Material Design

2018年5月8日 9:17

Toolbar

继承了ActionBar所有功能，而且灵活性高，可以配合其他控件来完成一些Material Design效果

activity_main.xml：

```
<android.support.v7.widget.Toolbar
android:id="@+id/toolbar"
android:layout_width="match_parent"
android:layout_height="?attr/actionBarSize" //高度为actionBar的高度
android:background="?attr/colorPrimary"
android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/> //将弹出的菜单项指定成淡色主题
```

MainActivity：

```
Toolbartoolbar=(Toolbar)findViewById(R.id.toolbar); //得到实例
setSupportActionBar(toolbar); //将ToolBar实例传入
```

AndroidManifest.xml

```
<activity
android:name=".MainActivity"
android:label="Fruits"> //指定在ToolBar中显示的文字内容
```

menu >>> toolbar.xml

<item>标签来定义action按钮，id指定按钮的id，icon指定按钮的图标，title指定按钮的文字

```
<item
android:id="@+id/backup"
android:icon="@drawable/ic_backup"
android:title="Backup"
app:showAsAction="always"/>
<item
android:id="@+id/delete"
android:icon="@drawable/ic_delete"
android:title="Delete"
app:showAsAction="ifRoom"/>
<item
android:id="@+id/settings"
android:icon="@drawable/ic_settings"
android:title="Settings"
app:showAsAction="never"/>
```

MainActivity：

```
public boolean onCreateOptionsMenu(Menu menu){
getMenuInflater().inflate(R.menu.toolbar,menu);
return true;
}    //加载toolbar.xml菜单文件
```

```
@Override   //处理各个按钮的点击事件
public boolean onOptionsItemSelected(MenuItemitem){
switch(item.getItemId()){
caseR.id.backup:
Toast.makeText(this,"YouclickedBackup",Toast.LENGTH_SHORT).show();
break;
caseR.id.delete:
Toast.makeText(this,"YouclickedDelete",Toast.LENGTH_SHORT).show();
break;
caseR.id.settings:
Toast.makeText(this,"YouclickedSettings",Toast.LENGTH_SHORT).show();
break;
default:
}
returntrue;
}
```

滑动菜单

DrawerLayout

将一些菜单选项隐藏起来，不放在主屏幕上，通过滑动的方式将菜单显示出来

activity_main.xml：

```
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:id="@+id/drawer_layout"
android:layout_width="match_parent"
android:layout_height="match_parent">
```

主屏幕中显示的内容：

```
<FrameLayout
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="?attr/actionBarSize"
android:background="?attr/colorPrimary"
android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>

</FrameLayout>
```

滑动菜单中显示的内容：

```
<TextView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_gravity="start" //一定要指定，表示滑动的菜单是在屏幕的左边还是右边
android:text="Thisismenu"
android:textSize="30dp"
android:background="#FFF"/>
```

在Toolbar的最左边加入一个导航按钮，点击了按钮将滑动菜单的内容展示出来（第二种方法打开滑动菜单）

MainActivity：

```
mDrawerLayout=(DrawerLayout)findViewById(R.id.drawer_layout); //得到DrawerLayout实例
ActionBar actionBar = getSupportActionBar(); //得到ActionBar实例
if(actionBar!=null){
actionBar.setDisplayHomeAsUpEnabled(true); //显示导航按钮
actionBar.setHomeAsUpIndicator(R.drawable.ic_menu); //设置导航按钮图标
}
```

```
case android.R.id.home: //HomeAsUp按钮的id
mDrawerLayout.openDrawer(GravityCompat.START); //openDrawer () 将滑动菜单展示出来
break;
```

NavigationView

优化滑动菜单页面

在app/build.grade文件中添加依赖

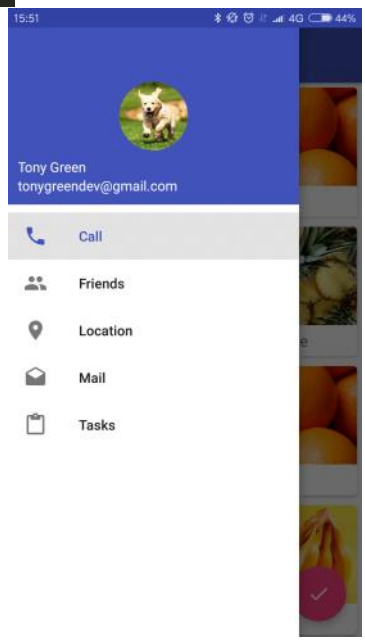
```
compile'com.android.support:design:24.2.1' //Design Support库
compile'de.hdodenhof:circleimageview:2.1.0' //开源项目，实现图片圆形化功能
```

menu：在NavigationView中显示具体的菜单项

headerLayout：在NavigationView中显示头部布局

menu >>> nav_menu.xml

```
<group android:checkableBehavior="single"> //group表示一个组，checkableBehavior属性表述组中的菜单项只能单选
<item
android:id="@+id/nav_call"
android:icon="@drawable/nav_call"
android:title="Call"/>
<item
android:id="@+id/nav_friends"
android:icon="@drawable/nav_friends"
android:title="Friends"/>
<item
android:id="@+id/nav_location"
android:icon="@drawable/nav_location"
android:title="Location"/>
<item
android:id="@+id/nav_mail"
android:icon="@drawable/nav_mail"
android:title="Mail"/>
<item
android:id="@+id/nav_task"
android:icon="@drawable/nav_task"
android:title="Tasks"/>
</group>
```



layout >>> nav_header.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="180dp"
android:background="?attr/colorPrimary" //背景色为colorPrimary
android:padding="10dp">
```

```
<de.hdodenhof.circleimageview.CircleImageView //将图片圆形化的控件
android:id="@+id/icon_image"
android:layout_width="70dp"
android:layout_height="70dp"
android:layout_centerInParent="true"
android:src="@drawable/nav_icon"/>
```

```
<TextView
android:id="@+id/username"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentBottom="true"
```

```
android:text="tonygreendev@gmail.com"
android:textColor="#FFF"
android:textSize="14sp"/>

<TextView
android:id="@+id/mail"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_above="@id/username"
android:text="TonyGreen"
android:textColor="#FFF"
android:textSize="14sp"/>

</RelativeLayout>
```

activity_main.xml：

```
<android.support.design.widget.NavigationView
android:id="@+id/nav_view"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_gravity="start"
app:headerLayout="@layout/nav_header"
app:menu="@menu/nav_menu"/>
```

MainActivity：处理菜单项的点击事件

```
NavigationView navView = (NavigationView)findViewById(R.id.nav_view); //获取NavigationView实例
navView.setCheckedItem(R.id.nav_call);  //将Call菜单项设置为默认选中
navView.setNavigationItemSelectedListener(new NavigationView.OnNavigationItemSelectedListener(){ //菜单项选中事件的监听器
@Override //当用户点击任意菜单项时，在此方法中写相应的逻辑处理
public boolean onNavigationItemSelected(MenuItem item){
mDrawerLayout.closeDrawers(); //关闭滑动菜单
return true;
}
});
```

悬浮按钮和可交互提示

FloatingActionButton（跟普通Button用法一样）

实现悬浮按钮的效果，默认使用colorAccent作为按钮的颜色，给按钮指定一个图标来表明按钮的作用

activity_main.xml：

```
<android.support.design.widget.FloatingActionButton
android:id="@+id/fab"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="bottom|end" //控件位于屏幕的右下角
android:layout_margin="16dp"
android:src="@drawable/ic_done"
app:elevation="8dp" /> //给FloatingActionButton指定一个高度值
```

MainActivity：设置点击事件

```
FloatingActionButton fab = (FloatingActionButton)findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener(){
@Override
public void onClick(View view){
    Toast.makeText(MainActivity.this,"FAB clicked",Toast.LENGTH_SHORT).show();
}
});
```

snackbar（提示工具，可以额外增加一个按钮的点击事件）

Toast的作用是告诉用户现在发生了是么事情，但同时用户只能被动接收这个事情，用户没办法进行选择

Snackbar允许在提示当中加入一个可交互按钮，让用户点击按钮的时候可以执行一些额外的逻辑操作

MainActivity：

```
Snackbar.make(view,"Datadeleted",Snackbar.LENGTH_SHORT)
.setAction("Undo",newView.OnClickListener(){   //创建一个Snackbar对象，调用 setAction () 设置一个动作
@Override
public void onClick(View v){
Toast.makeText(MainActivity.this,"Datarestored",Toast.LENGTH_SHORT).show();
}
})
.show(); //让Snackbar显示出来
```

CoordinatorLayout（加强版的FrameLayout）

监听其所有子控件的各种事件，然后自动做出最为合理的响应

activity_main.xml：替换原来的FrameLayout

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

（FloatingActionButton是CoordinatorLayout的子控件，所以可以监听到，snackbar的make（第一个参数是FloatingActionButton的））

卡片式布局

让页面中的元素看起来就像在卡片中一样，并且还能拥有圆角和投影

CardView

CardView额外提供了圆角和阴影等效果，有立体的感觉（也是一个FrameLayout）

```
<android.support.v7.widget.CardView
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:cardCornerRadius="4dp" //指定卡片圆角的弧度
android:elevation="5dp">

<TextView
android:id="@+id/info_text"
android:layout_width="match_parent"
android:layout_height="wrap_content"/>

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

----------高配版的水果列表效果----------

- app/build.gradle 文件中添加依赖库

```
compile'com.android.support:recyclerview-v7:26.1.0'
compile'com.android.support:cardview-v7:26.1.0'
compile'com.github.bumptech.glide:glide:3.7.0' //实现复杂的图片加载功能，实现加载水果图片
```

- 修改activity_main.xml

```
<android.support.v7.widget.RecyclerView
android:id="@+id/recycler_view"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

- RecyclerView的实现
  - 新建一个Fruit实体类

    //name表示水果的名字，imageId表示水果对应图片的资源

```
public Fruit(String name,int imageId){
this.name = name;
this.imageId = imageId;
}
```

  - 为RecyclerView的子项指定一个布局 layout>>>fruit_item.xml

```
<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="wrap_content"
android:layout_margin="5dp"
app:cardCornerRadius="4dp">

<LinearLayout
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="wrap_content">

<ImageView
android:id="@+id/fruit_image"
android:layout_width="match_parent"
android:layout_height="100dp"
android:scaleType="centerCrop"/> //指定图片的缩放模式，保持原有比例填充满ImageView，并将超出屏幕的部分裁剪掉

<TextView
android:id="@+id/fruit_name"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
android:layout_margin="5dp"
android:textSize="16sp"/>
</LinearLayout>

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

  - 为RecyclerView准备一个适配器（新建FruitAdapter类）

```
publicclassFruitAdapterextendsRecyclerView.Adapter<FruitAdapter.ViewHolder>{

static class ViewHolder extends RecyclerView.ViewHolder{

public ViewHolder(Viewview){    }
}

public FruitAdapter(List<Fruit>fruitList){   }

@Override
public ViewHolder onCreateViewHolder(ViewGroup parent,int viewType){    }

@Override
```

```
public void onBindViewHolder(ViewHolderholder,intposition){
Glide.with(mContext).load(fruit.getImageId()).into(holder.fruitImage);//Glide.with()传入参数，load（）去加载图片，into（）设置具体的ImageView中
  }

@Override
public int getItemCount(){    }
    }
}
```

- 修改MainActivity

```
public class MainActivity extends AppCompatActivity{

private Fruit[] fruits = {    };

@Override
protected void onCreate(Bundle savedInstanceState){
initFruits();
RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
GridLayoutManager layoutManager = new GridLayoutManager(this,2);
recyclerView.setLayoutManager(layoutManager);
adapter = new FruitAdapter(fruitList);
recyclerView.setAdapter(adapter);
}

private void initFruits(){
fruitList.clear();
for(int i=0;i<50;i++){
Random random = new Random();
int index = random.nextInt(fruits.length);
fruitList.add(fruits[index]);
  }
}
}
```

AppBarLayout（实际上是一个垂直方向的LinearLayout）

解决RecyclerView覆盖ToolBar的问题

1、将ToolBar嵌套到AppBarLayout中

2、给RecyclerView指定一个布局行为

activity_main.xml:

```
<android.support.design.widget.AppBarLayout
android:layout_width="match_parent"
android:layout_height="wrap_content">

<android.support.v7.widget.Toolbar
android:id="@+id/toolbar"
android:layout_width="match_parent"
android:layout_height="?attr/actionBarSize"
android:background="?attr/colorPrimary"
android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
app:layout_scrollFlags="scroll|enterAlways|snap"/> //随着滚动RecyclerView，Toolbar随之隐藏和显示
</android.support.design.widget.AppBarLayout>

<android.support.v7.widget.RecyclerView
android:id="@+id/recycler_view"
android:layout_width="match_parent"
android:layout_height="match_parent"
app:layout_behavior="@string/appbar_scrolling_view_behavior"/> //指定一个布局行为
```

下拉刷新

swipeRefreshLayout

activity_main.xml:

```
<android.support.v4.widget.SwipeRefreshLayout
android:id="@+id/swipe_refresh"
android:layout_width="match_parent"
android:layout_height="match_parent"
app:layout_behavior="@string/appbar_scrolling_view_behavior">

<android.support.v7.widget.RecyclerView
android:id="@+id/recycler_view"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
</android.support.v4.widget.SwipeRefreshLayout>
```

修改MainActivity:

```
swipeRefresh=(SwipeRefreshLayout)findViewById(R.id.swipe_refresh); //得到实例
swipeRefresh.setColorSchemeResources(R.color.colorPrimary);//设置下拉刷新进度条的颜色
swipeRefresh.setOnRefreshListener(newSwipeRefreshLayout.OnRefreshListener(){
@Override
public void onRefresh(){ //设置下拉刷新的监听器
```

```
refreshFruits();
}
});

private void refreshFruits(){
new Thread(newRunnable(){
@Override
public void run(){
try{
Threaa.sleep(2000);
}catch(InterruptedExceptione){
e.printStackTrace();
}
runOnUiThread(new Runnable(){
@Override
public void run(){
initFruits(); //重新生成数据
adapter.notifyDataSetChanged(); //通知数据发生了变化
swipeRefresh.setRefreshing(false);
}
});
}
}).start();
}
```

可折叠式标题栏

CollapsingToolbarLayout（作用于Toolbar基础上的布局）

只能作为AppBarLayout的直接布局（AppBarLayout是coordinatorLayout的子布局）

创建一个FruitActivity（编写水果详情展示界面的布局）

修改activity_fruit.xml

CoordinatorLayout >>> AppBarLayout >>> CollapsingToolbarLayout (ImageView 、 Toolbar) >>> NestedScrollView (LinearLayout >>CardVIew >>TestView) >>> FloatingActionButton

修改FruitActivity

1、在onCreate（）方法中，通过Intent获取到传入的水果名和水果图片的资源id，然后通过findViewById（）得到控件的实例。再使用Toolbar的标准用法，作为ActionBar显示，并启用HomeAsUp按钮。

2、填充界面上的内容，调用CollapsingToolbarLayout的setTitle（）方法将水果名设置成当前界面的标题

然后使用Glide加载传入的水果图片，并设置到标题栏的ImageView上面

填充水果的内容详情，（generateFruitContent（）将水果名循环拼接500次，设置在TextView上）

在onOptionsItemSelected（）方法中处理HomeAsUp按钮的点击事件（点击之后，关闭当前活动，从而返回上一个活动）

修改FruitAdapter（处理RecyclerView的点击事件）

给CardView注册一个点击事件监听器，获取当前点击项的水果名和水果资源id，传入到Intent中，最后调用startActivity（）方法启动FruitActivity

将背景图和系统状态栏融合，android：fitsSystemWindows

在coordinatorLayout、AppBarLayout、CollapsingToolbarLayout嵌套结构的布局中，将控件android：fitsSystemWindows属性指定成true

>>>

在程序的主题中将状态栏颜色指定成透明色：android：statusBarColor = "@android:color/transparent"

>>>

res===values-v21===style.xml