Material Design笔记

1：Toolbar专题

1.1：简介

从API11（Android 3.0）开始，所有使用默认主题的activity都用ActionBar行使app bar功能。但是，app bar的特性在不同android发布版本中被逐渐添加到ActionBar中，因此，ActionBar

在不同Android系统版本中会表现不一样。相较之下，最新的特性被添加到兼容包中的Toolbar中，因此，只要你的应用能使用兼容包，就可以使用Toolbar，而且Toolbar在不同的设备中表现一致。例如：Actionbar只有在API>=21(Android 5.0)时才有Material Design特性，而*兼容包中的Toolbar，只要应用能用兼容包中的Toolbar，就可以拥有Material Design特性。*

*1.2：向activity中添加Toolbar*

*1：添加v7 appcompat包*

*2：使你的activity继承自AppCompatActivity（注意：要让所有有Toolbar的activity都要 继承AppCompatActivity）*

*3：在manifest中，改变application的主题为NoActionBar，这样可以防止使用默认的 ActionBar。例如：*

android:theme="@style/Theme.AppCompat.Light.NoActionBar"

*4：将Toolbar添加到布局中，例如*

<android.support.v7.widget.Toolbar  
    android:id="@+id/my\_toolbar"  
    android:layout\_width="match\_parent"  
    android:layout\_height="?attr/actionBarSize"  
    android:background="?attr/colorPrimary"  
    android:elevation="4dp"  
    android:theme="@style/ThemeOverlay.AppCompat.ActionBar"  
    app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>

*备注：elevation为组件高度， Material Design中推荐app bar高度为4dp。*

*将它放在布局最顶端，因为你想让他行使app bar功能。*

*5：在activity中设置用此Toolbar来行使app bar的功能，代码如下*

**Toolbar myToolbar = (Toolbar) findViewById(R.id.my\_toolbar);  
    setSupportActionBar(myToolbar);**

*注意：通过以上步骤，你只是拥有了一个简单的action bar。默认情况下，此时app bar只有一个应用的名称，你可以向其中添加其他按钮，参考下一节。*

*1.3：app bar中添加其他动作按钮*

*1.3.1：简介*

*app bar允许你添加最重要的动作按钮，来控制the current context right at the top of the app。app bar的空间是有限的，如果添加的按钮超过了app bar的容量，app bar会将多 余的按钮移到overflow menu中。当然你也可以手动指定将特定按钮移到overlfow menu中。*

*1.3.2：添加动作按钮*

*动作按钮定义在res/menu/中，例子如下，每一个<item>代表一个动作按钮：*

<menu xmlns:android="http://schemas.android.com/apk/res/android" >  
  
    <!-- "Mark Favorite", should appear as action button if possible -->  
    <item  
        android:id="@+id/action\_favorite"  
        android:icon="@drawable/ic\_favorite\_black\_48dp"  
        android:title="@string/action\_favorite"  
        app:showAsAction="ifRoom"/>  
  
    <!-- Settings, should always be in the overflow -->  
    <item android:id="@+id/action\_settings"  
          android:title="@string/action\_settings"  
          app:showAsAction="never"/>  
  
</menu>

*备注：app:showAsAction：动作按钮是否展现在app bar上，还是overflow menu上。*

*1.3.3：为动作按钮添加事件*

*当你点你动作按钮时，系统会调用onOptionsItemSelected()回调函数，其中MenuItem表示你点击了哪个按钮，通过MenuItem.getItemId()可以获取按钮Id。例子如下：*

@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action\_settings:  
            // User chose the "Settings" item, show the app settings UI...  
            return true;  
  
        case R.id.action\_favorite:  
            // User chose the "Favorite" action, mark the current item  
            // as a favorite...  
            return true;  
  
        default:  
            // If we got here, the user's action was not recognized.  
            // Invoke the superclass to handle it.  
            return super.onOptionsItemSelected(item);  
  
    }  
}

*1.4：Toolbar中添加Up按钮*

*1.4.1：简介*

*你的应用应该能方便的让用户返回主界面。一种很简单的方式就是在除了主activity的所有其他activity中的Toolbar中添加Up按钮。*

*1.4.2：声明Parent Activity*

*要使用此功能，你需要声明Parent Activity，你可以在manifest设置android:parentActivityName属性。*

*注意：此属性只存在于API >=16（Android 4.1）中，要兼容以前版本，需要定义<meta-data>，例子如下：*

<application ... >  
    ...  
  
    <!-- The main/home activity (it has no parent activity) -->  
  
    <activity  
        android:name="com.example.myfirstapp.MainActivity" ...>  
        ...  
    </activity>  
  
    <!-- A child of the main activity -->  
    <activity  
        android:name="com.example.myfirstapp.MyChildActivity"  
        android:label="@string/title\_activity\_child"  
        **android:parentActivityName="com.example.myfirstapp.MainActivity"** >  
  
        <!-- Parent activity meta-data to support 4.0 and lower -->  
        <meta-data  
            **android:name="android.support.PARENT\_ACTIVITY"  
            android:value="com.example.myfirstapp.MainActivity" />**  
    </activity>  
</application>

*1.4.3：使能Up按钮*

*你需要在activity创建时调用setDisplayHomeAsUpEnabled()方法。例子如下：*

@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity\_my\_child);  
  
    // my\_child\_toolbar is defined in the layout file  
    Toolbar myChildToolbar =  
        (Toolbar) findViewById(R.id.my\_child\_toolbar);  
    setSupportActionBar(myChildToolbar);  
  
    // Get a support ActionBar corresponding to this toolbar  
    ActionBar ab = getSupportActionBar();  
  
    // Enable the Up button  
    **ab.setDisplayHomeAsUpEnabled(true);**  
}

*注意：你不需要在onOptionsItemSelected()回调方法中监听Up点击事件，因为他会由父类处理，根据manifest中的设置。如下，如果return true，则Up点击没有任何效果。*

@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
 return super.onOptionsItemSelected(item);  
}

*1.5：Action Views and Action Providers*

*1.5.1：简介*

*Action View：is an action that providers rich functionality within the app bar，例如搜索框*

*Action Provider：是一个有自己布局的action，the action initially appears as a button or menu item,but when the user clicks the action, the action provider controls the action's behaviour int any way you want to define.*

*Android兼容包中提供了一些专业的action view和action provider组件。例如：SearchView组件实现了Action View用来提供搜索功能。ShareActionProvider组件实现了Action Provider用来提供分享功能。*

*1.5.2：添加Action View*

*和添加Action Button一样，在menu中添加<item>，不过要额外设置一些属性：*

*以下两个属性二选一即可：actionViewClass或者actionLayout。*

*showAsAction属性可以额外添加一种选择：collapseActionView，该属性设置当用户没有 和该组件交互时该如何显示。当没有交互时，显示为图标；当用户点击时，会扩展 开来，在app bar上显示布局。*

*例子如下：*

<item android:id="@+id/action\_search"  
     android:title="@string/action\_search"  
     android:icon="@drawable/ic\_search"  
     **app:showAsAction="ifRoom|collapseActionView"**  
     **app:actionViewClass="android.support.v7.widget.SearchView" />**

*1.5.2.1：自定义action view动作*

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.main\_activity\_actions, menu);  
  
    **MenuItem searchItem = menu.findItem(R.id.action\_search);**  
    **SearchView searchView =  
            (SearchView) MenuItemCompat.getActionView(searchItem);**  
  
    // Configure the search info and add any event listeners...  
  
    return super.onCreateOptionsMenu(menu);  
}

*1.5.2.2：当用户和action view交互时（action view展开或者收缩时），添加自定义事件*

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.options, menu);  
    // ...  
  
    // Define the listener  
    OnActionExpandListener expandListener = new OnActionExpandListener() {  
        @Override  
        public boolean onMenuItemActionCollapse(MenuItem item) {  
            // Do something when action item collapses  
            return true;  // Return true to collapse action view  
        }  
  
        @Override  
        public boolean onMenuItemActionExpand(MenuItem item) {  
            // Do something when expanded  
            return true;  // Return true to expand action view  
        }  
    };  
  
    // Get the MenuItem for the action item  
    MenuItem actionMenuItem = menu.findItem(R.id.myActionItem);  
  
    // Assign the listener to that action item  
    MenuItemCompat.setOnActionExpandListener(actionMenuItem, expandListener);  
  
    // Any other things you have to do when creating the options menu…  
  
    return true;  
}

*1.5.3：添加Action Provider*

*要声明一个action provider，和action button一样在menu中添加<item>，设置actionProviderClass属性，例子如下，添加一个ShareActionProvider：*

<item android:id="@+id/action\_share"  
    android:title="@string/share"  
    app:showAsAction="ifRoom"  
    app:actionProviderClass="android.support.v7.widget.ShareActionProvider"/>

*然后在onCreateOptionsMenu()方法中添加：*

MenuItem shareItem = menu.findItem(R.id.action\_share);  
 ShareActionProvider myShareActionProvider =  
     (ShareActionProvider) MenuItemCompat.getActionProvider(shareItem);

Intent myShareIntent = new Intent(Intent.ACTION\_SEND);  
 myShareIntent.setType("image/\*");  
 myShareIntent.putExtra(Intent.EXTRA\_STREAM, myImageUri);

myShareActionProvider.setShareIntent(myShareIntent);

*注意：当分享内容变化时，需要改变intent：*

// Image has changed! Update the intent:  
 myShareIntent.putExtra(Intent.EXTRA\_STREAM, myNewImageUri);  
 myShareActionProvider.setShareIntent(myShareIntent);

*更多关于Action Provider，请参考Action Provider专题。*

*2：弹出式信息（类似鼠标右键的效果）*

*2.1：简介*

*很多时候你想向用户传递一些信息，但不需要用户的任何反馈（用户只需要看到信息而不需要对信息进行操作），比如删除邮件等。这些信息要足够醒目，同时不能影响用户对应用进行操作。*

*Android提供了Snackbar来解决此问题。Snackbar取代了Toast，虽然Toast现在仍然可以使用，但是推荐使用Snackbar。*

*2.2：用Snackbar来展示信息*

*2.2.1：注意：当Snackbar放在一般的视图层次中时，其只提供基础的功能。但是如果放在CoordinatorLayout中，Snackbar会获得额外的特性，如下：*

*用户可以取消Snackbar by swiping it away。*

*当Snackbar出现时，布局会移动其他组件。例如，如果布局有一个FloatingActionButton， 布局会将此Button抬高当Snackbar出现时，而不是直接展现在FloatingActionButton 上面。*

*CoordinatorLayout是FrameLayout的扩展。如果你的app使用了FrameLayout，你可以直接将其替换来获得Snackbar的全部功能特性。如果你使用的不是FrameLayout，最简单的方法就是将其整个放在CoordinatorLayout中，如下：*

<android.support.design.widget.CoordinatorLayout  
    android:id="@+id/myCoordinatorLayout"  
    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">  
  
    <!-- Here are the existing layout elements, now wrapped in  
         a CoordinatorLayout -->  
    <LinearLayout  
        android:layout\_width="match\_parent"  
        android:layout\_height="match\_parent"  
        android:orientation="vertical">  
  
        <!-- …Toolbar, other layouts, other elements… -->  
  
    </LinearLayout>  
  
</android.support.design.widget.CoordinatorLayout>

*注意：要在CoordinatorLayout中设置android:id属性，这个会在展示信息时使用。*

*2.2.2：创建Snackbar对象*

Snackbar mySnackbar = Snackbar.make(viewId, stringId, duration);

*viewId：将viewId放在哪儿，the method actually searches up the view hierarchy from the passed viewId until it reaches either a CoordinatorLayout,or the window decor's content view.通常 情况下，最简单的方式就是直接将CoordinatorLayout的id传进去。*

*stringId：信息的resource ID.This can be formatted or unformatted text.*

*duration：信息展示时长。*

*2.2.3：展示信息给用户*

mySnackbar.show();

*系统不会同时展示多个Snackbar，如果当前有Snackbar在展示中，会将新的Snackbar放到队列中依次展示。*

*注意：如果你只是想把信息展示给用户，而不需要额外的功能，你可以采用如下方式，该方式可保证reference被快速回收：*

Snackbar.make(findViewById(R.id.myCoordinatorLayout), R.string.email\_sent,  
                        Snackbar.LENGTH\_SHORT)  
        .show();

*2.2.3：为信息添加动作*

*要为Snackbar添加动作，你需要实现View.OnClickListener接口。当用户点击时，系统会调用onClick()方法，例子如下，添加一个undo动作：*

public class MyUndoListener implements View.OnClickListener{  
  
    &Override  
    public void onClick(View v) {  
  
        // Code to undo the user's last action  
    }  
}

*然后，将该监听器应用，如下：*

Snackbar mySnackbar = Snackbar.make(findViewById(R.id.myCoordinatorLayout),  
                                R.string.email\_archived, Snackbar.LENGTH\_SHORT);  
**mySnackbar.setAction(R.string.undo\_string, new MyUndoListener());**  
mySnackbar.show();

*注意：Snackbar出现一段时间后会自动消失，不要指望用户一定会有机会点击到该button，因此，你应该在应用中其他地方提供相应功能。*

*3：使用app bar提供的工具方法*

*一旦你使用Toolbar行使app bar功能，你就可以使用兼容包中ActionBar类提供的工具方法。例如隐藏展示app bar。*

*要使用这些工具方法，调用getSupportActionBar()方法。该方法返回ActionBar的引用，你可以调用ActionBar的任何方法来调整app bar。*