# UI组件属性

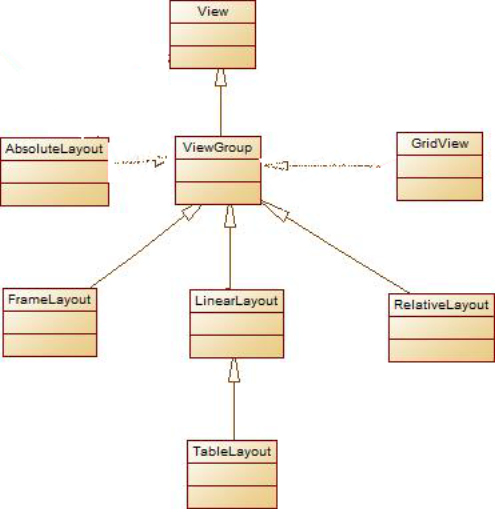
**属性**

|  |  |
| --- | --- |
| android:id=*"@+id/mobile"* | 定义ID |
| android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"* | 填充宽度和高度, 或*50dip 或 dp* |
| android:layout\_toRightOf=*"@id/button"* | 在哪个(id)的右边 |
| android:layout\_alignTop=*"@id/button"* | 和哪个(id)顶部对齐 |
| android:gravity=*"center"* | 居中对齐 |
| android:text=*"@string/button"* | 显示文本 |
| android:textSize=*"30sp"* | 字体大小 |
| android:minLines=*"4"* | 最少多少行 |
| android:numeric=*"integer"* | 数据类型: 整数 |
|  |  |

**各种UI控件**

|  |
| --- |
| **文本标签(TextView)**  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"@string/path"*  /> |
| **编辑框(EditText)**  <EditText  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:id=*"@+id/path"*  /> |
| **多项选择(CheckBox)** |
| **单项选择(RadioGroup)** |
| **下拉列表(Spinner)** |
| **自动完成文本(AutoCompleteTextView)** |
| **日期选择器(DatePicker)** |
| **时间选择器(TimePicker)** |
| **滚动视图(ScrollView)** |
| **进度条(ProgressBar)**  <ProgressBar  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"20dip"*  style=*"?android:attr/progressBarStyleHorizontal"*  android:text=*"@string/button"*  android:id=*"@+id/downloadbar"*  /> |
| **拖动条(SeekBar)** |
| **评分组件(RatingBar)** |
| **图片视图(ImageView)** |
| **图片按钮(ImageButton)** |
| **切换图片(ImageSwitcher&Gallery)** |
| **网格视图(GridView)** |
| **标签(Tab)** |
| <!-- 绘制画面的, 比如用于视频播放, 开发游戏 -->  <SurfaceView  android:id=*"@+id/surfaceView"*  android:layout\_width=*"320px"*  android:layout\_height=*"240px"* /> |

# 布局管理器



## AbsoluteLayout 绝对布局

|  |
| --- |
| <AbsoluteLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"vertical"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  >  <!-- 定义一个文本框，使用绝对定位 -->  <TextView  android:layout\_x=*"20dip"*  android:layout\_y=*"20dip"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"用户名："*  />  <!-- 定义一个文本编辑框，使用绝对定位 -->  <EditText  android:layout\_x=*"80dip"*  android:layout\_y=*"15dip"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:width=*"200px"*  />  <!-- 定义一个文本框，使用绝对定位 -->  <TextView  android:layout\_x=*"20dip"*  android:layout\_y=*"80dip"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"密 码："*  />  <!-- 定义一个文本编辑框，使用绝对定位 -->  <EditText  android:layout\_x=*"80dip"*  android:layout\_y=*"75dip"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:width=*"200px"*  android:password=*"true"*  />  <!-- 定义一个按钮，使用绝对定位 -->  <Button  android:layout\_x=*"130dip"*  android:layout\_y=*"135dip"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"登 录"*  />  </AbsoluteLayout>  未命名:Users:jinwanlin:Desktop:Screenshot_2015-05-05-17-26-17.png |

## FrameLayout 框架布局(层叠布局)

|  |
| --- |
| 先定义的View位于底层, 后定义的View位于上层 |
| <?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"*  >  <!-- 依次定义6个TextView,先定义的TextView位于底层后定义的TextView位于上层 -->  <TextView android:id=*"@+id/view01"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_gravity=*"center"*  android:width=*"320px"*  android:height=*"320px"*  android:background=*"#f00"*  />  <TextView android:id=*"@+id/view02"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_gravity=*"center"*  android:width=*"280px"*  android:height=*"280px"*  android:background=*"#0f0"*  />  <TextView android:id=*"@+id/view03"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_gravity=*"center"*  android:width=*"240px"*  android:height=*"240px"*  android:background=*"#00f"*  />  <TextView android:id=*"@+id/view04"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_gravity=*"center"*  android:width=*"200px"*  android:height=*"200px"*  android:background=*"#ff0"*  />  <TextView android:id=*"@+id/view05"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_gravity=*"center"*  android:width=*"160px"*  android:height=*"160px"*  android:background=*"#f0f"*  />  <TextView android:id=*"@+id/view06"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_gravity=*"center"*  android:width=*"120px"*  android:height=*"120px"*  android:background=*"#0ff"*  />  </FrameLayout>  未命名:Users:jinwanlin:Desktop:Screenshot_2015-05-05-17-28-09.png |

## GridLayout 九宫格

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"* ?>  <GridLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:rowCount=*"6"*  android:columnCount=*"4"*  android:id=*"@+id/root"*  >  <!-- 定义一个横跨4列的文本框，  并设置该文本框的前景色、背景色等属性 -->  <TextView  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:layout\_columnSpan=*"4"*  android:textSize=*"50sp"*  android:layout\_marginLeft=*"4px"*  android:layout\_marginRight=*"4px"*  android:padding=*"5px"*  android:layout\_gravity=*"right"*  android:background=*"#eee"*  android:textColor=*"#000"*  android:text=*"0"*/>  <!-- 定义一个横跨4列的按钮 -->  <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:layout\_columnSpan=*"4"*  android:text=*"清除"*/>  </GridLayout> |
| @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);  gridLayout = (GridLayout) findViewById(R.id.***root***);  **for** (**int** i = 0; i < chars.length; i++) {  Button bn = **new** Button(**this**);  bn.setText(chars[i]);  bn.setTextSize(40); // 设置该按钮的字体大小    GridLayout.Spec rowSpec = GridLayout.*spec*(i / 4 + 2); // 指定该组件所在的行  GridLayout.Spec columnSpec = GridLayout.*spec*(i % 4); // 指定该组件所在列  GridLayout.LayoutParams params = **new** GridLayout.LayoutParams(rowSpec, columnSpec);  params.setGravity(Gravity.***FILL***); // 指定该组件占满父容器  gridLayout.addView(bn, params);  }  }  未命名:Users:jinwanlin:Pictures:Screenshot_2015-05-05-18-02-01.png |

## LinearLayout线性布局(水平/垂直)

(Android:orientation="horizontal"水平布局, Android:orientation="vertical"垂直布局)

|  |
| --- |
| **LinearLayout线性布局(水平horizontal/垂直vertical)**  <LinearLayout  android:orientation=*"horizontal"*  android:layout\_height=*"wrap\_content"*  android:layout\_width=*"fill\_parent"*  android:padding=*"10dip"*>  <ImageButton  android:id=*"@+id/play"*  android:layout\_height=*"wrap\_content"*  android:layout\_width=*"wrap\_content"*  android:src=*"@drawable/play"* />  <ImageButton  android:id=*"@+id/pause"*  android:layout\_height=*"wrap\_content"*  android:layout\_width=*"wrap\_content"*  android:src=*"@drawable/pause"* />  <ImageButton  android:id=*"@+id/reset"*  android:layout\_height=*"wrap\_content"*  android:layout\_width=*"wrap\_content"*  android:src=*"@drawable/reset"* />  <ImageButton  android:id=*"@+id/stop"*  android:layout\_height=*"wrap\_content"*  android:layout\_width=*"wrap\_content"*  android:src=*"@drawable/stop"* />  </LinearLayout> |

## RelativeLayout相对布局

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"vertical"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  >  <!-- 定义该组件位于父容器中间 -->  <TextView  android:id=*"@+id/view01"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:background=*"@drawable/leaf"*  android:layout\_centerInParent=*"true"*  />  <!-- 定义该组件位于view01组件的上方 -->  <TextView  android:id=*"@+id/view02"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:background=*"@drawable/leaf"*  android:layout\_above=*"@id/view01"*  android:layout\_alignLeft=*"@id/view01"*  />  <!-- 定义该组件位于view01组件的下方 -->  <TextView  android:id=*"@+id/view03"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:background=*"@drawable/leaf"*  android:layout\_below=*"@id/view01"*  android:layout\_alignLeft=*"@id/view01"*  />  <!-- 定义该组件位于view01组件的左边 -->  <TextView  android:id=*"@+id/view04"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:background=*"@drawable/leaf"*  android:layout\_toLeftOf=*"@id/view01"*  android:layout\_alignTop=*"@id/view01"*  />  <!-- 定义该组件位于view01组件的右边 -->  <TextView  android:id=*"@+id/view05"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:background=*"@drawable/leaf"*  android:layout\_toRightOf=*"@id/view01"*  android:layout\_alignTop=*"@id/view01"*  />  </RelativeLayout>  **未命名:Users:jinwanlin:Desktop:Screenshot_2015-05-05-17-57-42.png** |

## TableLayout表格布局

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"vertical"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  >  <!-- 定义第一个表格布局，指定第2列允许收缩，第3列允许拉伸 -->  <TableLayout android:id=*"@+id/TableLayout01"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:shrinkColumns=*"1"*  android:stretchColumns=*"2"*  >  <!-- 直接添加按钮，它自己会占一行 -->  <Button android:id=*"@+id/ok1"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"独自一行的按钮"*  />  <!-- 添加一个表格行 -->  <TableRow>  <!-- 为该表格行添加3个按钮 -->  <Button android:id=*"@+id/ok2"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"普通按钮"*  />  <Button android:id=*"@+id/ok3"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"收缩的按钮"*  />  <Button android:id=*"@+id/ok4"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"拉伸的按钮"*  />  </TableRow>  </TableLayout>  <!-- 定义第二个表格布局 ，指定第二列隐藏-->  <TableLayout android:id=*"@+id/TableLayout01"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:collapseColumns=*"1"*  >  <!-- 直接添加按钮，它自己会占一行 -->  <Button android:id=*"@+id/ok5"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*" 独自一行的按钮 "*  />  <!--定义一个表格行-->  <TableRow>  <!-- 为该表格行添加3个按钮 -->  <Button android:id=*"@+id/ok6"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"普通按钮1"*  />  <Button android:id=*"@+id/ok7"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"被隐藏的按钮"*  />  <Button android:id=*"@+id/ok8"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"普通按钮 3"*  />  </TableRow>  </TableLayout>  <!-- 定义第三个表格布局 ，指定第2、3两列可以被拉伸-->  <TableLayout android:id=*"@+id/TableLayout01"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:stretchColumns=*"1,2"*  >  <!-- 直接添加按钮，它自己会占一行 -->  <Button android:id=*"@+id/ok9"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"独自一行的按钮"*  />  <!--定义一个表格行-->  <TableRow>  <!-- 为该表格行添加3个按钮 -->  <Button android:id=*"@+id/ok10"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"普通按钮"*  />  <Button android:id=*"@+id/ok11"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"拉伸的按钮"*  />  <Button android:id=*"@+id/ok12"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"拉伸的按钮"*  />  </TableRow>  <!--定义一个表格行-->  <TableRow>  <!-- 为该表格行添加2个按钮 -->  <Button android:id=*"@+id/ok13"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"普通按钮"*  />  <Button android:id=*"@+id/ok14"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"拉伸的按钮"*  />  </TableRow>  </TableLayout>  </LinearLayout>  **未命名:Users:jinwanlin:Pictures:Screenshot_2015-05-05-18-00-08.png** |

# 其它UI组件

## EditText

|  |
| --- |
| <!-- android:inputType="numberPassword"表明只能接收数字密码 -->  <EditText  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:inputType=*"numberPassword"* /> |
| <!-- inputType="number"表明是数值输入框 -->  <EditText  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:inputType=*"number"* /> |
| <!-- inputType="date"表明是日期输入框 -->  <EditText  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:inputType=*"date"* /> |
| <!-- inputType="phone"表明是输入电话号码的输入框 -->  <EditText  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:hint=*"请填写您的电话号码"*  android:inputType=*"phone"*  android:selectAllOnFocus=*"true"* /> |
|  |

## TextView

|  |
| --- |
| <!-- 设置字体为20pt，文本框结尾处绘制图片 -->  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:drawableEnd=*"@drawable/ic\_launcher"*  android:text=*"我爱Java"*  android:textSize=*"20pt"* /> |
| <!-- 设置中间省略， 所有字母大写 -->  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:ellipsize=*"middle"*  android:singleLine=*"true"*  android:text=*"我爱Java我爱Java我爱Java我爱Java我爱Java我aaaJava"*  android:textAllCaps=*"true"* /> |
| <!-- 对邮件、电话增加链接 -->  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:autoLink=*"email|phone"*  android:singleLine=*"true"*  android:text=*"邮件是kongyeeku@163.com，电话是02088888888"* /> |
| <!-- 设置文字颜色 、大小，并使用阴影 -->  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:shadowColor=*"#0000ff"*  android:shadowDx=*"10.0"*  android:shadowDy=*"8.0"*  android:shadowRadius=*"3.0"*  android:text=*"测试文字"*  android:textColor=*"#f00"*  android:textSize=*"18pt"* /> |
| <!-- 测试密码框 -->  <TextView  android:id=*"@+id/passwd"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:password=*"true"*  android:text=*"@string/hello"* /> |
| <!-- 测试CheckedTextView 通过checkMark设置该文本框的勾选图标 -->  <CheckedTextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:checkMark=*"@drawable/ok" //这里是图片*  android:text=*"可勾选的文本"* /> |
| <!-- 通过android:background指定背景 -->  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"带边框的文本"*  android:textSize=*"24pt"*  android:background=*"@drawable/bg\_border"*  />  bg\_border.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <shape xmlns:android=*"http://schemas.android.com/apk/res/android"*>  <!-- 设置背景色为透明色 -->  <solid android:color=*"#0000"*/>  <!-- 指定边框线条的宽度和颜色 -->  <stroke android:width=*"4px"* android:color=*"#f00"* />  </shape>  未命名:Users:jinwanlin:Desktop:Screen Shot 2015-05-05 at 6.07.01 PM.png |
| <!-- 通过android:drawableLeft绘制一张图片 -->  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"圆角边框、渐变背景的文本"*  android:textSize=*"24pt"*  android:background=*"@drawable/bg\_border2"*  />  bg\_border2.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <shape xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:shape=*"rectangle"*>  <!-- 指定圆角矩形的4个圆角的半径 -->  <corners android:topLeftRadius=*"20px"*  android:topRightRadius=*"5px"*  android:bottomRightRadius=*"20px"*  android:bottomLeftRadius=*"5px"*/>  <!-- 指定边框线条的宽度和颜色 -->  <stroke android:width=*"4px"* android:color=*"#f0f"* />  <!-- 指定使用渐变背景色，使用sweep类型的渐变颜色从红色→绿色→蓝色 -->  <gradient android:startColor=*"#f00"*  android:centerColor=*"#0f0"*  android:endColor=*"#00f"*  android:type=*"sweep"*/>  </shape>  未命名:Users:jinwanlin:Desktop:Screen Shot 2015-05-05 at 6.07.44 PM.png |

## ImageView图片

|  |
| --- |
| <!-- 定义显示图片整体的ImageView -->  <ImageView android:id=*"@+id/image1"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"240px"*  android:src=*"@drawable/shuangta"*  android:scaleType=*"fitCenter"*/>  <!-- 定义显示图片局部细节的ImageView -->  <ImageView android:id=*"@+id/image2"*  android:layout\_width=*"120dp"*  android:layout\_height=*"120dp"*  android:background=*"#00f"*  android:layout\_marginTop=*"10dp"*/> |
| // 控制ImageView显示下一张图片  image1.setImageResource(images[++currentImg % images.length]); |
| // 改变图片的透明度  image1.~~setAlpha~~(alpha); |
| 显示图片的指定区域  image1.setOnTouchListener(**new** OnTouchListener() {  @Override  **public** **boolean** onTouch(View view, MotionEvent event) {  BitmapDrawable bitmapDrawable = (BitmapDrawable) image1.getDrawable();  // 获取第一个图片显示框中的位图  Bitmap bitmap = bitmapDrawable.getBitmap();  // bitmap图片实际大小与第一个ImageView的缩放比例  **double** scale = bitmap.getWidth() / 320.0;    // 获取需要显示的图片的开始点  **int** x = (**int**) (event.getX() \* scale);  **int** y = (**int**) (event.getY() \* scale);  **if** (x + 120 > bitmap.getWidth()) {  x = bitmap.getWidth() - 120;  }  **if** (y + 120 > bitmap.getHeight()) {  y = bitmap.getHeight() - 120;  }    // 显示图片的指定区域  image2.setImageBitmap(Bitmap.*createBitmap*(bitmap, x, y, 120, 120));  image2.~~setAlpha~~(alpha);  **return** **false**;  }  });  } |

## Button 按钮

|  |
| --- |
| <!-- 文字带阴影的按钮 -->  <Button  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"文字带阴影的按钮"*  android:textSize=*"12pt"*  android:shadowColor=*"#aa5"*  android:shadowRadius=*"1"*  android:shadowDx=*"5"*  android:shadowDy=*"5"*  /> |
| <!— 带背景图片的普通文字按钮 -->  <Button  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:background=*"@drawable/red"*  android:text=*"普通按钮"*  android:textSize=*"10pt"*  /> |
| <!--带背景图片的带点击效果的文字按钮-->  <Button  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:background=*"@drawable/button\_selector"*  android:textSize=*"11px"*  android:text=*"带文字的图片按钮"*  />  button\_selector.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <selector xmlns:android=*"http://schemas.android.com/apk/res/android"*>  <!-- 指定按下按钮时的图片 -->  <item android:state\_pressed=*"true"*  android:drawable=*"@drawable/red"*  />  <!-- 指定松开按钮时的图片 -->  <item android:state\_pressed=*"false"*  android:drawable=*"@drawable/purple"*  />  </selector> |
| Button start = (Button) findViewById(R.id.***start***);  start.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View source) {  start.setEnabled(**false**);  //start.setEnabled(**true**);  }  }); |

## ToggleButton Switch 开关切换按钮

|  |
| --- |
| <ToggleButton  android:id=*"@+id/toggle"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:checked=*"true"*  android:textOff=*"横向排列"*  android:textOn=*"纵向排列"* />    <Switch  android:id=*"@+id/switcher"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:checked=*"true"*  android:textOff=*"横向排列"*  android:textOn=*"纵向排列"*  android:thumb=*"@drawable/check"* /> |
| **final** LinearLayout test = (LinearLayout) findViewById(R.id.***test***);    OnCheckedChangeListener listener = **new** OnCheckedChangeListener() {  @Override  **public** **void** onCheckedChanged(CompoundButton button, **boolean** isChecked) {  **if** (isChecked) {  // 设置LinearLayout垂直布局  test.setOrientation(1);  } **else** {  // 设置LinearLayout水平布局  test.setOrientation(0);  }  }  };    toggle = (ToggleButton) findViewById(R.id.***toggle***);  switcher = (Switch) findViewById(R.id.***switcher***);  toggle.setOnCheckedChangeListener(listener);  switcher.setOnCheckedChangeListener(listener); |

## RadioGroup 单选框组

|  |
| --- |
| <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"性别:"*  android:textSize=*"16px"* />  <!-- 定义一组单选框 -->  <RadioGroup  android:id=*"@+id/rg"*  android:layout\_gravity=*"center\_horizontal"*  android:orientation=*"horizontal"* >  <!-- 定义两个单选框 -->  <RadioButton  android:id=*"@+id/male"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:checked=*"true"*  android:text=*"男"* />  <RadioButton  android:id=*"@+id/female"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"女"* />  </RadioGroup> |
| RadioGroup rg = (RadioGroup) findViewById(R.id.***rg***);  // 为RadioGroup组件的OnCheck事件绑定事件监听器  rg.setOnCheckedChangeListener(**new** OnCheckedChangeListener() {  @Override  **public** **void** onCheckedChanged(RadioGroup group, **int** checkedId) {  // 根据用户勾选的单选按钮来动态改变tip字符串的值  String tip = checkedId == R.id.***male*** ? "您的性别是男人" : "您的性别是女人";  // 修改show组件中的文本。  show.setText(tip);  }  }); |

## CheckButton 复选框

|  |
| --- |
| <CheckBox  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:checked=*"true"*  android:text=*"红色"* />  <CheckBox  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"蓝色"* />  <CheckBox  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"绿色"* /> |

## 9Patch 图片作为按钮背景

|  |  |
| --- | --- |
| NinePatch图片以\*.9.png结尾，和普通图片的区别是四周多了一个边框  Android  SDK/tools目录下提供了编辑器draw9patch.bat | |
| <Button  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:background=*"@drawable/bg\_new"*  android:text=*"测试按钮"*  bg\_new.9.png | |
| 左边的黑色线 | 垂直拉伸的区域 |
| 上边的条黑线 | 水平拉伸区域 |
| 右边的黑色线 | 内容绘制的垂直区域 |
| 下边的黑色线 | 内容绘制的水平区域 |
| 右边和下边的线是可选的，左边和上边的线不能省略。 | |

## Chronometer 秒表计时器

|  |
| --- |
| <Chronometer  android:id=*"@+id/test"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:textSize=*"12pt"*  android:textColor=*"#ffff0000"*  /> |
| // 获取计时器组件  ch = (Chronometer) findViewById(R.id.***test***);  // 获取“开始”按钮  start = (Button) findViewById(R.id.***start***);  start.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View source) {  // 设置开始计时时间  ch.setBase(SystemClock.*elapsedRealtime*());  // 启动计时器  ch.start();  start.setEnabled(**false**);  }  });    // 为Chronometer绑定事件监听器。  ch.setOnChronometerTickListener(**new** OnChronometerTickListener() {  @Override  **public** **void** onChronometerTick(Chronometer ch) {  // 如果从开始计时到现在超过了20s。  **if** (SystemClock.*elapsedRealtime*() - ch.getBase() > 20 \* 1000) {  ch.stop();  start.setEnabled(**true**);  }  }  }); |

## Clock 时钟(数字时钟&模拟时钟)

|  |
| --- |
| <!-- 定义数字时钟, 显示秒 -->  <DigitalClock  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:textSize=*"14pt"*  android:textColor=*"#f0f"*  android:drawableRight=*"@drawable/ic\_launcher" //时间右边的图片*  /> |
| <!-- 定义模拟时钟, 不显示秒 -->  <AnalogClock  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  /> |
| <!-- 定义模拟时钟,并使用自定义表盘、时针图片 -->  <AnalogClock  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:dial=*"@drawable/watch"*  android:hand\_minute=*"@drawable/hand"*  />    *watch.png*  *hand.png* |

## QuickContactBadge(绑定联系人)

|  |
| --- |
| <QuickContactBadge  android:id=*"@+id/badge"*  android:layout\_height=*"wrap\_content"*  android:layout\_width=*"wrap\_content"*  android:src=*"@drawable/ic\_launcher"*/> |
| // 获取QuickContactBadge组件  badge = (QuickContactBadge) findViewById(R.id.***badge***);  // 将QuickContactBadge组件与特定电话号码对应的联系人建立关联  badge.assignContactFromPhone("020-88888888", **false**); |

# ProgressBar进度条

## ProgressBar进度条(环形或水平)

|  |
| --- |
| <!-- 定义一个大环形进度条 -->  <ProgressBar  style=*"@android:style/Widget.ProgressBar.Large"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"* />  <!-- 定义一个中等大小的环形进度条 -->  <ProgressBar  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"* />    <!-- 定义一个小环形进度条 -->  <ProgressBar  style=*"@android:style/Widget.ProgressBar.Small"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"* /> |
| <!-- 定义一个水平进度条 -->  <ProgressBar  android:id=*"@+id/bar"*  style=*"@android:style/Widget.ProgressBar.Horizontal"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:max=*"100"* /> |
| <!-- 定义一个水平进度条，并改变轨道外观 -->  <ProgressBar  android:id=*"@+id/bar2"*  style=*"@android:style/Widget.ProgressBar.Horizontal"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:max=*"100"*  android:progressDrawable=*"@drawable/my\_bar"* />  my\_bar.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <layer-list xmlns:android=*"http://schemas.android.com/apk/res/android"*>  <!-- 定义轨道的背景 -->  <item android:id=*"@android:id/background"*  android:drawable=*"@drawable/no"* />  <!-- 定义轨道上已完成部分的样式 -->  <item android:id=*"@android:id/progress"*  android:drawable=*"@drawable/ok"* />  </layer-list> |

## SeekBar拖动条

|  |
| --- |
| <SeekBar  android:id="@+id/seek"  android:layout\_width="fill\_parent"  android:layout\_height="wrap\_content"  android:max="100"  android:progress="50"  android:secondaryProgress="75"/>  类似下图 |
| public class Activity01 extends Activity implements  SeekBar.OnSeekBarChangeListener {  /\*\* Called when the activity is first created. \*/  //声明SeekBar 和 TextView对象  SeekBar mSeekBar;  TextView mProgressText;  TextView mTrackingText;  @Override  public void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.main);  //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);  }  @Override  //在拖动中--即值在改变  public void onProgressChanged(SeekBar seekBar, int progress,  boolean fromUser) {  //progress为当前数值的大小  mProgressText.setText("当前值:" + progress);  }  @Override  //在拖动中会调用此方法  public void onStartTrackingTouch(SeekBar seekBar) {  mTrackingText.setText("xh正在调节");  }  @Override  //停止拖动  public void onStopTrackingTouch(SeekBar seekBar) {  mTrackingText.setText("xh停止调节");  }  } |

## RatingBar星级评分条

|  |
| --- |
| <!-- 定义一个星级评分条 -->  <RatingBar  android:id=*"@+id/rating"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:numStars=*"5"*  android:max=*"255"*  android:progress=*"255"*  android:stepSize=*"0.5"*  /> |
| image = (ImageView) findViewById(R.id.***image***);  RatingBar ratingBar = (RatingBar) findViewById(R.id.***rating***);  ratingBar.setOnRatingBarChangeListener(**new** OnRatingBarChangeListener() {  // 当拖动条的滑块位置发生改变时触发该方法  @Override  **public** **void** onRatingChanged(RatingBar arg0, **float** rating, **boolean** fromUser) {  // 动态改变图片的透明度，其中255是星级评分条的最大值  // 5个星星就代表最大值255  image.~~setAlpha~~((**int**) (rating \* 255 / 5));  }  }); |

## TitleProgressBar

|  |
| --- |
| 无布局文件    bn1.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View source) {  // 显示不带进度的进度条。  setProgressBarIndeterminateVisibility(**true**);  // 显示带进度的进度条。  setProgressBarVisibility(**true**);  // 设置进度条的进度  setProgress(4500);  }  });  bn2.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View source) {  // 隐藏不带进度的进度条。  setProgressBarIndeterminateVisibility(**false**);  // 隐藏带进度的进度条。  setProgressBarVisibility(**false**);  }  }); |

# 各种AdapterView

## BaseAdapter 用java代码写布局

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"vertical"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"* >  <ListView  android:id=*"@+id/myList"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*/>  </LinearLayout>    BaseAdapter adapter = **new** BaseAdapter() {  @Override  **public** **int** getCount() {  // 指定一共包含40个选项  **return** 40;  }  @Override  **public** Object getItem(**int** position) {  **return** **null**;  }  // 重写该方法，该方法的返回值将作为列表项的ID  @Override  **public** **long** getItemId(**int** position) {  **return** position;  }  // 重写该方法，该方法返回的View将作为列表框  @Override  **public** View getView(**int** position, View convertView, ViewGroup parent) {  // 创建一个LinearLayout，并向其中添加2个组件  LinearLayout line = **new** LinearLayout(BaseAdapterTest.**this**);  line.setOrientation(0);  ImageView image = **new** ImageView(BaseAdapterTest.**this**);  image.setImageResource(R.drawable.***ic\_launcher***);  line.addView(image);  TextView text = **new** TextView(BaseAdapterTest.**this**);  text.setText("第" + (position + 1) + "个列表项");  text.setTextSize(20);  text.setTextColor(Color.***RED***);  line.addView(text);  // 返回LinearLayout实例  **return** line;  }  };  ListView myList = (ListView) findViewById(R.id.***myList***);  myList.setAdapter(adapter); |

## ArrayAdapter 简单布局文件+一个数组列表

|  |
| --- |
| ListView list1 = (ListView) findViewById(R.id.***list1***);  // 定义一个数组  String[] arr1 = {"孙悟空", "猪八戒", "牛魔王"};  // 将数组包装ArrayAdapter  ArrayAdapter<String> adapter1 = **new** ArrayAdapter<String>(**this**, R.layout.***array\_item***, arr1);  // 为ListView设置Adapter  list1.setAdapter(adapter1);  array\_item.xml  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <TextView  xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:id=*"@+id/TextView"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:textSize=*"24dp"*  android:padding=*"10px"*  android:shadowColor=*"#f0f"*  android:shadowDx=*"4"*  android:shadowDy=*"4"*  android:shadowRadius=*"2"*/> |
| ListView list2 = (ListView) findViewById(R.id.***list2***);  // 定义一个数组  String[] arr2 = {"Java", "Hibernate", "Spring", "Android"};  // 将数组包装ArrayAdapter  ArrayAdapter<String> adapter2 = **new** ArrayAdapter<String>(**this**, R.layout.***checked\_item***, arr2);  // 为ListView设置Adapter  list2.setAdapter(adapter2);  checked\_item.xml  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <CheckedTextView  xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:id=*"@+id/TextView"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:textSize=*"24dp"*  android:checkMark=*"@drawable/ok"*  android:shadowColor=*"#f0f"*  android:shadowDx=*"4"*  android:shadowDy=*"4"*  android:shadowRadius=*"2"*/> |

## ListView列表视图

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:orientation=*"vertical"* >  <com.baidu.mapapi.map.MapView  android:id=*"@+id/bmapView"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:clickable=*"true"* />  </LinearLayout>  // 无需使用布局文件  String[] arr = {"孙悟空", "猪八戒", "唐僧"};  // 创建ArrayAdapter对象  ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_list\_item\_multiple\_choice***, arr);  // 设置该窗口显示列表  setListAdapter(adapter);    右边选框无法选中 |
| 或者  <!-- 直接使用数组资源给出列表项 -->  <!-- 设置使用红色的分隔条 -->  <ListView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:entries=*"@array/books"*  android:divider=*"#f00"*  android:dividerHeight=*"2px"*  android:headerDividersEnabled=*"false"*  />  arrays.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <resources>  <string-array name=*"books"*>  <item>疯狂Java讲义</item>  <item>疯狂Ajax讲义</item>  <item>疯狂XML讲义</item>  <item>疯狂Android讲义</item>  </string-array>  </resources> |

## SimpleAdapter 复杂布局文件+多个数组

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"horizontal"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  >  <!-- 定义一个List -->  <ListView android:id=*"@+id/mylist"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  />  </LinearLayout>    simple\_item.xml  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"horizontal"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*>  <!-- 定义一个ImageView，用于作为列表项的一部分。 -->  <ImageView android:id=*"@+id/header"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:paddingLeft=*"10dp"*  />  <LinearLayout  android:orientation=*"vertical"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*>  <!-- 定义一个TextView，用于作为列表项的一部分。 -->  <TextView android:id=*"@+id/name"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:textSize=*"20dp"*  android:textColor=*"#f0f"*  android:paddingLeft=*"10dp"*  />  <!-- 定义一个TextView，用于作为列表项的一部分。 -->  <TextView android:id=*"@+id/desc"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:textSize=*"14dp"*  android:paddingLeft=*"10dp"*  />  </LinearLayout>  </LinearLayout> |
| **private** String[] names = **new** String[]{"虎头", "弄玉", "李清照", "李白"};  **private** String[] descs = **new** String[]{"可爱的小孩", "一个擅长音乐的女孩", "一个擅长的文学的女性", "浪漫主义诗人"};  **private** **int**[] imageIds = **new** **int**[]{R.drawable.***tiger***, R.drawable.***nongyu***, R.drawable.***qingzhao***, R.drawable.***libai***};  // 创建一个List集合，List集合的元素是Map  List<Map<String, Object>> listItems = **new** ArrayList<Map<String, Object>>();  **for** (**int** i = 0; i < names.length; i++) {  Map<String, Object> listItem = **new** HashMap<String, Object>();  listItem.put("header", imageIds[i]);  listItem.put("personName", names[i]);  listItem.put("desc", descs[i]);  listItems.add(listItem);  }  // 创建一个SimpleAdapter  SimpleAdapter simpleAdapter = **new** SimpleAdapter(**this**, listItems, R.layout.***simple\_item***, **new** String[]{"personName", "header", "desc"}, **new** **int**[]{R.id.***name***, R.id.***header***, R.id.***desc***});  ListView list = (ListView) findViewById(R.id.***mylist***);  // 为ListView设置Adapter  list.setAdapter(simpleAdapter);  // 为ListView的列表项单击事件绑定事件监听器  list.setOnItemClickListener(**new** OnItemClickListener() {  // 第position项被单击时激发该方法。  @Override  **public** **void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  System.***out***.println(names[position] + "被单击了");  }  });  list.setOnItemSelectedListener(**new** OnItemSelectedListener() {  // 第position项被选中时激发该方法。  @Override  **public** **void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  System.***out***.println(names[position] + "被选中了");  }  @Override  **public** **void** onNothingSelected(AdapterView<?> parent) {  }  }); |

## AutoCompleteTextView自动完成文本框

|  |
| --- |
| <!-- 定义一个自动完成文本框，指定输入一个字符后进行提示 -->  <AutoCompleteTextView  android:id=*"@+id/auto"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:completionHint=*"请选择您喜欢的图书："*  android:dropDownHorizontalOffset=*"10dp"*  android:completionThreshold=*"1"*/>  <!-- 定义一个MultiAutoCompleteTextView组件 -->  <MultiAutoCompleteTextView  android:id=*"@+id/mauto"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:completionThreshold=*"1"*/> |
| String[] books = **new** String[]{  "疯狂Java讲义",  "疯狂JavaSe讲义",  "疯狂Ajax讲义",  "疯狂XML讲义",  "疯狂Workflow讲义"  };  // 创建一个ArrayAdapter，封装数组  ArrayAdapter<String> aa = **new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_dropdown\_item\_1line***, books); |
| //输入框  AutoCompleteTextView actv = (AutoCompleteTextView) findViewById(R.id.***auto***);  actv.setAdapter(aa); // 设置Adapter |
| //弹出的自动完成下拉列表框  MultiAutoCompleteTextView mauto = (MultiAutoCompleteTextView) findViewById(R.id.***mauto***);  mauto.setAdapter(aa); // 设置Adapter  mauto.setTokenizer(**new** MultiAutoCompleteTextView.CommaTokenizer());// 为MultiAutoCompleteTextView设置分隔符 |

## GridView网格视图

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"vertical"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:gravity=*"center\_horizontal"*  >  <!-- 定义一个GridView组件 -->  <GridView  android:id=*"@+id/grid01"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:horizontalSpacing=*"1pt"*  android:verticalSpacing=*"1pt"*  android:numColumns=*"4"*  android:gravity=*"center"*  />  <!-- 定义一个ImageView组件 -->  <ImageView android:id=*"@+id/imageView"*  android:layout\_width=*"240dp"*  android:layout\_height=*"240dp"*  android:layout\_gravity=*"center\_horizontal"*  />  </LinearLayout>  cell.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"horizontal"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:gravity=*"center\_horizontal"*  android:padding=*"2pt"*  >  <ImageView  android:id=*"@+id/image1"*  android:layout\_width=*"50dp"*  android:layout\_height=*"50dp"*  />  </LinearLayout> |
| // 创建一个List对象，List对象的元素是Map  List<Map<String, Object>> listItems = **new** ArrayList<Map<String, Object>>();  **for** (**int** i = 0; i < imageIds.length; i++) {  Map<String, Object> listItem = **new** HashMap<String, Object>();  listItem.put("image", imageIds[i]);  listItems.add(listItem);  }    // 创建一个SimpleAdapter, 使用/layout/cell.xml文件作为界面布局  SimpleAdapter simpleAdapter = **new** SimpleAdapter(**this**, listItems, R.layout.***cell***, **new** String[]{"image"}, **new** **int**[]{R.id.***image1***});    grid = (GridView) findViewById(R.id.***grid01***);  // 为GridView设置Adapter  grid.setAdapter(simpleAdapter);  // 获取显示图片的ImageView  imageView = (ImageView) findViewById(R.id.***imageView***);    // 添加列表项被选中的监听器  grid.setOnItemSelectedListener(**new** OnItemSelectedListener() {  @Override  **public** **void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  // 显示当前被选中的图片  imageView.setImageResource(imageIds[position]);  }  @Override  **public** **void** onNothingSelected(AdapterView<?> parent) {  }  });  // 添加列表项被单击的监听器  grid.setOnItemClickListener(**new** OnItemClickListener() {  @Override  **public** **void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  // 显示被单击的图片的图片  imageView.setImageResource(imageIds[position]);  }  }); |

## ExpandableListAdapter可展开的列表组件(二级菜单)

|  |
| --- |
| <ExpandableListView  android:id=*"@+id/list"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:childIndicator=*"@drawable/ic\_launcher"*  /> |
| // 创建一个BaseExpandableListAdapter对象  ExpandableListAdapter adapter = **new** BaseExpandableListAdapter() {  **int**[] logos = **new** **int**[]{R.drawable.***p***, R.drawable.***z***, R.drawable.***t***};  **private** String[] armTypes = **new** String[]{"神族兵种", "虫族兵种", "人族兵种"};  **private** String[][] arms = **new** String[][]{{"狂战士", "龙骑士", "黑暗圣堂", "电兵"}, {"小狗", "刺蛇", "飞龙", "自爆飞机"}, {"机枪兵", "护士MM", "幽灵"}};  // 获取指定组位置处的组数据  @Override  **public** Object getGroup(**int** groupPosition) {  **return** armTypes[groupPosition];  }  @Override  **public** **int** getGroupCount() {  **return** armTypes.length;  }  @Override  **public** **long** getGroupId(**int** groupPosition) {  **return** groupPosition;  }  // 该方法决定每个组选项的外观  @Override  **public** View getGroupView(**int** groupPosition, **boolean** isExpanded, View convertView, ViewGroup parent) {  LinearLayout ll = **new** LinearLayout(ExpandableListViewTest.**this**);  ll.setOrientation(0);  ImageView logo = **new** ImageView(ExpandableListViewTest.**this**);  logo.setImageResource(logos[groupPosition]);  ll.addView(logo);  TextView textView = getTextView();  textView.setText(getGroup(groupPosition).toString());  ll.addView(textView);  **return** ll;  }  @Override  **public** **boolean** hasStableIds() {  **return** **true**;  }    // 获取指定组位置、指定子列表项处的子列表项数据  @Override  **public** Object getChild(**int** groupPosition, **int** childPosition) {  **return** arms[groupPosition][childPosition];  }  @Override  **public** **long** getChildId(**int** groupPosition, **int** childPosition) {  **return** childPosition;  }  @Override  **public** **int** getChildrenCount(**int** groupPosition) {  **return** arms[groupPosition].length;  }  **private** TextView getTextView() {  AbsListView.LayoutParams lp = **new** AbsListView.LayoutParams(ViewGroup.LayoutParams.***MATCH\_PARENT***, 64);  TextView textView = **new** TextView(ExpandableListViewTest.**this**);  textView.setLayoutParams(lp);  textView.setGravity(Gravity.***CENTER\_VERTICAL*** | Gravity.***LEFT***);  textView.setPadding(36, 0, 0, 0);  textView.setTextSize(20);  **return** textView;  }  // 该方法决定每个子选项的外观  @Override  **public** View getChildView(**int** groupPosition, **int** childPosition, **boolean** isLastChild, View convertView, ViewGroup parent) {  TextView textView = getTextView();  textView.setText(getChild(groupPosition, childPosition).toString());  **return** textView;  }  @Override  **public** **boolean** isChildSelectable(**int** groupPosition, **int** childPosition) {  **return** **true**;  }  };  ExpandableListView expandListView = (ExpandableListView) findViewById(R.id.***list***);  expandListView.setAdapter(adapter); |

## Spinner 下拉选择

|  |
| --- |
| <!-- 定义了一个Spinner组件， 指定该显示该Spinner组件的数组 -->  <Spinner  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:entries=*"@array/books"*  android:prompt=*"@string/tip"*  />  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <resources>  <string-array name=*"books"*>  <item>疯狂Java讲义</item>  <item>疯狂Ajax讲义</item>  <item>疯狂XML讲义</item>  </string-array>  </resources> |
| 另一种方法:  <Spinner  android:id=*"@+id/spinner"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:prompt=*"@string/tip"*  />  // 获取界面布局文件中的Spinner组件  Spinner spinner = (Spinner) findViewById(R.id.***spinner***);  String[] arr = {"孙悟空", "猪八戒", "唐僧"};  // 创建ArrayAdapter对象  ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_list\_item\_multiple\_choice***, arr);  // 为Spinner设置Adapter  spinner.setAdapter(adapter); |

## ~~Gallery画廊视图 (胶卷效果, 任何时候居中的选中)~~

|  |
| --- |
| <Gallery android:id=*"@+id/gallery"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:layout\_marginTop=*"2dp"*  android:unselectedAlpha=*"0.6"*  android:spacing=*"2pt"*  />    下面这行能左右滑动, 上面图片始终显示下面中间的图片. |
| // 创建一个BaseAdapter对象，该对象负责提供Gallery所显示的列表项  BaseAdapter adapter = **new** BaseAdapter() {  @Override  **public** **int** getCount() {  **return** imageIds.length;  }  @Override  **public** Object getItem(**int** position) {  **return** position;  }  @Override  **public** **long** getItemId(**int** position) {  **return** position;  }  // 该方法的返回的View就是代表了每个列表项  @Override  **public** View getView(**int** position, View convertView, ViewGroup parent) {  // 创建一个ImageView  ImageView imageView = **new** ImageView(GallaryTest.**this**);  imageView.setImageResource(imageIds[position]);  // 设置ImageView的缩放类型  imageView.setScaleType(ImageView.ScaleType.***FIT\_XY***);  // 为imageView设置布局参数  imageView.setLayoutParams(**new** ~~Gallery~~.LayoutParams(75, 100));  TypedArray typedArray = obtainStyledAttributes(R.styleable.***Gallery***);  imageView.setBackgroundResource(typedArray.getResourceId(R.styleable.***Gallery\_android\_galleryItemBackground***, 0));  **return** imageView;  }  };    ~~Gallery~~ gallery = (~~Gallery~~) findViewById(R.id.***gallery***);  gallery.setAdapter(adapter);  gallery.setOnItemSelectedListener(**new** OnItemSelectedListener() {  // 当Gallery选中项发生改变时触发该方法  @Override  **public** **void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  imageView.setImageResource(imageIds[position]);  }  @Override  **public** **void** onNothingSelected(AdapterView<?> parent) {  }  }); |

## AdapterViewFlipper 让多个view按动画效果切换

AdapterViewFlipper继承了AdapterViewAnimator. 它也是现实Adapter提供的多个View组件。AdapterViewFilpper可以在多个View切换过程中使用渐隐渐现的动画效果。

|  |
| --- |
| <AdapterViewFlipper  android:id=*"@+id/flipper"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:flipInterval=*"5000"*  android:layout\_alignParentTop=*"true"*/> |
| // 创建一个BaseAdapter对象，该对象负责提供Gallery所显示的列表项  BaseAdapter adapter = **new** BaseAdapter() {  @Override  **public** **int** getCount() {  **return** imageIds.length;  }  @Override  **public** Object getItem(**int** position) {  **return** position;  }  @Override  **public** **long** getItemId(**int** position) {  **return** position;  }  // 该方法的返回的View就是代表了每个列表项  @Override  **public** View getView(**int** position, View convertView, ViewGroup parent) {  // 创建一个ImageView  ImageView imageView = **new** ImageView(AdapterViewFlipperTest.**this**);  imageView.setImageResource(imageIds[position]);  // 设置ImageView的缩放类型  imageView.setScaleType(ImageView.ScaleType.***FIT\_XY***);  // 为imageView设置布局参数  imageView.setLayoutParams(**new** LayoutParams(LayoutParams.***MATCH\_PARENT***, LayoutParams.***MATCH\_PARENT***));  **return** imageView;  }  };  AdapterViewFlipper flipper = (AdapterViewFlipper) findViewById(R.id.***flipper***);  flipper.setAdapter(adapter); |
| **public** **void** prev(View source) {  // 显示上一个组件  flipper.showPrevious();  // 停止自动播放  flipper.stopFlipping();  }  **public** **void** next(View source) {  // 显示下一个组件。  flipper.showNext();  // 停止自动播放  flipper.stopFlipping();  }  **public** **void** auto(View source) {  // 开始自动播放  flipper.startFlipping();  } |

## StaticView栈 有动画效果

|  |
| --- |
| <StackView  android:id=*"@+id/mStackView"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:loopViews=*"true"* />  cell.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"horizontal"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:gravity=*"center\_horizontal"*  android:padding=*"2pt"*  >  <ImageView  android:id=*"@+id/image1"*  android:layout\_width=*"120dp"*  android:layout\_height=*"120dp"*  />  </LinearLayout> |
| StackView stackView;  **int**[] imageIds = **new** **int**[]{R.drawable.***bomb5***, R.drawable.***bomb6***, R.drawable.***bomb7***, R.drawable.***bomb8***, R.drawable.***bomb9***, R.drawable.***bomb10***, R.drawable.***bomb11***, R.drawable.***bomb12***, R.drawable.***bomb13***, R.drawable.***bomb14***, R.drawable.***bomb15***, R.drawable.***bomb16***};  @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);  // 创建一个List对象，List对象的元素是Map  List<Map<String, Object>> listItems = **new** ArrayList<Map<String, Object>>();  **for** (**int** i = 0; i < imageIds.length; i++) {  Map<String, Object> listItem = **new** HashMap<String, Object>();  listItem.put("image", imageIds[i]);  listItems.add(listItem);  }  // 创建一个SimpleAdapter, 使用/layout/cell.xml文件作为界面布局  SimpleAdapter simpleAdapter = **new** SimpleAdapter(**this**, listItems, R.layout.***cell***, **new** String[]{"image"}, **new** **int**[]{R.id.***image1***});    StackView stackView = (StackView) findViewById(R.id.***mStackView***);  stackView.setAdapter(simpleAdapter);  }  **public** **void** prev(View view) {  // 显示上一个组件  stackView.showPrevious();  }  **public** **void** next(View view) {  // 显示下一个组件  stackView.showNext();  } |

# 动画ViewAnimator

## ViewSwitcher视图切换(视图是通过java代码生成的)

|  |
| --- |
| <!-- 定义一个ViewSwitcher组件 -->  <ViewSwitcher  android:id=*"@+id/viewSwitcher"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"* />    <!-- 定义滚动到上一屏的按钮 -->  <Button  android:id=*"@+id/button\_prev"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignParentBottom=*"true"*  android:layout\_alignParentLeft=*"true"*  android:onClick=*"prev"*  android:text=*"&lt;"* />    <!-- 定义滚动到下一屏的按钮 -->  <Button  android:id=*"@+id/button\_next"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignParentBottom=*"true"*  android:layout\_alignParentRight=*"true"*  android:onClick=*"next"*  android:text=*"&gt;"* /> |
| ViewSwitcher switcher;  // 创建LayoutInflater对象  LayoutInflater inflater;  @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);    // 计算应用程序所占的总屏数。  // 如果应用程序的数量能整除NUMBER\_PER\_SCREEN，除法的结果就是总屏数。  // 如果不能整除，总屏数应该是除法的结果再加1。  screenCount = items.size() % ***NUMBER\_PER\_SCREEN*** == 0 ? items.size() / ***NUMBER\_PER\_SCREEN*** : items.size() / ***NUMBER\_PER\_SCREEN*** + 1;`  ViewSwitcher switcher = (ViewSwitcher) findViewById(R.id.***viewSwitcher***);  switcher.setFactory(**new** ViewFactory() {  // 实际上就是返回一个GridView组件  @Override  **public** View makeView() {  // 加载R.layout.slidelistview组件，实际上就是一个GridView组件。  **return** inflater.inflate(R.layout.***slidelistview***, **null**);  }  });  // 页面加载时先显示第一屏。  next(**null**);  }  **public** **void** next(View v) {  **if** (screenNo < screenCount - 1) {  screenNo++;  // 为ViewSwitcher的组件显示过程设置动画  switcher.setInAnimation(**this**, R.anim.***slide\_in\_right***);  // 为ViewSwitcher的组件隐藏过程设置动画  switcher.setOutAnimation(**this**, R.anim.***slide\_out\_left***);  // 控制下一屏将要显示的GridView对应的 Adapter  ((GridView) switcher.getNextView()).setAdapter(adapter);  // 点击右边按钮，显示下一屏，  // 学习手势检测后，也可通过手势检测实现显示下一屏.  switcher.showNext(); // ①  }  }  **public** **void** prev(View v) {  **if** (screenNo > 0) {  screenNo--;  // 为ViewSwitcher的组件显示过程设置动画  switcher.setInAnimation(**this**, android.R.anim.***slide\_in\_left***);  // 为ViewSwitcher的组件隐藏过程设置动画  switcher.setOutAnimation(**this**, android.R.anim.***slide\_out\_right***);  // 控制下一屏将要显示的GridView对应的 Adapter  ((GridView) switcher.getNextView()).setAdapter(adapter);  // 点击左边按钮，显示上一屏，当然可以采用手势  // 学习手势检测后，也可通过手势检测实现显示上一屏.  switcher.showPrevious(); // ②  }  } |

## ImageSwitcher图像切换器, 和ImageView区别仅在于有进出动画效果

|  |
| --- |
| <!-- 定义一个ImageSwitcher组件 -->  <ImageSwitcher android:id=*"@+id/switcher"*  android:layout\_width=*"300dp"*  android:layout\_height=*"300dp"*  android:layout\_gravity=*"center\_horizontal"*  android:inAnimation=*"@android:anim/fade\_in"*  android:outAnimation=*"@android:anim/fade\_out"*  />  cell.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"horizontal"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:gravity=*"center\_horizontal"*  android:padding=*"4pt"*  >  <ImageView  android:id=*"@+id/image1"*  android:layout\_width=*"50dp"*  android:layout\_height=*"50dp"*  />  </LinearLayout> |
| // 创建一个List对象，List对象的元素是Map  List<Map<String, Object>> listItems = **new** ArrayList<Map<String, Object>>();  **for** (**int** i = 0; i < imageIds.length; i++) {  Map<String, Object> listItem = **new** HashMap<String, Object>();  listItem.put("image", imageIds[i]);  listItems.add(listItem);  }  // 获取显示图片的ImageSwitcher  switcher = (ImageSwitcher) findViewById(R.id.***switcher***);  // 为ImageSwitcher设置图片切换的动画效果  switcher.setFactory(**new** ViewFactory() {  @Override  **public** View makeView() {  // 创建ImageView对象  ImageView imageView = **new** ImageView(ImageSwitcherTest.**this**);  imageView.setScaleType(ImageView.ScaleType.***FIT\_CENTER***);  imageView.setLayoutParams(**new** ImageSwitcher.LayoutParams(LayoutParams.***WRAP\_CONTENT***, LayoutParams.***WRAP\_CONTENT***));  // 返回ImageView对象  **return** imageView;  }  });  // 创建一个SimpleAdapter, 使用/layout/cell.xml文件作为界面布局  SimpleAdapter simpleAdapter = **new** SimpleAdapter(**this**, listItems, R.layout.***cell***, **new** String[]{"image"}, **new** **int**[]{R.id.***image1***});  GridView grid = (GridView) findViewById(R.id.***grid01***);  // 为GridView设置Adapter  grid.setAdapter(simpleAdapter);  // 添加列表项被选中的监听器  grid.setOnItemSelectedListener(**new** OnItemSelectedListener() {  @Override  **public** **void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {  // 显示当前被选中的图片  switcher.setImageResource(imageIds[position]);  }  @Override  **public** **void** onNothingSelected(AdapterView<?> parent) {  }  });  // 添加列表项被单击的监听器  grid.setOnItemClickListener(**new** OnItemClickListener() {  @Override  **public** **void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  // 显示被单击的图片的图片  switcher.setImageResource(imageIds[position]);  }  }); |
|  |

## TextSwitcher文字切换器(文字是通过java代码生成的)

|  |
| --- |
| <!-- 定义一个TextSwitcher，并指定了文本切换时的动画效果 -->  <TextSwitcher  android:id=*"@+id/textSwitcher"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:inAnimation=*"@android:anim/slide\_in\_left"*  android:outAnimation=*"@android:anim/slide\_out\_right"*  android:onClick=*"next"*  />    文字从左到右切换 |
| @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);    textSwitcher = (TextSwitcher) findViewById(R.id.***textSwitcher***);  textSwitcher.setFactory(**new** ViewSwitcher.ViewFactory() {  **public** View makeView() {  TextView tv = **new** TextView(TextSwitcherTest.**this**);  tv.setTextSize(40);  tv.setTextColor(Color.***MAGENTA***);  **return** tv;  }  });  // 调用next方法显示下一个字符串  next(**null**);  }  // 事件处理函数，控制显示下一个字符串  **public** **void** next(View source) {  textSwitcher.setText(strs[curStr++ % strs.length]); // ①  } |

## ViewFlipper视图切换(视图已经在布局文件中)

|  |
| --- |
| <ViewFlipper  android:id=*"@+id/details"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:persistentDrawingCache=*"animation"*  android:flipInterval=*"1000"*>  <ImageView  android:src=*"@drawable/java"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*>  </ImageView>  <ImageView  android:src=*"@drawable/android"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*>  </ImageView>  <ImageView  android:src=*"@drawable/ee"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*>  </ImageView>  </ViewFlipper>  <Button  android:text=*"&lt;"*  android:onClick=*"prev"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignParentBottom=*"true"*  android:layout\_alignParentLeft=*"true"* />  <Button  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignParentBottom=*"true"*  android:layout\_centerInParent=*"true"*  android:onClick=*"auto"*  android:text=*"自动播放"* />  <Button  android:text=*"&gt;"*  android:onClick=*"next"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignParentBottom=*"true"*  android:layout\_alignParentRight=*"true"* /> |
| @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);  viewFlipper = (ViewFlipper) findViewById(R.id.***details***);  }  **public** **void** prev(View source) {  viewFlipper.setInAnimation(**this**, R.anim.***slide\_in\_right***);  viewFlipper.setOutAnimation(**this**, R.anim.***slide\_out\_left***);  // 显示上一个组件  viewFlipper.showPrevious();  // 停止自动播放  viewFlipper.stopFlipping();  }  **public** **void** next(View source) {  viewFlipper.setInAnimation(**this**, android.R.anim.***slide\_in\_left***);  viewFlipper.setOutAnimation(**this**, android.R.anim.***slide\_out\_right***);  // 显示下一个组件。  viewFlipper.showNext();  // 停止自动播放  viewFlipper.stopFlipping();  }  **public** **void** auto(View source) {  viewFlipper.setInAnimation(**this**, android.R.anim.***slide\_in\_left***);  viewFlipper.setOutAnimation(**this**, android.R.anim.***slide\_out\_right***);  // 开始自动播放  viewFlipper.startFlipping();  } |

# 其它UI组件

## Toast

|  |
| --- |
| // 创建一个Toast提示信息, 第二个参数设置该Toast提示信息的持续时间  Toast toast = Toast.*makeText*(ToastTest.**this**, "简单的提示信息", Toast.***LENGTH\_SHORT***);  toast.show(); |
| 带图片的toast  Button bn = (Button) findViewById(R.id.***bn***);  // 为按钮的单击事件绑定事件监听器  bn.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View source) {  // 创建一个LinearLayout容器  LinearLayout ll = **new** LinearLayout(ToastTest.**this**);    // 创建一个ImageView  ImageView image = **new** ImageView(ToastTest.**this**);  image.setImageResource(R.drawable.***tools***);  ll.addView(image);    // 创建一个ImageView  TextView textView = **new** TextView(ToastTest.**this**);  textView.setText("带图片的提示信");  textView.setTextSize(30);  textView.setTextColor(Color.***MAGENTA***);  ll.addView(textView);    // 创建一个Toast提示信息  Toast toast = **new** Toast(ToastTest.**this**);  toast.setGravity(Gravity.***CENTER***, 0, 0); // 设置Toast的显示位置  toast.setView(ll); // 设置Toast显示自定义View  toast.setDuration(Toast.***LENGTH\_LONG***); // 设置Toast的显示时间  toast.show();  }  }); |

## 日历CalendarView

|  |
| --- |
| <!--  对该组件的日期时间进行了定制  -->  <CalendarView  android:id=*"@+id/calendarView"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:firstDayOfWeek=*"3"* 设置以星期二作为每周第一天  android:focusedMonthDateColor=*"#f00"*  android:selectedWeekBackgroundColor=*"#aff"*  android:shownWeekCount=*"4"* 设置该组件总共显示3个星期  android:unfocusedMonthDateColor=*"#f9f"*  android:weekSeparatorLineColor=*"#ff0"* /> |
| cv = (CalendarView) findViewById(R.id.***calendarView***);  // 为CalendarView组件的日期改变事件添加事件监听器  cv.setOnDateChangeListener(**new** OnDateChangeListener() {  @Override  **public** **void** onSelectedDayChange(CalendarView view, **int** year, **int** month, **int** dayOfMonth) {  // 使用Toast显示用户选择的日期  Toast.*makeText*(CalendarViewTest.**this**, "你生日是" + year + "年" + month + "月" + dayOfMonth + "日", Toast.***LENGTH\_SHORT***).show();  }  }); |

## 日期时间选择器DatePicker & TimePicker

|  |
| --- |
| <!-- 定义一个DatePicker组件 -->  <DatePicker  android:id=*"@+id/datePicker"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"200dp"*  android:layout\_gravity=*"center\_horizontal"*  android:calendarViewShown=*"true"*  android:endYear=*"2012"*  android:spinnersShown=*"true"*  android:startYear=*"2000"* />    // 获取当前的年、月、日、小时、分钟  Calendar c = Calendar.*getInstance*();  year = c.get(Calendar.***YEAR***);  month = c.get(Calendar.***MONTH***);  day = c.get(Calendar.***DAY\_OF\_MONTH***);  hour = c.get(Calendar.***HOUR***);  minute = c.get(Calendar.***MINUTE***);  DatePicker datePicker = (DatePicker) findViewById(R.id.***datePicker***);  // 初始化DatePicker组件，初始化时指定监听器  datePicker.init(year, month, day, **new** OnDateChangedListener() {  @Override  **public** **void** onDateChanged(DatePicker arg0, **int** year, **int** month, **int** day) {  ChooseDate.**this**.year = year;  ChooseDate.**this**.month = month;  ChooseDate.**this**.day = day;  }  }); |
| <!-- 定义一个TimePicker组件 -->  <TimePicker  android:id=*"@+id/timePicker"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"100dp"*  android:layout\_gravity=*"center\_horizontal"* />    TimePicker timePicker = (TimePicker) findViewById(R.id.***timePicker***);  // 为TimePicker指定监听器  timePicker.setOnTimeChangedListener(**new** OnTimeChangedListener() {  @Override  **public** **void** onTimeChanged(TimePicker view, **int** hourOfDay, **int** minute) {  ChooseDate.**this**.hour = hourOfDay;  ChooseDate.**this**.minute = minute;  }  }); |

## 数值选择器

|  |
| --- |
| <NumberPicker  android:id=*"@+id/np1"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"80dp"*  android:focusable=*"true"*  android:focusableInTouchMode=*"true"* /> |
| np1 = (NumberPicker) findViewById(R.id.***np1***);  // 设置np1的最小值和最大值  np1.setMinValue(10);  np1.setMaxValue(50);  // 设置np1的当前值  np1.setValue(minPrice);  np1.setOnValueChangedListener(**new** OnValueChangeListener() {  // 当NumberPicker的值发生改变时，将会激发该方法  @Override  **public** **void** onValueChange(NumberPicker picker, **int** oldVal, **int** newVal) {  minPrice = newVal;  showSelectedPrice();  }  }); |

## 搜索框 SearchView

|  |
| --- |
| <!— 定义一个SearchView -->  <SearchView  android:id=*"@+id/sv"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"* />  <!-- 为SearchView定义自动完成的ListView -->  <ListView  android:id=*"@+id/lv"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"0dp"*  android:layout\_weight=*"1"* /> |
| **private** SearchView sv;  **private** ListView lv;  // 自动完成的列表  **private** **final** String[] mStrings = {"aaaaa", "bbbbbb", "cccccc"};  @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);    lv = (ListView) findViewById(R.id.***lv***);  lv.setAdapter(**new** ArrayAdapter<String>(**this**, android.R.layout.***simple\_list\_item\_1***, mStrings));  lv.setTextFilterEnabled(**true**);    sv = (SearchView) findViewById(R.id.***sv***);  sv.setIconifiedByDefault(**false**); // 设置该SearchView默认是否自动缩小为图标  sv.setOnQueryTextListener(**this**); // 为该SearchView组件设置事件监听器  sv.setSubmitButtonEnabled(**true**); // 设置该SearchView显示搜索按钮  sv.setQueryHint("查找");// 设置该SearchView内默认显示的提示文本  }  // 用户输入字符时激发该方法  @Override  **public** **boolean** onQueryTextChange(String newText) {  **if** (TextUtils.*isEmpty*(newText)) {  lv.clearTextFilter();// 清除ListView的过滤  } **else** {  lv.setFilterText(newText); // 使用用户输入的内容对ListView的列表项进行过滤  }  **return** **true**;  }  // 单击搜索按钮时激发该方法  @Override  **public** **boolean** onQueryTextSubmit(String query) {  // 实际应用中应该在该方法内执行实际查询  // 此处仅使用Toast显示用户输入的查询内容  Toast.*makeText*(**this**, "您的选择是:" + query, Toast.***LENGTH\_SHORT***).show();  **return** **false**;  } |

## 选项卡 TabHost

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <TabHost xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:id=*"@android:id/tabhost"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:layout\_weight=*"1"* >  <LinearLayout  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:orientation=*"vertical"* >  <TabWidget  android:id=*"@android:id/tabs"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"* />  <FrameLayout  android:id=*"@android:id/tabcontent"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"* >  <!-- 定义第一个标签页的内容 -->  <LinearLayout  android:id=*"@+id/tab01"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:orientation=*"vertical"* >  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"女儿国国王 - 2012/12/12"*  android:textSize=*"11pt"* />  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"东海龙女 - 2012/12/18"*  android:textSize=*"11pt"* />  </LinearLayout>  <!-- 定义第二个标签页的内容 -->  <LinearLayout  android:id=*"@+id/tab02"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:orientation=*"vertical"* >  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"白骨精 - 2012/08/12"*  android:textSize=*"11pt"* />  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"蜘蛛精 - 2012/09/20"*  android:textSize=*"11pt"* />  </LinearLayout>  <!-- 定义第三个标签页的内容 -->  <LinearLayout  android:id=*"@+id/tab03"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:orientation=*"vertical"*  android:textSize=*"11pt"* >  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"孙悟空 - 2012/09/19"*  android:textSize=*"11pt"* />  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"猪八戒 - 2012/10/12"*  android:textSize=*"11pt"* />  </LinearLayout>  </FrameLayout>  </LinearLayout>  </TabHost> |
| // 获取该Activity里面的TabHost组件  TabHost tabHost = getTabHost();  // 创建第一个Tab页  TabSpec tab1 = tabHost.newTabSpec("tab1");  tab1.setIndicator("已接电话"); // 设置标题  tab1.setContent(R.id.***tab01***); // 设置内容  tabHost.addTab(tab1);// 添加第一个标签页    TabSpec tab2 = tabHost.newTabSpec("tab2");  // 在标签标题上放置图标  tab2.setIndicator("呼出电话", getResources().getDrawable(R.drawable.***ic\_launcher***)).setContent(R.id.***tab02***);  tabHost.addTab(tab2);// 添加第二个标签页    TabSpec tab3 = tabHost.newTabSpec("tab3").setIndicator("未接电话").setContent(R.id.***tab03***);  tabHost.addTab(tab3);// 添加第三个标签页 |

## 滚动视图

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <!-- 定义ScrollView，为里面的组件添加垂直滚动条 -->  <ScrollView xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"* >  <!-- 定义HorizontalScrollView，为里面的组件添加水平滚动条 -->  <HorizontalScrollView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"* >  <LinearLayout  android:layout\_width=*"match\_parent"*  android:layout\_height=*"fill\_parent"*  android:orientation=*"vertical"* >  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"滚动视图滚动视图滚动视图滚动视图"*  android:textSize=*"30dp"* />  <TextView  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"滚动视图滚动视图滚动视图滚动视图"*  android:textSize=*"30dp"* />  ...........  </LinearLayout>  </HorizontalScrollView>  </ScrollView> |

## Notification 系统通知

|  |
| --- |
| <Button  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"发送Notification"*  android:onClick=*"send"*  />  **static** **final** **int** ***NOTIFICATION\_ID*** = 0x123;  NotificationManager nm;  @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);  // 获取系统的NotificationManager服务  nm = (NotificationManager) getSystemService(***NOTIFICATION\_SERVICE***);  }  // 为发送通知的按钮的点击事件定义事件处理方法  **public** **void** send(View source) {  // 创建一个启动其他Activity的Intent  Intent intent = **new** Intent(NotificationTest.**this**, OtherActivity.**class**);  PendingIntent pi = PendingIntent.*getActivity*(NotificationTest.**this**, 0, intent, 0);  Notification notify = **new** Notification.Builder(**this**)  .setAutoCancel(**true**) // 设置打开该通知，该通知自动消失  .setTicker("有新消息")// 设置显示在状态栏的通知提示信息  .setSmallIcon(R.drawable.***notify***) // 设置通知的图标  .setContentTitle("一条新通知")// 设置通知内容的标题  .setContentText("恭喜你，您加薪了，工资增加20%!")// 设置通知内容  // // 设置使用系统默认的声音、默认LED灯  // .setDefaults(Notification.DEFAULT\_SOUND  // |Notification.DEFAULT\_LIGHTS)  // 设置通知的自定义声音 : res/raw/msg.mp3  .setSound(Uri.*parse*("android.resource://org.crazyit.ui/" + R.raw.***msg***)).setWhen(System.*currentTimeMillis*())  // 设改通知将要启动程序的Intent  .setContentIntent(pi).build();  // 发送通知  nm.notify(***NOTIFICATION\_ID***, notify);  } |
| // 为删除通知的按钮的点击事件定义事件处理方法  **public** **void** del(View v) {  // 取消通知  nm.cancel(***NOTIFICATION\_ID***);  } |
| <Button  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"删除Notification"*  android:onClick=*"del"*  />  // 为删除通知的按钮的点击事件定义事件处理方法  **public** **void** del(View v) {  // 取消通知  nm.cancel(***NOTIFICATION\_ID***);  } |

# 对话框

## AlertDialog

|  |
| --- |
| <!--*简单对话框*-->    <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"简单对话框"*  android:onClick=*"simple"*  />  **public** **void** simple(View source) {  AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  // 设置对话框标题  builder.setTitle("简单对话框");  // 设置图标  builder.setIcon(R.drawable.***tools***).setMessage("对话框的测试内容\n第二行内容");  // 为AlertDialog.Builder添加【确定】按钮  setPositiveButton(builder);  // 为AlertDialog.Builder添加【取消】按钮  setNegativeButton(builder).create().show();  } |
| <!--*简单列表项对话框*-->    <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"简单列表项对话框"*  android:onClick=*"simpleList"*  />  **public** **void** simpleList(View source) {  AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  // 设置对话框标题  builder.setTitle("简单列表项对话框");  // 设置图标  builder.setIcon(R.drawable.***tools***);  // 设置简单的列表项内容  builder.setItems(items, **new** OnClickListener() {  @Override  **public** **void** onClick(DialogInterface dialog, **int** which) {  show.setText("你选中了《" + items[which] + "》");  }  });  // 为AlertDialog.Builder添加【确定】按钮  setPositiveButton(builder);  // 为AlertDialog.Builder添加【取消】按钮  setNegativeButton(builder).create().show();  } |
| <!--*单选列表项对话框*-->    <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"单选列表项对话框"*  android:onClick=*"singleChoice"*  />  **public** **void** singleChoice(View source) {  AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  // 设置对话框标题  builder.setTitle("单选列表项对话框");  // 设置图标  builder.setIcon(R.drawable.***tools***);  // 设置单选列表项,默认选中第二项（索引为1）  builder.setSingleChoiceItems(items, 1, **new** OnClickListener() {  @Override  **public** **void** onClick(DialogInterface dialog, **int** which) {  show.setText("你选中了《" + items[which] + "》");  }  });  // 为AlertDialog.Builder添加【确定】按钮  setPositiveButton(builder);  // 为AlertDialog.Builder添加【取消】按钮  setNegativeButton(builder).create().show();  } |
| <!--*多选列表项对话框*-->    <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"多选列表项对话框"*  android:onClick=*"multiChoice"*  />  **public** **void** multiChoice(View source) {  AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  // 设置对话框标题  builder.setTitle("多选列表项对话框");  // 设置图标  builder.setIcon(R.drawable.***tools***);  // 设置多选列表项，设置勾选第2项、第4项  builder.setMultiChoiceItems(items, **new** **boolean**[]{**false**, **true**, **false**, **true**}, **null**);  // 为AlertDialog.Builder添加【确定】按钮  setPositiveButton(builder);  // 为AlertDialog.Builder添加【取消】按钮  setNegativeButton(builder).create().show();  } |
| <!-- 定义一个普通的按钮组件 -->    <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"自定义列表项对话框"*  android:onClick=*"customList"*  />  array\_item.xml  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <TextView xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:id=*"@+id/TextView"*  android:textColor=*"#f0f"*  android:textSize=*"30dp"*  android:shadowColor=*"#ff0"*  android:shadowRadius=*"2"*  android:shadowDx=*"5"*  android:shadowDy=*"5"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"* />  **public** **void** customList(View source) {  AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  // 设置对话框标题  builder.setTitle("自定义列表项对话框");  // 设置图标  builder.setIcon(R.drawable.***tools***);  // 设置自定义列表项  builder.setAdapter(**new** ArrayAdapter<String>(**this**, R.layout.***array\_item***, items), **null**);  // 为AlertDialog.Builder添加【确定】按钮  setPositiveButton(builder);  // 为AlertDialog.Builder添加【取消】按钮  setNegativeButton(builder).create().show();  } |
| <!--*自定义View对话框*-->    <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"自定义View对话框"*  android:onClick=*"customView"*  />  login.xml  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <TableLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:id=*"@+id/loginForm"*  android:orientation=*"vertical"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  >  <TableRow>  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"用户名:"*  android:textSize=*"10pt"*  />  <!-- 输入用户名的文本框 -->  <EditText  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:hint=*"请填写登录帐号"*  android:selectAllOnFocus=*"true"*  />  </TableRow>  <TableRow>  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"密码:"*  android:textSize=*"10pt"*  />  <!-- 输入密码的文本框 -->  <EditText  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:hint=*"请填写密码"*  android:password=*"true"*  />  </TableRow>  <TableRow>  <TextView  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"电话号码:"*  android:textSize=*"10pt"*  />  <!-- 输入电话号码的文本框 -->  <EditText  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:hint=*"请填写您的电话号码"*  android:selectAllOnFocus=*"true"*  android:phoneNumber=*"true"*  />  </TableRow>  </TableLayout>  **public** **void** customView(View source) {  // 装载/res/layout/login.xml界面布局  TableLayout loginForm = (TableLayout) getLayoutInflater().inflate(R.layout.***login***, **null**);  AlertDialog.Builder builder = **new** AlertDialog.Builder(**this**);  // 设置对话框的图标  builder.setIcon(R.drawable.***tools***);  // 设置对话框的标题  builder.setTitle("自定义View对话框");  // 设置对话框显示的View对象  builder.setView(loginForm);  // 为对话框设置一个“确定”按钮  builder.setPositiveButton("登录", **new** OnClickListener() {  @Override  **public** **void** onClick(DialogInterface dialog, **int** which) {  // 此处可执行登录处理  }  });  // 为对话框设置一个“取消”按钮  builder.setNegativeButton("取消", **new** OnClickListener() {  @Override  **public** **void** onClick(DialogInterface dialog, **int** which) {  // 取消登录，不做任何事情。  }  });  // 创建、并显示对话框  builder.create().show();  } |

## 悬停对话框(activity以对话框的形式显示)

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"vertical"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:gravity=*"center\_horizontal"*>  <ImageView  android:layout\_width=*"240dp"*  android:layout\_height=*"wrap\_content"*  android:src=*"@drawable/java"*  />  <Button android:id=*"@+id/bn"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"确定"*  />  </LinearLayout> |
| @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);  Button bn = (Button) findViewById(R.id.***bn***);  // 为按钮绑定事件监听器  bn.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View arg0) {  // 结束该Activity  // finish();  }  });  } |

## PopupWindow

|  |
| --- |
| ***popup.xml***  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:orientation=*"vertical"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:gravity=*"center\_horizontal"*>  <ImageView  android:layout\_width=*"240dp"*  android:layout\_height=*"wrap\_content"*  android:src=*"@drawable/java"*  />  <Button  android:id=*"@+id/close"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:text=*"关闭"*  />  </LinearLayout> |
| // 装载R.layout.popup对应的界面布局  View root = **this**.getLayoutInflater().inflate(R.layout.***popup***, **null**);  // 创建PopupWindow对象  **final** PopupWindow popup = **new** PopupWindow(root, 280, 360);    Button button = (Button) findViewById(R.id.***bn***);  button.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View v) {  // 以下拉方式显示。  // popup.showAsDropDown(v);  // 将PopupWindow显示在指定位置  popup.showAtLocation(findViewById(R.id.***bn***), Gravity.***CENTER***, 20, 20);  }  });    // 获取PopupWindow中的关闭按钮。  root.findViewById(R.id.***close***).setOnClickListener(**new** View.OnClickListener() {  **public** **void** onClick(View v) {  // 关闭PopupWindow  popup.dismiss(); // ①  }  }); |

## DatePickerDialog & TimePickerDialog

|  |
| --- |
| Button dateBn = (Button) findViewById(R.id.***dateBn***);  // 为“设置日期”按钮绑定监听器。  dateBn.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View source) {  Calendar c = Calendar.*getInstance*();  // 直接创建一个DatePickerDialog对话框实例，并将它显示出来  **new** DatePickerDialog(DateDialogTest.**this**,  // 绑定监听器  **new** DatePickerDialog.OnDateSetListener() {  @Override  **public** **void** onDateSet(DatePicker dp, **int** year, **int** month, **int** dayOfMonth) {  EditText show = (EditText) findViewById(R.id.***show***);  show.setText("您选择了：" + year + "年" + (month + 1) + "月" + dayOfMonth + "日");  }  }  // 设置初始日期  , c.get(Calendar.***YEAR***), c.get(Calendar.***MONTH***), c.get(Calendar.***DAY\_OF\_MONTH***)).show();  }  }); |
| Button timeBn = (Button) findViewById(R.id.***timeBn***);  // 为“设置时间”按钮绑定监听器。  timeBn.setOnClickListener(**new** OnClickListener() {  @Override  **public** **void** onClick(View source) {  Calendar c = Calendar.*getInstance*();  // 创建一个TimePickerDialog实例，并把它显示出来。  **new** TimePickerDialog(DateDialogTest.**this**,  // 绑定监听器  **new** TimePickerDialog.OnTimeSetListener() {  @Override  **public** **void** onTimeSet(TimePicker tp, **int** hourOfDay, **int** minute) {  EditText show = (EditText) findViewById(R.id.***show***);  show.setText("您选择了：" + hourOfDay + "时" + minute + "分");  }  }  // 设置初始时间  , c.get(Calendar.***HOUR\_OF\_DAY***), c.get(Calendar.***MINUTE***)  // true表示采用24小时制  , **true**).show();  }  }); |

## ProgressDialog 进度条对话框

|  |
| --- |
| 环形进度条    <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"环形进度条"*  android:onClick=*"showSpinner"* />  **public** **void** showSpinner(View source) {  // 调用静态方法显示环形进度条  ProgressDialog.*show*(**this**, "任务执行中", "任务执行中，请等待", **false**, **true**); // ①  } |
| 不显示进度的进度条    <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"不显示进度的进度条"*  android:onClick=*"showIndeterminate"* />  **public** **void** showIndeterminate(View source) {  pd1 = **new** ProgressDialog(ProgressDialogTest.**this**);  // 设置对话框的标题  pd1.setTitle("任务正在执行中");  // 设置对话框显示的内容  pd1.setMessage("任务正在执行中，敬请等待...");  // 设置对话框能用“取消”按钮关闭  pd1.setCancelable(**true**);  // 设置对话框的进度条风格  pd1.setProgressStyle(ProgressDialog.***STYLE\_HORIZONTAL***);  // 设置对话框的进度条是否显示进度  pd1.setIndeterminate(**true**);  pd1.show(); // ②  } |
| 显示进度的进度条  <Button  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"显示进度的进度条"*  android:onClick=*"showProgress"* />  **public** **void** showProgress(View source) {  // 将进度条的完成进度重设为0  progressStatus = 0;  // 重新开始填充数组。  hasData = 0;  pd2 = **new** ProgressDialog(ProgressDialogTest.**this**);  pd2.setMax(***MAX\_PROGRESS***);  // 设置对话框的标题  pd2.setTitle("任务完成百分比");  // 设置对话框 显示的内容  pd2.setMessage("耗时任务的完成百分比");  // 设置对话框不能用“取消”按钮关闭  pd2.setCancelable(**false**);  // 设置对话框的进度条风格  pd2.setProgressStyle(ProgressDialog.***STYLE\_HORIZONTAL***);  // 设置对话框的进度条是否显示进度  pd2.setIndeterminate(**false**);  pd2.show(); // ③  **new** Thread() {  **public** **void** run() {  **while** (progressStatus < ***MAX\_PROGRESS***) {  // 获取耗时操作的完成百分比  progressStatus = ***MAX\_PROGRESS*** \* doWork() / data.length;  // 发送空消息到Handler  handler.sendEmptyMessage(0x123);  }  // 如果任务已经完成  **if** (progressStatus >= ***MAX\_PROGRESS***) {  // 关闭对话框  pd2.dismiss();  }  }  }.start();  }  // 模拟一个耗时的操作。  **public** **int** doWork() {  // 为数组元素赋值  data[hasData++] = (**int**) (Math.*random*() \* 100);  **try** {  Thread.*sleep*(100);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  **return** hasData;  } |

# 菜单

## 选项菜单和子菜单(二级菜单)

|  |
| --- |
| // 当用户单击MENU键时触发该方法  @Override  **public** **boolean** onCreateOptionsMenu(Menu menu) {    // -------------向menu中添加字体大小的子菜单-------------  SubMenu fontMenu = menu.addSubMenu("字体大小");  fontMenu.setIcon(R.drawable.***font***);// 设置菜单的图标  fontMenu.setHeaderIcon(R.drawable.***font***);// 设置菜单头的图标  fontMenu.setHeaderTitle("选择字体大小");// 设置菜单头的标题  fontMenu.add(0, FONT\_10, 0, "10号字体");  fontMenu.add(0, FONT\_12, 0, "12号字体");  fontMenu.add(0, FONT\_14, 0, "14号字体");  fontMenu.add(0, FONT\_16, 0, "16号字体");  fontMenu.add(0, FONT\_18, 0, "18号字体");    // -------------向menu中添加普通菜单项-------------  menu.add(0, PLAIN\_ITEM, 0, "普通菜单项");    // -------------向menu中添加文字颜色的子菜单-------------  SubMenu colorMenu = menu.addSubMenu("字体颜色");  colorMenu.setIcon(R.drawable.***color***);  colorMenu.setHeaderIcon(R.drawable.***color***);// 设置菜单头的图标  colorMenu.setHeaderTitle("选择文字颜色");// 设置菜单头的标题  colorMenu.add(0, FONT\_RED, 0, "红色");  colorMenu.add(0, FONT\_GREEN, 0, "绿色");  colorMenu.add(0, FONT\_BLUE, 0, "蓝色");    **return** **super**.onCreateOptionsMenu(menu);  } |
| // 选项菜单的菜单项被单击后的回调方法  @Override  **public** **boolean** onOptionsItemSelected(MenuItem mi) {  // 判断单击的是哪个菜单项，并针对性的作出响应。  **switch** (mi.getItemId()) {  **case** FONT\_10 :  edit.setTextSize(10 \* 2);  **break**;  **case** FONT\_12 :  edit.setTextSize(12 \* 2);  **break**;  **case** FONT\_14 :  edit.setTextSize(14 \* 2);  **break**;  **case** FONT\_16 :  edit.setTextSize(16 \* 2);  **break**;  **case** FONT\_18 :  edit.setTextSize(18 \* 2);  **break**;  **case** FONT\_RED :  edit.setTextColor(Color.***RED***);  **break**;  **case** FONT\_GREEN :  edit.setTextColor(Color.***GREEN***);  **break**;  **case** FONT\_BLUE :  edit.setTextColor(Color.***BLUE***);  **break**;  **case** PLAIN\_ITEM :  Toast toast = Toast.*makeText*(MenuTest.**this**, "您单击了普通菜单项", Toast.***LENGTH\_SHORT***);  toast.show();  **break**;  }  **return** **true**;  } |

## 单选和复选菜单项, 并且用资源文件创建菜单项

|  |
| --- |
| @Override  **public** **boolean** onCreateOptionsMenu(Menu menu) {  MenuInflater inflator = **new** MenuInflater(**this**);  // 状态R.menu.context对应的菜单，并添加到menu中  inflator.inflate(R.menu.***my\_menu***, menu);  **return** **super**.onCreateOptionsMenu(menu);  }  my\_menu.xml  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <menu xmlns:android=*"http://schemas.android.com/apk/res/android"*>  <item android:title=*"@string/font\_size"*  android:icon=*"@drawable/font"*>  <menu>  <!-- 定义一组单选菜单项 -->  <group android:checkableBehavior=*"single"*>  <!-- 定义多个菜单项 -->  <item  android:id=*"@+id/font\_10"*  android:title=*"@string/font\_10"*/>  <item  android:id=*"@+id/font\_12"*  android:title=*"@string/font\_12"*/>  <item  android:id=*"@+id/font\_14"*  android:title=*"@string/font\_14"*/>  <item  android:id=*"@+id/font\_16"*  android:title=*"@string/font\_16"*/>  <item  android:id=*"@+id/font\_18"*  android:title=*"@string/font\_18"*/>  </group>  </menu>  </item>  <!-- 定义一个普通菜单项 -->  <item android:id=*"@+id/plain\_item"*  android:title=*"@string/plain\_item"*>  </item>  <item android:title=*"@string/font\_color"*  android:icon=*"@drawable/color"*>  <menu>  <!-- 定义一组允许复选的菜单项 -->  <group>  <!-- 定义3个菜单项 -->  <item  android:id=*"@+id/red\_font"*  android:title=*"@string/red\_title"*/>  <item  android:id=*"@+id/green\_font"*  android:title=*"@string/green\_title"*/>  <item  android:id=*"@+id/blue\_font"*  android:title=*"@string/blue\_title"*/>  </group>  </menu>  </item>  </menu> |
| // 创建上下文菜单时触发该方法  @Override  **public** **void** onCreateContextMenu(ContextMenu menu, View source, ContextMenu.ContextMenuInfo menuInfo) {  MenuInflater inflator = **new** MenuInflater(**this**);  // 状态R.menu.context对应的菜单，并添加到menu中  inflator.inflate(R.menu.***context***, menu);  menu.setHeaderIcon(R.drawable.***tools***);  menu.setHeaderTitle("请选择背景色");  }  context.xml  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <menu xmlns:android=*"http://schemas.android.com/apk/res/android"*>  <!-- 定义一组单选菜单项目 -->  <group android:checkableBehavior=*"single"*>  <!-- 定义3个菜单项 -->  <item  android:id=*"@+id/red"*  android:title=*"@string/red\_title"*  android:alphabeticShortcut=*"r"*/>  <item  android:id=*"@+id/green"*  android:title=*"@string/green\_title"*  android:alphabeticShortcut=*"g"*/>  <item  android:id=*"@+id/blue"*  android:title=*"@string/blue\_title"*  android:alphabeticShortcut=*"b"*/>  </group>  </menu> |

## 菜单关联Activity

|  |
| --- |
| @Override  **public** **boolean** onCreateOptionsMenu(Menu menu) {  // -------------向menu中添加子菜单-------------  SubMenu prog = menu.addSubMenu("启动程序");    prog.setHeaderIcon(R.drawable.***tools***);// 设置菜单头的图标  prog.setHeaderTitle("选择您要启动的程序");// 设置菜单头的标题  MenuItem item = prog.add("查看经典Java EE");// 添加菜单项  item.setIntent(**new** Intent(**this**, OtherActivity.**class**));// 为菜单项设置关联的Activity  **return** **super**.onCreateOptionsMenu(menu);  } |

## 上下文菜单ContextMenu

|  |
| --- |
| // 上下文菜单中菜单项被单击时触发该方法。  @Override  **public** **boolean** onContextItemSelected(MenuItem mi) {  mi.setChecked(**true**); // ①  **switch** (mi.getItemId()) {  **case** R.id.***red*** :  mi.setChecked(**true**);  txt.setBackgroundColor(Color.***RED***);  **break**;  **case** R.id.***green*** :  mi.setChecked(**true**);  txt.setBackgroundColor(Color.***GREEN***);  **break**;  **case** R.id.***blue*** :  mi.setChecked(**true**);  txt.setBackgroundColor(Color.***BLUE***);  **break**;  }  **return** **true**;  } |
| // 选项菜单的菜单项被单击后的回调方法  @Override  **public** **boolean** onOptionsItemSelected(MenuItem mi) {  **if** (mi.isCheckable()) {  mi.setChecked(**true**); // ②  }  // 判断单击的是哪个菜单项，并针对性的作出响应。  **switch** (mi.getItemId()) {  **case** R.id.***font\_10*** :  txt.setTextSize(10 \* 2);  **break**;  **case** R.id.***font\_12*** :  txt.setTextSize(12 \* 2);  **break**;  **case** R.id.***font\_14*** :  txt.setTextSize(14 \* 2);  **break**;  **case** R.id.***font\_16*** :  txt.setTextSize(16 \* 2);  **break**;  **case** R.id.***font\_18*** :  txt.setTextSize(18 \* 2);  **break**;  **case** R.id.***red\_font*** :  txt.setTextColor(Color.***RED***);  mi.setChecked(**true**);  **break**;  **case** R.id.***green\_font*** :  txt.setTextColor(Color.***GREEN***);  mi.setChecked(**true**);  **break**;  **case** R.id.***blue\_font*** :  txt.setTextColor(Color.***BLUE***);  mi.setChecked(**true**);  **break**;  **case** R.id.***plain\_item*** :  Toast toast = Toast.*makeText*(MenuResTest.**this**, "您单击了普通菜单项", Toast.***LENGTH\_SHORT***);  toast.show();  **break**;  }  **return** **true**;  } |
| // 为每个菜单定义一个标识  **final** **int** MENU1 = 0x111;  **final** **int** MENU2 = 0x112;  **final** **int** MENU3 = 0x113;  **private** TextView txt;    @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);    txt = (TextView) findViewById(R.id.***txt***);  registerForContextMenu(txt); // ① // 为文本框注册上下文菜单, 长按文本框弹出上下文菜单  }  // 创建上下文菜单时触发该方法  @Override  **public** **void** onCreateContextMenu(ContextMenu menu, View source, ContextMenu.ContextMenuInfo menuInfo) {  menu.add(0, MENU1, 0, "红色");  menu.add(0, MENU2, 0, "绿色");  menu.add(0, MENU3, 0, "蓝色");  menu.setGroupCheckable(0, **true**, **true**);// 将三个菜单项设为单选菜单项  menu.setHeaderIcon(R.drawable.***tools***);// 设置上下文菜单的标题、图标  menu.setHeaderTitle("选择背景色");  }  // 上下菜单的菜单项被单击时触发该方法。  @Override  **public** **boolean** onContextItemSelected(MenuItem mi) {  **switch** (mi.getItemId()) {  **case** MENU1 :  mi.setChecked(**true**);  txt.setBackgroundColor(Color.***RED***);  **break**;  **case** MENU2 :  mi.setChecked(**true**);  txt.setBackgroundColor(Color.***GREEN***);  **break**;  **case** MENU3 :  mi.setChecked(**true**);  txt.setBackgroundColor(Color.***BLUE***);  **break**;  }  **return** **true**;  } |

## 弹出式菜单PopupMenu

|  |
| --- |
| <Button  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_gravity=*"center"*  android:onClick=*"onPopupButtonClick"*  android:text=*"单击我"* /> |
| **public** **void** onPopupButtonClick(View button) {  // 创建PopupMenu对象  popup = **new** PopupMenu(**this**, button);  // 将R.menu.popup\_menu菜单资源加载到popup菜单中  getMenuInflater().inflate(R.menu.***popup\_menu***, popup.getMenu());  // 为popup菜单的菜单项单击事件绑定事件监听器  popup.setOnMenuItemClickListener(**new** PopupMenu.OnMenuItemClickListener() {  @Override  **public** **boolean** onMenuItemClick(MenuItem item) {  **switch** (item.getItemId()) {  **case** R.id.***exit*** :  // 隐藏该对话框  popup.dismiss();  **break**;  **default** :  // 使用Toast显示用户点击的菜单项  Toast.*makeText*(PopupMenuTest.**this**, "您单击了【" + item.getTitle() + "】菜单项", Toast.***LENGTH\_SHORT***).show();  }  **return** **true**;  }  });  popup.show();  } |

# 活动条

## ActionBar

|  |
| --- |
| // 只有当应用主题没有关闭ActionBar时，该代码才能返回ActionBar  ActionBar actionBar = getActionBar();    // 显示ActionBar  actionBar.show();  // 隐藏ActionBar  actionBar.hide(); |
| @Override  // 选项菜单的菜单项被单击后的回调方法  **public** **boolean** onOptionsItemSelected(MenuItem mi) {  **if** (mi.isCheckable()) {  mi.setChecked(**true**);  }  // 判断单击的是哪个菜单项，并针对性的作出响应。  **switch** (mi.getItemId()) {  **case** android.R.id.***home*** : //内置的ID ?  // 创建启动FirstActivity的Intent  Intent intent = **new** Intent(**this**, FirstActivity.**class**);  // 添加额外的Flag，将Activity栈中处于FirstActivity之上的Activity弹出  intent.addFlags(Intent.***FLAG\_ACTIVITY\_CLEAR\_TOP***);  // 启动intent对应的Activity  startActivity(intent);  **break**;  **case** R.id.***font\_10*** :  txt.setTextSize(10 \* 2);  **break**;  **case** R.id.***font\_12*** :  txt.setTextSize(12 \* 2);  **break**;  **.....**  **case** R.id.***plain\_item*** :  Toast toast = Toast.*makeText*(ActionHomeTest.**this**, "您单击了普通菜单项", Toast.***LENGTH\_SHORT***);  toast.show();  **break**;  }  **return** **true**;  } |

## ActionBar显示选项菜单

|  |
| --- |
| @Override  **public** **boolean** onCreateOptionsMenu(Menu menu) {  MenuInflater inflator = **new** MenuInflater(**this**);  // 状态R.menu.context对应的菜单，并添加到menu中  inflator.inflate(R.menu.***my\_menu***, menu);  **return** **super**.onCreateOptionsMenu(menu);  }  @Override  // 选项菜单的菜单项被单击后的回调方法  **public** **boolean** onOptionsItemSelected(MenuItem mi) {  **if** (mi.isCheckable()) {  mi.setChecked(**true**); // ②  }  // 判断单击的是哪个菜单项，并针对性的作出响应。  **switch** (mi.getItemId()) {  **case** R.id.***font\_10*** :  txt.setTextSize(10 \* 2);  **break**;  **case** R.id.***font\_12*** :  txt.setTextSize(12 \* 2);  **break**;  **case** R.id.***font\_14*** :  txt.setTextSize(14 \* 2);  **break**;  **case** R.id.***font\_16*** :  txt.setTextSize(16 \* 2);  **break**;  **case** R.id.***font\_18*** :  txt.setTextSize(18 \* 2);  **break**;  **case** R.id.***red\_font*** :  txt.setTextColor(Color.***RED***);  mi.setChecked(**true**);  **break**;  **case** R.id.***green\_font*** :  txt.setTextColor(Color.***GREEN***);  mi.setChecked(**true**);  **break**;  **case** R.id.***blue\_font*** :  txt.setTextColor(Color.***BLUE***);  mi.setChecked(**true**);  **break**;  **case** R.id.***plain\_item*** :  Toast toast = Toast.*makeText*(ActionItemTest.**this**, "您单击了普通菜单项", Toast.***LENGTH\_SHORT***);  toast.show();  **break**;  }  **return** **true**;  } |
| myMenu.xml  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <menu xmlns:android=*"http://schemas.android.com/apk/res/android"*>  <item android:title=*"@string/font\_size"*  android:showAsAction=*"always|withText"*  android:icon=*"@drawable/font"*>  <menu>  <!-- 定义一组单选菜单项 -->  <group android:checkableBehavior=*"single"*>  <!-- 定义多个菜单项 -->  <item  android:id=*"@+id/font\_10"*  android:title=*"@string/font\_10"*/>  <item  android:id=*"@+id/font\_12"*  android:title=*"@string/font\_12"*/>  <item  android:id=*"@+id/font\_14"*  android:title=*"@string/font\_14"*/>  <item  android:id=*"@+id/font\_16"*  android:title=*"@string/font\_16"*/>  <item  android:id=*"@+id/font\_18"*  android:title=*"@string/font\_18"*/>  </group>  </menu>  </item>  <!-- 定义一个普通菜单项 -->  <item android:id=*"@+id/plain\_item"*  android:showAsAction=*"always|withText"*  android:title=*"@string/plain\_item"*>  </item>  <item android:title=*"@string/font\_color"*  android:showAsAction=*"always"*  android:icon=*"@drawable/color"*>  <menu>  <!-- 定义一组允许复选的菜单项 -->  <group>  <!-- 定义3个菜单项 -->  <item  android:id=*"@+id/red\_font"*  android:title=*"@string/red\_title"*/>  <item  android:id=*"@+id/green\_font"*  android:title=*"@string/green\_title"*/>  <item  android:id=*"@+id/blue\_font"*  android:title=*"@string/blue\_title"*/>  </group>  </menu>  </item>  </menu> |
| colors.xml  <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <resources>  <color name=*"red"*>#ff0000</color>  <color name=*"green"*>#00ff00</color>  <color name=*"blue"*>#0000ff</color>  </resources>  string.xml  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <resources>  <string name=*"hello"*>Hello World, ActionItemTest!</string>  <string name=*"app\_name"*>添加Action Item</string>  <string name=*"red\_title"*>红色</string>  <string name=*"blue\_title"*>蓝色</string>  <string name=*"green\_title"*>绿色</string>  <string name=*"font\_size"*>字体大小</string>  <string name=*"font\_color"*>字体颜色</string>  <string name=*"plain\_item"*>普通菜单项</string>  <string name=*"font\_10"*>10号字体</string>  <string name=*"font\_12"*>12号字体</string>  <string name=*"font\_14"*>14号字体</string>  <string name=*"font\_16"*>16号字体</string>  <string name=*"font\_18"*>18号字体</string>  </resources> |

## ActionBar实现Tab标签页导航(点击切换)

|  |
| --- |
| **public** **class** ActionBar\_TabNav **extends** Activity **implements** ActionBar.~~TabListener~~ {  **private** **static** **final** String ***SELECTED\_ITEM*** = "selected\_item";  @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);  **final** ActionBar actionBar = getActionBar();  // 设置ActionBar的导航方式：Tab导航  actionBar.~~setNavigationMode~~(ActionBar.~~NAVIGATION\_MODE\_TABS~~);  // 依次添加3个Tab页，并为3个Tab标签添加事件监听器  actionBar.~~addTab~~(actionBar.~~newTab~~().setText("第一页").setTabListener(**this**));  actionBar.~~addTab~~(actionBar.~~newTab~~().setText("第二页").setTabListener(**this**));  actionBar.~~addTab~~(actionBar.~~newTab~~().setText("第三页").setTabListener(**this**));  }  @Override  **public** **void** onRestoreInstanceState(Bundle savedInstanceState) {  **if** (savedInstanceState.containsKey(***SELECTED\_ITEM***)) {  // 选中前面保存的索引对应的Fragment页  getActionBar().~~setSelectedNavigationItem~~(savedInstanceState.getInt(***SELECTED\_ITEM***));  }  }  @Override  **public** **void** onSaveInstanceState(Bundle outState) {  // 将当前选中的Fragment页的索引保存到Bundle中  outState.putInt(***SELECTED\_ITEM***, getActionBar().~~getSelectedNavigationIndex~~());  }  @Override  **public** **void** onTabUnselected(ActionBar.~~Tab~~ tab, FragmentTransaction fragmentTransaction) {  }    // 当指定Tab被选中时激发该方法  @Override  **public** **void** onTabSelected(ActionBar.~~Tab~~ tab, FragmentTransaction fragmentTransaction) {  // 创建一个新的Fragment对象  Fragment fragment = **new** DummyFragment();  // 创建一个Bundle对象，用于向Fragment传入参数  Bundle args = **new** Bundle();  args.putInt(DummyFragment.***ARG\_SECTION\_NUMBER***, tab.getPosition() + 1);  // 向fragment传入参数  fragment.setArguments(args);  // 获取FragmentTransaction对象  FragmentTransaction ft = getFragmentManager().beginTransaction();  // 使用fragment代替该Activity中的container组件  ft.replace(R.id.***container***, fragment);  // 提交事务  ft.commit();  }  @Override  **public** **void** onTabReselected(ActionBar.~~Tab~~ tab, FragmentTransaction fragmentTransaction) {  }  } |
| **public** **class** DummyFragment **extends** Fragment {  **public** **static** **final** String ***ARG\_SECTION\_NUMBER*** = "section\_number";  // 该方法的返回值就是该Fragment显示的View组件  @Override  **public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  TextView textView = **new** TextView(getActivity());  textView.setGravity(Gravity.***CENTER\_HORIZONTAL***);  // 获取创建该Fragment时传入的参数Bundle  Bundle args = getArguments();  // 设置TextView显示的文本  textView.setText(args.getInt(***ARG\_SECTION\_NUMBER***) + "");  textView.setTextSize(30);  // 返回该TextView  **return** textView;  }  } |

## ActionBar实现Tab标签页导航(点击左右滑动切换)

|  |
| --- |
| **public** **class** ActionBar\_TabSwipeNav **extends** FragmentActivity **implements** ActionBar.TabListener {  ViewPager viewPager;  ActionBar actionBar;  @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);  // 获取ActionBar对象  actionBar = getActionBar();  // 获取ViewPager  viewPager = (ViewPager) findViewById(R.id.***pager***);  // 创建一个FragmentPagerAdapter对象，该对象负责为ViewPager提供多个Fragment  FragmentPagerAdapter pagerAdapter = **new** FragmentPagerAdapter(getSupportFragmentManager()) {  // 获取第position位置的Fragment  @Override  **public** Fragment getItem(**int** position) {  Fragment fragment = **new** DummyFragment();  Bundle args = **new** Bundle();  args.putInt(DummyFragment.***ARG\_SECTION\_NUMBER***, position + 1);  fragment.setArguments(args);  **return** fragment;  }  // 该方法的返回值i表明该Adapter总共包括多少个Fragment  @Override  **public** **int** getCount() {  **return** 3;  }  // 该方法的返回值决定每个Fragment的标题  @Override  **public** CharSequence getPageTitle(**int** position) {  **switch** (position) {  **case** 0 :  **return** "第一页";  **case** 1 :  **return** "第二页";  **case** 2 :  **return** "第三页";  }  **return** **null**;  }  };  // 设置ActionBar使用Tab导航方式  actionBar.setNavigationMode(ActionBar.***NAVIGATION\_MODE\_TABS***);  // 遍历pagerAdapter对象所包含的全部Fragment。  // 每个Fragment对应创建一个Tab标签  **for** (**int** i = 0; i < pagerAdapter.getCount(); i++) {  actionBar.addTab(actionBar.newTab().setText(pagerAdapter.getPageTitle(i)).setTabListener(**this**));  }  // 为ViewPager组件设置FragmentPagerAdapter  viewPager.setAdapter(pagerAdapter); // ①  // 为ViewPager组件绑定事件监听器  viewPager.setOnPageChangeListener(**new** ViewPager.SimpleOnPageChangeListener() {  // 当ViewPager显示的Fragment发生改变时激发该方法  @Override  **public** **void** onPageSelected(**int** position) {  actionBar.setSelectedNavigationItem(position);  }  });  }  @Override  **public** **void** onTabUnselected(ActionBar.Tab tab, FragmentTransaction fragmentTransaction) {  }  // 当指定Tab被选中时激发该方法  @Override  **public** **void** onTabSelected(ActionBar.Tab tab, FragmentTransaction fragmentTransaction) {  viewPager.setCurrentItem(tab.getPosition()); // ②  }  @Override  **public** **void** onTabReselected(ActionBar.Tab tab, FragmentTransaction fragmentTransaction) {  }  } |
| **public** **class** DummyFragment **extends** Fragment {  **public** **static** **final** String ***ARG\_SECTION\_NUMBER*** = "section\_number";  // 该方法的返回值就是该Fragment显示的View组件  @Override  **public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  TextView textView = **new** TextView(getActivity());  textView.setGravity(Gravity.***CENTER\_HORIZONTAL***);  // 获取创建该Fragment时传入的参数Bundle  Bundle args = getArguments();  // 设置TextView显示的文本  textView.setText(args.getInt(***ARG\_SECTION\_NUMBER***) + "");  textView.setTextSize(30);  // 返回该TextView  **return** textView;  }  } |

## ActionBar实现下拉式导航

|  |
| --- |
| **public** **class** ActionBar\_DropDownNav **extends** Activity **implements** ActionBar.~~OnNavigationListener~~ {  **private** **static** **final** String ***SELECTED\_ITEM*** = "selected\_item";  @Override  **public** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.***main***);    **final** ActionBar actionBar = getActionBar();  // 设置ActionBar是否显示标题  actionBar.setDisplayShowTitleEnabled(**true**);  // 设置导航模式,使用List导航  actionBar.~~setNavigationMode~~(ActionBar.~~NAVIGATION\_MODE\_LIST~~);  // 为actionBar安装ArrayAdapter  actionBar.~~setListNavigationCallbacks~~(**new** ArrayAdapter<String>(ActionBar\_DropDownNav.**this**, android.R.layout.***simple\_list\_item\_1***, android.R.id.***text1***, **new** String[]{"第一页", "第二页", "第三页"}), **this**);  }  @Override  **public** **void** onRestoreInstanceState(Bundle savedInstanceState) {  **if** (savedInstanceState.containsKey(***SELECTED\_ITEM***)) {  // 选中前面保存的索引对应的Fragment页  getActionBar().~~setSelectedNavigationItem~~(savedInstanceState.getInt(***SELECTED\_ITEM***));  }  }  @Override  **public** **void** onSaveInstanceState(Bundle outState) {  // 将当前选中的Fragment页的索引保存到Bundle中  outState.putInt(***SELECTED\_ITEM***, getActionBar().~~getSelectedNavigationIndex~~());  }    // 当导航项被选中时激发该方法  @Override  **public** **boolean** onNavigationItemSelected(**int** position, **long** id) {  // 创建一个新的Fragment对象  Fragment fragment = **new** DummyFragment();  // 创建一个Bundle对象，用于向Fragment传入参数  Bundle args = **new** Bundle();  args.putInt(DummyFragment.***ARG\_SECTION\_NUMBER***, position + 1);  // 向fragment传入参数  fragment.setArguments(args);  // 获取FragmentTransaction对象  FragmentTransaction ft = getFragmentManager().beginTransaction();  // 使用fragment代替该Activity中的container组件  ft.replace(R.id.***container***, fragment);  // 提交事务  ft.commit();  **return** **true**;  }  } |
| **public** **class** DummyFragment **extends** Fragment {  **public** **static** **final** String ***ARG\_SECTION\_NUMBER*** = "section\_number";  // 该方法的返回值就是该Fragment显示的View组件  @Override  **public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  TextView textView = **new** TextView(getActivity());  textView.setGravity(Gravity.***CENTER\_HORIZONTAL***);  // 获取创建该Fragment时传入的参数Bundle  Bundle args = getArguments();  // 设置TextView显示的文本  textView.setText(args.getInt(***ARG\_SECTION\_NUMBER***) + "");  textView.setTextSize(30);  // 返回该TextView  **return** textView;  }  } |

## ActionView

|  |
| --- |
| @Override  **public** **boolean** onCreateOptionsMenu(Menu menu) {  getMenuInflater().inflate(R.menu.***menu***, menu);  **return** **true**;  } |
| menu.xml  <?xml version=*"1.0"* encoding=*"utf-8"* ?>  <menu xmlns:android=*"http://schemas.android.com/apk/res/android"*>  <item  android:id=*"@+id/search"*  android:title=*"@string/menu\_settings"*  android:orderInCategory=*"100"*  android:showAsAction=*"always"*  android:actionViewClass=*"android.widget.SearchView"*/>  <item  android:id=*"@+id/progress"*  android:title=*"@string/menu\_settings"*  android:orderInCategory=*"100"*  android:showAsAction=*"always"*  android:actionLayout=*"@layout/clock"*/>  </menu>  clock.xml  <?xml version=*"1.0"* encoding=*"utf-8"* ?>  <AnalogClock  xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  /> |
|  |