分析手段

Frameworks 的资源文件拷贝到settings，里面的原理

Frameworks

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

# TASK

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

# Theme基本概念

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

每家公司或者每个移动团队无不想开发出一套自己的UI框架，融入自己的设计和特性，这必然会去修改android的ui。

所以，学习和理解android的UI设计是最基础和非常有必要的。

android ui设计最重要的就是主题和样式。

## 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

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

themes.xml偏全局，styles.xml偏细化。

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

### 继承定义规则

一般在style中使用parent字段的继承适用于继承系统平台现有定义的style，而我们想要继承自己实现的style一般不会通过parent字段来实现，而是通过指定格式的name字段来实现，如下：

<style name="CodeFont.Red">

<item name="android:textColor">#FF0000</item>

</style>

看见name字段了吗？证明这个style继承自我们上面自定义的CodeFont style，可见我们自定义的继承是通过“.”来实现的，在使用时只需要@style/CodeFont.Red即可使用该继承重写的style啦，如果你还想继续在这基础上继承，那写法还是一样的，具体如下：

<style name="CodeFont.Red.Big">

<item name="android:textSize">30sp</item>

</style>

整明白Theme与Style的这个约定了么！就是这么简单而已，不过要注意这两种的严格区别，别乱用，譬如将系统预定义的通过name来使用时错误的。

接着我们来看看style中item属性是怎么搞来的，这玩意如果我们定义控件的style则可以直接在对应控件或者Window的API文档中或者R.attr文档中找到支持哪些属性，依次选择合适的进行使用即可（**特别提醒，这个技能很重要，譬如有时候你会说我自定义的Dialog为何背景周边多一个黑框啥玩意的问题，然后上网一顿复制别人的style，也不明白别人为啥这么写，其实一个很重要的技巧就是遇到这种问题自己去API查下相关的属性就搞定了。**）。使用系统已存在属性时切记不要忘记<item name="android:inputType">前面的android:前缀，还有就是item中存在的属性不见得对所有View都有效，譬如Theme中需要的以windowXXX开头的属性就不适用于View，但是不会报错，只是View会忽略这些不适合自己的属性，应用适合自己的属性。

应用层开发难就难在知识面很广

* 当我们想要知道Theme具体有哪些属性可以有效使用时，可以查阅API的R.styleable进行选择。
* 当我们想要知道Style具体有哪些属性可以有效使用时，可以查阅API的R.attr进行选择。
* 系统为我们提供了很多实用的Theme与Style，我们还可以通过查阅API的R.style进行选择（要注意的是这里的文档查到的不一定全，最好的办法是去查FW下base的res或者appcompat的res），不过要注意，在API中譬如Theme\_NoTitleBar主题样式在我们xml引用时要替换为@android:style/Theme.NoTitleBar的格式。

## 属性值作用域

作用域：

当个view,activity,application

优先级：属性值，sytle引用，系统默认主题



## 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。

### 系统如何去选择默认的主题呢？

在Android的frameworks/base/core/res/res/values目录下有一下几个文件：

|  |  |
| --- | --- |
| 1  2  3  4 | themes.xml  themes\_device\_defaults.xml  styles.xml  styles\_device\_defaults.xml |

分别定义了各种系统Theme，Style。

主要关注themes.xml,themes\_device\_defaults.xml两个文件。

themes.xml定义了android低版本的theme和Holo theme，themes\_device\_defaults.xml定义了DeviceDefault主题（继承自Holo主题），实际上就是在Holo主题上定制主题（For厂商）。

/\*\*frameworks/base/core/java/android/content/res/Resources.java\*/

/\*\* @hide \*/

public static int selectDefaultTheme(int curTheme, int targetSdkVersion) {

return selectSystemTheme(curTheme, targetSdkVersion,

com.android.internal.R.style.Theme,

com.android.internal.R.style.Theme\_Holo,

com.android.internal.R.style.Theme\_DeviceDefault);

}

/\*\* @hide \*/

public static int selectSystemTheme(int curTheme, int targetSdkVersion,

int orig, int holo, int deviceDefault) {

if (curTheme != 0) {

return curTheme;

}

if (targetSdkVersion < Build.VERSION\_CODES.HONEYCOMB) {

// < 11

return orig;

}

if (targetSdkVersion < Build.VERSION\_CODES.ICE\_CREAM\_SANDWICH) {

// < 14

return holo;

}

return deviceDefault;

}

当<11时，使用以前低版本主题;当>=11&&<14，使用Holo主题;>14的时候，使用DeviceDefault主题。

### 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。



## REF

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

# Theme详述

SDK目录：\android-sdk\platforms\android-16\data\res\values\

源码目录：framework/base/core/res/res/values/

## 家族图

Holo表示整天的意思

下面是一些android自带的属性（可以直接在Application或Activity中设置）：

android:theme="@android:style/Theme.Dialog" : Activity显示为对话框模式

android:theme="@android:style/Theme.NoTitleBar" : 不显示应用程序标题栏

android:theme="@android:style/Theme.NoTitleBar.Fullscreen" : 不显示应用程序标题栏，并全屏

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

android:theme="@android:style/Theme.Wallpaper" : 用系统桌面为应用程序背景

android:theme="@android:style/Theme.Wallpaper.NoTitleBar" : 用系统桌面为应用程序背景，且无标题栏

android:theme="@android:style/Theme.Wallpaper.NoTitleBar.Fullscreen" : 用系统桌面为应用程序背景，无标题栏，全屏

android:theme="@android:style/Theme.Translucent : 透明背景

android:theme="@android:style/Theme.Translucent.NoTitleBar" : 透明背景并无标题

android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen" : 透明背景并无标题，全屏

android:theme="@android:style/Theme.Panel ": 面板风格显示

android:theme="@android:style/Theme.Light.Panel" : 平板风格显示



当我们继承使用时只用在前加上android:即可，有些属性可能是找不到的

## 属性值K约定

其中style的name是可以自己定义的，需要拿来调用，而item的name是固定的，是android中已经定义好的。你会发现所有item的name都是android：下的！

一个sytle中 可以有一个或多个item的属性的设置。

一个完整的主题应该定义哪些内容呢，以Theme为例，如下：

到这里我们就很容易明白啦，Theme.Light的父类原来是Theme哇，也在这个文件中，如下：

<!-- The default theme for apps on API level 10 and lower. This is the theme used for

activities that have not explicitly set their own theme.

<p>You can count on this being a dark

background with light text on top, but should try to make no

other assumptions about its appearance. In particular, the text

inside of widgets using this theme may be completely different,

with the widget container being a light color and the text on top

of it a dark color.

<p>If you're developing for API level 11 and higher, you should instead use {@link

#Theme\_Holo} or {@link #Theme\_DeviceDefault}.</p>

-->

<style name="Theme">

......

<!-- Text styles -->

......

<!-- Button styles -->

......

<!-- List attributes -->

......

<!-- @hide -->

......

<!-- Gallery attributes -->

......

<!-- Window attributes -->

......

<!-- Define these here; ContextThemeWrappers around themes that define them should

always clear these values. -->

......

<!-- Dialog attributes -->

......

<!-- AlertDialog attributes -->

......

<!-- Presentation attributes (introduced after API level 10 so does not

have a special old-style theme. -->

......

<!-- Toast attributes -->

......

<!-- Panel attributes -->

......

<!-- These three attributes do not seems to be used by the framework. Declared public though -->

......

<!-- Scrollbar attributes -->

......

<!-- Text selection handle attributes -->

......

<!-- Widget styles -->

......

<!-- Preference styles -->

......

<!-- Search widget styles -->

......

<!-- Action bar styles -->

......

<!-- Floating toolbar styles -->

......

<!-- SearchView attributes -->

......

<!-- PreferenceFrameLayout attributes -->

......

<!-- NumberPicker style-->

......

<!-- CalendarView style-->

......

<!-- TimePicker style -->

......

<!-- TimePicker dialog theme -->

......

<!-- DatePicker style -->

......

<!-- DatePicker dialog theme -->

......

<!-- Pointer style -->

......

<!-- Accessibility focused drawable -->

......

<!-- Lighting and shadow properties -->

......

</style>

看注释吧，

### Color

Sdf

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. **<item** name="colorForeground"**>**@android:color/bright\_foreground\_dark**</item>**
3. **<item** name="colorForegroundInverse"**>**@android:color/bright\_foreground\_dark\_inverse**</item>**
5. **<item** name="colorBackground"**>**@android:color/background\_dark**</item>**
7. **<item** name="colorBackgroundCacheHint"**>**?android:attr/colorBackground**</item>**
9. **<item** name="colorPressedHighlight"**>**@color/legacy\_pressed\_highlight**</item>**
11. **<item** name="colorLongPressedHighlight"**>**@color/legacy\_long\_pressed\_highlight**</item>**
13. **<item** name="colorFocusedHighlight"**>**@color/legacy\_selected\_highlight**</item>**
15. **<item** name="colorMultiSelectHighlight"**>**@color/legacy\_selected\_highlight**</item>**
17. **<item** name="colorActivatedHighlight"**>**@color/legacy\_selected\_highlight**</item>**

### Text

Fgs’

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Text styles -->
2. android:textSize 设置字体大小
3. android:textColor 设置字体颜色
4. 设置文字的外观
5. **<item** name="textAppearance"**>**@android:style/TextAppearance**</item>**
7. **<item** name="textAppearanceInverse"**>**@android:style/TextAppearance.Inverse**</item>**
9. **<item** name="textColorPrimary"**>**@android:color/primary\_text\_dark**</item>**
11. **<item** name="textColorSecondary"**>**@android:color/secondary\_text\_dark**</item>**
13. **<item** name="textColorTertiary"**>**@android:color/tertiary\_text\_dark**</item>**
15. **<item** name="textColorPrimaryInverse"**>**@android:color/primary\_text\_light**</item>**
17. **<item** name="textColorSecondaryInverse"**>**@android:color/secondary\_text\_light**</item>**
19. **<item** name="textColorTertiaryInverse"**>**@android:color/tertiary\_text\_light**</item>**
21. **<item** name="textColorPrimaryDisableOnly"**>**@android:color/primary\_text\_dark\_disable\_only**</item>**
23. **<item** name="textColorPrimaryInverseDisableOnly"**>**@android:color/primary\_text\_light\_disable\_only**</item>**
25. **<item** name="textColorPrimaryNoDisable"**>**@android:color/primary\_text\_dark\_nodisable**</item>**
27. **<item** name="textColorSecondaryNoDisable"**>**@android:color/secondary\_text\_dark\_nodisable**</item>**
29. **<item** name="textColorPrimaryInverseNoDisable"**>**@android:color/primary\_text\_light\_nodisable**</item>**
31. **<item** name="textColorSecondaryInverseNoDisable"**>**@android:color/secondary\_text\_light\_nodisable**</item>**
32. 设置提示信息文字的颜色，与hint一起使用
33. **<item** name="textColorHint"**>**@android:color/hint\_foreground\_dark**</item>**
35. **<item** name="textColorHintInverse"**>**@android:color/hint\_foreground\_light**</item>**
37. **<item** name="textColorSearchUrl"**>**@android:color/search\_url\_text**</item>**
38. 被选中文字的底色
39. **<item** name="textColorHighlight"**>**@android:color/highlighted\_text\_dark**</item>**
41. **<item** name="textColorHighlightInverse"**>**@android:color/highlighted\_text\_light**</item>**
42. 文字链接的颜色
43. **<item** name="textColorLink"**>**@android:color/link\_text\_dark**</item>**
45. **<item** name="textColorLinkInverse"**>**@android:color/link\_text\_light**</item>**
47. **<item** name="textColorAlertDialogListItem"**>**@android:color/primary\_text\_light\_disable\_only**</item>**


51. **<item** name="textAppearanceLarge"**>**@android:style/TextAppearance.Large**</item>**
53. **<item** name="textAppearanceMedium"**>**@android:style/TextAppearance.Medium**</item>**
55. **<item** name="textAppearanceSmall"**>**@android:style/TextAppearance.Small**</item>**
57. **<item** name="textAppearanceLargeInverse"**>**@android:style/TextAppearance.Large.Inverse**</item>**
59. **<item** name="textAppearanceMediumInverse"**>**@android:style/TextAppearance.Medium.Inverse**</item>**
61. **<item** name="textAppearanceSmallInverse"**>**@android:style/TextAppearance.Small.Inverse**</item>**
63. **<item** name="textAppearanceSearchResultTitle"**>**@android:style/TextAppearance.SearchResult.Title**</item>**
65. **<item** name="textAppearanceSearchResultSubtitle"**>**@android:style/TextAppearance.SearchResult.Subtitle**</item>**


69. **<item** name="textAppearanceEasyCorrectSuggestion"**>**@android:style/TextAppearance.EasyCorrectSuggestion**</item>**
71. **<item** name="textAppearanceMisspelledSuggestion"**>**@android:style/TextAppearance.MisspelledSuggestion**</item>**
73. **<item** name="textAppearanceAutoCorrectionSuggestion"**>**@android:style/TextAppearance.AutoCorrectionSuggestion**</item>**


77. **<item** name="textAppearanceButton"**>**@android:style/TextAppearance.Widget.Button**</item>**


81. **<item** name="editTextColor"**>**@android:color/primary\_text\_light**</item>**
83. **<item** name="editTextBackground"**>**@android:drawable/edit\_text**</item>**


87. **<item** name="candidatesTextStyleSpans"**>**@android:string/candidates\_style**</item>**


91. **<item** name="textCheckMark"**>**@android:drawable/indicator\_check\_mark\_dark**</item>**
93. **<item** name="textCheckMarkInverse"**>**@android:drawable/indicator\_check\_mark\_light**</item>**


97. **<item** name="textAppearanceLargePopupMenu"**>**@android:style/TextAppearance.Widget.PopupMenu.Large**</item>**
99. **<item** name="textAppearanceSmallPopupMenu"**>**@android:style/TextAppearance.Widget.PopupMenu.Small**</item>**

#### 三种字体大小

?android:attr/textAppearanceLarge"

"?android:attr/textAppearanceMedium"

"?android:attr/textAppearanceSmall"

#### Android字体颜色:

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

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

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

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

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

#### 文字选中（Text Selection）

SDAF

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Text selection handle attributes -->
3. **<item** name="textSelectHandleLeft"**>**@android:drawable/text\_select\_handle\_left**</item>**
5. **<item** name="textSelectHandleRight"**>**@android:drawable/text\_select\_handle\_right**</item>**
7. **<item** name="textSelectHandle"**>**@android:drawable/text\_select\_handle\_middle**</item>**
9. **<item** name="textSelectHandleWindowStyle"**>**@android:style/Widget.TextSelectHandle**</item>**
11. **<item** name="textEditPasteWindowLayout"**>**@android:layout/text\_edit\_paste\_window**</item>**
13. **<item** name="textEditNoPasteWindowLayout"**>**@android:layout/text\_edit\_no\_paste\_window**</item>**
15. **<item** name="textEditSidePasteWindowLayout"**>**@android:layout/text\_edit\_side\_paste\_window**</item>**
17. **<item** name="textEditSideNoPasteWindowLayout"**>**@android:layout/text\_edit\_side\_no\_paste\_window**</item>**
19. **<item** name="textSuggestionsWindowStyle"**>**@android:style/Widget.TextSuggestionsPopupWindow**</item>**
21. **<item** name="textEditSuggestionItemLayout"**>**@android:layout/text\_edit\_suggestion\_item**</item>**
23. **<item** name="textCursorDrawable"**>**@null**</item>**

#### 其他

android:textFilterEnabled 设为真时，列表会过滤根据用户的要求，过滤结果集。列表的适配器必须实现了 Filterable接口，才能使其可用

android:textOff 未选中时按钮的文本

android:textOn 选中时按钮的文本

android:textStyle 字体，bold, italic, bolditalic

android:textScaleX控制字与字之间的间距

### CheckBox样式

style="?android:attr/starStyle"

### Button

Fgh

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Button styles -->
3. **<item** name="buttonStyle"**>**@android:style/Widget.Button**</item>**


7. **<item** name="buttonStyleSmall"**>**@android:style/Widget.Button.Small**</item>**
9. **<item** name="buttonStyleInset"**>**@android:style/Widget.Button.Inset**</item>**


13. **<item** name="buttonStyleToggle"**>**@android:style/Widget.Button.Toggle**</item>**


17. **<item** name="selectableItemBackground"**>**@android:drawable/item\_background**</item>**
19. **<item** name="borderlessButtonStyle"**>**?android:attr/buttonStyle**</item>**
21. **<item** name="homeAsUpIndicator"**>**@android:drawable/ic\_ab\_back\_holo\_dark**</item>**

### List

Sdfg

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- List attributes -->
3. **<item** name="listPreferredItemHeight"**>**64dip**</item>**
5. **<item** name="listPreferredItemHeightSmall"**>**?android:attr/listPreferredItemHeight**</item>**
7. **<item** name="listPreferredItemHeightLarge"**>**?android:attr/listPreferredItemHeight**</item>**
9. **<item** name="dropdownListPreferredItemHeight"**>**?android:attr/listPreferredItemHeight**</item>**
11. **<item** name="textAppearanceListItem"**>**?android:attr/textAppearanceLarge**</item>**
13. **<item** name="textAppearanceListItemSmall"**>**?android:attr/textAppearanceLarge**</item>**
15. **<item** name="listPreferredItemPaddingLeft"**>**6dip**</item>**
17. **<item** name="listPreferredItemPaddingRight"**>**6dip**</item>**
19. **<item** name="listPreferredItemPaddingStart"**>**6dip**</item>**
21. **<item** name="listPreferredItemPaddingEnd"**>**6dip**</item>**

、

#### 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" />

### Window

Saedf

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Window attributes -->
3. **<item** name="windowBackground"**>**@android:drawable/screen\_background\_selector\_dark**</item>**
4. 设置有没有标题栏true|false
5. **<item** name="windowFrame"**>**@null**</item>**
7. **<item** name="windowNoTitle"**>**false**</item>**
8. 设置全屏true|false
9. **<item** name="windowFullscreen"**>**false**</item>**
11. **<item** name="windowOverscan"**>**false**</item>**
12. 设置是否浮现在activity之上true|false
13. **<item** name="windowIsFloating"**>**false**</item>**
15. **<item** name="windowContentOverlay"**>**@null**</item>**
17. **<item** name="windowShowWallpaper"**>**false**</item>**
19. **<item** name="windowTitleStyle"**>**@android:style/WindowTitle**</item>**
21. **<item** name="windowTitleSize"**>**25dip**</item>**
23. **<item** name="windowTitleBackgroundStyle"**>**@android:style/WindowTitleBackground**</item>**
25. **<item** name="android:windowAnimationStyle"**>**@android:style/Animation.Activity**</item>**
27. **<item** name="android:windowSoftInputMode"**>**stateUnspecified|adjustUnspecified**</item>**
29. **<item** name="windowActionBar"**>**false**</item>**
31. **<item** name="windowActionModeOverlay"**>**false**</item>**
33. **<item** name="windowCloseOnTouchOutside"**>**false**</item>**
35. **<item** name="windowTranslucentStatus"**>**false**</item>**
37. **<item** name="windowTranslucentNavigation"**>**false**</item>**

windowIsTranslucent设置window是否为透明

android:windowContentOverlay这个是定义contentoverlay的背景的

### Dialog

Asdf

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Dialog attributes -->
3. **<item** name="dialogTheme"**>**@android:style/Theme.Dialog**</item>**
5. **<item** name="dialogTitleIconsDecorLayout"**>**@layout/dialog\_title\_icons**</item>**
7. **<item** name="dialogCustomTitleDecorLayout"**>**@layout/dialog\_custom\_title**</item>**
9. **<item** name="dialogTitleDecorLayout"**>**@layout/dialog\_title**</item>**

### AlertDialog

Dj

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- AlertDialog attributes -->
3. **<item** name="alertDialogTheme"**>**@android:style/Theme.Dialog.Alert**</item>**
5. **<item** name="alertDialogStyle"**>**@android:style/AlertDialog**</item>**
7. **<item** name="alertDialogCenterButtons"**>**true**</item>**
9. **<item** name="alertDialogIcon"**>**@android:drawable/ic\_dialog\_alert**</item>**

### Panel

Dfgs

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Panel attributes -->
3. **<item** name="panelBackground"**>**@android:drawable/menu\_background**</item>**
5. **<item** name="panelFullBackground"**>**@android:drawable/menu\_background\_fill\_parent\_width**</item>**
7. <!-- These three attributes do not seems to be used by the framework. Declared public though -->
9. **<item** name="panelColorBackground"**>**#000**</item>**
11. **<item** name="panelColorForeground"**>**?android:attr/textColorPrimary**</item>**
13. **<item** name="panelTextAppearance"**>**?android:attr/textAppearance**</item>**


17. **<item** name="panelMenuIsCompact"**>**false**</item>**
19. **<item** name="panelMenuListWidth"**>**296dip**</item>**

### scroll滚动条

SAD

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Scrollbar attributes -->
3. **<item** name="scrollbarFadeDuration"**>**250**</item>**
5. **<item** name="scrollbarDefaultDelayBeforeFade"**>**300**</item>**
7. **<item** name="scrollbarSize"**>**10dip**</item>**
9. **<item** name="scrollbarThumbHorizontal"**>**@android:drawable/scrollbar\_handle\_horizontal**</item>**
11. **<item** name="scrollbarThumbVertical"**>**@android:drawable/scrollbar\_handle\_vertical**</item>**
13. **<item** name="scrollbarTrackHorizontal"**>**@null**</item>**
15. **<item** name="scrollbarTrackVertical"**>**@null**</item>**

#### Android的ProgressBar样式:

style="?android:attr/progressBarStyleHorizontal"

style="?android:attr/progressBarStyleLarge"

style="?android:attr/progressBarStyleSmall"

style="?android:attr/progressBarStyleSmallTitle"

### Widget样式

ASDF

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Widget styles -->
2. **<item** name="absListViewStyle"**>**@android:style/Widget.AbsListView**</item>**
3. **<item** name="autoCompleteTextViewStyle"**>**@android:style/Widget.AutoCompleteTextView**</item>**
4. **<item** name="checkboxStyle"**>**@android:style/Widget.CompoundButton.CheckBox**</item>**
5. **<item** name="checkedTextViewStyle"**>**@android:style/Widget.CheckedTextView**</item>**
6. **<item** name="dropDownListViewStyle"**>**@android:style/Widget.ListView.DropDown**</item>**
7. **<item** name="editTextStyle"**>**@android:style/Widget.EditText**</item>**
8. **<item** name="expandableListViewStyle"**>**@android:style/Widget.ExpandableListView**</item>**
9. **<item** name="expandableListViewWhiteStyle"**>**@android:style/Widget.ExpandableListView.White**</item>**
10. **<item** name="galleryStyle"**>**@android:style/Widget.Gallery**</item>**
11. **<item** name="gestureOverlayViewStyle"**>**@android:style/Widget.GestureOverlayView**</item>**
12. **<item** name="gridViewStyle"**>**@android:style/Widget.GridView**</item>**
13. **<item** name="imageButtonStyle"**>**@android:style/Widget.ImageButton**</item>**
14. **<item** name="imageWellStyle"**>**@android:style/Widget.ImageWell**</item>**
15. **<item** name="listViewStyle"**>**@android:style/Widget.ListView**</item>**
16. **<item** name="listViewWhiteStyle"**>**@android:style/Widget.ListView.White**</item>**
17. **<item** name="popupWindowStyle"**>**@android:style/Widget.PopupWindow**</item>**
18. **<item** name="progressBarStyle"**>**@android:style/Widget.ProgressBar**</item>**
19. **<item** name="progressBarStyleHorizontal"**>**@android:style/Widget.ProgressBar.Horizontal**</item>**
20. **<item** name="progressBarStyleSmall"**>**@android:style/Widget.ProgressBar.Small**</item>**
21. **<item** name="progressBarStyleSmallTitle"**>**@android:style/Widget.ProgressBar.Small.Title**</item>**
22. **<item** name="progressBarStyleLarge"**>**@android:style/Widget.ProgressBar.Large**</item>**
23. **<item** name="progressBarStyleInverse"**>**@android:style/Widget.ProgressBar.Inverse**</item>**
24. **<item** name="progressBarStyleSmallInverse"**>**@android:style/Widget.ProgressBar.Small.Inverse**</item>**
25. **<item** name="progressBarStyleLargeInverse"**>**@android:style/Widget.ProgressBar.Large.Inverse**</item>**
26. **<item** name="seekBarStyle"**>**@android:style/Widget.SeekBar**</item>**
27. **<item** name="ratingBarStyle"**>**@android:style/Widget.RatingBar**</item>**
28. **<item** name="ratingBarStyleIndicator"**>**@android:style/Widget.RatingBar.Indicator**</item>**
29. **<item** name="ratingBarStyleSmall"**>**@android:style/Widget.RatingBar.Small**</item>**
30. **<item** name="radioButtonStyle"**>**@android:style/Widget.CompoundButton.RadioButton**</item>**
31. **<item** name="scrollViewStyle"**>**@android:style/Widget.ScrollView**</item>**
32. **<item** name="horizontalScrollViewStyle"**>**@android:style/Widget.HorizontalScrollView**</item>**
33. **<item** name="spinnerStyle"**>**@android:style/Widget.Spinner**</item>**
34. **<item** name="dropDownSpinnerStyle"**>**@android:style/Widget.Spinner.DropDown**</item>**
35. **<item** name="starStyle"**>**@android:style/Widget.CompoundButton.Star**</item>**
36. **<item** name="tabWidgetStyle"**>**@android:style/Widget.TabWidget**</item>**
37. **<item** name="textViewStyle"**>**@android:style/Widget.TextView**</item>**
38. **<item** name="errorMessageBackground"**>**@android:drawable/popup\_inline\_error**</item>**
39. **<item** name="errorMessageAboveBackground"**>**@android:drawable/popup\_inline\_error\_above**</item>**
40. **<item** name="webTextViewStyle"**>**@android:style/Widget.WebTextView**</item>**
41. **<item** name="webViewStyle"**>**@android:style/Widget.WebView**</item>**
42. **<item** name="dropDownItemStyle"**>**@android:style/Widget.DropDownItem**</item>**
43. **<item** name="spinnerDropDownItemStyle"**>**@android:style/Widget.DropDownItem.Spinner**</item>**
44. **<item** name="spinnerItemStyle"**>**@android:style/Widget.TextView.SpinnerItem**</item>**
45. **<item** name="dropDownHintAppearance"**>**@android:style/TextAppearance.Widget.DropDownHint**</item>**
46. **<item** name="keyboardViewStyle"**>**@android:style/Widget.KeyboardView**</item>**
47. **<item** name="quickContactBadgeOverlay"**>**@android:drawable/quickcontact\_badge\_overlay\_dark**</item>**
48. **<item** name="quickContactBadgeStyleWindowSmall"**>**@android:style/Widget.QuickContactBadge.WindowSmall**</item>**
49. **<item** name="quickContactBadgeStyleWindowMedium"**>**@android:style/Widget.QuickContactBadge.WindowMedium**</item>**
50. **<item** name="quickContactBadgeStyleWindowLarge"**>**@android:style/Widget.QuickContactBadge.WindowLarge**</item>**
51. **<item** name="quickContactBadgeStyleSmallWindowSmall"**>**@android:style/Widget.QuickContactBadgeSmall.WindowSmall**</item>**
52. **<item** name="quickContactBadgeStyleSmallWindowMedium"**>**@android:style/Widget.QuickContactBadgeSmall.WindowMedium**</item>**
53. **<item** name="quickContactBadgeStyleSmallWindowLarge"**>**@android:style/Widget.QuickContactBadgeSmall.WindowLarge**</item>**
54. **<item** name="listPopupWindowStyle"**>**@android:style/Widget.ListPopupWindow**</item>**
55. **<item** name="popupMenuStyle"**>**@android:style/Widget.PopupMenu**</item>**
56. **<item** name="activityChooserViewStyle"**>**@android:style/Widget.ActivityChooserView**</item>**
58. **<item** name="mediaRouteButtonStyle"**>**@android:style/Widget.DeviceDefault.MediaRouteButton**</item>**

SDFG

### Preference样式（Preference Style）

SDFAZ

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Preference styles -->
2. **<item** name="preferenceScreenStyle"**>**@android:style/Preference.PreferenceScreen**</item>**
3. **<item** name="preferenceFragmentStyle"**>**@style/PreferenceFragment**</item>**
4. **<item** name="preferenceCategoryStyle"**>**@android:style/Preference.Category**</item>**
5. **<item** name="preferenceStyle"**>**@android:style/Preference**</item>**
6. **<item** name="preferenceInformationStyle"**>**@android:style/Preference.Information**</item>**
7. **<item** name="checkBoxPreferenceStyle"**>**@android:style/Preference.CheckBoxPreference**</item>**
8. **<item** name="switchPreferenceStyle"**>**@android:style/Preference.SwitchPreference**</item>**
9. **<item** name="yesNoPreferenceStyle"**>**@android:style/Preference.DialogPreference.YesNoPreference**</item>**
10. **<item** name="dialogPreferenceStyle"**>**@android:style/Preference.DialogPreference**</item>**
11. **<item** name="editTextPreferenceStyle"**>**@android:style/Preference.DialogPreference.EditTextPreference**</item>**
12. **<item** name="ringtonePreferenceStyle"**>**@android:style/Preference.RingtonePreference**</item>**
13. **<item** name="preferenceLayoutChild"**>**@android:layout/preference\_child**</item>**
14. **<item** name="preferencePanelStyle"**>**@style/PreferencePanel**</item>**
15. **<item** name="preferenceHeaderPanelStyle"**>**@style/PreferenceHeaderPanel**</item>**
16. **<item** name="preferenceListStyle"**>**@style/PreferenceHeaderList**</item>**
17. **<item** name="preferenceFragmentListStyle"**>**@style/PreferenceFragmentList**</item>**
18. **<item** name="preferenceFragmentPaddingSide"**>**@dimen/preference\_fragment\_padding\_side**</item>**
19. **<item** name="detailsElementBackground"**>**@android:drawable/panel\_bg\_holo\_dark**</item>**

SDC

### search控件样式

ASDF

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Search widget styles -->
2. **<item** name="searchWidgetCorpusItemBackground"**>**@android:color/search\_widget\_corpus\_item\_background**</item>**
3. <!-- SearchView attributes -->
4. **<item** name="searchDropdownBackground"**>**@android:drawable/spinner\_dropdown\_background**</item>**
5. **<item** name="searchViewTextField"**>**@drawable/textfield\_searchview\_holo\_dark**</item><item** name="searchViewTextFieldRight"**>**@drawable/textfield\_searchview\_right\_holo\_dark**</item>**
6. **<item** name="searchViewCloseIcon"**>**@android:drawable/ic\_clear**</item>**
7. **<item** name="searchViewSearchIcon"**>**@android:drawable/ic\_search**</item>**
8. **<item** name="searchViewGoIcon"**>**@android:drawable/ic\_go**</item>**
9. **<item** name="searchViewVoiceIcon"**>**@android:drawable/ic\_voice\_search**</item>**
10. **<item** name="searchViewEditQuery"**>**@android:drawable/ic\_commit\_search\_api\_holo\_dark**</item>**
11. **<item** name="searchViewEditQueryBackground"**>**?attr/selectableItemBackground**</item>**
13. **<item** name="searchDialogTheme"**>**@style/Theme.SearchBar**</item>**

### ActionBar样式（ Action bar Style）

DFG

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- Action bar styles -->
2. **<item** name="actionDropDownStyle"**>**@android:style/Widget.Spinner.DropDown**</item>**
3. **<item** name="actionButtonStyle"**>**@android:style/Widget.ActionButton**</item>**
4. **<item** name="actionOverflowButtonStyle"**>**@android:style/Widget.ActionButton.Overflow**</item>**
5. **<item** name="actionModeBackground"**>**@android:drawable/cab\_background\_top\_holo\_dark**</item>**
6. **<item** name="actionModeSplitBackground"**>**@null**</item>**
7. **<item** name="actionModeCloseDrawable"**>**@android:drawable/ic\_menu\_close\_clear\_cancel**</item>**
8. **<item** name="actionModeCutDrawable"**>**@android:drawable/ic\_menu\_cut\_holo\_dark**</item>**
9. **<item** name="actionModeCopyDrawable"**>**@android:drawable/ic\_menu\_copy\_holo\_dark**</item>**
10. **<item** name="actionModePasteDrawable"**>**@android:drawable/ic\_menu\_paste\_holo\_dark**</item>**
11. **<item** name="actionModeSelectAllDrawable"**>**@android:drawable/ic\_menu\_selectall\_holo\_dark**</item>**
12. **<item** name="actionModeShareDrawable"**>**@android:drawable/ic\_menu\_share\_holo\_dark**</item>**
13. **<item** name="actionModeFindDrawable"**>**@android:drawable/ic\_menu\_find\_holo\_dark**</item>**
14. **<item** name="actionModeWebSearchDrawable"**>**@android:drawable/ic\_menu\_search**</item>**
15. **<item** name="actionBarTabStyle"**>**@style/Widget.ActionBar.TabView**</item>**
16. **<item** name="actionBarTabBarStyle"**>**@style/Widget.ActionBar.TabBar**</item>**
17. **<item** name="actionBarTabTextStyle"**>**@style/Widget.ActionBar.TabText**</item>**
18. **<item** name="actionModeStyle"**>**@style/Widget.ActionMode**</item>**
19. **<item** name="actionModeCloseButtonStyle"**>**@style/Widget.ActionButton.CloseMode**</item>**
20. **<item** name="actionBarStyle"**>**@android:style/Widget.ActionBar**</item>**
21. **<item** name="actionBarSplitStyle"**>**?android:attr/actionBarStyle**</item>**
22. **<item** name="actionBarSize"**>**@dimen/action\_bar\_default\_height**</item>**
23. **<item** name="actionModePopupWindowStyle"**>**?android:attr/popupWindowStyle**</item>**
24. **<item** name="actionMenuTextAppearance"**>**@android:style/TextAppearance.Holo.Widget.ActionBar.Menu**</item>**
25. **<item** name="actionMenuTextColor"**>**?android:attr/textColorPrimary**</item>**
26. **<item** name="actionBarWidgetTheme"**>**@null**</item>**
27. **<item** name="actionBarDivider"**>**?android:attr/dividerVertical**</item>**
28. **<item** name="actionBarItemBackground"**>**?android:attr/selectableItemBackground**</item>**
30. **<item** name="dividerVertical"**>**@drawable/divider\_vertical\_dark**</item>**
31. **<item** name="dividerHorizontal"**>**@drawable/divider\_vertical\_dark**</item>**
32. **<item** name="buttonBarStyle"**>**@android:style/ButtonBar**</item>**
33. **<item** name="buttonBarButtonStyle"**>**?android:attr/buttonStyle**</item>**
34. **<item** name="segmentedButtonStyle"**>**@android:style/SegmentedButton**</item>**

### 其他样式

DF

**[html]** [view plain](https://blog.csdn.net/kc58236582/article/details/50469581) [copy](https://blog.csdn.net/kc58236582/article/details/50469581)

1. <!-- PreferenceFrameLayout attributes -->
2. **<item** name="preferenceFrameLayoutStyle"**>**@android:style/Widget.PreferenceFrameLayout**</item>**
4. <!-- NumberPicker style-->
5. **<item** name="numberPickerStyle"**>**@style/Widget.NumberPicker**</item>**
7. <!-- CalendarView style-->
8. **<item** name="calendarViewStyle"**>**@style/Widget.CalendarView**</item>**
10. <!-- TimePicker style -->
11. **<item** name="timePickerStyle"**>**@style/Widget.TimePicker**</item>**
13. <!-- DatePicker style -->
14. **<item** name="datePickerStyle"**>**@style/Widget.DatePicker**</item>**
16. **<item** name="fastScrollThumbDrawable"**>**@android:drawable/scrollbar\_handle\_accelerated\_anim2**</item>**
17. **<item** name="fastScrollTrackDrawable"**>**@null**</item>**
18. **<item** name="fastScrollPreviewBackgroundRight"**>**@android:drawable/menu\_submenu\_background**</item>**
19. **<item** name="fastScrollPreviewBackgroundLeft"**>**@android:drawable/menu\_submenu\_background**</item>**
20. **<item** name="fastScrollOverlayPosition"**>**floating**</item>**
21. **<item** name="fastScrollTextColor"**>**@android:color/primary\_text\_dark**</item>**
23. <!-- Pointer style -->
24. **<item** name="pointerStyle"**>**@android:style/Pointer**</item>**
26. <!-- Accessibility focused drawable. -->
27. **<item** name="accessibilityFocusedDrawable"**>**@android:drawable/view\_accessibility\_focused**</item>**

<item name="disabledAlpha">0.5</item>

<item name="backgroundDimAmount">0.6</item>

### 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"

## 属性值V系统默认值

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

## 小结

两点说明，我们可以根据这些样式改变系统控件的样式

* 但不是所有的样式我们都可以修改，比如AlertDialog，布局和一些属性都是写在硬编码中了，我们是无法修改的。
* 通过学习这些主题和样式，我们能准确的把握我们能定制哪些样式，哪些我们不能定制，是否需要重新写一套替代的UI控件。

这个在项目决策和进度评估上是非常重要的。更多主题移步

https://github.com/CyanogenMod/android\_frameworks\_base/blob/cm-11.0/core/res/res/values/styles.xml

## 其他

<item name="isLightTheme">false</item>

## 子Theme定义

.Black为例，定义非常简单，几乎没有什么差别。。。

•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,

1. <!-- Variant on {@link #Theme} that ensures the background is  
    completely black. This is useful for things like image viewers and  
    media players. If you want the normal (dark background) theme  
    do <em>not</em> use this, use {@link #Theme}. -->  
   <style name="Theme.Black">  
    <item name="windowBackground">@color/black</item>  
    <item name="colorBackground">@color/black</item>  
   </style>  
     
   <!-- Variant of {@link #Theme\_Black} with no title bar -->  
   <style name="Theme.Black.NoTitleBar">  
    <item name="windowNoTitle">true</item>  
   </style>  
     
   <!-- Variant of {@link #Theme\_Black} that has no title bar and  
    no status bar. This theme  
    sets {@link android.R.attr#windowFullscreen} to true. -->  
   <style name="Theme.Black.NoTitleBar.Fullscreen">  
    <item name="windowFullscreen">true</item>  
    <item name="windowContentOverlay">@null</item>  
   </style>

# 系统主题引擎

## 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

## CTS

The Holo themes must not be modified in order to pass CTS.

Many related themes and styles depend on other values defined in this file.

If you would like to provide custom themes and styles for your device,

please see themes\_device\_defaults.xml.

一下文件不允许修改，否则过不了CTS认证

themes.xml(v2006)

themes\_material.xml (v2014)( Many related themes and styles depend on other values defined in this file.)

## 自定义系统主题

themes\_device\_defaults.xml.

## 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>

# 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

## public.xml 与symbols.

framework/base/core/res/res/values/

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）, 定义<java-symbol/>类别的id，系统私有的资源（不对app公开）

public.xml定义的资源进入到android.R.java（公有资源），定义<public />公共资源，可以在app中访问）。

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。

### 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

## 编译运行

android.jar 切记这个特例，这货是make sdk生成的，多方整合，别以为也可以找到对应目录，木有的！还有就是这个jar很实用的，很多时候我们想用AS直接调运系统的hide API等，自己编译一个就能派上用场啦！

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

## 为WinMS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

# 使用android系统自带资源

## **资源的引用语法**

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

感兴趣的同学可以移步到[API Guide的App Resources](http://androiddoc.qiniudn.com/guide/topics/resources/index.html)进行研读。这里我们主要简单说下资源的引用语法

**Android中资源在Java文件中引用的语法定义如下：**

[<package\_name>.]R.<resource\_type>.<resource\_name>

//注意：当资源在当前APP中则package\_name可以省略，当为系统的资源则可换位譬如android.

**Android中资源在XML文件中引用的语法定义如下：**

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

//注意：package\_name的规则同上java中，不过在XML中引入不是本包资源时要注意格式，譬如引用系统的资源格式为android:textColor="@android:color/secondary\_text\_dark"

**Android系统预制资源在XML文件中引用的特殊语法定义如下：**

//可以引用系统所有资源，public & private

@\*android:type/name

//只能引用系统public的资源

@android:type/name

//注意：没在frameworks/base/core/res/res/values/public.xml（也就是<sdk\_path>\platforms\android-X\data\res\values\public.xml）中申明的资源App时不推荐使用的。

**Android在XML文件中引用当前主题属性的语法定义如下：**

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

//资源值允许引用当前主题中的属性的值，这个属性值只能在style资源和XML中使用，随着当前主题的切换该值也在变换，该resource\_name不需要自己定义，系统会自己在当前主题下寻找，常见的譬如动画中等。

* 1

## Android在XML文件中创建或者引用资源语法定义如下：

//在R.java的type内部类中添加一条静态常量id资源标识符，如果标示符（包括系统资源）已经存在则表示引用该标示符。

@+type/name

//在R.java中寻找已经定义的标识符，如果找不到则提示失败错误，一般在xml中定义有先后关系。

@type/name

//所以一般推荐直接使用+号避免不必要的意外。

**Android在XML文件中xmlns语法定义如下：**

//xmlns（XML Namespaces）是XML的命名空间

//通用XML命名空间格式规则

xmlns:namespace-prefix="namespaceURI"

在Android的XML中命名空间规则如下：

xmlns:namespace-prefix=http://schemas.android.com/apk/res/应用程序包路径

* 1

在使用时规则如下：

namespace-prefix：属性

* 1

切记，xmlns的定义必须放在最外层开始的的标记中，譬如我们Activity的xml文件的根布局中的android前缀、tools前缀、自定义View的前缀等。常见的例子如下：

//android即为frameworks/base/core/res/res/values/attrs.xml中的属性

xmlns:android="http://schemas.android.com/apk/res/android"

//开发调试利器，不再过多说明

xmlns:tools="http://schemas.android.com/tools"

//Email App中res/values/attrs.xml等自定义属性

xmlns:settings="http://schemas.android.com/apk/res/com.android.email"

### 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:”。

*<!-- AlertDialog Styles:error: resource android:style/AlertDialog.DeviceDefault is private. -->*<**style name="AlertDialog.Rm500" parent="?android:style/AlertDialog.DeviceDefault"**/>

### 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**

**Siy9u**

## style attribute 'android:attr/keyboardViewStyle' is private.

*gestureOverlayViewStyle*

*seekBarPreferenceStyle*

windowFixedWidthMajor

windowFixedWidthMinor

**Theme.Rm500.Dialog.NoActionBar.FixedSize**

## error: resource android:style/AlertDialog.DeviceDefault is private.

style/AlertDialog is private.

<https://stackoverflow.com/questions/45865366/cardslib-error-resource-androidattr-foregroundinsidepadding-is-private-while-ta/46241073>

Try disabling the AAPT2 by adding android.enableAapt2=false to your gradle.properties fil

# REF

[Android Theme详细](https://blog.csdn.net/yanbober/article/details/51015630)