深入理解 Android 之界面构造

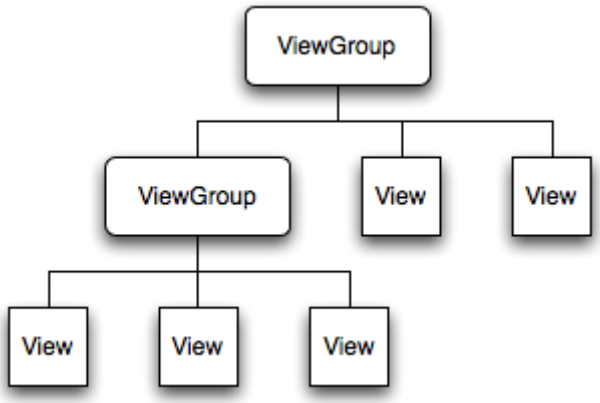
UI 界面，对于每个应用而言，是它与用户进行交互的门脸。好的门脸，不只是是要亮丽可人，最好还能秀色可餐过目不忘，甚至还应该有涵养有气质，彬彬有理温柔耐心。

对于开发者来说，锻造这样的面容，不但需要高超的技艺，也需要有称手的工具和对得起党的料子。俗话说，朽木不可雕也，芙蓉不是一日炼成的，不是什么平台都能叫特能书。有套好用的 UI 框架，对于开发者而言，真有如沙漠中的甘露，而要是撞见了杯具的 UI 套件，整个界面开发就有如梦魇了。

Android 的 UI 框架，最核心的，是资源和 Layout 体系，然后，通过完善的控件库，简明的接口设计，进一步帮助开发者，能够最快的搭建自己需要界面（听到这里，Symbian 同学开始钻土。..）。

UI 控件

做 UI，有时候就像搭积木，在 Android 中，这个最原子的积木块，就是 View。所有其他的 UI 元素，都是派生于此类的子孙类们。



又从 SDK 中偷来张图，用来描述 Android 的 UI 控件结构，在每一个 window 下，这都是一个标准而完整的树结构。View 有一个子类 ViewGroup，它相当于一个容器类或者是复合控件，所有派生与 ViewGroup 的子类在这颗 UI 树中都可以承担着父节点的职责，而另一些绕过 ViewGroup 从 View 直通下来的，就只能蜷局在叶节点的范畴内了。

之所有说这是一个很标准的控件树，是因为父控件对子控件有绝对的掌控权，每个子控件的占地面积和位置，都是基于父控件来分配的，它能够接受和处理的事件，也是父控件派发下去的。这样的结构，被很多平台和框架广泛的认可，和传统的 win 开发和杯具的 Symbian 相比，虽然因为事件传播途径变长了，很多操作的效率变低了，但整个结构更有层次性，每个控件只需要多其父控件负责指挥子控件就好，职责明确，逻辑简单，利于开发和设计。

谈及任何平台的控件，都有一些不可避免的主题，比如，每个控件如何标识，如何设定大小和位置，如何接受和处理事件，如何绘制，诸如此类。

标识

在 Android 中，你可以为每个控件选择设定一个 id，这个 id 的全局的唯一性不需要保证，但在某个局部的范围内具有可识别性，这样就可以通过这个 id 找到这个控件（如果不需要查找，就别设置了。..）。

但是，在父控件中逐级的 find 比较，找到 id 匹配的控件，然后再做转型，是一个比较重量的操作，于是 Android 又为控件憋出另一个属性，tag。它接受任意 object 类型的数据，你可以把和这个控件对象相关的内容堆在里面。比如，在 list 中，我们常常将和每个 list item 相关的所有控件元素封装成一个 object，扔到 tag 中，就不需要每次都去比较 id 进行寻找，更加高效快捷。

尺寸

在 Android 中，控件最重要的大小属性，就是 width/height，开发者可以明确的指明控件的大小，可以设定成为 fill_parent 和 wrap_content，这样的概念性的大小。丈量并设定控件的位置，是通过两步来进行的。

第一步是 measure。它传入此控件的 width/height 信息，控件会根据自己的参数，计算出真实需要的 width/height，然后调用 setMeasuredDimension 方法，缓存成成员变量，留作后用。

在计算出大小之后，会进行另一个步骤，layout。在这个过程中，父控件会计算其上各个子控件的位置，从而完成整个大小和位置的确定流程。整个 measure 和 layout 的流程，都是自上到下，从树顶往叶子来推进的。

当开发人员需要自定义控件的时候，可能需要关注这些内容，通过重载 onMeasure 和 onLayout 方法，可以定义自己控件的丈量方式。

事件

在 Android 中，所有的按键，触屏等事件，都是从顶至下进行分发的。每个 ViewGroup 的对象，会维系一个 focused 变量，它表示在这个父控件中具备 focus 的控件，当有按键时间发生的时候，会找到这个 focused 子控件，并传递给它。同理，触屏事件的分发也是类似，只不过和 focus 无关，父控件会遍历所有子控件，看看谁处于触碰位置，从而传递给谁。

另外还有一些事件，逻辑上并不是从顶至下发起的。比如，当你修改某个子控件的内容，使得该子控件的大小和内容都发生了变化，就需要进行控件的重排和重绘，这些操作不仅是子控件自己的事情，需要整个控件树上的所有控件都需要配合。在 Android 中，处理这类事情的实现策略是子控件维系一个 ViewParent 对象，该对象象征着整个控件树的管理者，子控件产生影响整个控件树的事件时，

会通知到 ViewParent，ViewParent 会将其转换成一个自顶向下的事件，分发下去。

Android 的事件处理逻辑，采用的是观察者模式。Android 的控件提供了一些列的 add/set Listener 的接口，使得外部观察者，有机会处理控件事件。比如，你需要在某个 button 被点击时做一些事情，你就需要派生一个 View.OnClickListener 对象作为观察者，调用该控件的 setOnClickListener 接口注册进去，当 button 被点击，就可以获得处理点击事件的机会了。当然，有的时候，你需要处理的逻辑更为复杂，光是站在外面围观叫好不能解决问题，可能就需要派生某个控件，去重载 onXXXX 之类的事件处理函数，进行更完整的控制。

焦点

对于一个非触屏的机器，焦点的维系是一个极其重要的事情，而在有触屏的年代，焦点的地位虽有所下降，但依然还是需要妥善保护的。

Android 中，是以控件树为单位，来管理焦点的。每个控件，可以设置上下左右四向的 focus 转移对象。当在一个控件上发生焦点转移事件，Android 会如前述，自顶向下根据设定好的焦点转移逻辑，跳转到正确的控件上。和 Symbian 相比，真是，真是。。。

Layout

Layout 是一类特殊的 ViewGroup 控件，它们本身没有任何可显示内容，形如透明的玻璃盒子，存活的唯一理由，就是其中的内部结构，能够更好的摆放它的子控件们。

比如线性的 Layout，LinearLayout。放入这个 Layout 的子控件，会按水平或垂直方向，排排坐，一个挨着一个按顺序排列下去。TableLayout，可以将子控件按照表格的形式，一枚枚放置好。而 RelativeLayout 则更灵活，可以设定各个控件之间的对齐和排列关系，适合定制复杂的界面。

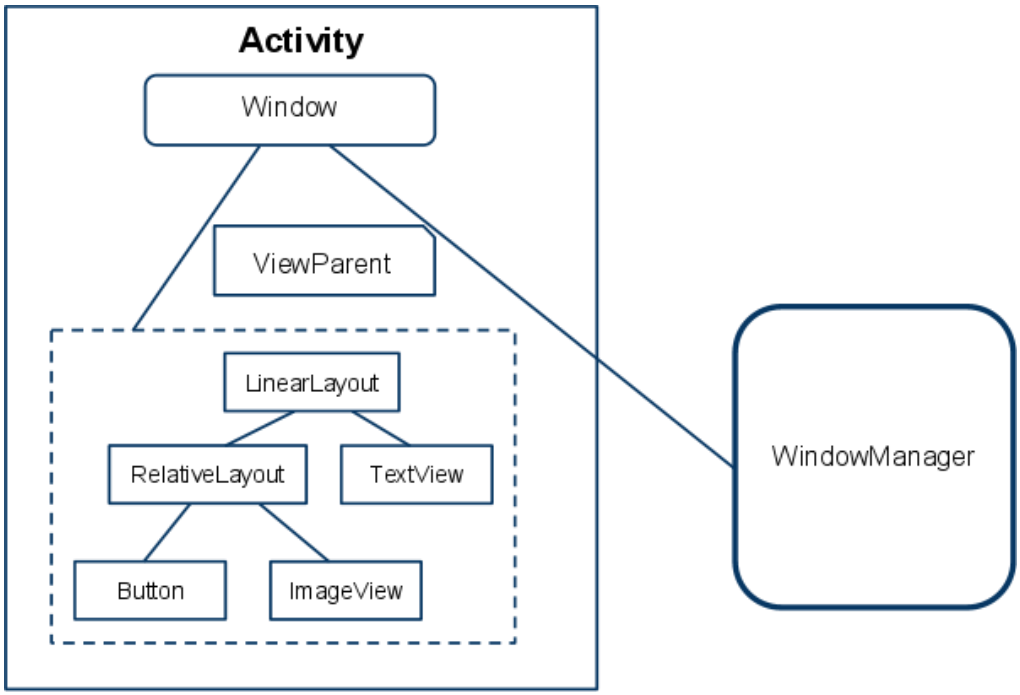
有了 Layout 的存在，控件和控件之间不再割裂的存在，而是更有机的结合在了一起，设定起来也更为方便。比 Symbian 那样人肉维系各个控件的关系，轻松自在多了。

更多

这些问题的完整答案，参见 SDK 中 View 的页面：/reference/android/view/View.html。

实现

有了这些对 Android 的 UI 控件的认知，可以看更整体性的实现细节，那就是 Activity 的 UI 实现。



如上图所示，假设你做了个如同虚线框中结构的一个界面，通过 Activity 的 setContentView 方法，塞进了 Activity 中，就会形成图示的一个逻辑关系。每一个 Activity，都包含一个 Window 对象，它表示的是一个顶级的一整屏幕上面的界面逻辑。在 Android 源码中，其实现是 MidWindow，它包含了一个 FrameLayout 对象，呈现出来就是那种带着一个 title 的界面样子。自定义的一堆控件，会插进 Window 的界面部分，在 Activity 中，所有事件的处理逻辑，是 Window 先享用，没消费掉在交由这堆控件吃剩的。

在整个控件树的最顶端，是一个逻辑的树顶，ViewParent，在源码中的实现是 ViewRoot。它是整个控件树和 WindowManager 之间的事件信息的翻译者。WindowManager 是 Android 中一个重要的服务。它将用户的操作，翻译成为指令，发送给呈现在界面上的各个 Window。Activity，会将顶级的控件注册到 WindowManager 中，当用户真是触碰屏幕或键盘的时候，WindowManager 就会通知到，而当控件有一些请求产生，也会经由 ViewParent 送回到 WindowManager 中。从而完成整个通信流程。