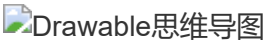
Drawable(67题)

版本：2018/5/11-1(0:44)

思维导图

思维导图详情点这里


基本知识点

1.Drawable是什么

1、Drawable是什么?

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

2.Drawable的优点

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

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分类

1-ColorDrawable(color)

4、ColorDrawable的作用

> 1. 纯色Drawable
> 2. 标签为 `color`

5、ColorDrawable的实例

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

2-BitmapDrawable(xml标签:bitmap)

6、BitmapDrawable的作用

> 1. 表示为一种图片
> 2. 标签为 `bitmap`

7、BitmapDrawable可以直接引用原始图片(如ImageView)

```
android:src="@drawable/ic_launcher" //引用原始图片


imageView.getDrawable(); //获取src指定的Drawable，本身就是BitmapDrawable
```

8、BitmapDrawable也可以通过 `XML` 进行描述

```
//mybitmap.xml
<?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"
    android:tint="@color/colorPrimaryDark"
    android:tintMode="screen"
    />


android:src="@drawable/bitmap" //引用xml描述的BitmapDrawable
```

属性

9、BitmapDrawable具有的基本属性

属性	作用	备注

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: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

android:tint和android:tintMode

9、BitmapDrawable的属性 `android:tint` 的作用

> 1. `android:tint="@color/colorPrimary"`
> 2. 会将所有有颜色的地方都着色为指定颜色，且保留透明度。

10、android:tintMode="xxx"有哪些着色模式

> 1. android:tint指定的颜色就是 `src源` ，原来的内容属于 `dst目标`
> 2. 着色模式按照 `PorterDuffMode` 进行混合，可以参考View绘制详解

11、着色模式的分类

着色模式	作用
src_in	【默认】着色有颜色区域
src_over	
src_atop	
add	相加
multiply	混合相乘
screen	混合变淡

12、着色模式的设置(Java)

```
View bitmapView = findViewById(R.id.bitmap_view);
BitmapDrawable bitmapDrawable = (BitmapDrawable) bitmapView.getBackground();
bitmapDrawable.setTintMode(PorterDuff.Mode.SCREEN);
```

android:gravity

13、android:gravity属性的作用

> 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

可选项	含义

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

3-NinePatchDrawable(nine-patch)

14、NinePatchDrawable的作用

> 1. 自动根据宽高进行缩放且不会失真
> 2. 使用：可以直接引用图片或者通过XML描述
> 3. 也具有 `tint`和`tintMode` 属性。

15、NinePatchDrawable的实例

```
<?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"
    />
```

4-GradientDrawable(shape)

16、GradientDrawable的作用

> 1. 能构造出纯色、渐变、圆角等效果的图形。
> 2. `shape` 标签创建的Drawable实体是 `GradientDrawable`

17、GradientDrawable的实例

```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>
```

属性

18、GradientDrawable的属性

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	
`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

属性/标签	作用	备注
`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

5-LayerDrawable(layer-list)

19、LayerDrawable的作用

> 1. 层次化的Drawable合集.
> 2. 可以包含多个 `item` ， 每个item表示一个Drawable, 后者覆盖在前者Item之上。
> 3. item中可以通过 `android:drawable` 直接引用资源。
> 4. `android:top` 等表示Drawable相当于View上下左右的偏移量。

20、LayerDrawable的实例

> 1、微信文本输入框:

```
<?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>
```

> 2、图片的默认图片-不会被拉伸

6-StateListDrawable(selector)

21、StateListDrawable的作用

> 1. 用于View根据不同状态选择不同的 `Drawable`
> 2. 标签为 `selector`

22、与 `AnimatedStateListDrawable` 的区别

> 1. `StateListDrawable` 瞬间切换图片，显得比较突兀。
> 2. `AnimatedStateListDrawable` 是根据不同状态选择不同的动画效果，更平滑流畅。

23、StateListDrawable的实例

```
<?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=根据状态而改变
    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之间切换的Vie
        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，因为可以匹配任何状态
</selector>
```

> 补充Item状态：
> state_checkable: 是否可以设置checked状态的效果

7-LevelListDrawable(level-list)

24、LevelListDrawable的作用

> 1. `LevelListDrawable` 根据 `level` 选择对应的 `Drawable`
> 2. 能用于实现进度条、音量调节等场景。

25、LevelListDrawable的Level等级

> 1. 每个item都有 `maxLevel` 和 `minLevel`
> 2. `Level` 的范围为 `0~10000`
> 3. Item的level一定要降序或者升序---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。

26、LevelListDrawable的实例

```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>
```

```
//1. 一般View: 获取Drawable并指定level登记
view.getBackground().setLevel(xxx);

//2. ImageView
imageView.setImageLevel(xxx);
```

8-TransitionDrawable(transition)

27、TransitionDrawable的作用

> 1. 实现两个Drawable之间的淡入淡出效果
> 2. 标签为 `transition`

28、TransitionDrawable的实例

```xml
// 1、src/drawable/transition.xml---指定两个item
<?xml version="1.0" encoding="utf-8"?>
<transition xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/transition_drawable"
        android:bottom="10dp"
        android:drawable="@drawable/beauty1"
        android:left="10dp"
        android:right="10dp"
        android:top="10dp" />
    <item android:drawable="@drawable/beauty2" />
</transition>

// 2、将该drawable作为背景
<View
    xxxxxx
    android:background="@drawable/transition"/>
```

```
// 1. 获取Drawable
TransitionDrawable drawable = (TransitionDrawable)view.getBackground();
// 2. 第一个Item渐变到第二个item
drawable.startTransition(1000);
// 3. 逆向动画
drawable.reverseTransition(1000);
```

9-InsetDrawable(inset)

28、InsetDrawable的作用

> 1. 将其他Drawable内嵌到自身，并在四周留出间距
> 2. 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

29、InsetDrawable的实例

```
<?xml version="1.0" encoding="utf-8"?>
<inset xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/watch_dog1"
    android:insetTop="10dp"
    android:insetBottom="10dp"
    android:insetLeft="10dp"
    android:insetRight="10dp">
</inset>
```

10-ScaleDrawable(scale)

30、ScaleDrawable的作用

1. 缩放图片，根据属性 `android:scaleHeight/Width` 以及 `level` 共同决定缩放比例。
2. `android:scaleWidth/Height="70%"` 用于指定宽高的缩放比例=为原来的 `30%` 。
3. `level` 取值范围为：0~10000
4. `android:scaleGravity` 属性和 `gravity` 属性完全一致(请参考BitmapDrawable的gravity属性)。

31、ScaleDrawable的level等级

1. level=0：不可见。(默认值)
2. level=1: 按照属性指定的比例缩放。(一般情况推荐)
3. level=10000时: 不缩放。
4. level=2~99999：在属性缩放基础上进行缩放。

32、ScaleDrawable的实例

> 1、xml定义ScaleDrawable

```
//src/drawable/scaledrawable.xml
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/call_of_duty"
    android:scaleGravity="center"
    android:scaleHeight="70%"
    android:scaleWidth="70%">
</scale>
```

> 2、引用Drawable

```
<View
    android:id="@+id/scale_view"
    xxx
    android:background="@drawable/scaledrawable"/>
```

> 3、Java中设置Level

```
// 必须要设置level等级，不然会不显示。
View scaleView = findViewById(R.id.scale_view);
ScaleDrawable scaleDrawable = (ScaleDrawable) scaleView.getBackground();
scaleDrawable.setLevel(1);
```

11-ClipDrawable(clip)

33、ClipDrawable的作用

> 1. 用于裁剪图片。
> 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。

gravity属性

34、ClipDrawable的属性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

35、ClipDrawable的实例

> 1、XML定义

```
<?xml version="1.0" encoding="utf-8"?>
<clip xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/ic_launcher"
    android:clipOrientation="horizontal"
    android:gravity="top"
    android:level="10000">
</clip>
```

> 2、使用

```
        <ImageView
            android:id="@+id/clip_imgaeview"
            android:layout_width="150dp"
            android:layout_height="150dp"
            android:src="@drawable/clipdrawable"/>


        ImageView imageView = findViewById(R.id.clip_imgaeview);
        imageView.setImageLevel(10000); //不裁剪，=8000表示裁剪掉2000
```

12-RotateDrawable(rotate)

36、RotateDrawable的作用

> 1. 用于旋转图片
> 2. `android:fromDegrees="0"` 和 `android:toDegrees="180"` :旋转范围
> 3. `android:pivotX="50%"` 和 `android:pivotY="50%"` : 旋转中心
> 4. 通过 `level` 设置旋转角度。范围：0~10000。

37、RotateDrawable的实例

> 1、xml定义RotateDrawable

```
<?xml version="1.0" encoding="utf-8"?>
<rotate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/ic_launcher"
    android:fromDegrees="0"
    android:toDegrees="180"
    android:pivotX="50%"
    android:pivotY="50%">
</rotate>
```

> 2、引用drawable

```
    <Button
          android:id="@+id/rotate_btn"
          xxx
          android:background="@drawable/rotatedrawable"/>
```

> 3、使用

```
    mRotateButton = findViewById(R.id.rotate_btn);
    mRotateDrawable = (RotateDrawable) mRotateButton.getBackground();
    //1. 不旋转
    mRotateDrawable.setLevel(0);
    //2. 完全旋转
    mRotateDrawable.setLevel(10000);
    //3. 旋转50%
    mRotateDrawable.setLevel(5000);
```

13-AnimationDrawable(animation-list)

38、AnimationDrawable的作用

> 1. 用于实现 逐帧动画 效果。
> 2. `item` 中设置一帧一帧的Drawable以及持续时间。

属性

39、AnimationDrawable的属性

属性	作用	Java代码设置

属性	作用	Java代码设置
android:oneshot="true"	是否循环一次：false=循环播放；true=播放一次。	setOneShot(boolean flag)
android:variablePadding="true"		
android:visible="true"		

40、AnimationDrawable的实例

> 1、xml定义AnimationDrawable

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false"
    android:variablePadding="true"
    android:visible="true">

    <item android:drawable="@drawable/ic_battery_charging_0_black_24dp" android:duration="200"/>
    <item android:drawable="@drawable/ic_battery_charging_20_black_24dp" android:duration="200"/>
    <item android:drawable="@drawable/ic_battery_charging_30_black_24dp" android:duration="400"/>
    <item android:drawable="@drawable/ic_battery_charging_50_black_24dp" android:duration="200"/>
    <item android:drawable="@drawable/ic_battery_charging_60_black_24dp" android:duration="400"/>
    <item android:drawable="@drawable/ic_battery_charging_80_black_24dp" android:duration="200"/>
    <item android:drawable="@drawable/ic_battery_charging_90_black_24dp" android:duration="200"/>
    <item android:drawable="@drawable/ic_battery_charging_full_black_24dp" android:duration="400"/>
</animation-list>
```

> 2、引用

```
        <ImageView
            android:id="@+id/animation_imageview"
            android:layout_width="100dp"
            android:layout_height="50dp"
            android:src="@drawable/animationdrawable"/>
```

> 3、使用

```
        ImageView animationImageView = findViewById(R.id.animation_imageview);
        AnimationDrawable animationDrawable = (AnimationDrawable) animationImageView.getDrawable();
        //1. 开启动画 or 继续播放
        animationDrawable.start();
        //2. 停止动画(会停留在当前一帧画面上)
        animationDrawable.stop();
        //3. 动态的添加一个图片进入该动画中。
        animationDrawable.addFrame(Drawable frame, int duration);
```

14-ShapeDrawable(无标签)

41、ShapeDrawable的作用

> 1. 用于获得具有形状的 `Drawable`
> 2. 需要通过具体的 `RectShape`、`PathShape`等 赋予 `ShapeDrawable` 来获得具有对应形状的 `Drawable`
> 3. `PaintDrawable` 更通用，本身继承自 `ShapeDrawable` 。

1-RectShape

42、矩形RectShape

```
//1. 创建Shape
RectShape rectShape = new RectShape();
//2. 创建ShapeDrawable
ShapeDrawable shapeDrawable = new ShapeDrawable(rectShape);
//3. 设置颜色等内容
shapeDrawable.getPaint().setColor(Color.BLUE);
shapeDrawable.getPaint().setStyle(Paint.Style.FILL);
//4. 使用ShapeDrawable
shapeImageView.setBackground(shapeDrawable);
```

> OvalShape、ArcShape、RoundRectShape均为RectShape的子类。

OvalShape

43、椭圆形-OvalShape

```
OvalShape ovalShape = new OvalShape();
shapeDrawable.setShape(rectShape);
```

ArcShape

44、扇形-ArcShape

```
/**
 * startAngle: 初始角度-x轴正方向为0
 * sweepAngle: 顺时针方向划过的度数
 */
ArcShape arcShape = new ArcShape(0, 100);
shapeDrawable.setShape(arcShape);
```

RoundRectShape

45、圆角矩形-RoundRectShape

```
// 外部矩形弧度-外部圆角矩阵的四个角弧度(两两一组)
float[] outerR = new float[] { 8, 8, 8, 8, 8, 8, 8, 8 };
// 内部矩形与外部矩形的距离
RectF inset = new RectF(100, 100, 50, 50);
// 内部矩形弧度
float[] innerRii = new float[] { 20, 20, 20, 20, 20, 20, 20, 20 };

RoundRectShape roundRectShape = new RoundRectShape(outerR, inset, innerRii);

shapeDrawable.setShape(roundRectShape);
```

2-PathShape

46、PathShape的使用

```
Path path = new Path();
path.moveTo(50, 0);
path.lineTo(0, 50);
path.lineTo(50, 100);
path.lineTo(100, 50);
path.lineTo(50, 0);
PathShape pathShape = new PathShape(path, 200, 100);
shapeDrawable.setShape(pathShape);
```

标准宽高的作用

47、PathShape构造方法中标准宽高的作用

> 1. 如果 标准宽高 和 View的实际宽高 都相等不会出现缩放。
> 2. 如果 标准宽高 < 实际宽高 ， 会按照比例进行放大(比例=实际/标准)。
> 3. 如果 标准宽高 > 实际宽高 ， 会按照比例进行缩小(比例=实际/标准)。

3-PaintDrawable

48、PaintDrawable的使用

```
PaintDrawable paintDrawable = new PaintDrawable(Color.GREEN);
//1. 圆角，所有角的半径相同。
paintDrawable.setCornerRadius(30);
//2. 为四个角的每一个指定半径。 对于每个角落，数组包含2个值[X_radius，Y_radius].
paintDrawable.setCornerRadii (new float[]{20,20,8,8,12,12,12,12});

mShapeImageView.setBackground(paintDrawable);
```

ShapeDrawable如何与Bitmap相结合?

49、ShapeDrawable如何与Bitmap相结合?

> 1. 需要结合 BitmapShader

```
/**
 * 已经设置过ShapDrawable基础上：
 * 1. 获取Bitmap，并构造BitmapShader
 * 2. 对BitmapShader用Matrix进行缩放
 * 3. getShapeDrawable的Paint设置Shader
 */
Bitmap bitmap = ((BitmapDrawable) getResources().getDrawable(R.drawable.beauty1)).getBitmap();
BitmapShader bitmapShader = new BitmapShader(bitmap, Shader.TileMode.MIRROR, Shader.TileMode.REPEAT
Matrix matrix = new Matrix();
//根据View控件的宽高进行缩放
matrix.preScale(100f / bitmap.getWidth(),
        100f / bitmap.getHeight());//view:w=100,h=100
bitmapShader.setLocalMatrix(matrix);

shapeDrawable.getPaint().setShader(bitmapShader);
```

15-RippleDrawable(ripple)

50、RippleDrawable的作用

1. Material Design 的触摸反馈动画(波纹效果)
2. API 21(Andorid 5.0)-推出
3. 要实现波纹效果，可以直接给 控件 指定特殊的背景，也可以自定义RippleDrawable。
4. xml的标签为 ripple ， android:color= 属性指定的是波纹颜色。
5. xml中 item 标签内指定的是背景色，且多个item存在时，以最后一个为准。

51、通过背景设置波纹效果

```
//有界波纹(控件的大小)
android:background="?android:attr/selectableItemBackground"
//无界波纹(圆形波纹)
android:background="?android:attr/selectableItemBackgroundBorderless"
```

52、RippleDrawable的自定义(xml)

1、无界波纹

```
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="@color/colorPrimary">
</ripple>
```

2、有界波纹，item中直接引用drawable

```
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#f7ef07" //波纹色
    android:radius="145dp"  //波纹半径
    >
    <item android:drawable="@color/colorPrimary"/> //背景色
</ripple>
```

3、有界波纹，item中用shape标签

```
<?xml version="1.0" encoding="utf-8"?>
<ripple xmlns:android="http://schemas.android.com/apk/res/android"
    android:color="#f7ef07"//波纹色
    android:radius="145dp" //波纹半径
    >
    <item>
        <shape android:shape="rectangle">
            <solid android:color="#cc0000"/> //背景色
        </shape>
    </item>
</ripple>
```

16-AnimatedStateListDrawable

53、AnimatedStateListDrawable的作用

1. 【动画型StateListDrawable】在View状态改变时，展示动画
2. API 21(Android 5.0)-推出。
3. item 用于定义不同状态所用的 Drawable 。
4. transition 用于定义从 fromId 到 toId 之间 帧动画 的具体内容。

54、AnimatedStateListDrawable的使用

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

    <!-- 为每种状态提供不同的图片 -->
    <item android:id="@+id/pressed" android:drawable="@drawable/ic_battery_charging_full_black_24dp"
        android:state_pressed="true"/>
    <item android:id="@+id/unpressed" android:drawable="@drawable/ic_battery_charging_0_black_24dp"
        android:state_pressed="false"/>
    <item android:id="@+id/mydefault"
        android:drawable="@drawable/ic_launcher"/>

    <!-- 指定转场效果 -->
    <transition android:fromId="@+id/unpressed" android:toId="@+id/pressed">
        <animation-list>
            <item android:duration="15" android:drawable="@drawable/ic_battery_charging_20_black_24dp"/>
            <item android:duration="15" android:drawable="@drawable/ic_battery_charging_30_black_24dp"/>
            <item android:duration="15" android:drawable="@drawable/ic_battery_charging_50_black_24dp"/>
            <item android:duration="15" android:drawable="@drawable/ic_battery_charging_60_black_24dp"/>
            <item android:duration="15" android:drawable="@drawable/ic_battery_charging_80_black_24dp"/>
            <item android:duration="15" android:drawable="@drawable/ic_battery_charging_90_black_24dp"/>
            <item android:duration="15" android:drawable="@drawable/ic_battery_charging_full_black_24dp"/>
        </animation-list>
    </transition>


</animated-selector>
```

> 使用:

```
    <Button
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:background="@drawable/animated_statelist_drawable"/>
```

SVG矢量图

55、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)

56、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。

17-VectorDrawable(vector)

57、VectorDrawable的作用

> 1. 是静态矢量图(基于XML)
> 2. API 21(Android 5.0)-推出
> 3. `vector` 中 `path` 是最小单位，用于创建SVG-用指令绘制SVG图形
> 4. `vector` 中 `group` 将不同 `path` 组合起来

58、VectorDrawable的具体实例(绘制一个圆角直线)

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="200dp"
    android:height="200dp"
    android:viewportHeight="100"
    android:viewportWidth="100">
    <group>
        <path
            android:name="path1"
            android:pathData="M 20,80 L 50,80 80,80"
            android:strokeColor="@color/colorAccent"
            android:strokeLineCap="round"
            android:strokeWidth="3" />
    </group>
</vector>
```

VectorDrawable的属性

59、VectorDrawable的属性

属性	作用
android:width="200dp"	宽度
android:height="200dp"	高度
android:viewportWidth="100"	将宽度分为多少份，与path配合(50份等于100dp)
android:viewportHeight="100"	将高度200dp分为100份，50时=100dp
android:tintMode="src_in"	着色模式-参考BitmapDrawable
android:tint="@color/colorPrimary"	着色
android:name="vector"	名字
android:alpha="233"	透明度，范围0~255，0-完全可见；255-完全不可间
android:autoMirrored="false"	指示当布局方向是RTL(从右向左)时drawable是否需要镜像，默认为false

group

60、VectorDrawable的中group标签的作用

> 1. 用于将不同 `path` 组合起来

61、group的属性

属性	作用
android:name="menu_group"	名称-用于附加动画和变化效果
android:pivotX="45"	轴点X-相对于viewportWidth(宽度的份)
android:pivotY="37.5"	轴点Y-相对于viewportHeight(高度的份)
android:rotation="180"	旋转-相对于轴点，0~360
android:scaleX="0.5"	缩放-相对于轴点
android:scaleY="0.8"	缩放-相对于轴点
android:translateX="10"	平移
android:translateY="10"	平移

path

62、VectorDrawable的中path标签的作用和属性

> 1. `path` 是最小单位，用于通过指令绘制SVG矢量图。

属性	作用	取值
android:name="path1"	名称	
android:fillAlpha="255"	填充的透明度	(0~255)0-看不见；1-完全可见；255-完全不可见
android:fillColor="#ffffff"	填充的颜色	
android:fillType="evenOdd"		
android:pathData="M 45,10 L 70,35 60,35"	路径	
android:strokeAlpha="128"	路线的透明度	(0~255)
android:strokeColor="#ffffff"	路线的颜色	
android:strokeLineCap="round"	路线的线头形状	
android:strokeLineJoin="round"	路线的连接方式	
android:strokeMiterLimit="10"	斜角的上线	对应于android:strokeLineJoin=“miter”，防止斜线过长。
android:strokeWidth="2"	路线的宽度	
android:trimPathStart="0.1"	从路径起始位置截断路径的比率	0~1
android:trimPathEnd="0.8"	从路径结束位置截断路径的比率	0~1
android:trimPathOffset="0.5"	路径截断的初始值的偏移量	0~1

63、VectorDrawable的中path标签的 `pathData`

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. 是最小单位，用于通过指令绘制SVG矢量图。
> 2. 坐标轴以(0, 0)为中心， X轴水平向右， Y轴水平向下
> 3. 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
> 4. 指令和数据间空格可以省略
> 5. 同一指令出现多次，可以只用一个。
> 6. 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

18-AnimatedVectorDrawable(animated-vector)

64、AnimatedVectorDrawable的作用

> 1. 【动画矢量图】针对VectorDrawable来做动画
> 2. API 21(Android 5.0)-推出
> 3. 将 `VectorDrawable` 与 属性动画`objectAnimator`、属性动画集`set` 相关联

65、AnimatedVectorDrawable的使用实例(4步骤)

> 1、矢量图-VectorDrawable

```
//menu_vector
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="200dp"
    android:height="200dp"
    android:viewportHeight="75"
    android:viewportWidth="90">
    <group android:name="menu_group"
        android:pivotX="45"
        android:pivotY="37.5">
        <path
            android:name="path1"
            android:fillColor="#ffffff"
            android:pathData="M 20,20 L 70,20 70,25 20,25, 20,20"
            android:strokeColor="#ffffff"
            android:strokeLineJoin="round"/>

        <path
            android:name="path2"
            android:pathData="M 20,35 L 70,35 70,37.5 70,40 20,40, 20,35"
            android:fillColor="#ffffff"
            android:strokeColor="#ffffff"
            android:strokeLineJoin="round"
            />

        <path
            android:name="path3"
            android:pathData="M 20,50 L 70,50 70,55 20,55, 20,50"
            android:fillColor="#ffffff"
            android:strokeColor="#ffffff"
            android:strokeLineJoin="round"/>
    </group>

</vector>
```

> 2、属性动画(res/animator/xxx.xml)

```
//menu_topline_animator.xml-顶线变化
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="500"
    android:propertyName="pathData"
    android:valueFrom="M 20,20 L 70,20 70,25 20,25, 20,20"
    android:valueTo="M 45,10 L 70,35 60,35 40,15 45,10"
    android:valueType="pathType"
    android:interpolator="@android:interpolator/accelerate_decelerate">
</objectAnimator>

//menu_midline_animator.xml-中线变化
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="500"
    android:propertyName="pathData"
    android:valueFrom="M 20,35 L 70,35 70,37.5 70,40 20,40, 20,35"
    android:valueTo="M 20,35 L 70,35 72.5,37.5 70,40 20,40, 20,35"
    android:valueType="pathType"
    android:interpolator="@android:interpolator/accelerate_decelerate">
</objectAnimator>

//menu_bottomline_animator.xml-底线变化
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="500"
    android:propertyName="pathData"
    android:valueFrom="M 20,50 L 70,50 70,55 20,55, 20,50"
    android:valueTo="M 40,60 L 60,40 70,40 45,65 40,60"
    android:valueType="pathType"
    android:interpolator="@android:interpolator/accelerate_decelerate">
</objectAnimator>

//menu_rotate_animator.xml-整个drawble旋转的效果
<?xml version="1.0" encoding="utf-8"?>
<objectAnimator xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="500"
    android:propertyName="rotation"
    android:valueFrom="0"
    android:valueTo="180"
    android:interpolator="@android:interpolator/accelerate_decelerate">
</objectAnimator>
```

> 3、矢量图动画-AnimatedVectorDrawable:连接动画与SVG

```
//menu_animated_vector_drawable.xml
<?xml version="1.0" encoding="utf-8"?>
<animated-vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:drawable="@drawable/menu_vector">

    <target
        android:animation="@animator/menu_topline_animator"
        android:name="path1"></target>

    <target
        android:animation="@animator/menu_midline_animator"
        android:name="path2"></target>

    <target
        android:animation="@animator/menu_bottomline_animator"
        android:name="path3"></target>

    <target
        android:animation="@animator/menu_rotate_animator"
        android:name="menu_group"></target>

</animated-vector>
```

> 4、使用矢量图

```
<ImageView
    xxx
    android:src="@drawable/menu_animated_vector_drawable"
    android:background="@color/colorPrimary"/>


((Animatable)imageView.getDrawable()).start();
```

自定义Drawable

66、自定义Drawable的作用

> 1. 一般作为ImageView的图像或者作为View的背景。
> 2. 核心：实现 `draw` 方法。
> 3. `setAlpha` 、 `setColorFilter` 、 `getOpacity` 也需要重写，但是模板固定
> 4. 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
> 5. 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

67、自定义Drawable的实例

```
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
    }
}
```

实例

圆形/圆角图片的实现方法(6种)

1、Glide

2、圆形图片

> 采用圆形的图片(png等等)

3、GradientDrawable

> `<shape>` 标签中制作圆角，将该 `Drawable` 作为背景

4、自定义View

> 1. onMeasure()中测量宽高
> 2. onDraw()方法中绘制 圆形 并通过Paint和BitmapShader与 `Bitmap` 结合
> 3. 对Bitmap根据圆形大小进行缩放。

5、ShapeDrawable

> 1. OvalShape
> 2. RoundRectShape

6、PaintDrawable

> PaintDrawable设置圆角

SVG ICON

菜单：箭头

确定与取消

微博点赞

参考资料

1. Drawable基础与自定义Drawable
2. Icon动画技术

矢量图相关网站

1. 在线制作
2. SVG转VectorDrawable
3. Picture No White
4. PNG2SVG
5. PNG转SVG-质量更高