**Android布局文件：**

**包含内容：（自定义View、<style>和<item>、自定义图形shape、selector）**

**写出高效清晰Layout布局文件的一些技巧**

链接地址：<http://www.jianshu.com/p/961442b8e5ef>

**Android自定义View的实现方法**

链接地址：<http://blog.csdn.net/guolin_blog/article/details/17357967>

大概可以分为三种，自绘控件、组合控件、以及继承控件

自绘控件在：这个View上所展现的内容全部都是我们自己绘制出来的。绘制的代码是写在onDraw()方法中的

组合控件（标题栏）：不需要自己去绘制视图上显示的内容，而只是用系统原生的控件就好了，但我们可以将几个系统原生的控件组合到一起，这样创建出的控件就被称为组合控件。

继承控件：不需要自己重头去实现一个控件，只需要去继承一个现有的控件，然后在这个控件上增加一些新的功能，就可以形成一个自定义的控件了。

**<style>和<item>**

链接地址：<http://www.3g-edu.org/Android_free/art145.htm>

可以减少布局文件中的重复

Android也可以像HTML/CSS中的style一样，使用自定义的style样式。这就像Web开发中的CSS样式表，使我们的样式独立于内容进行设计开发。Android一般通过value 文件夹下面新建一个styles.xml文件来设置自定义样式。主题和样式都是通过在xml文件中预定义一系列属性值，通过这些属性值来形成统一的显示风格。不同的是，样式只能应用于某种类型的View；而主题刚好相反，不能应用于特定的View，而只能作用于一个或多个Activity，或是整个应用。

引用style这样做的好处是当需要改变属性的时候，只要改变style文件中的内容，布局结果就会发生变化

首先是自定义样式和主题。在项目的res/values/目录下添加styles.xml。如代码清单1所示。

代码清单1 styles.xml

<?xml version="1.0" encoding="utf-8"?>  
    <resources>  
    <!-- 定义my\_style\_1，没有指定parent，用系统默认的 -->  
    <style name="my\_style\_1">  
        <!-- 定义与指定View相关的若干属性 -->  
        <item name="android:hint">load from style 1</item>  
    </style>

</resources>

主题和样式都可以通过在<style>下添加多个<item>来完成其定义。<style>的属性，如表1所示。

表1 <style>的属性

|  |  |
| --- | --- |
| 属性名称 | 描述 |
| name | style的名字，引用该style时使用此名字 |
| parent | 一些在自定义的style中没有指定的属性会继承parent style中的值。parent可以是android预定义的resource，在父样式中的所有属性都将被应用于本样式中，同时可以覆盖父样式中的样式，也可以是自己定义的style。为可选项 |

下面讲解如何在程序中使用样式：一方面可以在layout的布局文件中指定自定义的样式；另一方面可以在java代码中指定。具体代码如代码清单2、3所示。

代码清单2 在布局文件中指定自定义样式

<EditText android:id="@+id/EditText03"   
        style="@style/my\_style\_3"  
        android:layout\_width="fill\_parent"   
        android:layout\_height="wrap\_content">  
    </EditText>

代码清单3 在java代码中指定自定义样式

public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        LinearLayout ll = (LinearLayout)findViewById(R.id.main);  
  
        EditText et = new EditText(this, null, R.attr.myStyle);//指定样式  
        ll.addView(et);  
    }

**背景选择器selector和shape：**

链接地址：<http://blog.csdn.net/cs_li1126/article/details/11781577>

 android:shape=["rectangle" | "oval" | "line" | "ring"]

 shape的形状，默认为矩形，可以设置为矩形（rectangle）、椭圆形(oval)、线性形状(line)、环形(ring)

## **Shape：**

## 作用：XML中定义的几何形状

位置：res/drawable/文件的名称.xml

使用的方法：

Java代码中：R.drawable.文件的名称

XML中：android:background="@drawable/文件的名称"

Android上支持以下几种属性gradient（渐变）、stroke（描边）、corners（圆角）、padding（填充）、solid（实心）、padding等

**常用的Shape布局**

**布局文件：**

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<shape** xmlns:android="http://schemas.android.com/apk/res/android" **>**
3. **</shape>**

**填充**组件内部颜色：<solid android:color="#aa00bb" />

切换**圆角**有俩种方式：

所有4个角，都是10dp：<corners  android:radius="10dp"/>

分别为:    上左角 - 上右角 -下左角 - 下右角

1. **<corners**
2. android:topLeftRadius="10dp"
3. android:topRightRadius="20dp"
4. android:bottomLeftRadius="60dp"
5. android:bottomRightRadius="10dp"
6. **/>**

 技巧：如果你把4个角设成为90的话，那么改图片是一个圆！

**描边：**

1. **<stroke**
2. android:width="1dp"
3. android:color="#000000"
4. **/>**

android:width 描边的宽度

android:color 描边的颜色

android:dashWidth 表示'-'横线的宽度

android:dashGap 表示'-'横线之间的距离

**<gradient>  渐变**

android:startColor  起始颜色

android:endColor  结束颜色

android:centerColor 整型   渐变中间颜色，即开始颜色与结束颜色之间的颜色

android:angle       整型   渐变角度(PS：当angle=0时，渐变色是从左向右。 然后逆时针方向转，当angle=90时为从下往上。angle必须为45的整数倍)

android:type  渐变的样式 liner线性渐变 radial放射性渐变 sweep  扫描线式的渐变。

android:useLevel   ["true" | "false"] 如果要使用LevelListDrawable对象，就要设置为true。设置为true无渐变。false有渐变色

android:gradientRadius 整型 渐变色半径.当 android:type="radial" 时才使用。单独使用 android:type="radial"会报错。

android:centerX     整型   渐变中心X点坐标的相对位置

android:centerY   整型   渐变中心Y点坐标的相对位置

**padding**

有bottom、top、left、right属性：

<padding android:top="2dp" android:bottom="2dp"/>

比如我们要自定义一个圆角Button，点击Button有些效果的变化，就要用到shape和selector。

**Selector**

位置：res/drawable/文件的名称.xml

使用的方法：

Java代码中：R.drawable.文件的名称

XML中：android:background="@drawable/文件的名称"

1. <?xml version="1.0" encoding="utf-8"?>
2. <selector xmlns:android="http://schemas.android.com/apk/res/android">
4. <item android:state\_selected="true">
5. <shape>
6. <gradient android:angle="270" android:endColor="#99BD4C"
7. android:startColor="#A5D245" />
8. <size android:height="60dp" android:width="320dp" />
9. <corners android:radius="8dp" />
10. </shape>
11. </item>
12. <item android:state\_pressed="true">
13. <shape>
14. <gradient android:angle="270" android:endColor="#99BD4C"
15. android:startColor="#A5D245"/>
16. <size android:height="60dp" android:width="320dp" />
17. <corners android:radius="8dp" />
18. </shape>
19. </item>
20. <item>
21. <shape>
22. <gradient android:angle="270" android:endColor="#A8C3B0"
23. android:startColor="#C6CFCE"/>
24. <size android:height="60dp" android:width="320dp" />
25. <corners android:radius="8dp" />
26. </shape>
27. </item>
28. </selector>

**可供设置的属性如下：**

drawable:引用的Drawable位图,我们可以把他放到最前面,就表示组件的正常状态~

state\_focused:是否获得焦点

state\_window\_focused:是否获得窗口焦点

state\_enabled:控件是否可用

state\_checkable:控件可否被勾选,eg:checkbox

state\_checked:控件是否被勾选

state\_selected:控件是否被选择,针对有滚轮的情况

state\_pressed:控件是否被按下

state\_active:控件是否处于活动状态,eg:slidingTab

state\_single:控件包含多个子控件时,确定是否只显示一个子控件

state\_first:控件包含多个子控件时,确定第一个子控件是否处于显示状态

state\_middle:控件包含多个子控件时,确定中间一个子控件是否处于显示状态

state\_last:控件包含多个子控件时,确定最后一个子控件是否处于显示状态

**Android中的13种Drawable：**

链接地址：<http://www.runoob.com/w3cnote/android-tutorial-drawable3.html>

**通过shape对drawable进行自定义：**

链接地址：<http://blog.csdn.net/hknock/article/details/6313722>

**如何修改组件的样式:**

链接地址：<http://www.cnblogs.com/vanezkw/archive/2012/09/04/2670366.html>