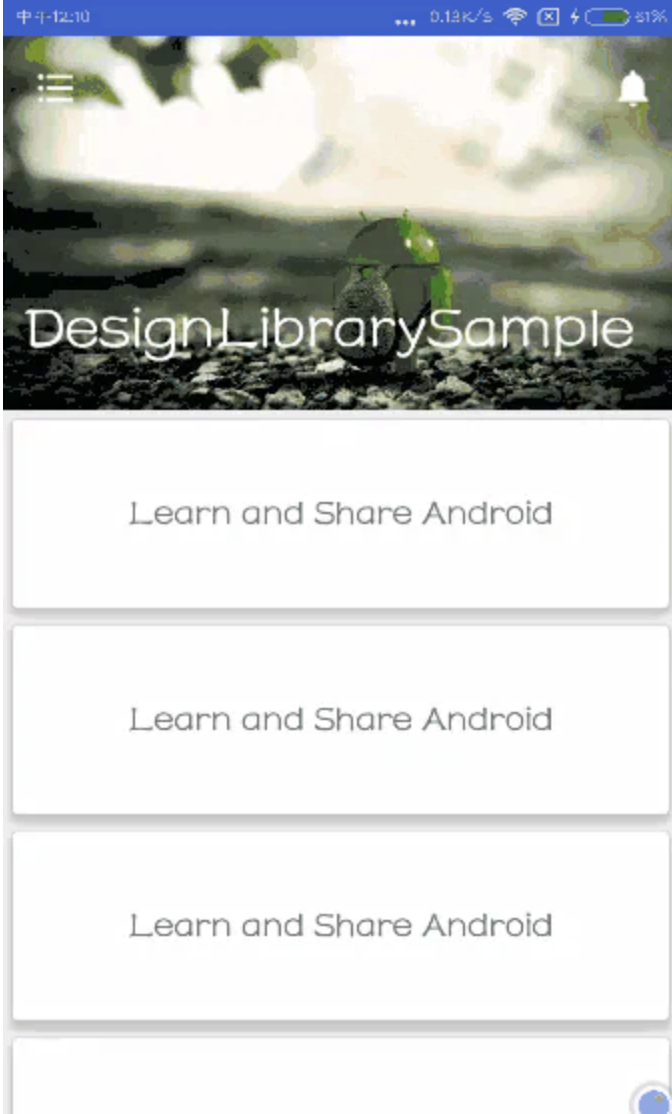
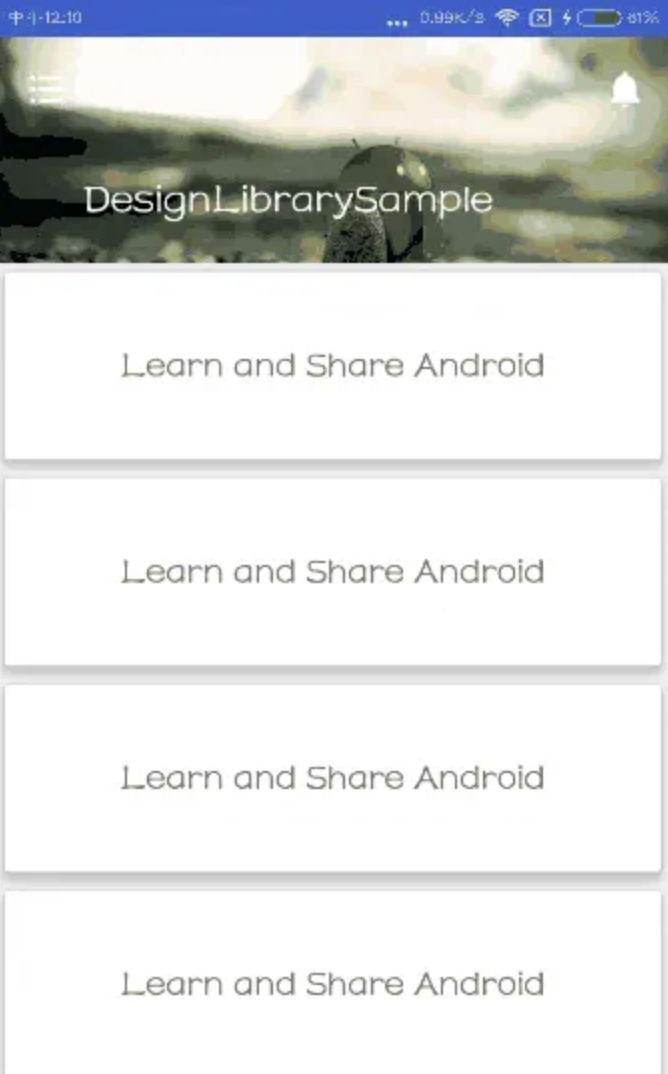
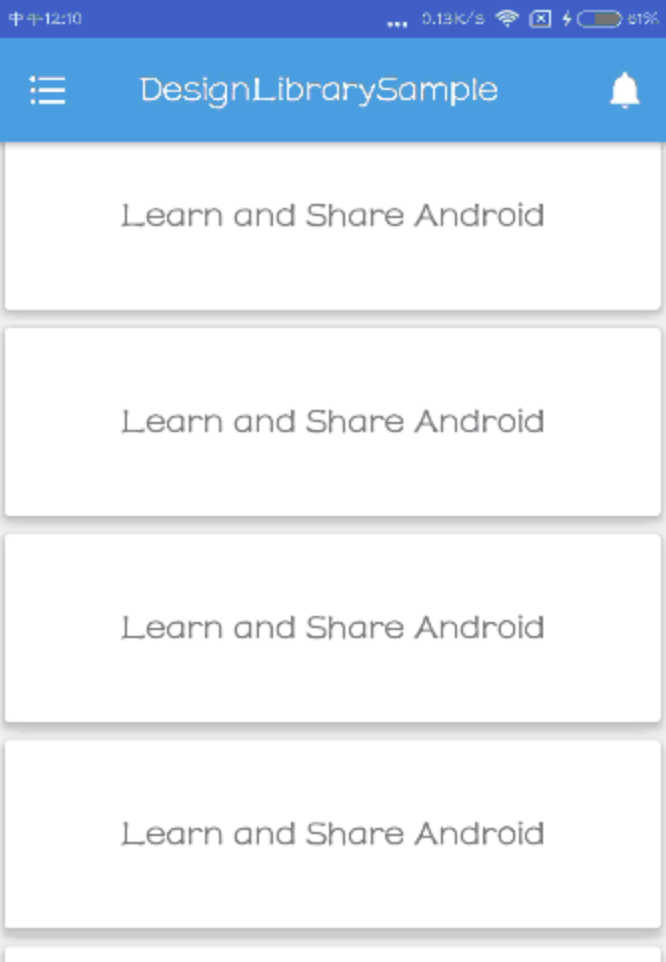
实现折叠式Toolbar：CollapsingToolbarLayout

## 1.简介

在各种不同的应用中，大家可能会经常见到这样一个效果：Toolbar是透明的，有着一个背景图片以及大标题，随着页面向上滑动，其标题逐渐缩放到Toolbar上，而背景图片则在滑动到一定程度后变成了Toolbar的颜色，这种效果也即是折叠式效果。其实这种效果在GitHub上面已经有很多开源库实现了，但是Google在其推出的Design Library库中也给出了一个这种控件，让我们很方便地实现了这种效果。这个控件是CollapsingToolbarLayout，它是一个增强型的FrameLayout。那么，本篇文章就给大家详细地介绍该控件的使用方法以及注意事项。

## 效果

本文结合一个Demo来进行演示，下面是最终的显示效果，也即折叠式Toolbar的效果：

效果1

## 1.引入

使用该控件，需要引入Android Design Library这个库，同时地，我们需要把app的主题也要做相应的修改以便适应这个控件，所以我们也需要appcompat这个库，那么我们在build.gradle文件中引入如下：

dependencies {

compile 'com.android.support:cardview-v7:24.1.0' //cardview

compile 'com.android.support:design:24.1.0'

compile 'com.android.support:appcompat-v7:24.1.0'

}

## 2.常用xml属性介绍

CollapsingToolbarLayout的属性，用来显示自带的文字，可以移动

1)contentScrim：当Toolbar收缩到一定程度时的所展现的主体颜色。即Toolbar的颜色。  
2)title:当titleEnable设置为true的时候，在toolbar展开的时候，显示大标题，toolbar收缩时，显示为toolbar上面的小标题。  
3)scrimAnimationDuration：该属性控制toolbar收缩时，颜色变化的动画持续时间。即颜色变为contentScrim所指定的颜色进行的动画所需要的时间。  
4)expandedTitleGravity：指定toolbar展开时，title所在的位置。类似的还有expandedTitleMargin、collapsedTitleGravity这些属性。  
5)collapsedTitleTextAppearance：指定toolbar收缩时，标题字体的样式，类似的还有expandedTitleTextAppearance。

## 二、常见的标志位

一般开发中，CollapsingToolbarLayout不会单独出现在布局文件中，而是作为另一个控件CoordinatorLayout的子元素出现，那么CoordinatorLayout又是什么呢？其实CoordinatorLayout这个控件很强大，能对其子元素实现多种不同的功能，一个常见的用法就是：给它的一个子元素A设置一个layout\_scrollFlags的属性，然后给另外一个子元素B设置一个layout\_behavior="@string/appbar\_scrolling\_view\_behavior"的属性，这个子元素B一般是一个可以滑动的控件，比如RecyclerView、NestedScrollView等，那么当子元素B滑动的时候，子元素A就会根据其layout\_scrollFlags的属性值而做出不同的改变，所以我们要为CollapsingToolbarLayout设置layout\_scrollFlags属性。

1.layout\_scrollFlags

我们来看看layout\_scrollFlags有哪几个属性可以选择：

1. scroll:所有想要滑动的控件都要设置这个标志位。如果不设置这个标志位，那么View会固定不动。
2. enterAlways:设置了该标志位后，若View已经滑出屏幕，此时手指向下滑，View会立刻出现，这是另一种使用场景。
3. enterAlwaysCollapsed:设置了minHeight，同时设置了该标志位的话，view会以最小高度进度屏幕，当滑动控件滑动到顶部的时候才会拓展为完整的高度。
4. exitUntilCollapsed:向上滑动时收缩当前View。但view可以被固定在顶部。  
   可能直接用语言来描述还是有点太抽象，下面会以实际的效果给大家展示这几个标志位的具体作用。
5. snap 表示在滑动过程中如果停止滑动，则头部会就近折叠（要么恢复原状，要么折叠成一个Toolbar）

上面的这些事组合使用的scroll(必须) | exitUnitlCollapsed | snap(可选)

2.layout\_collapseMode

上面提到CollapsingToolbarLayout是一个FrameLayout，它内部能有多个子元素，而子元素也会有不同的表现。比如说，在上面的GIF图中，toolbar在缩放后是固定在顶部的，而imageview则是随着布局的滚动而滚动，也即存在一个相对滚动的过程。所以这些子元素可以添加layout\_collapseMode标志位进而产生不同的行为。其实这里也只有两种标志位，分别是：

1. pin：有该标志位的View在页面滚动的过程中会一直停留在顶部，比如Toolbar可以被固定在顶部
2. parallax：有该标志位的View表示能和页面同时滚动。与该标志位相关联的一个属性是：layout\_collapseParallaxMultiplier=0.7，该属性是视差因子，表示该View与页面的滚动速度存在差值，造成一种相对滚动的效果。

## 三、常用的层级关系

上面说到CollapsingToolbarLayout一般作为CoordinatorLayout的子元素出现，其实如果要实现上面的效果，还需要另外一个控件：AppBarLayout。该控件也是Design库的控件，作用是把其所有子元素当做一个AppBar来使用。一般来说，实现折叠式Toolbar可以使用以下的层级关系：

<android.support.design.widget.CoordinatorLayout...>

<android.support.design.widget.AppBarLayout...>

<android.support.design.widget.CollapsingToolbarLayout

app:layout\_scrollFlags="scroll|exitUntilCollapsed|snap"

...>

<!-- your collapsed view -->

<Framlayout

app:layout\_collapseMode="parallax"  
 app:layout\_collapseParallaxMultiplier="0.7".../>

<android.support.v7.widget.Toolbar

app:layout\_collapseMode="pin".../>

</android.support.design.widget.CollapsingToolbarLayout>

</android.support.design.widget.AppBarLayout>

<!-- Scroll view -->

<android.support.v7.widget.RecyclerView

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

.../>

</android.support.design.widget.CoordinatorLayout>

从上面的层级关系来看，最外面的一层是CoordinatorLayout，它有两个子元素，分别是AppBarLayout和RecyclerView(可滑动控件)，而AppBarLayout则包裹着CollapsingToolbarLayout，CollapsingToolbarLayout的子元素分别是被折叠的View(可以是一张图片，也可以是一个布局)以及我们的Toolbar。

## 四.例子①

有了以上的知识储备，我们就可以开始动手写代码了，我们的目标是实现上面的gif图的效果。  
1、在activity\_main.xml文件中(注：以下注释只是为了方便说明)：

<android.support.design.widget.CoordinatorLayout

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

xmlns:app="http://schemas.android.com/apk/res-auto" <!-- 自定义命名空间 -->

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<android.support.design.widget.AppBarLayout

android:layout\_width="match\_parent"

android:layout\_height="200dp"

android:fitsSystemWindows="true">

<android.support.design.widget.CollapsingToolbarLayout

android:id="@+id/collapsing\_toolbar\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:contentScrim="?attr/colorPrimary" <!--toolbar折叠后的主体颜色 -->

app:expandedTitleMarginEnd="10dp" <!--文字展开时的Margin -->

app:expandedTitleMarginStart="10dp"

app:collapsedTitleTextAppearance="@style/TextAppearance.AppCompat.Title" <!--字体的表现 -->

app:layout\_scrollFlags="scroll|exitUntilCollapsed">

<ImageView

android:id="@+id/iv"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:scaleType="centerCrop"

app:layout\_collapseMode="parallax" <!--设置imageView可随着滑动控件的滑动而滑动 -->

app:layout\_collapseParallaxMultiplier="0.5"/> <!--视差因子 -->

<android.support.v7.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

android:layout\_height="?attr/actionBarSize"

app:layout\_collapseMode="pin" /> <!--toolbar折叠后固定于顶部 -->

</android.support.design.widget.CollapsingToolbarLayout>

</android.support.design.widget.AppBarLayout>

<android.support.v4.widget.NestedScrollView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"> <!--为滑动控件设置Behavior,这样上面的控件才能做出相应改变 -->

<LinearLayout

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<include layout="@layout/item\_card"/>

<include layout="@layout/item\_card"/>

<include layout="@layout/item\_card"/>

<include layout="@layout/item\_card"/>

<include layout="@layout/item\_card"/>

<include layout="@layout/item\_card"/>

<include layout="@layout/item\_card"/>

<include layout="@layout/item\_card"/>

</LinearLayout>

</android.support.v4.widget.NestedScrollView>

</android.support.design.widget.CoordinatorLayout>

与其相关联的item\_card.xml布局文件：

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

<android.support.v7.widget.CardView

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

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

android:layout\_width="match\_parent"

android:layout\_height="100dp"

android:layout\_margin="5dp"

app:cardElevation="5dp"

app:contentPaddingTop="2dp"

app:contentPaddingBottom="2dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Learn and Share Android"

android:textSize="20sp"

android:layout\_gravity="center"/>

</android.support.v7.widget.CardView>

2、在MainActivity.java文件中再做出一些处理：

private ImageView iv;

private CollapsingToolbarLayout collapsingToolbarLayout;

private Toolbar toolbar;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

collapsingToolbarLayout = (CollapsingToolbarLayout) findViewById(R.id.collapsing\_toolbar\_layout);

iv = (ImageView) findViewById(R.id.iv);

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

setSupportActionBar(toolbar);

toolbar.setNavigationIcon(R.mipmap.ic\_drawer\_home);

collapsingToolbarLayout.setTitle("DesignLibrarySample");

collapsingToolbarLayout.setCollapsedTitleTextColor(Color.WHITE);

collapsingToolbarLayout.setExpandedTitleColor(Color.WHITE);

iv.setImageResource(R.mipmap.ic\_bg);

}

上面，我们为collapsingToolbarLayout设置了标题，以及收缩时标题的颜色和展开时标题的颜色等。经过上面的一个简单例子，就能实现上面gif图所显示的折叠式Toolbar的效果了。

这里先小结一下：在CoordinatorLayout作为父布局的情况下，给滑动控件设置一个layout\_behavior="@string/appbar\_scrolling\_view\_behavior"标志位(该Behavior系统以及帮我们实现)，那么当带有这个标志位的控件滑动的时候会触发带有scroll\_flags标志位的另一个控件进行滑动，此时imageview的layout\_collapseMode是parallax，所以它会以有视差的方式来相对滑动，而toolbar设置了pin的标记位，所以在收缩后会固定在屏幕顶部。

例子②

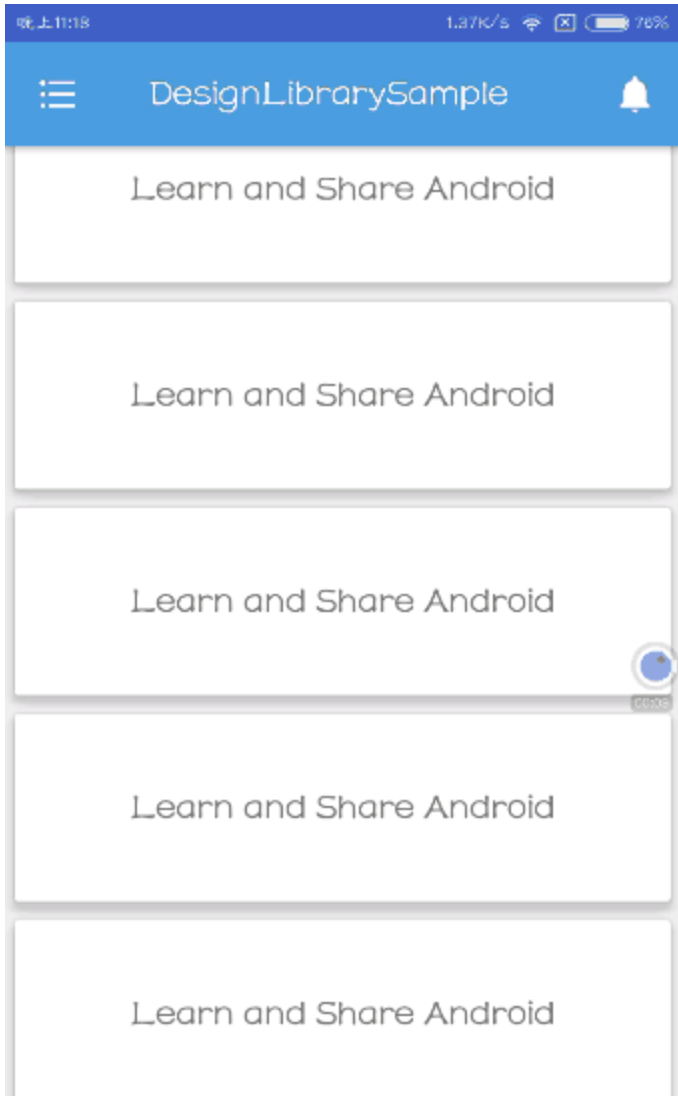
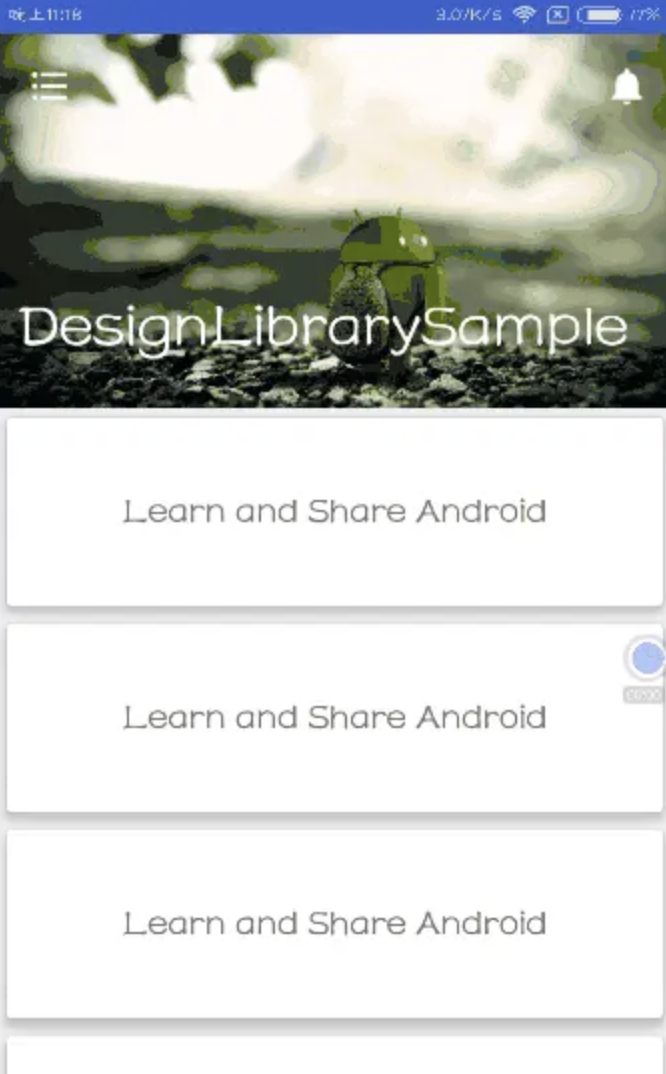
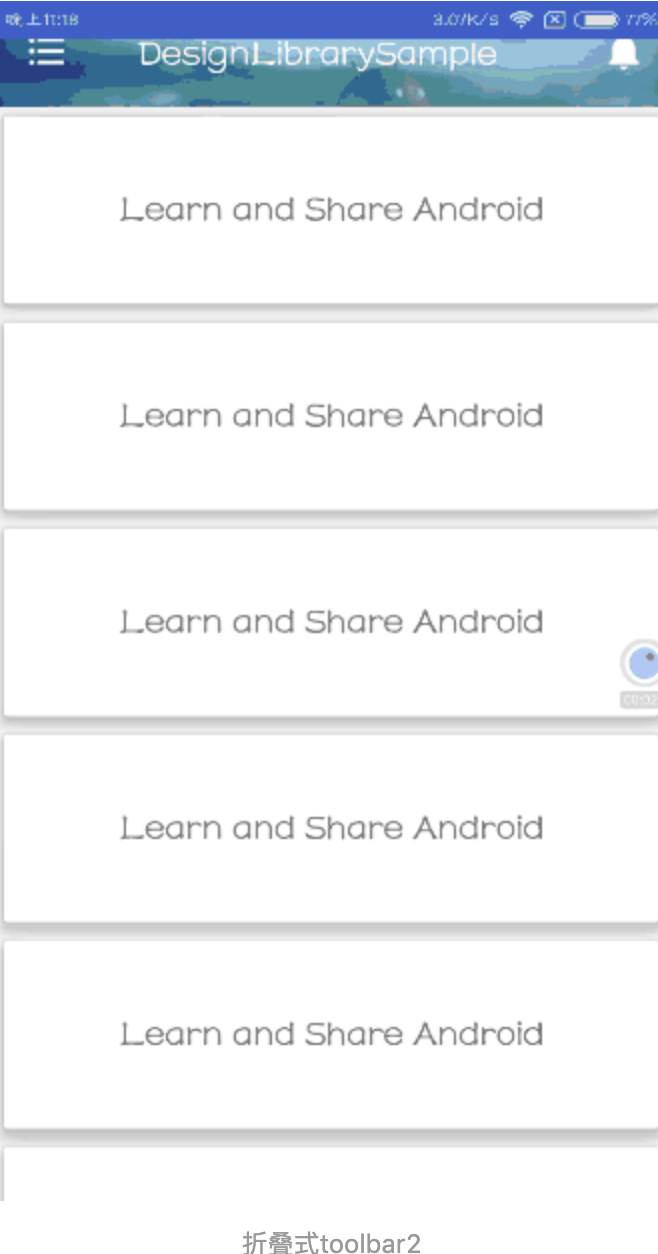
在例子①内，我们为CollapsingToolbarLayout设置的scroll\_flags是"scroll | exitUntilCollapsed"，那么我们把标志位换成别的会有什么不同的效果呢？  
在activity\_main.xml内，作如下修改：

<android.support.design.widget.CollapsingToolbarLayout

...

app:layout\_scrollFlags="scroll|enterAlwaysCollapsed|enterAlways">

然后别的不作改动，效果如下：

折叠式toolbar2

显然，所造成的效果发生了变化，这里toolbar并不一致固定在顶部了，而是随着滑动而滑出了屏幕之外，同时如果手指向下滑动，toolbar会逐渐出现并保持着最小的高度，等到回到了最顶部后，toolbar会展开成原来的样子。  
那么，基于以上的例子，如果上面少了一个“enterAlwaysCollapsed”这个标志位又会怎样呢？该标志位的作用上面也已经解释过了，是控制toolbar以最小的高度进入屏幕，并且在滑动控件滑动到最顶端的时候再展开成完整的高度。如果少了这个标志位，在我们手指向下滑的时候，toolbar也会逐渐出现，但是与上面gif图不同的是，toolbar会继续展开变成原来的样子，即出现imageview。图这里就不放出来了，读者可以自行验证~

通过以上的两个小例子，我们对CollapsingToolbarLayout有了一定的认识，也学会了它的使用方法了，使用它能让我们的应用变得更加美观。那么最后，我们再来谈谈注意事项，也即笔者开发过程中遇到的坑。

## 注意事项

1、Android Design Support Library的使用需要配合特定的主题，一般用AppCompat下的主题即可，也可以自定义主题，继承自AppCompat的主题，否则会报错。另外如果使用Android Studio的话，主题的相关代码需要在styles.xml(v21)文件内做出相应的修改，否则使用Android 5.0以上的机子做测试的话也会报错。  
2、由于使用了AppCompat的主题，那么我们的Activity应该继承自AppCompatActivity。  
3、笔者之前使用design support library的版本号是23.1.0，在此版本上，CollapsingToolbarLayout没有设置collapsedTitleTextAppearance属性，标题可以正常显示，然而到了24.1.0版本，即上面所用的版本，如果没有设置collapsedTitleTextAppearance属性，则当toolbar收缩后，其标题文字变得非常小。所以我们要设置collapsedTitleTextAppearance="@style/TextAppearance.AppCompat.Title"这个属性，才能变得正常。  
4、如果没有为CollapsingToolbarLayout设置一个title，那么会使用ActionBar自带的标题来显示应用的名称，这是因为调用了setSupportActionBar(toolbar)函数。

最后附上代码的地址：[DesignSupportLibrarySample](https://link.jianshu.com?t=https:/github.com/chenyuAndroid/DesignSupportLibrarySample)

链接：<https://www.jianshu.com/p/8ce727308f29>

5.折叠布局下方的布局显示不全

在折叠布局AppBarLayout下方布局外层需要嵌套

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

这个属性使得下方布局在顶部toolbar下方。但是如果下方的RecyclerView或者其他控件取消联动效果 recyclerView.setNestedScrollingEnabled(false)

或者因某些其他原因在顶部不折叠的情况下需要显示到最底部布局，换句话说，在顶部toolbar未折叠情况下，下方RecyclerView或者其他布局无法显示到最后一项。  
如果想要在未折叠情况下显示到最后一项，需要使用自定义的behavior，代码如下

public class FixScrollingFooterBehavior extends AppBarLayout.ScrollingViewBehavior {

private AppBarLayout appBarLayout;

public FixScrollingFooterBehavior() {

super();

}

public FixScrollingFooterBehavior(Context context, AttributeSet attrs) {

super(context, attrs);

}

@Override

public boolean onDependentViewChanged(CoordinatorLayout parent, View child, View dependency) {

if (appBarLayout == null) {

appBarLayout = (AppBarLayout) dependency;

}

final boolean result = super.onDependentViewChanged(parent, child, dependency);

final int bottomPadding = calculateBottomPadding(appBarLayout);

final boolean paddingChanged = bottomPadding != child.getPaddingBottom();

if (paddingChanged) {

child.setPadding(

child.getPaddingLeft(),

child.getPaddingTop(),

child.getPaddingRight(),

bottomPadding);

child.requestLayout();

}

return paddingChanged || result;

}

private int calculateBottomPadding(AppBarLayout dependency) {

final int totalScrollRange = dependency.getTotalScrollRange();

return totalScrollRange + dependency.getTop();

}

}

然后在xml中替换app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"属性为如下

app:layout\_behavior="com.news.detail.FixScrollingFooterBehavior"

链接：https://www.jianshu.com/p/bc56afbd88f2