转载请注明链接：http://blog.csdn.net/feather_wch/article/details/79124608

> 本文要点：
>
> 1. 介绍Android中Drawable的相关知识点，并且介绍如何自定义Drawable。
> 2. Drawable能实现缩放、渐变、逐帧动画、静态矢量图、矢量图动画等功能
> 3. Drawable提供一种比自定义View更轻量级的解决办法，用于实现特定的效果
> 4. 布局使用 `xml` ，代码采用 `kotlin/java` 实现

Android的Drawable(36题)

版本：2018/2/11

> 个人总结的知识点外，部分知识点选自《Android开发艺术探索》-第六章 Drawable

1、Drawable是什么?

> 1. 一种可以在Canvas上进行绘制的抽象的概念
> 2. 颜色、图片等都可以是一个Drawable
> 3. Drawable可以通过XML定义，或者通过代码创建
> 4. Android中Drawable是一个抽象类，每个具体的Drawable都是其子类

2、Drawable的优点

> 1. 使用简单，比自定义View成本低
> 2. 非图片类的Drawable所占空间小，能减小apk大小

3、Drawable的内部宽/高

> 1. 一般 `getIntrinsicWidth/Height` 能获得内部宽/高
> 2. 图片Drawable其内部宽高就是图片的宽高
> 3. 颜色Drawable没有内部宽高的概念
> 4. 内部宽高不等同于它的大小，一般Drawable没有大小概念(作为View背景时，会被拉伸至View的大小)

Drawable的分类

4、BitmapDrawable的作用和使用

> 表示一种图片，可以直接引用原始图片或者通过XML进行描述

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@color/colorPrimary"
    android:antialias="true"
    android:dither="true"
    android:filter="true"
    android:gravity="center"
    android:mipMap="false"
    android:tileMode="disabled"
    />
```

5、Bitmap的属性

属性	作用	备注
android:src	图片资源ID	
android:antialias	图片抗锯齿-图片平滑，清晰度降低	应该开启
android:dither	开启抖动效果- 用于高质量图片在低质量屏幕上保存较好的显示效果 (不会失真)	应该开启
android:filter	开启过滤-在图片尺寸拉伸和压缩时保持较好的显示效果	应该开启
android:gravity	图片小于容器尺寸时，对图片进行定位-选项之间用‘	’来组合使用
android:mipMap	纹理映射-图像处理技术	默认false
android:tileMode	平铺模式-repeat单纯重复、mirror镜面反射、 clamp图片四周像素扩散	默认disable关闭

6、gravity属性详情

可选项	含义
top/bottom/left/right	将图片放在容器上/下/左/右，不改变图片大小
center_vertical/horizontal	垂直居中/水平居中，不改变图片大小
center	水平和垂直方向同时居中，不改变图片大小
fill_vertical/horizontal	垂直/水平方向填充容器
fill	水平和垂直方向同时填充容器

可选项	含义
clip_vertical/horizontal	垂直/水平方向的裁剪-较少使用

7、NinePatchDrawable(.9图片)的作用

> 1. 自动根据宽高进行缩放且不会失真
> 2. 实际使用，可以直接引用图片或者通过XML描述

```
<?xml version="1.0" encoding="utf-8"?>
<nine-patch
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@color/colorPrimary"
    android:antialias="true"
    android:dither="true"
    android:filter="true"
    android:gravity="center"
    android:mipMap="false"
    android:tileMode="disabled"
    />
```

8、ShapeDrawable的作用

> 1. 通过颜色构造的图形
> 2. 可以是纯色的图形
> 3. 也可以是有渐变效果的图形
> 4. `shape` 标签创建的Drawable实体是 `GradientDrawable`

9、ShapeDrawable的使用

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">

    <corners
        android:radius="10dp"
        android:topLeftRadius="10dp"
        android:topRightRadius="10dp"
        android:bottomLeftRadius="10dp"
        android:bottomRightRadius="10dp"/>
    <gradient
        android:angle="45"
        android:centerX="30"
        android:centerY="30"
        android:centerColor="@color/colorAccent"
        android:endColor="@color/colorPrimary"
        android:startColor="@color/colorPrimaryDark"
        android:gradientRadius="20"
        android:type="linear"
        android:useLevel="true" />
    <padding
        android:left="10dp"
        android:top="10dp"
        android:right="10dp"
        android:bottom="10dp" />
    <size
        android:width="200dp"
        android:height="200dp" />
    <solid
        android:color="@color/colorPrimary"/>
    <stroke
        android:width="10dp"
        android:color="@color/colorAccent"
        android:dashWidth="5dp"
        android:dashGap="3dp"/>

</shape>
```

10、ShapeDrawable的属性介绍

属性/标签	作用	备注
android:shape	图形的形状： rectangle矩形、 oval椭圆、line横线、 ring圆环	`corners` 标签对应于矩形； line和ring通过 `stroke` 指定线的宽度和颜色； ring圆环有五个特殊的shape属性
`corners` 标签	四个角的角度	

属性/标签	作用	备注
`gradient` 标签	渐变效果- android:angle表示渐变角度，必须为45的倍数	android:type指明渐变类型：linear线性， radial径向、sweep扫描
`solid` 标签	纯色填充	与gradient标签排斥
`stroke` 标签	描边	有描边线和虚线
`size` 标签	表示shape的固有大小， 并非最终显示的大小	没有时getIntrinsicWidth返回-1； 能指明Drawable的固有宽高， 但如果作为View背景还是会被拉伸

11、LayerDrawable的作用

> 1. XML标签为 `layer-list`
> 2. 层次化的Drawable合集
> 3. 可以包含多个 `item` ， 每个item表示一个Drawable
> 4. item中可以通过 `android:drawable` 直接引用资源
> 5. `android:top` 等表示Drawable相当于View上下左右的偏移量

12、LayerDrawable的使用(微信文本输入框)

```xml
<?xml version="1.0" encoding="utf-8"?>
<layer-list
    xmlns:android="http://schemas.android.com/apk/res/android">

    <item>
        <shape android:shape="rectangle">
            <solid
                android:color="#0ac39e"/>
        </shape>
    </item>

    <item
        android:bottom="6dp">
        <shape android:shape="rectangle">
            <solid
                android:color="#FFFFFF"/>
        </shape>
    </item>

    <item
        android:bottom="1dp"
        android:left="1dp"
        android:right="1dp">
        <shape android:shape="rectangle">
            <solid
                android:color="#FFFFFF"/>
        </shape>
    </item>

</layer-list>
```

13、StateListDrawable的作用

> 1. 对应于 `selector` 标签
> 2. 用于View根据状态选择不同的Drawable

14、StateListDrawable的使用和要点

```xml
<?xml version="1.0" encoding="utf-8"?>
<selector
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:constantSize="false" //StateListDrawable的固有大小是否根据状态而改变，默认false=根据状
    android:dither="true"        //是否开启抖动-让高质量图片在低质量屏幕上依旧效果好，默认true开启
    android:variablePadding="false" //padding是否根据状态的改变而改变，不建议开启(false)
    >
    <item android:state_pressed="true"  //Button被按下后却没有松开的状态
        android:drawable="@color/colorAccent"/>
    <item android:state_focused="true"  //View获取了焦点
        android:drawable="@color/colorPrimary"/>
    <item android:state_selected="true" //用户选择了View
        android:drawable="@color/colorPrimary"/>
    <item android:state_checked="true" //用户选中了View，一般用于CheckBox这类在选中和没有选中状态
        android:drawable="@drawable/ic_launcher_background"/>
    <item android:state_enabled="true" //View处于可用状态
        android:drawable="@drawable/ic_launcher_foreground"/>
    <item android:drawable="#FFFFFF"/> //默认Drawable: 按顺序向下匹配，需要放在最下方，因为可以匹
</selector>
```

15、LevelListDrawable的作用

> 1. 对应于 `level-list` 标签
> 2. 拥有多个item，每个item都有 `maxLevel` 和 `minLevel`
> 3. `Level` 的范围为 `0~10000`
> 4. 给定level后，会按 从上至下 的顺序匹配，直到找到范围合适的Drawable，并返回
> 5. item的level一定要降序或者升序
> 6. 调用View的 `getBackground` 获得Drawable对象，并调用 `setLevel` 设置等级 `level`
> 7. ImageView的 `setImageLevel()` 能快速指定 `src` 引用的Drawable的 `Level`
> 8. LevelListDrawable是根据 `level` 改变，选择不同的Drawable，能用于实现进度条、音量调节等等

16、LevelListDrawable的使用

```xml
<?xml version="1.0" encoding="utf-8"?>
<level-list xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:minLevel="0" android:maxLevel="10" android:drawable="@drawable/d1" />
    <item android:minLevel="11" android:maxLevel="20" android:drawable="@drawable/d2" />
    <item android:minLevel="21" android:maxLevel="30" android:drawable="@drawable/d3" />
    <item android:minLevel="31" android:maxLevel="40" android:drawable="@drawable/d4" />
</level-list>
```

17、TransitionDrawable的作用

> 1. 对应于 `transition` 标签
> 2. 实现两个Drawable之前的淡入淡出效果

3. 获得背景的 `TransitionDrawable` 后，通过 `startTransition` 和 `reverseTransition` 方法实现效果和逆过程

18、TransitionDrawable的使用

```
<?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/transition_drawable"
        android:drawable="@drawable/ic_launcher"
        android:top="10dp"    //四周的偏移量
        android:bottom="10dp"
        android:right="10dp"
        android:left="10dp"/>
    <item android:drawable="@drawable/ic_launcher_round" />
</transition>
```

19、InsetDrawable的作用和使用

1. 对应 `inset` 标签
2. 将其他Drawable内嵌到自身，并在四周留出间距
3. View需要背景比自己实际区域要小的时候，可以使用 `inset` ， `layer-list` 也可以实现该需求

```
<inset xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/ic_launcher"
    android:insetTop="10dp"
    android:insetBottom="10dp"
    android:insetLeft="10dp"
    android:insetRight="10dp">
</inset>
```

20、ScaleDrawable的作用

1. 对应于 `scale` 标签
2. 根据自己的等级 `level` (0~10000)将指定的Drawable缩放到一定比例
3. `android:scaleHeight="70%"` 用于指定宽高的缩放比例=为原来的 `30%`
4. ScaleDrawable的 `level` 为0，不可见。为10000时，不缩放。
5. 一般将 `level` 设置为 `1` ，就会按照属性指定的比例缩放。其他值也会改变缩放效果。
6. `android:scaleGravity` 属性和 `gravity` 属性完全一致

21、ScaleDrawable的使用

```
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/ic_launcher"
    android:scaleGravity="center"
    android:scaleHeight="70%"
    android:scaleWidth="70%">
</scale>
```

22、ClipDrawable的作用

> 1. 对应于 `clip` 标签
> 2. 根据自己当前的等级 `level` (0~10000)来裁剪另一个Drawable
> 3. 裁剪方向由 `clipOrientation` 和 `gravity` 属性共同控制
> 4. `level` 为0，Drawable不可见；10000表示不裁剪；为8000，表示裁减了2000；为1，表示裁剪了9999

23、ClipDrawable的gravity

可选项	含义
top/bottom	将图片放在容器上/下。若为 垂直裁剪 ，从另一头开始裁剪； 若为 水平裁剪 ，则从水平方向左/右两头开始裁剪
left/right	将图片放在容器左/右。若为 水平裁剪 ，从另一头开始裁剪； 若为 垂直裁剪 ，则从垂直方向上/下两头开始裁剪
center_vertical/horizontal/center	垂直居中/水平居中/两个方向均居中。 效果只与 `clipOrientation` 有关：水平裁剪， 左右两头开始裁剪；垂直裁剪，上下两头开始裁剪
fill_vertical/horizontal	垂直/水平方向填充容器。gravity和orientation方向相同时， 不裁剪；方向不同时，按照orientation的方向， 从两头开始裁剪
fill	水平和垂直方向同时填充容器，没有裁剪效果
clip_vertical/horizontal	效果类似center_center

24、AnimationDrawable的作用

> 1. 对应于 `animation-list` 标签
> 2. 用于实现 逐帧动画 效果
> 3. `android:oneShot` 决定是循环播放还是播放一次，false：循环播放
> 4. `item` 中设置一帧一帧的Drawable以及持续时间
> 5. AnimationDrawable的 `setOneShot(boolean flag)` 和 `android:oneShot` 配置一样
> 6. `addFrame (Drawable frame, int duration)` 动态的添加一个图片进入该动画中
> 7. `stop()`和`start()` 用于停止和开始/继续播放，停止时会停留在当前一帧上

25、AnimationDrawable的使用

```xml
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/shake_anim_01" android:duration="100"/>
    <item android:drawable="@drawable/shake_anim_02" android:duration="100"/>
    <item android:drawable="@drawable/shake_anim_03" android:duration="100"/>
    <item android:drawable="@drawable/shake_anim_04" android:duration="100"/>
    <item android:drawable="@drawable/shake_anim_05" android:duration="100"/>
    <item android:drawable="@drawable/shake_anim_06" android:duration="100"/>
    <item android:drawable="@drawable/shake_anim_07" android:duration="100"/>
    <item android:drawable="@drawable/shake_anim_08" android:duration="100"/>
    <item android:drawable="@drawable/shake_anim_09" android:duration="100"/>
    <item android:drawable="@drawable/shake_anim_10" android:duration="100"/>
</animation-list>
```

```
val imageview = findViewById<ImageView>(R.id.imaview)
(imageview.drawable as AnimationDrawable).start() //开始播放
```

27、ShapeDrawable的OvalShape、RectShape、ArcShape和PaintDrawable的作用和使用

> 1. 用于获得有 `shape形状` 的 `drawable` (椭圆、长方形、扇形以及更为通用PaintDrawable-具有圆角和边界)

```
/**=====================================================
 *  一个继承自ShapeDrawable更为通用的Drawable：具有圆角
 *=====================================================*/
PaintDrawable drawable3 = new PaintDrawable(Color.GREEN);
drawable3.setCornerRadius(30);
findViewById(R.id.textView3).setBackgroundDrawable(drawable3);

/**=============================================
 *   通过Shape构造出相应的ShapeDrawable
 *=============================================*/
//椭圆形形状 : shape赋予ShapeDrawable
OvalShape ovalShape = new OvalShape();
ShapeDrawable drawable1 = new ShapeDrawable(ovalShape);
drawable1.getPaint().setColor(Color.BLUE);
drawable1.getPaint().setStyle(Paint.Style.FILL);
findViewById(R.id.textView1).setBackgroundDrawable(drawable1);

//矩形形状  : shape赋予ShapeDrawable
RectShape rectShape = new RectShape();
ShapeDrawable drawable2 = new ShapeDrawable(rectShape);
drawable2.getPaint().setColor(Color.RED);
drawable2.getPaint().setStyle(Paint.Style.FILL);
findViewById(R.id.textView2).setBackgroundDrawable(drawable2);

//扇形、扇面形状 : shape赋予ShapeDrawable
//顺时针,开始角度30，  扫描的弧度跨度180
ArcShape arcShape = new ArcShape(30, 180);
ShapeDrawable drawable4 = new ShapeDrawable(arcShape);
drawable4.getPaint().setColor(Color.YELLOW);
drawable4.getPaint().setStyle(Paint.Style.FILL);
findViewById(R.id.textView4).setBackgroundDrawable(drawable4);
```

26、其余Drawable及其功能

Drwable分类	xml标签	功能
ColorDrawable	`color`	纯色Drawable
RotateDrawable	`rotate`	实现旋转效果
RippleDrawable	`ripple`	触摸反馈动画，Andorid 5.0推出
VectorDrawable	`vector` 中使用 `path`	【绘制静态图】使用矢量图SVG, 能绘制一张图就能适配不同分辨率, Android 5.0推出
AnimatedVectorDrawable	`animated-vector`	【动画矢量图】 针对VectorDrawable来做动画， Android 5.0

Drwable分类	xml标签	功能
AnimatedStateListDrawable	`animated-selector`	【动画型StateListDrawable】 在View状态改变时，展示动画

自定义Drawable

27、自定义Drawable

> 1. 一般作为ImageView的图像来显示
> 2. 另一个是作为View的背景
> 3. 自定义Drawable主要就是实现 `draw` 方法
> 4. `setAlpha` 、 `setColorFilter` 、 `getOpacity` 也需要重写，但是模板固定
> 5. 当自定义Drawable有固定大小时(比如绘制一张图片)，需要重写 `getIntrinsicWidth()/getIntrinsicHeight()` 方法(默认返回-1)，会影响到 `View` 的 `wrap_content` 布局
> 6. 内部固定大小不等于Drawable的实际区域大小， `getBounds` 能获得实际区域大小

28、自定义Drawable模板代码

```
class CustomDrawable(color: Int) : Drawable(){
    var mPaint: Paint
    init {
        mPaint = Paint(Paint.ANTI_ALIAS_FLAG)
        mPaint.color = color
    }
    override fun draw(canvas: Canvas) {
        val rect = bounds
        canvas.drawCircle(rect.exactCenterX(),
                rect.exactCenterY(),
                Math.min(rect.exactCenterX(), rect.exactCenterY()),
                mPaint)
    }

    override fun setAlpha(alpha: Int) {
        mPaint.alpha = alpha
        invalidateSelf()
    }
    override fun setColorFilter(colorFilter: ColorFilter?) {
        mPaint.colorFilter = colorFilter
        invalidateSelf()
    }
    override fun getOpacity(): Int {
        //not sure, so be safe
        return PixelFormat.TRANSLUCENT
    }
}
```

SVG矢量图

29、SVG是什么?(Scalable Vetor Graphics)

> 1. 可伸缩矢量图(Android 5.0推出)
> 2. 定义用于网络的基于矢量的图形(在Web上应用非常广泛)
> 3. 使用XML格式定义图形
> 4. 图像缩放不会影响质量
> 5. 万维网联盟标准(与DOM和XSL之类的W3C标准是一个整体)

30、SVG和Bitmap区别

> 1. SVG是一个绘图标准。
> 2. Bitmap是通过每个像素点上存储色彩信息来表示图像。
> 3. SVG放大不会失真, Bitmap会失真。
> 4. Bitmap需要为不同分辨率设计多套图表，SVG绘制一张图就能适配不同分辨率。

31、静态矢量图SVG-VectorDrawable

> 1. 基于XML的静态矢量图
> 2. 采用标签 `vector`
> 3. `vector` 中 `path` 是最小单位，创建SVG-用指令绘制SVG图形
> 4. `vector` 中 `group` 将不同 `path` 组合起来

32、VectorDrawable的 `vector` 标签有哪些属性

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="200dp"  // SVG的具体大小
    android:height="200dp"
    android:viewportWidth="100" //将宽度分为多少份，与path配合(50份等于100dp)
    android:viewportHeight="100">
    <group>   //将不同`path`组合起来
        <path    //SVG树形结构的最小单位，用指令绘制SVG图形
            android:name="path1" //该path的名称
            android:pathData="M 20,80 L 50,80 80,80"
            android:strokeColor="@color/colorAccent"
            android:strokeWidth="3"
            android:strokeLineCap="round"/>
        <path
            .../>
    </group>
</vector>
```

33、VectorDrawable的 `path` 标签的全部指令

指令	含义

指令	含义
M = moveto(M X, Y)	将画笔移动到指定的坐标位置，但并未绘制
L = lineto(L X, Y)	画直线到指定的坐标位置
H = horizontal lineto(H X)	画水平线到指定X坐标位置
V = vertical lineto(V Y)	画水平线到指定Y坐标位置
C = curveto(C X1, Y1, X2, Y2, ENDX, ENDY)	三次贝赛曲线
S = smooth curveto(S X2, Y2, ENDX, ENDY)	三次贝赛曲线
Q = quadratic Belzier curve(Q X, Y, ENDX, ENDY)	二次贝赛曲线
T = smooth quadratic Belzier curveTO(T ENDX, ENDY)	映射前面路径后的终点
A = elliptical Arc(A RX, RY, XROTATION, FLAG1, FLAG2, X, Y)	弧线(RX/RY：椭圆半轴大小 XROTATION：椭圆X轴与水平方向顺时针方向夹角)
Z = closepath()	关闭路径

1. 坐标轴以(0, 0)为中心， X轴水平向右， Y轴水平向下
2. 指令大写-绝对定位，参考全局坐标系；指令小写-相对定位，参考父容器坐标系
3. 指令和数据间空格可以省略
4. 同一指令出现多次，可以只用一个。
5. A的参数：RX/RY：椭圆半轴大小 XROTATION：椭圆X轴与水平方向顺时针方向夹角 FLAG1：1-大角度弧线 0-小角度弧线 FLAG2：起点到终点的方向，1-顺时针，2-逆时针 X/Y：终点坐标

34、VectorDrawable实例

```
//1. 使用`vector`标签定义矢量图VectorDrawable(ic_black_24dp.xml)
<vector xmlns:android="http://schemas.android.com/apk/res/android"
        android:width="24dp"
        android:height="24dp"
        android:viewportWidth="24.0"
        android:viewportHeight="24.0">
        <group
           android:name="test"> //该组的名称：可以在AnimatedVectorDrawable中指定动画效果
           <path
              android:fillColor="#FF000000"
              android:pathData="M12,6c1.11,0 2,-0.9 2,-2 0,-0.38 -0.1,-0.73 -0.29,-1.03L12,0l-1
        </group>
</vector>
//2. 使用矢量图
<ImageView
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:src="@drawable/ic_black_24dp"/>
```

35、矢量图动画-AnimatedVectorDrawable

1. 针对 静态矢量图-VectorDrawable 来做动画
2. xml 标签为 animated-vector
3. 在 target 子标签下指明 VectorDrawable 的名字(都是 android:name="..." 属性指明)，并指定动画效果 android:animation="@animator/..."

36、AnimatedVectorDrawable实例

```
//1. 静态矢量图-VectorDrawable(vector_two_line.xml)
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="200dp"
    android:height="200dp"
    android:viewportWidth="100"
    android:viewportHeight="100">
    <group>
        <path
            android:name="path1" //路径1的名称
            android:pathData="M 20,80 L 50,80 80,80"
            android:strokeColor="@color/colorAccent"
            android:strokeWidth="3"
            android:strokeLineCap="round"/>
        <path
            android:name="path2" //路径2的名称
            android:pathData="M 20,20 L 50,20 80,20"
            android:strokeColor="@color/colorAccent"
            android:strokeWidth="3"
            android:strokeLineCap="round"/>
    </group>
</vector>

//2. 轨迹动画效果-属性动画ObjectAnimator(res/animator/trimpath_animator)
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="1000"
    android:propertyName="trimPathEnd"
    android:valueFrom="0"
    android:valueTo="1"
    android:valueType="floatType"
    android:interpolator="@android:interpolator/accelerate_decelerate">
</objectAnimator>

//3. 粘合静态SVG和属性动画：AnimatedVectorDrawable(vector_trimpath_anim.xml)
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/vector_two_line"> //静态SVG
   <target
       android:animation="@animator/trimpath_animator" //属性动画
       android:name="path1"> //静态SVG中路径1的名称
   </target>
   <target
       android:animation="@animator/trimpath_animator" //属性动画
       android:name="path2"> //静态SVG中路径2的名称
   </target>
</animated-vector>

//4. 布局中使用AnimatedVectorDrawable
<ImageView
            android:id="@+id/trimpath_anim_imageview"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1"
            android:src="@drawable/vector_trimpath_anim"/> //动画矢量图
```

> 代码中开启动画：

```
ImageView imageView = (ImageView) findViewById(R.id.trimpath_anim_imageview);
((Animatable)imageView.getDrawable()).start();
```