**自定义view**

参考文章：

<http://blog.csdn.net/lmj623565791/article/details/45022631；>

<http://www.cnblogs.com/jiayongji/p/5560806.html>

<http://blog.csdn.net/y444400/article/details/25559925>

<http://www.cnblogs.com/yishujun/p/5556145.html>

# 基本分类

组合view、自绘view、继承view

## 组合view

组合控件的意思就是，我们并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。一般继承GroupView极其子类。

1. **public** **class** TitleView **extends** FrameLayout {
3. **private** Button leftButton;
5. **private** TextView titleText;
7. **public** TitleView(Context context, AttributeSet attrs) {
8. **super**(context, attrs);
9. LayoutInflater.from(context).inflate(R.layout.title, **this**);
10. titleText = (TextView) findViewById(R.id.title\_text);
11. leftButton = (Button) findViewById(R.id.button\_left);
12. leftButton.setOnClickListener(**new** OnClickListener() {
13. @Override
14. **public** **void** onClick(View v) {
15. ((Activity) getContext()).finish();
16. }
17. });
18. }
20. **public** **void** setTitleText(String text) {
21. titleText.setText(text);
22. }
24. **public** **void** setLeftButtonText(String text) {
25. leftButton.setText(text);
26. }
28. **public** **void** setLeftButtonListener(OnClickListener l) {
29. leftButton.setOnClickListener(l);
30. }
32. }

## 自绘view

自绘控件的意思就是，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。继承view。

## 继承控件

继承控件的意思就是，我们并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。这种自定义控件的特点就是不仅能够按照我们的需求加入相应的功能，还可以保留原生控件的所有功能。

# 构造函数

View一般有三个构造方法，如下：

|  |
| --- |
| [View](http://blog.csdn.net/y444400/article/details/25559925" \t "http://blog.csdn.net/y444400/article/details/_blank)([Context](http://blog.csdn.net/y444400/article/details/25559925" \o "class in android.content" \t "http://blog.csdn.net/y444400/article/details/_blank) context)  //Simple constructor to use when creating a view from code. |
| [View](http://blog.csdn.net/y444400/article/details/25559925" \t "http://blog.csdn.net/y444400/article/details/_blank)([Context](http://blog.csdn.net/y444400/article/details/25559925" \o "class in android.content" \t "http://blog.csdn.net/y444400/article/details/_blank) context,[AttributeSet](http://blog.csdn.net/y444400/article/details/25559925" \o "interface in android.util" \t "http://blog.csdn.net/y444400/article/details/_blank) attrs)  //Constructor that is called when inflating a view from XML. |
| [View](http://blog.csdn.net/y444400/article/details/25559925" \t "http://blog.csdn.net/y444400/article/details/_blank)([Context](http://blog.csdn.net/y444400/article/details/25559925" \o "class in android.content" \t "http://blog.csdn.net/y444400/article/details/_blank) context,[AttributeSet](http://blog.csdn.net/y444400/article/details/25559925" \o "interface in android.util" \t "http://blog.csdn.net/y444400/article/details/_blank) attrs, int defStyle)  //Perform inflation from XML and apply a class-specific base style. |

## 自定义属性

1. 定义：

在values目录下建立attrs.xml文件，名字可随意。

<resources>   
    <!--name 是自定义属性名，一般采用驼峰命名，可以随意。 format 是属性的单位-->   
    <attr name="viewText" format="string" />   
    <attr name="viewTextColor" format="color" />   
    <attr name="viewTextSize" format="dimension" />   
    <!--name 是自定义控件的类名-->   
    <declare-styleable name="YText">   
        <attr name="viewText" />   
        <attr name="viewTextColor" />   
        <attr name="viewTextSize" />   
        <!--注意：一般情况是按照上面这样写，把属性单独定义在上面，然后在styleable这里面引用-->   
        <attr name="viewTextBg" format="color" />   
    </declare-styleable>   
</resources>

xmlns:customview=<http://schemas.android.com/apk/res-auto>

注意：在eclipse中如果要使用你自定义的属性 是不能用res-auto的必须得替换成你自定义view所属的包名，如果你在恰好使用的自定义属性被做成了lib那就只能使用res-auto了，而在android-studio里，无论你是自己写自定义view还是引用的lib里的自定义的view 都只能使用res-auto这个写法。以前那个包名的写法在android-studio里是被废弃无法使用的。

1. 使用：

TypedArray ta = context.obtainStyledAttributes(attrs, R.styleable.test);

String text = ta.getString(R.styleable.test\_testAttr);

int textAttr = ta.getInteger(R.styleable.test\_text, -1);

ta.recycle();

//TypedArray是一个用来存放由context.obtainStyledAttributes获得的属性的数组

        //在使用完成后，一定要调用recycle方法  ，释放资源

        //属性的名称是styleable中的名称+“\_”+属性名称

# 方法理解

## onDraw（）

此方法主要完成绘制工作，主要了解Paint，Canvas两个类

1. Paint：

画笔,包含了画几何图形、文本等的样式和颜色信息

<http://www.cnblogs.com/aibuli/p/efef9d774df97c553a8a0c0c3495ba35.html?utm_source=tuicool&utm_medium=referral>

[http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2014/1105/1907.html](http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2014/1105/1907.html" \o "http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2014/1105/1907.html)

 paint.setColor(Color.*RED*);                      //设置画笔颜色

  paint.setStyle(Style.*FILL*);   //设置画笔为空心     如果将这里改为Style.STROKE  这个图中的实线圆柱体就变成了空心的圆柱体

1. Canvas：

Canvas类简单理解就是表示一块画布,可以在上面画我们想画的东西

Canvas中的方法很多,Canvas可以绘制的对象有：

* **弧线(arcs) canvas.**
* **填充颜色(argb和color)**
* **Bitmap**
* **圆(circle和oval)**
* **点(point)**
* **线(line)**
* **矩形(Rect)**
* **图片(Picture)**
* **圆角矩形 (RoundRect)**
* **文本(text)**
* **顶点(Vertices)**
* **路径(path)**

canvas.save()：默认原点为(0,0)

canvas.scale（x,y）;扩大。x为水平方向的放大倍数，y为竖直方向的放大倍数   
canvas.rotate(angel):旋转.angle指旋转的角度，顺时针旋转。   
canvas.transform():切变。所谓切变，其实就是把图像的顶部或底部推到一边。   
canvas.saveLayer(bounds, paint, saveFlags);

推荐文章：[http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2012/1212/703.html](http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2012/1212/703.html" \o "http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2012/1212/703.html)

把当前的绘制的图像保存起来，让后续的操作相当于是在一个新的图层上的操作。   
canvas.restore();

把当前画布返回（调整）到上一个save()状态之前   
canvas.translate(dx, dy); //把当前画布的原点移到(dx,dy),后面的操作都以(dx,dy)作为参照点，