**AndroidUI设计技巧**

# 1、页面设计

## 1.1、TabHost以及TabActivity的使用

使用RadioGroup和RadioButton替换默认的选项卡

### 1.1.1 TabHost第一中使用方法

1、定义一个Activity继承于TabActivity

2、定义布局文件的根标签为<TabHost> (id的定义可以参考TabHost.java)

<TabHost> 标签的id为@android:id/tabhost

在<TabHost>标签内有两个的子标签

1)、<TabWidget>,id必须为@android:id/tabs，TabWidget就是显示的标签切换按钮，如果不显示切换按钮，而是利用RadioGroup和RadioButton来代替的话，可以设置TabWidget的android:visibility=”gone”

2)、<FrameLayout>，id必须为@android:id/tabcontent

3、添加标签页，每一个标签页可以利用tabhost.TabSpec对象，创建TabSpec对象的方法：tabhost.newTabSpec,指定tabspec名字属性setIndicator(String label)指定标签页的名字；指定标签页内容：setContent(Intent intent)

使用TabHost各个Activity的生命周期：

1、先初始化默认的第一个tabSpec中的Content Activity

2、初始化主TabHost（例如：MainUI）

3、点击那个标签就对于初始化那个标签页中的Activity

4、当各个标签页中的Acitivty都已经初始化后，再点击切换标签只会掉用onResume()方法，这说明离开某个标签的话，这个标签页中的Activity是处于暂停状态的

|  |
| --- |
| 标签页布局文件：  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <TabHost xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:id=*"@android:id/tabhost"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:orientation=*"horizontal"*  android:background=*"@drawable/splash\_background"*>    <TabWidget android:id=*"@android:id/tabs"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:visibility=*"gone"*/>    <FrameLayout android:id=*"@android:id/tabcontent"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*/>    <RadioGroup android:id=*"@+id/rgBar"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:background=*"@drawable/main\_bar\_bg"*  android:orientation=*"horizontal"*  android:layout\_gravity=*"bottom"*>    <RadioButton android:id=*"@+id/rbtnHome"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:drawableTop=*"@drawable/main\_home\_icon"*  android:text=*"@string/main\_home"*  style=*"@style/main\_rbtn\_style"* />    <RadioButton android:id=*"@+id/rbtnMsg"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:drawableTop=*"@drawable/main\_msg\_icon"*  android:text=*"@string/main\_msg"*  style=*"@style/main\_rbtn\_style"* />      <RadioButton android:id=*"@+id/rbtnUserinfo"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:drawableTop=*"@drawable/main\_userinfo\_icon"*  android:text=*"@string/main\_userinfo"*  style=*"@style/main\_rbtn\_style"* />    <RadioButton android:id=*"@+id/rbtnMore"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:drawableTop=*"@drawable/main\_more\_icon"*  android:text=*"@string/main\_more"*  style=*"@style/main\_rbtn\_style"* />    </RadioGroup>  </TabHost>  用到的样式文件：  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <resources>  <style name=*"main\_rbtn\_style"*>  <item name=*"android:button"*>@null</item>  <item name=*"android:background"*>@drawable/main\_rbtn\_d</item>  <item name=*"android:gravity"*>center</item>  <item name=*"android:layout\_weight"*>1</item>  <item name=*"android:drawablePadding"*>2dp</item>  <item name=*"android:paddingTop"*>4dp</item>  <item name=*"android:textSize"*>8dp</item>  </style>  </resources>  对于的Activity：  /\*\*  \* 主窗口UI  \* **@author** YJ  \*/  **public** **class** MainUI **extends** TabActivity {  **private** **final** **static** String *TAB\_HOME* = "homeTab" ;  **private** **final** **static** String *TAB\_MSG* = "msgTab" ;  **private** **final** **static** String *TAB\_USERINFO* = "userinfoTab" ;  **private** **final** **static** String *TAB\_MORE* = "moreTab" ;    **private** TabHost tabHost ;  **private** RadioGroup rgBar ;    @Override  **protected** **void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  setContentView(R.layout.*main*) ;    init() ; //初始化    }  /\*\*  \* 初始化操作  \*/  **private** **void** init() {  TabHost.TabSpec tabSpec ;  tabHost = **this**.getTabHost() ;    //添加首页tab  tabSpec = tabHost.newTabSpec(*TAB\_HOME*).setIndicator(*TAB\_HOME*)  .setContent(**new** Intent(**this**, HomeUI.**class**)) ;  tabHost.addTab(tabSpec) ;    //添加信息tab  tabSpec = tabHost.newTabSpec(*TAB\_MSG*).setIndicator(*TAB\_MSG*)  .setContent(**new** Intent(**this**, MsgUI.**class**)) ;  tabHost.addTab(tabSpec) ;    //添加用户信息tab  tabSpec = tabHost.newTabSpec(*TAB\_USERINFO*).setIndicator(*TAB\_USERINFO*)  .setContent(**new** Intent(**this**, UserInfoUI.**class**)) ;  tabHost.addTab(tabSpec) ;    //添加更多信息tab  tabSpec = tabHost.newTabSpec(*TAB\_MORE*).setIndicator(*TAB\_MORE*)  .setContent(**new** Intent(**this**, MoreUI.**class**)) ;  tabHost.addTab(tabSpec) ;    **this**.rgBar = (RadioGroup) **this**.findViewById(R.id.*rgBar*) ;  **this**.rgBar.setOnCheckedChangeListener(**new** RadioGroup.OnCheckedChangeListener() {  @Override  **public** **void** onCheckedChanged(RadioGroup group, **int** checkedId) {  **switch** (checkedId) {  **case** R.id.*rbtnHome*: //首页  tabHost.setCurrentTabByTag(*TAB\_HOME*) ;  **break**;  **case** R.id.*rbtnMsg*: //信息  tabHost.setCurrentTabByTag(*TAB\_MSG*) ;  **break**;  **case** R.id.*rbtnUserinfo*: //用户信息  tabHost.setCurrentTabByTag(*TAB\_USERINFO*) ;  **break**;  **case** R.id.*rbtnMore*: //更多  tabHost.setCurrentTabByTag(*TAB\_MORE*) ;  **break**;  **default**:  **break**;  }  }  }) ;    //默认选中首页  //tabHost.setCurrentTabByTag(TAB\_HOME) ;  rgBar.check(R.id.*rbtnHome*) ;  }  } |

### 1.1.2 TabHost第二种使用方法（不继承于TabActivity）

<TabHost>标签的id自定义

在<TabHost>标签内有两个的子标签（这两个子标签的id，在预览显示效果的时候eclipse会提示该设置什么id）

1)、<TabWidget>,id必须为@android:id/tabs，TabWidget就是显示的标签切换按钮，如果不显示切换按钮，而是利用RadioGroup和RadioButton来代替的话，可以设置TabWidget的android:visibility=”gone”

2)、<FrameLayout>，id必须为@android:id/tabcontent

定义一个Activity使用该布局文件，利用findViewById方法查找到TabHost

## 1.2、顶部导航栏滑动效果

例子程序

E:\AppsDevelop\Resources\Android\UI设计\顶部导航滑动效果

## 1.3、ViewPager实现页面滑动效果

E:\AppsDevelop\Resources\Android\UI设计\ViewPager实现页面滑动效果

## 1.4、布局文件包含

在定义布局文件的时候，可能有好几个页面都用到了相同的topBar或者footerBar，这时候就可以定义两个模板布局文件，在需要的布局中进行包含

布局文件的包含 <include layout=”@layout/xxx” />，类似于jsp中的页面包含，用layout属性指定要包含页面的layout id

### 1.4.1 ViewStub、requestFocus、marge、include

<ViewStup ...../>

ViewStub此标签可以使用UI在特殊的情况下，直观效果类似于设置View的不可见性，但是其更大的意义在于被这个标签所包裹的View在默认的情况下不会占用任何内存空间。ViewStub通过include从外部导入View元素。用法通过android:layout来指定所包含的内容，默认情况下ViewStub所包含的标签都属于visibility=GONE，ViewStub通过方法inflate()，来召唤系统加载其内部的View

<include ...../>

可以通过这个标签直接加载外部的xml到当前结构中，是复用UI资源的常用标签，用法是将需要复用的xml文件路径赋予include标签的layout属性

<requestFocus .... />

用于指定屏幕内的焦点View，用法是将标签至于View标签的内部

<EditText ...>

<requestFocus/>

</EditText>

## 1.5 布局

### 1.5.1 两个控件：一个居右对齐，另一个居左对齐

使用RelativeLayout并设置并设置相对于父容器的对齐方法

左对齐： android:layout\_alignParentLeft=*"true"*

右对齐：android:layout\_alignParentRight=*"true"*



### 1.5.2 实现控件集合居中（居底部都可以）



1、控件集合要用一个布局包含（例如LinearLayout）

2、设置父容器的对齐方式为居中就可以了（居底部就设置：android:gravity=*"bottom"*）

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"fill\_parent"*  android:gravity=*"center"*  android:orientation=*"vertical"*  android:padding=*"20dp"* >  <LinearLayout android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:orientation=*"vertical"* >    <ImageView android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  />    <EditText android:id=*"@+id/etUsername"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:hint=*"@string/login\_username"*  android:drawableRight=*"@drawable/ic\_launcher"*/>  <EditText android:id=*"@+id/etPassword"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:hint=*"@string/login\_password"*  android:drawableRight=*"@drawable/ic\_launcher"*/>    <RelativeLayout android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:orientation=*"horizontal"* >    <CheckBox android:id=*"@+id/cbRemeberPwd"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*  android:layout\_alignParentLeft=*"true"*  android:layout\_gravity=*"center\_vertical"*  android:text=*"@string/login\_remeber\_password"* />    <Button android:id=*"@+id/btnLogin"*  android:layout\_width=*"wrap\_content"*  android:layout\_alignParentRight=*"true"*  android:layout\_height=*"wrap\_content"*  android:layout\_gravity=*"center\_vertical"*  android:text=*"@string/login\_button\_login"* />  </RelativeLayout>  </LinearLayout>  </LinearLayout> |

## 1.6 设置Activity背景透明

### 方法一

只要在配置文件内activity属性配置内加上

android:theme="@android:style/Theme.Translucent"

就好了。这样就调用了android的透明样式！

### 方法二

**1、在res/values下建colors.xml文件，写入：**

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

　　<resources>

　　 <color name="transparent">#9000</color>

　　</resources>

　　这个值设定了整个界面的透明度，为了看得见效果，现在设为透明度为56%(9/16)左右。

**2、在res/values/下建styles.xml,设置程序的风格**

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

　　<resources>

　　 <style name="Transparent">

　　 <item name="android:windowBackground">@color/transparent</item>

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

　　 <item name="android:windowAnimationStyle">@+android:style/Animation.Translucent</item>

　　 </style>

　　</resources>

当然在定义主题的时候还可以继承Android系统自定义的主题，指定**parent**属性即可

<style name=*"Theme.CustomDialog"* parent=*"android:style/Theme.Dialog"*>

<item name=*"android:windowBackground"*>@color/transparent</item>

<item name=*"android:windowIsTranslucent"*>true</item>

<item name=*"android:windowAnimationStyle"*>@+android:style/Animation.Translucent</item>

</style>

**3、把这个styles.xml用在相应的Activity上。即在AndroidManifest.xml中的任意<activity>标签中添加**

　　android:theme="@style/transparent"

如果想设置所有的activity都使用这个风格，可以把这句标签语句添加在<application>中。

最后运行程序，哈哈，是不是发现整个界面都被蒙上一层半透明了。最后可以把背景色#9000换成#0000，运行程序后，就全透明了，看得见背景下的所有东西可以却都操作无效。

## 1.7 横竖屏切换

### 1、禁止横竖屏切换

不想横竖屏之间切换，最简单的办法就是在项目的 AndroidManifest.xml中找到你所指定的activity中加上android:screenOrientation属性，他有以下几个参数：

"unspecified"

默认值 由系统来判断显示方向.判定的策略是和设备相关的，所以不同的设备会有不同的显示方向.

"landscape"

横屏显示（宽比高要长）

"portrait"

竖屏显示(高比宽要长)

"user"

用户当前首选的方向

"behind"

和该Activity下面的那个Activity的方向一致(在Activity堆栈中的)

"sensor"

有物理的感应器来决定。如果用户旋转设备这屏幕会横竖屏切换。

"nosensor"

忽略物理感应器，这样就不会随着用户旋转设备而更改了 （ "unspecified"设置除外 ）。

也可以在Java代码中通过setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_LANDSCAPE);来设置

### 2、如果要让软件在横竖屏之间切换，由于横竖屏的高宽会发生转换，有可能会要求不同的布局。可以通过以下方法来切换布局：

1）、在res目录下建立layout-land和layout-port目录,相应的layout文件不变，比如main.xml。layout-land是横屏的layout,layout-port是竖屏的layout，其他的不用管，模拟器会自动寻找。

2）、通过this.getResources().getConfiguration().orientation来判断当前是横屏还是竖屏然后来加载相应的xml布局文件。因为当屏幕变为横屏的时候,系统会重新呼叫当前Activity的OnCreate方法,你可以把以下方法放在你的 OnCreate中来检查当前的方向,然后可以让你的SetContentView来载入不同的Layout xml.

if (this.getResources().getConfiguration().orientation == Configuration.ORIENTATION\_LANDSCAPE) {

Log.i("info", "landscape");

}

else if (this.getResources().getConfiguration().orientation == Configuration.ORIENTATION\_PORTRAIT) {

Log.i("info", "portrait");

}

# 2、使用自定义样式替换系统默认的ProgressBar显示图片

1、定义一个样式style，继承于android:style/Widget.ProgressBar.Large；覆盖属性android:indeterminateDrawable，将该属性指定一个rotate动画，该图片动画定义在drawable资源文件夹下

|  |
| --- |
| D:\视频教程\Android\3G手机Android应用新浪微博\sina\第19讲、ListView美化\myweibo19\myweibo\res\drawable-mdpi\progress.png  <?xml version=*"1.0"* encoding=*"utf-8"*?>  <resources>  <style name=*"customProgressBarStyle"*  parent=*"android:style/Widget.ProgressBar.Large"*>  <item  name=*"android:indeterminateDrawable"*>@drawable/progressbar\_rotate</item>  </style>    </resources> |

2、定义rotate动画

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <animated-rotate xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:drawable=*"@drawable/progress"*  android:pivotX=*"50%"*  android:pivotY=*"50%"* /> |

# 3、ListView美化

用心的朋友应该会发现，listview中在设置了背景之后。会有些问题。

1.、listview在拖动的时候背景图片消失变成黑色背景。等到拖动完毕我们自己的背景图片才显示出来。

问题1 有如下代码结解决 android:scrollingCache="false"

2 、listview的上边和下边有黑色的阴影。

问题2 用如下代码解决：android:fadingEdge="none"

3、lsitview的每一项之间需要设置一个图片做为间隔。

问题3用如下代码解决： android:divider="@drawable/list\_driver",其中 @drawable/list\_driver 是一个图片资源

**总体如下**

<ListView android:id="@+id/myListView01"

android:layout\_width="fill\_parent"

android:layout\_height="287dip"

android:fadingEdge="none"

android:divider="@drawable/list\_driver"

android:scrollingCache="false"

android:background="@drawable/list">

</ListView>

4. 自定义listview的时候，当你不使用android:cacheColorHint=“#00000000”会出现下面选中一个空间黑色底色的情况，破坏整体美观度：

5. 当你不使用android:listSelector属性，默认会显示选中的item为橙黄底色，有时候我们需要去掉这种效果。listview.setSelector()

6．scrollbars属性，作用是隐藏listView的滚动条，

android:scrollbars="none"与setVerticalScrollBarEnabled(true);的效果是一样的，不活动的时候隐藏，活动的时候也隐藏

7.fadeScrollbars属性，android:fadeScrollbars="true" 配置ListView布局的时候，设置这个属性为true就可以实现滚动条的自动隐藏和显示。

8.stackFromBottom属性，设置该属性之后你做好的列表就会显示你列表的最下面，值为true和false

android:stackFromBottom="true"

9.transciptMode属性，需要用ListView或者其它显示大量Items的控件实时跟踪或者查看信息，并且希望最新的条目可以自动滚动到可视范围内。通过设置的控件transcriptMode属性可以将Android平台的控件（支持ScrollBar）自动滑动到最底部。

android:transcriptMode="alwaysScroll"

## 2.1 自定义ListView的分割线

ListView设定分割线的样式和宽度用到两个属性

android:divider 设定分割线来自图片或者是颜色，没有分割线可设置：android:divider=”@drawable/@null”

android:dividerHeight 设定分割线的高度

1、选定一个要作为分割线的图片

2、在/drawable资源文件夹下定义一个divider\_line的资源文件，文件内容如下：

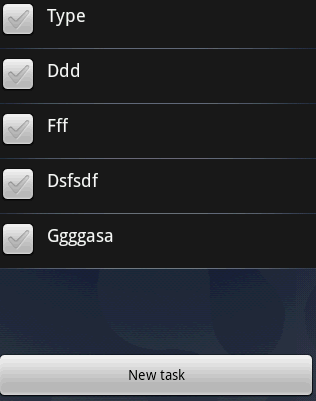
android:tileMode=”repeat” : 表示图片平铺

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <bitmap xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:src=*"@drawable/dot\_bg"*  android:tileMode=*"repeat"* /> |

3、在ListView中指定分割线属性 android:divider=”@drawable/divider\_line”

## 2.2 自定义ListView的背景，而且拖动的时候不会出现黑色背景

改变背景最简单的方法就是找一张图片，然后指定ListView的android:background=”@drawable/xxx”就可以了，但是这样做背景是改变了，但是当你拖动或者点击ListView空白的位置的时候发现ListItem都变成了黑色，破坏了整体效果，如下图



这是因为ListView的效果默认ListView的item的背景是透明的，而ListView的北京是固定不变的，所以在滚动条滚动的过程中会实时的去将当前的item的显示内容跟背景进行混合运算，所以android系统为了优化这一过程，就使用了一个叫做android:cacheColorHint属性，在黑色主题下默认颜色为#191919，所以就出现了刚才的画面，有一半是黑色的

解决方案：

1、如果你只是想要换掉背景颜色的话，那么直接指定android:cacheColorHint为你所要的颜色就可以了

2、如果要用图片做背景的话那么就将android:cacheColorHint指定为透明（#00000000）就可以了，当然美化是要牺牲一些效率的。

**美化后：**



## 2.3 ListView中Button点击无响应解决方案

Android ListView添加按钮后自身无响应。

在ListView上加了个Button 后就只响应Button 而不响应ListView的单击了。 后来终于解决了。

是因为Button获取了焦点， 导致ListView没有了焦点， 所以响应不了。

我们只要在Button上加上两句Xml的属性即可。

android:focusable="false"

android:focusableInTouchMode="false"

listView 异步更新完后，不能点击条目的问题

希望对今后碰到此问题的哥们提个醒，listview需要重新获得焦点

request.requestFocusFromTouch();

# 4 控件共同特性美化

## 3.1、控件中文字和图片的排列

文本框和单选按钮

在文本框中也可以直接设置图片和文字，具体用到属性为：

android:drawableTop

android:drawableBottom

android:drawableLeft

android:drawableRight

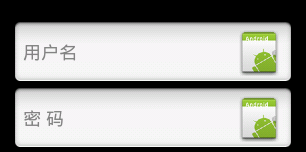
android:drawablePadding //图片和文字之间的间隙

用到的属性：

android:drawableLeft=*"@drawable/comment\_icon" //图片在文字的什么位置，这里是文字的左边*

android:drawablePadding=*"2dp" //图片和文字之间的距离*

### EditText控件实现图文混排方式



<EditText android:id=*"@+id/etPassword"*

android:layout\_width=*"fill\_parent"*

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

android:hint=*"@string/login\_password"*

android:drawableRight=*"@drawable/ic\_launcher"*/>

## 3.2、单选按钮去掉默认样式

单选按钮去掉默认样式： android:button=”@null”

复选框？

# 5 ProgressBar & SeekBar 美化

1、准备图片（背景、进度背景（、拖动条））

2、定义ProgressBar样式

第一个条目：ProgressBar背景

第二个条目：走过的进度条背景

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <layer-list xmlns:android=*"http://schemas.android.com/apk/res/android"*>  <item android:id=*"@+android:id/background"*  android:drawable=*"@drawable/fast\_seek\_progress"* />    <item android:id=*"@+android:id/progress"*  android:drawable=*"@drawable/media\_player\_progress\_bg"* />  </layer-list> |

2、主要修改如下属性

进度条样式：android:progressDrawable=*"@drawable/seekbar\_img"*

*@drawable/seekbar\_img*样式是在drawable中定义的

拖动按钮样式（只在SeekBar）：android:thumb=*"@drawable/media\_player\_progress\_button"*

下面修改为ProgressBar也可以

|  |
| --- |
| <SeekBar  android:id=*"@+id/seekbar"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:paddingLeft=*"5dip"*  android:paddingRight=*"5dip"*  android:layout\_marginLeft=*"10dip"*  android:layout\_marginRight=*"10dip"*  android:layout\_gravity=*"center"*  android:layout\_marginTop=*"90dip"*  android:progressDrawable=*"@drawable/seekbar\_img"*  android:thumb=*"@drawable/media\_player\_progress\_button"* /> |