在Android3.0之后，Google对UI导航设计上进行了一系列的改革，其中有一个非常好用的新功能就是引入的ActionBar，他用于取代3.0之前的标题栏，并提供更为丰富的导航效果。

一、添加ActionBar

1、不需要导入ActionBar，因为在android3.0或者以上版本中，ActionBar已经默认的包含在Activity中  
2、在Android3.0的默认主题“holographic”中，已经创造了ActionBar  
3、当android:minSdkVersion 或者 android:targetSdkVersion 属性被设置成11或者更高时，应用会被认为是Android3.0版本（默认含有ActionBar）

二、隐藏ActionBar

1、我们可以通过把主题设置为Theme.Holo.NoActionBar为一个Activity去除掉ActionBar。

<activity android:theme="@android:style/Theme.Holo.NoActionBar">

2、我们也可以通过在代码中调用show()或者hide()方法来相应的显示或者隐藏ActionBar

3、在我们使用actionBar.hide();方法是系统默认会将ActionBar占用的空间分配给界面，此时显示的界面将会重新绘制。  
同样调用actionBar.show();时也会重新绘制界面。如果我们一个程序需要频繁的显示或者隐藏ActionBar的话，这样  
就不会有良好的效果。Google提供给一个属性可以让ActionBar浮在界面上，当然你可以让ActionBar的背景为透明，  
这样会有良好的体验效果。

三、添加ActionBar的Item

1、通过程序动态布局

package com.xys.actionbar;

import android.os.Bundle;

import android.app.ActionBar;

import android.app.Activity;

import android.view.Menu;

import android.view.MenuItem;

public class MainActivity extends Activity {

ActionBar actionBar;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

actionBar=getActionBar();

actionBar.show();

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

super.onCreateOptionsMenu(menu);

//添加菜单项

MenuItem add=menu.add(0,0,0,"add");

MenuItem del=menu.add(0,0,0,"del");

MenuItem save=menu.add(0,0,0,"save");

//绑定到ActionBar

add.setShowAsAction(MenuItem.SHOW\_AS\_ACTION\_IF\_ROOM);

del.setShowAsAction(MenuItem.SHOW\_AS\_ACTION\_IF\_ROOM);

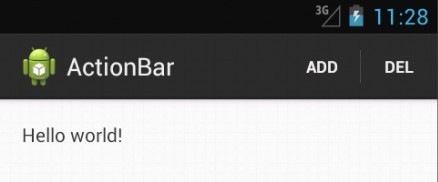
save.setShowAsAction(MenuItem.SHOW\_AS\_ACTION\_IF\_ROOM);

return true;

}

}

效果如图：



2、通过Menu.xml文件布局

menu/menu.xml

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

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

<item

android:id="@+id/item1"

android:orderInCategory="1"

android:showAsAction="ifRoom|withText"

android:title="@string/save"/>

<item

android:id="@+id/item1"

android:orderInCategory="2"

android:showAsAction="ifRoom|withText"

android:title="@string/edit"/>

</menu>

程序：

public class ActionBarXMLActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

}

@Override

public boolean onCreateOptionsMenu(Menu menu)

{

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.optionsmenu, menu);

return true;

}

}

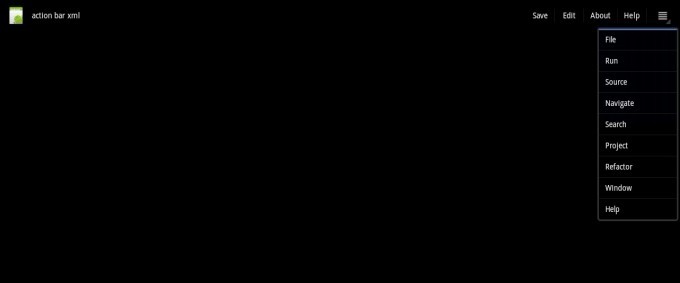
效果和上面一样

四、关于菜单项过多

当菜单项过多时，Android会分两种情况进行处理：

1、手机有MENU实体键：则按下Menu键后会显示剩余菜单项

2、手机没有MENU实体键：则会在最左边显示一个Action OverFlow按钮，按下后会显示剩余项菜单，如图：



五、使用带字的Action Item

1、Action item 默认格式是如果菜单项含字和图标的话，只显示图标  
2、如果想要显示字的效果，在xml里如下设置：android:showAsAction="ifRoom|withText" 或者在代码里调用 setShowAsAction()：setShowAsAction(MenuItem.SHOW\_AS\_ACTION\_IF\_ROOM)

六、ActionBar的单击事件

1、ActionBar中一个菜单项（即Action item）的触发像以前版本中的活动菜单回调方法（onOptionsItemSelected()）一样。  
2、当用户选择一个Fragment的菜单项时，首先会调用Activity的onOptionsItemSelected()方法，如果该方法返回false，则调用Fragment实现的onOptionsItemSelected()方

法。

public boolean onOptionsItemSelected(MenuItem item) {

// TODO Auto-generated method stub

switch (item.getItemId()) {

case R.id.menu\_save:

Toast.makeText(getApplicationContext(), "menu\_save", 0).show();

break;

case android.R.id.home:

Intent intent = new Intent(this, MainActivity.class);

intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP

| Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intent);

Toast.makeText(getApplicationContext(), "android.R.id.home", 0)

.show();

break;

default:

break;

}

return super.onOptionsItemSelected(item);

}

七、应用的图标

1、默认情况下，应用图标在ActionBar的左侧  
2、当用户点击应用bar的时候，系统会调用你的Activity里面的onOptionsItemSelected() 方法里面的Android.R.id.Home ID(你的主ID)里定义的事件。我们可以重写这个方法，添加一个条件执行该行为：让它跳转到这个应用的主Activity而不是返回的主屏幕。  
3、如果选择返回应用的主activity，最好的方式是在创建的Intent中添加FLAG\_ACTIVITY\_CLEAR\_TOP这个标签。  
使用这个FLAG\_ACTIVITY\_CLEAR\_TOP标签，如果应用的主Activity已经在activity栈中存在，所有在其上的activity都会被销毁，然后主activity会到栈顶，而不用创建主Activity的的新实例。

@Override

public boolean onOptionsItemSelected(MenuItem item)

{

switch (item.getItemId())

{

case android.R.id.home:

Intent intent = new Intent(this, MainActivity.class);

intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP);

startActivity(intent);

return true;

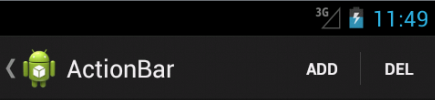
default:

return super.onOptionsItemSelected(item);

}

}

当我们想实现用户点击ActionBar 图标后返回前一个activity，可以通过在Actionbar设置setDisplayHomeAsUpEnabled(true) 来实现。如图：



八、添加活动视图

1、ActionBar中还可以添加很多其他的视图

我们也可以添加一个视图作为一个Action Item.我们可以通过在xml元素的android:actionLayout属性制定我们希望现实布局资源的ID（例如@layout/mysearchview)  
我们可以选择添加android:actionViewClass属性分配与完全限定的类名描述我们想要显示的视图(例如android.widget.SearchView)

package com.xys.actionbar;

import android.app.ActionBar;

import android.app.Activity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuInflater;

public class MainActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

@Override

protected void onStart() {

super.onStart();

ActionBar actionBar = this.getActionBar();

actionBar.setDisplayHomeAsUpEnabled(true);

}

@Override

public boolean onCreateOptionsMenu(Menu menu)

{

MenuInflater inflater = getMenuInflater();

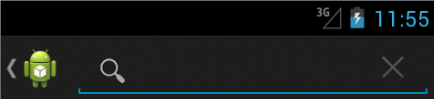
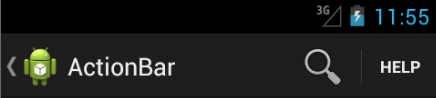
inflater.inflate(R.menu.main, menu);

return true;

}

}

这样我们就添加了一个Search的视图，效果如下：



2、其他Item

ActionBar的Item会分有好几个种类，分别有  
1）.普通类型（带图片带文字）。  
2）.分享类型。   
3）.可拓展类型。  
4）.列表类型。

如图：

`$Q5LHT7})3AFG~5(8XL}$S.jpg

<item

android:id="@+id/menu\_collapse"

android:actionLayout="@layout/collapsibleview"

android:icon="@drawable/search"

android:showAsAction="ifRoom|collapseActionView"

android:title="@string/menu\_search">

</item>

<item

android:id="@+id/menu\_share"

android:actionProviderClass="android.widget.ShareActionProvider"

android:enabled="true"

android:showAsAction="ifRoom"

android:title="@string/menu\_share"/>

<item

android:id="@+id/menu\_settings"

android:icon="@drawable/save"

android:orderInCategory="100"

android:title="@string/menu\_settings">

</item>

<item

android:id="@+id/menu\_save"

android:icon="@drawable/save"

android:showAsAction="ifRoom|withText"

android:title="@string/menu\_save">

<menu>

<item

android:id="@+id/action\_sort\_size"

android:icon="@android:drawable/ic\_menu\_sort\_by\_size"

android:onClick="onSort"/>

<item

android:id="@+id/action\_sort\_alpha"

android:icon="@android:drawable/ic\_menu\_sort\_alphabetically"

android:onClick="onSort"/>

</menu>

</item>

menu\_save Item 就是一个最普通的带文字带图片的Item。  
能否显示在ActionBar的Item主要看android:showAsAction="ifRoom|withText"这个属性。  
看menu\_sitting 就永远不会显示出来，只有在按menu时候会显示。谷歌文档中提供android：showAsAction总共有五个属性。  
never：永远不会显示。只会在溢出列表中显示。  
ifRoom：会显示在Item中，但是如果已经有4个或者4个以上的Item时会隐藏在溢出列表中。  
always：无论是否溢出，总会显示。  
withText：Title会显示。  
collapseActionView：可拓展的Item。

3、可分享的Item  
看上面代码片段 menu\_share 按钮就是一个可分享的Item。  
android:actionProviderClass="android.widget.ShareActionProvider"表示这个Item是个分享Item  
在java代码中进行实现:

public boolean onCreateOptionsMenu(Menu menu) {

// 添加分享按钮的事件

// 4.0以下没有ShareActionProvider

MenuItem menuItem\_share = menu.findItem(R.id.menu\_share);

// Get the provider and hold onto it to set/change the share intent.

mShareActionProvider = (ShareActionProvider) menuItem\_share

.getActionProvider();

// If you use more than one ShareActionProvider, each for a different

// action,

// use the following line to specify a unique history file for each one.

// mShareActionProvider.setShareHistoryFileName("custom\_share\_history.xml");

// Set the default share intent

mShareActionProvider.setShareIntent(getDefaultShareIntent());

}

private Intent getDefaultShareIntent() {

// TODO Auto-generated method stub

Log.i(TAG, "getDefaultShareIntent");

Intent intent = new Intent(Intent.ACTION\_SEND);

intent.setType("text/plain");

intent.putExtra(Intent.EXTRA\_SUBJECT, "小清新短信闪");

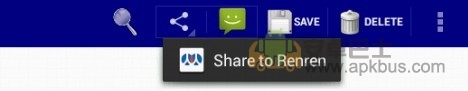
intent.putExtra(Intent.EXTRA\_TEXT, "小清新短信闪 1.0");

intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

return intent;

}

效果如图：



4、可拓展的Item

看到那个放大镜了，就是一个可拓展试的Item，看xml 中的 menu\_collapse。  
他有俩个属性是android:actionLayout="@layout/collapsibleview" android:showAsAction="ifRoom|collapseActionView"  
android:actionLayout="@layout/collapsibleview" 是点item后拓展成的View  
android:showAsAction="collapseActionView"代表他是一个可拓展的Item

Button btnn;

EditText editText;

private ShareActionProvider mShareActionProvider;

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.activity\_main, menu);

// 给可折叠的view上的控件添加事件

LinearLayout viewGroup = (LinearLayout) menu.findItem(

R.id.menu\_collapse).getActionView();

btnn = (Button) viewGroup.findViewById(R.id.colla\_btn);

editText = (EditText) viewGroup.findViewById(R.id.colla\_edit);

btnn.setOnClickListener(new OnClickListener() {

public void onClick(View v) {

// TODO Auto-generated method stub

Toast.makeText(getApplicationContext(),

editText.getText() + "~", 0).show();

}

});

}

可拓展的Item同时还支持一个拓展事件。只有在4.0以后的Api提供：

MenuItem menuItem = menu.findItem(R.id.menu\_collapse);

menuItem.setOnActionExpandListener(new OnActionExpandListener() {

@Override

public boolean onMenuItemActionExpand(MenuItem item) {

// TODO Auto-generated method stub

Toast.makeText(getApplicationContext(),

"onMenuItemActionExpand", 0).show();

return true;

}

@Override

public boolean onMenuItemActionCollapse(MenuItem item) {

// TODO Auto-generated method stub

Toast.makeText(getApplicationContext(),

"onMenuItemActionCollapse", 0).show();

return true;

}

});

5、自定义View

mCustomView = getLayoutInflater().inflate(

R.layout.action\_bar\_display\_options\_custom, null);

mCustomView.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Toast.makeText(getApplicationContext(), "自定义view", 0).show();

}

});

actionBar.setCustomView(mCustomView, new ActionBar.LayoutParams(

LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT));

九、ActionBar标签

1、ActionBar可以现实选项卡供用户切换浏览该Activity中不同的Fragment  
2、每个标签都可以是一个图标或文本标题

package com.xys.actionbar;

import android.app.ActionBar;

import android.app.ActionBar.Tab;

import android.app.Activity;

import android.app.Fragment;

import android.app.FragmentTransaction;

import android.os.Bundle;

public class MainActivity extends Activity

{

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ActionBar bar = getActionBar();

//设置为Tab模式

bar.setNavigationMode(ActionBar.NAVIGATION\_MODE\_TABS);

//新建2个Tab

ActionBar.Tab tabA = bar.newTab().setText("A Tab");

ActionBar.Tab tabB = bar.newTab().setText("B Tab");

//绑定到Fragment

Fragment fragmentA = new AFragmentTab();

Fragment fragmentB = new BFragmentTab();

tabA.setTabListener(new MyTabsListener(fragmentA));

tabB.setTabListener(new MyTabsListener(fragmentB));

bar.addTab(tabA);

bar.addTab(tabB);

}

protected class MyTabsListener implements ActionBar.TabListener

{

private Fragment fragment;

public MyTabsListener(Fragment fragment)

{

this.fragment = fragment;

}

@Override

public void onTabSelected(Tab tab, FragmentTransaction ft)

{

ft.add(R.id.fragment\_place, fragment, null);

}

@Override

public void onTabReselected(Tab arg0, FragmentTransaction arg1) {

// TODO Auto-generated method stub

}

@Override

public void onTabUnselected(Tab arg0, FragmentTransaction arg1) {

// TODO Auto-generated method stub

}

}

}

AB Tab页面：

package com.xys.actionbar;

import android.app.Fragment;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

public class AFragmentTab extends Fragment

{

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState)

{

return inflater.inflate(R.layout.fragment\_a, container, false);

}

}

布局：

activity\_main.xml：相当于容器

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

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical" >

<LinearLayout

android:id="@+id/fragment\_place"

android:layout\_width="match\_parent"

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

</LinearLayout>

</LinearLayout>

fragment\_a.xml：每个fragment的布局：

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

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical" >

<LinearLayout

android:id="@+id/fragment\_a"

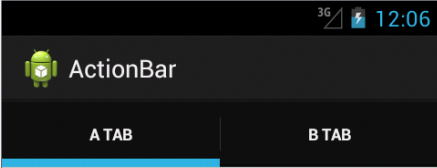
android:layout\_width="match\_parent"

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

</LinearLayout>

</LinearLayout>

显示效果如图：



九-1、下拉模式

前面讲到了

bar.setNavigationMode(ActionBar.NAVIGATION\_MODE\_TABS)

这次我们使用下拉模式

public class ActionBarTestActivity extends Activity {

/\*\* Called when the activity is first created. \*/

Button button1,button2;

ActionBar actionBar;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

actionBar=getActionBar();

// 生成一个SpinnerAdapter

SpinnerAdapter adapter = ArrayAdapter.createFromResource(this, R.array.student, android.R.layout.simple\_spinner\_dropdown\_item);

// 将ActionBar的操作模型设置为NAVIGATION\_MODE\_LIST

actionBar.setNavigationMode(ActionBar.NAVIGATION\_MODE\_LIST);

// 为ActionBar设置下拉菜单和监听器

actionBar.setListNavigationCallbacks(adapter, new DropDownListenser());

class DropDownListenser implements OnNavigationListener

{

// 得到和SpinnerAdapter里一致的字符数组

String[] listNames = getResources().getStringArray(R.array.student);

/\* 当选择下拉菜单项的时候，将Activity中的内容置换为对应的Fragment \*/

public boolean onNavigationItemSelected(int itemPosition, long itemId)

{

// 生成自定的Fragment

Ftest student = new Ftest();

FragmentManager manager = getFragmentManager();

FragmentTransaction transaction = manager.beginTransaction();

// 将Activity中的内容替换成对应选择的Fragment

transaction.replace(android.R.id.content, student, listNames[itemPosition]);

transaction.commit();

return true;

}

xml布局：

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

<resources>

<string name="hello">Hello World, ActionBarTestActivity!</string>

<string name="app\_name">ActionBarTest</string>

<string name="ActionBar\_name">ActionBar下拉菜单</string>

<string-array name="student">

<item>item1</item>

<item>item2</item>

<item>item3</item>

<item>item4</item>

</string-array>

</resources>

fragment：

public class Ftest extends Fragment

{

// Fragment对应的标签，当Fragment依附于Activity时得到

private String tag;

@Override

public void onAttach(Activity activity)

{

super.onAttach(activity);

tag = getTag();

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)

{

// View view = (View) inflater.inflate(R.layout.ftest, container, false);

// TextView textView=(TextView) view.findViewById(R.id.topView);

// textView.setText(tag);

TextView textView = new TextView(getActivity());

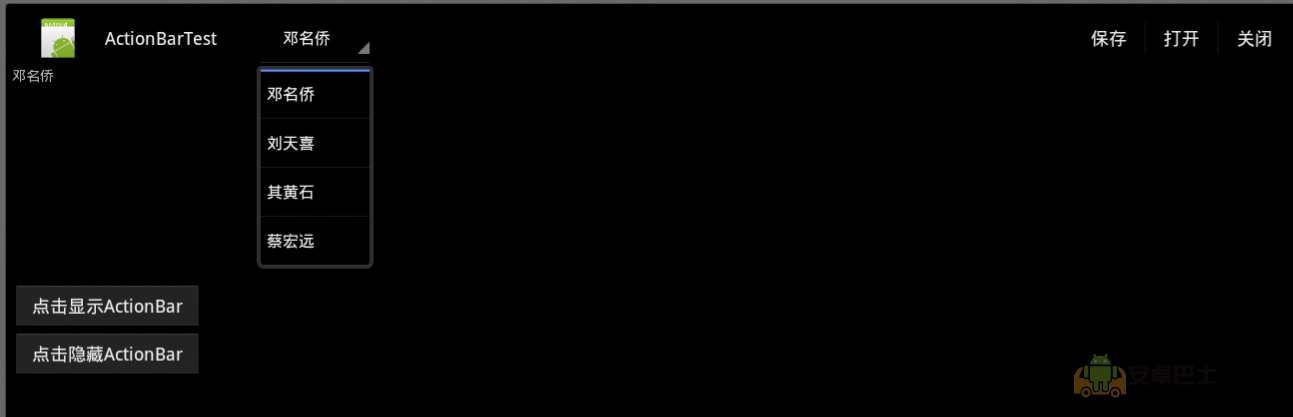
textView.setText(tag);

return textView;

}

}

效果图如下：



十、ActionBar的美化

1、

<style name="MyActionBar" parent="@android:style/Widget.Holo.ActionBar">

<item name="android:windowActionBarOverlay">true</item>

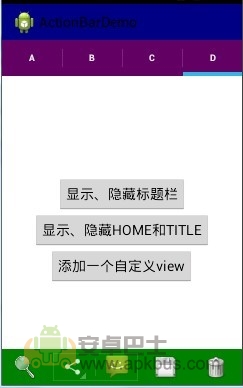
<item name="android:background">@color/ab\_background\_color</item>

<item name="android:backgroundStacked">@color/ab\_backgroundStacked\_color</item>

<item name="android:backgroundSplit">@color/ab\_backgroundSplit\_color</item>

</style>

item name="android:background:ActionBar本身的背景颜色。  
item name="android:backgroundStacked"：ActionBar被分离时Tab 的背景颜色。  
item name="android:backgroundSplit"：ActionBar Item 在分割到底部时的背景颜色。  
item name="android:background：蓝色 item name="android:backgroundStacked"：紫色 item name="android:backgroundSplit"：绿色



2、其他选项的Style

Action items

android:actionButtonStyle

<style name="Widget.Holo.ActionButton" parent="Widget.ActionButton">

<item name="android:minWidth">56dip</item>

<item name="android:gravity">center</item>

<item name="android:paddingLeft">12dip</item>

<item name="android:paddingRight">12dip</item>

<item name="android:scaleType">center</item>

</style>

从上面的定义看出，它定义了Button的最小宽度、居中样式、左右padding以及图片的放缩方式。再看看该样式的父样式

<style name="Widget.ActionButton">

<item name="android:background">?android:attr/actionBarItemBackground</item>

<item name="android:paddingLeft">12dip</item>

<item name="android:paddingRight">12dip</item>

<item name="android:minWidth">56dip</item>

<item name="android:minHeight">?android:attr/actionBarSize</item>

</style>

7.jpg

android:actionBarItemBackground

定义显示的item选项的背景 Api 14以上可以使用android:itemBackground定义全部Item的背景选项  
这俩个选项其实和定义actionBarButtonStyle的背景已经有覆盖 所以用的时候定义一个就行

android:actionBarDivider

样式属性actionBarDivider定义了ActionBar操作项之间的分隔条。可以看出，它直接引用了另一个ActionBar相关属性 dividerVertical，其定义如下：

<item name="dividerVertical">@drawable/divider\_vertical\_dark</item>

8.jpg

android:actionMenuTextColor  
android:actionMenuTextAppearance  
这俩个属性是定义menu item 文字大小和颜色

<style name="myactionMenuTextAppearance" parent="@android:style/TextAppearance.Holo.Large">

<item name="android:textSize">20sp</item>

<item name="android:textStyle">bold</item>

<item name="android:textAllCaps">true</item>

</style>

<item name="android:actionMenuTextAppearance">@style/myactionMenuTextAppearance</item>

<item name="android:actionMenuTextColor">@color/ab\_backgroundStacked\_color</item>

9.jpg

android:actionBarWidgetTheme   
Navigation tabs  
android:actionBarTabStyle  
样式属性 actionBarTabStyle 为Tab页的标签定义样式

<style name="Widget.Holo.ActionBar.TabView" parent="Widget.ActionBar.TabView">

<item name="android:background">@drawable/tab\_indicator\_holo</item>

<item name="android:paddingLeft">16dip</item>

<item name="android:paddingRight">16dip</item>

</style>

android:actionBarTabBarStyle  
样式属性actionBarTabBarStyle 定义了Tab标签之间分隔条。

<style name="Widget.Holo.ActionBar.TabBar" parent="Widget.ActionBar.TabBar">

<item name="android:divider">?android:attr/actionBarDivider</item>

<item name="android:showDividers">middle</item>

<item name="android:dividerPadding">12dip</item>

</style>

android:actionBarTabTextStyle  
样式属性 actionBarTabTextStyle 定义了Tab的文本样式，如下：

<style name="Widget.Holo.ActionBar.TabText" parent="Widget.ActionBar.TabText">

<item name="android:textAppearance">@style/TextAppearance.Holo.Medium</item>

<item name="android:textColor">?android:attr/textColorPrimary</item>

<item name="android:textSize">12sp</item>

<item name="android:textStyle">bold</item>

<item name="android:textAllCaps">true</item>

<item name="android:ellipsize">marquee</item>

<item name="android:maxLines">2</item>

</style>

android:actionDropDownStyle  
样式属性 actionDropDownStyle 定义了下拉导航列表的样式，如下：

<style name="Widget.Holo.Spinner.DropDown.ActionBar">

<item name="android:background">@android:drawable/spinner\_ab\_holo\_dark</item>

</style>

其父样式

<style name="Widget.Holo.Spinner" parent="Widget.Spinner.DropDown">

<item name="android:background">@android:drawable/spinner\_background\_holo\_dark</item>

<item name="android:dropDownSelector">@android:drawable/list\_selector\_holo\_dark</item>

<item name="android:popupBackground">@android:drawable/menu\_dropdown\_panel\_holo\_dark</item>

<item name="android:dropDownVerticalOffset">0dip</item>

<item name="android:dropDownHorizontalOffset">0dip</item>

<item name="android:dropDownWidth">wrap\_content</item>

<item name="android:popupPromptView">@android:layout/simple\_dropdown\_hint</item>

<item name="android:gravity">left|center\_vertical</item>

</style>

说下ActionBar 总体系统提供的样式  
样式属性actionBarStyle定义了ActionBar的基本样式。

<style name="Widget.Holo.ActionBar" parent="Widget.ActionBar">

<item name="android:titleTextStyle">@android:style/TextAppearance.Holo.Widget.ActionBar.Title</item>

<item name="android:subtitleTextStyle">@android:style/TextAppearance.Holo.Widget.ActionBar.Subtitle</item>

<item name="android:background">@android:drawable/ab\_transparent\_dark\_holo</item>

<item name="android:backgroundStacked">@android:drawable/ab\_stacked\_transparent\_dark\_holo</item>

<item name="android:backgroundSplit">@android:drawable/ab\_bottom\_transparent\_dark\_holo</item>

<item name="android:divider">?android:attr/dividerVertical</item>

<item name="android:progressBarStyle">@android:style/Widget.Holo.ProgressBar.Horizontal</item>

<item name="android:indeterminateProgressStyle">@android:style/Widget.Holo.ProgressBar</item>

<item name="android:progressBarPadding">32dip</item>

<item name="android:itemPadding">8dip</item>

</style>

其父样式

<style name="Widget.ActionBar">

<item name="android:background">@android:drawable/action\_bar\_background</item>

<item name="android:displayOptions">useLogo|showHome|showTitle</item>

<item name="android:divider">@android:drawable/action\_bar\_divider</item>

<item name="android:height">?android:attr/actionBarSize</item>

<item name="android:paddingLeft">0dip</item>

<item name="android:paddingTop">0dip</item>

<item name="android:paddingRight">0dip</item>

<item name="android:paddingBottom">0dip</item>

<item name="android:titleTextStyle">@android:style/TextAppearance.Widget.ActionBar.Title</item>

<item name="android:subtitleTextStyle">@android:style/TextAppearance.Widget.ActionBar.Subtitle</item>

<item name="android:progressBarStyle">@android:style/Widget.ProgressBar.Horizontal</item>

<item name="android:indeterminateProgressStyle">@android:style/Widget.ProgressBar.Small</item>

<item name="android:homeLayout">@android:layout/action\_bar\_home</item>

</style>