视频地址：P:\01\_在传智当老师获取的视频\00\_AndroidStudio\唐杨-Android-MaterialDesign\

**Android 新特性(5.0/6.0)**

* MD-CoordinatorLayout (协调者布局)
* MD-FloatingActionButton (悬浮操作按钮)
* MD-Toolbar 工具栏(替代之前的ActionBar)
* MD-AppBarLayout (应用标题栏容器)
* MD-CollapsingToolbarLayout 折叠效果的布局容器
* MD-CoordinatorLayout Behaviors(变化反应)的使用
* MD-TabLayout 标签导航
* MD-TextInputLayout 输入框控件的悬浮标签
* MD-NavigationView 抽屉导航
* RecyclerView 代替ListView/GridView
* 运行时权限(6.0)

## 概述

* Google在2015的IO大会上，发布了Android5.0及Material Design设计规范，同时，带来了全新的Android Design Support Library（向下兼容到Android 2.2）。安卓5.0是有史以来最重要的安卓版本之一，这其中有很大部分要归功于Material Design的引入，这种新的设计语言让整个安卓的用户体验焕然一新。
* 要使用Material Design需要进行下面的配置方：  
  1、添加design依赖：compile 'com.android.support:design:25.3.1'  
  2、AppCompatActivity界面不再继承 Activity, FragmentActivity 或ActionBarActivity, 而是继承AppCompatActivity, 目的是为了将MD的风格,及Toolbar等新的特效兼容到低  
  3、Theme.AppCompat Activity的主题配置为Theme.AppCompat, 在主题中可以配置许多系统自带属性值. 目的同样是为了将新的风格兼容到API 7

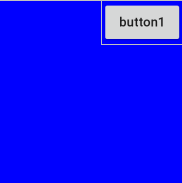
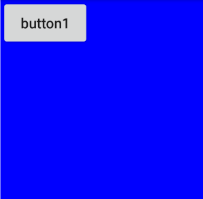
# 新创建AndroidStudio项目分析

创建一个带EmptyActivity的新项目，运行结果如下：



## 清单文件分析

查看项目的清单文件，发现application有如下一些奇怪的属性，如下：

* android:allowBackup="true" 声明是否允许备份应用数据(如系统备份或者手动adb备份），危险，最好设置为false。这个属性不写默认为true。
* android:roundIcon="@mipmap/ic\_launcher\_round" 这是Android 7.1（API 25）属性，运行到这个版本或更高版本的手机时，优先使用圆形图标来作为桌面图标。可以右击🡪New🡪Image Asset创建圆形图标。
* android:supportsRtl="true" 默认值是false。声明应用程序是否支持从右到左（Right to left）布局，如果设置为true，并且targetSdkVersion设置为17或更高，所使用的各种RTL api将被激活,系统应用程序可以显示RTL布局。如果设置为false或者targetSdkVersion设置为16或更低,RTL api将被忽略,你的布局总是会从左到右。在开发者模式中有如下开关：  
    
  有一个水平线性布局如下：  
    
  打开上面的从右到左开关后效果如下：  
    
  保持从右到左开关是打开状态，看如下布局：  
    
  蓝色是FrameLayout，这时的Button是没有指定layout\_gravity属性的， 我们给Button设置layout\_gravity值是right，会看到效果和上面一样，没有把方向给我们反过来，因为right就是代表right，如果变到左边显示的话，则就不能叫right了，所以这个时候layout\_gravity应该使用end值，结果如下：  
    
  因为方向是从右到左的，所以end就代表了最左边。通过这个例子也可以得出这样的结论：layout\_gravity属性默认值是start而不是left  
  也可在代码中单独改变某个布局的方向：ViewCompat.setLayoutDirection(View, ViewCompat.LAYOUT\_DIRECTION\_RTL : ViewCompat.LAYOUT\_DIRECTION\_LTR);

清单文件中，默认使用的主题是：Theme.AppCompat.Light.DarkActionBar，如下：

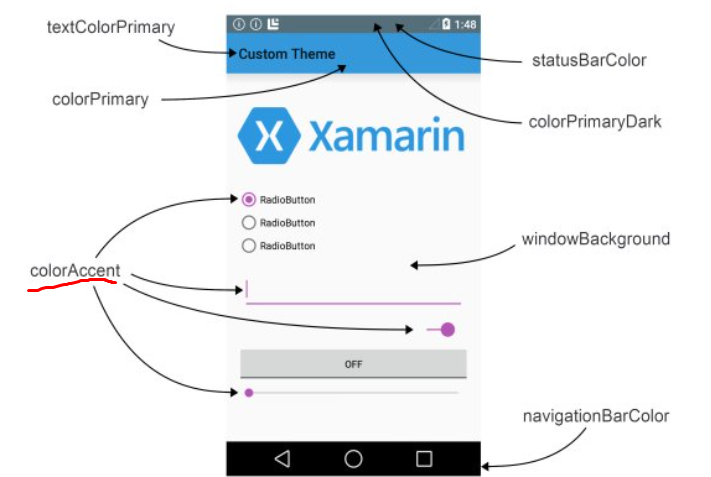
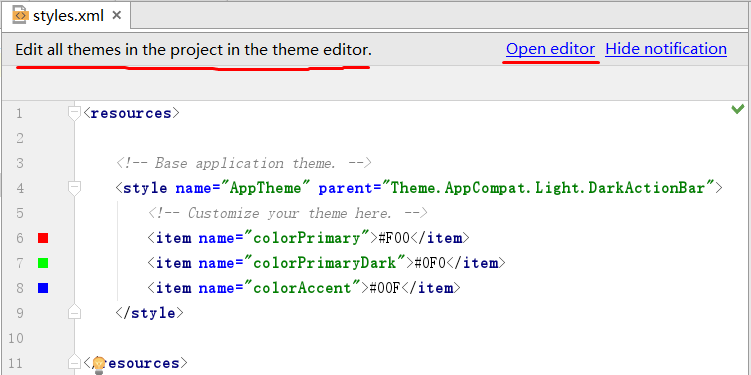
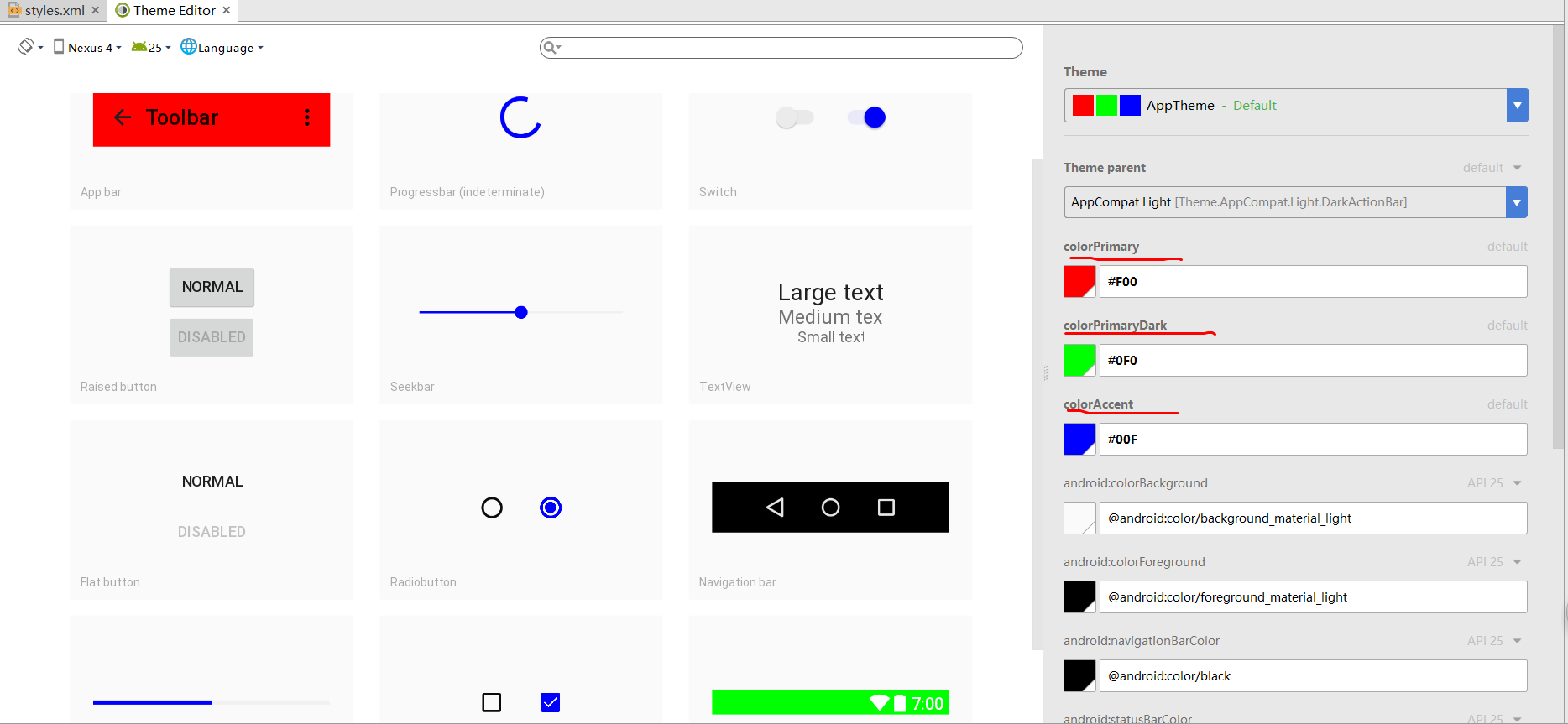


这里我是修改了默认值，颜色分别设置为红绿蓝，运行效果如下：



colorPrimary：应用的主要品牌颜色。默认情况下,这是应用于ActionBar的背景颜色。

colorPrimaryDark：主要品牌颜色的黑暗的变异。默认情况下,这个颜色应用到状态栏(通过statusBarColor)和导航栏(通过navigationBarColor)。

colorAccent：主要品牌颜色的明亮补充。默认情况下,这个颜色应用于框架控制(通过colorControlActivated)。上图中看不到这个颜色的体现，到底是什么地方的颜色呢，从网上找的一张图，如下：  
  
其实AS是有一个主题编辑器的，使用它能很方便的看到主题的预览，如下：  
  
点击Open editor就可以打开主题编辑器了，如下：  


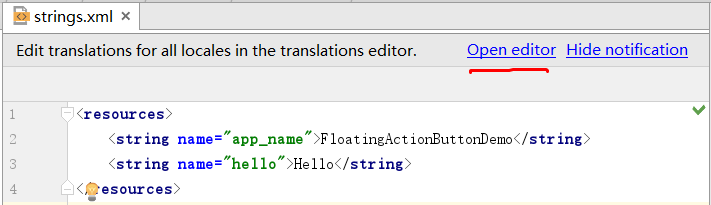
## 布局文件分析

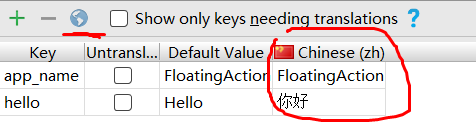
tools命令空间的属性是用来预览的，编译的时候会被忽略，也就是编译后就没有这些tools属性了。

* context属性：
* 如上面的属性是告诉系统说这个布局是用于MainActivity的，假设我们在清单文件中有给这个MainActivity设置主题，则预览布局的时候就可以看到主题的显示情况。
* 再比如在ListView中添加tools:listitem="@layout/item"属性，可以在布局中以该item预览ListView。
* 再比如TextView，我们习惯使用text属性设置一些文字以便在xml中进行预览，但是运行的时候希望显示成服务返回的数据，所以运行的时候不希望TextView有默认值，则可以使用：tools:text="你好"，所有控件的所有属性，如果你希望这个属性仅仅应用于预览阶段，则可以把该属性的android:改成tools:，属性名字不用变。
* xml中的Fragment可以使用：tools:layout="@android:layout/list\_content"进行Fragemnt的界面预览
* tools:ignore 用于忽略警告，如ImageView没有设置contentDescription属性会有警告，或者文字硬编码也会有警告
* tools:targetApi 如代的最小api是15，但是xml中使用了api21的控件，如果你确认这个控件不会在api 21以下运行，则可以使用这个属性消除警告
* tools:menu 菜单预览
* tools:showIn="@layout/activity\_main" 告诉AS这个布局是要嵌入到activity\_main.xml中显示的，如在Fragment的布局中可以使用这个属性。
*   
  这表示在预览界面直接可看到抽屉打开的效果

更多关于tools属性可查看博文：<http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2015/0309/2567.html> 或者直接百度xmlns:tools

## 翻译编辑器





点击地球按钮添加了一个中文语言，然后对照默认值增加对应语言的值

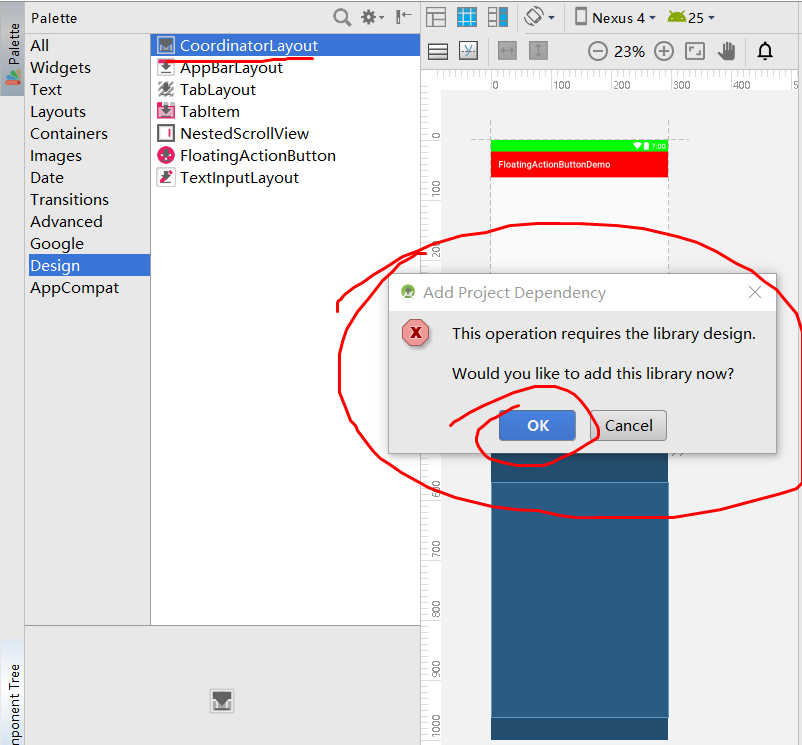
## 布局根元素变化

## 继承的Activity类型变化

## 多了mipmap

# 添加Design或其它Android支持库的快捷方式

不需要手动在gradle中添加依赖，我们打开随意一个布局，在Palette面板中找到Design，随便选择一个控制进行预览，这时就会弹出对话框询问你是否要添加对应的库，点OK即可。



# FloatingActionButton与Snackbar

布局：

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"** >  
  
 <**android.support.design.widget.FloatingActionButton  
 android:id="@+id/fab"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="end|bottom"  
 android:layout\_margin="16dp"**/>  
  
</**android.support.design.widget.CoordinatorLayout**>

代码：

**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.***fab***);  
 fab.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Toast.*makeText*(MainActivity.**this**, **"哈哈哈"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
}

运行效果：



我们并没有指定颜色，它是受到了主题中colorAccent颜色的影响。

属性：

app:backgroundTint，Tint单词含意“给…染色”，给背景染色

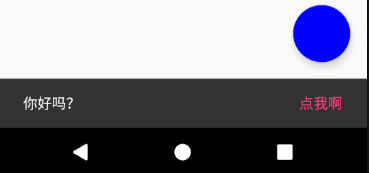
app:rippleColor为波纹颜色，我们点击的时候会显示，系统会处理我们的颜色以波纹形式过渡显示

app:fabSize="mini" 设置FAB的大小，该属性有两个值，分别为normal和mini，对应的FAB大小分别为56dp和40dp。

app:srcCompat="@android:drawable/ic\_dialog\_email" 设置FAB的图标，Google建议符合Design设计的该图标大小为24dp。

修改FAB的单击事件，如下：

fab.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Snackbar snackbar = Snackbar.*make*(v, **"你好吗？"**, Snackbar.***LENGTH\_SHORT***);  
 snackbar.setAction(**"点我啊"**, **new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 Toast.*makeText*(MainActivity.**this**, **"你点我了"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 snackbar.show();  
 }  
});



点击FAB的时候，底部出现了Snackbar，它的显示类似于Toast，时间到了会自动隐藏，或者向右滑动消除，或者点击 “点我阿”则马上消失。

我们看到Snackbar出现时，FloatingActionButton向上移动了，在构造Snackbar的时候需要传一个View参数，它会从这个View里向上查找出一个View作为父View，这个父View要么是CoordinatorLayout，要么是window decor's content view。优先使用CoordinatorLayout。

# Toolbar

Toolbar使用原理：把系统的ActionBar去除，把我们的Toolbar当作ActionBar。

在主题中把ActionBar去除，如下：



在布局中，使用控件面板直接拖一个Toolbar到布局中，如下：

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
 <**android.support.v7.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="?attr/colorPrimary"  
 android:minHeight="?attr/actionBarSize"  
 android:theme="?attr/actionBarTheme"** />  
  
</**android.support.design.widget.CoordinatorLayout**>

代码：

Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  
setSupportActionBar(toolbar);

运行效果如下：

