Android 和 UI 设计以及 LinearLayout

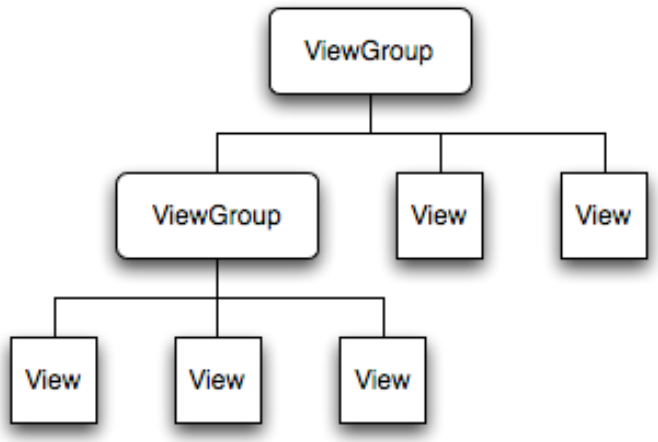
知识解析

UI(User Interface)是介于用户与硬件而设计彼此之间互动沟通相关软件，目的在用户能够方便有效率地去操作硬件以达成双向之互动，完成希望借助硬件完成的工作。用户接口定义广泛，包含了人机互动与图形用户接口，凡参与人类与机械的信息交流的领域都存在着用户接口。

Android 应用中编写 UI 的 2 种方式：

- 与主程序混合写在一起，一般不使用这种方式来定义 UI。
- 写在 XML 中：通过在 res/layout 中定义对应的 xml 资源文件来定义 UI。建议使用这种方式

Android UI 结构



用于显示数据、图片或者其他信息的组件，叫做“View”。

ViewGroup 是一种 View 容器，本身也是一种 View，但是可以包含 View 及其他

ViewGroup 组件的 View。例如： LinearLayout。通常会先建构出 ViewGroup 容器组件，像是 LinearLayout 对象实体后，接着调用 addView(View 对象实例,LinearLayout.Params 对象实例)的方法，将 View 对象实体，以指定的参数 LinearLayout.Params 对象实体加进来组合。

上图说明了在 Android Appliation 中的 UI 架构，但与 API 中的面向对象层级架构并不相同。因此了解到整个 Application 架构就是以 ViewGroup 组件为一个大容器，可以放置 View 及 ViewGroup。但是从面向对象的观点来看，ViewGroup 继承自 View，所以 ViewGroup 也是一个 View，只是 ViewGroup 有容器的特色。

组件布局——线性布局

LinearLayout 是线性布局控件，它包含的子控件将以横向或竖向的方式排列，按照相对位置来排列所有的 widgets 或者其他的 containers,超 过边界时，某些控件将缺失或消失。因此一个垂直列表的每一行只会有一个 widget 或者是 container，而不管他们有多宽，而一个水平列表将会只有 一个行高（高度为最高子控件的高度加上边框高度）。LinearLayout 保持其所包含的 widget 或者是 container 之间的间隔以及互相对齐 （相对一个控件的右对齐、中间对齐或者左对齐）。

xml 属性

android:baselineAligned：是否允许用户调整它内容的基线。

android:baselineAlignedChildIndex：当一个线性布局与另一个布局是按基线对齐的一部分，它可以指定其内容的基线对齐方式。

android:gravity：指定如何在该对象中放置此对象的内容（x/y 坐标值）。

android:orientation：设置它内容的对其方向（横向/竖向）。

gravity 这个英文单词是重心的意思，在这里就表示停靠位置的意思。

android:layout_gravity 和 android:gravity 的区别

从名字上可以看到，android:gravity 是对元素本身说的，元素本身的文本显示在什么地方靠着换个属性设置，不过不设置默认是在左侧的。

android:layout_gravity 是相对与它的父元素说的，说明元素显示在父元素的什么位置。

比如说 button：android:layout_gravity 表示按钮在界面上的位置。 android:gravity 表示 button 上的字在 button 上的位置。

可选值

这两个属性可选的值有：top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical。

而且这些属性是可以多选的，用“|”分开。

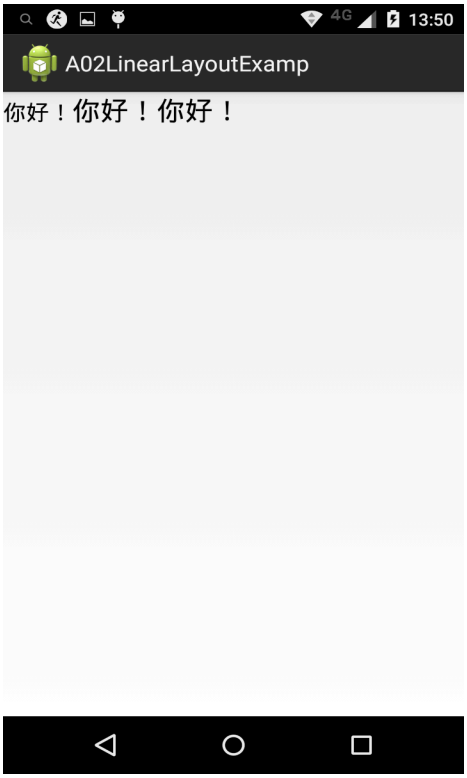
默认这个的值是：Gravity.LEFT

LinearLayout 还支持为其包含的 widget 或者是 container 指定填充权值。好处就是允许其包含的 widget 或者是 container 可以填充屏幕上的剩余空 间。这也避免了在一个大屏幕中，一串 widgets 或者是 containers 挤成一堆的情况，而是允许他们放大填充空白。剩余的空间会按这些 widgets 或者是 containers 指定的权值比例分配屏幕。默认的 weight 值为 0，表示按照 widgets 或者是 containers 实际大小来显示，若高于 0 的值，则将 Container 剩余可用空间分割，分割大小具体取决于 每一个 widget 或者是 container 的 layout_weight 及该权值在所有 widgets 或者是 containers 中的比例。例如，如果 有三个文本框，其中两个指定的权值为 1，那么，这两个文本框将等比例地放大，并填满剩余的空间，而第三个文本框不会放大，按实际大小来显示。如果前两个文 本框的取值一个为 2，一个为 1，显示第三个文本框后剩余的空间的 2/3 给权值为 2 的，1/3 大小给权值为 1 的。也就是权值越大，重要度越大。

如果 LinearLayout 包含子 LinearLayout，子 LinearLayout 之间的权值越大的，重要度则越小。如果有 LinearLayout A 包含 LinearLayout C,D，C 的权值为 2，D 的权值为 1，则屏幕

的 2/3 空间分给权值为 1 的 D，1/3 分给权值为 2 的 C。在 LinearLayout 嵌套的情况下，子 LinearLayout 必须要设置权值，否则默认的情况是未设置权值的子 LinearLayout 占据整个屏幕。

功能演示



实战操作

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:textAppearance="?android:attr/textAppearanceMedium"
/>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:textAppearance="?android:attr/textAppearanceLarge" />
<TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:textAppearance="?android:attr/textAppearanceLarge" />
</LinearLayout>
```

职业素质

线性布局有 LinearLayout 类来代表，线性布局有点像 Swing 编程里的 Box，他们都会将容器里的组件一个挨着一个地排列起来。LinearLayout 可以控制各组件的横向排列，也可以控制各组件纵向排列。