布局与控件

# 1 布局

为了更好地管理Android应用的用户界面里的各组件，Android提供了布局管理器。而已管理器可以根据运行平台来调整组件的大小。与Swing不同的是，Android布局管理器本身就是一个UI组件，所有的布局管理器都是ViewGroup的子类。

实际上，我们完全可以用一个布局管理器嵌套到其他布局管理器， 因为布局管理器也继承了View，也可以作为普通UI组件使用。

## 1.1 线性布局

线性布局由LinearLayout类来代表，线性布局有点像AWT中的FlowLayout.它们都会将窗口里的组件一个挨着一个地排列起来。LinearLayout不仅可以控制各组件横向排列(通过设置android:orientation属性控制)，也可控制各组件纵向排列。

线性布局与AWT中FlowLayout的最大区别在于:Android线性布局不会换行：当组件一个挨着一个排列到头之后，剩下的组件将不会被显示出来。在AWT中FlowLayout则会另起一行来排列多出来的组件。

表格 1LinearLayout 的常用XML属性及相关方法

|  |  |  |
| --- | --- | --- |
| XML属性 | 相关方法 | 说明 |
| android:gravity | setGravity(int) | 设置布局管理器内组件的对齐方式。该属性支持  top  bottom  left  right  center\_vertical  fill\_vertical  center\_horizontal  fill\_horizontal  center  fill  clip\_vertical  clip\_horizontal  这些属性值。也可以同时指定多种对齐方式的组合，如  left|center\_vertical 代表出现在屏幕左边，而且垂直居中。 |
| android:orientation | setOrientation(int) | 设置布局管理器内组件的排列方式，可以设置为horizontal(水平排列),vertical(垂直排列-默认值) 之一。 |

## 1.2 表格布局

表格布局由TableLayout所代表，表格布局采用行，列的开工来管理UI组件，TableLayout并不需要明确地声明包含多少行，多少列，而是通过添加TableRow、其他组件来控制表格的行数和列数。

每次向TableLayout中添加一个TableRow，该TableRow就是一个表格行，TableROw也是窗口，因此它也可以不断地添加其他组件，每添加一个子组件该表格就增加一列。

如果直接向TableLayout中添加组件，那么这个组件将直接占用一行。

在表格布局中，列的宽度由该列中最宽的那个单元格决定，整个表格布局的宽度则取决于父窗口的宽度（默认总是占满父窗口本身）。

表格而已管理器中，可以为单元格设置如下三种行为方式：

* Shrinkable ： 如果某个列被设为Shrinkable，那么该列的所有单元格的宽度可以被收缩，以保证该表格能适应父窗口的宽度
* Stretchable ： 如果某个列被设置为Stretchable,那么该列的所有单元格的宽度可以被拉伸，以保证组件能完全填满表格空余空间。
* Collapsed ： 如果某个列被设为Collapsed,那么该列的所有单元格会被隐藏。

TableLayout 继承了LinearLayout，因此它完全可以支持LinearLayout所支持的全部XML属性，除此之外，还支持以下XML属性：

表格 2 TableLayout的常用XML属性及相关方法

|  |  |  |
| --- | --- | --- |
| XML属性 | 相关方法 | 说明 |
| android:collapseCOlumns | setColumnCollapsed(int ,Boolean) | 设置需要被隐藏的列序号，多个列序号之间用逗号隔开 |
| android:shrinkColumns | setShrinkAllColumns(Boolean) | 设置允许被收缩的列的序号 |
| android:stretchColumns | setStretchAllColumns(Boolean) | 设置允许被拉伸的列的列序号 |

## 1.3 帧布局

帧布局由FrameLayout所代表，FrameLayout直接继承了ViewGroup组件。

帧布局窗口为每个加入其中的组件创建一个空白的区域（称为一帧），所有每一个子组件占据一帧，这些帧会根据gravity属性执行自动对齐，也就是说，帧布局的效果有点类似于AWT的CardLayout，都是把组件一个个叠加在一起，与CardLayout的区别在于，CardLayout可以将下面的Card移上来，但FrameLayout则没有提供相应的方法。

|  |  |  |
| --- | --- | --- |
| XML属性 | 相关方法 | 说明 |
| android:foreground | setForeground(Drawable) | 设置该帧布局窗口的前景图像 |
| android:foregroundGravity | setForegroundGravity(int) | 定义绘制前景图像的Gravity属性 |

## 1.4 相对布局

相对布局由RelativeLayout代表，相对布局窗口内子组件的位置总是相对兄弟组件，父窗口来决定的，因此这种布局方式被称为相对布局。

如果A组件位置由B组件的位置来决定，那么要先定义B组件，再定义A组件。

|  |  |  |
| --- | --- | --- |
| XML属性 | 相关方法 | 说明 |
| android:gravity | setGravity(int) | 设置该布局窗口内部各子组件的对齐方式 |
| android:ignoreGravity | setIngoreGravity(int) | 设置哪个组件不受gravity组件的影响 |

为了控制该布局窗口中各子组件的布局分布，RelativeLayout提供了一个内部类：RelativeLayout.LayoutParams，该类提供了大量的XML属性来控制RelativeLayout布局窗口中组件而已分布。

RelativeLayout.LayoutParams里只能设置为true,false的XML如下所示：

表格 3RelativeLayout.LayoutParams 只能设置为boolean值的属性

|  |  |
| --- | --- |
| 属性 | 说明 |
| android.layout\_centerHorizontal | 控制该子组件是否位于布局窗口的水平居中位置 |
| android.layout\_centerVertical | 控制该子组件是否位于布局窗口的垂直居中位置 |
| android.layout\_centerInParent | 控制该子组件是否位于布局窗口的中央位置 |
| android.layout\_alignParentBottom | 控制该子组件是否与布局容器底端对齐 |
| android.layout\_alignParentLeft | 是否与布局容器左边对齐 |
| android.layout\_alignParentRight | 是否与布局容器右边对齐 |
| android.layout\_alignParentTop | 是否与布局容器顶端对齐 |

表格 4RelativeLayout.LayoutParams里只能设置 为其他UI组件ID的属性

|  |  |
| --- | --- |
| 属性 | 说明 |
| android:layout\_toRightOf | 位于给出ID组件的右侧 |
| android:layout\_toLeftOf | 位于给出ID组件的左侧 |
| android:layout\_above | 位于给出ID组件的上方 |
| android:layout\_below | 下方 |
| android:layout\_alignTop | 与给出ID组件上边界对齐 |
| android:layout\_alignBottom | 与给出ID组件下边界对齐 |
| android:layout\_alignLeft | 与给出ID组件左边界对齐 |
| android:layout\_alignRight | 右边界对齐 |

## 1.5 绝对布局

绝对布局由AbsoluteLayout代表。绝对布局就像Java AWT编程中的空布局，就是Android不提供任何布局控制，而是由开发人员自己通过X,Y坐标来控制组件的位置；

当使用AbsoluteLayout作为布局窗口时，布局窗口不再管理子组件的位置，大小，这些都需要开发人员自己控制。

大部分时候，使用绝对布局都不好，因为运行Android应用的手机千差万别，因此屏幕大小，分辨率都可能存在圈套差异，使用绝对布局会很难兼顾不同屏幕大小，分辨率的问题。

使用绝对布局时，每个子组件都可指定如下两个XML属性：

* layout\_x : 组件X坐标
* layout\_y : 组件Y坐标。