

宝贵建议请发送至：[wangzhenyang@itcast.cn](mailto:wangzhenyang@itcast.cn)



黑马程序员

itheima.com

-编程，始于黑马

# Android 课程同步笔记

Beta 0.02 版

By 阳哥

---

1. 手机适配方式 (★★)	2
1.1 适配方式之 dp	2
1.2 适配方式之 dimens	4
1.3 适配方式之 layout	5
1.4 适配方式之 java 代码适配	5
1.5 适配方式之 weight 权重适配	7
2. 类库的创建和引用 (★★★)	9
3. 完成侧拉菜单 (★★)	12
3.1 SlideMenu 简介	12
3.2 代码实现 SlideMenu-left 工程	14
3.3 代码实现 SlideMenu-left-right 工程	20
4. 完成引导界面 (★★★)	22
4.1 引导界面简介	22
4.2 代码实现引导界面	22

# Android 智慧北京-01 手机适配&lib 工程生成&侧拉栏

## 1. 手机适配方式 (★★)

### 1.1 适配方式之 dp

名词解释：

◆ 分辨率：eg：480\*800,1280\*720。表示物理屏幕区域内像素点的总和。(切记：跟屏幕适配没有任何关系)

因为我们既可以把 1280\*720 的分辨率做到 4.0 的手机上面。我也可以把 1280\*720 的分辨率做到 5.0 英寸的手机上面，如果分辨率相同，手机屏幕越小清晰。

◆ px(pix)：像素，就是屏幕中最小的一个显示单元

◆ dpi (像素密度)：即每英寸屏幕所拥有的像素数，像素密度越大，显示画面细节就越丰富。

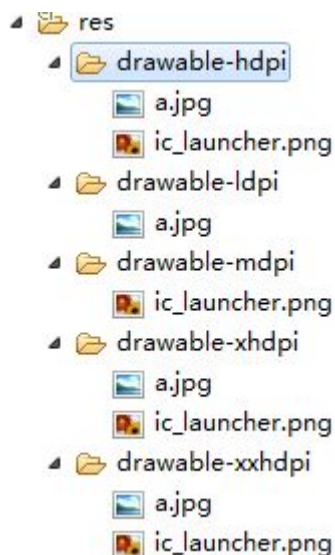
计算公式：像素密度 =  $\sqrt{(\text{长度像素数}^2 + \text{宽度像素数}^2)}$  / 屏幕尺寸

注：屏幕尺寸单位为英寸 例：分辨率为 1280\*720 屏幕宽度为 6 英寸 计算所得像素密度约等于 245，屏幕尺寸指屏幕对角线的长度。

在 Android 手机中 dpi 分类：

ldpi	Resources for low-density ( <i>ldpi</i> ) screens (~120dpi).
mdpi	Resources for medium-density ( <i>mdpi</i> ) screens (~160dpi). (This is the baseline density.)
hdpi	Resources for high-density ( <i>hdpi</i> ) screens (~240dpi).
xhdpi	Resources for extra high-density ( <i>xhdpi</i> ) screens (~320dpi).

在我们的 Android 工程目录中有如下 drawable-\*dpi 目录，这些目录是用来适配不同分辨率手机的。



Android 应用在查找图片资源时会根据其分辨率自动从不同的文件目录下查找（这本身就是 Android 系统的适配策略），如果在低分辨的文件目录中比如 `drawable-mdpi` 中没有图片资源，其他目录中都有，当我们将该应用部署到 `mdpi` 分辨率的手机上时，那么该应用会查找分辨率较高目录下的资源文件，如果较高分辨率目录下也没有资源则只好找较低目录中的资源了。

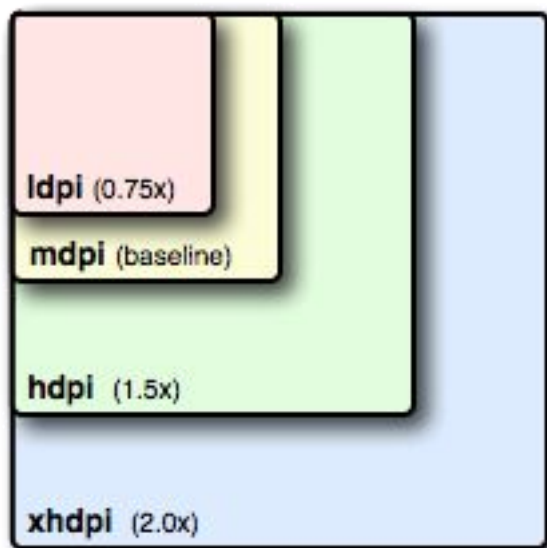
常见手机屏幕像素及对应分辨率级别：

- ◆ ldpi 320\*240
- ◆ mdpi 480\*320
- ◆ hdpi 800\*480
- ◆ xhdpi 1280\*720
- ◆ xxhdpi 1920\*1080

dp 和 px 之间的简单换算关系：

- ◆ ldpi 的手机 1dp=0.75px
- ◆ mdpi 的手机 1dp=1.0px
- ◆ hdpi 的手机 1dp=1.5px
- ◆ xhdpi 的手机 1dp=2.0px

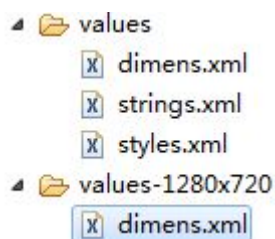
◆ xxhdpi 的手机 1dp=3.0px



**Tips**：根据上面的描述我们得出如下结论，对于 mdpi 的手机，我们的布局通过 dp 单位可以达到适配效果。

## 1.2 适配方式之 `dimens`

跟 drawable 目录类似的，在 Android 工程的 res 目录下有 values 目录，这个是默认的目录，同时为了适配不同尺寸手机我们可以创建一个 values-1280x720 的文件夹，同时将 `dimens.xml` 文件拷贝到该目录下。



在 `dimens.xml` 中定义一个尺寸，如下图所示。

```
dimens.xml
1 <resources>
2   <dimen name="width">160dp</dimen>
3 </resources>
```

在 values-1280x720 目录中的 `dimens.xml` 中定义同样的尺寸名称，但是使用不同的尺寸，如下图所示。

```
<resources>
  <dimen name="width">180dp</dimen>
</resources>
```

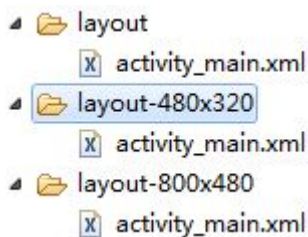
当我们在布局文件中使用长或者宽度单位时，比如下图所示，应该使用@dimen/width 来灵活的定义宽度。

```
<TextView
    android:background="#000000"
    android:layout_width="@dimen/width"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
<LinearLayout
```

**Tips**：在 values-1280x720 中，中间的是大写字母 X 的小写形式 x，而不是加减乘除的乘号。如果我们在 values-1280x720 中放置了 dimens 常量，一定记得也将该常量的对应值在 values 目录下的 dimens.xml 中放一份，因为该文件是默认配置，当用户的手机不是 1280\*720 的情况下系统应用使用的是默认 values 目录中的 dimens.xml。

## 1.3 适配方式之 layout

跟 values 一样，在 Android 工程目录中 layout 目录也支持类似 values 目录一样的适配，在 layout 中我们可以针对不同手机的分辨率制定不同的布局，如下图所示。



## 1.4 适配方式之 java 代码适配

为了演示用 java 代码控制适配的效果，因此假设有这样的需求，让一个 TextView 控件的宽和高分别为屏幕的宽和高的一半。

我们新建一个 Android 工程，修改 main\_activity.xml，布局文件清单如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
```

```
<!-- 当前控件宽高为屏幕宽度的各 50% -->
<TextView
    android:id="@+id/tv"
    android:background="#000000"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
</RelativeLayout>
```

在 MainActivity.java 类中完成用 java 代码控制 TextView 的布局效果，其代码清单如下：

```
public class MainActivity extends Activity {

    private static final String tag = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //去掉 title
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_main);
        //获取 TextView 控件
        TextView tv = (TextView) findViewById(R.id.tv);
        //找到当前控件的父控件(父控件上给当前的子控件去设定一个规则)
        DisplayMetrics metrics = new DisplayMetrics();
        //给当前 metrics 去设置当前屏幕信息(宽(像素)高(像素))
        getWindowManager().getDefaultDisplay().getMetrics(metrics);
        //获取屏幕的高度和宽度
        Constant.srceenHeight = metrics.heightPixels;
        Constant.srceenWidth = metrics.widthPixels;
        //日志输出屏幕的高度和宽度
        Log.i(tag, "Constant.srceenHeight = "+Constant.srceenHeight);
        Log.i(tag, "Constant.srceenWidth = "+Constant.srceenWidth);
        //宽高各 50%
        RelativeLayout.LayoutParams layoutParams = new RelativeLayout.LayoutParams(
            //数学角度上 四舍五入
            (int)(Constant.srceenWidth*0.5+0.5),
            (int)(Constant.srceenHeight*0.5+0.5));
        //给 tv 控件设置布局参数
        tv.setLayoutParams(layoutParams);
    }
}
```

其中 Constant 类是一个常量类，很简单，只有两个常量用来记录屏幕的宽和高，其代码清单如下：

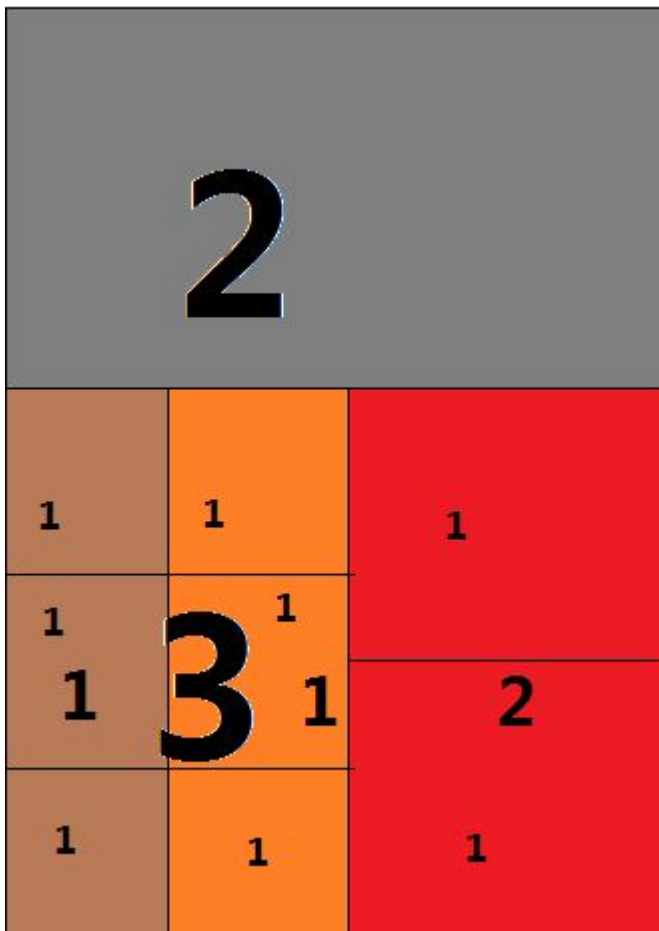
```
public class Constant {  
    public static int srceenHeight;  
    public static int srceenWidth;  
}
```

## 1.5 适配方式之 weight 权重适配

在控件中使用属性 `android:layout_weight="1"` 可以起到适配效果，但是该属性的使用有如下规则：

- 1、只能用在线性控件中，比如 LinearLayout。
- 2、竖直方向上使用权重的控件高度必须为 0dp（Google 官方的推荐用法）
- 3、水平方向上使用权重的控件宽度必须为 0dp（Google 官方的推荐用法）

为了演示权重的使用，我们完成如下布局。





布局清单如下所示：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <EditText
        android:id="@+id/a1"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />

    <LinearLayout
        android:id="@+id/a2"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="3"
        android:orientation="horizontal" >

        <LinearLayout
            android:id="@+id/b1"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="2"
            android:background="@android:color/black"
            android:orientation="vertical" >

                <LinearLayout
                    android:orientation="vertical"
                    android:id="@+id/c1"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="3"
                    android:background="#b40e0e" >

                <LinearLayout
                    android:orientation="vertical"
                    android:id="@+id/c2"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="3"
```

```
        android:background="#1c241c" >
    </LinearLayout>
</LinearLayout>

<LinearLayout
    android:orientation="vertical"
    android:id="@+id/b2"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="#11f70c" >
</LinearLayout>

<LinearLayout
    android:orientation="horizontal"
    android:id="@+id/b3"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:background="@android:color/white" >
    </LinearLayout>
</LinearLayout>

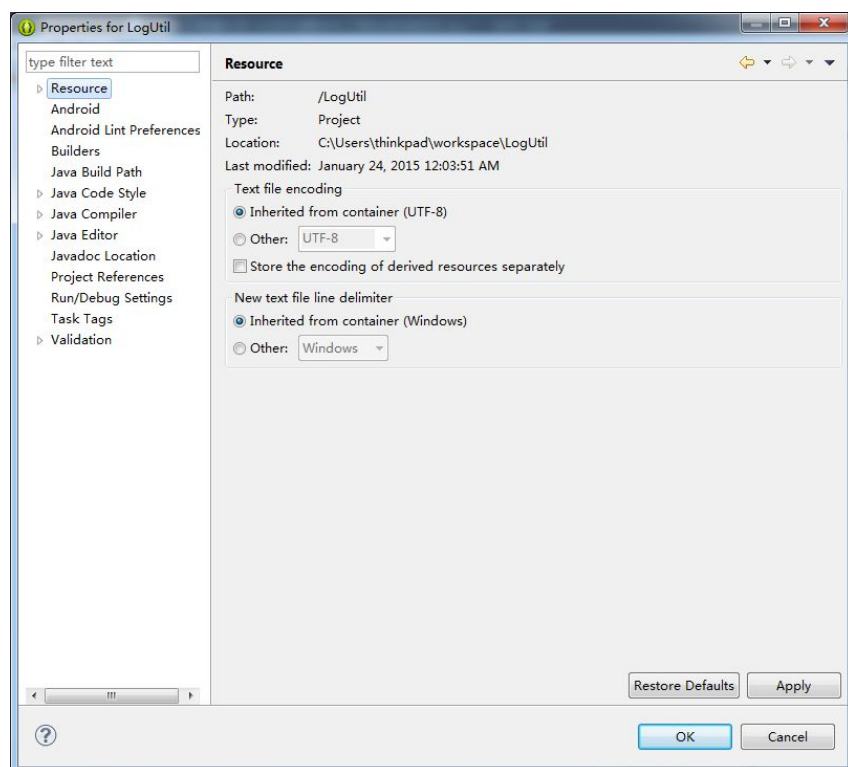
</LinearLayout>
```

## 2. 类库的创建和引用 ( ★★★ )

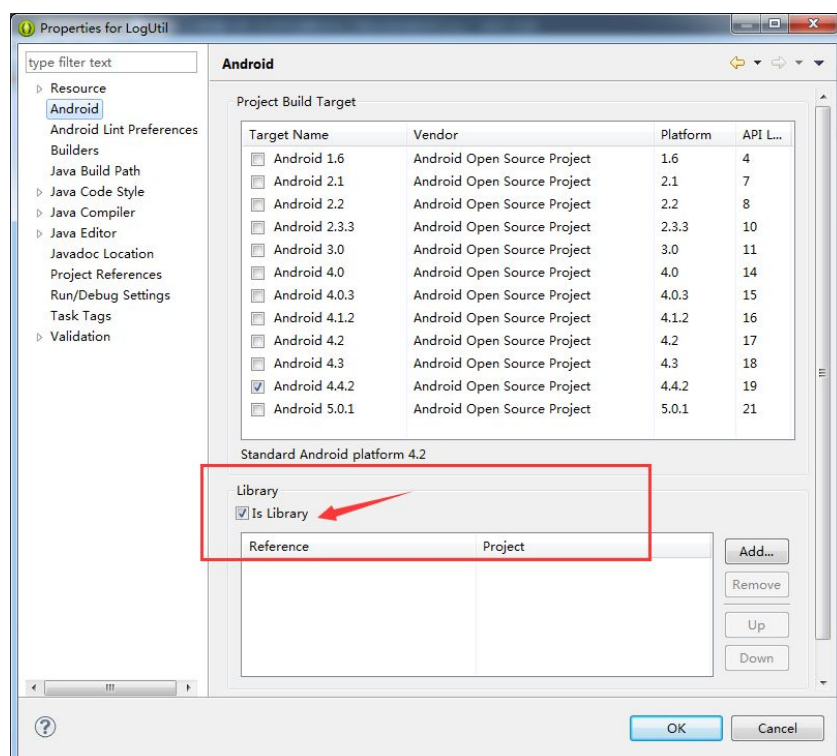
我们创建一个 Android 工程，LogUtil，并把该工程作为库工程，库工程只能被其他应用使用，而不能单独作为 Android 应用使用。

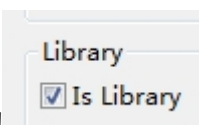
### ◆ 创建库文件

右击 LogUtil->properties，弹出如下界面：



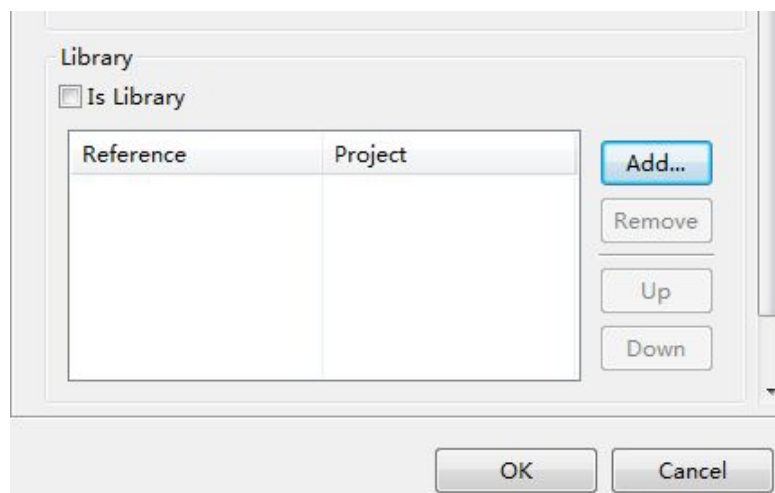
选中 Android，进入如下界面：



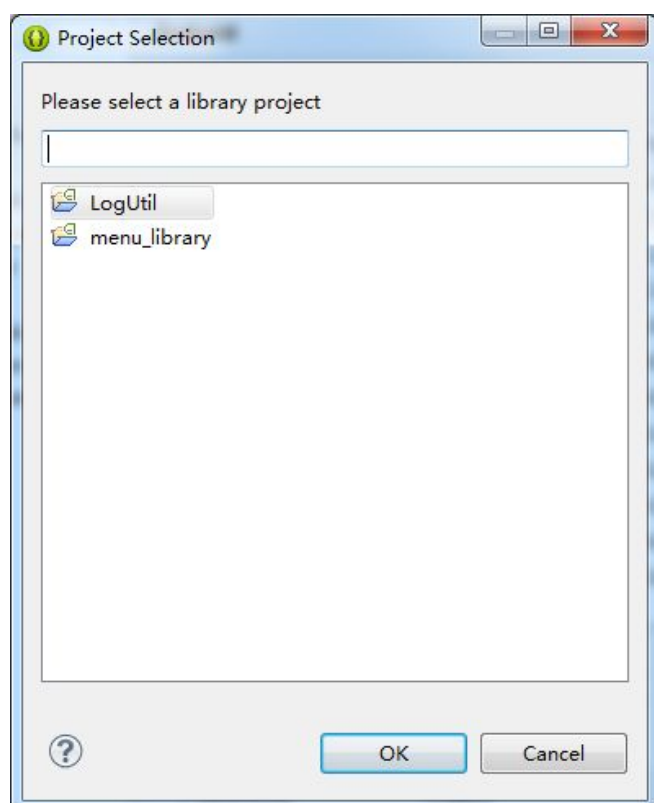
将上图中的  打上勾则该工程就为了库工程。

### ◆ 引用库文件：

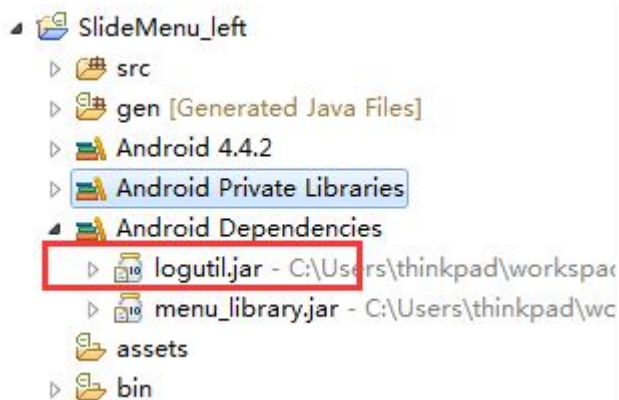
引用库文件也很简单，在一个右击一个 Android 工程，选择 properties，然后在弹出的对话框中选中 Android，然后选择右下角的 Add 按钮，进行添加。



点击 Add 弹出如下对话框，该对话框列出了当前工作空间中所有可用的库工程，我们选择 LogUtil，然后点击 OK。



然后观察我们的工程，发现引用库文件后多了如下目录，发现其将我们引入的库工程打包为了 logutil.jar 文件。



引入库文件后，我们就可以在代码中使用库文件中的类。

**Tips**：引入库文件的好处是如果库文件提供的功能不能满足我们的使用，那么我们可以直接将库文件源码进行修改和加强。

## 3. 完成侧拉菜单 (★★)

### 3.1 SlideMenu 简介

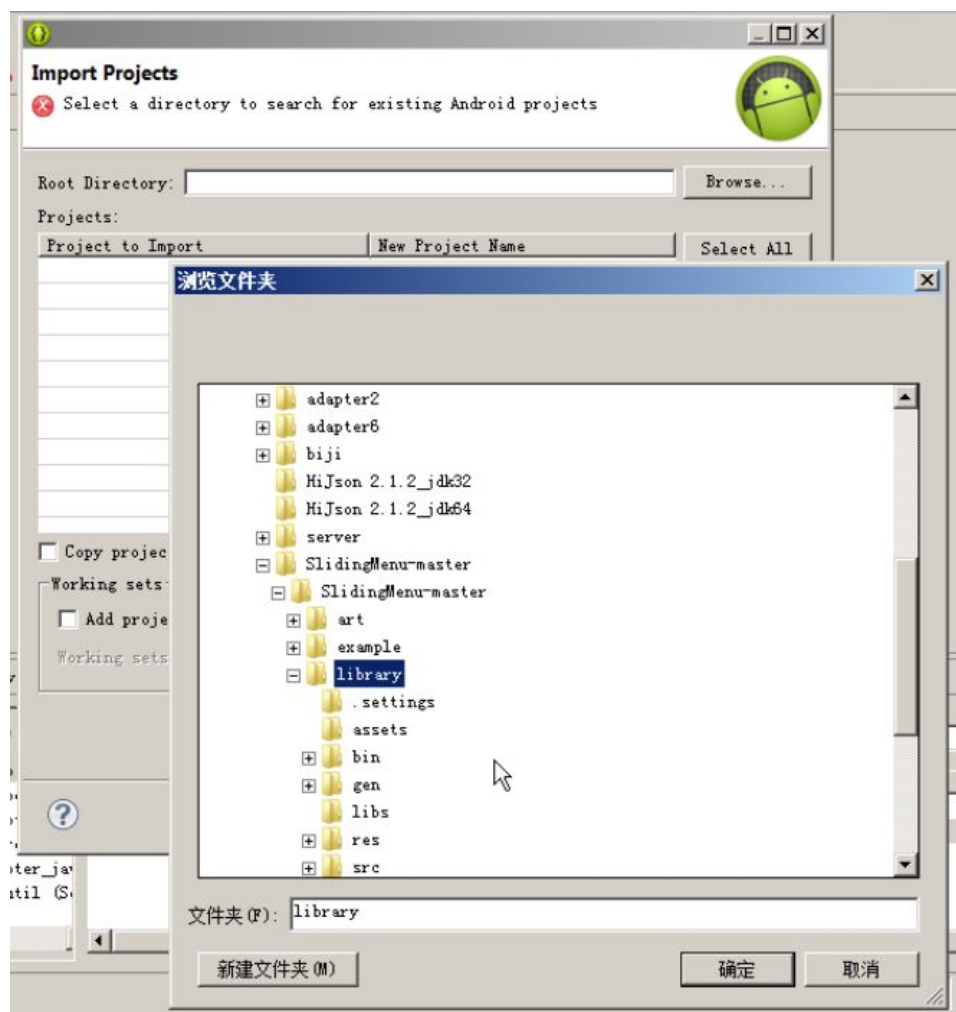
在开发 Android 项目中我们经常使用第三方的类库文件，同样的在我们智慧北京项目中，我们的侧拉菜单也将使用第三方的类库文件。

slideMenu 在本人的 Android 自定义控件中有详细的说明，并且自己用代码实现了一个 slideMenu。这里我们直接使用第三方的 slideMenu，该类库要比自己写的更加强大。

SlidingMenu 百度网盘地址：<http://pan.baidu.com/s/1hq3oLvu>

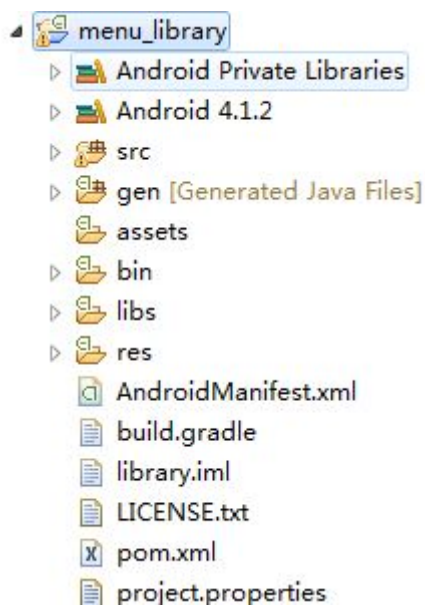
◆ 将 SlidingMenu 导入到工作空间中

操作步骤很简单，跟导入普通工程一样，导入过程截图如下：



**Tips**：如上图，导入的不是整个 SlidingMenu-master 工程，而是里面的 library 文件目录。

导入成功以后的目录结果如下图所示：



## 3.2 代码实现 SlideMenu-left 工程

我们将左侧拉菜单作为单独的一个工程来展示。

**1** 创建一个新的 Android 工程，工程名 SlidingMenu-left。然后引入之前已经在工作空间中引入的库文件 menu\_left。

**Tips**：引入的时候可能会报如下异常。

```
[2014-07-11 11:59:42 - news_wh1] Found 2 versions of android-support-v4.jar in the dependency list,
[2014-07-11 11:59:42 - news_wh1] but not all the versions are identical (check is based on SHA-1 only at this time).
[2014-07-11 11:59:42 - news_wh1] All versions of the libraries must be the same at this time.
[2014-07-11 11:59:42 - news_wh1] Versions found are:
[2014-07-11 11:59:42 - news_wh1] Path: D:\mwqi\workspace\news_wh1\libs\android-support-v4.jar
[2014-07-11 11:59:42 - news_wh1] Length: 627582
[2014-07-11 11:59:42 - news_wh1] SHA-1: db0f122c99ef9f90dbab3fada6d191f2880cbb8e
[2014-07-11 11:59:42 - news_wh1] Path: D:\mwqi\workspace\menulibrary\libs\android-support-v4.jar
[2014-07-11 11:59:42 - news_wh1] Length: 385685
[2014-07-11 11:59:42 - news_wh1] SHA-1: 48c94ae70fa65718b382098237806a5909bb096e
[2014-07-11 11:59:42 - news_wh1] Jar mismatch! Fix your dependencies
```

引起该异常的原因是我们自己创建的 Android 中引入了 android-support-v4.jar 同时 menu\_left 库文件中也有 android-support-v4.jar，这样就导致了 jar 包冲突。

开发工具之所以能监测到两个相同名称的 jar 包不一致是因为工具采用了 SHA-1 算法来分别获取两个 jar 包的值，然后进行比较（这一点根据异常也能看出如此）。

### ◆ SHA-1 算法简介

SHA-1 是一种数据加密算法，该算法的思想是接收一段明文，然后以一种不可逆的方式将它转换成一段（通常更小）密文，也可以简单的理解为取一串输入码（称为预映射或信息），并把它们转化为长度较短、位数固定的输出序列即散列值（也称为信息摘要或信息认证代码）的过程。

**2** 创建用于显示菜单的 Fragment 类 MenuFragment。

我们创建好的侧拉菜单效果如下图所示，左侧菜单整体是一个 Fragment，叫 MenuFragment，MenuFragment 中有 ListView 布局，布局中的每一个条目都是一个对应的 Fragment。





MenuFragment 代码清单：

```
public class MenuFragment extends Fragment {

    private View view;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.i("MenuFragment", "onCreate");
    }

    //在 activity 中等同于 setContentView
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        Log.i("MenuFragment", "onCreateView");
        view = inflater.inflate(R.layout.list_view, null); //把一个 xml 布局加载到内存中
        return view;
    }

    //数据填充的操作
    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
    }
}
```



```
Log.i("MenuFragment", "onActivityCreated");

ListView listView = (ListView) view.findViewById(R.id.list_view);
listView.setAdapter(new ArrayAdapter<String>(
    getActivity(),
    android.R.layout.simple_list_item_1,
    android.R.id.text1,
    initData()));

listView.setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3)
{
    Fragment fragment = null;
    switch (arg2) {
        case 0:
            fragment = new Fragment1();
            break;
        case 1:

            fragment = new Fragment2();
            break;
        case 2:
            fragment = new Fragment3();

            break;
        case 3:
            fragment = new Fragment4();

            break;
        case 4:
            fragment = new Fragment5();
            break;
    }

    switchFragment(fragment);
    }
});

//回调 MainActivity 对应对象的方法
private void switchFragment(Fragment fragment) {
```

```
        if(getActivity() instanceof MainActivity){
            //如果获取的上下文环境就是 MainActivity 再去操作
            ((MainActivity)getActivity()).switchFragment(fragment);//this
//            MainActivity.context.switchFragment(fragment);
        }
    }

    private List<String> initData() {
        List<String> arrayList = new ArrayList<String>();
        arrayList.add("fragment1");
        arrayList.add("fragment2");
        arrayList.add("fragment3");
        arrayList.add("fragment4");
        arrayList.add("fragment5");
        return arrayList;
    }
}
```

上面代码中使用到的 list\_view.xml 文件清单如下：

```
<?xml version="1.0" encoding="utf-8"?>
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@id/list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
</ListView>
```

上面的代码中用到了 Fragment1、Fragment2....Fragment5,这里因为没有复杂业务逻辑并且这 5 个菜单 Fragment 都用友类似的功能，因此我们可以抽取一个公用的 Fragment，BaseFragment。

BaseFragment.java 代码清单如下：

```
public class BaseFragment extends Fragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
```

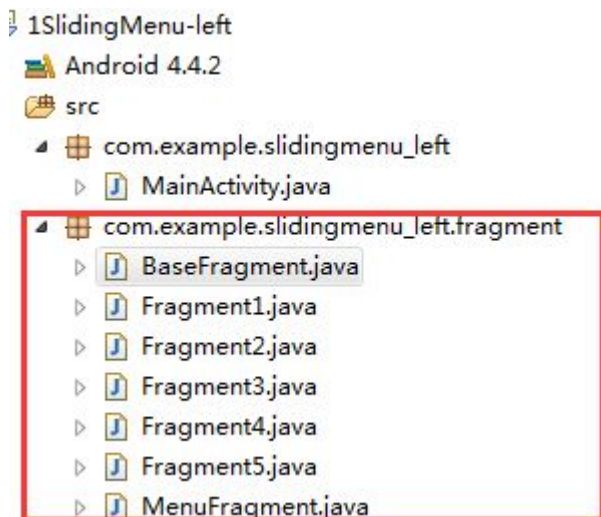
```

        Bundle savedInstanceState) {
            TextView textView = new TextView(getActivity());
            //去获取当前对象的名称
            textView.setText(this.getClass().getSimpleName());
            return textView;
        }

        @Override
        public void onActivityCreated(Bundle savedInstanceState) {
            super.onActivityCreated(savedInstanceState);
        }
    }
}

```

同时创建 Fragment1....Fragment5，并继承 BaseFragment，此时项目的工程目录如下所示：



3

实现 MainActivity，在该类中实现核心业务逻辑。

因为需要在 MainActivity 中实现侧拉菜单，因此 MainActivity 需要继承 SlidingFragmentActivity 类。

MainActivity.java 代码清单如下：

```

public class MainActivity extends SlidingFragmentActivity {

    private SlidingMenu slidingMenu;
    public static MainActivity context;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```

```
context = this;
requestWindowFeature(Window.FEATURE_NO_TITLE);
// 内容页布局
setContentView(R.layout.content);
// 设置侧拉条目布局
setBehindContentView(R.layout.menu_frame);
// 获取侧拉栏目对象
slidingMenu = getSlidingMenu();

/*
 * SlidingMenu.TOUCHMODE_FULLSCREEN 全屏触摸有效 SlidingMenu.TOUCHMODE_MARGIN
 * 拖拽边缘有效 SlidingMenu.TOUCHMODE_NONE 不响应触摸事件
 */
slidingMenu.setTouchModeAbove(SlidingMenu.TOUCHMODE_FULLSCREEN);

// 设置内容显示页对应的 dp 大小
slidingMenu.setBehindOffsetRes(R.dimen.slidingmenu_offset);
// //设置左侧侧拉栏目宽度
// slidingMenu.setBehindWidth(140);
// 设置侧拉栏目所在位置
/*
 * SlidingMenu.LEFT SlidingMenu.LEFT_RIGHT SlidingMenu.RIGHT
 */
slidingMenu.setMode(SlidingMenu.LEFT_RIGHT);
// 给侧拉栏目和左侧内容页区分开(加线)
slidingMenu.setShadowDrawable(R.drawable.shadow);
// 设置线的宽度
slidingMenu.setShadowWidthRes(R.dimen.shadow_width);

// fragment 去替换布局中节点
MenuFragment menuFragment = new MenuFragment();
// FragmentManager 管理者
getSupportFragmentManager()
// 开启事物
    .beginTransaction()
    // 通过 fragment 去替换对应布局
    .replace(R.id.menu, menuFragment, "MENU")
    // 提交事物
    .commit();
}

// 当前类中做替换当前显示内容界面的操作
public void switchFragment(Fragment fragment) {
```

```

        if (fragment != null) {

            getSupportFragmentManager().beginTransaction().replace(R.id.content_frame,
            fragment, "HOME").commit();
            // 缩回(点击弹出左侧侧拉栏目)
            slidingMenu.toggle();
        }
    }
}

```

**Tips**：上面代码使用到了多个资源文件，我把其中的布局文件展示出来。

◆ content.xml

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/content_frame"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
</FrameLayout>

```

◆ menu\_frame.xml

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/menu"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
</FrameLayout>

```

4

部署并运行。将上面工程部署到模拟器上，发现我们想要的效果已经出来。

### 3.3 代码实现 SlideMenu-left-right 工程

在 3.2 节我们实现了左侧菜单，SlideMenu 还支持左右同时都有菜单。实现很简单只需做以下修改即可。

1

在 3.2 节中 MainActivity.java 的基础上将 SlidingMenu 的模式改为 LEFT\_RIGHT。

```

slidingMenu.setMode(SlidingMenu.LEFT_RIGHT);

```

2

将 3.2 节中 MainActivity.java 的基础上添加用于显示右侧菜单的 Fragment 类，并设置显示阴影效果

```
//添加第二个菜单也就是右侧菜单
slidingMenu.setSecondaryMenu(R.layout.menu_frame_right);
//设置右侧分割线的图片
slidingMenu.setSecondaryShadowDrawable(R.drawable.shadow);
//创建一个用于显示右侧菜单的 Fragment
RightMenuFragment rightMenuFragment = new RightMenuFragment();
//FragmentManager 管理者
getSupportFragmentManager()
//开启事物
.beginTransaction()
//通过 fragment 去替换对应布局
.replace(R.id.menu_right, rightMenuFragment, "MENU_RIGHT")
//提交事物
.commit();
```

**Tips**：上面的 RightMenuFragment 类跟 MenuFragment 类内容是一模一样的，直接拷贝就行。

3

运行上面工程，拉出右侧菜单效果如下：



## 4. 完成引导界面 (★★★)

### 4.1 引导界面简介

在软件第一次安装并打开的时候，我们通常会看见一些漂亮的引导界面。引导界面不仅可以展示软件丰富多彩的功能也能供用户设置一些必要的信息。我们要做的引导界面效果如下图所示。



该界面主要是通过 ViewPager 来实现的。

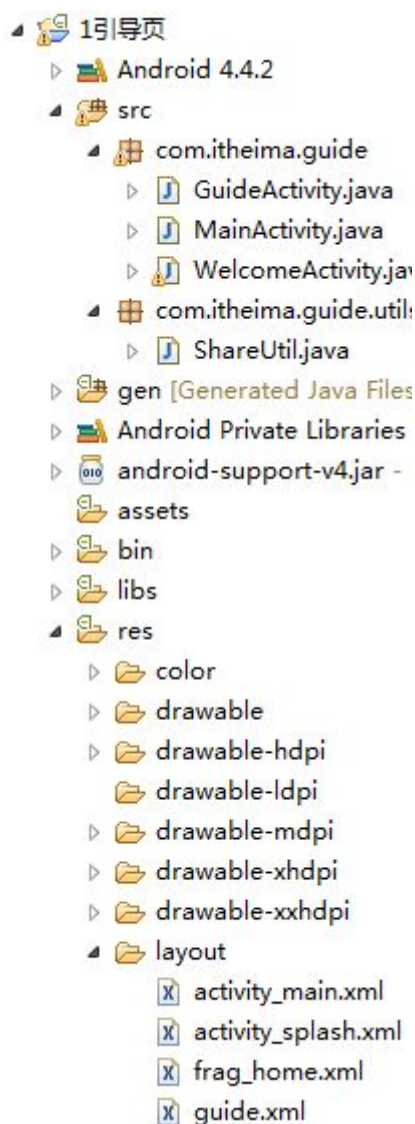
### 4.2 代码实现引导界面

1

创建一个新的 Android 工程，《引导页》包名 com.itheima.guide。

工程目录结构如下图所示：





2

编写引导页的 Activity，WelcomeActivity.java。该类中的代码业务逻辑是当用户进来时从 sp 文件中判断是否是第一次进来，如果是则通过显式意图打开 GuideActivity，如果不是则进入主界面。

WelcomeActivity.java 代码清单如下所示：

```
public class WelcomeActivity extends Activity {

    private static final String tag = "MainActivity";
    private SharedPreferences sharedPreferences;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
    }
}
```



```
        boolean b = ShareUtil.getBooleanData(getApplicationContext(), "is_first",
true);
    if(b){
        //第一次进入，进入导航页，is_first 改成 false
        Log.i(tag, "这是第一次进入页面");
        Intent intent = new Intent(WelcomeActivity.this, GuideActivity.class);

        startActivity(intent);

        ShareUtil.saveBooleanData(getApplicationContext(), "is_first", false);
    }else{
        //第二次进入，进入详情页
        Log.i(tag, "这是第二次进入页面");
        Toast.makeText(this, "第二次进入", 0).show();
        Intent intent = new Intent(WelcomeActivity.this, MainActivity.class);

        startActivity(intent);
    }

    finish();
}
```

**Tips**：这里我们需要将 WelcomeActivity 设置为默认启动界面。在 AndroidManifest.xml 中进行如下改动：

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen" >
    <activity
        android:name="com.itheima.guide.WelcomeActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <!-- 入口 activity -->
            <action android:name="android.intent.action.MAIN" />
            <!-- 在桌面是否显示图标 -->
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activityv
```

```

        android:name="com.itheima.guide.GuideActivity">
    </activity>
    <activity
        android:name="com.itheima.guide.MainActivity">
    </activity>

</application>

```

**Tips**：在 WelcomeActivity 中用到了 ShareUtil.java 类，该类很简单的一个工具类，其清单如下：

```

public class ShareUtil {
    public static String CONFIG = "config";
    private static SharedPreferences sharedPreferences;

    public static void saveBooleanData(Context context,String key,boolean value){
        if(sharedPreferences==null){
            sharedPreferences = context.getSharedPreferences(CONFIG,
Context.MODE_PRIVATE);
        }
        sharedPreferences.edit().putBoolean(key, value).commit();
    }

    public static boolean getBooleanData(Context context,String key,boolean
defValue){
        if(sharedPreferences==null){
            sharedPreferences = context.getSharedPreferences(CONFIG,
Context.MODE_PRIVATE);
        }
        return sharedPreferences.getBoolean(key, defValue);
    }
}

```

**3** 编码实现 GuideActivity.java。该类主要实现了 ViewPager 控件，该类也是我们当前项目的核心类。

```

public class GuideActivity extends Activity {
    public static final String tag = "GuideActivity";
    private Button button;
    private ViewPager pager;
    private List<ImageView> imageList;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.guide);
//初始化控件
button = (Button) findViewById(R.id.button);
pager = (ViewPager) findViewById(R.id.view_pager);
//创建 3 个向导页，对应的 3 张图片
ImageView imageView1 = new ImageView(getApplicationContext());
imageView1.setBackgroundResource(R.drawable.guide_1);

ImageView imageView2 = new ImageView(getApplicationContext());
imageView2.setBackgroundResource(R.drawable.guide_2);

ImageView imageView3 = new ImageView(getApplicationContext());
imageView3.setBackgroundResource(R.drawable.guide_3);
//将 3 个 ImageView 控件放到集合中
imageList = new ArrayList<ImageView>();
imageList.add(imageView1);
imageList.add(imageView2);
imageList.add(imageView3);
//给 ViewPager 对象设置适配器
pager.setAdapter(new MyAdatper());
//给 ViewPager 对象设置监听事件
pager.setOnPageChangeListener(new OnPageChangeListener() {

    @Override
    public void onPageSelected(int arg0) {
        //如果当前页正好是最好一页的时候，显示 Button（让用户点击进去主界面）
        if(arg0 == imageList.size()-1){
            //选中某页
            button.setVisibility(View.VISIBLE);
            button.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {

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

                    finish();
                }
            });
        }
    }
});
}
```

```

        @Override
        public void onPageScrolled(int arg0, float arg1, int arg2) {
            //滚动完成

        }

        @Override
        public void onPageScrollStateChanged(int arg0) {
            //滚动状态发生改变

        }
    });
}

class MyAdatper extends PagerAdapter{
    //返回页面个数
    @Override
    public int getCount() {
        return imageUrl.size();
    }
    @Override
    public boolean isViewFromObject(View arg0, Object arg1) {
        return arg0==arg1;
    }
    //在当前的 ViewPager 中去添加一个 item
    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        ((ViewPager)container).addView(imageList.get(position));
        Log.i(tag, "instantiateItem position="+position);
        return imageList.get(position);
    }
    //销毁一个 view
    @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        Log.i(tag, "destroyItem position="+position);
        ((ViewPager)container).removeView((View)object);
    }
}
}

```

**Tips** : GuideActivity.java 中用到了 guide 布局文件，其清单文件如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <android.support.v4.view.ViewPager
        android:id="@+id/view_pager"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="30dp"
        android:paddingLeft="30dp"
        android:paddingRight="30dp"
        android:text="开始体验"
        android:textSize="20sp"
        android:visibility="gone" />

</RelativeLayout>
```

4 上面用到了多种图片资源文件，这些不是我们要演示的重点，图片资源的处理过程就不再展示。

将上面的工程部署到模拟器上，并运行，就会看到我们想要的效果了。

**至此，本文档完！**

2015 年 3 月 2 日 星期一 23:45:33

北京市海淀区东馨园小区