大话企业级 Android 开发·第九部分

本教程说明及版权声明

- 《大话企业级 Android 开发》是国士工作室为了方便中国 Android 开发者，推动 Android 企业级应用开发，特投入大量心血撰写的书籍，并在网络上免费发布，希望为移动互联网和智能手机时代贡献绵薄之力！所有相关文档版权均属国士工作室所有。
- 本教程是由国士工作室参考官方文档，综合市面相关书籍，经过充分的吸收消化，结合开发实践的一部原创作品，为了本教程及早与广大读者同仁见面、分享，特采用定稿一部分就发布一部分的连载方式发布。读者可以在本博客获取最新内容。
- 未经国士工作室授权，禁止将此文档及其衍生作品以标准（纸质）书籍形式发行。
- 本文档受有关法律的版权保护，对本文档内容的任何未经同意的复制和抄袭行为，将导致相应的法律责任。未经国士工作室同意，任何团体及个人不能用此教程牟利，违者必究。但是：在不收取其他人费用的前提下，您可以自由传播此文档，但必须保证版权信息、文档及其自带标示的完整性。
- 如果对该文档有任何疑问或者建议，请进入官方博客 http://www.cnblogs.com/guoshiandroid/留言或者直接与国士工作室联系（后附联系方式），我们会慎重参考您的建议并根据需要对本文档进行修改，以造福更多开发者！
- 《大话企业级 Android 开发》的最新及完整内容会在国士工作室官方博客定期更新，请访问国士工作室博客 http://www.cnblogs.com/guoshiandroid/获取更多更新内容。

关于国士工作室

我们(国士工作室)是一支专注于 Android 平台企业级应用开发的技术团队，对娱乐多媒体应用有着深刻的理解及研发能力，致力服务于企业用户。为音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络应用向移动互联网发展提供解决方案和技术支持，为企业提供 Android 培训服务等多种业务。

我们尤其擅长于提供从 Android 客户端到服务端的一站式解决方案和技术支持，服务端可以采用 Java EE，也可以采用轻量级流行的 LAMP 技术体系。目前，研发出了比 KU6、优酷更加强大和完善的 Android 视频网站娱乐多媒体客户端软件，并在持续升级中。

目前，我们正在务实而卓有成效的与音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络服务商合作，发展迅速，渴望有志之士的加入，和我们一起为成为世界最好的 Android 软件开发和咨询、培训公司而奋斗，为移动互联网和智能手机时代贡献力量！

联系我们

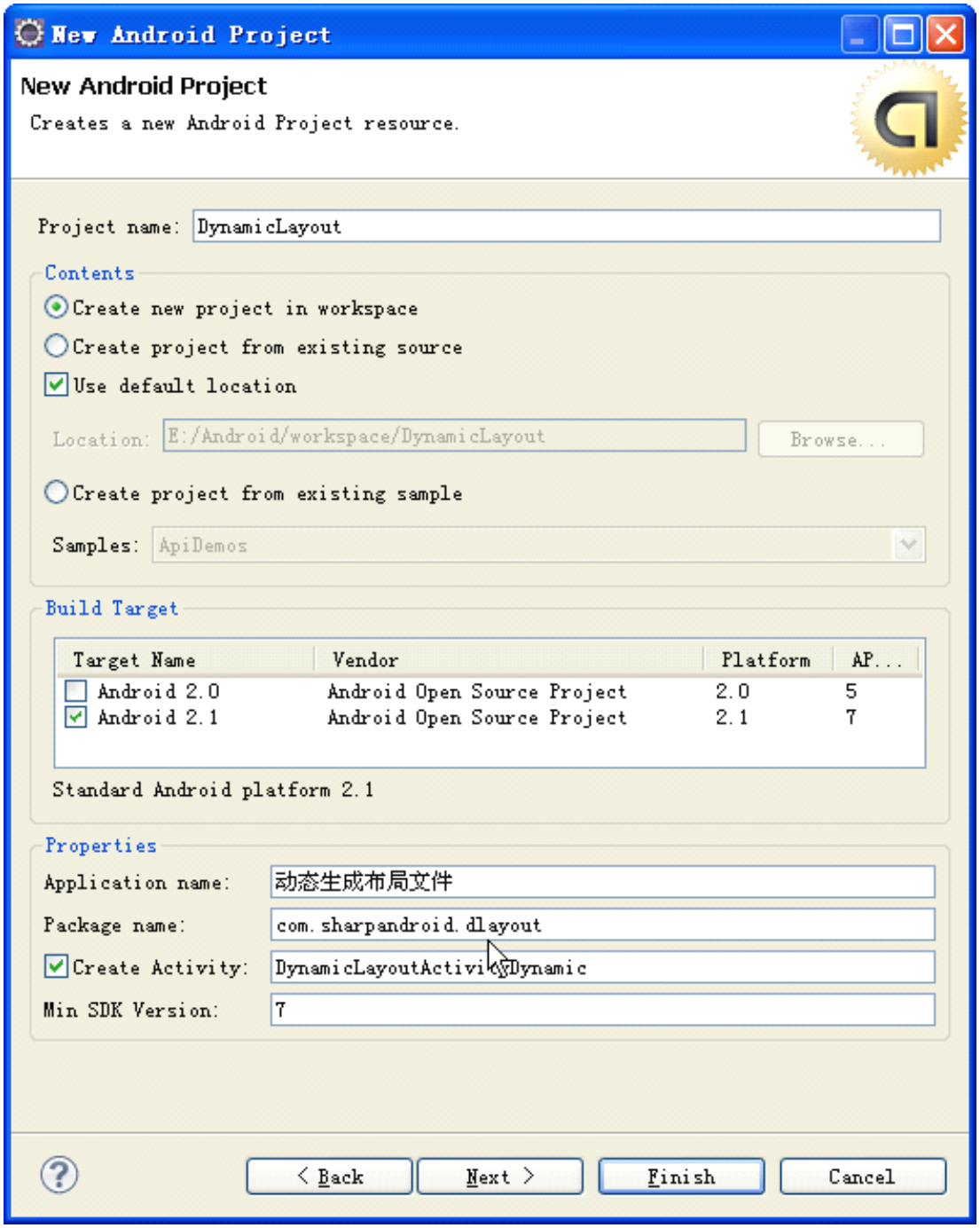
电话:15711060468
Email:guoshiandroid@gmail.com
博客: http://www.cnblogs.com/guoshiandroid/

1.3 硬编码生成界面

在 Android 中不仅可以使用 Xml 格式的布局文件进行界面设计，Layout 的存在本来就是要解决显示层的问题，可是大部分人总是习惯性的使用编码的方式进行实现对布局操作。这样做的不仅使 Activity 变的臃肿，也使代码变的杂乱无章。但有时会有些特殊需要，我们需要在代码中直接生成界面。但鉴于经典的 MVC 模式，界面属于视图层，我们建议在可能的情况下将视图层分离出来。不过有时也会用到在代码中编写视图，下面通过一个小例子来介绍这种使用方法。

创建项目

创建一个名为“DynamicLayout”的项目，界面如下：



编写 DynamicLayoutActivity.java

```
package com.sharpandroid.dlayout;

import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;
```

```
public class DynamicLayoutActivityDynamic extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout layout = new LinearLayout(this);
        LinearLayout.LayoutParams layoutparams =
            new LinearLayout.LayoutParams
                        (ViewGroup.LayoutParams.FILL_PARENT,
                            ViewGroup.LayoutParams.FILL_PARENT);
        setContentView(layout, layoutparams);
        TextView textView = new TextView(this);
        textView.setText("我从 java 代码中来，而不是 XML 布局文件！");
        LinearLayout.LayoutParams textViewparams =
                new LinearLayout.LayoutParams
                        (ViewGroup.LayoutParams.FILL_PARENT,
                         ViewGroup.LayoutParams.WRAP_CONTENT);
        layout.addView(textView, textViewparams);
    }
}
```

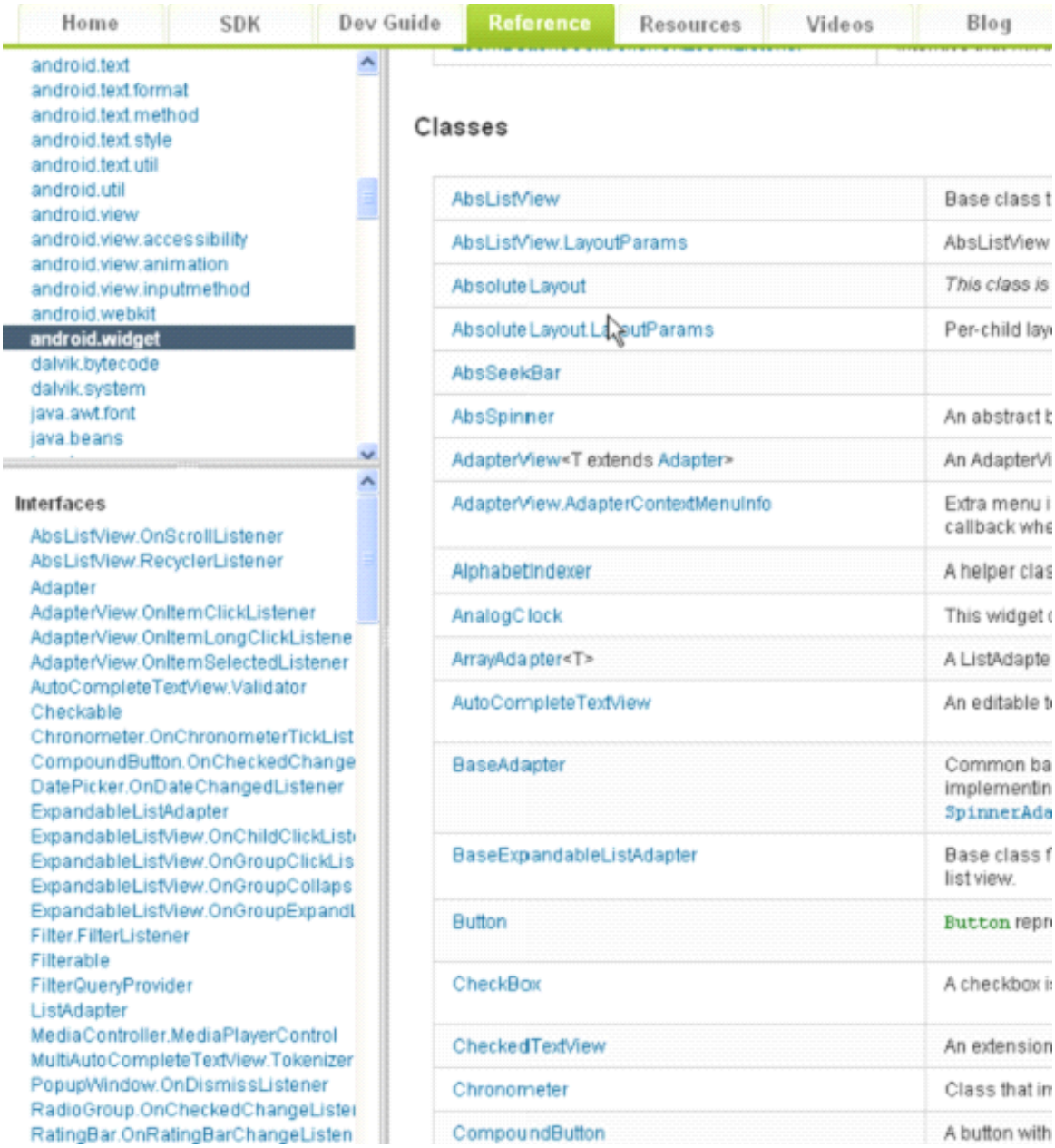
说明：

- `LinearLayout layout = new LinearLayout(this);`
  创建一个线性布局对象
- `LinearLayout.LayoutParams layoutparams = new LinearLayout.LayoutParams(ViewGroup.LayoutParams.FILL_PARENT, ViewGroup.LayoutParams.FILL_PARENT);`
  创建一个布局参数，确定该线性布局的宽和高。
- `setContentView(layout, layoutparams);`
  把该组件显示在屏幕上。
- `TextView textView = new TextView(this);`
  创建一个 TextView
- `textView.setText("我从 java 代码中来，而不是 XML 布局文件！");`
  设置 textView 上显示的文字信息
- `layout.addView(textView, textViewparams);`
  将 textView 添加入布局文件中。

执行程序

结果如下图：

该例子比较简单，但基本的使用思路是通用的。如果在开发中需要用到动态生成界面，可以查看帮助文档中的相关信息。方式如下图。

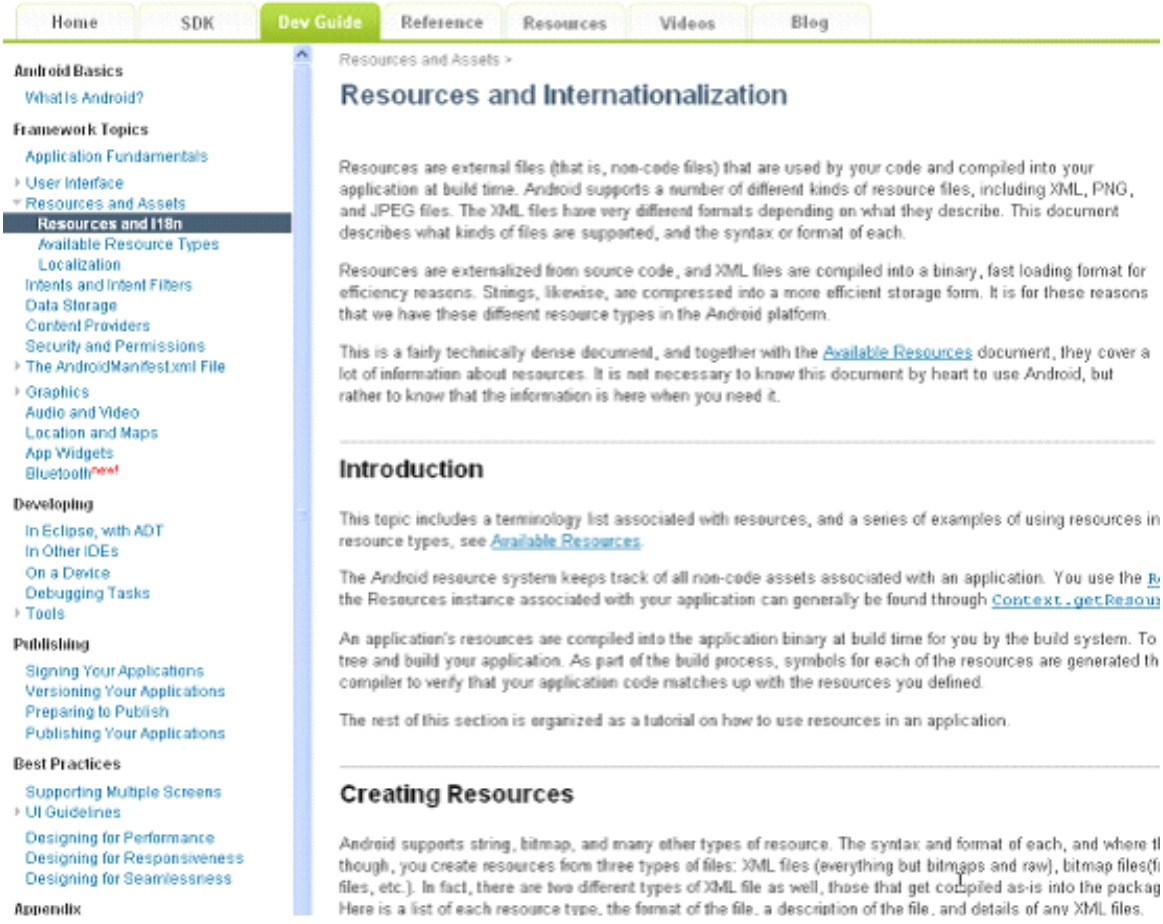


1.4 国际化

如今全球各国间的交流与合作越来越多，3G 开发作为与互联网类似，一个好的应用往往能够能满足许多国家用户的需求。但各国语言不同，因此必须针对不同的语言开发不同的界面，但应用的处理逻辑是不用改变的。Android 为我们提供了灵活的方式处理这一点，让我们很方便的就能使自己的应用支持多国语言实现国际化。

Android 中国际化的操作非常方便，只需要将欲国际化的资源，如文字、图片、界面等建立一个以原有存放的文件夹 values、drawable、layout 的后面加上“-语言代码-r 国家代码”，如“values-en-rUS”目录下的文件会在系统的地区设置为美国时被自动使用。如果没有相关的语言包系统将使用默认的 values 包中的资源。

可以自己查看文档上相关的资源。查看的方式如下图。



相关国家代码可以点击如下图中的框中的链接查看。

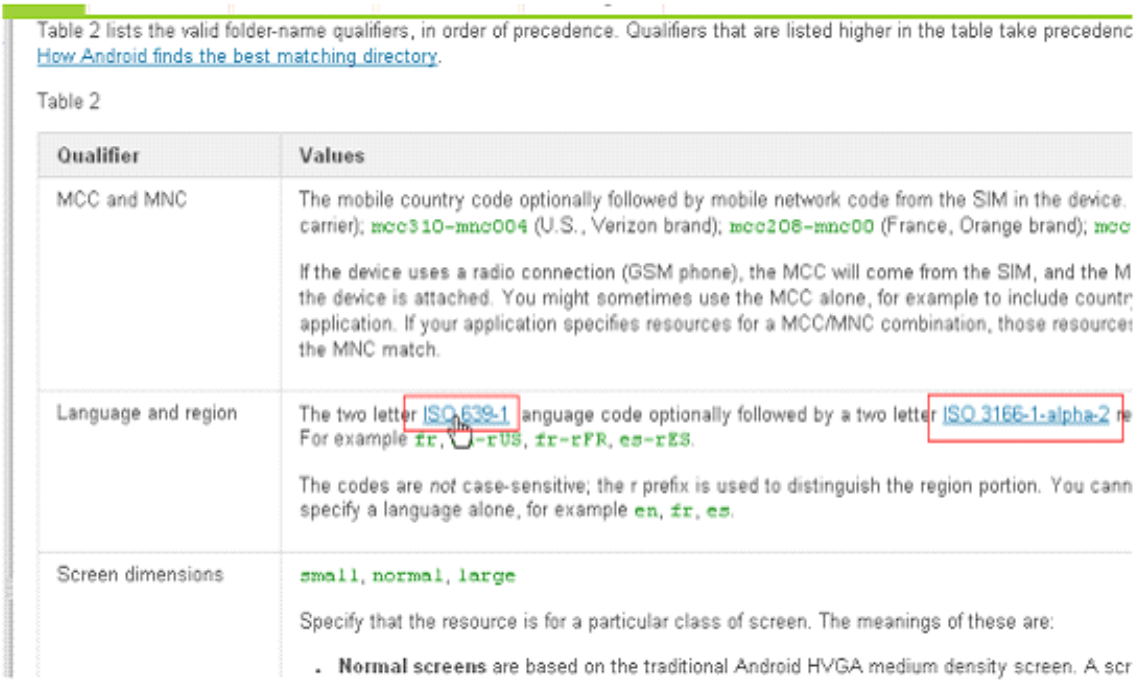
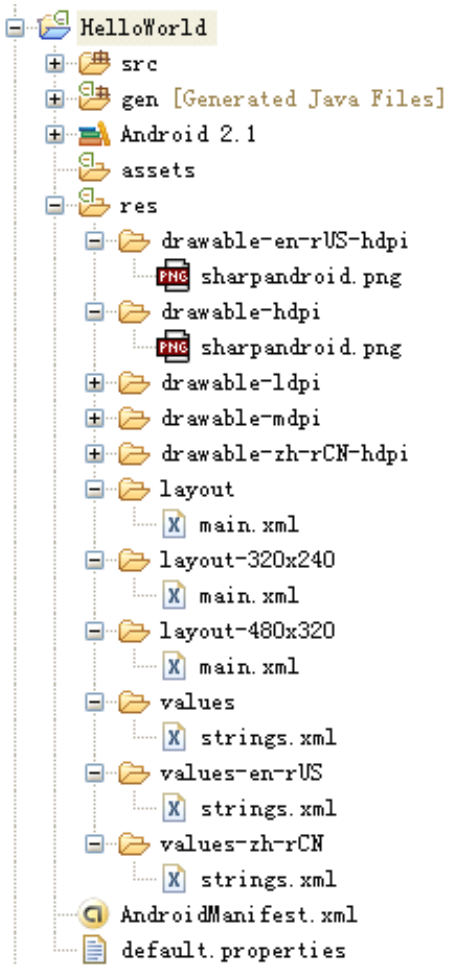


Table 2 lists the valid folder-name qualifiers, in order of precedence. Qualifiers that are listed higher in the table take precedenc
How Android finds the best matching directory.

Table 2

Qualifier	Values
MCC and MNC	The mobile country code optionally followed by mobile network code from the SIM in the device. carrier); mcc310-mnc004 (U.S., Verizon brand), mcc208-mnc00 (France, Orange brand); mcc If the device uses a radio connection (GSM phone), the MCC will come from the SIM, and the M the device is attached. You might sometimes use the MCC alone, for example to include countr application. If your application specifies resources for a MCC/MNC combination, those resource the MNC match.
Language and region	The two letter ISO 639-1 anguage code optionally followed by a two letter ISO 3166-1-alpha-2 re For example fr, -rUS, fr-rFR, es-rES. The codes are *not* case-sensitive; the r prefix is used to distinguish the region portion. You cann specify a language alone, for example en, fr, es.
Screen dimensions	small, normal, large Specify that the resource is for a particular class of screen. The meanings of these are: • **Normal screens** are based on the traditional Android HVGA medium density screen. A scr

下图为做国际化与自动适应屏幕分辨率处理后的项目的目录结构。

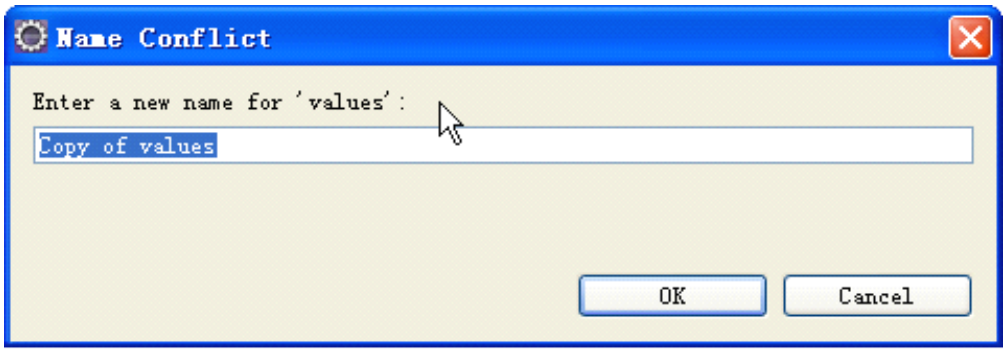


下面将我们的第一个项目 HelloWorld 做成支持中英文两种语言的国际化项目。步骤如下。

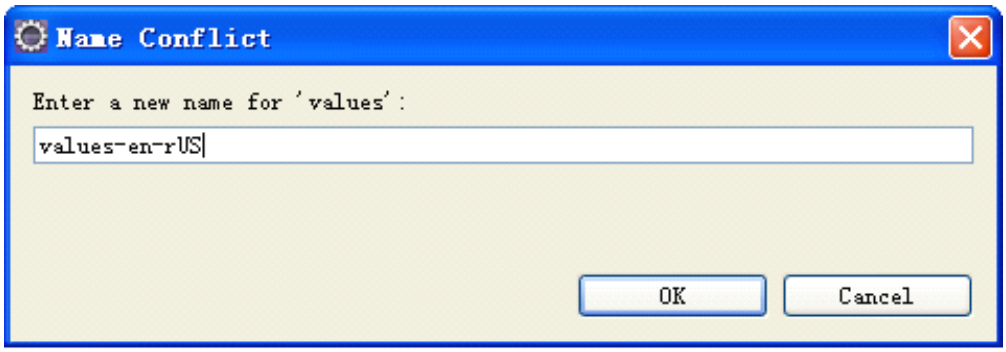
创建国际化文字信息文件

打开 HelloWorld 项目，进入 res/目录下。

拷贝 values 文件夹，将其粘贴进同目录下，出现如下对话框，



输入如下内容，



其中 en 代表英语，r 是国家地区的前缀，US 表示美国。

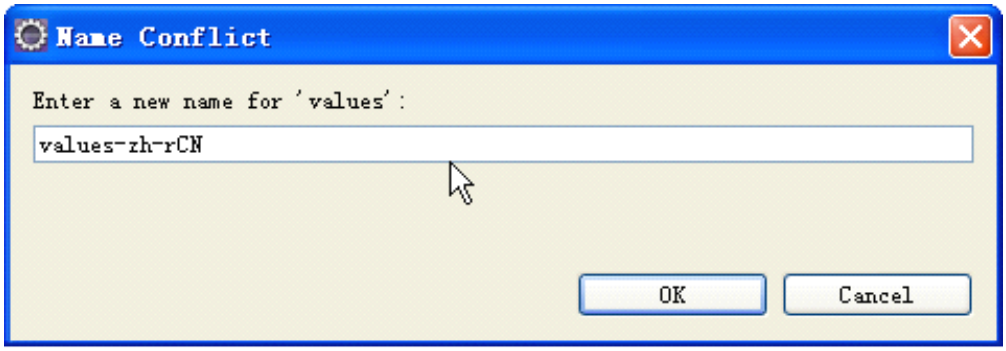
values-en-rUS 合起来就是当手机系统上的地区设置为美国时，程序中引用到 strings.xml 中文字的地方会使用该文件夹下的 strings.xml 的文本。

点击【OK】，进入“values-en-rUS”文件夹下。打开其 strings.xml 文件。进行修改。

修改所有文字为英文，修改后如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello,Android!</string>
    <string name="app_name">First Project! </string>
</resources>
```

再将 values 文件夹拷贝一次，粘贴到 res 文件夹中，此次输入的内容如下。



values-zh-rCN 代表中国汉语。

修改该文件夹下的 strings.xml 文件，内容如下。

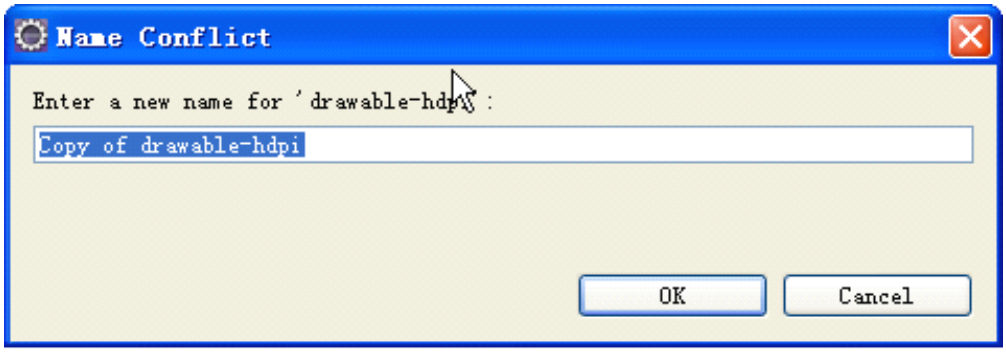
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">安致，你好！</string>
    <string name="app_name">第一个应用</string>
</resources>
```

此时，中国中文和美国英文两个国际化文件夹都已经创建完成。

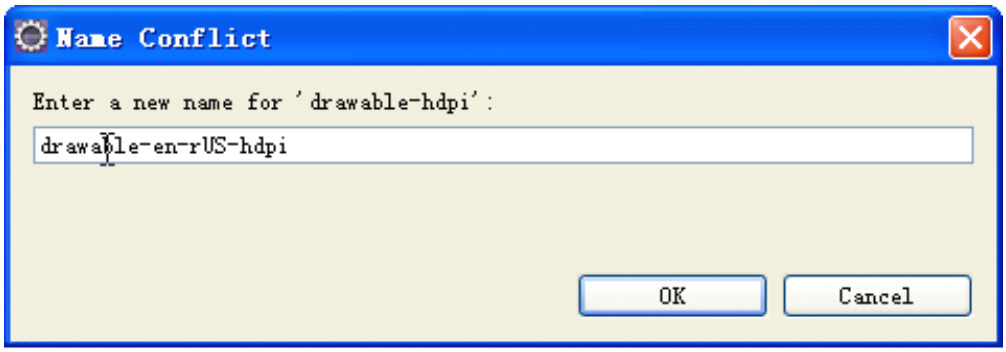
创建国际化图片文件

由于图片上可能有文字显示,或者国家之间风俗习惯不同,对图片的认知可能也有差异,因此图片也需要国际化。我们拷贝 res 下 drawable -hdpi 文件夹，粘贴到 res 下，出现如下对话框，

创建英文图片包。

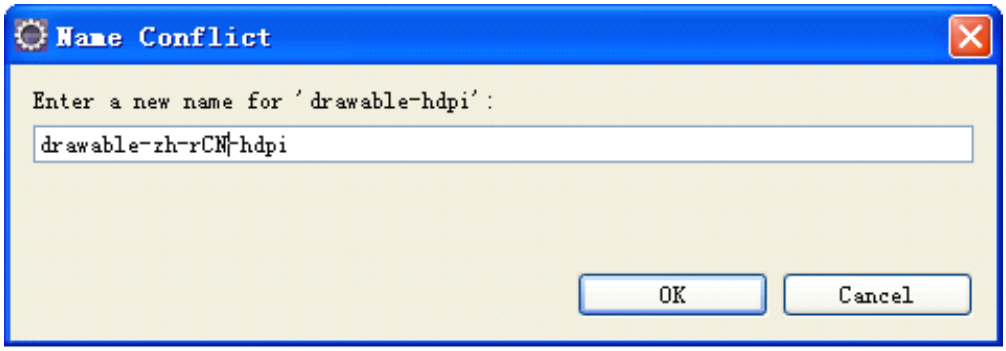


输入如下内容。



drawable-en-rUS-hdpi 含义与 values-en-rUS 一致。

创建中文图片包，输入如下。

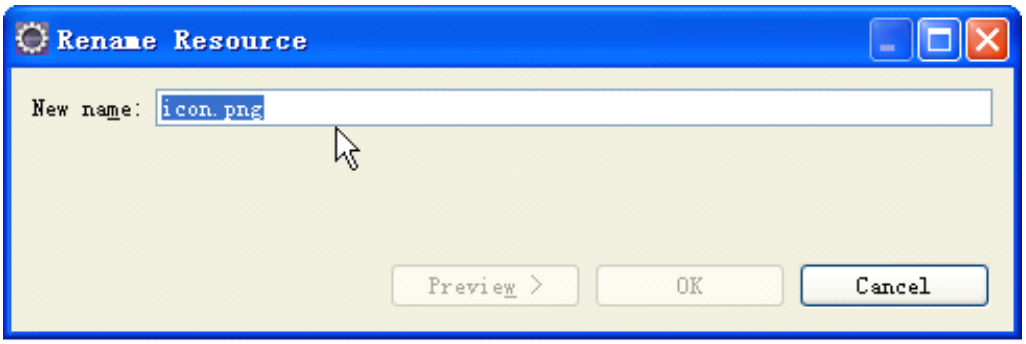


为了将两个包中的图片加以区别，我们在手机语言地区设为【English（Unitied States）】时，显示的图片为默认的图片。

语言设为【中文（简体）】时，显示的图片为 sharpandroid 图片。

将 drawable-en-rUS-hdpi 文件夹下的原来的“sharpandroid.png”文件删除，将 icon.png 重命名为“sharpandroid.png”。

重命名方式为在其文件上点击“F2”快捷键。如下图所示。



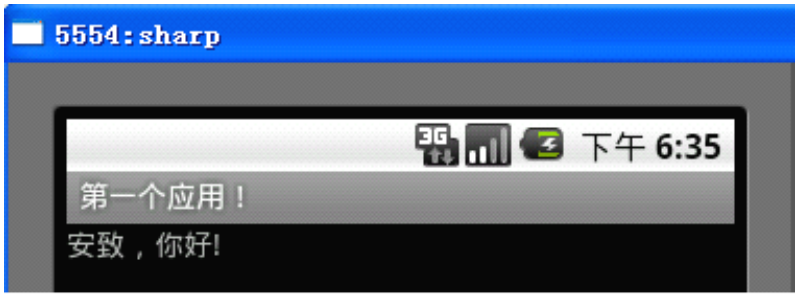
将 drawable-zh-rCN-hdpi 文件夹下的 icon.png 文件删除。

国际化图片文件包创建完成。

当然 res 下的其他资源，如 layout 等也可以完成国际化，方式与文字图片国际化方式基本一致。

第三步 执行程序

执行程序，此时手机系统上的语言地区为中国。运行效果如下图。



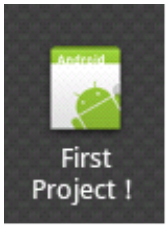
相应的打开“抽屉”，里面的显示方式为下图：

以上之文字来自于 values-zh-rCN 文件夹下的 strings.xml 文件中。
以上之图片来自于 drawable-zh-rCN-hdpi 文件夹下的 sharpandroid.png 文件。

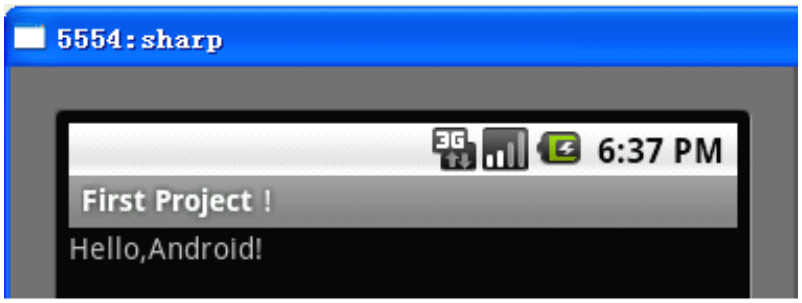
将系统的语言更换为【English（Unitied States）】，步骤如下，先进入【抽屉】，再打开【设置】。之后如下图。


5554:sharp
16:25
设置
搜索
语言和键盘
辅助功能
文字转语音
日期和时间
关于手机

再返回主界面，打开抽屉，查看该应用的图标及标题，如下图。



点击打开该程序，其界面如下：



以上之文字来自于 values-en-rUS 文件夹下的 strings.xml 文件中。
以上之图片来自于 drawable-en-rUS-hdpi 文件夹下的 sharpandroid.png 文件。

最后补充一点，如果没有对应的国家的 values、drawable 包，如没有 values-en-rUS，那么默认会使用 values 包下的内容。

1.5 程序界面自动适应屏幕分辨率

Android 中的显示单位

Android 中的显示单位应该有所了解，作如下简介：

- px (pixels)像素
  一般 HVGA 代表 320x480 像素，这个用的比较多。
- dip 或 dp (device independent pixels)设备独立像素
  这个和设备硬件有关，一般为了支持 WVGA、HVGA 和 QVGA 推荐使用这个，不依赖像素。
- sp (scaled pixels — best for text size)比例像素
  主要处理字体的大小，可以根据系统的字体自适应。

下面几个不太常用：

- in (inches)英寸
- mm (millimeters)毫米
- pt (points)点，1/72 英寸

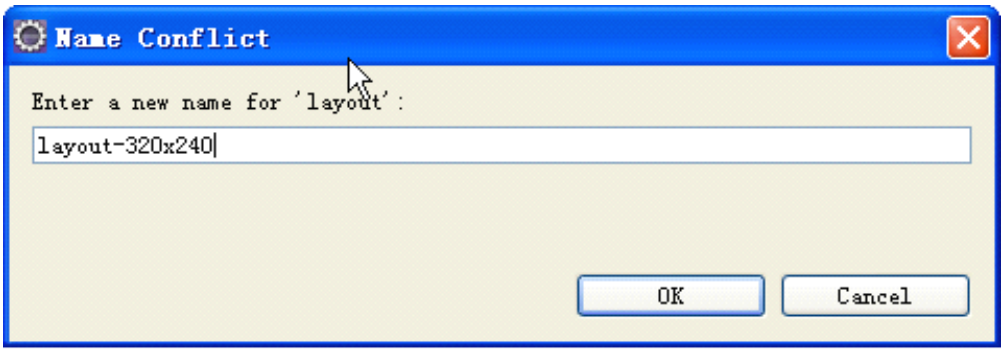
为了适应不同分辨率，不同的像素密度，推荐使用 dip，文字使用 sp。

为不同分辨率的手机创建界面

仍然在 HelloWorld 项目中进行改进。
首先进入 res 文件夹下。

创建一个名为“layout-320x240”文件夹，

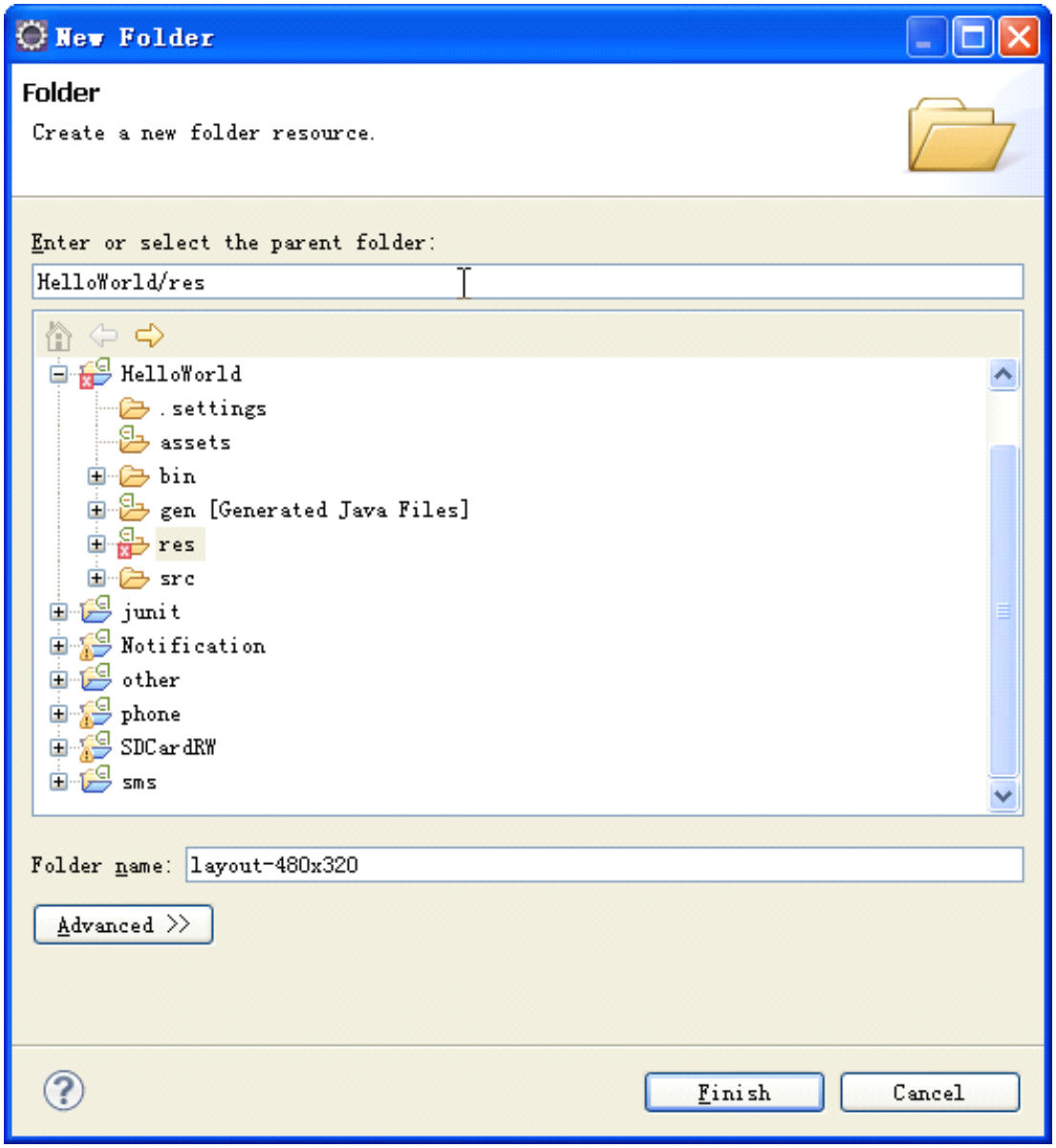
其中 320x240 是屏幕分辨率的大小，值得注意的是分辨率中大的数字必须写到前面，否则会产生语法错误。如 layout-240x320 的写法是错误的。



编写 main.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="320x240"
    />
</LinearLayout>
```

再创建一个文件夹“layout-480x320”



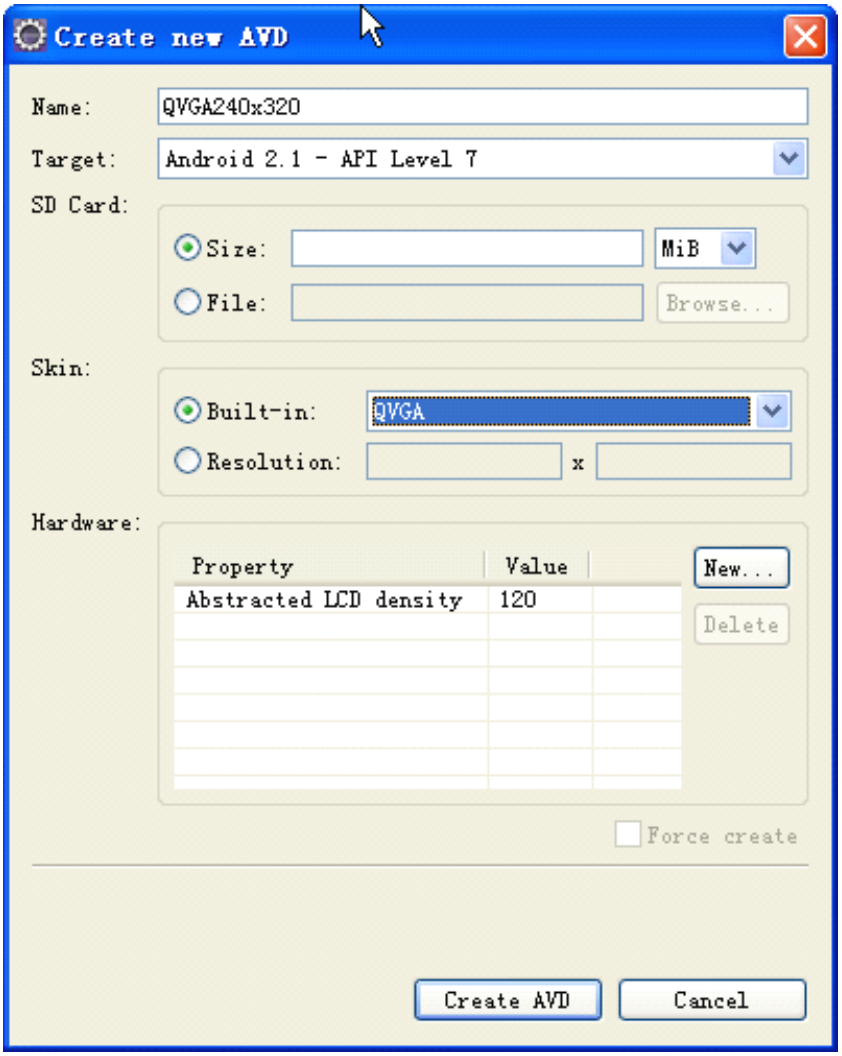
编写 main.xml 文件

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
    <TextView
    android:layout_width="fill_parent"
```

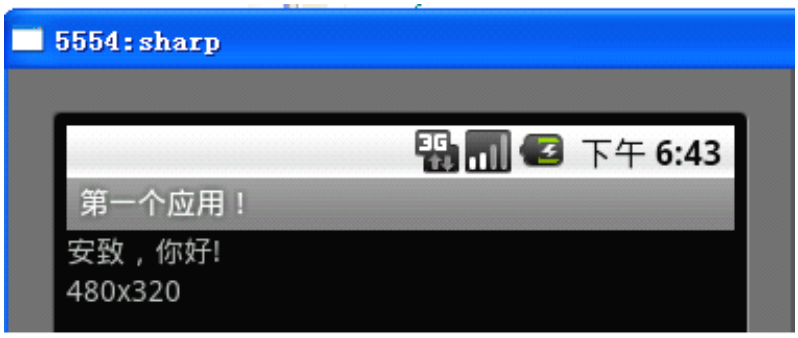
```
    android:layout_height="wrap_content"
    android:text="480x320"
    />
</LinearLayout>
```

执行程序

为了显示效果，我们再创建一个 AVD——QVGA240x320，分辨率为 320x240。

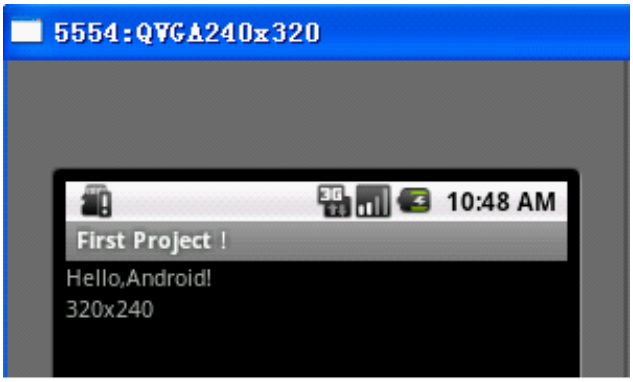


将 sharp 的文字调为中文，将项目发布到 sharp 上，运行结果如下图：



启动 QVGA240x320：

将项目发布上去。结果如下图：



如此实现了对不同分辨率的手机显示不同的界面。

1.6 Android 样式和主题(style&theme)

或许你对默认的样式和主题不是那么满意。为了解决这个问题，你可以创建自己的风格和主题。

- 风格是一套包含一个或多个格式化属性的整体，你可以把它们加诸于你布局中的单个元素之上。比如，你可以定义一个包含特定文本字体大小和颜色的风格，并将它单独施用于特定的视图元素。
- 主题也是一套包含一个或多个格式化属性的整体，但却应用于一个应用程序中的所有 Activity，或单独一个 Activity。比如说，你可以定义一个包含了特定窗口边框颜色和版面背景、以及一套字体大小和菜单颜色的主题。这个主题可以用于特定的 Activity 或整个应用程序。

风格与主题隶属于资源。Android 提供了一些默认的风格和主题供你使用，你也可以定制你自己的风格和主题资源。

Android 中的样式和 CSS 样式作用相似，都是用于为界面元素定义显示风格，它是一个包含一个或者多个 view 控件属性的集合。如：需要定义字体的颜色和大小。

在 CSS 中是这样定义的：

```
<style>
    .sharp{COLOR:#0000CC;font-size:18px;}
</style>
```

可以像这样使用上面的 CSS 样式：

```
<div class="sharp"> Android 样式和主题(style&theme)</div>
```

在 Android 中可以这样定义样式：

在 res/values/styles.xml 文件中添加以下内容：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name=“sharp”> <!-- 为样式定义一个全局唯一的名字-->
em name="android:textSize">18px</item> <!-- name 属性为样式要用在的 View 控件特有的属性 -->
        <item name="android:textColor">#0000CC</item>
    </style>
</resources>
```

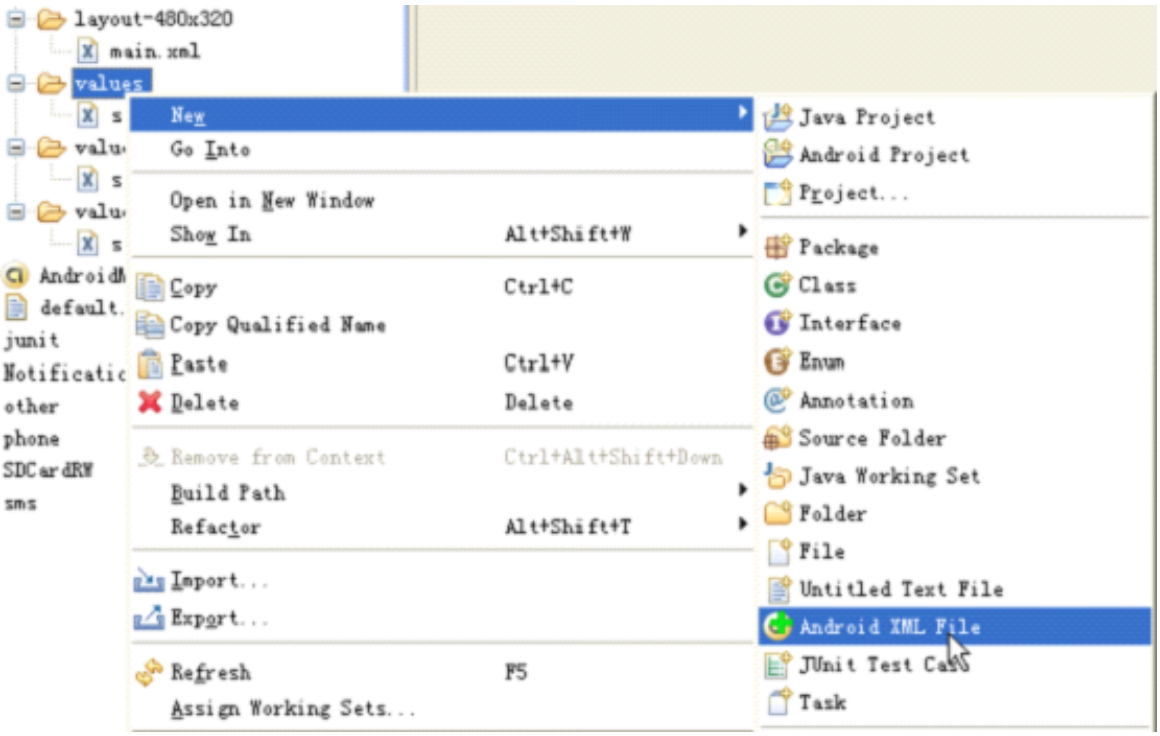
在 layout 文件中可以像下面这样使用上面的 Android 样式：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" ....>
    <TextView style="@style/sharp"
        .....  />
```

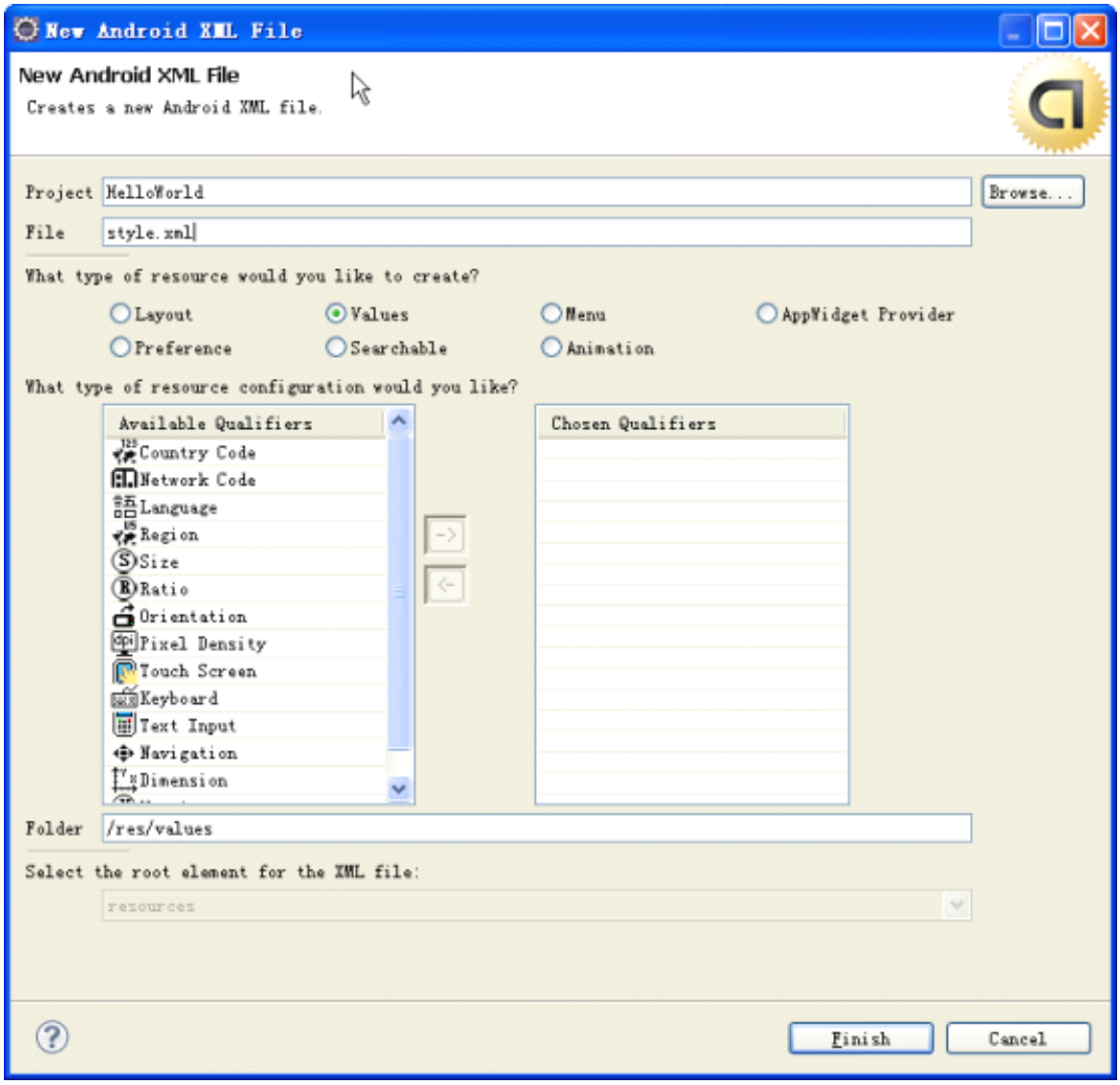
</LinearLayout>

1.6.1 为 HelloWorld 应用指定样式

在 HelloWorld 项目 res/values 文件夹中的创建如下文件。创建文件方式如下：



文件的名称可以根据喜好而定。并无严格要求。只需要以“.xml”结尾，且符合命名规范即可。



编写 style.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<style name="sharp"> <!-- 为样式定义一个全局唯一的名字-->
        <item name="android:textSize">18px</item> <!-- name 属性为样式要用在的
View 控件持有的属性 -->
        <item name="android:textColor">#0000CC</item>
    </style>
</resources>
```

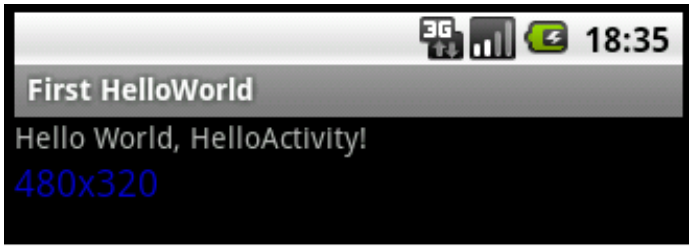
本例仅仅指定了两个属性的值，实际应用中可以把自己关心的重复较多的属性都在此定义。

在 layout-480x320 中 main.xml 的标签中引用该样式。代码如下：

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

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
    <TextView
    style="@style/sharp"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="480x320"
    />
</LinearLayout>
```

执行代码到 sharp 模拟器上。效果如下：



其显示风格与未引用 style 是不同的。

<style>元素中有一个 parent 属性。这个属性可以让当前样式继承一个父样式，当前样式可以继承到父样式的值。当然，如果父样式的值不符合你的需求，你也可以对它进行修改，如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="sharp">
        <item name="android:textSize">18px</item> <!-- name 属性为样式要用
                                                    在的 View 控件特有的属性 -->
        <item name="android:textColor">#0000CC</item>
    </style>
    <style name="subsharp" parent="@style/sharp">
        <item name="android:textColor">#FF0000</item>
    </style>
</resources>
```

1.6.2 为应用指定主题

Android 中主题也是用于为应用定义显示风格，它的定义和样式的定义相同，主题定义文件放于 values 目录下，如下：

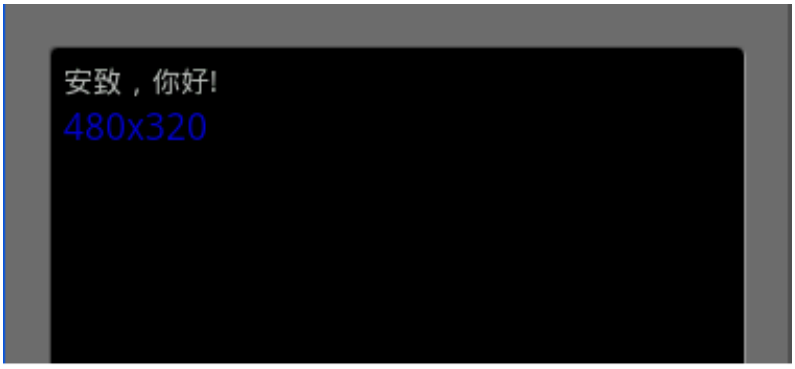
sharptTheme.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<style name=“sharpTheme">
    <item name=“android:windowNoTitle” >true</item> <!－无标题-->
    <item name=“android:windowFullscreen”>?android:windowNoTitle</item><!－ 全屏显示-->
</style>
</resources>
```

上面“?android:windowNoTitle”中的问号用于引用在当前主题中定义过的资源的值。下面代码显示在 AndroidManifest.xml 中如何为应用设置上面定义的主题：

```
<application android:icon="@drawable/icon" android:label="@string/app_name"
     android:theme="@style/sharpTheme">
   ......
</application>
```

效果如下，没有标题栏，并且全屏显示：



除了可以在 AndroidManifest.xml 中设置主题，同样也可以在代码中设置主题，如下：

```
setTheme(R.style.SharpTheme);
```

尽管在定义上，样式和主题基本相同，但是它们使用的地方不同。样式用在单独的 View，如：EditText、TextView 等；主题通过 AndroidManifest.xml 中的<application>和<activity>用在整个应用或者某个 Activity，主题对整个应用或某个 Activity 进行全局性影响。如果一个应用使用了主题，同时应用下的 view 也使用了样式，那么当主题与样式属性发生冲突时，样式的优先级高于主题。

另外 android 系统也定义了一些主题，例如：

<activity android:theme=“@android:style/Theme.Dialog”>，该主题可以让 Activity 看起来像一个对话框，如果需要查阅这些主题，可以在文档的 reference—>android—>R.style 中查看。