Android自定义控件之自定义组合控件

前言：

前两篇介绍了自定义控件的基础原理Android自定义控件之基本原理（一）、自定义属性Android自定义控件之自定义属性（二）。今天重点介绍一下如何通过自定义组合控件来提高布局的复用，降低开发成本，以及维护成本。

自定义控件相关文章地址：

Android自定义控件之基本原理

Android自定义控件之自定义属性

Android自定义控件之自定义组合控件

Android自定义控件之自定义ViewGroup实现标签云

使用自定义组合控件的好处？

我们在项目开发中经常会遇见很多相似或者相同的布局，比如APP的标题栏，我们从三种方式实现标题栏来对比自定义组件带来的好处，毕竟好的东西还是以提高开发效率，降低开发成本为导向的。

1.）第一种方式：直接在每个xml布局中写相同的标题栏布局代码

复制代码

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<RelativeLayout

android:layout\_width="match\_parent"

android:background="@color/green"

android:layout\_height="45dp">

<Button

android:id="@+id/title\_bar\_left"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_centerVertical="true"

android:layout\_marginLeft="5dp"

android:background="@mipmap/titlebar\_back\_icon"

android:minHeight="45dp"

android:minWidth="45dp"

android:textSize="14sp" />

<TextView

android:id="@+id/title\_bar\_title"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:text="登录"

android:singleLine="true"

android:textSize="17sp" />

<Button

android:id="@+id/title\_bar\_right"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentRight="true"

android:layout\_centerVertical="true"

android:layout\_marginRight="7dp"

android:text="提交"

android:textColor="@android:color/white"

android:background="@null"

android:minHeight="45dp"

android:minWidth="45dp"

android:textSize="14sp" />

</RelativeLayout>

</LinearLayout>

复制代码

这种方式没有任何布局复用的概念，同时也让当前的布局变得臃肿难以维护，开发效率低下，而且这个还需要要求每个开发人员必须细心否则有可能会做出参差不齐的标题栏，所以这种方式是最不推荐使用的。

2.）第二种方式：使用include标签

首先定义标题栏布局

复制代码

<RelativeLayout

android:layout\_width="match\_parent"

android:background="@color/green"

android:layout\_height="45dp">

<Button

android:id="@+id/title\_bar\_left"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_centerVertical="true"

android:layout\_marginLeft="5dp"

android:minHeight="45dp"

android:minWidth="45dp"

android:textSize="14sp" />

<TextView

android:id="@+id/title\_bar\_title"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:singleLine="true"

android:textSize="17sp" />

<Button

android:id="@+id/title\_bar\_right"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentRight="true"

android:layout\_centerVertical="true"

android:layout\_marginRight="7dp"

android:background="@null"

android:minHeight="45dp"

android:minWidth="45dp"

android:textSize="14sp" />

</RelativeLayout>

复制代码

然后在需要的地方通过include标签实现引用

复制代码

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<include layout="@layout/view\_title\_bar" />

</LinearLayout>

复制代码

通过上面的布局代码，我们可以使用上面这种方式确实实现了布局的复用，而且也避免了开发人员开发出参差不齐标题栏的问题，但是同时也引入了新的问题，比如更加降低了开发效率，加大了开发成本，问题就在我们该如何为每个布局文件定义标题栏？只有通过代码的方式来设置标题问题，左右按钮等其他的属性，导致布局属性和Activity代码耦合性比较高，所以这种方式也不是推荐的方式。

3.）第三种方式：通过自定义组合控件

这里先不具体介绍如何实现一个自定义组合控件，这里先介绍一下自定义组合控件带来的好处。

提高布局文件开发效率

降低布局文件维护成本

降低布局文件和Activity代码耦合性

容易扩展

简单易用

如何实现一个自定义组合控件

1.）先定义一个布局文件

复制代码

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

<Button

android:id="@+id/title\_bar\_left"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_centerVertical="true"

android:layout\_marginLeft="5dp"

android:background="@null"

android:minHeight="45dp"

android:minWidth="45dp"

android:textSize="14sp" />

<TextView

android:id="@+id/title\_bar\_title"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:singleLine="true"

android:textSize="17sp" />

<Button

android:id="@+id/title\_bar\_right"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentRight="true"

android:layout\_centerVertical="true"

android:layout\_marginRight="7dp"

android:background="@null"

android:minHeight="45dp"

android:minWidth="45dp"

android:textSize="14sp" />

</merge>

复制代码

注意：这里为何要使用merge标签，自定义组合控件时会继承RelativeLayout、LinearLayout等控件，这样导致布局的层级无形中增加了一层，如下是对比：

未使用merge标签

使用merge标签

2.）定义自定义属性

比如标题文字、标题栏左边按钮图标等。

复制代码

<declare-styleable name="CustomTitleBar">

<attr name="title\_background\_color" format="reference|integer" />

<attr name="left\_button\_visible" format="boolean" />

<attr name="right\_button\_visible" format="boolean" />

<attr name="title\_text" format="string" />

<attr name="title\_text\_color" format="color" />

<attr name="title\_text\_drawable" format="reference|integer" />

<attr name="right\_button\_text" format="string" />

<attr name="right\_button\_text\_color" format="color" />

<attr name="right\_button\_drawable" format="reference|integer" />

<attr name="left\_button\_text" format="string" />

<attr name="left\_button\_text\_color" format="color" />

<attr name="left\_button\_drawable" format="reference|integer" />

</declare-styleable>

复制代码

3.）自定义一个View根据需求继承不同的ViewGroup子类，比如：RelativeLayout、LinearLayout等，我们这里继承RelativeLayout。

复制代码

public class CustomTitleBar extends RelativeLayout {

private Button titleBarLeftBtn;

private Button titleBarRightBtn;

private TextView titleBarTitle;

public CustomTitleBar(Context context, AttributeSet attrs) {

super(context, attrs);

LayoutInflater.from(context).inflate(R.layout.custom\_title\_bar, this, true);

titleBarLeftBtn = (Button) findViewById(R.id.title\_bar\_left);

titleBarRightBtn = (Button) findViewById(R.id.title\_bar\_right);

titleBarTitle = (TextView) findViewById(R.id.title\_bar\_title);

TypedArray attributes = context.obtainStyledAttributes(attrs, R.styleable.CustomTitleBar);

if (attributes != null) {

//处理titleBar背景色

int titleBarBackGround = attributes.getResourceId(R.styleable.CustomTitleBar\_tlb\_title\_background\_color, Color.GREEN);

setBackgroundResource(titleBarBackGround);

//先处理左边按钮

//获取是否要显示左边按钮

boolean leftButtonVisible = attributes.getBoolean(R.styleable.CustomTitleBar\_tlb\_left\_button\_visible, true);

if (leftButtonVisible) {

titleBarLeftBtn.setVisibility(View.VISIBLE);

} else {

titleBarLeftBtn.setVisibility(View.INVISIBLE);

}

//设置左边按钮的文字

String leftButtonText = attributes.getString(R.styleable.CustomTitleBar\_tlb\_left\_button\_text);

if (!TextUtils.isEmpty(leftButtonText)) {

titleBarLeftBtn.setText(leftButtonText);

//设置左边按钮文字颜色

int leftButtonTextColor = attributes.getColor(R.styleable.CustomTitleBar\_tlb\_left\_button\_text\_color, Color.WHITE);

titleBarLeftBtn.setTextColor(leftButtonTextColor);

}

//设置左边图片icon 这里是二选一 要么只能是文字 要么只能是图片

int leftButtonDrawable = attributes.getResourceId(R.styleable.CustomTitleBar\_tlb\_left\_button\_drawable, R.mipmap.titlebar\_back\_icon);

if (leftButtonDrawable != -1) {

titleBarLeftBtn.setCompoundDrawablesWithIntrinsicBounds(leftButtonDrawable, 0, 0, 0); //设置到哪个控件的位置（）

}

//处理标题

//先获取标题是否要显示图片icon

int titleTextDrawable = attributes.getResourceId(R.styleable.CustomTitleBar\_tlb\_title\_text\_drawable, -1);

if (titleTextDrawable != -1) {

titleBarTitle.setBackgroundResource(titleTextDrawable);

} else {

//如果不是图片标题 则获取文字标题

String titleText = attributes.getString(R.styleable.CustomTitleBar\_tlb\_title\_text);

if (!TextUtils.isEmpty(titleText)) {

titleBarTitle.setText(titleText);

}

//获取标题显示颜色

int titleTextColor = attributes.getColor(R.styleable.CustomTitleBar\_tlb\_title\_text\_color, Color.WHITE);

titleBarTitle.setTextColor(titleTextColor);

}

//先处理右边按钮

//获取是否要显示右边按钮

boolean rightButtonVisible = attributes.getBoolean(R.styleable.CustomTitleBar\_tlb\_right\_button\_visible, true);

if (rightButtonVisible) {

titleBarRightBtn.setVisibility(View.VISIBLE);

} else {

titleBarRightBtn.setVisibility(View.INVISIBLE);

}

//设置右边按钮的文字

String rightButtonText = attributes.getString(R.styleable.CustomTitleBar\_tlb\_right\_button\_text);

if (!TextUtils.isEmpty(rightButtonText)) {

titleBarRightBtn.setText(rightButtonText);

//设置右边按钮文字颜色

int rightButtonTextColor = attributes.getColor(R.styleable.CustomTitleBar\_tlb\_right\_button\_text\_color, Color.WHITE);

titleBarRightBtn.setTextColor(rightButtonTextColor);

}

//设置右边图片icon 这里是二选一 要么只能是文字 要么只能是图片

int rightButtonDrawable = attributes.getResourceId(R.styleable.CustomTitleBar\_tlb\_right\_button\_drawable, -1);

if (rightButtonDrawable != -1) {

titleBarRightBtn.setCompoundDrawablesWithIntrinsicBounds(0, 0, rightButtonDrawable, 0); //设置到哪个控件的位置（）

}

attributes.recycle();

}

}

public void setTitleClickListener(OnClickListener onClickListener) {

if (onClickListener != null) {

titleBarLeftBtn.setOnClickListener(onClickListener);

titleBarRightBtn.setOnClickListener(onClickListener);

}

}

public Button getTitleBarLeftBtn() {

return titleBarLeftBtn;

}

public Button getTitleBarRightBtn() {

return titleBarRightBtn;

}

public TextView getTitleBarTitle() {

return titleBarTitle;

}

}

复制代码

4.）在不同的XML布局中引用

关于如何使用自定义属性这里就不再说明了，为了更加直观的查看效果，我这里在一个布局文件中实现不同要求的标题栏

复制代码

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

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<com.whoislcj.views.CustomTitleBar

android:layout\_width="match\_parent"

android:layout\_height="45dp"

android:layout\_marginTop="10dp"

lee:right\_button\_drawable="@mipmap/titlebar\_add\_icon"

lee:title\_background\_color="@color/green"

lee:title\_text="标题1" />

<com.whoislcj.views.CustomTitleBar

android:layout\_width="match\_parent"

android:layout\_height="45dp"

android:layout\_marginTop="10dp"

lee:right\_button\_visible="false"

lee:title\_background\_color="@color/green"

lee:title\_text="标题2" />

<com.whoislcj.views.CustomTitleBar

android:layout\_width="match\_parent"

android:layout\_height="45dp"

android:layout\_marginTop="10dp"

lee:left\_button\_text="左边"

lee:right\_button\_text="右边"

lee:title\_background\_color="@color/red"

lee:title\_text="标题3" />

<com.whoislcj.views.CustomTitleBar

android:layout\_width="match\_parent"

android:layout\_height="45dp"

android:layout\_marginTop="10dp"

lee:left\_button\_text="左边"

lee:right\_button\_drawable="@mipmap/titlebar\_add\_icon"

lee:title\_background\_color="@color/red"

lee:title\_text="标题4" />

<com.whoislcj.views.CustomTitleBar

android:layout\_width="match\_parent"

android:layout\_height="45dp"

android:layout\_marginTop="10dp"

lee:left\_button\_text="左边"

lee:left\_button\_text\_color="@color/red"

lee:right\_button\_drawable="@mipmap/titlebar\_add\_icon"

lee:title\_background\_color="@color/blue"

lee:title\_text="标题5" />

<com.whoislcj.views.CustomTitleBar

android:layout\_width="match\_parent"

android:layout\_height="45dp"

android:layout\_marginTop="10dp"

lee:left\_button\_text="左边"

lee:left\_button\_text\_color="@color/red"

lee:right\_button\_drawable="@mipmap/titlebar\_add\_icon"

lee:title\_background\_color="@color/blue"

lee:title\_text="标题6"

lee:title\_text\_color="@color/black" />

</LinearLayout>

复制代码

显示效果



总结：

通过本篇文章我们得知，通过自定义组合控件确实能够提高开发效率，降低维护成本，但是也需要UI设计风格保持高度一致，不然的话只能呵呵哒了！所以想要做好一个app需要一个有共识的团队才行。本篇介绍到此为止，下一篇要更新什么我还没有想好！有可能是自定义控件的事件回调，也有可能自定义ViewGroup实现流式布局。