分析手段

Frameworks 的资源文件拷贝到settings，里面的原理

Frameworks

# TASK

Android Theme详细

https://blog.csdn.net/kc58236582/article/details/50469581

https://blog.csdn.net/yanbober/article/details/51015630

https://blog.csdn.net/wenzhi20102321/article/details/65631474

Android开发之Theme、Style探索及源码浅

https://blog.csdn.net/yanbober/article/details/51015630

Overlay

https://my.oschina.net/kingguary/blog/160190

https://www.jianshu.com/p/9304089c513d

# framework-res

## Android.mk 分析

### **特点**

**# 1.支持应用程序编译的资源能够被其它应用使用**

**# 2.该APK具有系统权限**

**# 3.能够指定一个全局的依赖于LOCAL\_BUILT\_MODULE的中间目标**

路径：./frameworks/base/core/res/Android.mk

LOCAL\_NO\_STANDARD\_LIBRARIES := true

## framework/base/core/res/res/values/public.xml

symbols.xml 和 public.xml的作用是一样的。  
区别：  
symbols.xml定义的资源进入到com.android.internal.R.java（私有资源）,  
public.xml定义的资源进入到android.R.java（公有资源），  
并不是严格定义，在一个xml文件里也可以通过不同的标签来控制资源是公共的还是私有的。

public.xml中有两种类型的资源描述，一种是<java-symbol/>系统私有的资源（不对app公开），另一种是<public />公共资源（可以在app中访问）。

如果需要添加一个资源：

如果要添加的资源是私有的，则加上 <java-symbol type="drawable" name="test" />（不需要id），如果是公共的则先找到最后一个type为 drawable 的 public 项，4.1.2系统中最后一项为<public type="drawable" name="dialog\_holo\_light\_frame" id="0x010800b3" />，然后添加一句 <public type="drawable" name="test" id="0x010800b4" />，即id为最后一个的id+1（为了避免 id 冲突）

如果添加的是公共资源，则还需要在源码目录下执行 make update-api 以更新 frameworks/base/api/current.txt 文件。

mm，编译后生成framework-res.apk 并会更新com.android.internal.R，

在 frameworks/base/core/res/ 中执行mm，编译后生成framework-res.apk 并会更新com.android.internal.R，该 R 所对应的文件为 /out/target/common/R/com/android/internal/R.java，可以在该文件中查看自己的资源是否已经加进去了。（如果mm出错可以尝试先执行一下“touch 资源文件”命令）

另外，如果要在系统中添加一个新的id，比如为myid，则上述第一步改为在 frameworks/base/core/res/res/values/ids.xml 中增加一列<item type="id" name="myid" />，之后的步骤都一样

## 为系统增加图片资源

在WindowManagerService.java中使用一个图片资源pic.png.顺序如下.

1,将文件pic.png拷贝到位置:frameworks/base/core/res/res/drawable下.

2,在frameworks/base/core/res/res/drawable目录下执行touch pic.png.

3,进入目录frameworks/base/core/res/ 执行mm命令, 编译 framework-res.apk

4,执行完后com.android.internal.R 会新生成一个R.drawable.pic的引用.在程序中使用即可.

5,在目录frameworks/base/ 下执行mm 编译 framework.jar.

6,在frameworks/base/services/java/com/android/server 下的WindowManagerService.java中使用com.android.internal.R.drawable.pic,使用完后保存文件.

7,进入目录frameworks/base/services/java/ 执行mm 编译  services.jar

8,替换机器上(虚拟机或者真机)的jar apk文件,编译出的包在out/target/product/xxxx/system/framework下

  adb push framework-res.apk /system/framework/

  adb push services.jar /system/framework/

  adb push framework.jar /system/framework/

  执行命令时注意framework-res.apk 的真实路径.

9,adb reboot 机器,查看修改结果.大功告成!

## themes\_device\_defaults

themes\_holo

themes\_leanback

themes\_material

styles\_device\_defaults

## 性能优化

<array name="preloaded\_drawables"> 是系统预加载

# Android-Theme

关键字：Android 主题Theme style activity和dialog的样式

## 基本概念

Attr:属性,风格样式的最小单元;

Style：风格,它是一系列Attr的集合用以定义一个View的样式,比如height、width、padding等.主要是用来定义内部元素(说白了就是内部控件)的样式。

Theme：主题,它与Style作用一样,不同于Style作用于个一个单独View,而它是作用于Activity上或是整个应用。是用来定义activity和dialog的样式,还可以定义内部元素的样式(这样一般使用style)。

关系：从Theme定义在style.xml中我们就可以看出,Theme的本质其实也是Style。

### 举例

#### TextView布局

Sdfg

其中layout\_width对应到框架中的attr信息如下:

<declare-styleable name="ViewGroup\_Layout">

<attr name="layout\_width" format="dimension">

<enum name="fill\_parent" value="-1" />

<enum name="match\_parent" value="-1" />

<enum name="wrap\_content" value="-2" />

</attr>

...

</declare-styleable>

从上可以看到layout\_width可以使用三个枚举值,并且其中fill\_parent和match\_parent的value值都为-1。做过Android开发的童鞋肯定知道,从2.2开始Android框架就推荐用match\_parent代替fill\_parent,而以上代码正实现了兼容,因为它们对应的值都为-1。

以上的textStyle的属性信息在源码中如下:

<attr name="textStyle">

<flag name="normal" value="0" />

<flag name="bold" value="1" />

<flag name="italic" value="2" />

</attr>

它也对应了三个值,但这里却使用了flag标签。细心的童鞋可能已经明白了flag与enum的差别,flag表示这几个值可以做或运算,比如上面的textStyle,你可以叠加使用,如用bold|italic表示既加粗也变成斜体,而enum只能让你选择其中一个值。

试着自己自定义一个自己的属性,在values目录下创建一个attrs.xml文件,在<resources>元素里面首先申明一个自己的<declare-styleable>表示一个属性组,再在里面加上属性就行。如下我们定义一个DogStyle的属性组,其中有三个属性一个是dogSex,一个是dogName,dogName的格式我们设置为string,最后一个是dogColor,这样一个属于我们自己的属性就定义成功了。

DogStyle attr的format根据字面意思也挺容易理解的,这里我解释下reference的用法。它用在一些可以设置引用值的情况,比如@drawable/myImage、@color/myColor等。当然format也可以进行或运算,一般我们定义color类型的属性时,也一般会把format写成format="reference|color"

TIPS:format即使用错,只要你自定义的View中获取对应类型值也是可以的,只是在布局中写代码时,IDE就不会根据你定义的format给出相应的提示了,所以最好在自定义View时还是仔细斟酌下类型。

Style的使用

如下我们在styles.xml中定义一个雪纳瑞风格

<style name="SchnauzerStyle">

<item name="dogName">雪纳瑞</item>

<item name="dogColor">@drawable/schnauzer</item>

<item name="dogSex">boy</item>

</style>

下面我们看下如何让一个Style作用在一个View上的。

首先我们自定义了一个View命名为DogView,然后创建一个布局文件中加入该DogView视图,并让该View使用SchnauzerStyle风格。代码如下:

<cn.hadcn.test.DogView

android:layout\_height="wrap\_content"

android:layout\_width="wrap\_content"/>

移步到DogView的Java代码中,我们可以通过theme的obtainStyledAttributes方法来获得我们刚刚定义的几个Attr属性在Style中的内容,如下我们举一个获得dogName的例子:

final Resources.Theme theme = context.getTheme();

TypedArray dogArray = theme.obtainStyledAttributes(attrs, R.styleable.DogStyle, defStyleAttr, defStyleRes);

String name = dogArray.getString(R.styleable.DogStyle\_dogName);

Log.e("dog", "name = " + name);

dogArray.recycle();

以上obtainStyledAttributes有四个入参,前两个比较容易理解,后两个用作指定默认的Style,表示如果attrs中没有你想获得的属性,但如果你指定了默认Style,它会去从该默认的Style里面找你想要的属性。defStyleAttr和defStyleRes功能一样,指定的资源形式不同,前者表示一个默认的指向一个style风格的attr属性,而后者你可以直接传入一个style风格的id。注意以上定义的Style只能在这个DogView中被使用,如果你想在其他View使用,就需要再在需要使用的View中增加这个Style。这就是先前我们说的Style只能作用于一个View。

#### 1、使用Theme

阿里云 > 教程中心 > android教程 > Android里面的Attr、Style和Theme 解析

Android里面的Attr、Style和Theme 解析

发布时间：2018-01-18 来源：网络 上传者：用户

关键字: 解析 面的 Android theme style attr

发表文章

摘要：前言这三个概念贯穿Android框架的方方面面,是Android程序设计中很重要的一环,理解它们,并能学以致用,不但可以让你的代码变得简洁明了,还可以让你的应用更加灵活。但目前网上资料对这块介绍的知识点往往比较散,不是很系统全面,在此总结此文一篇。参考资料:Android开发之Theme、Style探索及源码浅析概念说明Attr:属性,风格样式的最小单元;Style:风格,它是一系列Attr的集合用以定义一个View的样式,比如height、width、padding等;Th

前言

这三个概念贯穿Android框架的方方面面,是Android程序设计中很重要的一环,理解它们,并能学以致用,不但可以让你的代码变得简洁明了,还可以让你的应用更加灵活。但目前网上资料对这块介绍的知识点往往比较散,不是很系统全面,在此总结此文一篇。

参考资料:Android 开发之 Theme、Style 探索及源码浅析

概念说明

Attr:属性,风格样式的最小单元;

Style:风格,它是一系列Attr的集合用以定义一个View的样式,比如height、width、padding等;

Theme:主题,它与Style作用一样,不同于Style作用于个一个单独View,而它是作用于Activity上或是整个应用。

Attr的定义

我们先举一个框架中的源码例子,用来介绍下Android中是如何定义一个Attr的,比如以下创建一个简单的TextView布局

TextView,其中layout\_width对应到框架中的attr信息如下:

<declare-styleable name="ViewGroup\_Layout">

<attr name="layout\_width" format="dimension">

<enum name="fill\_parent" value="-1" />

<enum name="match\_parent" value="-1" />

<enum name="wrap\_content" value="-2" />

</attr>

...

</declare-styleable>

从上可以看到layout\_width可以使用三个枚举值,并且其中fill\_parent和match\_parent的value值都为-1。做过Android开发的童鞋肯定知道,从2.2开始Android框架就推荐用match\_parent代替fill\_parent,而以上代码正实现了兼容,因为它们对应的值都为-1。

以上的textStyle的属性信息在源码中如下:

<attr name="textStyle">

<flag name="normal" value="0" />

<flag name="bold" value="1" />

<flag name="italic" value="2" />

</attr>

它也对应了三个值,但这里却使用了flag标签。细心的童鞋可能已经明白了flag与enum的差别,flag表示这几个值可以做或运算,比如上面的textStyle,你可以叠加使用,如用bold|italic表示既加粗也变成斜体,而enum只能让你选择其中一个值。

看完上例后,我们来试着自己自定义一个自己的属性,在values目录下创建一个attrs.xml文件,在<resources>元素里面首先申明一个自己的<declare-styleable>表示一个属性组,再在里面加上属性就行。如下我们定义一个DogStyle的属性组,其中有三个属性一个是dogSex,一个是dogName,dogName的格式我们设置为string,最后一个是dogColor,这样一个属于我们自己的属性就定义成功了。

DogStyle attr的format根据字面意思也挺容易理解的,这里我解释下reference的用法。它用在一些可以设置引用值的情况,比如@drawable/myImage、@color/myColor等。当然format也可以进行或运算,一般我们定义color类型的属性时,也一般会把format写成format="reference|color"

TIPS:format即使用错,只要你自定义的View中获取对应类型值也是可以的,只是在布局中写代码时,IDE就不会根据你定义的format给出相应的提示了,所以最好在自定义View时还是仔细斟酌下类型。

Style的使用

如下我们在styles.xml中定义一个雪纳瑞风格

<style name="SchnauzerStyle">

<item name="dogName">雪纳瑞</item>

<item name="dogColor">@drawable/schnauzer</item>

<item name="dogSex">boy</item>

</style>

下面我们看下如何让一个Style作用在一个View上的。

首先我们自定义了一个View命名为DogView,然后创建一个布局文件中加入该DogView视图,并让该View使用SchnauzerStyle风格。代码如下:

<cn.hadcn.test.DogView

android:layout\_height="wrap\_content"

android:layout\_width="wrap\_content"/>

移步到DogView的Java代码中,我们可以通过theme的obtainStyledAttributes方法来获得我们刚刚定义的几个Attr属性在Style中的内容,如下我们举一个获得dogName的例子:

final Resources.Theme theme = context.getTheme();

TypedArray dogArray = theme.obtainStyledAttributes(attrs, R.styleable.DogStyle, defStyleAttr, defStyleRes);

String name = dogArray.getString(R.styleable.DogStyle\_dogName);

Log.e("dog", "name = " + name);

dogArray.recycle();

以上obtainStyledAttributes有四个入参,前两个比较容易理解,后两个用作指定默认的Style,表示如果attrs中没有你想获得的属性,但如果你指定了默认Style,它会去从该默认的Style里面找你想要的属性。defStyleAttr和defStyleRes功能一样,指定的资源形式不同,前者表示一个默认的指向一个style风格的attr属性,而后者你可以直接传入一个style风格的id。注意以上定义的Style只能在这个DogView中被使用,如果你想在其他View使用,就需要再在需要使用的View中增加这个Style。这就是先前我们说的Style只能作用于一个View。

Theme的使用

Theme与Style使用同一个元素标签区别在于所包含的属性不同,并且使用的地方也不一样。Theme你需要设置到AndroidManifest.xml的<application>或者<activity>标签下,设置后,被设置的Activity或整个应用下所有的View都可以使用《style》里面的属性了。

比如在上例中,我们直接把SchnauzerStyle设置到<activity>标签中,并把布局文件中DogView元素的style="@style/SchnauzerStyle"栏位删除,以此来测试下,这个Activity下的所有View是不是可以直接使用theme中声明的这些属性。

<activity

android:name=".MainActivity"

android:theme="@style/SchnauzerStyle">

...

以上理论上是可行的,不过运行后,程序却出现奔溃,出现以下错误提示:

java.lang.IllegalStateException: You need to use a Theme.AppCompat theme (or descendant) with this activity.

有些同学一眼可能就看出,因为在这里Activity或Application的需要很多属性才能工作的,而此处我们只给它传一个SchnauzerStyle,这当然不行,所以我们需要对这个Style做下处理,让SchnauzerStyle继承一个系统主题,如下:

<style name="SchnauzerStyle" parent="Theme.AppCompat">

<item name="dogName">雪纳瑞</item>

<item name="dogColor">@drawable/schnauzer</item>

<item name="dogSex">boy</item>

</style>

这样一个雪纳瑞主题就诞生了,而在这个Activity下的所有View都可以用雪纳瑞的信息了。Application中定义theme的原理一样,这里就不多说了。

TIPS:框架使用Attr的顺序是:View中的Style会优先于Activity中的Theme,Activity中的Theme会优先于Application中的Theme,所以说你可以定义整个应用的总体风格,但局部风格你也可以做出自己的调整。

在Manifest.xml文件中application节点下android:theme=""属性就是给我们添加了一个默认的主题,表示我们整个app都建使用这个主题。



在style.xml文件中我们可以找到这个主题的定义:

1、colorPrimary表示actionBar的颜色

2、coorPrimaryDark表示状态栏的颜色

3、colorAccent是定义重要的颜色,如EditText的下划线,和光标



#### Attr的获得方法

有些情况下,我们可能需要使用theme中的属性值,比如下面我们想让一个TextView直接显示dogName这个属性的内容,并且使用系统字体的颜色,则可以如下做:

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="?android:textColorSecondary"

android:text="?attr/dogName"/>

获得一个Attr的方法,不同于普通资源使用@符号获得的方式,而是需要使用?符号来获得属性,整体的表达方式如下:

?[<package\_name>:][<resource\_type>/]<resource\_name>

如果是本应用中的attr使用,则可以省去<package\_name>部分。

此处的textColor使用当前主题的android:textColorSecondary属性内容。因为资源工具知道此处是一个属性,所以省去了attr (完整写法:?android:attr/textColorSecondary)。

#### 2、使用style

首相我们在style.xml文件中新建一个style,名字叫TextStyle,属性如下:



然后我们再布局文件中将这个样式给一个textview使用,然后我们再运行一下,看一下效果。





布局文件中并无给textview添加任何属性,但是我们给textview添加了一个style,然后我们再看看我们所定义的TextStyle样式,item中所定义的属性,是不是就是我们平时在布局文件中给控件添加的属性了?对!style中所能定义的属性,就是控件能支持的属性,至于能定义一些什么属性,可以到api中进行详细的查看,

### Theme和Style使用

Style的定义很简单,想要给控件增加那些效果,然后查看对应api是否支持,在将需要定义的属性放到item中进行定义就好

支持父级的

Theme和Style的兼容性

建立一个名为valus-v19的文件夹,建立style.xml文件,添加名字相同的theme,然后这里定义sdk为19以上版本的theme

### 追根溯源Theme

建立的Theme都有一个父级,那我们我们就一步一步的往上找,找到Theme的根源。

我们翻找AppThem的父级可以发现,我们可以看到父级的里面也是一层一层的套着的





在往上找几层我们可以找到Theme.Light 这个Style,可以发现到这里就没有Theme父了,而是Style地继承,者也很好的说明Theme的本质就是Style,然后我们发现这里面定义了许多的item。

然后我们继续往上找一层,发现了Theme的老巢了!Theme!这个里面定义了几百个各种各样的属性,然后我们仔细的看看这些属性,不都是一些控件的属性吗? 对的,Theme里面就是定义了各种控件,窗口(Dialog、Activity)等的样式属性。我们平时用到的Theme.NoActionBar等以及其他一些Theme都是对Theme的扩展、修改。



## 属性值



Theme编辑器界面:

接下来说明每一个颜色控制器,控制控件哪些显示效果:

colorPrimary:toolbar

colorPrimaryDark:statusbar

colorAccent:seekbar、progressbar、edittext底线、Radiobutton选中状态、Checkbox选选中状态、Swith选中状态

android:colorBackgroud:activity背景色、fragment背景色

android:colorForeground:FrameLayout前景色、与foregroundTintMode结合使用

android:navigationBarColor:虚拟键背景色

android:statusBarColor:状态栏背景色

android:textColorPrimary:控制button、checkedTextView、flatButton、toggleButton的enable和false状态颜色,以及toolbar中tittle字体颜色,左侧和右侧按钮颜色

android:textColorPrimaryInverse:不知道控制什么的

android:textColorSecondary:checkBox、radioButton选择框默认颜色,progressbar底色,SeekBar底色,RatingBar默认颜色

android:textColorSeoundaryInverser:不知道控制什么

windowBackground:控制activity和fragment背景色

## android常用Style的Theme汇总

fdsg

•android:theme="@android:style/Theme.Light" 背景为白色

•android:theme="@android:style/Theme.Light.NoTitleBar" 白色背景并无标题栏

•android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen" 白色背景,无标题栏,全屏

•android:theme="@android:style/Theme.Black" 背景黑色

•android:theme="@android:style/Theme.Black.NoTitleBar" 黑色背景并无标题栏

•android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen" 黑色背景,无标题栏,全屏

### 三种字体大小

?android:attr/textAppearanceLarge"

"?android:attr/textAppearanceMedium"

"?android:attr/textAppearanceSmall"

### Android字体颜色:

ndroid:textColor="?android:attr/textColorPrimary"

android:textColor="?android:attr/textColorSecondary"

android:textColor="?android:attr/textColorTertiary"

android:textColor="?android:attr/textColorPrimaryInverse"

android:textColor="?android:attr/textColorSecondaryInverse"

### Android的ProgressBar样式:

style="?android:attr/progressBarStyleHorizontal"

style="?android:attr/progressBarStyleLarge"

style="?android:attr/progressBarStyleSmall"

style="?android:attr/progressBarStyleSmallTitle"

### 分隔符

Dfgh

横向:

android:layout\_width="fill\_parent"

android:layout\_height="1dip"

android:background="?android:attr/listDivider" />

纵向:

android:layout\_height="fill\_parent"

android:background="?android:attr/listDivider" />

### CheckBox样式

style="?android:attr/starStyle"

### 类似标题栏效果的TextView

style="?android:attr/listSeparatorTextViewStyle"

### 其它有用的样式

android:layout\_height="?android:attr/listPreferredItemHeight"

android:paddingRight="?android:attr/scrollbarSize"

style="?android:attr/windowTitleBackgroundStyle"

style="?android:attr/windowTitleStyle"

android:layout\_height="?android:attr/windowTitleSize"

android:background="?android:attr/windowBackground"

## Theme与Style加载原理浅浅析

之前我们说过我们设置Theme可以在Manifest.xml文件中定义,还能在java代码中设置使用setTheme(这个我没在实际代码中用过,而且必须是在setContentView()方法之前设置),而android初始化界面的入口,是setContentView(),那么我们就从这里开始着手浅析。

现在我们写activity时一般是继承AppComatActivity,这个是用来进行版本适配的,最近继承的还是Activity,所以我们直接来看Activity中的setContentView()方法。

### 流程分析



### 核心处理

Ddfgh

@Override

public Resources.Theme getTheme() {

if (mTheme != null) {

return mTheme;

}

// 这里是选择默认的主题资源

mThemeResource = Resources.selectDefaultTheme(mThemeResource,

getApplicationInfo().targetSdkVersion);

initializeTheme();

return mTheme;

}

// 设置Theme,通过调用此方法然后mTheme和mThemeResource就不为null了,然后再初始化题。

// 当我们在Manifest.xml 中配置了them,在应用程序启动时会调用此方法,具体的需要去查找android启动activity的过程。

@Override

public void setTheme(int resid) {

if (mThemeResource != resid) {

mThemeResource = resid;

initializeTheme();

}

}

如主题不为空就直接返回主题,然后先选择一个默认的主题资源。我们也可以进去看看,没有什么特别的,就是对不同的sdk选择对应的主题资源。这里我们主要看initializeTheme()初始化主题。

private void initializeTheme() {

final boolean first = mTheme == null;

if (first) {

mTheme = getResources().newTheme();//调用方法获取主题,就是new了一个Theme。

final Resources.Theme theme = getBaseContext().getTheme();//调用ContextImpl类的getTheme(),获取默认的Theme

if (theme != null) {

mTheme.setTo(theme);//将获取到的默认的主题设置给mTheme

}

}

onApplyThemeResource(mTheme, mThemeResource, first);//应用主题资源

}

主题为空时,new一个Theme,然后调用ContextImpl类的getTheme(),获取默认的Theme ,然后将得到的Theme设置给mTheme,然后通过onApplyThemeResource(),应用主题资源,最后是AsserManger中调用原生的方法(c 或 c++)去加载资源

然后我们再回到getWindowStyle()方法中,这里对用了obtainStyledAttributes()方法,并且传进去com.android.internal.R.styleable.Window这是一组自定义属性集合,在android内置的系统资源中,里面定义者window的基本属性属性。而这个方法最终是通过AssetManager获取这一组自定义的属性集合,得到一个TypeArry以供使用,去设置界面窗口的样式。

<declare-styleable name="Window">

<attr name="windowBackground" /> //该界面所对应的背景图片, drawable / color

<attr name="windowFrame" /> //该界面所对应的前景frontground图片, drawable / color

<attr name="windowNoTitle" /> //是否带有title , boolean类型

<attr name="windowFullscreen" /> //是否全屏 , boolean类型

<attr name="windowIsFloating" /> //是否是悬浮窗类型 , boolean类型

<attr name="windowIsTranslucent" /> //是否透明 , boolean类型

<attr name="windowSoftInputMode" /> //设置键盘弹出来的样式 , 例如: adjustsize 等 ,其实也是int类型

<......>

<......>

<declare-styleable />

其中的而在Theme节点中有包含着window的定义,特殊的是如果某个自定义属性如果没有指名 format属性,那么该属性必须在当前已经定义,即该属性只是一个别名。而我们自定义的Theme则必须是要指定一个parent字段,我们所自定义的也只是对系统所定义的做一个修改与扩展。

<declare-styleable name="Theme">

<attr name="windowBackground" format="reference" />

<!-- Drawable to use as a frame around the window. -->

<attr name="windowFrame" format="reference" />

<!-- Flag indicating whether there should be no title on this window. -->

<attr name="windowNoTitle" format="boolean" />

<!-- Flag indicating whether this window should fill the entire screen. -->

<attr name="windowFullscreen" format="boolean" />

<!-- Flag indicating whether this is a floating window. -->

<attr name="windowIsFloating" format="boolean" />

<!-- Flag indicating whether this is a translucent window. -->

<attr name="windowIsTranslucent" format="boolean" />

<!-- Flag indicating that this window's background should be the

user's current wallpaper. -->

<attr name="windowShowWallpaper" format="boolean" />

<!-- This Drawable is overlaid over the foreground of the Window's content area, usually

to place a shadow below the title. -->

<!-- This Drawable is overlaid over the foreground of the Window's content area, usually

to place a shadow below the title. -->

<attr name="windowContentOverlay" format="reference" />

<!--more -->

</declare-styleable>

知道Theme的使用与原理了,那么对于Style则就能够理解了,在控件初始化时获取style属性,回去style定义,获取TypeArry对象给控件进行设置,然后控件绘制是就会依据TypeArry里面获取的属性来绘制控件,从而实现样式效果。

## REF

[android常用Style的Theme汇总](https://www.aliyun.com/jiaocheng/4050.html?spm=5176.100033.1.26.BJY6Em)

# 系统主题引擎

关键字：系统应用 系统设置 android7.1 settings 主题引擎 系统主题定制

## 概述

需求：

### 行业现状

芯片公司：无

手机厂商：hw/小米/酷派， ？

## Material design的主题颜色控制

ColorPrimary，定义应用的主色调

colorPrimaryDark，定义按钮点击后的颜色（由于状态使用半透着色实现渐变效果，该属性在原先主题中已失效）

colorBackground，定义标题栏渐变色的结束颜色



### tint特性

tint是Material Design中的新特性，可以理解为着色，把一张图片渲染成另外一种颜色。在Android L中，EditText的下划线背景和光标颜色变化，就是使用这种特性。

### 定义主题theme

## REF

[Android 党福利，Android 8.0 将可无需 Root 改变系统主题](http://www.ifanr.com/898217)

[substratum](https://github.com/substratum/substratum)

[Android Oreo将在1周内获得无需root主题支持](http://t.cj.sina.com.cn/articles/view/6420358328/17eaee4b8001004c5f)

GIT 源码

<http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2017/0502/7901.html>

<http://yifeng.studio/2017/04/18/android-theme-appcompat-color-attrs/>

<http://www.ifanr.com/898217>

<https://blog.csdn.net/maetelibom/article/details/54956877>

<https://blog.csdn.net/archer_zoro/article/details/56304498>

# 系统应用规范

SDK:专门团队在整，我们只是做了个精简版的freamework

系统应用 android studio

系统设置 android studio：无

## 使用android系统自带资源

### 系统定义的id："@android:id/list"

这里我们定义了一个ListView，定义它的id是"@+id/mylist"。实际上，如果没有特别的需求，就可以利用系统定义的id，类似下面的样子。

|  |
| --- |
| <ListView      android:id="@android:id/list"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"/> |

在xml文件中引用系统的id，只需要加上“@android:”前缀即可。如果是在Java代码中使用系统资源，和使用自己的资源基本上是一样的。不同的是，需要使用android.R类来使用系统的资源，而不是使用应用程序指定的R类。这里如果要获取ListView可以使用android.R.id.list来获取。

android.R.id

resource android:id/seekbar is private.

则定义为：<**SeekBar android:id="@\*android:id/seekbar"**

com.android.internal.R.id.***seekbar***

### 系统的图片资源"@\*android:drawable/ic\_menu\_attachment"

假设我们在应用程序中定义了一个menu，xml文件如下。

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <menu xmlns:android="http://schemas.android.com/apk/res/android">      <item          android:id="@+id/menu\_attachment"          android:title="附件"          android:icon="@android:drawable/ic\_menu\_attachment" />  </menu> |

其中代码片段android:icon="@android:drawable/ic\_menu\_attachment"本来是想引用系统中已有的Menu里的“附件”的图标。但是在Build工程以后，发现出现了错误。提示信息如下：

|  |
| --- |
| error: Error: Resource is not public. (at 'icon' with value '@android:drawable/ic\_menu\_attachment'). |

从错误的提示信息大概可以看出，由于该资源没有被公开，所以无法在我们的应用中直接引用。既然这样的话，我们就可以在Android SDK中找到相应的图片资源，直接拷贝到我们的工程目录中，然后使用类似android:icon="@drawable/ic\_menu\_attachment"的代码片段进行引用。

这样做的好处，一个是美工不需要重复的做一份已有的图片了，可以节约不少工时；另一个是能保证我们的应用程序的风格与系统一致。

|  |
| --- |
| 经验分享：  Android中没有公开的资源，在xml中直接引用会报错。除了去找到对应资源并拷贝到我们自己的应用目录下使用以外，我们还可以将引用“@android”改成“@\*android”解决。比如上面引用的附件图标，可以修改成下面的代码。  android:icon="@\*android:drawable/ic\_menu\_attachment"  修改后，再次Build工程，就不会报错了。 |

### 系统的字符串资源"@android:string/yes"

 假设我们要实现一个Dialog，Dialog上面有“确定”和“取消”按钮。就可以使用下面的代码直接使用Android系统自带的字符串。

|  |
| --- |
| <LinearLayout          android:orientation="horizontal"          android:layout\_width="fill\_parent"          android:layout\_height="wrap\_content">          <Button              android:id="@+id/yes"              android:layout\_width="fill\_parent"              android:layout\_height="wrap\_content"              android:layout\_weight="1.0"              android:text="@android:string/yes"/>          <Button              android:id="@+id/no"              android:layout\_width="fill\_parent"              android:layout\_height="wrap\_content"              android:layout\_weight="1.0"              android:text="@android:string/no"/>      </LinearLayout> |

如果使用系统的字符串，默认就已经支持多语言环境了。如上述代码，直接使用了@android:string/yes和@android:string/no，在简体中文环境下会显示“确定”和“取消”，在英文环境下会显示“OK”和“Cancel”。

### 系统的Style："?android:attr/textAppearanceMedium"

假设布局文件中有一个TextView，用来显示窗口的标题，使用中等大小字体。可以使用下面的代码片段来定义TextView的Style。

|  |
| --- |
| <TextView          android:id="@+id/title"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:textAppearance="?android:attr/textAppearanceMedium" /> |

其中android:textAppearance="?android:attr/textAppearanceMed ium"就是使用系统的style。需要注意的是，使用系统的style，需要在想要使用的资源前面加“?android:”作为前缀，而不是“@android:”。

### 利用系统的颜色定义@android:color/transparent

Sdfa

除了上述的各种系统资源以外，还可以使用系统定义好的颜色。在项目中最常用的，就是透明色的使用。代码片段如下。

|  |
| --- |
| android:background ="@android:color/transparent" |

|  |
| --- |
| 经验分享：  Android系统本身有很多资源在应用中都可以直接使用，具体的，可以进入android-sdk的相应文件夹中去查看。例如：可以进入$android-sdk$\platforms\android-8\data\res，里面的系统资源就一览无余了。  开发者需要花一些时间去熟悉这些资源，特别是图片资源和各种Style资源，这样在开发过程中，能够想到有相关资源并且直接拿来使用。 |

**?android:attr/colorAccent**