链滴

浅谈 Android 主题样式

作者：xuexiangjys

原文链接：https://ld246.com/article/1678550915417

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

浅谈Android主题样式

文章末尾有附带例子的源码链接, 感兴趣的可以下载源码研究, 味道更佳.

在讲Android主题之前, 让我们先回顾一下Android中自定义View的实现方法.

自定义View

完全自定义View实现自定义控件

自定义View、ViewGroup或者SurfaceView：

- 自定义View：主要重写onDraw（绘制）方法。 自定义View实现例子
- 自定义ViewGroup：主要重写：onMeasure（测量）、onLayout（布局）这两个方法。 自定义ViewGroup实现例子
- 自定义SurfaceView：创建RenderThread，然后调用 SurfaceHolder的.lockCanvas方法获取画，再调用 SurfaceHolder的.unlockCanvasAndPost方法将绘制的画布投射到屏幕上。

```
class CustomSurfaceView @JvmOverloads constructor(
    context: Context,
    attrs: AttributeSet? = null,
) : SurfaceView(context, attrs), SurfaceHolder.Callback {

    private var mSurfaceHolder: SurfaceHolder = holder
    private lateinit var mRenderThread: RenderThread
    private var mIsDrawing = false

    override fun surfaceChanged(holder: SurfaceHolder, format: Int, width: Int, height: Int) {}
    override fun surfaceCreated(holder: SurfaceHolder) {
        // 开启RenderThread
        mIsDrawing = true
        mRenderThread = RenderThread()
        mRenderThread.start()
    }

    override fun surfaceDestroyed(holder: SurfaceHolder) {
        // 销毁RenderThread
        mIsDrawing = false
        mRenderThread.interrupt()
    }

    /**
     * 绘制界面的线程
     */
    private inner class RenderThread : Thread() {

        override fun run() {
            // 不停绘制界面
            while (mIsDrawing) {
                drawUI()
```

```
                try {
                    sleep(...) // 刷新间隔
                } catch (_: InterruptedException) {
                }
            }
        }
    }

    /**
     * 界面绘制
     */
    private fun drawUI() {
        val canvas = mSurfaceHolder.lockCanvas()
        try {
            drawCanvas(canvas)
        } catch (e: Exception) {
            e.printStackTrace()
        } finally {
            mSurfaceHolder.unlockCanvasAndPost(canvas)
        }
    }
}
```

自定义SurfaceView实现例子

继承组件的方式实现自定义控件

最简单的自定义组件的方式，直接继承需要拓展/修改的控件，重写对应的方法即可。

一般是希望在原有系统控件基础上做一些修饰性的修改（功能增强），而不会做大幅度的改动。

继承组件实现例子

组合的方式实现自定义控件

组合控件就是将多个控件组合成一个新的控件，可以重复使用。

实现组合控件的一般步骤如下：

- 编写布局文件
- 实现构造方法
- 初始化UI，加载布局
- 对外提供修改的接口api

可以看到，组合的方式和我们平时写一个Fragment的流程是很类似的。

组合组件实现例子

Theme主题

应用于窗体级别，是一整套样式的组合，采取就近原则：Application > Activity > ViewGroup > Vi

w。 一般而言，Theme主要应用于Application和Activity这样的窗体，主要放在 /res/values/themes xml。

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

Application中的Theme

Application的主题一般在 Manifest中，它只对在 Manifest中未设置Theme的Activity生效。

```
<application android:theme="@style/AppTheme">

</application>
```

Activity中的Theme

Activity的主题可以在 Manifest和代码中调用 setTheme设置。一般在Activity的onCreate()中，setC ntentView方法之前设置。

1.在 Manifest中设置。

```
<activity android:theme="@style/DialogTheme">

</activity>
```

2.代码中调用 setTheme设置，注意一定要在调用 setContentView(View)和 inflate(int, ViewGroup) 法前。

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setTheme(R.style.AppTheme)
    setContentView(R.layout.layout_main)
}
```

ViewGroup和View中的Theme

ViewGroup和View的主题一般在布局xml中设置，使用 android:theme设置。

```
<ViewGroup
    android:theme="@style/ThemeOverlay.App.Foo">

    <Button android:theme="@style/ThemeOverlay.App.Bar" />

</ViewGroup>
```

Style样式

仅应用于单个View这种窗体元素级别的外观，主要放在 /res/values/styles.xml。

Style的声明

样式的声明，一般放在 /res/values/...目录下带 styles的文件中，使用 <style name="style-name"></style>进行设置。

```
<style name="style-name" parent="parent-style-name">
    <item name="attr-name1">value1</item>
    <item name="attr-name2">value2</item>
    <item name="attr-name3">value3</item>
</style>
```

Style的使用

样式一般在布局xml中设置，使用 android:style设置，不同于主题，样式只能应用于单个View，对其子View并不会生效。

```
<ViewGroup
    android:style="@style/ActionContainerStyle">

    <Button android:style="@style/BlueButtonStyle" />

</ViewGroup>
```

Style的优先级顺序

如果我们在多个地方给控件指定了style的属性，那么最终是由谁生效呢？这里我们就以TextView为，介绍一下Style的生效规则：

- 1.通过文本span将字符设置的样式应用到TextView派生的类。
- 2.以代码方式动态设置的属性。
- 3.将单独的属性直接应用到View。
- 4.将样式应用到View。
- 5.控件的默认样式，在View构造方法中定义的。
- 6.控件所处应用、Activity、父布局所应用的主题。
- 7.应用某些特定于View的样式，例如为TextView设置TextAppearance。

具体代码可参考: StyleRuleFragment

Attribute属性

Attribute属性是组成Style的基本单位。如果说主题是各种样式的组合，那么样式就是各种属性的组，主要放在 /res/values/attrs.xml。

Attribute的声明

1.单个属性的定义

```
<resource>

    <attr name="attr-name" format="format-type" />

</resource>
```

2.一组属性的定义

```
<resource>

    <declare-styleable name="XXXXView">
        <attr name="attr-name" format="format-type" />
        <attr name="attr-name" format="format-type" />
    </declare-styleable>

</resource>
```

3.属性的赋值

```
<style name="xx">

  <item name="attr-name">value</item>

</style>
```

Attribute的使用

使用 ?attr/xxx或者 ?xxx进行引用。这里xxx是定义的属性名（attr-name）。

```
<TextView
    android:foreground="?attr/selectableItemBackground"
    android:textColor="?colorAccent" />
```

Attribute的获取

- 属性集的获取: 使用 context.obtainStyledAttributes进行整体获取。

```
val array = context.obtainStyledAttributes(attrs, R.styleable.CustomTextView, defStyleAttr, def
tyleRes)
size = array.getInteger(R.styleable.CustomTextView_ctv_size, size)
isPassword = array.getBoolean(R.styleable.CustomTextView_ctv_is_password, isPassword)
array.recycle()
```

- 单个属性的获取: 使用 context.theme.resolveAttribute进行获取。

```
fun Resources.Theme.resolveAttributeToDimension(@AttrRes attributeId: Int, defaultValue: Fl
at = 0F) : Float {
    val typedValue = TypedValue()
    return if (resolveAttribute(attributeId, typedValue, true)) {
        typedValue.getDimension(resources.displayMetrics)
    } else {
        defaultValue
    }
```

```
}

fun Context.resolveDimension(@AttrRes attributeId: Int, defaultValue: Float = 0F) : Float {
    val typedArray = theme.obtainStyledAttributes(intArrayOf(attributeId))
    return try {
        typedArray.getDimension(0, defaultValue)
    } finally {
        typedArray.recycle()
    }
}
```

最后

以上内容的全部源码我都放在了github上, 感兴趣的小伙伴可以下下来研究和学习.

项目地址: https://github.com/xuexiangjys/UIThemeSample

我是xuexiangjys，一枚热爱学习，爱好编程，勤于思考，致力于Android架构研究以及开源项目经分享的技术up主。获取更多资讯，欢迎微信搜索公众号：**【我的Android开源之旅】**