# 自定义View最基础元素

1. 资源文件，attrs.xml，描述自定义控件属性。
2. 在View的构造方法中获得我们自定义的属性
3. 自定义View类，实现onDraw等方法。
4. 在布局文件中添加自定义的view，“报名+类名”作为标签。
5. 在布局文件中添加自定义的命名空间xmls，形式：xmls：自定义命名=<http://schemas.android.com/apk/res/包名>

# 资源文件

例子：自定义空间属性，例如Android定义的属性“layout\_width”。

|  |
| --- |
| <declare-styleable name=*"rainbowbar"*>  <attr name=*"rainbowbar\_hspace"* format=*"dimension"*></attr>  <attr name=*"rainbowbar\_vspace"* format=*"dimension"*></attr>  <attr name=*"rainbowbar\_color"* format=*"color"*></attr>  </declare-styleable> |

分析：

标签declare-styleable声明样式名。

Attr 声明属性名，格式。

格式分为：

|  |  |
| --- | --- |
| reference： | 参考某一资源ID |
| color | 颜色值 |
| boolean | 布尔值 |
| dimension | 尺寸，dp |
| float | 浮点值 |
| integer | 整型值 |
| string | 字符串 |
| Fraction | 百分数 |
| Flag | 位或运算 |
| Enum | 枚举值 |

枚举类型举例：

   （1）属性定义：

**<declare-styleable name="名称">  
                   <attr name="orientation">  
                          <enum name="horizontal" value="0" />  
                          <enum name="vertical" value="1" />  
                   </attr>**

**</declare-styleable>**

    （2）属性使用：

**<LinearLayout**

**xmlns:android = "http://schemas.android.com/apk/res/android"  
                    android:orientation = "vertical"  
                    android:layout\_width = "fill\_parent"  
                    android:layout\_height = "fill\_parent"  
                    >  
            </LinearLayout>**

# 自定义View

基本步骤

1. 创建类继承于view，创建构造方法，基本上是三个构造方法。三参数为Context context, AttributeSet attrs, int defStyleAttr

在构造方法中获取自定义样式，自定义属性的值，以此来设计Paint对象。

具体用到TypedArray类。

代码例子：

|  |
| --- |
| TypedArray t = context.obtainStyledAttributes(attrs,  R.styleable.rainbowbar, 0, 0);  barColor = t.getColor(R.styleable.rainbowbar\_rainbowbar\_color, barColor);  mPaint.setColor(barColor); |

1. 实现Ondraw方法，该方法参数是Canvas。

该方法中可以调用Canvas绘图，其中画笔可以用自定义的paint。

基本有如下绘制方法：

|  |
| --- |
| **drawBitmap** |
| **clipPath** |
| **clipRect** |
| **drawText** |
| **drawRect ……** |

1. 实现onMeasure方法，在该方法中会传入布局xml中设定的长宽大小，经过自定义的处理最后调用setMeasuredDimension方法设置最终的View长和宽。

一个MeasureSpec封装了父布局传递给子布局的布局要求，每个MeasureSpec代表了一组宽度和高度的要求。一个MeasureSpec由大小和模式组成。它有三种模式：

|  |  |
| --- | --- |
| UNSPECIFIED，未指定 | 一般都是父控件是AdapterView，通过measure方法传入的模式 |
| EXACTLY，自身限定 | 设定了精确值，就是长宽=xxxdp |
| AT\_MOST，至多 | 当控件的layout\_width或layout\_height指定为WRAP\_CONTENT时，控件大小一般随着控件的子空间或内容进行变化，此时控件尺寸只要不超过父控件允许的最大尺寸即可。因此，此时的mode是AT\_MOST，size给出了父控件允许的最大尺寸。 |

简单例子:

|  |
| --- |
| protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {  super.onMeasure(widthMeasureSpec, heightMeasureSpec);  int width = getMySize(200, widthMeasureSpec);  int height = getMySize(200, heightMeasureSpec);  setMeasuredDimension(width, height);  }  private int getMySize(int defaultSize, int measureSpec) {  int mySize = defaultSize;  int mode = MeasureSpec.*getMode*(measureSpec);  int size = MeasureSpec.*getSize*(measureSpec);  switch (mode) {  case MeasureSpec.*UNSPECIFIED*: {//如果没有指定大小，就设置为默认大小  mySize = defaultSize;  break;  }  case MeasureSpec.*AT\_MOST*: {//如果测量模式是最大取值为size  //我们将大小取最大值,最大值是该View在父View中能显示的最大size。  mySize = size;  break;  }  case MeasureSpec.*EXACTLY*: {//如果是固定的大小，那就不去改变  mySize = size;  break;  }  }  return mySize;  } |

# 4 布局文件设置

首先布局文件要添加命名空间xmls，形式：（xmls：自定义命名=<http://schemas.android.com/apk/res/包名>）

该形式会**自动检查**包下面是否有定义attr属性描述。

如果不是该种形式，随便设定一个文本内容，则**不会自动检查。**

自定义View的布局时，要在使用自定义的属性前添加命名空间名。

例子：其中rainbow就是命名空间的名字。

|  |
| --- |
| <com.example.viewlearn.RainbowBar  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  rainbow:rainbowbar\_color=*"#766698"*  rainbow:rainbowbar\_hspace=*"80dp"*  rainbow:rainbowbar\_vspace=*"10dp"*  /> |