导航菜单的制作方式多种多样，网上也有各种炫酷效果的具体实现方式，那么今天我主要是想来说说Google在Android5.0之后推出的NavigationView的具体使用方式。

NavigationView在很多App上都已经可以看到其效果图，国内的比如知乎（侧拉菜单滑出来的那一部分属于NavigationView），如下图：



还有Google自己的那些app，也基本全都采用了NavigationView，如下图，分别是Gmail、Google Map以及Google Play：



OK，看完了图，接下来我们就来说说这个NavigationView吧。

**1.NavigationView是什么**

long long ago，我们做抽屉菜单的时候，左边滑出来的那一部分的布局都是由我们自己来定义的，自己写的话，花点时间也能做出来好看的侧拉菜单，但总是要耗费时间，于是Google在5.0之后推出了NavitationView，就是我们左边滑出来的那个菜单。这个菜单整体上分为两部分，上面一部分叫做HeaderLayout，下面的那些点击项都是menu，这样的效果如果我们要自己写肯定也能写出来，但是没有必要，既然Google提供了这个控件，那我们就来看看这个控件要怎么用吧。

**2.NavigationView怎么用**

和普通的侧拉菜单制作方式一样，首先所有的东西还是都放在一个DrawerLayout中（如果你对DrawerLayout的使用还不熟悉，请参考这篇文章[使用DrawerLayout实现侧拉菜单](http://blog.csdn.net/u012702547/article/details/49562747)），只不过这次我们把左边滑出菜单的布局用一个NavigationView来代替，代码如下：

**[java]** [view plain](http://blog.csdn.net/u012702547/article/details/51253222) [copy](http://blog.csdn.net/u012702547/article/details/51253222)

[print?](http://blog.csdn.net/u012702547/article/details/51253222)

1. <?xml version="1.0" encoding="utf-8"?>
2. <android.support.v4.widget.DrawerLayout
3. xmlns:android="http://schemas.android.com/apk/res/android"
4. xmlns:app="http://schemas.android.com/apk/res-auto"
5. xmlns:tools="http://schemas.android.com/tools"
6. android:layout\_width="match\_parent"
7. android:layout\_height="match\_parent"
8. tools:context="org.mobiletrain.drawerlayout.MainActivity">
10. <LinearLayout
11. android:layout\_width="match\_parent"
12. android:layout\_height="match\_parent"
13. android:orientation="vertical">
15. <TextView
16. android:layout\_width="wrap\_content"
17. android:layout\_height="wrap\_content"
18. android:text="主页面"/>
19. </LinearLayout>
21. <android.support.design.widget.NavigationView
22. android:id="@+id/navigation\_view"
23. android:layout\_width="wrap\_content"
24. android:layout\_height="match\_parent"
25. android:layout\_gravity="left"
26. android:fitsSystemWindows="true"
27. app:headerLayout="@layout/header\_layout"
28. app:menu="@menu/main"></android.support.design.widget.NavigationView>
29. </android.support.v4.widget.DrawerLayout>

OK，下面我来分别解释一下这里边几个属性的含义：

1.[**Android**](http://lib.csdn.net/base/android):layout\_gravity="left"属性表示该View是左边的滑出菜单，这个属性的含义不用多说，这是DrawerLayout使用方式中的知识点。

2.app:headerLayout="@layout/header\_layout"表示引用一个头布局文件，这个头就是我们在上面看到的那个背景图片，包括背景图片上面的显示用户名的控件等等。

3.app:menu="@menu/main"表示引用一个menu作为下面的点击项

OK，那我们再来看看头布局文件：

**[java]** [view plain](http://blog.csdn.net/u012702547/article/details/51253222) [copy](http://blog.csdn.net/u012702547/article/details/51253222)

[print?](http://blog.csdn.net/u012702547/article/details/51253222)

1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3. android:layout\_width="match\_parent"
4. android:layout\_height="200dp"
5. android:orientation="vertical">
7. <ImageView
8. android:id="@+id/iv"
9. android:layout\_width="match\_parent"
10. android:layout\_height="match\_parent"
11. android:scaleType="centerCrop"
12. android:src="@drawable/p1"/>
13. </LinearLayout>

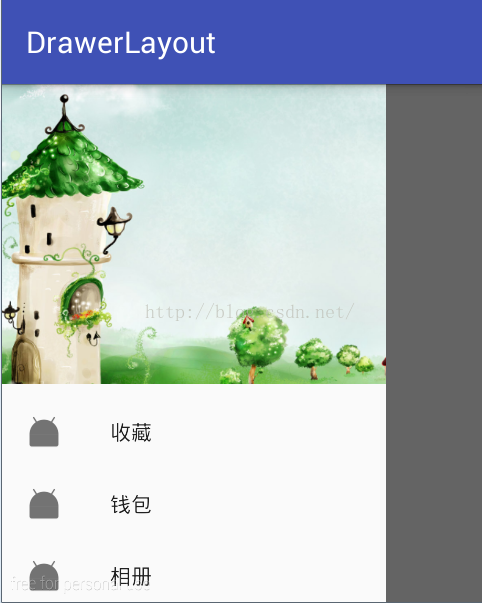
再来看看menu文件：

**[java]** [view plain](http://blog.csdn.net/u012702547/article/details/51253222) [copy](http://blog.csdn.net/u012702547/article/details/51253222)

[print?](http://blog.csdn.net/u012702547/article/details/51253222)

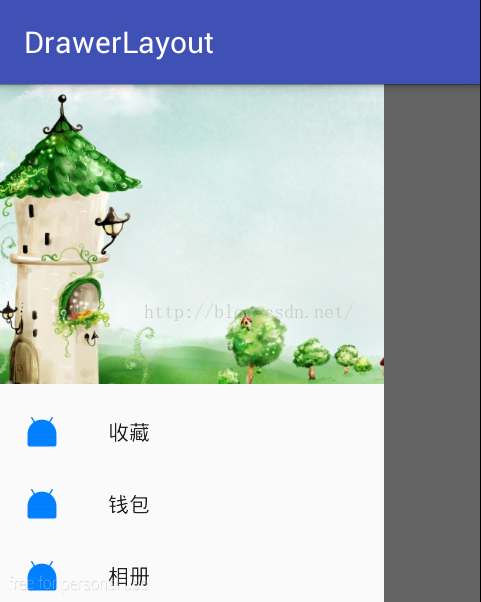
1. <?xml version="1.0" encoding="utf-8"?>
2. <menu xmlns:android="http://schemas.android.com/apk/res/android">
3. <item
4. android:id="@+id/favorite"
5. android:icon="@mipmap/ic\_launcher"
6. android:title="收藏"/>
7. <item
8. android:id="@+id/wallet"
9. android:icon="@mipmap/ic\_launcher"
10. android:title="钱包"/>
11. <item
12. android:id="@+id/photo"
13. android:icon="@mipmap/ic\_launcher"
14. android:title="相册"/>
15. <item
16. android:id="@+id/file"
17. android:icon="@mipmap/ic\_launcher"
18. android:title="文件"/>
19. </menu>

OK，运行来看看效果：



OK，已经显示出来了，但是有一个问题，图片都变为灰色了，怎么破？有两种方式：

1.在布局文件中添加app:itemIconTint="@color/blue"属性，表示设置图片的颜色全都为蓝色，效果如下：



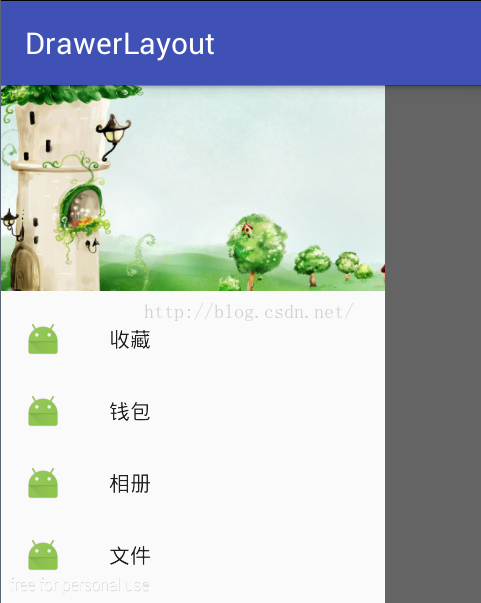
2.第一种解决方案会让所有的图片以一种颜色来显示，如果我想让图片就是显示他本身的颜色该怎么办呢？在[**Java**](http://lib.csdn.net/base/java)代码中调用如下方法：

**[java]** [view plain](http://blog.csdn.net/u012702547/article/details/51253222) [copy](http://blog.csdn.net/u012702547/article/details/51253222)

[print?](http://blog.csdn.net/u012702547/article/details/51253222)

1. NavigationView navigationView = (NavigationView) findViewById(R.id.navigation\_view);
2. navigationView.setItemIconTintList(**null**);

显示效果如下：



这下正常了。

还有下面两个常用的API：

1.app:itemBackground="@color/colorAccent"设置每一个item的背景颜色

2.app:itemTextColor=""设置item的背景颜色

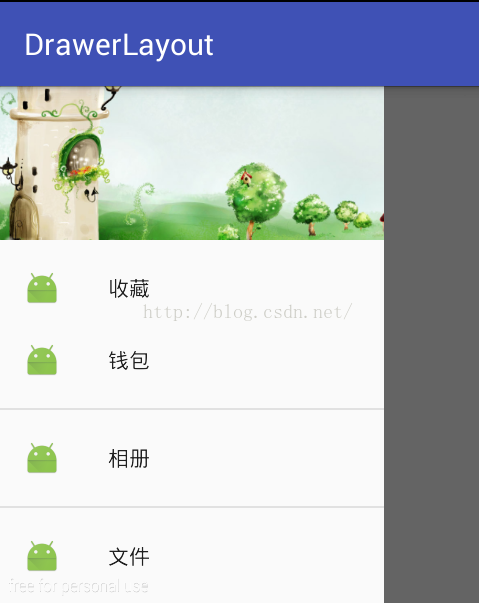
OK，如果我想在NavigationView的item之间添加上一条分隔线呢？很简单，只需要在menu中将相应的item放到一个group中，并给该group取一个id即可，代码如下：

**[java]** [view plain](http://blog.csdn.net/u012702547/article/details/51253222) [copy](http://blog.csdn.net/u012702547/article/details/51253222)

[print?](http://blog.csdn.net/u012702547/article/details/51253222)

1. <?xml version="1.0" encoding="utf-8"?>
2. <menu xmlns:android="http://schemas.android.com/apk/res/android">
3. <group android:id="@+id/g1">
4. <item
5. android:id="@+id/favorite"
6. android:icon="@mipmap/ic\_launcher"
7. android:title="收藏"/>
8. <item
9. android:id="@+id/wallet"
10. android:icon="@mipmap/ic\_launcher"
11. android:title="钱包"/>
12. </group>
13. <group android:id="@+id/g2">
14. <item
15. android:id="@+id/photo"
16. android:icon="@mipmap/ic\_launcher"
17. android:title="相册"/>
18. </group>
19. <item
20. android:id="@+id/file"
21. android:icon="@mipmap/ic\_launcher"
22. android:title="文件"/>
23. </menu>

显示效果如下：



OK，分割线添加成功了。

接下来我们来看看怎么处理NavigationView中的事件监听。

NavigationView中的事件处理主要是两个方面，一个头部的点击事件，还有一个是itemView的点击事件，下面我们分别来看看：

**1.头部点击事件**

处理头部点击事件，我们需要先获得到头部控件，在Java代码中我们可以通过下面的方式获得头部控件，

**[java]** [view plain](http://blog.csdn.net/u012702547/article/details/51253222) [copy](http://blog.csdn.net/u012702547/article/details/51253222)

[print?](http://blog.csdn.net/u012702547/article/details/51253222)

1. //获取头布局文件
2. View headerView = navigationView.getHeaderView(0);

然后通过调用headerView中的findViewById方法来查找到头部的控件，设置点击事件即可。

**2.item点击事件**

**[java]** [view plain](http://blog.csdn.net/u012702547/article/details/51253222) [copy](http://blog.csdn.net/u012702547/article/details/51253222)

[print?](http://blog.csdn.net/u012702547/article/details/51253222)

1. navigationView.setNavigationItemSelectedListener(**new** NavigationView.OnNavigationItemSelectedListener() {
2. @Override
3. **public** **boolean** onNavigationItemSelected(MenuItem item) {
4. //在这里处理item的点击事件
5. **return** **true**;
6. }
7. });

**ActionBarDrawerToggle**   
drawerLayout左侧菜单（或者右侧）的展开与隐藏可以被DrawerLayout.DrawerListener的实现监听到，不过还是建议用ActionBarDrawerToggle来监听，ActionBarDrawerToggle实现了DrawerListener，所以他能做DrawerListener可以做的任何事情，同时他还能将drawerLayout的展开和隐藏与actionbar的app 图标关联起来，当展开与隐藏的时候图标有一定的平移效果，点击图标的时候还能展开或者隐藏菜单。   
activity代码添加代码如下

ActionBarDrawerToggle actionBarDrawerToggle = new ActionBarDrawerToggle(

this,

drawerLayout,

toolbar,

R.string.drawer\_open,

R.string.drawer\_close

){

@Override

public void onDrawerOpened(View drawerView) {

super.onDrawerOpened(drawerView);

//打开

}

@Override

public void onDrawerClosed(View drawerView) {

super.onDrawerClosed(drawerView);

//关闭

}

};

drawerLayout.setDrawerListener(actionBarDrawerToggle);

actionBarDrawerToggle.syncState();