Android样式的开发:Style篇

2016-03-07 安卓应用频道

(点击上方公众号，可快速关注)

> 来源：Keegan小钢
> 链接：http://keeganlee.me/post/android/20150905

本系列：

Android 项目重构之路：架构篇

Android 项目重构之路：界面篇

Android 项目重构之路：实现篇（上）

Android 项目重构之路：实现篇（下）

Android技术积累：开发规范

Android样式的开发：shape篇

Android样式的开发：selector篇

前面铺垫了那么多，终于要讲到本系列的终篇，整合所有资源，定义成统一的样式。

哪些该定义成统一的样式呢？举几个例子吧：

1. 每个页面标题栏的标题基本会有一样的字体大小、颜色、对齐方式、内间距、外间距等，这就可以定义成样式；

2. 很多按钮也都使用一致的背景、内间距、文字颜色、文字大小、文字的对齐方式等，这也可以定义成样式；

3. 网络加载的进度条基本也都是一样的，同样可以定义成样式；

4. 不喜欢系统的弹出框样式，那也可以自定义样式。

样式的定义

Android的样式一般定义在res/values/styles.xml文件中，其中有一个根元素<resource>，而具体的每种样式定义则是通过<resource>下的子标签<style>来完成，<style>通过添加多个<item>来设置样式不同的属性。

另外，样式是可以继承的，可通过<style>标签的parent属性声明要继承的样式，也可通过点前缀 (.) 继承，点前面为父样式名称，后面为子样式名称。点前缀方式只适用于自定义的样式，若要继承Android内置的样式，则只能通过parent属性声明。

用个实例说明具体的用法吧，以下代码为Android 5.0系统默认的按钮样式：

```
<style name="Widget.Material.Button">
      <item name="background">@drawable/btn_default_material</item>
      <item name="textAppearance">?attr/textAppearanceButton</item>
      <item name="minHeight">48dip</item>
      <item name="minWidth">88dip</item>
      <item name="stateListAnimator">@anim/button_state_list_anim_material</item>
      <item name="focusable">true</item>
      <item name="clickable">true</item>
      <item name="gravity">center_vertical|center_horizontal</item></style>
```

其中，stateListAnimator指定状态改变时的动画，button_state_list_anim_material的代码如下：

```
<!-- res/anim/button_state_list_anim_material.xml --><?xml version="1.0" encoding="utf-8"?><selector xmln
s:android="http://schemas.android.com/apk/res/android">
      <item android:state_pressed="true" android:state_enabled="true">
            <set>
                  <objectAnimator
                        android:propertyName="translationZ"
                        android:duration="@integer/button_pressed_animation_duration"
                        android:valueTo="@dimen/button_pressed_z_material"
                        android:valueType="floatType" />
                  <objectAnimator
                        android:propertyName="elevation"
                        android:duration="0"
                        android:valueTo="@dimen/button_elevation_material"
                        android:valueType="floatType" />
            </set>
      </item>
      <!-- base state -->
      <item android:state_enabled="true">
            <set>
```

```
            <objectAnimator
                android:propertyName="translationZ"
                android:duration="@integer/button_pressed_animation_duration"
                android:valueTo="0"
                android:startDelay="@integer/button_pressed_animation_delay"
                android:valueType="floatType"/>
            <objectAnimator
                android:propertyName="elevation"
                android:duration="0"
                android:valueTo="@dimen/button_elevation_material"
                android:valueType="floatType" />
        </set>
    </item>
    <item>
        <set>
            <objectAnimator
                android:propertyName="translationZ"
                android:duration="0"
                android:valueTo="0"
                android:valueType="floatType"/>
            <objectAnimator
                android:propertyName="elevation"
                android:duration="0"
                android:valueTo="0"
                android:valueType="floatType"/>
        </set>
    </item></selector>
```

可以看到，每种状态的动画为属性动画集，属性动画的用法请参考Property Animation篇。

现在我想继承Widget.Material.Button样式，改变背景和文字颜色，那么，代码如下：

```
<!-- res/values/styles.xml --><resources>
    <style name="ButtonNormal" parent="@android:style/Widget.Material.Button" >
        <item name="android:background">@drawable/bg_btn_selector</item>
        <item name="android:textColor">@color/text_btn_selector</item>
    </style></resources>
```

其中，@drawable/bg_btn_selector和@color/text_btn_selector的实现请参照selector篇。

有些按钮，我只想改变文字颜色，但背景想让它透明，这时就可以用点前缀的方式继承以上的样式，代码如下：

```
<!-- res/values/styles.xml --><resources>
      <style name="ButtonNormal" parent="@android:style/Widget.Material.Button">
            <item name="android:background">@drawable/bg_btn_selector</item>
            <item name="android:textColor">@color/text_btn_selector</item>
      </style>

      <style name="ButtonNormal.Transparent">
            <item name="android:background">@drawable/bg_btn_transparent</item>
            <item name="android:textColor">@color/text_btn_selector</item>
      </style></resources>
```

引用的时候只要在相应的Button里添加style就可以了，代码如下：

```
<Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:onClick="onAction"
      android:text="@string/btn_action"
      style="@style/ButtonNormal.Transparent" />
```

有时候，定义的样式太多，如果都放在styles.xml文件里，那这文件也太臃肿了。因此，可以将样式分类拆分成多个文件。Android系统本身也拆分为多个文件存放的，如下列表全都是样式文件：

```
styles.xml
styles_device_defaults.xml
styles_holo.xml
styles_leanback.xml
styles_material.xml
styles_micro.xml
themes.xml
themes_device_defaults.xml
themes_holo.xml
themes_leanback.xml
themes_material.xml
```

```
themes_micro.xml
```

其中，主要分为两大类，styles定义了简单的样式，而themes则定义了主题。

主题

以上的简单例子只用于单个View，这是样式最简单的用法。但样式的用法不只是用于单个View，也能用于Activity或整个Application，这时候需要在相应的<activity>标签或<application>标签里设置android:theme*属性，引用的其实也是style*，但一般称为主题。

Android系统提供了多套主题，查看Android的frameworks/base/core/res/res/values目录，就会看到有以下几个文件(目前为止)：

- themes.xml：低版本的主题，目标API level一般为10或以下
- themes_holo.xml：从API level 11添加的主题
- themes_device_defaults.xml：从API level 14添加的主题
- themes_material.xml：从API level 21添加的主题
- themes_micro.xml：应该是用于Android Wear的主题
- themes_leanback.xml： 还不清楚什么用

不过在实际应用中，因为大部分都采用兼容包的，一般都会采用兼容包提供的一套主题：Theme.AppCompat。AppCompat主题默认会根据不同版本的系统自动匹配相应的主题，比如在Android 5.0系统，它会继承Material主题。不过这也会导致一个问题，不同版本的系统使用不同主题，就会出现不同的体验。因此，为了统一用户体验，最好还是自定义主题。

自定义主题也很简单，只要继承某一父主题，然后在<activity>标签或<application>中引用就可以了。

主题的定义示例如下：

```
<resources>
      <style name="AppTheme" parent="Theme.AppCompat">
            <item name="windowActionBar">false</item>
            <item name="windowNoTitle">true</item>
            <item name="windowAnimationStyle">@style/WindowAnimation</item>
      </style>

      <!-- Standard animations for a full-screen window or activity. -->
      <style name="WindowAnimation" parent="@android:style/Animation.Activity">
```

```
        <item name="activityOpenEnterAnimation">@anim/activity_open_enter</item>
        <item name="activityOpenExitAnimation">@anim/activity_open_exit</item>
        <item name="activityCloseEnterAnimation">@anim/activity_close_enter</item>
        <item name="activityCloseExitAnimation">@anim/activity_close_exit</item>
    </style></resources>
```

其中，WindowAnimation重新指定了Activity的转场动画，以下为activity_close_exit的示例代码：

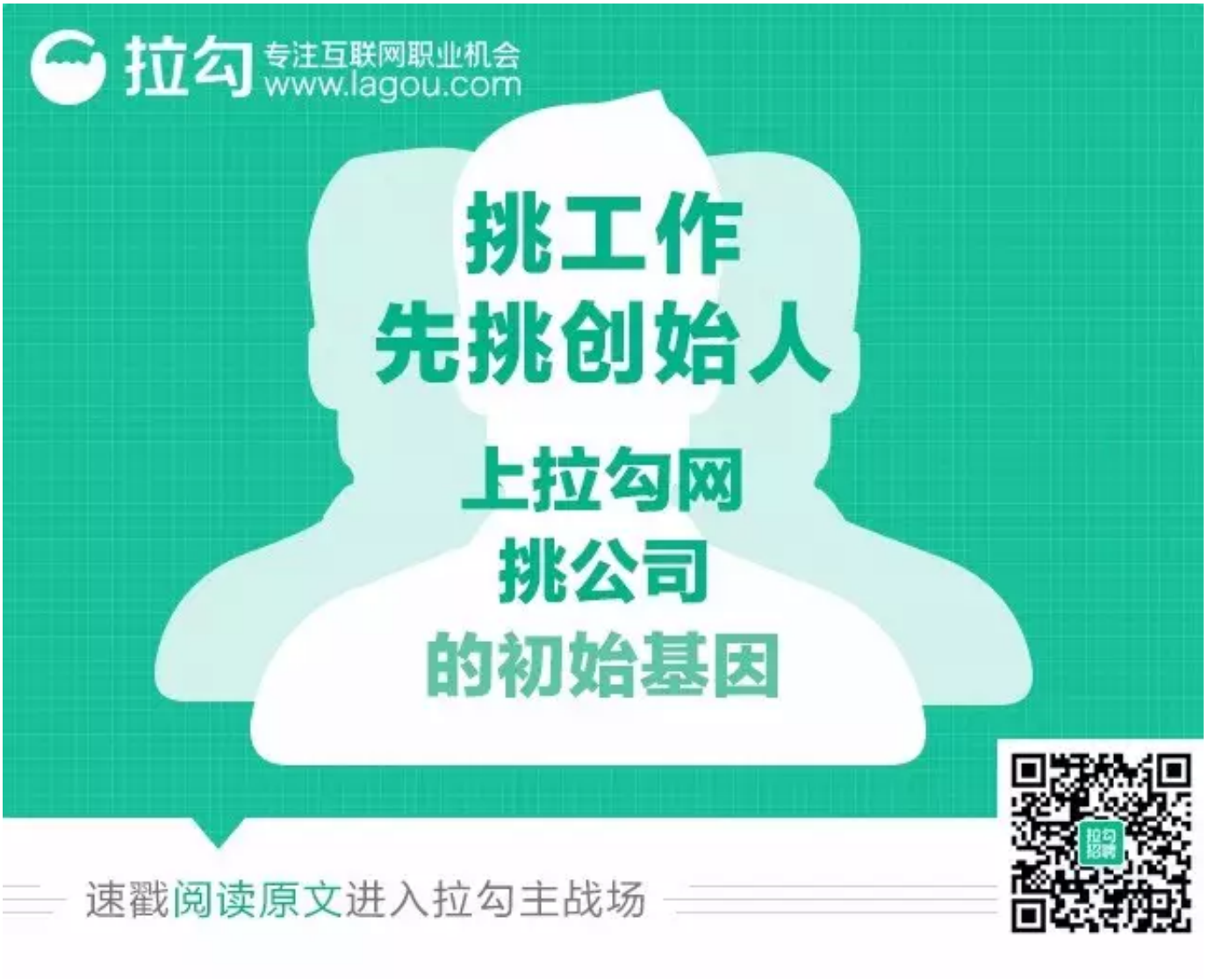
```
<!-- res/anim/activity_close_exit.xml --><?xml version="1.0" encoding="utf-8"?><set xmlns:android="http:/
/schemas.android.com/apk/res/android"
    android:shareInterpolator="false"
    android:zAdjustment="top">
    <alpha
        android:fromAlpha="0.0"
        android:toAlpha="1.0"
        android:interpolator="@interpolator/decelerate_quart"
        android:fillEnabled="true"
        android:fillBefore="false"
        android:fillAfter="true"
        android:duration="200" />
    <translate
        android:fromYDelta="8%"
        android:toYDelta="0"
        android:fillEnabled="true"
        android:fillBefore="true"
        android:fillAfter="true"
        android:interpolator="@interpolator/decelerate_quint"
        android:duration="350" /></set>
```

这是比较简单的视图动画，视图动画具体用法可参考View Animation篇。

接着，若要使用到整个Application，则在AndroidManifest.xml的<application>标签设置android:theme属性，示例代码如下：

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
```

```
    android:theme="@style/AppTheme">
    <!-- activity here --></application>
```



阅读原文

微信扫一扫
关注该公众号