# Material Design(一)认识与了解

一.What is Material Design

<p>

核心思想

Material design的核心思想，就是把物理世界的体验带进屏幕。去掉现实中的杂质和随机性，保留其最原始纯净的形态、空间关系、变化与过渡，配合虚拟世界的灵活特性，还原最贴近真实的体验，达到简洁与直观的效果。

Material design是最重视跨平台体验的一套设计语言。由于规范严格细致，保证它在各个平台使用体验高度一致。

Material Design是扁平化吗？

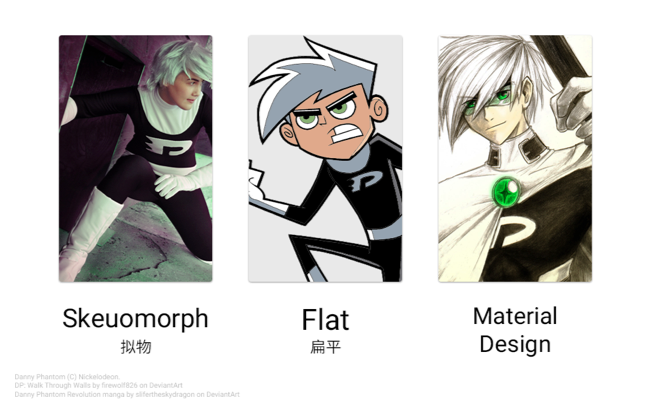
首先我们看下在移动端当下或者曾经出现过的设计风格

拟物化风格 – IOS6及以前与Smartisan OS

扁平风格 – IOS6以后 与WindowsPhone

拟真风格 – 安卓（ Material Design 出现以前）（ “拟真”的本意是，保留控件本身的“立体”属性和光影效果，而去掉了对材质和样式的模仿）

而Material Design则不属于其中的任何一种，如果说拟物化与扁平化是两种极端的话，那么Material Design则是处于拟物化与扁平化之间（更偏向于扁平化）的某个平衡点



Material 顾名思义，材质，材料的意思，那么首先我们说明下Material design中的材料问题

二.材料

<p>

Material design中的材料总结成一句话就是：魔法纸片(纸的形态模拟)

纸片层叠、合并、分离，拥有现实中的厚度、惯性和反馈，同时拥有液体的一些特性，能够自由伸展变形。

<p>

这些是纸片的魔法特性，真实纸片所不具备的能力：

纸片可以伸缩、改变形状

纸片变形时可以裁剪内容，比如纸片缩小时，内容大小不变，而是隐藏超出部分

多张纸片可以拼接成一张

一张纸片可以分裂成多张

纸片可以在任何位置凭空出现

不过，魔法纸片有些效果是禁止的：

一项操作不能同时触发两张纸片的反馈

层叠的纸片，海报高度不能相同

纸片不能互相穿透

纸片不能弯折

纸片不能产生透视，必须平行于屏幕

<p>

前面提到了我对Material Design理解为只扁不平，接下来就解释下为什么

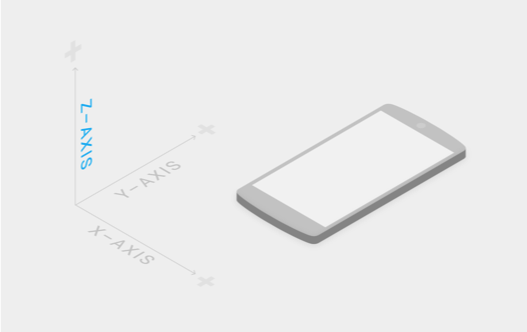
<p>



三.空间

<p>

Material 环境是一个三维的空间，这意味着每个对象都有 x ， y ， z 三维坐标属性，z 轴垂直于显示平面，并延伸向用户视角,每个 material 元素在 z 轴上占据一定的位置并且有一个 1dp 厚度的标准。



只扁不平？

<p>

扁：每个 material 元素在 z 轴上占据一定的位置并且有一个 1dp 厚度的标准

不平：如图所示的Z轴，Z轴维度代表着什么？高度。

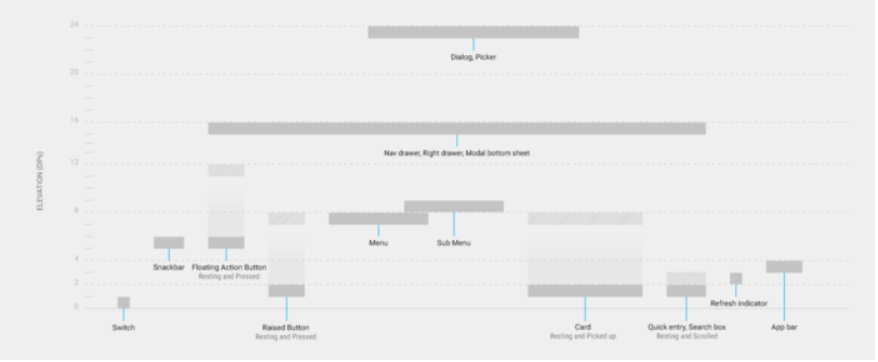
Material 对象都有一个默认高度（静止高度）

对应的Material 某些对象也有一个动态高度，如下：



当一个对象的高度产生变化时，它将会尽快恢复到自身的静止高度

所以高度其实也代表了层次关系



那么我们接下来看看不同高度控件在一起的显示效果

好像并没有高度的显示，所以接下来就要介绍到Material Design中的阴影了



四.阴影

<p>

在 material 环境中，虚拟的光线照射使场景中的对象投射出阴影，主光源投射出一个定向的阴影，而环境光从各个角度投射出连贯又柔和的阴影

“阴影”提供了对象深度和方向性移动的重要视觉线索。它们是唯一一种标示不同平面之间分离程度的视觉线索。

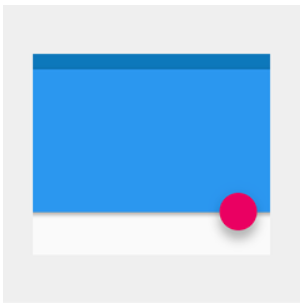
某一对象的“高度”决定了其具体“阴影”的表现形式。

看看加上了阴影后的上图的显示效果

在传统的Android控件加入高度和阴影后会发现很神奇的化学效果

比如说Button加入高度和阴影后，会衍生出不同的几种Button,而且在不同的情况下使用不同的Button

悬浮响应按钮(Floating action button): 点击后会产生墨水扩散效果的圆形按钮。



五.动画

<p>

在Material Design的世界中，动效不仅仅因为美学意义而存才，同时是为了构建有意义的空间关系、凸显功能、维系整个系统的一致性。

1、迅捷响应交互行为

<p>

当用户与应用交互时，所反馈的动效不但极具美感，符合物理逻辑，而且能够愉悦用户。反馈动效的设计必须深思熟虑且具有目的性，而不能随性设计，反馈动效应温和 ，不让用户分心。鼓励用户进一步探索应用

<p>

如下的示例图





作者：前世小书童

链接：https://www.jianshu.com/p/bb083cef5799

來源：简书

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

# Material Design(四)Support Design包的使用

想要使用Android Design Support Library首先得在gradle添加依赖

compile 'com.android.support:design:23.2.0'

## 一.CoordinatorLayout

<p>

CoordinatorLayout是Support Design中新添加的一个增强型的FrameLayout。在CoordinatorLayout中，我们可以在FrameLayout的基础上完成很多新的操作。

CoordinatorLayout 是一个让子 Views 协调工作的布局。每个 View 中肯定是设计和实现了和 CoordinatorLayout 协同工作的，很多子View与CoordinatorLayout配合使用会发生美妙的化学反应，这就是为什么要把CoordinatorLayout作为第一个来介绍

CoordinatorLayout使用如下

<android.support.design.widget.CoordinatorLayout

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="match\_parent"

android:fitsSystemWindows="true">

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

## 二.Floating Action Button

<p>

FAB(Floating Action Button)其实在之前有介绍过,是一个有一些阴影的圆形按钮,这个其实是在Material Design的世界中一个很重要的组成部分。

FAB使用如下

<android.support.design.widget.FloatingActionButton

android:id="@+id/fab"

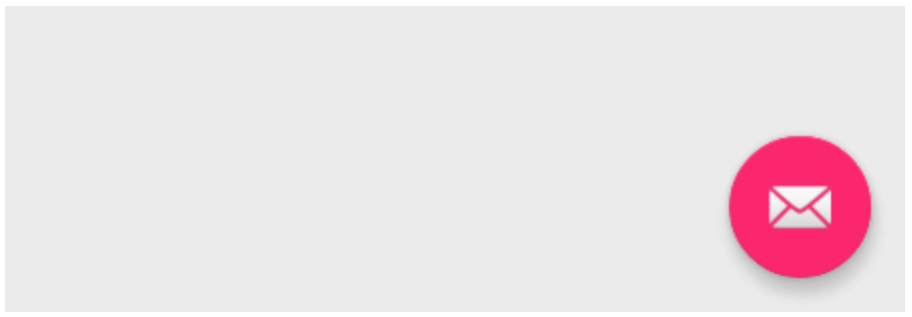
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="bottom|end"

android:layout\_margin="@dimen/fab\_margin"

android:src="@android:drawable/ic\_dialog\_email" />



## 三.Snackbar

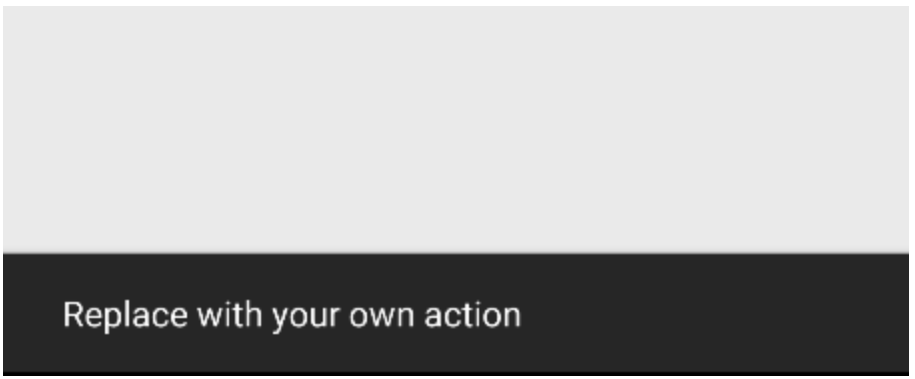
<p>

Snackbar是在屏幕的地步一个微小的黑色条显示着一条简短的消息，与Toast 有着相同的概念，但是不同于 Toast，它的表现是作为 UI 的一部分而不是覆盖在屏幕上。

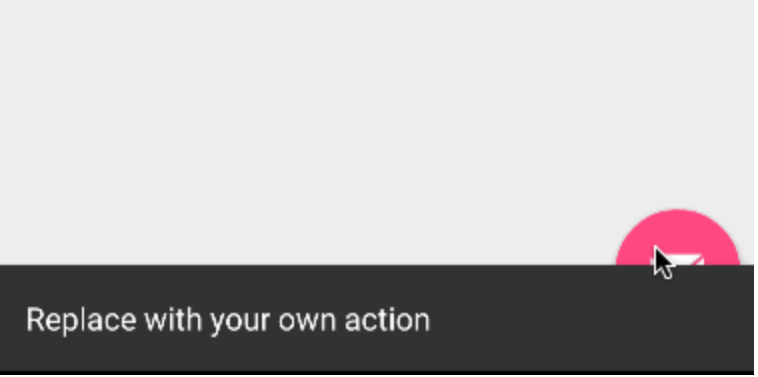
Snackbar使用如下

Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH\_LONG)

.setAction("Action", null).show();



当我们使用普通的layout包裹FAB，我们会发现



但是使用CoordinatorLayout包裹FAB就可以发现如下的效果

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

<android.support.design.widget.CoordinatorLayout 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="match\_parent"

android:fitsSystemWindows="true">

<android.support.design.widget.FloatingActionButton

android:id="@+id/fab"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="bottom|end"

android:layout\_margin="@dimen/fab\_margin"

android:src="@android:drawable/ic\_dialog\_email" />

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



## 四.Toolbar

<p>

Toolbar其实是用来替代ActionBar的，因为在这个Support Design中，很多组件都是设计为与Toolbar协同使用，从而达到Material Design设计中的一些效果

要使用Toolbar需要如下的工作

1.在AppTheme的style属性中隐藏Action Bar

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

<item name="windowActionBar">false</item>

<item name="windowNoTitle">true</item>

</style>

1. 在CoordinatorLayout里面放入AppBarLayout与Toolbar

<android.support.design.widget.CoordinatorLayout

...>

<android.support.design.widget.AppBarLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:fitsSystemWindows="true"

android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

<android.support.v7.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

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

android:background="?attr/colorPrimary"

app:popupTheme="@style/ThemeOverlay.AppCompat.Light"

app:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar" />

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

<android.support.design.widget.FloatingActionButton

...>

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

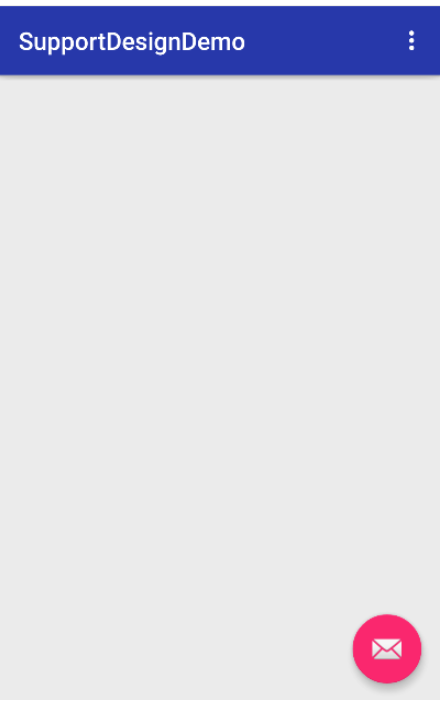
</android.support.design.widget.CoordinatorLayout>

1. 代码设置将Toolbar作为ActionBar

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

setSupportActionBar(toolbar);

运行如下



注意事项

1.内容区域

内容区域放置于AppBarLayout与FloatingActionButton之间

并且一些古老的原因，LinearLayout 没有被设计成与 CoordinatorLayout 协同工作。在这样的情况下，没有任何布局用来包裹 LinearLayout，使它像 Toolbar 的做法那样。但它是更加容易的，你只需要在 LinearLayout 添加一个属性告诉它的滚动行为

如下

<LinearLayout

...

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

...>

2.当随着内容滚动，让AppBarLayout退出屏幕

怎么让内容滚动，有几种方式，RecycleView,ListView,ScrollView,但是ListView与ScrollView并没有设计为与CoordinatorLayout协同工作，所以Support Design提供了一个新的ScrollView为NestedScrollView，所以RecycleView与NestedScrollView可以用来随着内容滚动，让AppBarLayout退出屏幕。

AppBarLayout可以这样响应滚动事件：根据子view的滚动标志（scroll flag）来控制它们如何进入（滚入屏幕）与退出（滚出屏幕）。

Flag包括：

scroll: 所有想滚动出屏幕的view都需要设置这个flag- 没有设置这个flag的view将被固定在屏幕顶部。

enterAlways: 这个flag让任意向下的滚动都会导致该view变为可见，启用快速“返回模式”。

app:layout\_scrollFlags="scroll|enterAlways"



enterAlwaysCollapsed: 顾名思义，这个flag定义的是何时进入（已经消失之后何时再次显示）。假设你定义了一个最小高度（minHeight）同时enterAlways也定义了，那么view将在到达这个最小高度的时候开始显示，并且从这个时候开始慢慢展开，当滚动到顶部的时候展开完。

app:layout\_scrollFlags="scroll|enterAlwaysCollapsed"

android:minHeight="?attr/actionBarSize"

exitUntilCollapsed: 同样顾名思义，这个flag时定义何时退出，当你定义了一个minHeight，这个view将在滚动到达这个最小高度的时候消失。



## CollapsingToolbarLayout



其实在Material Design的设计中上面这张图的效果是挺典范的一种设计，那么这样的效果怎么实现的呢？其实上面的Toolbar也可以展开折叠，但是不够完美，这里介绍下CollapsingToolbarLayout

使用步骤如下

1.用 CollapsingToolbarLayout 包裹 Toolbar，但仍然在 AppBarLayout 中

2.从 Toolbar 中删除 layout\_scrollFlags,为 CollapsingToolbarLayout 声明 layout\_scrollFlags，并且将 layout\_scrollFlags 设置成 scroll|exitUntilCollapsed

3.改变 AppBarLayout 扩张状态时的布局高度大小

最终代码如下

<android.support.design.widget.AppBarLayout

android:layout\_width="match\_parent"

android:layout\_height="256dp"

android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

<android.support.design.widget.CollapsingToolbarLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:contentScrim="?attr/colorPrimary"

app:expandedTitleMarginEnd="64dp"

app:expandedTitleMarginStart="48dp"

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

<android.support.v7.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

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

app:layout\_collapseMode="pin"

/>

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

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



可以通过一些属性修改展开状态改变标题文字的位置，以及文本的显示

app:expandedTitleMargin 展开状态的边距

app:expandedTitleMarginBottom 展开状态距下的边距

app:expandedTitleMarginEnd 展开状态结束的边距

app:expandedTitleMarginStart 展开状态开始的边距

app:collapsedTitleTextAppearance 折叠状态的文本显示

app:expandedTitleTextAppearance 展开状态的文本显示

我们可以看到Toolbar icons 仍然滚出了屏幕。我们可以声明这个属性给 Toolbar 来固定住它，让它总是在屏幕的顶部。

<android.support.v7.widget.Toolbar

...

app:layout\_collapseMode="pin"/>

怎么实现图片中的效果呢，步骤如下

1.添加一个ImageView作为Toolbar的背景图层

2.添加视差模式与视差系数来使视觉上看起来效果更好

3.为CollapsingToolbarLayout添加属性可以让收缩的时候自动的变化到普通的颜色

最后的代码如下

<android.support.design.widget.AppBarLayout

android:layout\_width="match\_parent"

android:layout\_height="256dp"

android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

<android.support.design.widget.CollapsingToolbarLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:contentScrim="?attr/colorPrimary"

app:expandedTitleMarginEnd="64dp"

app:expandedTitleMarginStart="48dp"

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

<ImageView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:scaleType="centerCrop"

android:src="@drawable/header"

android:fitsSystemWindows="true"

app:layout\_collapseMode="parallax"

app:layout\_collapseParallaxMultiplier="0.7"/>

<android.support.v7.widget.Toolbar

android:id="@+id/toolbar"

android:layout\_width="match\_parent"

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

/>

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

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

最后效果如下



## TetInputLayout

在material design中，即便是简单的 EditText ，也有提升的余地。通常，单独的EditText会在用户输入第一个字母之后隐藏提示信息，但是现在你可以使用TextInputLayout 来将EditText封装起来，提示信息会变成一个显示在EditText之上的floating label，这样用户就始终知道他们现在输入的是什么。

要做到这一点，只需要简单的用 TextInputLayout 包裹住一个 EditText就可以了

如下

<android.support.design.widget.TextInputLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="input2" />

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



还可以通过调用setError()在EditText下面显示一条错误信息

显示如下



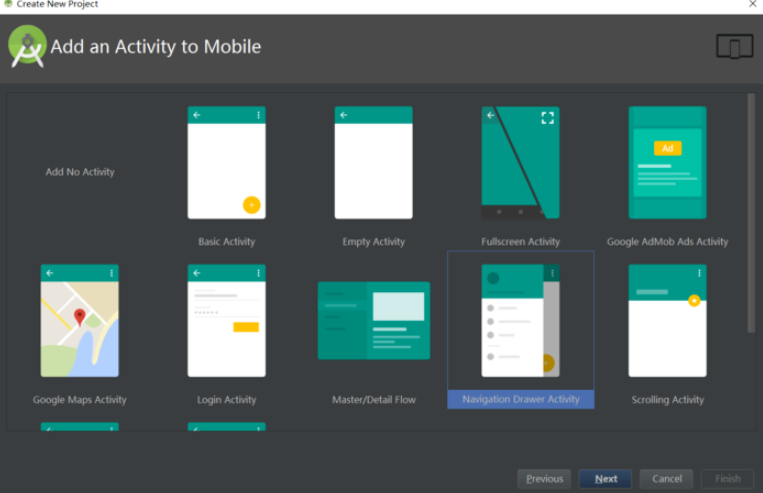
## 七,Navigation Drawer使用

Navigation Drawer是在 Material Design 中推出的一种侧滑导航菜单栏控件。包含两个部分，一部分是侧滑（DrawerLayout），一部分是导航菜单栏（NavigationView）。

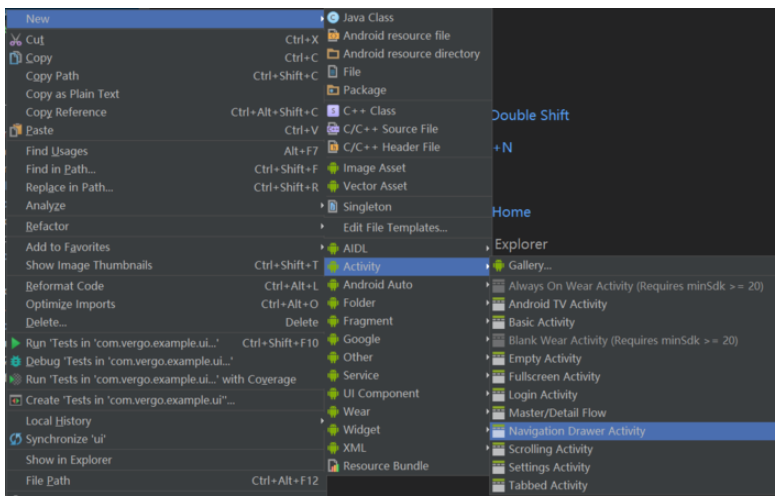
AS新建

利用Android Studio可以快速建立这个控件

在新建项目时，在最后选择Navigation Drawer Activity



在新建Activity时，选择Navigation Drawer Activity



DrawerLayout

DrawerLayout布局中，由两部分组成，一部分是内容布局，一部分是侧滑菜单布局。其中侧滑菜单布局通过设置 android:layout\_gravity 属性，来控制是左侧滑，还是右侧滑。参考实例代码

<android.support.v4.widget.DrawerLayout 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"

tools:context=".ui.MainActivity">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<android.support.v7.widget.Toolbar

android:id="@+id/main\_toolbar"

android:layout\_width="match\_parent"

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

android:background="@color/colorPrimary"

app:popupTheme="@style/AppTheme.PopupOverlay"/>

<TextView

android:id="@+id/content\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="@dimen/activity\_horizontal\_margin"

android:text="Hello World!"/>

</LinearLayout>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_gravity="start"

android:gravity="center"

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

android:text="导航菜单页左"/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_gravity="end"

android:gravity="center"

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

android:text="导航菜单页右"/>

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



DrawerLayout中也有一些常会用到的方法，来控制例如打开、关闭菜单，监听滑动事件等，这里暂时列举一些，还是得多看api和源码。

//打开左侧菜单，同理右侧就是 GravityCompat.END

drawerLayout.openDrawer(GravityCompat.START);

//关闭左侧菜单，同理右侧就是 GravityCompat.END

drawerLayout.closeDrawer(GravityCompat.START);

//设置抽屉打开时，剩余挡住内容区域部分的颜色

drawerLayout.setScrimColor(Color.parseColor("#4cff0000"));

//设置抽屉锁定模式 LOCK\_MODE\_LOCKED\_OPEN:锁定 无法滑动

drawerLayout.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_LOCKED\_OPEN, GravityCompat.START);

//监听滑动事件

drawerLayout.addDrawerListener(new DrawerLayout.DrawerListener() {

@Override

public void onDrawerSlide(View drawerView, float slideOffset) {

//抽屉滑动时回调

}

@Override

public void onDrawerOpened(View drawerView) {

//抽屉打开后回调

}

@Override

public void onDrawerClosed(View drawerView) {

//抽屉关闭后回调

}

@Override

public void onDrawerStateChanged(int newState) {

//抽屉滑动状态改变时回调

switch (newState) {

case DrawerLayout.STATE\_DRAGGING:

//拖动状态

break;

case DrawerLayout.STATE\_IDLE:

//静止状态

break;

case DrawerLayout.STATE\_SETTLING:

//设置状态

break;

default:

break;

}

}

});

NavigationView

NavigationView是兼容包中提供用来实现导航菜单控件。使用menu资源填充数据，可以更简单高效的实现导航菜单。

添加依赖

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

布局中引用

android.support.v4.widget.DrawerLayout 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:id="@+id/drawerlayout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".ui.MainActivity">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<android.support.v7.widget.Toolbar

android:id="@+id/main\_toolbar"

android:layout\_width="match\_parent"

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

android:background="@color/colorPrimary"

app:popupTheme="@style/AppTheme.PopupOverlay"/>

<TextView

android:id="@+id/content\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="@dimen/activity\_horizontal\_margin"

android:text="Hello World!"/>

</LinearLayout>

<android.support.design.widget.NavigationView

android:id="@+id/navigation\_view"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_gravity="start"

app:headerLayout="@layout/navigation\_drawer\_header"

app:menu="@menu/navigation\_drawer\_menu"/>

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

NavigationView分为两部分，一部分是headerLayout，一部分是menu。headerLayout就是对应菜单的顶部部分，一般用来显示用户信息什么的，menu则对应实际的菜单选项。

定义headerLayout

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

android:layout\_width="match\_parent"

android:layout\_height="200dp"

android:background="@color/colorPrimary">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_margin="10dp"

android:text="HeaderLayout"

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

android:textSize="18sp" />

</RelativeLayout>

定义menu

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

<group android:checkableBehavior="single">

<item

android:id="@+id/item\_dync"

android:icon="@mipmap/ic\_menu\_dync\_selected"

android:title="首页" />

<item

android:id="@+id/item\_explore"

android:icon="@mipmap/ic\_menu\_explore\_selected"

android:title="发现" />

<item

android:id="@+id/item\_message"

android:icon="@mipmap/ic\_menu\_message\_selected"

android:title="消息" />

<item

android:id="@+id/item\_person"

android:icon="@mipmap/ic\_menu\_person\_selected"

android:title="我的" />

</group>

<item android:title="其他">

<menu>

<item

android:id="@+id/subitem\_01"

android:icon="@mipmap/ic\_launcher"

android:title="分享" />

<item

android:id="@+id/subitem\_02"

android:icon="@mipmap/ic\_launcher"

android:title="设置" />

<item

android:id="@+id/subitem\_03"

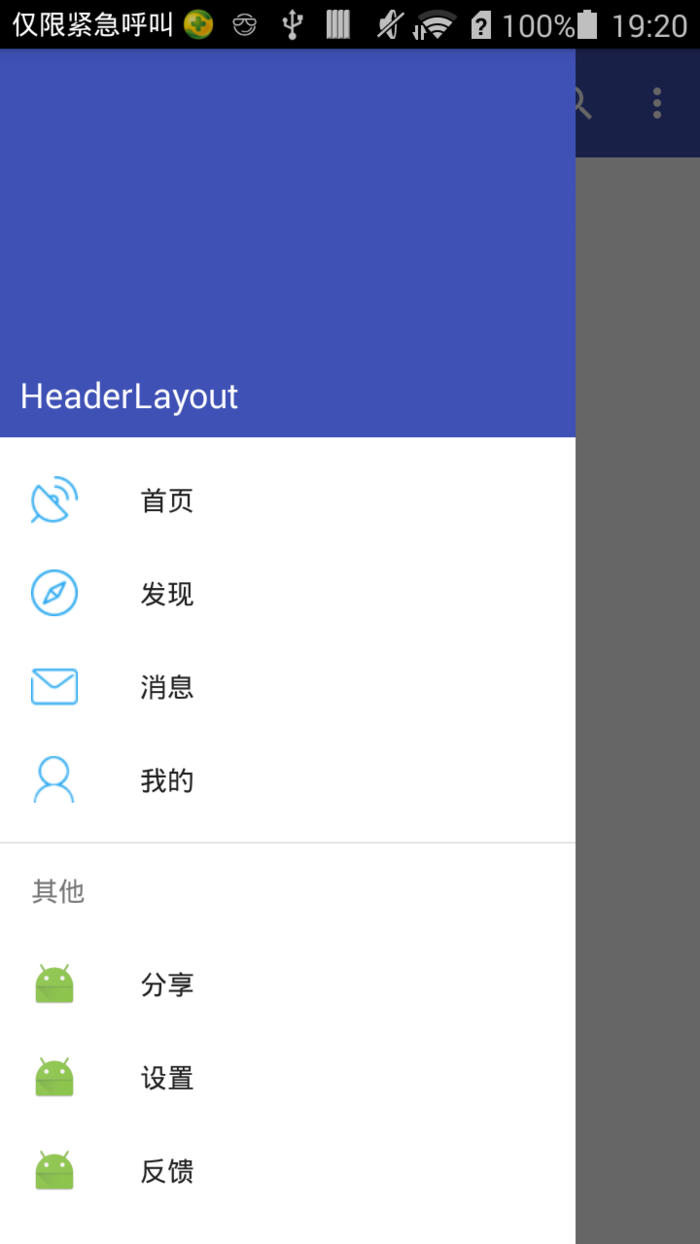
android:icon="@mipmap/ic\_launcher"

android:title="反馈" />

</menu>

</item>

</menu>



Menu Item 的点击监听

直接使用 NavigationView 的 setNavigationItemSelectedListener() 方法即可

navigationView.setNavigationItemSelectedListener(new NavigationView.OnNavigationItemSelectedListener() {

@Override

public boolean onNavigationItemSelected(MenuItem item) {

int itemId = item.getItemId();

switch (itemId) {

case R.id.item\_dync:

break;

case R.id.item\_explore:

break;

//......

}

return true;

}

});

https://www.jianshu.com/p/8da6a8ccb66e

https://www.jianshu.com/p/c1f21dc2e45f

http://design.1sters.com/

https://www.jianshu.com/p/c1f21dc2e45f