**Android Style**

# 背景介绍

最近接触了一款APP，从Android5.1.1升级到Android8.1，整个APP使用的主题为”@android:style/Theme.DeviceDefault.Light”。测试在Bugzilla上提了好几个关于UI方面的Bug。研发一看就知道这些不是Bug，但是测试他们毕竟不是开发，只会看表面现象，同一个APP在不同版本Android平台上UI不一致都是Bug。要给测试、PM说清楚这些不是Bug，就必须了解Android系统的Style.Style可以简单理解为一组属性的集合，方便对APP的样式做一个统一处理。下面从属性开始了解Style.

# 属性

属性（attr）可以简单理解为特性。比如名字、皮肤颜色等等都是属性。

## 2.1申明属性

假如要实现人类皮肤和名字这样一组属性，那在Android中要如何实现呢？

首先在“res/values”目录下创建一个文件名为attrs.xml。具体类容如下:

<?xml version="1.0" encoding="utf-8"?>

<resources>

<!-- 自定义属性skinColor -->

<attr name="customSkinColor" format="color" />

<!-- 自定义属性name -->

<attr name="customName" format="string" />

<!-- 申明一个人类的属性的集合 -->

<declare-styleable name="customPerson">

<attr name="customSkinColor" />

<attr name="customName" />

</declare-styleable>

</resources>

这样申明之后发生了什么？

## 2.2 属性实现

当申明了属性之后，在R.java中会有如下定义存在:

public final class R {

public static final class attr {

public static final int customSkinColor=0x7f03006f;

public static final int customName=0x7f03006d;

}

public static final class styleable {

public static final int[] customPerson={

0x7f03006d, 0x7f03006f

};

public static final int customPerson\_customName=0;

public static final int customPerson\_customSkinColor=1;

}

}

Android中的每个资源，都有它唯一的编号。编号是一个32位数字，用十六进制表示就是0xPPTTEEEE。PP为package id,TT 为type id,EEEE 为entry id.比如上面这个例子，执行下面的命令:

aapt d resources app-debug.apk | grep 0x7f03006d

spec resource 0x7f03006d com.crab.test:attr/customName: flags=0x00000000

resource 0x7f03006d com.crab.test:attr/customName: <bag>

这就表示customName这个资源的编号是 0x7f03006d 。它的package id是0x7f，资源类型的id为0x03(代表attr)，它的entry id为0x006d。

备注:package id的值为0x01是系统资源，也就是包名为”android”的资源。type id代表的是drawable,string,layout等等类型的资源，默认是从值0x01开始。entry id代表这个资源在type id中的位置，默认是从0x0000开始。

## 2.3 属性的使用

### 2.3.1 属性组使用

属性组(styleable)是属性的集合。前面定义了一个属性组(customPerson),它有两个属性(customName、customSkinColor)。下面我们来使用属性组:

public class PersonView extends TextView {

public PersonView(Context context, @Nullable AttributeSet attrs) {

super(context, attrs);

//customPerson是一个数组，在刚才R.java中定义过

final TypedArray a = context.obtainStyledAttributes(

attrs, R.styleable.customPerson, 0, 0);

//customPerson\_customSkinColor的值为1，数组第二个元素

int skinColor = a.getColor(R.styleable.customPerson\_customSkinColor, Color.WHITE);

//customPerson\_customName的值为0，数组第一个元素

String name = a.getString(R.styleable.customPerson\_customName);

setText(name);

setTextColor(skinColor);

a.recycle();

}

}

属性组的使用是不是很简单，下面设置具体属性值。

### 2.3.2 属性值的设定

现在来给具体的属性设置值:

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"

tools:context="com.crab.test.MainActivity"

tools:showIn="@layout/activity\_main">

<com.crab.test.PersonView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:customName="xiechaojun" //自己定义的属性

app:customSkinColor="#FFFF0000" //自己定义的属性

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

/>

</android.support.constraint.ConstraintLayout>

属性的使用也很简单。现在考虑一个问题，如果APP里面有10个PersonView，想让所有PersonView的皮肤颜色都是黄色，要怎么办？下面介绍Style.

# 3 Android Style

## 3.1 定义集合属性

10个PersonView？咋办？首先在attrs.xml中创建一个属性名字以及使用这个属性的集合

<?xml version="1.0" encoding="utf-8"?>

<resources>

<!-- 自定义属性skinColor -->

<attr name="customSkinColor" format="color" />

<!-- 自定义属性name -->

<attr name="customName" format="string" />

<!-- 申明一个人类的属性的集合 -->

<declare-styleable name="customPerson">

<attr name="customSkinColor" />

<attr name="customName" />

</declare-styleable>

<!-- 自定义属性personViewStyle -->

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

<!-- 申明一整个应用人类要使用的属性的集合 -->

<declare-styleable name="customPersonStyle">

<attr name="personViewStyle" />

</declare-styleable>

</resources>

## 3.2 自定义Style

在“res/values/styles.xml”文件中添加自定义的Style。

<resources>

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

<!-- Customize your theme here. -->

<item name="colorPrimary">@color/colorPrimary</item>

<item name="colorPrimaryDark">@color/colorPrimaryDark</item>

<item name="colorAccent">@color/colorAccent</item>

<!-- 与上边属性名字对应 -->

<item name="personViewStyle">@style/PersonStyle</item>

</style>

<!-- 定义所有PersonView的皮肤颜色为黄色 -->

<style name="PersonStyle">

<item name="customSkinColor">#FFFFFF00</item>

</style>

</resources>

## 3.3 使用自定义的Style

### 3.3.1 PersonView的使用

请注意与2.3.1小节实现不同的地方:

public class PersonView extends TextView {

public PersonView(Context context, @Nullable AttributeSet attrs) {

super(context, attrs);

//系统主题

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

//获取自定义的person style属性集合

final TypedArray a = theme.obtainStyledAttributes(

attrs, R.styleable.customPersonStyle, 0, 0);

//获取到具体某个person style ID.

int personStyleId =

a.getResourceId(R.styleable.customPersonStyle\_personViewStyle,0);

if(personStyleId!=-1) {

TypedArray aa =

theme.obtainStyledAttributes(personStyleId, R.styleable.customPerson);

//获取到主题中设定的皮肤值

int skinColor = aa.getColor(R.styleable.customPerson\_customSkinColor, Color.BLUE);

setTextColor(skinColor);

a.recycle();

}

a.recycle();

}

}

### 3.3.2 布局文件中使用

在布局文件中定义2个PersonView，不设置它皮肤颜色的属性，但是最后效果是两个PersonView的字体颜色都是黄色。

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"

tools:context="com.crab.firstjniapplication.MainActivity"

tools:showIn="@layout/activity\_main">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

>

<!-- 这里没有设置皮肤颜色，但是系统主题中有设置为黄色-->

<com.crab.firstjniapplication.PersonView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="xiechaojun"

/>

<com.crab.firstjniapplication.PersonView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="xiechaojun1"

/>

</LinearLayout>

</android.support.constraint.ConstraintLayout>

# 4 解Bug

比如想让APP UI在不同Android版本上表现一样，可以自定Style，让APP的样式统一。比如想让所有的EditText的样式保持统一不随平台改变，可以下面这样实现:

<style name="Theme.Base" parent="@android:style/Theme.DeviceDefault.Light">

<item name="android:buttonStyle">@style/Button.Base</item>

<!-- 使用自定义样式 -->

<item name="android:editTextStyle">@style/CustomEditTextStyle</item>

</style>

<style name="CustomEditTextStyle" parent="@android:style/Widget.Material.EditText">

<!-- 自定义样式 -->

</style>

**5 Android support包属性遇到的问题**

在做Android P平台应用开发时，自己定义了一个AlertDialog,继承制android.app.AlertDialog,想给这个自定义的AlertDialog设置主题，使用的下面方法:

<!-- 使用的是support包的主题 -->

<style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">

<!-- 自定义的AlertDialog样式 -->

<item name="alertDialogTheme">@style/AppAlertDialogTheme</item>

</style>

<style name="AppAlertDialogTheme" parent="ThemeOverlay.AppCompat.Dialog.Alert">

<!-- 给AlertDialog的List项目添加一条分割线-->

<item name="listDividerAlertDialog">@drawable/preference\_list\_divider\_material</item>

</style>

这样设置之后发现并没有效果,那是什么原因呢？

我们来看下android.app.AlertDialog的源码

/\*\*

\* Creates an alert dialog that uses the default alert dialog theme.

\* <p>

\* The default alert dialog theme is defined by

\* {@link android.R.attr#alertDialogTheme} within the parent

\* {@code context}'s theme.

\*

\* @param context the parent context

\* @see android.R.styleable#Theme\_alertDialogTheme

\*/

protected AlertDialog(Context context) {

this(context, 0);

}

从源码可以看出alertDialogTheme是定义在属性android.R.attr下的，而像我们这样使用的support包定义的属性:

<!-- 自定义的AlertDialog样式 -->

<item name="alertDialogTheme">@style/AppAlertDialogTheme</item>

是在自己的包名.R.attr下面的.知道原因了都很好改：

1.在自定义的AlertDialog主题前都加上android，比如android:alertDialogTheme,android:listDividerAlertDialog

2.使用Support包里面的AlertDialog.