v　　要删除的视图

返回

若视图被成功删除，则返回 true；若此视图不是页眉视图，则返回 false

（30）public boolean requestChildRectangleOnScreen (View child, Rect rect, boolean immediate)

当组里的某个子项需要被定位在屏幕的某个矩形范围时，调用此方法。

重载此方法的 ViewGroup 可确认以下几点：

· 子项目将是组里的直系子项

· 矩形将在子项目的坐标体系中

重载此方法的 ViewGroup 必须保证以下几点：

· 若矩形已经是可见的，则没有东西会改变

· 为使矩形区域全部可见，视图将可以被滚动显示

参数

child 发出请求的子项目

rect 子项目坐标系内的矩形，即此子项目希望在屏幕上的定位

immediate 设为 true，则禁止动画和缓释移动滚动条

返回

这个可滚动显示的组，是否接受请求

（31）public void setAdapter (ListAdapter adapter)

设置 ListView 背后的数据。根据 ListView 目前使用的特性，adapter 可能被 WrapperListAdapter 收起。例如：加页眉和/或页脚会使 adapter 被收起。

参数

adapter 负责维护列表背后的数据，以及生成视图来显示数据里的项目

参见

getAdapter()

（32）public void setCacheColorHint (int color)

当 color 的值不为 0 时，此值表示的颜色将提示使用者，列表正在一片单色不透明的背景上被画出。

参数

color 背景色

（３３）public void setChoiceMode (int choiceMode)

设置 List 的选择模式。缺省情况下，列表没有选择模式（即值为 CHOICE_MODE_NONE）。

参数

choiceMode 值可为 CHOICE_MODE_NONE，CHOICE_MODE_NONE 和 CHOICE_MODE_NONE 中的一种

（３４）public void setDivider (Drawable divider)

设置将画在列表中每个项目之间的图形。如果图形没有已设定好的高度，则必须同时调用 setDividerHeight(int)。

参数

divider 将用作分隔符的图形

（３５）public void setDividerHeight (int height)

设置分隔符（画在列表中每个项目之间）的高度。调用此方法将覆盖由 setDivider(Drawable)设置的高度。

参数

height 分隔符的新高度，单位为像素

（３６）public void setFooterDividersEnabled (boolean footerDividersEnabled)

设置可以或者不可以为页脚视图画上分隔符。

参数

headerDividersEnabled 设为 true，表明可以画；设为 false 则不可以

参见

setHeaderDividerEnabled(boolean)

addFooterView(android.view.View)

（３７）public void setHeaderDividersEnabled (boolean headerDividersEnabled)

设置可以或者不可以为页眉视图画上分隔符。

参数

headerDividersEnabled 设为 true，表明可以画；设为 false 则不可以

参见

setFooterDividerEnabled(boolean)

addHeaderView(android.view.View)

（３８）public void setItemChecked (int position, boolean value)

设置 position 所指定项目的选择状态。只有选择模式为 CHOICE_MODE_SINGLE 或者 CHOICE_MODE_MULTIPLE 时，此设置才有效。

参数

position 需要改变选择状态的项目的索引

value　　新的选择状态

（３９）public void setItemsCanFocus (boolean itemsCanFocus)
表明在由 ListAdapter 创建的视图中，可包含能获得焦点的项目。
参数
itemsCanFocus　　若项目能获得焦点，则设为 true；否则为 false

（４０）public void setSelection (int position)
选中 position 指定的项目。若为触摸模式，则指定项目不会被选中，但位置变化一样。若 position 的值小于 0，则 position 为 0 的项目将被选中。
参数
position　　需要选中的项目的索引（从 0 开始）

（４１）public void setSelectionAfterHeaderView ()
选中页眉视图下的第一个列表项目。

（４２）public void setSelectionFromTop (int position, int y)
选中 position 指定的项目，并将所选项置于距离 ListView 顶端 y 像素的位置（若为触摸模式，则指定项目不会被选中，但位置变化一样）。
参数
position　　需要选中的项目的索引（从 0 开始）
y　　距离 ListView（包括间隙）顶端的位置

受保护方法
（４３） protected boolean canAnimate ()
表示此视图组是否可以在第一次被布局后，仍可以动态调整其子项。
返回
若可以则为 true，否则为 false

（４４）protected void dispatchDraw (Canvas canvas)
调用此方法来绘出子视图。可被衍生类重写，以便在其子项被画出之前取得控制权。
参数
canvas　　绘出 View 所用的 canvas（画布？）

（４５）protected View findViewTraversal (int id)
参数
id　　要找的 View 的 id
返回值
有此 id 的 View，若没有找到则为 null

（４６）protected View findViewWithTagTraversal (Object tag)
参数
tag　　要找的 View 的标签
返回值
有此标签的 View，若没有找到则为 null

（４７）protected void layoutChildren ()
子类必须重写此方法来布局其子项。

（４８）protected void onFinishInflate ()
当 View 以及所有子项从 XML 中导入时被调用，是导入的最后一步。即使子类重写 onFinishInflate，也必须保证有调用超方法，这样，方法才会被调用。

（４９）protected void onFocusChanged (boolean gainFocus, int direction, Rect previouslyFocusedRect)
当 View 的焦点改变时被调用。重写时，要确保超类的直接调用，这样取得焦点的方式才是标准的。
参数
gainFocus　　若 View 有焦点，则为 True；否则为 False。
direction　　当 requestFocus()被调用时，方向焦点被移动。其值可为 FOCUS_UP，FOCUS_DOWN，FOCUS_LEFT 或 FOCUS_RIGHT。在使用缺省条件的情况下，direction 并不总是可用。
previouslyFocusedRect　　之前得到焦点的 View 的坐标系统所构成的矩形。如果可用，这个将被当成精确信息（表明焦点从何而来以及从何方向而来）来传递；否则将传递 null。

（５０）protected void onMeasure (int widthMeasureSpec, int heightMeasureSpec)
View 调用此方法来确定本身和所包含内容的大小。此方法被 measure(int,int)唤起，而且必须被子类重写以得到所包含内容的确切大小。
注意：当重写此方法时，必须调用 setMeasureDimension(int,int)来保存 View 的大小。如果没有做到，将会引发一个 measure(int,int)抛出的 IllegalStateException（非法状态错误）。超类 onMeasure(int,int)可以被调用。

编写基类的确认大小的方法，缺省情况下是根据其背景大小来确认，除非 MeasureSepc 允许有更大的高度或宽度。子类必须重写 onMeasure(int,int)以得到对其内容大小的更准确的测量。

若此方法被重写，它的子类需要确保其高度和宽度至少达到 View 所规定的最小值（可通过 getSuggestedMinimumHeight()和 getSuggestedMinimumWidth()得到）。

参数

widthMeaureSpec　　受上一层大小影响下的对水平空间的要求。可参看 View.MeasureSpec。
heightMeasureSpec　　受上一层大小影响下的对垂直空间的要求。可参看 View.MeasureSpec。

protected void onSizeChanged (int w, int h, int oldw, int oldh)

当 VIew 的大小改变时此方法被调用。如果 VIew 是刚刚被加入，则视之前的值为 0。

参数

w　　View 的当前宽度
h　　View 的当前高度
oldw　　View 大小改变之前的宽度
oldh　　View 大小改变之前的高度

9.3 ListView 的属性及包含的几个常量

XML 属性

属性名称	描述
android:choiceMode	规定此 ListView 所使用的选择模式。缺省状态下，list 没有选择模式。属性值必须设置为下列常量之一： none，值为 0，表示无选择模式；singleChoice，值为 1，表示最多可以有一项被选中；multipleChoice，值为 2，表示可以多项被选中。可参看全局属性资源符号 choiceMode。
android:divider	规定 List 项目之间用某个图形或颜色来分隔。可以用"@[+][package:]type:name"或者"?[package:][type:]name"（主题属性）的形式来指向某个已有资源；也可以用"#rgb"，"#argb"，"#rrggbb"或者"#aarrggbb"的格式来表示某个颜色。可参看全局属性资源符号 divider。
android:dividerHeight	分隔符的高度。若没有指明高度，则用此分隔符固有的高度。必须为带单位的浮点数，如"14.5sp"。可用的单位如 px（pixel 像素），dp（density-independent pixels 与密集度无关的像素）， sp（scaled pixels based on preferred font size 基于字体大小的固定比例的像素）， in (inches 英寸), mm (millimeters 毫米)。可以用"@[package:]type:name "或者"?[package:][type:]name"（主题属性）的格式来指向某个包含此类型值的资源。可参看全局属性资源符号 dividerHeight。
android:entries	引用一个将使用在此 ListView 里的数组。若数组是固定的，使用此属性将比在程序中写入更为简单。必须以"@[+][package:]type:name"或者 "?[package:][type:]name"的形式来指向某个资源。可参看全局属性资源符号 entries。
android:footerDividersEnabled	设成 flase 时，此 ListView 将不会在页脚视图前画分隔符。此属性缺省值为 true。属性值必须设置为 true 或 false。可以用"@[package:]type:name "或者"?[package:][type:]name"（主题属性）的格式来指向某个包含此类型值的资源。可参看全局属性资源符号 footerDividersEnabled。
android:headerDividersEnabled	设成 flase 时，此 ListView 将不会在页眉视图后画分隔符。此属性缺省值为 true。

	属性值必须设置为 true 或 false。 可以用"@[package:]type:name "或者"?[package:][type:]name"（主题属性）的格式来指向某个包含此类型值的资源。 可参看全局属性资源符号 headerDividersEnabled。

常量：

Int CHOICE_MODE_MULTIPLE

（常量值为 2）列表允许同时选取多项

Int CHOICE_MODE_NONE

（常量值为 0）普通列表，不指明选取模式

Int CHOICE_MODE_SINGLE

（常量值为 1）列表只允许选取最多一项

9.4 ListView 的使用示例

首先在ＸＭＬ中声明一个 ListView，一个 TextView 用来显示 ListView 的监听结果：

```
<TextView    android:id="@+id/TextView01"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/hello" />
      <ListView android:id="@+id/ListView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
      </ListView>
```

然后在 Avticity 中：

```
final TextView textView = (TextView)findViewById(R.id.TextView01);
            ListView listView = (ListView)findViewById(R.id.ListView01);
            //创建数组list，用于存放要显示的内容
            final List<String> list    = new ArrayList<String>();
            list.add("ListView子项1");
            list.add("ListView子项2");
            list.add("ListView子项3");
            //将数据与适配器adapter连接
            ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, list );
           //将adapter添加到ListView
            listView.setAdapter(adapter);

            //实现ListView的监听
            listView.setOnItemClickListener(new AdapterView.OnItemClickListener(){

                  @Override
                  public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                              long arg3) {
                        // TODO Auto-generated method stub
                        textView.setText("你当前选择的是"+list.get(arg2).toString());
                  }

            });
```

示例的效果图如下：

9.5 关于 ListView 使用的参考链接

Android 入门第六篇之 ListView (一)
android ListView 详解
android 异步加载 ListView 中的图片
Google I/O 2010 - The world of ListView
Android: 显示 SD 卡文件列表
Android: 带图标的 ListView 实现
ListView 和 getView 的原理＋如何在 ListView 中放置多个 item

10 TabHost 标签页

10.1 TabHost 类的层次关系

继承关系

public class TabHost extends FrameLayout implements ViewTreeObserver.OnTouchModeChangeListener

java.lang.Object
android.view.View
android.view.ViewGroup
android.widget.FrameLayout
android.widget.TabHost

10.2 TabHost 常用的方法

内部类

interface TabHost.OnTabChangeListener
接口定义了当选项卡更改时被调用的回调函数

interface TabHost.TabContentFactory
当某一选项卡被选中时生成选项卡的内容

class TabHost.TabSpec
单独的选项卡，每个选项卡都有一个选项卡指示符,内容和 tag 标签,以便于记录.

公共方法

（1）public void **addTab** (TabHost.TabSpec tabSpec)
新增一个选项卡
参数
tabSpec　指定怎样创建指示符和内容.

（2）public void **clearAllTabs** ()
从 tab widget 中移除所有关联到当前 tab host 的选项卡

（3）public boolean **dispatchKeyEvent** (KeyEvent event)
分发按键事件到焦点传递路线上的下一视图。焦点传递路线从视图树的顶层开始一直到当前获取焦点的视图停止。如果此视图已经获取焦点,将分发给它自身。否则，将分发到焦点传递路线的下一节点。此方法会触发任何一个按键监听器.
（译者注：关于 focus path,可以参考以下地址:
http://blog.csdn.net/maxleng/archive/2010/05/04/5557758.aspx)
参数
event 分发的按键事件
返回值
如果事件已经处理则返回 true,否则返回 false.

（4）public void **dispatchWindowFocusChanged** (boolean hasFocus)
当窗口包含的此视图获取或丢失焦点时触发此方法.ViewGroups 应该重写以路由到他的子元素
参数
hasFocus 如果窗口包含的此 view 依获取焦点,返回 true,否则返回 false.

（5）public int **getCurrentTab** ()
（译者注：获取当前选项卡的 id）

（6）public String **getCurrentTabTag** ()
（译者注：当前选项卡的 Tag 标签内容）

（7）public View **getCurrentTabView** ()
（译者注：获取当前选项卡的视图 view）

（8）public View **getCurrentView** ()
（译者注：获取当前的视图 view）

（9）public FrameLayout **getTabContentView** ()
获取保存 tab 内容的 FrameLayout

（10）public TabWidget **getTabWidget** ()
（译者注：根据系统规定的 id：tabs 来找到 TabWidget，并返回，注意，这里的 ID 必须是 tabs。源代码中表示如下：
private TabWidget mTabWidget;
mTabWidget=(TabWidget)findViewById(com.android.internal.R.id.tabs);）

（11）public TabHost.TabSpec **newTabSpec** (String tag)
获取一个新的 TabHost.TabSpec，并关联到当前 tab host
参数
tag 所需的选项卡标签(tag)

（12）public void **onTouchModeChanged** (boolean isInTouchMode)
当触摸模式发生改变时调用的回调函数.
参数
isInTouchMode 如果视图结构当前处于触摸模式,返回 true,否则返回 false.

（13）public void **setCurrentTab** (int index)
（译者注：设置当前的选项卡
参数
Index 为当前选项卡的索引。）

（14）public void **setCurrentTabByTag** (String tag)
（译者注：根据选项卡的 Tab 标签来设置当前的选项卡
参数
tag 想要被设置为当前选项卡的 tag 标签值。）

(15)public void **setOnTabChangedListener**(TabHost.OnTabChangeListener l)
注册一个回调函数，当选项卡中的任何一个 tab 的选中状态发生改变时调用。
（译者注：setCurrentTab(index)时会触发调用）
参数
l 将运行的回调函数

（16）public void **setup** ()
如果使用 findViewById()加载 TabHost，那么在新增一个选项卡 tab 之前，需要调用 setup()。然而，当你在一个 TabActivity 里使用 getTabHost()获取 TabHost，你就不再需要调用 setup()了。（译者注：实现 tab 窗口的两种方法：继承 activity 时，

使用 findViewById()查找 TabHost，然后调用 setup()；继承 TabActivity，通过 getTabHost()查找，此时不用调用 setup()）例子：

```
mTabHost = (TabHost)findViewById(R.id.tabhost);
mTabHost.setup();
mTabHost.addTab(TAB_TAG_1, "Hello, world!", "Tab 1");
```

（17）public void **setup** (LocalActivityManager activityGroup)

如果你使用 setContent(android.content.Intent)，那么当 activityGroup 用于加载本地 activity 之时，必须调用此方法。如果你拓展（继承）TabActivity 将自动调用 setup()方法。

参数

activityGroup　用来为选项卡内容加载 activities 的 ativityGroup

受保护方法

protected void **onAttachedToWindow** ()

当视图附加到窗口上时被调用。在这个点的表面进行绘制。注意此函数确保在 onDraw(Cancas)之前调用，然而它可能在第一次执行 onDraw 之前的任何时间被调用——包括的 onMeasure(int,int)的之前或之后。

protected void onDetachedFromWindow ()

当视图从窗口分离时被调用。在这个点的表面不再有画面绘制。

10.3 TabHost 的使用示例

我们这里使用 3 个 XML 来实现

tab1.xml 文件代码：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:id = "@+id/layout01"
  ……
  ……
</LinearLayout>
```

tab2.xml 文件代码：

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout android:id="@+id/layout02"
  ……
  ……
</AbsoluteLayout>
```

tab3.xml 文件代码：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout android:id="@+id/layout03"
  ……
……
</RelativeLayout>
```

注意：在 Activity 中我们继承的是 TabActivity：

```
public class TabDemo extends TabActivity {
TabHost tabHost = getTabHost();
    LayoutInflater.from(this).inflate(R.layout.tab1,
            tabHost.getTabContentView(),true);
    LayoutInflater.from(this).inflate(R.layout.tab2,
            tabHost.getTabContentView(),true);
    LayoutInflater.from(this).inflate(R.layout.tab3,
            tabHost.getTabContentView(),true);
    tabHost.addTab(tabHost.newTabSpec("TAB1")
```

```
        .setIndicator("线性布局").setContent(R.id.layout01));
    tabHost.addTab(tabHost.newTabSpec("TAB2")
        .setIndicator("绝对布局").setContent(R.id.layout02));
    tabHost.addTab(tabHost.newTabSpec("TAB3")
        .setIndicator("相对布局").setContent(R.id.layout03));
```

10.4 补充

补充

文章链接

[史上最全的 Android 的 Tab 与 TabHost 讲解]
[Android UI 设计 Tab TabHost 标签页的使用]
[Android 控件之 TabHost Tab 页]
[动态 Tab 页]

11 LinearLayout 线性布局

11.1 线性布局介绍

LinearLayout 是一种线性排列的布局，在线性布局中，所有的子元素都按照垂直或水平的顺序在界面上排列：
如果垂直排列，则每行仅包含一个界面元素，如果水平排列，则每列仅包含一个界面元素。可以通过属性 android：orientation 定义布局中子元素的排列方式。android.widget.LinearLayout 类是 android.view.viewGroup 的子类，其中又派生了 RadioGroup、TabWidget、TableLayout、TableRow、ZoomControls 等类。

11.2 线性布局的常用属性

Android:id 为控件指定相应的 ID

Android:text 指定控件当中显示的文字，需要注意的是，这里尽量使用 strings.xml 文件当中的字符

Android:grivity 指定控件的基本位置，比如说居中，居右等位置 Android:textSize 指定控件当中字体的大小
Android:background 指定该控件所使用的背景色,RGB 命名法
Android:width 指定控件的宽度
Android:height 指定控件的高度
Android:padding* 指定控件的内边距，也就是说控件当中的内容

Android:sigleLine　如果设置为真的话，则将控件的内容在同一行当中进行显示

11.3 Linear 常用的方法

方法	功能描述	返回值
LinearLayout	两个构造函数： LinearLayout（Contex context） LinearLayout（Contex contex，AttributeSet attrs）	null
getOrientation()	获取布局的方向设置。0 代表水平方向，1 代表垂直方向	int
isBaselineAligned()	判断布局是否按照基线对齐	boolean
setBaselineAligned(boolen baselineAligned)	根据参数设置基线对齐	void
setGravity(int gravity)	根据指定的重力设置元素的大小	void
setHorizontalGravity(int gravity)	设置水平方向的重力	void
setVerticalGravity(int gravity)	设置垂直方向的重力	void
gennerateDefaultLayoutParams	返回包含宽度和高度的布局参数的集合	LayoutParams
setGravity(int gravity)	设置布局的重力	void

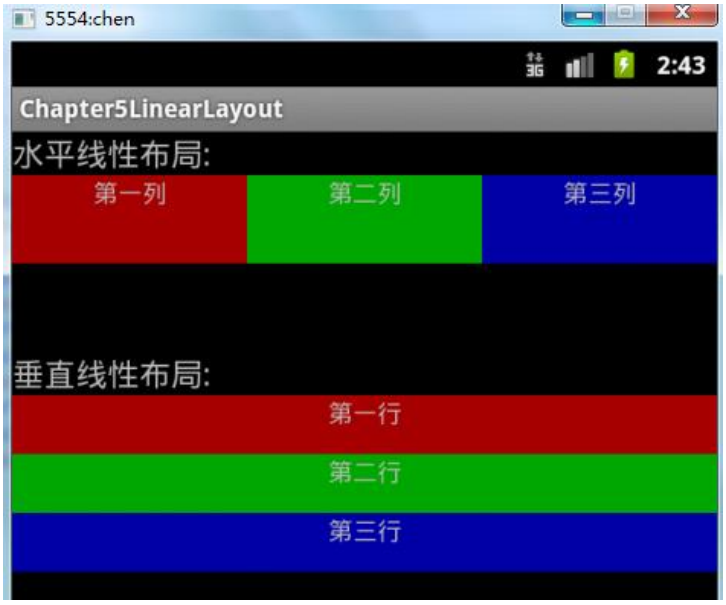
11.4 LinearLayout 的使用

在 XML 中设置如下：

```
<!-- 水平线性布局 -->
    <LinearLayout android:layout_height="100dip"
        android:orientation="horizontal" android:layout_width="fill_parent">
        <TextView android:background="#aa0000" android:layout_width="wrap_content"
            android:gravity="center_horizontal" android:text="第一列"
            android:layout_weight="1" android:layout_height="50dip"></TextView>
        <TextView android:background="#00aa00" android:layout_width="wrap_content"
            android:gravity="center_horizontal" android:text="第二列"
            android:layout_weight="1" android:layout_height="50dip"></TextView>
        <TextView android:background="#0000aa" android:layout_width="wrap_content"
            android:gravity="center_horizontal" android:text="第三列"
            android:layout_weight="1" android:layout_height="50dip"></TextView>
    </LinearLayout>

<!-- 垂直线性布局 -->
<LinearLayout
     android:layout_height="100dip"
        android:orientation="vertical" android:layout_width="fill_parent">
        <TextView android:background="#aa0000" android:layout_width="fill_parent"
            android:gravity="center_horizontal" android:text="第一行"
            android:layout_weight="1" android:layout_height="50dip"></TextView>
        <TextView android:background="#00aa00" android:layout_width="fill_parent"
            android:gravity="center_horizontal" android:text="第二行"
```

```
            android:layout_weight="1" android:layout_height="50dip"></TextView>
        <TextView android:background="#0000aa" android:layout_width="fill_parent"
            android:gravity="center_horizontal" android:text="第三行"
            android:layout_weight="1" android:layout_height="50dip"></TextView>
    ></LinearLayout>
```

然后在就可以直接显示：



12 FrameLayout 框架布局

12.1 FrameLayout 的简单介绍

FrameLayout 是布局中最简单的一个布局，在这个布局中，整个界面被当成一块空白备用区域，所有的子元素都不能被指定放置的位置，它们统统放于这块区域的左上角，并且后面的子元素直接覆盖在前面的子元素之上，将前面的子元素部分和全部遮挡。

12.2 FrameLayout 的使用

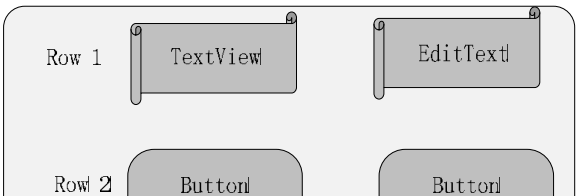
FrameLayout 的使用很简单，在 XML 中的声明为 FrameLayout 就可以：

```
<?xml version="1.0" encoding="utf-8"?>

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="50dip"
    android:textColor="#ffffff"
    android:text="第一层"/>
<TextView

    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="40dip"
    android:textColor="#ffff00"
    android:text="第二层"/>
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="30dip"
    android:textColor="#ff00ff"
    android:text="第三层"/>
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="20dip"
    android:textColor="#00ffff"
    android:text="第四层"/>
</FrameLayout>
```

13 TableLayout 表格布局

13.1 TableLayout 的层次关系

表格布局（TableLayout）也是一种常用的界面布局，它将屏幕划分网格，通过指定行和列可以将界面元素添加的网格中。



TableLayout 事实上是 LinearLayout 的子类，它的层次关系如下：

java.lang.Object
 android.view.View
  android.widget.ViewGroup
   android.widget.LinearLayout
    android.widget.TableLayout

13.2 TableLayout 常用的方法

构造函数

public TableLayout (Context context)
为给定的上下文创建表格布局。
参数
context 应用程序上下文

public TableLayout (Context context, AttributeSet attrs)
使用指定的属性集合为给定的上下文创建表格布局。
参数
context 应用程序上下文
attrs 属性集合

公共方法

public void **addView** (View child)
添加子视图。如果子视图没有设置布局参数，则使用视图组(ViewGroup)的布局参数为该视图布局。
参数
child 添加的子视图

public void **addView** (View child, int index)
添加子视图。如果子视图没有设置布局参数，则使用视图组(ViewGroup)的布局参数为该视图布局。
参数
child 添加的子视图
index 子视图加入的位置索引

public void **addView** (View child, int index, ViewGroup.LayoutParams params)
用指定的布局参数添加一个子视图。
参数
child 添加的子视图
index 子视图加入的位置索引
params 为子视图指定得布局参数

public void addView (View child, ViewGroup.LayoutParams params)
使用指定的布局参数添加子视图。
参数
child 添加的子视图
params 设置到子视图上的布局参数

public TableLayout.LayoutParams **generateLayoutParams** (AttributeSet attrs)
返回一组基于提供的属性集合的布局参数集合。
参数
attrs 用于生成布局参数的属性集
返回值
ViewGroup.LayoutParams 或其子类的实例

public boolean **isColumnCollapsed** (int columnIndex)
返回指定列的折叠状态。

参数

columnIndex 列索引

返回值

折叠时为 true；否则为 false

public boolean **isColumnShrinkable** (int columnIndex)
返回指定的列是否可收缩。

参数

columnIndex 列索引

返回值

如果列可以收缩，返回 true；否则返回 false

public boolean **isColumnStretchable** (int columnIndex)
返回指定的列是否可拉伸。

参数

columnIndex 列索引

返回值

如果列可以拉伸，返回 true；否则返回 false

public boolean **isShrinkAllColumns** ()
指示是否所有的列都是可收缩的。

返回值

如果所有列都可收缩，返回 true；否则返回 false

public boolean **isStretchAllColumns** ()
指示是否所有的列都是可拉伸的。

返回值

如果所有列都可拉伸，返回 true；否则返回 false

public void **requestLayout** ()
当某些变更导致视图的布局失效时调用该方法。该方法按照视图树的顺序调用。

public void **setColumnCollapsed** (int columnIndex, boolean isCollapsed)
折叠或恢复给定列。折叠时，列从屏幕上消失，其空间由其它列占用。 当列属于 TableRow 时才可以进行折叠/恢复操作。
调用该方法会请求布局操作。

相关 XML 属性

android:collapseColumns

参数

columnIndex 列索引

isCollapsed 折叠时为 true；否则为 false

public void **setColumnShrinkable** (int columnIndex, boolean isShrinkable)
设置指定列是否可收缩。当行太宽时，表格可以收缩该列以提供更多空间。
调用该方法会请求布局操作。

相关 XML 属性

android:shrinkColumns

参数

columnIndex 列索引

isShrinkable 如果列可以收缩，设为真；否则设为假。默认是假。

public void **setColumnStretchable** (int columnIndex, boolean isStretchable)
设置指定列是否可拉伸。可拉伸时，列会尽可能多的占用行中的可用空间。
调用该方法会请求布局操作。

相关 XML 属性

android:stretchColumns

参数

columnIndex 列索引

isStretchable 如果列可以拉伸，设为真；否则设为假.默认是假

public void **setOnHierarchyChangeListener** (ViewGroup.OnHierarchyChangeListener listener)
注册当从视图中添加或移除子视图时发生的回调函数。

参数

listener 层次结构变更时执行的回调函数

public void **setShrinkAllColumns** (boolean shrinkAllColumns)
标记所有列为可收缩的便利的方法。

相关 XML 属性
android:shrinkColumns
参数
shrinkAllColumns 如果标记所有列为可收缩时为 true

public void **setStretchAllColumns** (boolean stretchAllColumns)
标记所有列为可拉伸的便利的方法。
相关 XML 属性
android:stretchColumns
参数
stretchAllColumns 如果标记所有列为可拉伸时为 true

受保护方法

protected boolean **checkLayoutParams** (ViewGroup.LayoutParams p)
（译者注：检测是不是 AbsoluteLayout.LayoutParams 的实例）

protected LinearLayout.LayoutParams **generateDefaultLayoutParams** ()
返回宽度为 MATCH_PARENT， 高度为 WRAP_CONTENT 的布局参数集合。
返回值
默认布局参数集合或空

protected LinearLayout.LayoutParams **generateLayoutParams** (ViewGroup.LayoutParams p)
基于提供的布局参数返回一组安全的布局参数集合.当传入 ViewGroup 的视图的参数没有通过 checkLayoutParams(android.view.ViewGroup.LayoutParams) 的检测时，调用该方法. 该方法会返回适合 ViewGroup 的新的布局参数，可能从指定的布局参数中复制适当的属性。
参数
p 要转换为适合于 ViewGroup 的布局参数的集合
返回值
ViewGroup.LayoutParams 或其子类的实例

protected void **onLayout** (boolean changed, int l, int t, int r, int b)
该视图设置其子视图的大小及位置时调用.派生类可以重写此方法，并为其子类布局。
参数
changed 是否为视图设置了新的大小和位置
l 相对于父视图的左侧的位置
t 相对于父视图的顶部的位置
r 相对于父视图的右侧的位置
b 相对于父视图的底部的位置

protected void **onMeasure** (int widthMeasureSpec, int heightMeasureSpec)
评估视图及其内容，以决定其宽度和高度.此方法由 measure(int, int) 调用，子类可以重载以提供更精确、更有效率的衡量其内容尺寸的方法。
约定： 覆盖该方法时，*必须*调用 setMeasuredDimension(int, int) 方法来保存评估结果的视图的宽度和高度.如果忘记将导致 measure(int, int) 方法抛出 IllegalStateException 异常.要有效的利用父类的 onMeasure(int, int)方法。
基类测量的是背景的大小，除非 MeasureSpec 允许超过背景.子类应该重写 onMeasure(int, int) 方法，以为其内容提供更适合的大小。
如果重写了该方法，子类要确保其高度和宽度大于等于视图的最小高度和宽度. （getSuggestedMinimumHeight() 和 getSuggestedMinimumWidth()）
参数
widthMeasureSpec 父视图要求的横向空间大小.该要求由 View.MeasureSpec 进行了编码处理。
heightMeasureSpec 父视图要求的纵向空间大小.该要求由 View.MeasureSpec 进行了编码处理。

13.3 TableLayout 常用的属性

android:collapseColumns：以第 0 行为序，隐藏指定的列：

android:collapseColumns 该属性为空时，如下图：

把 android:collapseColumns=0,2----------》意思是把第 0 和第 2 列去掉，如下图：

android:shrinkColumns：以第 0 行为序，自动延伸指定的列填充可用部分：

当 LayoutRow 里面的控件还没有布满布局时，shrinkColumns 不起作用，如下图：

设置了 shrinkColumns=0,1,2，布局完全没有改变，因为 LayoutRow 里面还剩足够的空间。

当 LayoutRow 布满控件时，如下图：

设置设置了 shrinkColumns=2，则结果如下图，控件自动向垂直方向填充空间：

android:stretchColumns：以第 0 行为序，尽量把指定的列填充空白部分：

设置 stretchColumns=1，则结果如下图，第 1 列被尽量填充(Button02 与 TextView02 同时向右填充,直到 TextView03 被压挤到最后边)。

13.4 TableLayout 的使用

如果现在我们要实现下图所示的布局效果 ：

我们要在 XML 文件中作如下的相关声明和设置：

```
<TableLayout android:id="@+id/TableLayout01"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <TableRow android:id="@+id/TableRow01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/label"
                android:layout_height="wrap_content"
                android:layout_width="160dip"
                android:gravity="right"
                android:text="用户名："
                android:padding="3dip" >
        </TextView>
      <EditText android:id="@+id/entry"
                android:layout_height="wrap_content"
                android:layout_width="160dip“
                android:padding="3dip" >
        </EditText>
 </TableRow>

 <TableRow android:id="@+id/TableRow02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <Button android:id="@+id/ok"
                android:layout_height="wrap_content"
                android:padding="3dip"
                android:text="确认">
        </Button>
      <Button android:id="@+id/Button02"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="3dip"
```

```
            android:text="取消">
        </Button>
    </TableRow>
      </TableLayout>
```

14 RelativeLayout 相对布局

14.1 RelativeLayout 类的层次关系

相对布局（RelativeLayout）是一种非常灵活的布局方式，能够通过指定界面元素与其他元素的相对位置关系，确定界面中所有元素的布局位置

特点：能够最大程度保证在各种屏幕类型的手机上正确显示界面布局。它的层次关系如下所示：

```
java.lang.Object
 android.view.View
    android.widget.ViewGroup
       android.widget.RelativeLayout
```

14.2 RelativeLayout 常用的方法

方法	功能描述	返回值
RelativeLayout	两个构造函数： RelativeLayout(Contex contex) RelativeLayout(Contex contex,AttributeSet attrs	null
checkLayoutParams(viewGroup.LayoutParams p	检查参数指定的布局参数是否是 LayoutParams 实例	boolen
isBaselineAligned（）	判断布局是否按照基线对齐	boolean
set BaselineAligned（boolean baseelineAligned）	根据参数设置基线对齐	void
setGravityt （int gravity）	根据指定的重力设置元素大小	void
setHorizontalGravity(int gravity)	设置水平方向的重力	void
setVerticalGravity(int gravity)	设置垂直方向的重力	void
gennerateDefaultLayoutParams	生成默认的布局参数实例	ViewGroup.LayoutParams

14.3 RelativeLayout 常用的属性

android:layout_above 将该控件的底部至于给定 ID 的控件之上

android:layout_below 将该控件的顶部至于给定 ID 的控件之下

android:layout_toLeftOf 将该控件的右边缘和给定 ID 的控件的左边缘对齐

android:layout_toRightOf 将该控件的左边缘和给定 ID 的控件的右边缘对齐 android:layout_alignBaseline 该控件的 baseline 和给定 ID 的控件的 baseline 对齐

android:layout_alignBottom 将该控件的底部边缘与给定 ID 控件的底部边缘

android:layout_alignLeft 将该控件的左边缘与给定 ID 控件的左边缘对齐

android:layout_alignRight 将该控件的右边缘与给定 ID 控件的右边缘对齐

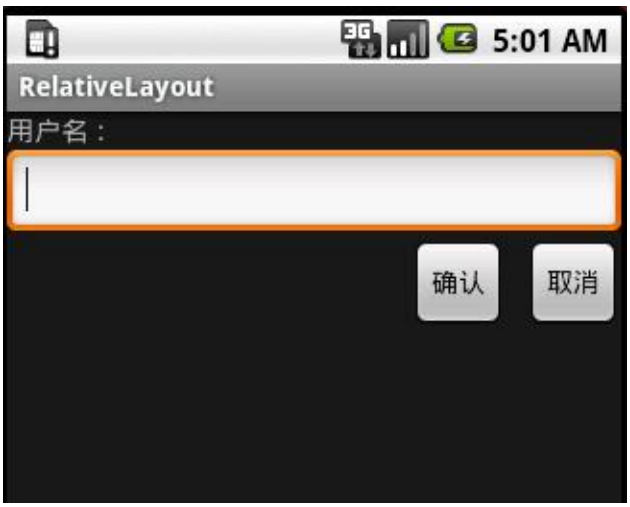
android:layout_alignTop 将给定控件的顶部边缘与给定 ID 控件的顶部对齐

android:alignParentBottom 如果该值为 true，则将该控件的底部和父控件的底部对齐

android:layout_alignParentLeft 如果该值为 true，则将该控件的左边与父控件的左边对齐

android:layout_alignParentRight　　如果该值为 true，则将该控件的右边与父控件的右边对齐
android:layout_alignParentTop　　如果该值为 true，则将空间的顶部与父控件的顶齐
android:layout_centerHorizontal　　如果值为真，该控件将被至于水平方向的中央
android:layout_centerInParent　　如果值为真，该控件将被至于父控件水平方向和垂直方向的中央
android:layout_centerVertical　　如果值为真，该控件将被至于垂直方向的中央

14.4 RelativeLayout 的使用



比如要实现上面的布局效果，我们需要在 在 xml 中做如下的声明：

```
<RelativeLayout android:id="@+id/RelativeLayout01"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <TextView android:id="@+id/label"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:text="用户名："
 </TextView>
 <EditText android:id="@+id/entry"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
    android:layout_below="@id/label"> //确定 EditText 控件在 ID 为 label 的元素下方

 </EditText>
   <Button android:id="@+id/cancel"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
    android:layout_alignParentRight="true"   //声明该元素在其父元素的右边边界对齐
        android:layout_marginLeft="10dip" //左移 10dip
        android:layout_below="@id/entry"
        android:text="取消" >
 </Button>
 <Button android:id="@+id/ok"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
```

```
        android:layout_toLeftOf="@id/cancel" //声明该元素在 ID 为 cancel 元素的左边
        android:layout_alignTop="@id/cancel“ //声明该元素与 ID 为 cancel 的元素在相同的水平位置

        android:text=“确认”>
    </Button>
  </RelativeLayout>
```

15 AbsoluteLayout 绝对布局

15.1 AbsoluteLayout 层次关系

绝对布局（AbsoluteLayout）能通过指定界面元素的坐标位置，来确定用户界面的整体布局

绝对布局是一种不推荐使用的界面布局，因为通过 X 轴和 Y 轴确定界面元素位置后，Android 系统不能够根据不同屏幕对界面元素的位置进行调整，降低了界面布局对不同类型和尺寸屏幕的适应能力。

同其他布局一样，android.widget.AbsoluteLayout 是 android.view.ViewGroup 类的子类，其层次关系如下：

```
java.lang.Object
  android.view.View
    android.widget.ViewGroup
      android.widget.AbsoluteLayout
```

15.2 AbsoluteLayout 常用的方法

方法	功能描述	返回值
AbsoluteLayout	提供了 3 个构造函数	null
checkLayoutParams(ViewGroup.LayoutParams p)	检查参数指定的布局参数是否是 LayoutParams 实例	
onLayout(boolean changed,int l,int t,int r,int b)	视图的布局改变时，该方法被调用	boolean
setBaselineAligned(boolean baseelineAligned)	根据参数设置基线对齐	boolen
onMeasure(int widthMesureSpec, int h)	该方法被 measure 调用，用于测量视图的高度和宽度	void
setHorizontalGravity(int gravity)	设置水平方向是的重力	void

15.3 AbsoluteLayout 常用属性

对与 AbsoluteLayout 标签，最主要的两个参数是

android：layout_x　指定 X 坐标的位置

android：layout_y　指定 Y 坐标的位置

AbsoluteLayout 常用的属性有：

第一类:属性值为 true 或 false

android:layout_centerHrizontal 水平居中
android:layout_centerVertical 垂直居中
android:layout_centerInparent 相对于父元素完全居中
android:layout_alignParentBottom 贴紧父元素的下边缘
android:layout_alignParentLeft 贴紧父元素的左边缘
android:layout_alignParentRight 贴紧父元素的右边缘
android:layout_alignParentTop 贴紧父元素的上边缘
android:layout_alignWithParentIfMissing 如果对应的兄弟元素找不到的话就以父元素做参照物

第二类：属性值必须为 id 的引用名“@id/id-name”

android:layout_below 在某元素的下方
android:layout_above 在某元素的的上方
android:layout_toLeftOf 在某元素的左边
android:layout_toRightOf 在某元素的右边

android:layout_alignTop 本元素的上边缘和某元素的的上边缘对齐
android:layout_alignLeft 本元素的左边缘和某元素的的左边缘对齐
android:layout_alignBottom 本元素的下边缘和某元素的的下边缘对齐
android:layout_alignRight 本元素的右边缘和某元素的的右边缘对齐

第三类：属性值为具体的像素值，如 30dip，40px

android:layout_marginBottom 离某元素底边缘的距离
android:layout_marginLeft 离某元素左边缘的距离
android:layout_marginRight 离某元素右边缘的距离
android:layout_marginTop 离某元素上边缘的距离

15.4 AbsoluteLayout 的使用

要实现上图所示的效果，我们可以在 XML 布局声明如下：

```
<AbsoluteLayout android:id="@+id/AbsoluteLayout01"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 xmlns:android="http://schemas.android.com/apk/res/android">
 <TextView android:id="@+id/label"
        android:layout_x="40dip"
```

```
        android:layout_y="40dip"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="用户名："> 
    </TextView>
    <EditText android:id="@+id/entry"
        android:layout_x="40dip"
        android:layout_y="60dip"
        android:layout_height="wrap_content" 8.                    android:layout_width="150dip">
     </EditText>
    <Button android:id="@+id/ok"
            android:layout_width="70dip"
           android:layout_height="wrap_content"
           android:layout_x="40dip"
            android:layout_y="120dip"
            android:text="确认">
    </Button>
   <Button android:id="@+id/cancel"
            android:layout_width="70dip"
           android:layout_height="wrap_content"
            android:layout_x="120dip"
           android:layout_y="120dip"
           android:text="取消">
  </Button>
   </AbsoluteLayout>
```

15.5 补充

文章连接：

我的 Android 学习之旅[6]——以示例程序来展示 Android 的几种布局方式

第六讲：用户界面 View（二）

如何动态改变 AbsoluteLayout 布局中其它布局的坐标

16 Menu 菜单

16.1 Menu 的介绍

菜单是应用程序中非常重要的组成部分，能够在不占用界面空间的前提下，为应用程序提供了统一的功能和设置界面，并为程序开发人员提供了易于使用的编程接口

Android 系统支持三种菜单

选项菜单（Option Menu）

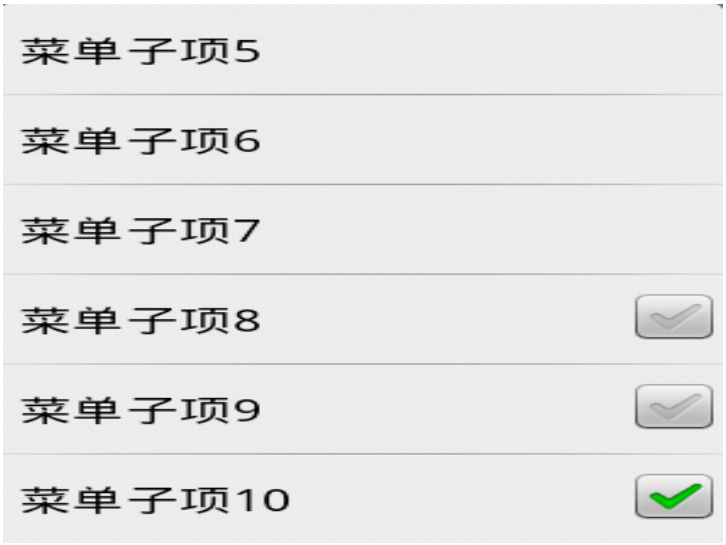
子菜单（Submenu）

上下文菜单（Context Menu）

16.2 选项菜单的介绍以及使用

选项菜单是一种经常被使用的 Android 系统菜单
打开方式：通过“菜单键”（MENU key）打开
选项菜单分类：
图标菜单（Icon Menu）
扩展菜单（Expanded Menu）
（1）图标菜单能够同时显示文字和图标的菜单，最多支持 6 个子项
图标菜单不支持单选框和复选框

（2）扩展菜单是在图标菜单子项多余 6 个时才出现，通过点击图标菜单最后的子项“More”才能打开
扩展菜单是垂直的列表型菜单
不能够显示图标
支持单选框和复选框



下面我们就要来实现选项菜单的功能：

16.3 子菜单的介绍以及使用

子菜单是能够显示更加详细信息的菜单子项
菜单子项使用了浮动窗体的显示形式，能够更好适应小屏幕的显示方式

子菜单不支持嵌套

子菜单的添加是使用 addSubMenu()函数实现

16.4 上下文菜单的介绍以及使用

快捷菜单同样采用了动窗体的显示方式，与子菜单的实现方式相同，但两种菜单的启动方式却截然不同

启动方式：快捷菜单类似于普通桌面程序中的“右键菜单”，当用户点击界面元素超过 2 秒后，将启动注册到该界面元素的快捷菜单

使用方法：与使用选项菜单的方法非常相似，需要重载 onCreateContextMenu()函数和 onContextItemSelected()函数

onCreateContextMenu()函数主要用来添加快捷菜单所显示的标题、图标和菜单子项等内容

上下文菜单常用的方法：

setHeaderIcon（int iconRes） 设置上下文菜单的图标

setHeaderIcon（Drawable icon） 上下文菜单的图标

setHeaderTitle（Charsequence title） 设置上下文菜单的标题

setHeaderTitle（int titleRes）设置上下文菜单的标题

add（int groupId，int itemId,int order, CharSequence title） 使用 add 方法添加 子菜单：

groupId：组 Id

itemId：菜单项 Id

order ：顺序号

title ：菜单项标题

17 AutoCompleteTextView 自动提示

17.1 AutoCompleteTextView 类层次关系

AutoCompleteTextView 控件是用户在文本框中输入的时候，在控件下方会显示一个类似 百度的下拉提示框，提示当前与输入相匹配的选项，用户可以直接选择，方便了用户的直接体验。

AutoCompleteTextView 类的层次关系如下：

```
java.lang.Object
    android.view.view
        android.view.TextView
            android.widget.EditText
                android.widget. AutoCompleteTextView
```

17.2 AutoCompleteTextView 常用的方法

方法	功能描述	返回值
setMarqueeRepeatLimit	在指定的 marquee 的情况下，设置重复滚动的次数，当设置为 marquee_forever 时表示无限次	void
enoughToFilter	当文本长度超过阀值时过滤	boolean（true/False）
performValidation	确定文本中单个符号的有效性	void
setTokenzier	设置分词组件，该组件决定用户正在输入文本的范围	void
performFiltering	过滤从函数 findTokenStart()到函数 getSelectionEnd()获得的长度为 0 或者超过了预定值的文本内容	void
replaceText	根据参数的文本替换从函数 findTokenStart()到函数 getSelectionEnd()获得的文本	void

17.3 AutoCompleteTextView 的实现

AutoCompleteTextView 在 XML 中的声明方式如下：

```
<AutoCompleteTextView
  android:id="@+id/AutoCompleteTextView01"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  >
  </AutoCompleteTextView>
  <MultiAutoCompleteTextView
  android:id="@+id/MultiAutoCompleteTextView01"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  >

  </MultiAutoCompleteTextView>
```

AutoCompleteTextView 在代码中的实现步骤：

```
        //关联关键字
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line, autoString);

        AutoCompleteTextView m_AutoCompleteTextView = (AutoCompleteTextView)
findViewById(R.id.AutoCompleteTextView01);

        //将adapter添加到AutoCompleteTextView中
        m_AutoCompleteTextView.setAdapter(adapter);
        ////////////////////
        MultiAutoCompleteTextView mm_AutoCompleteTextView = (MultiAutoCompleteTextView)
findViewById(R.id.MultiAutoCompleteTextView01);
        //将adapter添加到AutoCompleteTextView中
        mm_AutoCompleteTextView.setAdapter(adapter);

        mm_AutoCompleteTextView.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
```

17.4 补充

补充

相关文章链接

[Android 控件之 AutoCompleteTextView、MultiAutoCompleteTextView 探究]

[AutoCompleteTextView 和 MultiAutoCompleteTextView]

[Auto Complete Text]

18 DatePicker 日期

18.1 DatePicker 的类层次关系

DatePicker 日期选择器是一个选择年月日的日历布局视图，它的类层次关系如下：

public class DatePicker extends FrameLayout

```
java.lang.Object
    android.view.View
        android.view.ViewGroup
            android.widget.FrameLayout
                android.widget.DatePicker
```

18.2 DatePicker 的常用方法

公共方法

（1）public int getDayOfMonth ()

获取选择的天数

（2）public int getMonth ()

获取选择的月份。（注意：返回数值为 0..11，需要自己+1 来显示）

（3） public int getYear ()

获取选择的年份

（4）public void init (int year, int monthOfYear, int dayOfMonth, DatePicker.OnDateChangedListener onDateChangedListener)

初始化状态。（初始化年月日）

参数：

year ： 初始年（注意使用 *new Date()*初始化年时，需要+1900，如下：*date.getYear() + 1900*）

monthOfYear： 初始月。

dayOfMonth： 初始日。

onDateChangedListener： 日期改变时通知用户的事件监听，可以为空(null)。

（5）public void setEnabled (boolean enabled)

设置视图的启用状态。该启用状态随子类的不同而有不同的解释。

参数

enabled True if this view is enabled, false otherwise.设置为 true 表示启动视图，反之禁用。

（6）public void updateDate (int year, int monthOfYear, int dayOfMonth)

更新日期

受保护方法

（7）protected void dispatchRestoreInstanceState (SparseArray<Parcelable> container)

重写使我们能够完全控制这小部件的保存或恢复。（译者注：此处直接调用了父类的 *ViewGroup. dispatchThawSelfOnly* 方法）

参数

container SparseArray 持有保存以前的状态。The SparseArray which holds previously saved state.

（8）protected void onRestoreInstanceState (Parcelable state)

允许视图重新应用以前通过 onSaveInstanceState()生成代表内部的状态。这个函数决不调用一个空的状态。

参数

state 返回以前调用 onSaveInstanceState()保存下来的状态。

（9） protected Parcelable onSaveInstanceState ()

允许视图生成一个代表内部的状态，以后可用于创建一个与之相同的新的实例。这种状态应该只包含非持久或以后不能够重建的信息。例如，你决不存储你当前在屏幕上的位置，因为这会在视图的层面上重新计算放置一个新的实例。

你可以存储到这里的一些例子：一个文本框中当前光标的位置（但通常不是文字本身，文字通常保存在内容提供者(content provider)或其他持久的储存中），一个列表视图中的当前选中项。

返回值

返回一个包含视图当前状态的 Parcelable 对象，或没有什么状态保存时返回 null。默认实现返回 null。

18.3 DatePicker 的使用

首先在 XML 中声明一个 DatePicker,和一个用户显示监听结果的 TextView:

```
<DatePicker android:id="@+id/datePicker1"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"></DatePicker>
    <TextView
    android:id="@+id/textview1"
      android:layout_height="wrap_content"
      android:text="@string/hello"
        android:layout_width="wrap_content"/>
```

然后在 Activity 中显示 DatePicker 并实现监听事件

```
textview = (TextView)findViewById(R.id.textview1);
            datepicker = (DatePicker)findViewById(R.id.datePicker1);
            //获取当前时间
            calendar = Calendar.getInstance();
            //显示当前的时间
            textview.setText("当前时间："+calendar.get(Calendar.YEAR)+"年"
                    +(calendar.get(Calendar.MONTH)+1)+"月"+calendar.get(Calendar.DAY_OF_MONTH)+"日");

            //注册监听事件,当日期改变时，更新TextView的内容
            datepicker.init(calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH), calendar.get(Calendar.DAY_OF_MONTH),
new DatePicker.OnDateChangedListener() {

                    @Override
                    public void onDateChanged(DatePicker view, int year, int monthOfYear,
                            int dayOfMonth) {
                        // TODO Auto-generated method stub
                        textview.setText("当前时间："+year+"年"
                            +(monthOfYear+1)+"月"+dayOfMonth+"日");
                    }
                    });
```

效果如下显示：

19 TimePicker 时间选择器

19.1 Timepicker 类的层次关系

TimePicker 时间选择器是用于选择一天中时间的视图，TimePicker 类层次关系如下：

public class TimePicker extends FrameLayout

java.lang.Object
    android.view.View
        android.view.ViewGroup
            android.widget.FrameLayout
                android.widget.TimePicker

19.2 TimePicker 类常用的方法

公共方法

（1）public int getBaseline ()

返回窗口空间的文本基准线到其顶边界的偏移量。如果这个部件不支持基准线对齐，这个方法返回-1/。

返回值

基准线的偏移量，如果不支持基准线对齐则返回-1。

（2） public Integer getCurrentHour ()

获取当前时间的小时部分。

返回值

当前小时（0-23）

（3）public Integer getCurrentMinute ()

获取当前时间的分钟部分。

返回值

当前分钟。

（4）public boolean is24HourView ()

获取当前系统设置是否是 24 小时制。

返回值

如果是 24 小时制返回 true，否则返回 false。

（5）public void setCurrentHour (Integer currentHour)

设置当前小时。

（6）public void setCurrentMinute (Integer currentMinute)

设置当前分钟（0-59）。

（7）public void setEnabled (boolean enabled)
设置可用的视图状态。可用的视图状态的解释在子类中改变。
参数
enabled 如果可用为 true，否则为 false。

（8）public void setIs24HourView (Boolean is24HourView)
设置是 24 小时还是上午/下午制。
参数
is24HourView True 表示 24 小时制. False 表示上午/下午制.

（9）public void setOnTimeChangedListener (TimePicker.OnTimeChangedListener onTimeChangedListener)
设置时间调整事件的回调函数。
参数
onTimeChangedListener 回调函数，不能为空。

受保护方法

（10）protected void onRestoreInstanceState (Parcelable state)
允许一个视图回复到之前用 onSaveInstanceState()保存的状态，state 参数不能为空。
参数
state 之前调用 onSaveInstanceState()返回的状态。

（11）protected Parcelable onSaveInstanceState ()

用来允许一个视图保存当前的内部状态，之后可以创建新的实例应用相同的状态。状态信息不能包含常量或在之后重新构造。例如，你永远不能保存在屏幕上的当前位置，因为当创建一个新的视图时，它将会被放置到它的层次结构中，它的位置会被重新计算。

你可以存储到这里的一些例子：一个文本框中当前光标的位置（但通常不是文字本身，文字通常保存在内容提供者(content provider)或其他持久的储存中），一个列表视图中的当前选中项。

返回值
返回一个包含视图当前状态的 Parcelable 对象，或没有什么状态保存时返回 null。默认实现返回 null。

19.3 TimePicker 的使用实例

TimePicker 和 DatePicker 的使用一样：

XML 中：

```
<TimePicker android:id="@+id/timePicker1"
    android:layout_width="wrap_content" android:layout_height="wrap_content"></TimePicker>

<TextView
    android:id="@+id/textview1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
```

然后在 Activity 中显示 TimePicker 并实现监听：

```
imepicker = (TimePicker)findViewById(R.id.timePicker1);
        textview =(TextView)findViewById(R.id.textview1);
      //获取当前时间
        calendar = Calendar.getInstance();
        //显示当前的时间
        textview.setText("当前时间："+calendar.get(Calendar.HOUR)+"时"
                +calendar.get(Calendar.MINUTE)+"分");

        //注册监听事件,当时间改变时，更新TextView的内容
        timepicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {
```

```
            @Override
            public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
                // TODO Auto-generated method stub
                textview.setText("当前时间："+hourOfDay+"时"
                        +minute+"分");
            }
        });
```

19.4 补充

补充

文章链接

[示例代码]Hello, TimePicker
[示例代码]日期選擇器(DatePicker)和時間選擇器(TimePicker)
Android TimePicker DatePicker 简单说明

20 Dialog 对话框

20.1 Dialog 对话框

Android 中实现对话框可以使用 AlterDialog.Builder 类，还可以自定义对话框。如果对话框设置了按钮，那么就需要对其设置监听 OnClickListener。

20.2Dialog 的使用

首先在 Dialog.xml 中定义一个对话框，我们这里定义一个包含两个 TextView 和两个 EditView 的对话框
Dialog.xml 中的布局如下:

```
<TextView
        android:id="@+id/username"
```

```
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_marginLeft="20dip"
        android:layout_marginRight="20dip"
        android:text="账号"
        android:gravity="left"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/username"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_marginLeft="20dip"
        android:layout_marginRight="20dip"
        android:scrollHorizontally="true"
        android:autoText="false"
        android:capitalize="none"
        android:gravity="fill_horizontal"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <TextView
        android:id="@+id/password"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_marginLeft="20dip"
        android:layout_marginRight="20dip"
        android:text="密码"
        android:gravity="left"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/password"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_marginLeft="20dip"
        android:layout_marginRight="20dip"
        android:scrollHorizontally="true"
        android:autoText="false"
        android:capitalize="none"
        android:gravity="fill_horizontal"
        android:password="true"
        android:textAppearance="?android:attr/textAppearanceMedium" />
```

main.xml 布局中：

```
<TextView
```

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
```

然后在 Activity 中使用 AlterDialog 创建对话框：

```
Dialog dialog = new AlertDialog.Builder(DialogActivity.this)
                .setTitle("登陆提示")//设置标题
                .setMessage("这里需要登录！ ")//设置内容
                .setPositiveButton("确定",//设置确定按钮
                new DialogInterface.OnClickListener()
                {
                    public void onClick(DialogInterface dialog, int whichButton)
                    {
                        //点击“确定”转向登陆框

                        LayoutInflater factory = LayoutInflater.from(DialogActivity.this);
                        //得到自定义对话框
                      final View DialogView = factory.inflate(R.layout.dialog, null);
                      //创建对话框
                      AlertDialog dlg = new AlertDialog.Builder(DialogActivity.this)
                      .setTitle("登录框")
                      .setView(DialogView)//设置自定义对话框的样式
                      .setPositiveButton("确定", //设置"确定"按钮
                      new DialogInterface.OnClickListener() //设置事件监听
                      {
                          public void onClick(DialogInterface dialog, int whichButton)
                          {
                              //输入完成后，点击“确定”开始登陆
                              m_Dialog = ProgressDialog.show
                                          (
                                        DialogActivity.this,
                                            "请等待...",
                                            "正在为你登录...",
                                            true
                                        );

                            new Thread()
                            {
                              public void run()
                              {
                                try
                                {
                                  sleep(3000);
                                }
                                catch (Exception e)
                                {
                                  e.printStackTrace();
                                }
```

```
                    finally
                    {
                        //登录结束，取消m_Dialog对话框
                        m_Dialog.dismiss();
                    }
                }
            }.start();
        }
    })
    .setNegativeButton("取消", //设置“取消”按钮
    new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int whichButton)
        {
            //点击"取消"按钮之后退出程序
            DialogActivity.this.finish();
        }
    })
    .create();//创建
    dlg.show();//显示
  }
}).setNeutralButton("退出",
new DialogInterface.OnClickListener()
{
public void onClick(DialogInterface dialog, int whichButton)
{
    //点击"退出"按钮之后推出程序
    DialogActivity.this.finish();
}
}).create();//创建按钮

// 显示对话框
dialog.show();
```

实现的效果图：

21 ImageView 图片视图

21.1 ImageView 类的结构

ImageView 显示任意图像，例如图标。ImageView 类可以加载各种来源的图片（如资源或图片库），需要计算图像的尺寸，比便它可以在其他布局中使用，并提供例如缩放和着色（渲染）各种显示选项。

继承关系

public class View.OnClickListner extends View

```
java.lang.Object
    android.view.View
        android.widget.ImageView
```

直接子类

ImageButton, QuickContactBadge

间接子类

ZoomButton

21.2 ImageView 类属性

XML 属性

属性名称	描述
android:adjustViewBounds	设置该属性为真可以在 ImageView 调整边界时保持图片的纵横比例。（译者注：需要与 maxWidth、MaxHeight 一起使用，否则单独使用没有效果。）
android:baseline	视图内基线的偏移量
android:baselineAlignBottom	如果为 true，图像视图将基线与父控件底部边缘对齐。
android:cropToPadding	如果为真，会剪切图片以适应内边距的大小。（译者注：是否截取指定区域用空白代替。单独设置无效果，需要与 scrollY 一起使用，效果如下，实现代码见代码部分： ）
android:maxHeight	为视图提供最大高度的可选参数。（译者注：单独使用无效，需要与 setAdjustViewBounds 一起使用。如果想设置图片固定大小，又想保持图片宽高比，需要如下设置： 1） 设置 setAdjustViewBounds 为 true； 2） 设置 maxWidth、MaxHeight； 3） 设置 layout_width 和 layout_height 为 wrap_content。）
android:maxWidth	为视图提供最大宽度的可选参数。
android:scaleType	控制为了使图片适合 ImageView 的大小，应该如何变更图片大小或移动图片。一定是下列常量之一： 常量 | 值 | 描述 matrix | 0 | 用矩阵来绘图 fitXY | 1 | 拉伸图片（不按比例）以填充 View 的宽高 fitStart | 2 | 按比例拉伸图片，拉伸后图片的高度为 View 的高度，且显示在 View 的左边 fitCenter | 3 | 按比例拉伸图片，拉伸后图片的高度为 View 的高度，且显示在 View 的中间

	fitEnd	4	按比例拉伸图片，拉伸后图片的高度为 View 的高度，且显示在 View 的右边
	center	5	按原图大小显示图片，但图片宽高大于 View 的宽高时，截图图片中间部分显示
	centerCrop	6	按比例放大原图直至等于某边 View 的宽高显示。
	centerInside	7	当原图宽高或等于 View 的宽高时，按原图大小居中显示；反之将原图缩放至 View 的宽高居中显示。
	（译者注：设置图片的填充方式。）		
android:src	设置可绘制对象作为 ImageView 显示的内容		
android:tint	为图片设置着色颜色。（译者注：将图片渲染成指定的颜色。见下图： 左边为原图，右边为设置后的效果，见后面代码。）		

21.3 ImageView 常用的方法

公共方法

（1）public final void **clearColorFilter** ()
（译者注：清除颜色过滤，参见这里）

（2）public int **getBaseline** ()
返回部件顶端到文本基线的偏移量。如果小部件不支持基线对齐，该方法返回 -1。
返回值
小部件顶端到文本基线的偏移量；或者是 -1 当小部件不支持基线对齐时。

（3）public boolean **getBaselineAlignBottom** ()
返回当前视图基线是否将考虑视图的底部。
参见
setBaselineAlignBottom(boolean)

（4）public Drawable **getDrawable** ()
返回视图的可绘制对象；如果没有关联可绘制对象，返回空。

（5）public Matrix **getImageMatrix** ()
返回视图的选项矩阵。当绘制时，应用于视图的可绘制对象。如果没有矩阵， 函数返回空。不要更改这个矩阵。如果你要为可

绘制对象设置不同的矩阵， 请调用 setImageMatrix()。

（6）public ImageView.ScaleType **getScaleType** ()
返回当前 ImageView 使用的缩放类型。
相关 XML 属性
android:scaleType
参见
ImageView.ScaleType

（7）public void **invalidateDrawable** (Drawable dr)
使指定的可绘制对象失效。
参数
dr 要设为失效的可绘制对象。

（8）public void **jumpDrawablesToCurrentState** ()
调用与视图相关的所有可绘制对象的 Drawable.jumpToCurrentState()方法。

（9）public int[] **onCreateDrawableState** (int extraSpace)
为当前视图生成新的 Drawable 状态时发生。当视图系统检测到缓存的可绘制对象失效时，调用该方法.你可以使用 getDrawableState() 方法重新取得当前的状态。
参数
extraSpace 如果为非零，该值为你要在返回值的数组中存放的你自己的状态信息的数量。
返回值
返回保存了视图的当前 Drawable 状态的数组。

（10）public void **setAdjustViewBounds** (boolean adjustViewBounds)
当你需要在 ImageView 调整边框时保持可绘制对象的比例时，将该值设为真。
参数
adjustViewBounds 是否调整边框，以保持可绘制对象的原始比例。
相关 XML 属性
android:adjustViewBounds

（11）public void **setAlpha** (int alpha)
（译者注：设置透明度）

（12）public void **setBaseline** (int baseline)
设置部件顶部边界文本基线的偏移量。这个值覆盖 setBaselineAlignBottom(boolean)设置的属性值。
参数
baseline 使用的基线，或不提供设置为-1。
相关 XML 属性
android:baseline
参见
setBaseline(int)

（13）public void **setBaselineAlignBottom** (boolean aligned)
设置是否设置视图底部的视图基线。设置这个值覆盖 setBaseline()的所有调用。
参数
aligned 如果为 true，图像视图将基线与父控件底部边缘对齐。
相关 XML 属性
android:baselineAlignBottom

（14）public final void **setColorFilter** (int color)
为图片设置着色选项。采用 SRC_ATOP 合成模式。
参数
color 应用的着色颜色。
相关 XML 属性
android:tint

（15）public void **setColorFilter** (ColorFilter cf)
为图片应用任意颜色滤镜。
参数
cf 要应用的颜色滤镜（可能为空）

（16）public final void **setColorFilter** (int color, PorterDuff.Mode mode)
为图片设置着色选项。
参数
color 应用的着色颜色。
mode 如何着色。标准模式为 SRC_ATOP。

相关 XML 属性
android:tint

（17）public void **setImageBitmap** (Bitmap bm)
设置位图作为该 ImageView 的内容。
参数
bm 设置的位图。

（18）public void **setImageDrawable** (Drawable drawable)
设置可绘制对象为该 ImageView 显示的内容。
参数
drawable 设置的可绘制对象。

（19）public void **setImageLevel** (int level)
设置图片的等级，当图片来自于 LevelListDrawable 时。（译者注：使用参见这里）
参数
level 图片的新的等级。

（20）public void **setImageMatrix** (Matrix matrix)
（译者注：矩阵变换）

（21） public void **setImageResource** (int resId)
通过资源 ID 设置可绘制对象为该 ImageView 显示的内容。
注意：该操作读取位图，并在 UI 线程中解码，因此可能导致反应迟缓。 如果反应迟缓，可以考虑用 setImageDrawable(Drawable)、 setImageBitmap(Bitmap) 或者 BitmapFactory 代替。
参数
resId 可绘制对象的资源标识。
相关 XML 属性
android:src

（21）public void **setImageState** (int[] state, boolean merge)
（译者注：设置视图的可见和不可见，使用参见这里）

（22）public void **setImageURI** (Uri uri)
设置指定的 URI 为该 ImageView 显示的内容。
注意：该操作读取位图，并在 UI 线程中解码，因此可能导致反应迟缓。 如果反应迟缓，可以考虑用 setImageDrawable(Drawable)、 setImageBitmap(Bitmap) 或者 BitmapFactory 代替。
参数
uri 图像的 URI。

（23）public void **setMaxHeight** (int maxHeight)
用于设置该视图支持的最大高度的可选参数。只有 setAdjustViewBounds(boolean) 为真时有效。要设置图像最大尺寸为 100×100，并保持原始比率，做法如下：
1) 设置 adjustViewBounds 为真；
2) 设置 maxWidth 和 maxHeight 为 100；
3) 设置宽、高的布局参数为 WRAP_CONTENT。
注意，如果原始图像较小，即使设置了该参数，图像仍然要比 100×100 小。如果要设置图片为 固定大小，需要在布局参数中指定大小，并使用 setScaleType(ImageView.ScaleType) 函数来检测，如何 将其调整到适当的大小。
参数
maxHeight 该视图的最大高度。
相关 XML 属性
android:maxHeight

（24）public void **setMaxWidth** (int maxWidth)
用于设置该视图支持的最大宽度的可选参数。只有 setAdjustViewBounds(boolean) 为真时有效。要设置图像最大尺寸为 100×100，并保持原始比率，做法如下：
4) 设置 adjustViewBounds 为真；
5) 设置 maxWidth 和 maxHeight 为 100；
6) 设置宽、高的布局参数为 WRAP_CONTENT。
注意，如果原始图像较小，即使设置了该参数，图像仍然要比 100×100 小。如果要设置图片为 固定大小，需要在布局参数中指定大小，并使用 setScaleType(ImageView.ScaleType) 函数来检测，如何 将其调整到适当的大小。
参数
maxWidth 该视图的最大宽度。
相关 XML 属性
android:maxWidth

（25）public void **setScaleType** (ImageView.ScaleType scaleType)

控制图像应该如何缩放和移动，以使图像与 ImageView 一致。

参数

scaleType 需要的缩放方式。

相关 XML 属性

android:scaleType

（26）public void **setSelected** (boolean selected)

改变视图的选中状态。视图有选中和未选中两个状态。注意，选择状态不同于焦点。典型的选中的视图是象 ListView 和 GridView 这样的 AdapterView 中显示的 内容；选中的内容会显示为高亮。

参数

selected 为真，将视图设为选中状态；否则为假。

受保护方法

（27）protected void **drawableStateChanged** ()

在视图状态的变化影响到所显示可绘制对象的状态时调用该方法。

覆盖该方法时，要确保调用了父类的该方法。

（28）protected void **onDraw** (Canvas canvas)

实现该方法，用于自己绘制内容。

参数

canvas 用于绘制背景的画布。

（29）protected void **onMeasure** (int widthMeasureSpec, int heightMeasureSpec)

评估视图及其内容，以决定其宽度和高度.此方法由 measure(int, int) 调用，子类可以重载以提供更精确、更有效率的衡量其内容尺寸的方法。

约定： 覆盖该方法时，必须调用 setMeasuredDimension(int, int) 方法来保存评估结果的视图的宽度和高度.如果忘记将导致 measure(int, int) 方法抛出 IllegalStateException 异常。要有效的利用父类的 onMeasure(int, int)方法。

基类测量的是背景的大小，除非 MeasureSpec 允许超过背景.子类应该重写 onMeasure(int, int) 方法，以为其内容提供更适合的大小。

如果重写了该方法，子类要确保其高度和宽度大于等于视图的最小高度和宽度.（getSuggestedMinimumHeight() 和 getSuggestedMinimumWidth()）

参数

widthMeasureSpec 父视图要求的横向空间大小.该要求由 View.MeasureSpec 进行了编码处理。

heightMeasureSpec 父视图要求的纵向空间大小.该要求由 View.MeasureSpec 进行了编码处理。

（30）protected boolean **onSetAlpha** (int alpha)

透明度改变时执行。子类可以使用该方法指定透明度值，然后返回真； 在调用 onDraw() 时，使用该透明度值。如果返回假，则先在不可见的缓存中绘制视图， 完成该请求；看起来不错，但是可能相对于在子类中绘制要慢。默认实现返回假。

参数

alpha 应用到视图的透明度值 (0…255)。

返回值

如果该类可以绘制该阿尔法值返回真。

（31）protected boolean **setFrame** (int l, int t, int r, int b)

为视图指定大小和位置。 该方法有布局调用。

参数

l 左侧位置，相对于父容器。

t 顶部位置，相对于父容器。

r 右侧位置，相对于父容器。

b 底部位置，相对于父容器。

返回值

true 如果新的大小和位置与之前的不同，返回真。

（32）protected boolean **verifyDrawable** (Drawable dr)

如果你的视图子类显示自己的可绘制对象，他应该重写此方法并为自己的每个可绘制对象返回真。该函数允许为这些可绘制对象准备动画效果。

重写此方法时，要保证调用其父类的该方法。

参数

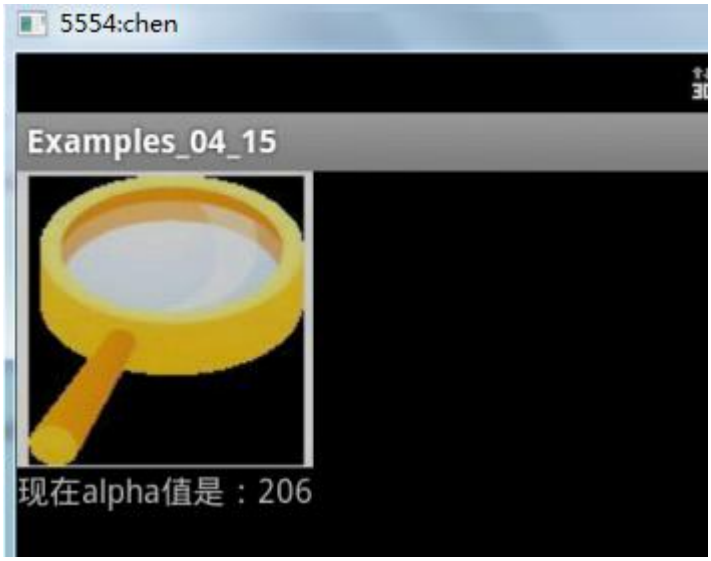
dr 待校验的可绘制对象.如果是你显示的对象之一，返回真；否则返回调用父类的返回值。

返回值

boolean 如果可绘制对象已经显示在视图上了，返回真；否则返回假，不允许动画效果。

21.4 ImageView 的使用

如果现在我们要显示一张图片，并动态更新它的透明度 alpha，效果如下图所示



则我们首先要在 XML 文件中声明一个 ImageView 和一个 TextView :

```
<ImageView
    android:id="@+id/ImageView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    >
</ImageView>
<TextView
    android:id="@+id/TextView01"
    android:layout_below="@id/ImageView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    >
</TextView>
```

然后在 Activity 中设置显示 ImageView:

```
isrung          = true;

        //获得ImageView的对象
        imageview = (ImageView) this.findViewById(R.id.ImageView01);
        textview = (TextView) this.findViewById(R.id.TextView01);

        //设置imageview的图片资源。同样可以再xml布局中像下面这样写
        //android:src="@drawable/logo"
        imageview.setImageResource(R.drawable.imag2);

        //设置imageview的Alpha值
        imageview.setAlpha(image_alpha);
```

之后就需要用一个线程来实时更新 alpha：

```
//开启一个线程来让Alpha值递减
        new Thread(new Runnable() {
            public void run()
            {
                while (isrung)
                {
                    try
                    {

                        Thread.sleep(200);
                        //更新Alpha值
                        updateAlpha();
                    }
                    catch (InterruptedException e)
                    {
                        e.printStackTrace();
                    }
                }

            }
        }).start();

        //接受消息之后更新imageview视图
        mHandler = new Handler() {
            @Override
            public void handleMessage(Message msg)
            {
                super.handleMessage(msg);
                imageview.setAlpha(image_alpha);
                textview.setText("现在alpha值是："+Integer.toString(image_alpha));
                //更新
                imageview.invalidate();
            }
        };
    }

    public void updateAlpha()
    {
        if (image_alpha - 7 >= 0)
        {
            image_alpha -= 7;
        }
        else
        {
            image_alpha = 0;
            isrung = false;
        }
        //发送需要更新imageview视图的消息
        mHandler.sendMessage(mHandler.obtainMessage());
```

}

21.5 补充

补充

文章精选

Android ImageView 加边框
Android 用 ImageView 显示本地和网上的图片
imageView 动画效果

22 Gallery 拖动效果

22. 1 Gallery 类的层次关系

Gallery 是一个锁定中心条目并且拥有水平滚动列表的视图。
Gallery（画廊）使用 Theme_galleryItemBackground 作为 Gallery（画廊）适配器中的各视图的默认参数。如果你没有设置，你就需要调整一些 Gallery（画廊）的属性，比如间距。
Gallery（画廊）中的视图应该使用 Gallery.LayoutParams 作为它们的布局参数类型。
参见 Gallery tutorial。

内部类

class Gallery.LayoutParams

Gallery(画廊)扩展了 LayoutParams，以此提供可以容纳当前的转换信息和先前的位置转换信息的场所。

Galery 是一个实现图片拖动的非常炫的一个效果，Gallery 类的层次关系如下：

结构

继承关系

```
public class Gallery extends AbsSpinner
                    implements GestureDetector.OnGestureListener

java.lang.Object
  android.view.View
    android.view.ViewGroup
      android.widget.AdapterView<T extends android.widget.Adapter>
        android.widget.AbsSpinner
          android.widget.Gallery
```

22．2 Gallery 的 XML 属性

XML 属性

属性名称	描述
android:animationDuration	设置布局变化时动画的转换所需的时间（毫秒级）。仅在动画开始时计时。该值必须是整数，比如：100。

<table>
<tr><td>android:gravity</td><td>指定在对象的 X 和 Y 轴上如何放置内容。指定一下常量中的一个或多个（使用“|”分割）
<table>
<tr><th>Constant</th><th>Value</th><th>Description</th></tr>
<tr><td>top</td><td>0x30</td><td>紧靠容器顶端，不改变其大小</td></tr>
<tr><td>bottom</td><td>0x50</td><td>紧靠容器底部，不改变其大小</td></tr>
<tr><td>left</td><td>0x03</td><td>紧靠容器左侧，不改变其大小</td></tr>
<tr><td>right</td><td>0x05</td><td>紧靠容器右侧，不改变其大小</td></tr>
<tr><td>center_vertical</td><td>0x10</td><td>垂直居中，不改变其大小</td></tr>
<tr><td>fill_vertical</td><td>0x70</td><td>垂直方向上拉伸至充满容器</td></tr>
<tr><td>center_horizontal</td><td>0x01</td><td>水平居中，不改变其大小</td></tr>
<tr><td>Fill_horizontal</td><td>0x07</td><td>水平方向上拉伸使其充满容器</td></tr>
<tr><td>center</td><td>0x11</td><td>居中对齐，不改变其大小</td></tr>
<tr><td>fill</td><td>0x77</td><td>在水平和垂直方向上拉伸，使其充满容器</td></tr>
<tr><td>clip_vertical</td><td>0x80</td><td>垂直剪切（当对象边缘超出容器的时候，将上下边缘超出的部分剪切掉）</td></tr>
<tr><td>clip_horizontal</td><td>0x08</td><td>水平剪切（当对象边缘超出容器的时候，将左右边缘超出的部分剪切掉）</td></tr>
</table></td></tr>
<tr><td>android:spacing</td><td>（译者注:设置图片之间的间距）</td></tr>
<tr><td>android:unselectedAlpha</td><td>设置未选中的条目的透明度(Alpha)。该值必须是 float 类型，比如：“1.2”。</td></tr>
</table>

22.3 Gallery 类的常用方法

公共方法

（1）public boolean **dispatchKeyEvent** (KeyEvent event)

在焦点路径上分发按钮事件到下一个视图。该路径从视图树的顶端遍历到当前获得焦点的视图。如果当前视图已获得焦点，就分发给自身。否则，就分发到下一个节点的焦点路径上。该方法监听任何按钮事件。

参数

event　　被分发的按钮事件

返回值

boolean　时间被处理返回 true，否则 false

（2）public void **dispatchSetSelected** (boolean selected)

分发 setSelected 给视图的子类。

参数

selected　新选中的状态

（4）public ViewGroup.LayoutParams **generateLayoutParams** (AttributeSet attrs)

返回一个新的已设置属性集合的布局参数。

参数

attrs 用于生成布局参数的属性集合

返回值

一个 ViewGroup.LayoutParams 实例或者它的子类

（5）public boolean **onDown** (MotionEvent e)

当轻击和按下手势事件发生时通知该方法。任何按下事件都会直接触发该方法。所有其他的事件都要先于该方法。

参数

e 按下动作事件

（6）public boolean **onFling** (MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)

当初始化的按下动作事件和松开动作事件匹配时通知 fling（译者注：快滑，用户按下触摸屏、快速移动后松开）事件。该动作的速度通过计算 X 和 Y 轴上每秒移动多少像素得来。

参数

e1 导致开始 fling 的按下动作事件。

e2 触发当前 onFling 方法的移动动作事件

velocityX 测量 fling 沿 X 轴上的速度，像素/每秒

velocityY 测量 fling 沿 Y 轴上的速度，像素/每秒

返回值

如果该事件被消耗返回 true，否则 false。

（7）public boolean **onKeyDown** (int keyCode, KeyEvent event)

处理左，右和点击事件

参数

keyCode 代表按下按钮的按键码，来自 KeyEvent。

event 定义按钮动作的 KeyEvent 对象。

返回值

如果已经处理了按钮事件，则返回 true。如果你想让下一个事件接收者处理，就返回 false

参见

onKeyDown(int, KeyEvent)

（8）public boolean **onKeyUp** (int keyCode, KeyEvent event)

KeyEvent.Callback.onKeyMultiple()方法的默认实现：当 KEYCODE_DPAD_CENTER 或者 KEYCODE_ENTER 被释放时，执行点击视图操作。

参数

keyCode 代表按下按钮的按键码，来自 KeyEvent。

event 定义按钮动作的 KeyEvent 对象。

返回值

如果已经处理了按钮事件，则返回 true。如果你想让下一个事件接收者处理，就返回 false

（9）public void **onLongPress** (MotionEvent e)

MotionEvent 初始化并按下触发长按并通知本方法。

参数

e 导致开始长按的初始按下动作事件。

（10）public boolean **onScroll** (MotionEvent e1, MotionEvent e2, float distanceX, float distanceY)

当初始按下动作事件和当前移动动作事件导致滚动时通知本方法。为了方便提供了 X 和 Y 轴上的距离。

监听屏幕滚动事件。为了方便提供了 X 和 Y 轴上的距离。

参数

e1 导致滚动开始按下的动作事件。

e2 触发当前 onScroll 方法的移动动作事件。

distanceX 距离是自上一次调用 onScroll 方法在 X 轴上的距离。不是 e1 和 e2 之间的距离。

distanceY 距离是自上一次调用 onScroll 方法在 Y 轴上的距离。不是 e1 和 e2 之间的距离。

返回值

如果该事件被消耗返回 true 否则 false

（11）public void **onShowPress** (MotionEvent e)

用户已经执行按下动作还没有执行移动或者弹起动作。该事件常通过高亮一个元素来向用户提供一个视觉反馈即用户的操作已经被辨识了。

参数

e 按下动作事件

（12）public boolean **onSingleTapUp** (MotionEvent e)

在轻击动作和 up 动作事件触发时通知本方法。（译者注：点击屏幕上的某项的执行流程 有两种情况，一种是时间很短，一种时间稍长：时间很短：onDown--->onSingleTapUp--->onSingleTapConfirmed，见这里 1，这里 2。）

参数

e 完成开始轻击的 up 动作事件

返回值

如果该事件被消耗返回 true 否则 false

（13）public boolean **onTouchEvent** (MotionEvent event)

实现该方法来处理触摸屏动作事件

参数

event　　动作事件

返回值

如果该事件被消耗返回 true 否则 false

（14）public void **setAnimationDuration** (int animationDurationMillis)

设置当子视图改变位置时动画转换时间。仅限于动画开始时生效。

参数

animationDurationMillis　　动画转换时间（毫秒级）

（15）public void **setCallbackDuringFling** (boolean shouldCallback)

当 flinged 时是否回调每一个 getOnItemSelectedListener()。如果设为 false，只回调最终选中的项。如果为 true，则所有的项都会回调。

参数

shouldCallback　　设置抛滑的过程中是否回调

（16）public void **setGravity** (int gravity)

描述子视图的对齐方式。

（17）public void **setSpacing** (int spacing)

设置 Gallery 中项的间距

参数

spacing　Gallery 中项的间距，以像素为单位

（18）public void **setUnselectedAlpha** (float unselectedAlpha)

设置 Gallery 中未选中项的透明度(alpha)值。

参数

unselectedAlpha　　未选中项的透明度(alpha)值

（19）public boolean **showContextMenu** ()

显示该视图上下文菜单。

返回值

上下文菜单是否显示。

（20）public boolean **showContextMenuForChild** (View originalView)

为指定的视图或者其父类显示上下文菜单。

大部分情况下，子类不需要重写该方法。但是，如果子类被直接添加到窗口管理器（例如：addView(View.android.view.ViewGroup.LayoutParams)），此时就需要重写来显示上下文菜单

参数

originalView　上下文菜单初次调用的源视图

返回值

如果上下文菜单被显示了 则返回 true。

受保护方法

（21）protected int **computeHorizontalScrollExtent** ()

在水平范围内计算滚动条滑块的滚动范围。该值用来计算滚动条滑块的长度。

该范围可以使用任意的单位但是必须跟 computeHorizontalScrollRange()和 computeHorizontalScrollOffset()的单位保持一致。

默认范围是视图的宽度。

返回值

滚动条滑块的水平滚动范围

（22）protected int **computeHorizontalScrollOffset** ()

在水平范围内计算滚动条滑块的偏移量。该值用来计算水平滑块的位置。

该范围可以使用任意的单位但是必须跟 computeHorizontalScrollRange()和 computeHorizontalScrollExtent()的单位保持一致。

默认偏移量是视图的偏移量。

返回值

滚动条滑块的水平偏移量。

（23）protected int **computeHorizontalScrollRange** ()

计算滚动条水平方向上的滚动范围。

该范围可以使用任意的单位但是必须跟 computeHorizontalScrollExtent()和 computeHorizontalScrollOffset()的单位保持一致。

返回值

水平滚动条代表的滑动总范围。

（24）protected void **dispatchSetPressed** (boolean pressed)

分发 setPressed 到 View 的子类。

参数

pressed 新按下的状态

（25）protected ViewGroup.LayoutParams **generateDefaultLayoutParams** ()

返回默认的布局参数。当 View 作为参数传递给 addView(View)而没有布局参数时就会请求这些参数。如果返回 null，则 addView 会抛出异常。

返回值

默认的布局参数或 null

(26)protected ViewGroup.LayoutParams **generateLayoutParams** (ViewGroup.LayoutParams p)

返回一组合法的受支持的布局参数。当把 ViewGroup 传递给 View 而该 View 的布局参数并没有通过 checkLayoutParams(android.view.ViewGroup.LayoutParams)的测试时，就会调用该方法。该方法应该返回一组适合该 ViewGroup 的新的布局参数，该过程可能需要从指定的一组布局参数中复制相关的属性。

参数

p 被转换成适合该 ViewGroup 的一组参数。

返回值

返回一个 ViewGroup.LayoutParams 的实例或者一个它的子类。

(27)protected int **getChildDrawingOrder** (int childCount, int i)

返回迭代的绘制子类索引。如果你想改变子类的绘制顺序就要重写该方法。默认返回 i 值。

提示：为了能够调用该方法，你必须首先调用 setChildrenDrawingOrderEnabled(boolean)来允许子类排序。

参数

childCount 子类个数

i 当前迭代顺序

返回值

绘制该迭代子类的索引

(28)protected boolean **getChildStaticTransformation** (View child, Transformation t)

（译者注：setStaticTransformationsEnabled 这个属性设成 true 的时候每次 viewGroup(看 Gallery 的源码就可以看到它是从 ViewGroup 间接继承过来的)在重新画它的 child 的时候都会促发 getChildStaticTransformation 这个函数。这里 1、这里 2）

(29)protected ContextMenu.ContextMenuInfo **getContextMenuInfo** ()

Views 如果有额外的信息跟上下文菜单有联系的话就需要实现该方法。返回的结果被用作回调方法 onCreateContextMenu(ContextMenu, View, ContextMenuInfo)的参数。

返回值

显示上下文菜单的条目的额外信息。这些信息将会改变 View 不同的子类

(30)protected void **onFocusChanged** (boolean gainFocus, int direction, Rect previouslyFocusedRect)

当该视图的焦点状态发生改变时将会调用视图系统。当导向的方向触发焦点事件时，方向和先前获得焦点的矩形提供焦点事件的来源。当用户重写该方法，必须调用父类方法来触发标准的焦点处理事件。

参数

gainFocus 如果 View 获得焦点为 true，否则 false

direction 当调用 requestFocus()方法来给该视图焦点时焦点的移动方向。该值:FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT 或 FOCUS_RIGHT。该参数不常用，通常使用它的默认值。

previouslyFocusedRect 该视图坐标系统中先前获得焦点的视图的矩形。如果适用，这将获得焦点事件来源的更细致的信息（除了方向以外）。否则为 null。

(31)protected void **onLayout** (boolean changed, int l, int t, int r, int b)

当视图为每一个子类分配大小和位置时从布局中调用该方法。有子类的派生类应该重写该方法在子类中调用布局。

参数

changed 该视图新的大小和位置。

l 相对父容器的左侧位置

t 相对父容器的顶部位置

r 相对父容器的右侧位置

b 相对父容器的底部位置

22.4 Gallery 的使用

在 XML 中声明一个 Gallery：

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
        <Gallery xmlns:android="http://schemas.android.com/apk/res/android" android:id="@+id/gallery"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:background="?android:galleryItemBackground"
    />
```

然后需要声明一个存放图片的容器 ImageAdapter，之后就可以通过 setAdapter 方法把资源添加到 Gallery 中来显示，并设置好监听事件处理：

```
gallery = (Gallery)findViewById(R.id.gallery);
                gallery.setAdapter(new ImageAdapter(this));//设置图片适配器
                 //设置监听器
                 gallery.setOnItemClickListener(new OnItemClickListener() {
                  @Override
                  public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,   long arg3) {
                     Toast.makeText(MyGallery.this, "点击了第"+arg2+"张图片", Toast.LENGTH_LONG).show();
                  }
               });
              }
            }
```

声明一个存放图片的容器 ImageAdapter：

```
class ImageAdapter extends BaseAdapter{
  private Context context;
  //图片源数组
  private Integer[] imageInteger={
        R.drawable.gallery_photo_1,
        R.drawable.gallery_photo_2,
        R.drawable.gallery_photo_3,
        R.drawable.gallery_photo_4,
        R.drawable.gallery_photo_5,
        R.drawable.gallery_photo_6,
        R.drawable.gallery_photo_7,
        R.drawable.gallery_photo_8
  };
  public ImageAdapter(Context c){
    context = c;
  }
  @Override
  public int getCount() {
    return imageInteger.length;
  }
  @Override
  public Object getItem(int position) {
    return position;
  }
  @Override
  public long getItemId(int position) {
    // TODO Auto-generated method stub
    return position;
  }
  @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    ImageView imageView = new ImageView(context);
    imageView.setImageResource(imageInteger[position]);
    imageView.setScaleType(ImageView.ScaleType.FIT_XY);
    imageView.setLayoutParams(new Gallery.LayoutParams(136, 88));
    return imageView;
  }}
```

22.5 补充

补充

文章精选

- Android 开发——使用 Gallery 实现“多级联动”
- android 图片拖动效果(Gallery)

23 ImageSwitcher 切换图片

23.1 ImageSwitcher 类概述

ImageSwitcher 是 Android 中控制图片展示效果的一个控件，如：幻灯片效果...，颇有感觉啊，做相册一绝。

ImageSwitcher 类的层次关系如下：

public class ImageSwitcher extends ViewSwitcher

java.lang.Object
android.view.View
android.view.ViewGroup
android.widget.FrameLayout
android.widget.ViewAnimator
android.widget.ViewSwitcher
android.widget.ImageSwitcher

23.2 ImageSwitcher 常用的方法

方法	功能描述	返回值
ImageSwitcher	两个构造方法： ImageSwitcher（Context contex） ImageSwitcher（Contex contex，AttributeSet attrs）	null
setImageDrawable（Drawable drawable）	设置图片为参数 drawable 指定的资源	void
setImageResource（int resid）	设置图片为参数 resid 所指定的资源。resid 是编译器为资源生成唯一标识，例如 R.id.pcil	void
setInAnimation（Context context，in id）	设置图片进入屏幕时所显示的动画	void
setOutAnimation（Context context，in id）	设置图片退出屏幕时，所显示的动画	void
setImageURI（Uri uri）	设置图片为参数 uri（例如一个图片的 web 链接）所指定的资源	void
addView（View child，int index，ViewGroup.LayoutParams params）	继承自 ViewSwitcher 的方法，用于添加视图。child 是要添加的视图，params 是视图的布局	View
getNextView()	继承自 ViewSwitcher 的方法，用于获取视图	

getCurrentView()	继承自 ViewAnimator 的方法，用于获取视图	View
getDisplayedChild();	当前显示的子视图的索引	int
removeAllViews()	继承在 ViewAnimator 的方法，用于移除所有的子视图	void
removeViewAt(int index)	继承自 ViewAnimator 的方法，用于移除 index 所指定的视图	void
setAnimateFirstView(boolean animate)	继承自 ViewAnimator 的方法，用于设置视图是否在首次显示时播放动画	void
removeViews(int start,int count)	继承自 ViewAnimator 的方法，用于移除自 start 的 count 个子视图	void
showNext()	继承自 ViewAnimator 的方法，由于显示下一张图片	void

23.3 ImageSwitcher 的使用

我们直接在 Activity 中实现，继承监听 OnClickListenner 和 ViewFactory 接口：

```
public class ImageSwitcherActivity extends Activity implements OnClickListener,ViewFactory
 {
/* 所有要显示的图片资源索引 */
    private static final Integer[] imagelist =
    {
        R.drawable.img1,
        R.drawable.img2,
        R.drawable.img3,
        R.drawable.img4,
        R.drawable.img5,
        R.drawable.img6,
        R.drawable.img7,
        R.drawable.img8,
    };

    //创建ImageSwitcher对象
    private ImageSwitcher          m_Switcher;
    //索引
    private static int             index          = 0;

    //“下一页”按钮ID
    private static final int       BUTTON_DWON_ID     = 0x123456;
    //“上一页”按钮ID
    private static final int       BUTTON_UP_ID     = 0x123457;
    //ImageSwitcher对象的ID
    private static final int       SWITCHER_ID     = 0x123458;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        //创建一个线性布局LinearLayout
```

```
        LinearLayout main_view = new LinearLayout(this);
        //创建ImageSwitcher对象
        m_Switcher = new ImageSwitcher(this);
        //在线性布局中添加ImageSwitcher视图
        main_view.addView(m_Switcher);
        //设置ImageSwitcher对象的ID
        m_Switcher.setId(SWITCHER_ID);
        //设置ImageSwitcher对象的数据源
        m_Switcher.setFactory(this);
        m_Switcher.setImageResource(imagelist[index]);

        //设置显示上面创建的线性布局
        setContentView(main_view);

        //创建“下一张”按钮
        Button next = new Button(this);
        next.setId(BUTTON_DWON_ID);
        next.setText("下一张");
        next.setOnClickListener(this);
        LinearLayout.LayoutParams param = new LinearLayout.LayoutParams(100, 100);
        main_view.addView(next, param);

        //创建“上一张”按钮
        Button pre = new Button(this);
        pre.setId(BUTTON_UP_ID);
        pre.setText("上一张");
        pre.setOnClickListener(this);
        main_view.addView(pre, param);

    }

    //事件监听、处理
    public void onClick(View v)
    {
        switch (v.getId())
        {
            //下一页
            case BUTTON_DWON_ID:
                index++;
                if (index >= imagelist.length)
                {
                    index = 0;
                }
                //ImageSwitcher对象资源索引
                m_Switcher.setImageResource(imagelist[index]);
                break;
            //上一页
            case BUTTON_UP_ID:
                index--;
```

```
                if (index < 0)
                {
                    index = imagelist.length - 1;
                }
                //ImageSwitcher对象资源索引
                m_Switcher.setImageResource(imagelist[index]);
                break;
            default:
                break;
        }
    }

    public View makeView()
    {
        //将所有图片通过ImageView来显示
        return new ImageView(this);
    }
```

实现的效果如下所示：

23.4 补充资料

补充

文章链接

[Android ImageSwitcher]
[Image Switcher View | Android Developer Tutorial]

24 GridView 网络视图

24.1 GridView 类概述

GridView 是一个在平面上可显示多个条目的可滚动的视图组件，该组件中的条目通过一个 ListAdapter 和该组件进行关联。比如 android 手机中显示的应用：

GridView 类的层次关系如下：

public final class GridView extends AbsListView

java.lang.Object
    android.view.View
        android.view.ViewGroup
            android.widget.AdapterView<T extends android.widget.Adapter>
        android.widget.AbsListView
            android.widget.GridView

24.2 GridView 常用属性

XML 属性

属性名称	描述
android:columnWidth	设置列的宽度。关联的方法为：setColumnWidth(int)
android:gravity	设置此组件中的内容在组件中的位置。可选的值有：top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical 可以多选，用“|”分开。关联方法：setGravity (int gravity)
android:horizontalSpacing	两列之间的间距。关联方法：setHorizontalSpacing(int)
android:numColumns	列数。关联方法：setNumColumns(int)
android:stretchMode	缩放模式。关联方法：setStretchMode(int)
android:verticalSpacing	两行之间的间距。关联方法：setVerticalSpacing(int)

24.3 GridView 常用的方法

构造函数

（1）public GridView (Context context)
创建一个默认属性的 GridView 实例
public GridView (Context context, AttributeSet attrs)
创建一个带有 attrs 属性的 GridView 实例
（2）public GridView (Context context, AttributeSet attrs, int defStyle)
创建一个带有 attrs 属性，并且指定其默认样式的 GridView 实例

公共方法

（3）public ListAdapter getAdapter ()
获得与此组件相关的适配器..
返回值
ListAdapter 适配器实例

（4）public int getStretchMode ()
获得 GridView 的缩放模式..

（5）public boolean onKeyDown (int keyCode, KeyEvent event)
默认由 KeyEvent.Callback.onKeyMultiple（）实现，如果视图是可用的并且是可点击的，那么传入 KEYCODE_DPAD_CENTER 或 KEYCODE_ENTER 值是执行的是按下视图操作。
参数

keyCode 一个表示按下操作的键值.
event 表示按钮事件的对象.
返回值
如果你认为已经完成事件处理,不想让让下一个处理器来处理此事件，则返回 true,否则返回 false。

（6）public boolean onKeyMultiple (int keyCode, int repeatCount, KeyEvent event)
默认由 KeyEvent.Callback.onKeyMultiple()实现，总是返回 false（不处理此事件）。
参数
keyCode 键值.
repeatCount 该动作发生的次数.
event 事件对象.
返回值
如果你认为已经完成事件处理,不想让让下一个处理器来处理此事件，则返回 true,否则返回 false。

（7）public boolean onKeyUp (int keyCode, KeyEvent event)
默认由 KeyEvent.Callback.onKeyMultiple（）实现，如果视图是可用的并且是可点击的，那么传入 KEYCODE_DPAD_CENTER 或 KEYCODE_ENTER 值是执行的是点击视图操作。
参数
keyCode 键值.
event 事件对象.
返回值
如果你认为已经完成事件处理,不想让让下一个处理器来处理此事件，则返回 true,否则返回 false。

（8）public void setAdapter (ListAdapter adapter)
设置 GridView 的数据。
参数
adapter 为 grid 提供数据的适配器

（9）public void setColumnWidth (int columnWidth)
设置 GridView 的列宽.
参数
columnWidth 列的宽度，以像素为单位

（10）public void setGravity (int gravity)
设置控件内容的位置，默认值为：Gravity.LEFT.
参数
gravity 位置值

（11）public void setHorizontalSpacing (int horizontalSpacing)
设置列间距.
参数
horizontalSpacing 列间距值

（12）public void setNumColumns (int numColumns)
设置 grid 的列数
参数
numColumns 列数值.

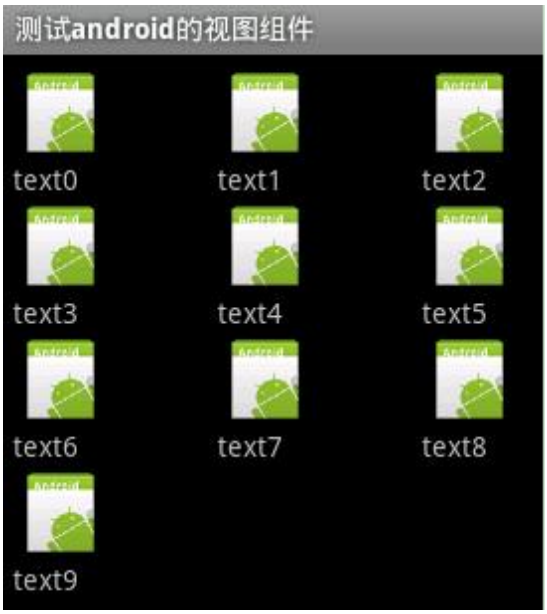
（13）public void setSelection (int position)
设置选中的条目.
参数
position 数据条目在列表中的索引值（从 0 开始），如果在可触摸的模式下，在该索引值下的条目将不会被选中，但是该索引值仍然指向该条目。

（14）public void setStretchMode (int stretchMode)
设置 grid 中的条目以什么缩放模式去填充空间。.
参数
stretchMode 可选值：NO_STRETCH，STRETCH_SPACING，STRETCH_SPACING_UNIFORM，或 STRETCH_COLUMN_WIDTH

（15）public void setVerticalSpacing (int verticalSpacing)
设置行间距.
参数
verticalSpacing 间距值，以像素为单位..

24.4 GridView 的简单使用

现在要实现如下图所示的效果：



我们声明两个 xml 文件，一个是 gridview.xml 声明一个 GridView：

gridview.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content">
   <GridView android:id="@+id/mygridview"
      android:numColumns="3"
      android:gravity="center_horizontal"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content">
   </GridView>
</LinearLayout>
```

一个是 Grid_item.xml，包含要显示的一个 ImageView 和一个 TextView

grid_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout android:id="@+id/RelativeLayout01"
   android:layout_width="fill_parent" android:layout_height="fill_parent"
   xmlns:android="http://schemas.android.com/apk/res/android">
   <ImageView android:id="@+id/image_item"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content">
   </ImageView>
   <TextView android:id="@+id/text_item"
      android:layout_below="@+id/image_item"
      android:layout_height="wrap_content"
      android:layout_width="wrap_content">
   </TextView>
</RelativeLayout>
```

然后在 Activity 中就可以实现显示了：

```
            //准备要添加的数据条目
      List<Map<String, Object>> items = new ArrayList<Map<String,Object>>();
      for (int i = 0; i < 10; i++) {
        Map<String, Object> item = new HashMap<String, Object>();
        item.put("imageItem", R.drawable.icon);
        item.put("textItem", "text" + i);
        items.add(item);
      }
//实例化一个适配器
```

```
    SimpleAdapter adapter = new SimpleAdapter(this, items, R.layout.grid_item, new String[]{"imageItem", "textItem"}, new int[]{R.id.image_item, R.id.text_item});
    //获得 GridView 实例
    gridview = (GridView)findViewById(R.id.mygridview);
    //gridview.setNumColumns(3);//可以在 xml 中设置
    //gridview.setGravity(Gravity.CENTER);//同上
//将 GridView 和数据适配器关联
    gridview.setAdapter(adapter);
```

25 ScrollView 卷轴视图

25.1 ScrollView 类概述

ScrollView 是一种可供用户滚动的层次结构布局容器，允许显示比实际多的内容。ScrollView 是一种 FrameLayout，意味需要在其上放置有自己滚动内容的子元素。子元素可以是一个复杂的对象的布局管理器。通常用的子元素是垂直方向的 LinearLayout，显示在最上层的垂直方向可以让用户滚动的箭头。

TextView 类也有自己的滚动功能，所以不需要使用 ScrollView，但是只有两个结合使用，才能保证显示较多内容时候的效率。但只有两者结合使用才可以实现在一个较大的容器中一个文本视图效果。

ScrollView 只支持垂直方向的滚动。

ScrollView 类的结构如下：

继承关系

public class ScrollView extends FrameLayout

```
java.lang.Object
    android.view.View
        android.view.ViewGroup
            android.widget.FrameLayout
                android.widget.ScrollView
```

25.2 ScrollView 常用的方法

构造函数

（1）public **ScrollView** (Context context)

创建一个默认属性的 ScrollView 实例。

（2）public **ScrollView** (Context context, AttributeSet attrs)

创建一个带有 attrs 属性的 ScrollView 实例。

(3)public **ScrollView** (Context context, AttributeSet attrs, int defStyle)

创建一个带有 attrs 属性，并且指定其默认样式的 ScrollView 实例。

公共方法

(4)public void **addView** (View child)

添加子视图。如果事先没有给子视图设置 layout 参数，会采用当前 ViewGroup 的默认参数来设置子视图。

参数

child　　所添加的子视图

(5)public void **addView** (View child, int index)

添加子视图。如果事先没有给子视图设置 layout 参数，会采用当前 ViewGroup 的默认参数来设置子视图。

参数

child　　所添加的子视图

index　　添加子视图的位置

(6)public void **addView** (View child, int index, ViewGroup.LayoutParams params)

根据指定的 layout 参数添加子视图

参数

child　　所添加的子视图

index　　添加子视图的位置

params　　为子视图设置的 layout 参数

(7)public void **addView** (View child, ViewGroup.LayoutParams params)
根据指定的 layout 参数添加子视图。
参数
child 所添加的子视图
params 为子视图设置的 layout 参数

(8)public boolean **arrowScroll** (int direction)
响应点击上下箭头时对滚动条滚动的处理。
参数
direction 按下的箭头所对应的方向
返回值
如果我们处理（消耗）了此事件返回 true，否则返回 false。

(9) public void **computeScroll** ()
被父视图调用，用于必要时候对其子视图的值（mScrollX 和 mScrollY）进行更新。典型的情况如：父视图中某个子视图使用一个 Scroller 对象来实现滚动操作，会使得此方法被调用。

(10)public boolean **dispatchKeyEvent** (KeyEvent event)
发送一个 key 事件给当前焦点路径的下一个视图。此焦点路径从视图树的顶层执行直到当前焦点视图。如果此视图为焦点视图，将为自己发送。否则，会为当前焦点路径的下一个节点发送。此方法也会激起一个 key 监听器。
参数
event 发送的 key 事件
返回值
事件被处理返回 true，否则返回 false。

(11)public void **draw** (Canvas canvas)
手动绘制视图（及其子视图）到指定的画布(Canvas)。这个视图必须在调用这个函数之前做好了整体布局。当实现一个视图时，不需要继承这个方法；相反，你应该实现 onDraw(Canvas)方法。
参数
canvas 绘制视图的画布

(12)public boolean **executeKeyEvent** (KeyEvent event)
当接收到 key 事件时，用户可以调用此函数来使滚动视图执行滚动，类似于处理由视图体系发送的事件。
参数
event 需要执行 key 的事件
返回值
事件被处理返回 true，否则返回 false。

(13)public void **fling** (int velocityY)
滚动视图的滑动（fling）手势。（译者注： 如何监听 android 的屏幕滑动停止事件）
参数
velocityY Y 方向的初始速率。正值表示手指/光标向屏幕下方滑动，而内容将向上滚动。

(14)public boolean **fullScroll** (int direction)
对响应“home/end”短按时响应滚动处理。此方法将视图滚动到顶部或者底部，并且将焦点置于新的可视区域的最顶部/最底部组件。若没有适合的组件做焦点，当前的 ScrollView 会收回焦点。
参数
direction 滚动方向：FOCUS_UP 表示视图向上滚动；FOCUS_DOWN 表示视图向下滚动
返回值
若 key 事件被消耗(consumed)返回 true，其他情况返回 false。

(15)public int **getMaxScrollAmount** ()
返回值
当前滚动视图响应箭头事件能够滚动的最大数。

(16)public boolean **isFillViewport** ()
指示当前 ScrollView 的内容是否被拉伸以填充视图可视范围（译者注：viewport 可视范围，参见决定 Scrollviewer 里面 Control 的可视范围）。
返回值
内容填充视图返回 true，否则返回 false。

(17)public boolean **isSmoothScrollingEnabled** ()
返回值
按箭头方向滚动时，是否显示滚动的平滑效果。

(18)public boolean **onInterceptTouchEvent** (MotionEvent ev)
实现此方法是为了拦截所有触摸屏幕时的运动事件。可以像处理发送给子视图的事件一样去监视这些事件，并且获取当前手势在任意点的 ownership

使用此方法时候需要注意，因为它与 View.onTouchEvent(MotionEvent)有相当复杂的交互，并且前提需要正确执行 View.onTouchEvent(MotionEvent)。事件将按照如下顺序接收到：

1. 收到 down 事件
2. Down 事件或者由视图组的一个子视图处理，或者被用户自己的 onTouchEvent()方法处理；此处理意味你应该执行 onTouchEvent()时返回 true，这样才能继续看到剩下的手势（取代找一个父视图处理）。如果 onTouchEvent()返回 true 时，你不会收到 onInterceptTouchEvent()的任何事件并且所有对触摸的处理必须在 onTouchEvent()中发生。
3. 如果此方法返回 false，接下来的事件（up to and including the final up）将最先被传递当此，然后是目标的 onTouchEvent()。
4. 如果返回 true，将不会收到以下任何事件：目标 view 将收到同样的事件但是会伴随 ACTION_CANCEL，并且所有的更进一步的事件将会传递到你自己的 onTouchEvent()方法中而不会再在这里出现。

参数

ev 体系向下发送的动作事件

返回值

如果将运动事件从子视图中截获并且通过 onTouchEvent()发送到当前 ViewGroup ，返回 true。当前目标将会收到 ACTION_CANCEL 事件，并且不再会有其他消息传递到此。

（译者注：onInterceptTouchEvent 和 onTouchEvent 调用时序）

(19)public boolean **onTouchEvent** (MotionEvent ev)

执行此方法为了处理触摸屏幕的运动事件。

参数

ev 运动事件

返回值

事件被处理返回 true，其它返回 false。

(20)public boolean **pageScroll** (int direction)

响应短按“page up/ down”时候对滚动的处理。此方法将向上或者向下滚动一屏，并且将焦点置于新可视区域的最上/最下。如果没有适合的 component 作为焦点，当前 scrollView 将收回焦点。

参数

direction 滚动方向：FOCUS_UP 表示向上翻一页，FOCUS_DOWN 表示向下翻一页。

返回值

此 key 事件被消耗(cosumed)返回 true，其他返回 false。

(21)public void **requestChildFocus** (View child, View focused)

当父视图的一个子视图的要获得焦点时，调用此方法。

参数

child 要获得焦点的父视图的子视图。此视图包含了焦点视图。如果没有特殊徐要求，此视图实际上就是焦点视图。

focused 子视图的子孙视图并且此子孙视图是真正的焦点视图

(22)public boolean **requestChildRectangleOnScreen** (View child, Rect rectangle, boolean immediate)

当组里的某个子视图需要被定位在屏幕的某个矩形范围时，调用此方法。重载此方法的 ViewGroup 可确认以下几点：

* 子项目将是组里的直系子项
* 矩形将在子项目的坐标体系中

重载此方法的 ViewGroup 应该支持以下几点：

* 若矩形已经是可见的，则没有东西会改变
* 为使矩形区域全部可见，视图将可以被滚动显示

参数

child 发出请求的子视图

rectangle 子项目坐标系内的矩形，即此子项目希望在屏幕上的定位

immediate 设为 true，则禁止动画和平滑移动滚动条

返回值

进行了滚动操作的这个组（group），是否处理此操作。

(23)public void **requestLayout** ()

当有改变引起当前视图重新布局时，调用此函数。它将规划一个视图树的 layout 路径。

(24)public void **scrollTo** (int x, int y)

设置当前视图滚动到的位置。此函数会引起对 onScrollChanged(int, int, int, int)函数的调用并且会让视图更新。

当前版本取消了在子视图中的滚动。

参数

x 滚动到的 X 位置

y 滚动到的 Y 位置

(25)public void **setFillViewport** (boolean fillViewport)

设置当前滚动视图是否将内容高度拉伸以填充视图可视范围（译者注：viewport 可视范围，参见决定 Scrollviewer 里面 Control 的可视范围）。

参数

fillViewport　设置为 true 表示拉伸内容高度来适应视口边界；其他设为 false。

(26)public void **setOverScrollMode** (int mode)
为视图设置 over-scroll 模式。有效的 over-scroll 模式有 OVER_SCROLL_ALWAYS（缺省值），OVER_SCROLL_IF_CONTENT_SCROLLS（只允许当视图内容大过容器时，进行 over-scrolling）和 OVER_SCROLL_NEVER。只有当视图可以滚动时，此项设置才起作用。
（译者注：这个函数是 2.3 r1 中新增的，API Level 9。关于 over-scroll 这里译为弹性滚动，即，参见帖子：类似 iPhone 的弹性 ListView 滚动）
参数
mode　The new over-scroll mode for this view.

(27)public void **setSmoothScrollingEnabled** (boolean smoothScrollingEnabled)
用来设置箭头滚动是否可以引发视图滚动。
参数
smoothScrollingEnabled　设置箭头滚动是否可以引起内容的滚动的 bool 值

(28)public final void **smoothScrollBy** (int dx, int dy)
类似于 scrollBy(int, int)，但是滚动时候是平缓的而不是立即滚动到某处。
参数
dx　在 X 方向滚动的像素数
dy　在 Y 方向滚动的像素数

(29)public final void **smoothScrollTo** (int x, int y)
类似于 scrollTo(int, int)，但是滚动时候是平缓的而不是立即滚动到某处。
参数
x　要滚动到位置的 X 坐标
y　要滚动到位置的 Y 坐标

受保护方法

(30)protected int **computeScrollDeltaToGetChildRectOnScreen** (Rect rect)
计算 X 方向滚动的总合，以便在屏幕上显示子视图的完整矩形（或者，若矩形宽度超过屏幕宽度，至少要填满第一个屏幕大小）。
参数
rect　矩形
返回值
滚动差值

(31)protected int **computeVerticalScrollOffset** ()
计算垂直方向滚动条的滑块的偏移。此值用来计算滚动条轨迹的滑块的位置。
范围可以以任意单位表示，但是必须与 computeVerticalScrollRange()和 computeVerticalScrollExtent()的单位一致。
缺省的偏移是在当前视图滚动的偏移。
返回值
滚动条的滑块垂直方向的偏移。

(32)protected int **computeVerticalScrollRange** ()
滚动视图的可滚动范围是所有子元素的高度。
返回值
由垂直方向滚动条代表的所有垂直范围，缺省的范围是当前视图的画图高度。

(33)protected float **getBottomFadingEdgeStrength** ()
返回滚动底部的能见度。能见度的值的范围是 0.0（没有消失）到 1.0（完全消失）之间。缺省的执行返回值为 0.0 或者 1.0，而不是他们中间的某个值。滚动时子类需要重载这个方法来提供一个平缓的渐隐的实现。
返回值
滚动底部能见度，值的范围在浮点数 0.0f 到 1.0f 之间。

(34)protected float **getTopFadingEdgeStrength** ()
返回滚动顶部的能见度。能见度的值的范围是 0.0（没有消失）到 1.0（完全消失）之间。缺省的执行返回值为 0.0 或者 1.0，而不是他们中间的某个值。滚动时子类需要重载这个方法来提供一个平缓的渐隐的实现。
返回值
滚动顶部能见度，值的范围在浮点数 0.0f 到 1.0f 之间。

(35)protected void **measureChild** (View child, int parentWidthMeasureSpec, int parentHeightMeasureSpec)
要求当前视图的一个子视图测量自己，同时兼顾到当前视图的 MeasureSpec 的要求和它的空白。子视图必须有 MarginLayoutParams。比较复杂的工作是在 getChildMeasureSpec 中完成的。
参数
child　需要自己测量的子视图
parentWidthMeasureSpec　当前视图要求的宽度
parentHeightMeasureSpec　当前视图要求的宽度

(36)protected void **measureChildWithMargins** (View child, int parentWidthMeasureSpec, int widthUsed, int parentHeightMeasureSpec, int heightUsed)

要求当前视图的一个子视图测量自己，同时兼顾到当前视图的 MeasureSpec 的要求和它的空白和边界。子视图必须有 MarginLayoutParams。比较复杂的工作是在 getChildMeasureSpec 中完成的。

参数

child 需要测量的子视图

parentWidthMeasureSpec 当前视图要求的宽度

widthUsed 水平方向上由父视图使用的空白 (也可能是视图的其他子视图使用的)

parentHeightMeasureSpec 当前视图要求的宽度

heightUsed 垂直方向上由父视图使用的空白 (也可能是视图的其他子视图使用的)

(37)protected void **onLayout** (boolean changed, int l, int t, int r, int b)

当前视图需要为子视图分配大小和位置时候调用，子类继承必须要重载此方法并调用自己子视图的 layout 函数。

参数

changed 当前视图的新的大小或者位置

l 相对父视图，左边界位置

t 相对父视图，上边界位置

r 相对父视图，右边界位置

b 相对父视图，下边界位置

(38)protected void **onMeasure** (int widthMeasureSpec, int heightMeasureSpec)

测量视图以确定其内容宽度和高度。此方法被 measure(int, int)调用。需要被子类重写以提供对其内容准确高效的测量。

约定：当重写此方法时，你必须调用 setMeasuredDimension(int, int)来保存当前视图 view 的宽度和高度。不成功调用此方法将会导致一个 IllegalStateException 异常，是由 measure(int, int)抛出。所以调用父类的 onMeasure(int, int)方法是必须的。

父类的实现是以背景大小为默认大小，除非 MeasureSpec（测量细则）允许更大的背景。子类可以重写 onMeasure(int,int)以对其内容提供更佳的尺寸。

如果此方法被重写，那么子类的责任是确认测量高度和测量宽度要大于视图 view 的最小宽度和最小高度（getSuggestedMinimumHeight() 和 getSuggestedMinimumWidth()），使用这两个方法可以取得最小宽度和最小高度。

参数

widthMeasureSpec 受主窗口支配的水平空间要求。这个需求通过 View.MeasureSpec.进行编码。

heightMeasureSpec 受主窗口支配的垂直空间要求。这个需求通过 View.MeasureSpec.进行编码。

(39)protected void **onOverScrolled** (int scrollX, int scrollY, boolean clampedX, boolean clampedY)

被 overScrollBy(int, int, int, int, int, int, int, int, boolean)调用，来对一个 over-scroll 操作的结果进行响应。（译者注：这个函数是 2.3 r1 中新增的，API Level 9）

参数

scrollX 新的 X 滚动像素值

scrollY 新的 Y 滚动像素值

clampedX 当 scrollX 被 over-scroll 的边界限制时，值为 true

clampedY 当 scrollY 被 over-scroll 的边界限制时，值为 true

(40)protected boolean **onRequestFocusInDescendants** (int direction, Rect previouslyFocusedRect)

当在滚动视图的子视图中查找焦点视图时，需要注意不要将焦点设置在滚动出屏幕外的控件上。此方法会比执行缺省的 ViewGroup 代价高，否则此行为也会设置为缺省

参数

direction 指定下列常量之一：FOCUS_UP, FOCUS_DOWN, FOCUS_LEFT, FOCUS_RIGHT

previouslyFocusedRect 能够给出一个较好的提示的矩形（当前视图的坐标系统）表示焦点从哪里得来。如果没有提示为 null。

返回值

是否取得了焦点

(41)protected void **onSizeChanged** (int w, int h, int oldw, int oldh)

布局期间当视图的大小发生改变时调用。如果只是添加到视图，调用时显示的是旧值 0。（译者注：也就是添加到视图时，oldw 和 oldh 返回的是 0）。

参数

w 视图当前宽度

h 视图当前高度

oldw 视图改变前的宽度

oldh 视图改变前的高度

25.3 ScrollView 的几个属性

（1）android:scrollbarFadeDuration

设置滚动条淡出效果（从有到慢慢的变淡直至消失）时间，以毫秒为单位。Android2.2 中滚动条滚动完之后会消失，再滚动又会出来，在 1.5、1.6 版本里面会一直显示着。

（2）android:scrollbarSize

设置滚动条的宽度。

（3）android:scrollbarStyle

（4）设置滚动条的风格和位置。设置值：insideOverlay、insideInset、outsideOverlay、outsideInset。这里没有试出太多效果。

（5）android:scrollbarThumbHorizontal

设置水平滚动条的 drawable。

(6)android:scrollbarThumbVertical

设置垂直滚动条的 drawable.

(7)android:scrollbarTrackHorizontal

设置水平滚动条背景（轨迹）的色 drawable

(8)android:scrollbarTrackVertical

设置垂直滚动条背景（轨迹）的色 drawable

（9）android:background

设置背景色/背景图片。可以通过以下两种方法设置背景为透明："@android:color/transparent"和"@null"。注意 TextView 默认是透明的，不用写此属性，但是 Buttom/ImageButton/ImageView 想透明的话就得写这个属性了

25.4 ScrollView 的使用

ScrollView 卷轴视图是指当拥有很多内容，一屏显示不完时，需要通过滚动跳来显示的视图.的使用，因此我们这里利用动态的增加 Button，当一屏显示不完时，就可以看到滚动的效果了。
首先我们在 xml 布局中作如下声明，

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/ScrollView" android:layout_width="fill_parent"
 android:layout_height="wrap_content" android:scrollbars="vertical">
 <LinearLayout android:id="@+id/LinearLayout"
  android:orientation="vertical" android:layout_width="fill_parent"
  android:layout_height="wrap_content">
  <TextView android:id="@+id/TestView" android:layout_width="fill_parent"
   android:layout_height="wrap_content" android:text="TestView0" />
```

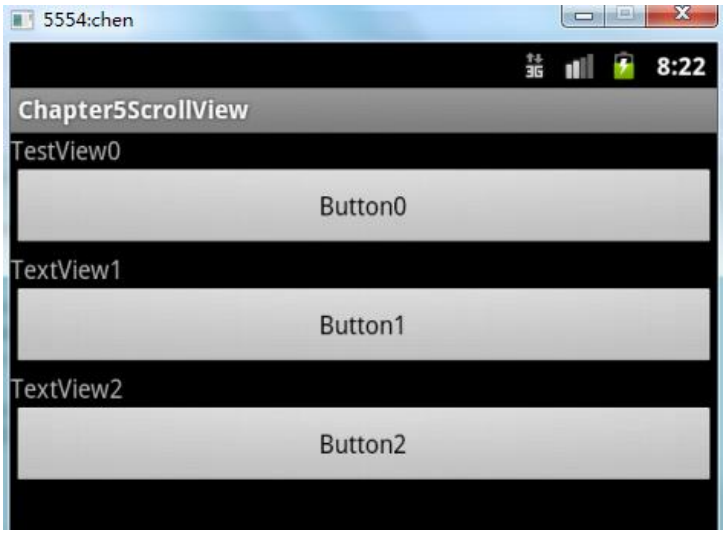
```
    <Button android:id="@+id/Button" android:text="Button0" android:layout_width="fill_parent"
     android:layout_height="wrap_content"></Button>
  </LinearLayout>
</ScrollView>
```

然后在 Activity 中实现显示并创建监听：

```
// 创建一个线性布局
    mLayout = (LinearLayout) this.findViewById(R.id.LinearLayout);
    // 创建一个ScrollView对象
    sView = (ScrollView) this.findViewById(R.id.ScrollView);
    Button mBtn = (Button) this.findViewById(R.id.button_01);
    mBtn.setOnClickListener(mClickListener);// 添加点击事件监听
  }

  public boolean onKeyDown(int keyCode, KeyEvent event){
    Button b = (Button) this.getCurrentFocus();
    int count = mLayout.getChildCount();
    Button bm = (Button) mLayout.getChildAt(count-1);

    if(keyCode==KeyEvent.KEYCODE_DPAD_UP && b.getId()==R.id.button_01){
      bm.requestFocus();
      return true;
    }else if(keyCode==KeyEvent.KEYCODE_DPAD_DOWN && b.getId()==bm.getId()){
      this.findViewById(R.id.button_01).requestFocus();
      return true;
    }
    return false;
  }
  // Button事件监听,当点击第一个按钮时增加一个button和一个textview
  private Button.OnClickListener mClickListener = new Button.OnClickListener() {

    private int index = 1;

    @Override
    public void onClick(View v) {
      TextView tView = new TextView(ScrollViewActivity.this);//定义一个TextView
      tView.setText("TextView" + index);//设置TextView的文本信息
      //设置线性布局的属性
      LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
          LinearLayout.LayoutParams.FILL_PARENT,
          LinearLayout.LayoutParams.WRAP_CONTENT);
      mLayout.addView(tView, params);//添加一个TextView控件
      Button button = new Button(ScrollViewActivity.this);//定义一个Button
      button.setText("Button" + index);//设置Button的文本信息
      button.setId(index++);
      mLayout.addView(button, params);//添加一个Button控件
      mHandler.post(mScrollToButton);//传递一个消息进行滚动
    }
```

```
};
private Runnable mScrollToButton = new Runnable() {

  @Override
  public void run() {
    int off = mLayout.getMeasuredHeight() - sView.getHeight();
    if (off > 0) {
      sView.scrollTo(0, off);//改变滚动条的位置
    }
  }

};
```

显示的效果如下：



25.5 补充资料

文章精选

Android ApiDemos/ScrollView2 添加自动滚动和智能焦点切换
[Android 学习指南]使用 ScrollView 实现滚动效果
Android 中 ScrollView 与 ListView 共用问题的解决方案

26 ProgressBar 进度条

26.1 ProgressBar 类概述

progressBar 进度条是一个显示进度的控件，Android 提供了两大类进度条样式，长形进度条样式 progress-BarStyleHorizontal 和圆形进度条 progressBarStyleLarge.ProgressBar 类的层次关系如下：

```
java.lang.Object
  android.view.View
    android.widget.ProgressBar
```

26.2 ProgressBar 常用的方法

方法	功能描述	返回值
ProgressBar	3 个构造函数： ProgressBar（Context contex） ProgressBar（Context contex，AttributeSet attrs） ProgressBar（Context contex，AttributeSet attrs，intdefStyle）	null
onAttachedToWindow（）	视图附加到窗体时调用	void
onDraw（Canvas canvas）	绘制视图时，该方法被调用	void
onMeasure（int widthMeasureSpec，int heightMeasureSpec）	该方法被重写时，必须调用 setMeasedDimension（int，int）来存储已测量视图的高度和宽度，否则将通过 measure（int，int）抛出一个 IllegalStateException 异常	void
onDetachedFromWindow（）	从窗体分离事件的响应方法。当视图 Progress 从窗体上分离或移除时调用	void
addFocusables（ ArrayList<View>views,int direction）	继承自 android.view.View 的方法，用于为当前 ViewGroup 中的所有子视图添加焦点获取能力	void
addTouchables(ArrayList<View views>)	继承自 android.view.View 的方法，用于为子视图添加触摸能力	void
getBaseline	继承自 android.view.View 的方法。返回窗口空间的文本基准线到其顶边界的偏移量	int
dispatchDisplayHint(int hint)	继承自 android.view.View 的方法。分发视图是否显示的提示	void
dispatchDraw(Canvas canvas)	继承自 android.view.View 的方法。调用次方法来绘制子视图	void

26.3 ProgressBar 的使用

要实现一个简单的进度条的效果，我们可以首先在 XML 文件中声明一个长形进度条和一个圆形进度条：

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
  <ProgressBar
    android:id="@+id/ProgressBar01"
     style="?android:attr/progressBarStyleHorizontal" //设置样式为长形进度条
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:visibility="gone"  //设置当前不可见
  />
  <ProgressBar
          android:id="@+id/ProgressBar02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="?android:attr/progressBarStyleLarge" //设置样式为圆形进度条
        android:max="100"
        android:progress="50"
```

```
        android:secondaryProgress="70"
          android:visibility="gone"   //设置当前不可见
  />
  <Button
    android:id="@+id/Button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
     android:text="开始" />
```

然后在 Activity 中实现监听并通过线程来改变 ProgressBar 的值：

```
/设置窗口模式，，因为需要显示进度条在标题栏
        requestWindowFeature(Window.FEATURE_PROGRESS);
        setProgressBarVisibility(true);
        setContentView(R.layout.main);

        //取得ProgressBar
        m_ProgressBar = (ProgressBar) findViewById(R.id.ProgressBar01);
        m_ProgressBar2= (ProgressBar) findViewById(R.id.ProgressBar02);
        mButton01 = (Button)findViewById(R.id.Button01);

        m_ProgressBar.setIndeterminate(false);
        m_ProgressBar2.setIndeterminate(false);

        //当按钮按下时开始执行,
       mButton01.setOnClickListener(new Button.OnClickListener()
       {
          @Override
          public void onClick(View v)
          {
            // TODO Auto-generated method stub

           //设置ProgressBar为可见状态
           m_ProgressBar.setVisibility(View.VISIBLE);
           m_ProgressBar2.setVisibility(View.VISIBLE);
           //设置ProgressBar的最大值
           m_ProgressBar.setMax(100);
           //设置ProgressBar当前值
           m_ProgressBar.setProgress(0);
           m_ProgressBar2.setProgress(0);

           //通过线程来改变ProgressBar的值
                   new Thread(new Runnable() {
                         public void run()
                         {
                               for (int i = 0; i < 10; i++)
                               {
                                     try
                                     {
                                           intCounter = (i + 1) * 20;
```

```
                                Thread.sleep(1000);

                                if (i == 4)
                                {
                                    Message m = new Message();

                                    m.what = ProgressBarActivity.GUI_STOP_NOTIFIER;
                                    ProgressBarActivity.this.myMessageHandler.sendMessage(m);
                                    break;
                                }
                                else
                                {
                                    Message m = new Message();
                                    m.what = ProgressBarActivity.GUI_THREADING_NOTIFIER;
                                    ProgressBarActivity.this.myMessageHandler.sendMessage(m);
                                }
                            }
                            catch (Exception e)
                            {
                                e.printStackTrace();
                            }
                        }
                    }
                }).start();
            }
        });
    }

    Handler myMessageHandler = new Handler()
    {
        // @Override
        public void handleMessage(Message msg)
        {
            switch (msg.what)
            {
            //ProgressBar已经是对大值
            case ProgressBarActivity.GUI_STOP_NOTIFIER:
                m_ProgressBar.setVisibility(View.GONE);
                m_ProgressBar2.setVisibility(View.GONE);
                Thread.currentThread().interrupt();
                break;
            case ProgressBarActivity.GUI_THREADING_NOTIFIER:
                if (!Thread.currentThread().isInterrupted())
                {
                    // 改变ProgressBar的当前值
                    m_ProgressBar.setProgress(intCounter);
                    m_ProgressBar2.setProgress(intCounter);

                    // 设置标题栏中前景的一个进度条进度值
```

```
                    setProgress(intCounter*100);
                    // 设置标题栏中后面的一个进度条进度值
                    setSecondaryProgress(intCounter*100);//
                }
                break;
            }
            super.handleMessage(msg);
        }
    };
```

效果如下：

27 SeekBar 拖动条

27.1 SeekBar 类概述

SeekBar 是 ProgressBar 的扩展，在其基础上增加了一个可滑动的滑片(注：就是那个可拖动的图标)。用户可以触摸滑片并向左或向右拖动，再或者可以使用方向键都可以设置当前的进度等级。不建议把可以获取焦点的 widget 放在 SeekBar 的左边或右边。

SeekBar 可以附加一个 SeekBar.OnSeekBarChangeListener 以获得用户操作的通知。

SeekBar 类的层次关系如下：

public class SeekBar extends AbsSeekBar

```
java.lang.Object
    android.view.View
        android.widget.ProgressBar
            android.widget.AbsSeekBar
                android.widget.SeekBar
```

27.2 SeekBar 常用的方法

方法	功能描述	返回值
SeekBar	3 个构造函数： SeekBar（Context contex）	null

	SeekBar（Context contex，AttributeSet attrs） SeekBar（Context contex，AttributeSet attrs，intdefStyle）	
setOnSeekBarChangeListener（SeekBar.OnSeekBarChangeListener）	用于注册拖动条的监听器，这个监听器用于监听改变拖动条状态的事件	void
drawableStateChanged（）	继承自 ProgressBar 的方法。该方法在视图状态的变化影响到所显示的可绘制对象的状态时调用。	void
onAttacedToWindow（）	继承自 ProgressBar 的方法。该方法在视图添加到窗体时调用	
onDraw（Canvas canvas）	继承自 ProgressBar 的方法。该方法在绘制视图时调用	void
onMessure（int w，int h）	继承自 ProgressBar 的方法。该方法被重写时，必须调用 setMeasedDimension（int，int）来存储已测量视图的高度和宽度，否则将通过 measure（int，int）抛出一个 IllegalStateException 异常	void

27.3 SeekBar 属性

XML 属性

属性名称	描述
android:thumb	Seekbar 上绘制的 thumb（可拖动的那个图标）

27.4 SeekBar 的使用

在 xml 中声明一个 SeekBar，两个 TextView，一个用于显示当前 SeekBar 的值，一个用于显示调节的状态（正在调节或者以及停止调节）：

```
<SeekBar android:id="@+id/seek"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:max="100"
        android:progress="50"
        android:secondaryProgress="75" />

    <TextView android:id="@+id/progress"
         android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <TextView android:id="@+id/tracking"
         android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
```

然后在 Activity 中：

```
//取得SeekBar对象
        mSeekBar = (SeekBar) findViewById(R.id.seek);
        mSeekBar.setOnSeekBarChangeListener(this);
        mProgressText = (TextView) findViewById(R.id.progress);
        mTrackingText = (TextView) findViewById(R.id.tracking);
    }

    //在拖动中--即值在改变
```

```
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromTouch)
    {
        mProgressText.setText("当前值："+progress);
    }
    public void onStartTrackingTouch(SeekBar seekBar)
    {
        mTrackingText.setText("正在调节");
    }
    //停止拖动
    public void onStopTrackingTouch(SeekBar seekBar)
    {
        mTrackingText.setText("停止调节");
    }
```

效果如下：

27.5 补充资料

补充

文章链接

在 android 里做一个竖着的 seekbar
http://blog.csdn.net/saintswordsman/archive/2010/01/23/5248233.aspx
Android UI 设计 SeekBar 拖动条用法
http://www.pocketdigi.com/20100813/36.html

28 RatingBar 评分条

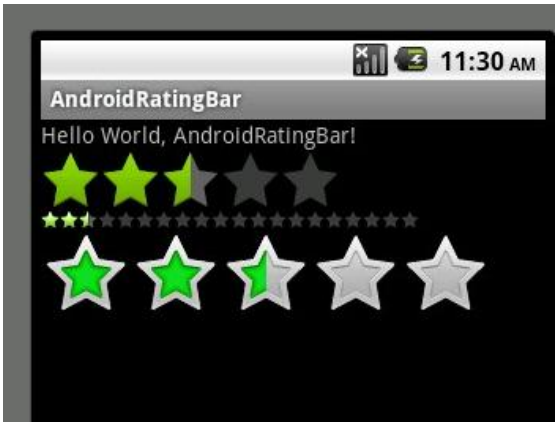
28.1 RatingBar 类概述

RatingBar 是基于 SeekBar 和 ProgressBar 的扩展，用星型来显示等级评定。使用 RatingBar 的默认大小时，用户可以触摸/拖动或使用键来设置评分，它有两种样式(小风格用 ratingBarStyleSmall，大风格用 ratingBarStyleIndicator)，其中大的只适合指示，不适合于用户交互。

当使用可以支持用户交互的 RatingBar 时，无论将控件(widgets)放在它的左边还是右边都是不合适的。

只有当布局的宽被设置为 wrap content 时，设置的星星数量（通过函数 setNumStars(int)或者在 XML 的布局文件中定义）将显示出来（如果设置为另一种布局宽的话，后果无法预知）。

次级进度一般不应该被修改，因为他仅仅是被当作星型部分内部的填充背景。



RatingBar 的层次关系如下所示：

public class RatingBar extends AbsSeekBar

java.lang.Object
    android.view.View
        android.widget.ProgressBar
            android.widget.AbsSeekBar
                android.widget.RatingBar

28.2 RatingBar 的属性

XML 属性

属性名称	描述
android:isIndicator	RatingBar 是否是一个指示器（用户无法进行更改）
android:numStars	显示的星型数量，必须是一个整形值，像“100”。
android:rating	默认的评分，必须是浮点类型，像“1.2”。
android:stepSize	评分的步长，必须是浮点类型，像“1.2”。

28.3 RatingBar 的常用方法

公共方法

（1） public int getNumStars ()

返回显示的星型数量

返回值： 显示的星型数量

(2)public RatingBar.OnRatingBarChangeListener getOnRatingBarChangeListener ()

返回值

监听器（可能为空）监听评分改变事件

(3)public float getRating ()

获取当前的评分（填充的星型的数量）

返回值

当前的评分

(4)public float getStepSize ()

获取评分条的步长

返回值

The step size.

步长

(5)public boolean isIndicator ()

返回值

判断当前的评分条是否仅仅是一个指示器（注：即能否被修改）

(6)public void setIsIndicator (boolean isIndicator)

设置当前的评分条是否仅仅是一个指示器（这样用户就不能进行修改操作了）

参数

isIndicator　　Bool 值，是否是一个指示器

(7)public synchronized void setMax (int max)

设置评分等级的范围，从 0 到 max

参数

max　　评分条最大范围。

(8)public void setNumStars (int numStars)

设置显示的星型的数量。为了能够正常显示它们，建议将当前 widget 的布局宽度设置为 wrap content

参数

numStars　　星型的数量

(9)public void setOnRatingBarChangeListener (RatingBar.OnRatingBarChangeListener listener)

设置当评分等级发生改变时回调的监听器

参数

listener　监听器

(10)public void setRating (float rating)

设置分数（星型的数量）

参数

rating　　设置的分数

(11)public void setStepSize (float stepSize)

设置当前评分条的步长（step size）

参数

stepSize 评分条的步进。例如：如果想要半个星星，它的值为 0.5。

受保护方法

(12)protected synchronized void onMeasure (int widthMeasureSpec, int heightMeasureSpec)

权衡 view 和 content 来决定它的宽度和高度的整齐。它被 measure(int, int) 调用 并且应该被子类所覆盖，以便提供准确高效的布局测量。

规定：当覆盖这个方法的时候，你必须调用 setMeasuredDimension(int, int)以便存储精确的视图的宽和高。如果不这样做的话将触发 IllegalStateException 异常，被函数 measure(int, int)抛出。调用父类 onMeasure(int, int)是合理的。

尺寸的基本类的实现默认是背景大小，除非通过 MeasureSpec 允许大的尺寸。子类应该覆盖 onMeasure(int, int) 以便提供更好的布局大小。

如果这个方法被覆盖，子类应该负责确保标准的宽和高至少是视图的最小宽度和高度的值（分别为 getSuggestedMinimumHeight() 和 getSuggestedMinimumWidth()两方法）。

参数

widthMeasureSpec　受主窗口支配的水平空间要求。这个需求通过 View.MeasureSpec.进行编码。

heightMeasureSpec　受主窗口支配的垂直空间要求。这个需求通过 View.MeasureSpec.进行编码。

28.4 RatingBar 的使用

我们在 xml 中声明三种 样式的 RatingBar：

```
<RatingBar android:layout_width="wrap_content"

        android:layout_height="wrap_content" style="?android:attr/ratingBarStyleIndicator"

        android:id="@+id/ratingbar_Indicator" />

    <RatingBar android:layout_width="wrap_content"

        android:layout_height="wrap_content" style="?android:attr/ratingBarStyleSmall"
```

```
        android:id="@+id/ratingbar_Small" android:numStars="20" />

    <RatingBar android:layout_width="wrap_content"

        android:layout_height="wrap_content" style="?android:attr/ratingBarStyle"

        android:id="@+id/ratingbar_default" />
```

在 Activity 中声明并显示：

```
final RatingBar ratingBar_Small = (RatingBar)findViewById(R.id.ratingbar_Small);

        final RatingBar ratingBar_Indicator = (RatingBar)findViewById(R.id.ratingbar_Indicator);

        final RatingBar ratingBar_default = (RatingBar)findViewById(R.id.ratingbar_default);



        ratingBar_default.setOnRatingBarChangeListener(new RatingBar.OnRatingBarChangeListener(){



    public void onRatingChanged(RatingBar ratingBar, float rating,

      boolean fromUser) {

     ratingBar_Small.setRating(rating);

     ratingBar_Indicator.setRating(rating);

     Toast.makeText(RatingBarActivity.this, "rating:"+String.valueOf(rating),

       Toast.LENGTH_LONG).show();

    }});
```

28.5 补充资料

补充

文章链接

Android 控件之 RatingBar 评分条
Android 更换 RatingBar 图片
[Android 学习指南]RatingBar 评分条

29 ProgressDialog 对话框中的进度条

29.1 ProgressDialog 类概述

前面，我们学习了对话框和进度条，现在将进度条加入到对话框，使得对话框更加的完善。我们知道进度条有长形进度条和圆形进度条，所以也就有对话框中的长形进度条和对话框中的圆形进度条。

29.2 ProgressDialog 中常用的方法

setProgressStyle ： 设置进度条风格，风格为圆形，旋转的或者风格为长形的；
setTitle ： 设置 ProgressDialog 的标题
setMessage ： 设置 ProgressDialog 的提示信息
setIcon ：设置 ProgressDialog 的标题图标
setIndeterminate ：设置 ProgressDialog 的进度条是否不明确
setCancelable ：设置 ProgressDialog 是否可以按回退键取消
setButton ：设置 ProgressDialog 的一个 Button（需要监听 Button 事件）
show ：显示 ProgressDialog

29.3 ProgressDialog 的使用

要实现 29.1 节中图形中所示的效果，我们首先在 xml 中声明两个 Button：

```
<Button
    android:id="@+id/yuan_button01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="圆形进度条" />
```

```
<Button
  android:id="@+id/chang_button02"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
   android:text="长形进度条" />
```

在 Activity 中得到 Button 对象，并分别实现监听，在监听里面生成我们所需要的带进度条的对话框，对于长形进度条我们需要用线程来实时更新进度条的值。具体代码如下：

```
/得到按钮对象
        mButton01 = (Button)findViewById(R.id.yuan_button01);
        mButton02 = (Button)findViewById(R.id.chang_button02);

        //设置mButton01的事件监听
        mButton01.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v)
            {
                // TODO Auto-generated method stub

                //创建ProgressDialog对象
                m_pDialog = new ProgressDialog(ProgressDialogActivity.this);

                // 设置进度条风格，风格为圆形，旋转的
                m_pDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);

                // 设置ProgressDialog 标题
                m_pDialog.setTitle("提示");

                // 设置ProgressDialog 提示信息
                m_pDialog.setMessage("这是一个圆形进度条对话框");

                // 设置ProgressDialog 标题图标
                m_pDialog.setIcon(R.drawable.imag1);

                // 设置ProgressDialog 的进度条是否不明确
                m_pDialog.setIndeterminate(false);

                // 设置ProgressDialog 是否可以按退回按键取消
                m_pDialog.setCancelable(true);

                // 设置ProgressDialog 的一个Button
                m_pDialog.setButton("确定", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int i)
                    {
                        //点击“确定按钮”取消对话框
                        dialog.cancel();
                    }
                });
```

```
                // 让ProgressDialog显示
                m_pDialog.show();
            }
        });

    //设置mButton02的事件监听
        mButton02.setOnClickListener(new Button.OnClickListener() {
            @Override
            public void onClick(View v)
            {
                // TODO Auto-generated method stub

                m_count = 0;

                // 创建ProgressDialog对象
                m_pDialog = new ProgressDialog(ProgressDialogActivity.this);

                // 设置进度条风格，风格为长形
                m_pDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);

                // 设置ProgressDialog 标题
                m_pDialog.setTitle("提示");

                // 设置ProgressDialog 提示信息
                m_pDialog.setMessage("这是一个长形对话框进度条");

                // 设置ProgressDialog 标题图标
                m_pDialog.setIcon(R.drawable.imag2);

                // 设置ProgressDialog 进度条进度
                m_pDialog.setProgress(100);

                // 设置ProgressDialog 的进度条是否不明确
                m_pDialog.setIndeterminate(false);

                // 设置ProgressDialog 是否可以按退回按键取消
                m_pDialog.setCancelable(true);

                // 让ProgressDialog显示
                m_pDialog.show();

                new Thread()
                {
                    public void run()
                    {
                        try
                        {
                            while (m_count <= 100)
                            {
```

```
                    // 由线程来控制进度。
                    m_pDialog.setProgress(m_count++);
                    Thread.sleep(100);
                }
                m_pDialog.cancel();
            }
            catch (InterruptedException e)
            {
                m_pDialog.cancel();
            }
        }
    }.start();

    }
});
```

30 Notification、NotificationManager 状态栏提示

30.1 概述

当有未接电话或者短信的时候，在 Android 手机的顶部状态栏会出现一个小图标，提示用户有没有处理的快讯，这时用触笔按住状态栏往下拖动时就可以展开并查看这些快讯，Android 提供了 NotificationManager 来管理这些状态栏信息，提供了 Notification 来处理这些快讯信息。

30.2 Notification 属性

Notification 常用的属性：

audioStreamType ：当声音响起时，所用的音频流的类型

contentIntent ：当通知条目被点击时，就执行被设置的 Intent

contentView ：当通知被显示在状态条上的时候，同时这个被设置的视图被显示

default 指定哪个值是要被设置为默认的

deleteIntent ：当用户点击“Clear All Notifications”按钮区域删除所有通知的时候，这个被设置的 Intent 被执行

icon 状态条所用的图片

iconLevel ：假如状态条的图片有几个级别，就设置这里

ledARGB ： LED 灯的颜色

ledOffMS LED ： 关闭时的闪光时间，以毫秒计算
ledOnMS LED ：开始时的闪光时间，以毫秒计算

number ：这个通知代表事件的号码
sound ：通知的声音
tickerText ：通知被显示在状态栏时，所显示的信息
vibrate ：振动模式
when ：通知的时间戳

Notification 常用的字段：

DEFAULT_ALL 使用所有默认值，比如声音，震动，闪屏等等

DEFAULT_LIGHTS 使用默认闪光提示

DEFAULT_SOUNDS 使用默认提示声音

DEFAULT_VIBRATE 使用默认手机震动

【说明】：加入手机震动，一定要在 manifest.xml 中加入权限：

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

以上的效果常量可以叠加,即通过

mNotifaction.defaults =DEFAULT_SOUND | DEFAULT_VIBRATE ;

或 mNotifaction.defaults |=DEFAULT_SOUND (最好在真机上测试，震动效果模拟器上没有)

//设置 flag 位

FLAG_AUTO_CANCEL 该通知能被状态栏的清除按钮给清除掉

FLAG_NO_CLEAR 该通知能被状态栏的清除按钮给清除掉

FLAG_ONGOING_EVENT 通知放置在正在运行

FLAG_INSISTENT 是否一直进行，比如音乐一直播放，知道用户响应

30.3 Notification 的方法

Notification.build 构造 Notification 方法介绍：

void setLatestEventInfo(Context context , CharSequencecontentTitle,CharSequence contentText,PendingIntent contentIntent)

功能： 显示在拉伸状态栏中的 Notification 属性，点击后将发送 PendingIntent 对象

参数： context 上下文环境

contentTitle 状态栏中的大标题

contentText 状态栏中的小标题

contentIntent 点击后将发送 PendingIntent 对象

说明：要是在 Notification 中加入图标，在状态栏和状态条中显示图标一定要用这个方法，否则报错！

NotificationManager 类的常用方法：

过获取系统服务来获取该对象：

NotificationManager mNotificationManager = (NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE) ;

public void cancelAll() 移除所有通知（只是针对当前 Context 下的 Notification）

public void cancel(int id) 移除标记为 id 的通知（只是针对当前 Context 下的所有 Notification）

public void notify(String tag ,int id, Notification notification) 将通知加入状态栏，标签为 tag，标记为 id

public void notify(int id, Notification notification) 将通知加入状态栏,，标记为 id

30.4 Notification 的使用

首先我们定义了 2 个 XMl 文件，一个 xml 中声明了一个 TextView 和一个 Button。一个 xml 中我们只有一个 TextView.，这个 TextView 是用来查看快讯的。

main.xml:

```
<TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:paddingBottom="4dip"
        />
<Button
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button01"
        >
        <requestFocus/>
    </Button>
```

main2.xml:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="谢谢使用！"
    />
```

在 Activity NotificantionActivity01 里面：

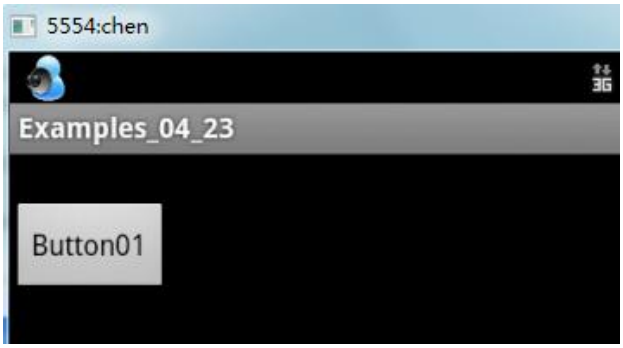
```
Button                  m_Button1;

    //声明通知（消息）管理器
    NotificationManager m_NotificationManager;
    Intent              m_Intent;
    PendingIntent       m_PendingIntent;
    //声明Notification对象
    Notification        m_Notification;


    /** Called when the activity is first created. */
    @Override
```

```
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    //初始化NotificationManager对象
    m_NotificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

    //获取4个按钮对象
    m_Button1 = (Button) findViewById(R.id.Button01);


    //点击通知时转移内容
    m_Intent = new Intent(NotificantionActivity01.this, NotificantionActivity02.class);
    //主要是设置点击通知时显示内容的类
    m_PendingIntent = PendingIntent.getActivity(NotificantionActivity01.this, 0, m_Intent, 0);
    //构造Notification对象
    m_Notification = new Notification();

    m_Button1.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v)
        {
            //设置通知在状态栏显示的图标
            m_Notification.icon = R.drawable.img1;
            //当我们点击通知时显示的内容
            m_Notification.tickerText = "Button1通知内容.....";
            //通知时发出默认的声音
            m_Notification.defaults = Notification.DEFAULT_SOUND;
            //设置通知显示的参数
            m_Notification.setLatestEventInfo(NotificantionActivity01.this, "Button1", "Button1通知", m_PendingIntent);
            //可以理解为执行这个通知
            m_NotificationManager.notify(0, m_Notification);
        }
    });
```

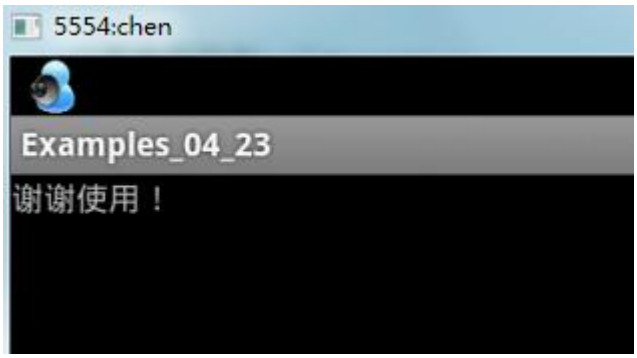
然后在 NotificantionActivity02 里：

```
//这里直接限制一个TextView
        setContentView(R.layout.main2);
```

显示的效果为：



5554:chen
February 8, 2012
3:00
Android
Clear
Notifications
Button1
Button1通知
2:46 AM


5554:chen
Examples_04_23
谢谢使用！