【eoeAndroid 特刊】

第五期：资源与应用国际化

发布版本： Ver 1.0.0(build 2009.07.05)

本篇简介

android 是个国际化的手机操作系统，android market 也是个世界共享的大市场，要想使的自己的应用被更多人用到，你必须考虑的一个问题就是“国际化”，说到国际化，其不仅仅代表界面显示的语言文字，其还包括诸如界面风格，图片等等资源的国际化！

android 刚推出的时候，在这个方面做的并不完善，随着 1.5 版本 SDK 的发布，其中涉及到国际化的原理和模型已经俱备了雏形及其方向，虽然这个方面在后续版本的 SDK 可能会做些调整和完善，但其大方向是不会变化的，于是我们认为是时候了解、掌握、实践国际化了。

所以本期 eoeAndroid 特刊组织策划了这个主题，我们将翻译一些文章，并参考一些例子自己撰写一批实际的例子讲解国际化的步骤和过程。本期内容包含但不限于如下方面：

1. 源和资产(Resources and Assets)
   -大体介绍一下 Android 上的资源系统
2. Android 中可用的资源类型
   -Android 可以使用哪一些资源类型。
3. Resources and Internationalization（资源和国际化）
   -通过 Android 的 framework 来做国际化
4. 本地化你的 Android 应用程序
   -如何将你的 Android 应用程序本地化
5. 在 Android 中轻松实现横竖屏的布局
   -在横竖屏切换的时候如何获取事件
6. 如何获取当前 Locale ， 设定 Locale
   -当前的 Local，我们如何用常规的方法来获取
7. 如何在代码中强行指定自己 App 的 locale
   -指定我们的 app 使用一种 Local
8. Android Applications Localization Helper (Android 本地化助手)
   -一个小工具让你的国际化工作，事半功倍

希望本期内容能让更多的开发者熟悉 Android 资源与应用国际化的其相关内容，了解如何使用 Android 上的资源，如何让自己的应用程序国际化、本地化。

活动发起地址：

【eoeAndroid 特刊】策划 第五期：资源与应用国际化

http://www.eoeandroid.com/viewthread.php?tid=854&extra=page%3D1

致谢：

本期专题得到如下同学的大力支持和积极响应，谢谢你们辛苦劳动，谢谢你们为 Android 发展和普及做出的贡献。

- Iceskysl
- xiangyong
- pcr
- 游利卡(http://www.cnblogs.com/pcedb0189/)

1.源和资产(Resources and Assets)

http://developer.android.com/guide/topics/resources/index.html
WangHeyun

（assets 实在是不知道译为什么好，暂且译为资产）

源是 Android 应用程序的一部分。总体上来说，源就是你想要在应用程序中引入和涉及的一些外部元素，如图像，音频，视频，文本，字符串，布局文件，主题等等。每一个 Android 应用程序都包含一个源目录（res/）和一个资产目录(assets/)。资产很少被用，因为和它相关的应用程序很少。当你的程序去读取原始的字节时，你只需要将数据保存为一个资产。源和资产的目录以同样的优先等级存放在 Android 目录树的源代码目录(src/)下。

在表面上，源和资产没有什么不同，但是总体上来说，用源来存储你的外部内容比用资产来存储更频繁。在编译时，真正的不同实际上在于应用程序通过 R 类可以轻易的访问源（res/）目录下的任何文件。因此，在资产(assets/)目录下的文件将保持它的原始的文件格式，为了能够读取它们，必须用 AssetManager 以字节流的形式读取文件。因此，在源目录下的文件和数据更容易被访问。

在这篇文章中，你会找到各种被用在 Android 应用程序中的标准源以及如何从代码中引用它们。Resources 和 Internationalization 就是你的起点，未来了解更多关于如何利用源，这篇 Available Resource Types 文章提到了源的各种类型以及如何设定引用

王河云

声明:由于本篇文章涉及到了数量巨大的专业术语，有一些翻译起来着实有很多困难。Android 的显示使用的是与 web 类似的方式，所以如果想探究这里介绍的一些专业术语，烦请大家动员自己的的力量，谢谢理解！

2.Android 中可用的资源类型

By 游利卡

这一章描述了 Android 中可用的资源类型，你可以在你的代码和应用中检验这些类型。
如果想知道更多细节的问题，请参考本期特刊的另一篇文章——资源与国际化。

比较简单的可用资源

这些简单的资源包括字符串，还有一些没有定义的资源。因此，这些资源不仅仅可以作为标准的资源，也可以直接用来支持风格和主题的设置，还有 XML 和布局中的一些属性的设置。

2.1 颜色

所有的颜色信息都支持十六进制字符，我们可以把这些以文本的形式放到文件中去。一个颜通信息常以首字母#为紧接着是红绿啦三色的信息。有关颜色更多的信息，建议大家可以去 Google 一下相关的信息。

- #RGB
- #ARGB
- #RRGGBB
- #AARRGGBB

如果想要获取颜色的 ID，你可以使用 Resources.getColor() 这个方法

资源文件格式:需要定义一下 xml 文件的标准<?xml version="1.0" encoding="utf-8"?>，这里第一个参数定义的是 xml 的版本，第二个则是使用何种方式编码。颜色等标签都应该放到<resources>这个标签下。

路径 res/values/*colors*.xml (文件名随意)

编译后的资源 指向 Java int。

与资源文件相关的名称

- **Java:** `R.color.some_name`
- **XML:** `@[package:]color/some_name` (这里的 some_name 指向一个颜色的代码)

Syntax

```
<color name=color_name>#color_value</color>
<color>
```

所有的颜色全部是用网站通用的编辑颜色的方法，通常只有一种属性。

- name-这里指的是 xml 中的颜色信息的名称。

XML 示例

下面的代码声明了两个颜色，第一个为不透明的，第二个透明

```
<resources>
   <color name="opaque_red">#f00</color>
   <color name="translucent_red">#80ff0000</color>
```

示例代码的使用。

在 Java 中

```
// Retrieve a color value.
int color = getResources.getColor(R.color.opaque_red);
```

XML 的示例

```
<TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:textAlign="center"
          android:textColor="@color/translucent_red"
          android:text="Some Text"/>
```

2.2 字符串与风格

字符串可以选择各种简单的格式 simple formatting，并且存储并重复调用。你可以使用标准的 HTML 标签，例如<b>、<i>、和<u>，来添加任何一种格式到你的字符串资源中。为了保证这些没有用任何修饰的字符创能够被 toString()方法，转换成有序字符。方法必须能够处理这些加载字符前面的修饰符标签。

如果想要在通过资源的 ID 来调用这些字符串，你可以使用 Context.getString()方法

注意:这里你必须加双引号或者是引号。

```
<string name="good_example">"This'll work"</string>
<string name="good_example_2">This\'ll also work</string>
<string name="bad_example">This won't work!</string>
<string   name="bad_example_2">XML   encodings   won&apos;t   work
either!</string>
```

- **资源文件格式:**需要定义一下 xml 文件的标准 <?xml version="1.0" encoding="utf-8"?> ，这里第一个参数定义的是 xml 的版本，第二个则是使用何种方式编码。字符串等标签都应该放到<String>这个标签下。
- **路径** res/values/*strings*.xml (文件名随意)
- **编译后的资源** 指向 Java int。
- **与资源文件相关的名称**
- **Java:** R.string.*some_name*
- **XML:** @[*package*:]string/some_name (*some_name* 是用来解析的字符字符串)

Syntax

```
<string name=string_name>string_value</string>
<string>
```

可以选择标签的字符串，这里有一些属性

- name-这里指的是 xml 中的字符串的名称。

XML 示例

以下的代码声明了两种字符串:第一种-简单的没有格式的文本文档(将简单的字符串作为一段有序的文字);第二种包括了又格式字符串信息(主要作用于复杂的结构)。如果你使用 PC 上的工具列入 Eclips 来编辑 XML 文件，那么这些 XML 文件会被自动的突出，你可以使用 ContextGetString()和 fromHTML(String 来重新调用这些资源)然后重新格式化他们。

```
<resources>
   <string name="simple_welcome_message">Welcome!</string>
```

```
    <string  name="styled_welcome_message">We  are  <b><i>so</i></b>
glad to see you.</string>
</resources>
```

Java 代码的示例

```
// Assign a styled string resource to a TextView
// on the current screen.
CharSequence str = getString(R.string.styled_welcome_message);
TextView tv = (TextView)findViewByID(R.id.text);
tv.setText(str);
```

XML 代码

```
<TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:textAlign="center"
          android:text="@string/simple_welcome_message"/>
```

2.3 使用有风格的有格式的字符串

当你要想来创建一组有风格的文字资源时，就需要加一些格式。但在我们编译的时候，如果不能跳过这些字符前面的风格信息，你是无法直接使用这些字符串的。我们在工作的时候，风格字符串是作为额外的标签存储的，而在格式化之后，他们才能将我们的字符串转换为我们需要的格式，这是这些标签已经转换为另外一种形式存在了。

.

如果想要使用这些有格式的字符串，你应该这样做

1.存储的你应该像下面的例子一样来添加修饰标签。

```
<resources>
  <string  name="search_results_resultsTextFormat">%1$d  results  for
&lt;b>&amp;quot;%2$s&amp;quot;&lt;/b></string>
</resources>
```

2.在上面的例子中 %1$d 是一个十进制数字, %2$s 是个字符串。

为了让我们的程序知道，我们的字符有没有使用诸如'<'或者'&'的字符，你需要用 htmlEncode(String)方法。

```
String escapedTitle = TextUtil.htmlEncode(title);
```

使用 String.format()来格式化这些 HTML 额外那本，然后使用 fromHtml(String) 来转换这些 HTML 文本。

```
String                      resultsTextFormat                      =
getContext().getResources().getString(R.string.search_results_resul
tsTextFormat);
String   resultsText   =   String.format(resultsTextFormat,   count,
escapedTitle);
CharSequence styledResults = Html.fromHtml(resultsText);
```

你可以再不同的 XML 文件中，制定根据不同的屏幕元素定义的规格。我们需要了解一些相关的尺度大小，比如说 10px、2in、5sp。这里列出了这些度量单位。

（下列介绍的单位过于专业，而相应的解释也过于简单，所以这里大部分没有被翻译）

Px

像素

in

英寸-屏幕的物理大小

mm

毫米-物理上的屏幕的毫米

pt

点

dp

依赖于设备的像素

sp

带比例的像素

这些单位不仅被原生态的资源文件使用，也可以被属性资源使用。你可以，但是创建的时候必须符合以下的格式。

资源文件格式:需要定义一下 xml 文件的标准<?xml version="1.0" encoding="utf-8"?>，这里第一个参数定义的是 xml 的版本，第二个则是使用何种方式编码。颜色等标签都应该放到< dimen>这个标签下。

路径 res/values/dements.xml(文件的名字是随意的，但是不要把所有的单位都放进一个文件，要专注一些)

编译后的资源 指向 Java int。

与资源文件相关的名称

- **Java:** R.dimen.*some_name*
- **XML:** @[*package*:]dimen/*some_name* (where *some_name* is the *name* of a specific <dimen> element)

Syntax

```
<dimen name=dimen_name>dimen_value</dimen>
```

<dimen>

有效的单位值

- name-这里指的是 xml 中单位的名称。

XML 示例声明

下面的代码声明了单位

```
<resources>
    <dimen name="one_pixel">1px</dimen>
    <dimen name="double_density">2dp</dimen>
    <dimen name="sixteen_sp">16sp</dimen>
</resources>
```

Example Code Use

Java 代码

```
float dimen = Resources.getDimen(R.dimen.one_pixel);
```

XML 代码

```
<TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:textSize="@dimen/sixteen_sp"/>
```

2.4 图片资源

图片资源可以使用 Resources.getDrawable() 将图片显示到屏幕上，有以下的图片资源可以被创建。

位图

Android 支持的位图有 png(最先支持)，jpg(支持)，gif(不支持)。位图自身被编译前，调用时不需要加后缀名，比如说，res/drawable/my_picture.png 被调用的时候使用这样的名称 R.drawable.my_picture

图片文件的格式 png(最先支持)，jpg(支持)，gif(不支持)

图片的路径: res/drawable/*some_file*.png or *some_file*.jpg or *some_file*.gif.

被编译后 资源指向一个 BitmapDrawable.

资源相关的名称:

Java: R.drawable.*some_file*

- **XML:** @[*package*:]drawable/*some_file*

For more discussion and examples using drawable resources, see the discussion in 2D Graphics.

如果想了解更多关于 drawable 资源的例子，参见 2D Graphics.

2.5 图片的颜色

你可以创建一个形状无论是矩形还是圆形，你都可以给他们上色，这个元素可以被放在 res/values 的任意部分

资源文件格式:需要定义一下 xml 文件的标准<?xml version="1.0" encoding="utf-8"?>，这里第一个参数定义的是 xml 的版本，第二个则是使用何种方式编码。颜色等标签都应该放到< drawable >这个标签下。

路径 res/values/colors.xml (文件名随意;标准的做法是将 PaintDrawable 和 numeric color values.放进一个文件里)

编译后的资源 指向 PaintDrawable.

与资源文件相关的名称

- **Java:** R.drawable.*some_name*
- **XML:** @[*package*:]drawable/*some_name* (where *some_name* is the name of a specific resource)

Syntax

```
<drawable name=color_name>color_value</drawable>
<drawable>
```

有效的颜色信息

- name-这里指的是 xml 中图片的名称。

XML 示例

以下的代码中声明了几种图片的颜色

```
<resources>
```

```
    <drawable name="solid_red">#f00</drawable>
    <drawable name="solid_blue">#0000ff</drawable>
    <drawable name="solid_green">#f0f0</drawable>
</resources>
```

Java 代码

```
// Assign a PaintDrawable as the background to
// a TextView on the current screen.
Drawable redDrawable = Resources.getDrawable(R.drawable.solid_red);
TextView tv = (TextView)findViewByID(R.id.text);
tv.setBackground(redDrawable);
```

XML 代码

```
<TextView android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:textAlign="center"
          android:background="@drawable/solid_red"/>
```

2.6Nine-patch(被拉伸或者缩小的)图像

Android 支持图像的拉伸，这个叫做 NinePath

Android 支持图像的拉伸，这个叫做 NinePath。如果你的 PNG 图片无法使用 Android 需要的大小，那么他就会被重新设定大小，并进行拉伸，直到适合显示出来。这样你就可以分配这些图片到 view 的背景了。

这里有一个把图片的例子，按钮必须吧图片缩小

资源文件格式: PNG — 单个的资源文件。

路径 res/drawable/*some_name*.9.png (只能是 PNG 图片)

编译后的资源 指向 NinePatchDrawable.

与资源文件相关的名称

- **Java:** R.drawable.*some_file*
- **XML:** @[*package*:]drawable.*some_file*

想知道更多关于 NinePatch drawables, 参见 2D Graphics.

动画

Android 可以图形或者一些列图形上显示一些简单的动画。这些包括旋转、暗化、移动还有拉伸。

资源文件的格式: 只能是 xml 文件， <?xml> 这个必须被声明。

路径: res/anim/*some_file*.xml

编译后的资源 指向 Animation.

资源相关的名称:

- **Java:** R.anim.*some_file*
- **XML:** @[*package*:]anim/*some_file*

Syntax

该文件必须要有一个独立的根元素：该根元素也可以是<alpha>, <scale>, <translate>,

<rotate>中的单独的一个加载器元素或者是 <set> 元素，<set>元素包含了这些元素的组合（也可以包含其他<set>）。默认地，所有元素都同时被应用。要让它们按顺序发生，你必须指定 startOffset 属性。

```
<set android:shareInterpolator=boolean>  // 如果要使用多个 Tag 标签那么 set 就必须的。
   <alpha android:fromAlpha=float
          android:toAlpha=float >   |
   <scale android:fromXScale=float
          android:toXScale=float
          android:fromYScale=float
          android:toYScale=float
          android:pivotX=string
          android:pivotY=string >    |
   <translate android:fromX=string
              android:toX=string
              android:fromY=string
              android:toY=string >   |
   <rotate android:fromDegrees=float
           android:toDegrees=float
           android:pivotX=string
           android:pivotY=string > |
   <interpolator tag>
   <set>
</set>
```

元素和属性

<set>

一个可以递归存放自己或是其他动画的一个容器。描绘了一个动画序列 AnimationSet. 你可以包含多个同样的子元素或是你喜欢的不同类型的元素。它支持以下属性：

·shareInterpolator - 是否和所有直系子元素共享相同的动画加载器。

<alpha>

一个渐变褪色动画. 表现一个 AlphaAnimation. 它支持以下属性：

·fromAlpha -范围 0.0—1.0, 0.0 表示全透明

·toAlpha -范围 0.0—1.0, 0.0 表示全透明

<scale>

一个伸缩尺寸的动画。表现一个 ScaleAnimation.你可以通过指定 pivotX 和 pivotY 来指定图片的中心点（锚点）,以该锚点来向外或是向内伸缩。因此，例如：如果那些属性为零（pivotX＝0，pivotY＝0--->左上中心），所有的伸缩将会向下和向右。scale 支持以下属性：

·fromXScale - 起始时 X 方向的尺寸，1.0 表示无伸缩。

·toXScale - 终止时 X 方向的尺寸, 1.0 表示无伸缩。

·fromYScale - 起始时 Y 方向的尺寸，1.0 表示无伸缩。

·toYScale - 终止时 Y 方向的尺寸, 1.0 表示无伸缩。

·pivotX - X 坐标上锚点中心。

·pivotY - X 坐标上锚点中心。

<translate>
一个垂直/水平向的移动动画。表现一个 TranslateAnimation。支持任意以下三种格式的属性：
①取值从-100 到 100，以%结尾，指定一个相对于自己的百分率；
②取值从-100 到 100，以%p 结尾，指定一个相对于父布局的百分率；
③一个无后缀浮点型，指定一个绝对数值 。
·fromXDelta -起始 X 位置。
·toXDelta --结束 X 位置。
·fromYDelta --起始 Y 位置。
·toYDelta --结束 Y 位置。
<rotate>

A rotation animation. Represents a RotateAnimation. 支持以下属性:

- *fromDegrees* – 起始值
- *toDegrees* –结束值
- *pivotX* – X 坐标, 像素中， (0,0) 指的是最左上角.
- *pivotY* – Y 坐标, 像素中，(0,0) 指的是最右上角

<interpolator tag>

你可以使用在 R.styleable 子类元素中定义的任何 interpolator . 例如包括 <CycleInterpolator>, <EaseInInterpolator>,好<EaseOutInterpolator>. 这些对象定义一个区县如何快速的用一个无形的动作来占据时间线 (先快后满,一开始很快，速度逐渐降低)

每一个元素额外的属性, 这些元素 <alpha>, <scale>, <translate>, <rotate>, 和<set> 都支持以下属性 (继承自 Animation 类):

duration

支持时间，以毫秒来计算

startOffset

启动时间，以毫秒来计算

fillBefore

当设置为 ture 后，在真正的动画开始前的转场动画。

fillAfter

当设置为 true 后，在真正的动画开始后的转场动画。

repeatCount

定义了每段时间动画的节奏

repeatMode

定义了当重复数大雨 0 时，动画播放的行为。可以选择是重启动画或者是倒转动画。

zAdjustment.

定义了动画播放时的 z 轴效果(正常、顶部、底部。(译者:这里应该和 3D 有关)

interpolator

你可以为每一个元素建立 interpolator，来决定在播放时间内他们播放的快慢。 比如, 对于 EaseInInterpolator 开始时快，结束时慢, 对于 EaseOutInterpolator 的反转. R.anim 中有一系列的 interpolators. 为了解析他们你需要使用这个前缀 @android:anim/*interpolatorName*.

更多讨论详见 2D Graphics。

2.7 菜单

应用的 menus（可以选择的 menu，Context Menu 或是 Sub menu）可以像被其他的 XML 中一样被定义，并且被的应用通过 MenuInflater.扩展

资源文件的格式: 只能是 xml 文件， <?xml> 这个必须被声明。

路径: res/menu/*some_file*.xml

编译后的资源 指向 Menu 或其子类

资源相关的名称:

- **Java:** R.menu.*some_file*

Syntax

这个文件必须有一个根元素<menu> .无论如何这里都必须有三个元素 <menu>, <group> 和<item>。 <item>和 <group> 他们必须有<menu>这个子元素。 但是<item>也可以作为 <group>子元素，另一个 <menu> 可以作为<item> 的子元素，也就是创建子菜单。

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

    <item android:id="@+id/example_item
          android:title="Example Item"
          android:icon="@drawable/example_item_icon" />

    <group android:id="@+id/example_group">
        <item android:id="@+id/example_item2
              android:title="Example Item 2"
              android:icon="@drawable/example_item2_icon" />
    </group>

    <item android:id="@+id/example_submenu
          android:title="Example Sub Menu" >
        <menu>
            <item android:id="@+id/example_submenu_item
                  android:title="Example Sub Menu Item" />
        </menu>
    </item>

</menu>
```

Elements and Attributes

所有的属性都被 Android 的命名空间定义(e.g., *android:icon="@drawable/icon"*)

<menu>

这个元素是一个菜单，包含 <item> 和 <group> 节点，注意他们没有属性。

<group>

一个菜单及包含 <item> elements. 属性可被识别

- *id* – 唯一可用标识。
- *menuCategory* – 对应 Menu CATEGORY_* 内容的项 — 定义了一组的优先权. 有效值: *container*, *system*, *secondary*, and*alternative*.

- *orderInCategory* – 一个整型值用来定义菜单中每一项的序列
- *checkableBehavior* - 是否被用户点击. 有效值: 没有选项类型，单选 / 或是单选按钮型 ， 非单选 / 复选类型
- visible – 是否可见 *true* or *false*.
- *enabled* –是否可用. *true* or *false*.

<item>

A menu item. 包含<menu> 元素 (作为子菜单). Valid attributes:

- *id* – 唯一可用标识.
- *menuCategory* -用来定义一个类别.
- *orderInCategory* – 用来定义次序，与一个组在一起.
- *title* – 标题.
- *titleCondensed* – 一个压缩的字符串标题, 当原标题太长的时候使用
- *icon* – icon 图标.
- *alphabeticShortcut* – 希腊字母的快捷方式
- *numericShortcut* – 一些数学符号的快捷方式
- *checkable* – 是否可编辑. *true* or *false*.
- *checked* – 是否被检验. *true* or *false*.
- *visible* –是否可见 *true* or *false*.
- *enabled* –是否可用. *true* or *false*.

更多讨论参见 Creating Menus。

2.8Layout 布局

Android 是用 XML 解析屏幕的，这和我们设计网页时用 HTML 类似。每一个项目都包含了整个大屏幕还有屏幕中的小部分，他们都会被编译进一个视图的资源文件，或者作为布局的元素被使用。文件被存放在 res/layout/目录中，会被 Android 的资源编译器编译。
每一个布局文件是一个单个的文件。首先我们要了 Android 能识别的标准的 xml 标签是什么样的，这样才能继续下一步的工作，然后会给出一些信息，告诉你如何在用户界面定义 XML 元素，使之控制客户端的视图布局。
根元素必须包含此命名空间"http://schemas.android.com/apk/res/android" 的声明在根元素中
更多讨论详见 User Interface
资源文件格式:需要定义一下 xml 文件的标准<?xml version="1.0" encoding="utf-8"?>，这里第一个参数定义的是 xml 的版本，第二个则是使用何种方式编码。
路径 res/layout/*some_file*.xml.
编译后的资源 指向 View 及其子集。
与资源文件相关的名称

- **Java:** R.layout.*some_file*
- **XML:** @[*package*:]layout/*some_file*

Syntax

```
<ViewGroupClass xmlns:android="http://schemas.android.com/apk/res/android"
                id="@+id/string_name" (attributes)>
  <widget or other nested ViewGroupClass>+
```

```
   <requestFocus/>(0 or 1 per layout file, assigned to any element)
</ViewGroupClass>
<ViewGroupClass>
```

这个文件必须有一个单独的根元素。它可以是一个 ViewGroup 类，包含其他元素，或者是一个 widget(不是 appwidget)。如果他只有一个单独的文件。默认情况下，我们可以使用任何 Android 的 Widget 或者是是 ViewGroup 类的名字来作为这个元素的名字。这些元素支持的那些类中基础的属性，但是命名却并不是那么明了，如何描述这些被支持的属性将在下面讨论。这里不需假设，标签之间混乱的嵌套会造成难以辨别的困难(举个例子，你不能将<TextView> 放进<ListLayout>)

如果一个类派生于其他的类，相应的 XML 文件也会继承到这个类继承的那个类的所有元素。所以，这里举个例子，<EditText> 这个 XML 元素是与 EditText 对应的，那么他也会暴露他所有的特殊属性(例如 EditText_numeric) 同时所有的属性都会被 <TextView> 和 <View> 。对于 XML 标签内的属性 id 必须使用加上前缀,例如这样”@+id/somestringvalue”。这个”@+”在 R 类中创建了相应的内容，除非是一个不存在,或没有用这个元素。当在XML标签中声明了一个id后,使用前缀,例如<TextView id="@+id/nameTextbox"/>,还有Java代码中的: findViewById(R.id.nameTextbox) 所有的元素都支持一下的值：

一个 ID 可以让 Java 来使用元素，通常他们都必须要使用@+的前缀生成一个 ID 在 XML 文件中，除非你不自己创建它。

- xmlns:android="http://schemas.android.com/apk/res/android" 这个命名空间我们需要加入进去.

<requestFocus>

任何表现在视图上的元素包括哪些空元素，都会在屏幕上给出他们的初始状态。每个文件中你只可能有一个这样的元素。

哪些属性是被元素支持的?

Android 使用 LayoutInflater 类在运行的时候装载布局的资源文件，并把他们转换为可视的。默认情况下,所有的组件都可以直接被标签支持,除非这是一个大的标签的列表,或是 R.styleable 涉及的页面中包含的属性。然而，这些属性都不是很重要。如果名称中出现了下划线，这表明包含一个属性——一般属性在下划线之后。所以，举个例子，EditText_autoText 意味着 <EditText> 标签支持 *autoText*.属性 当你正要在一个元素中使用这些属性的时候， 只能使用下划线前的一部分和前缀"android:"中的属性，举个例子，如果 R.styleable 列出了一下的属性

- TextView
- TextView_lines
- TextView_maxlines

你可以创建一个类似这样的元素

```
<TextView android:lines="10" android:maxlines="20"/>
```

他们会创建一个 Textview 并且给它加 line 和 maxline 这样的属性。

属性通常来自三个地方:

- **属性来源与元素本身.** 比如 TextView 支持 TextView_text,就像上面讨论的一样.
- **属性来源他们的元素的朝里.** 比如 TextView 类继承了 View 类, 所以 <TextView> 支持<View> 元素的所有的属性，包括 View_paddingBottom 和 View_scrollbars.

这两个除外 <TextView android:paddingBottom="20" android:scrollbars="horizontal" />.

-
- **ViewGroup.LayoutParams 子集的属性.** 所有 View 支持 LayoutParams 的成员(参见 Declaring Layout).为了建立一个 LayoutParams 的成员,属性就需要使用"android:layout_*layoutParamsProperty*"这样的方式. 例如: android:layout_gravity 就是指一个被 <LinearLayout>l 包裹的元素。请记住他们每一个子类，都支持继承的属性，在 *someLayoutParamsSubclass*_Layout_layout_*someproperty*. 的格式中，属性被暴露给他们的每个子类。这里有一个例子，是 Android 文件列出来的 LinearLayout.LayoutParams 类的道具。
-
- LinearLayout_Layout // The actual object — not used.
- LinearLayout_Layout_layout_gravity // Exposes a gravity attribute
- LinearLayout_Layout_layout_height // Exposes a height attribute
- LinearLayout_Layout_layout_weight // Exposes a weight attribute
- LinearLayout_Layout_layout_width // Exposes a width attribute

这个例子阐释了一些对象，包括直接的属性，继承的属性，以及布局元素自身的属性

```
<?xml version="1.0" encoding="utf-8"?>
<!-- res/main_screen.xml -->
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"    // The object's own
orientation property
               android:padding="4"                // Inherited View
property
              android:gravity="center"          // The object's own
property
                 android:layout_width="fill_parent"     //    Parent
object's LinearLayout.LayoutParams.width
                android:layout_height="fill_parent">    //    Parent
object's LinearLayout.LayoutParams.height

        <TextView      android:layout_width="fill_parent"         //
TextView.LayoutParams.width
                    android:layout_height="wrap_content"         //
TextView.LayoutParams.height
                    android:layout_weight="0"                    //
TextView.LayoutParams.weight
                    android:paddingBottom="4"                    //
TextView.paddingBottom
                  android:text="@string/redirect_getter"/>       //
TextView.text

   <EditText id="@+id/text"
```

```
                    android:layout_width="fill_parent"          //
EditText.LayoutParams.width
                    android:layout_height="wrap_content"        //
EditText.LayoutParams.height
                    android:layout_weight="0"                   //
EditText.LinearLayoutParams.weight
                    android:paddingBottom="4">                  //
EditText.paddingBottom
      <requestFocus />
  </EditText>

  <Button id="@+id/apply"
                    android:layout_width="wrap_content"         //
Button.LayoutParams.width
                    android:layout_height="wrap_content"        //
Button.LayoutParams.height
          android:text="@string/apply" />      // TextView.text
</LinearLayout>
```

代码使用

最常见的是装载 XML，我们要在整个屏幕使用它，如下

```
setContentView(R.layout.main_screen);
```

然后，布局元素也可以被木板来表现出

更多资料参见 User Interface

2.9 用户的布局资源

你可以定义用户层面的元素来使用 layout resources.这些用户元素可以同样被 Android 的布局元素来使用，你可以在其他的资源中，使用它们并且解析他们. SDK 中的 samples 应用有一个 ApiDemos 例子，阐述了如何来创建用户层的 layout XML 和 LabelView.创建用户元素，你须要使用下面的文件：

- **Java implementation file** – 这是一个编译文件. 这个类必须继承 View 或其子类 subclass. 详见 ApiDemos 的 LabelView.java。
- **res/values/attrs.xml** –定义了 XML 中的元素，及其支持的属性，客户端可以使用你的他们布局 XML 文件中实例好的对象. 这是格式
- <declare-styleable id=*your_java_class_name*>. 具体内容详见 ApiDemo 中的 res/layout/attrs.xml。
- **res/layout/*your_class*.xml** [可选] - 一个可选的 XML，用来描述你的对象的布局。他也可以在 java 中被创建. 详见 ApiDemos 中的 See custom_view_1.xml 。

资源文件格式: 不需要声明 <?xml> ，一个<resources>跟元素包含一个或者多个用户层面的标签。

路径: res/values/*attrs*.xml (文件名随意).

编译后的资源文件: 指向 View (或其子类) resource.

其他相关的资源名称: R.styleable.*some_file* (Java).

2.10 Styles and Themes 风格和主题

style 可以为元素提供一个或者多个属性(比如，一个文本框中 10px 大小的 Arial 字体)，style 是以 XML 形式来提供元素的属性。
Theme 可以为整个屏幕提供一个或者多个属性——这样说吧，Theme 为整个屏幕的显示效果提供了大量的属性。一个为 Activity 提供属性的 Themes 应该在 Manifest 里面注册。
两者都是在<stlye>里存储一个或者多个的数值，或者是相关的资源(比如说图片等等)。而且还支持继承，所以你能用这样来使用 Theme，MyBaseTheme，MyBaseTheme.Fancy，MyBaseTheme.Small。
更多讨论参见 Applying Styles and Themes。
资源文件的格式”资源文件中，需要定义一下 xml 文件的标准 <?xml version=”1.0” encoding=”utf-8”?> ，这里第一个参数定义的是 xml 的版本，第二个则是使用何种方式编码。颜色等标签都应该放到<style>这个标签下。
路径 res/values/*colors*.xml (文件名随意)
编译后的资源 指向 Java int。
其他相关:

- **Java:** R.style.*styleID* 整体的风格

R.style.*styleID.itemID* 单个元素

- **XML:** @[*package*:]style/*styleID* 整体风格，

@[*package*:]style/*styleID*/*itemID* 单个元素

Note: to refer to a value in the *currently* applied theme, use “?” instead of “@” as described below (XML).
注意:如果要在当前的主题中提交元素，需要用”?”来代替”@” 。
Syntax

```
<style name=string [parent=string] >
   <item name=string>Hex value | string value | reference</item>+
</style>
<style>
```

用一个或者多个<item>，描述每一个属性。

- *name* – 名称
- *parent* –这里可以选择父 theme。所有细节方面的风格设置，都可以从这个 theme 中继承。你的 app 需要去识别的属性都会被重载集成。名称需要被包限制，但是如果你的这些风格设置就在一个包里，那就不用那么严格了(比如，你的 theme 是 android:Theme 这样的系统 theme，或者一个 theme 已经在你的包中定义好了)。

<item>

被主题使用的资源很丰富，他可以是一个标准的字符串，十六进制的颜色信息，或者是其他相关的可以被使用的任何资源类型。

更多讨论参见 Applying Styles and Themes

3.Resources and Internationalization（资源和国际化）

By xiangyong

资源是外部文件（即非代码文件），它在代码中被使用，是在编译的时候加载到应用程序的。Android 支持很多种不同类型的资源文件，包括 XML，PNG 和 JPEG 文件。XML 文件根据描述的不同格式也不相同。这份文档说明了 Android 可以支持什么样类型的资源文件，以及语法或格式。

资源是外在的源代码，XML 文件被编译成二进制文件，这种格式能使资源加载的速度加快。字符串同样也被压缩成更高效的存储形式。正因为如此，Android 平台才有这些不同的资源文件类型。

本文和下一节的资源引用都是技术相当密集的文档，他们覆盖了大量和资源有关的信息。如果是使用 Android 平台，不需要很深入的了解本篇文章，当你需要时，知道在这里可以查到你要的信息就可以了。

3.1 介绍（introduction）

本主题包含了一个和资源有关的术语列表，以及一些在代码中怎么使用资源的例子。如果想看一个完整的资源类型的指南，请看下一节的资源引用。

Android 资源系统能通过应用程序跟踪所有的非代码的资源。你可以使用 Resources 这个类访问你的应用程序的资源；资源的实例和应用程序联系在一起，你通常可以通过 Context.getResources()来获取到资源。

应用程序的资源在编译时被编译系统编译到应用程序的二进制文件里。为了使用资源，你必须将它放在正确的资源树里，然后编译。作为编译的一部分，每个资源文件产生的代号你可以在你的代码中使用，这允许编译器验证你的程序代码和你定义的资源是否一致。

下面的章节组织成一个在应用程序如何使用资源的教材。

创建资源（Creating Resources）

Android 支持字符串，位图和其他很多的资源类型。对象的类型决定了语法，格式和存储位置。通常，你通过三种类型的文件来创建资源：XML 文件（除位图和原始数据），位图文件（对应图片）和原始数据（其他类型，例如声音文件）。事实上，有两种不同类型的 XML 文件，一种是编译包里的，另外一种是通过 aapt 产生的资源文件。下表列出了所有资源的类型，文件的格式，文件的描述和所有 XML 文件的详细信息。

你可以在项目的 res/目录下创建和存储资源文件。Android 有一个资源编译器（aapt），它会编译这个目录下的所有子目录中的资源。这里有个各种资源的列表，你可以在资源引用小节中看到各种类型的对象，包含格式或者语法。

目录	资源类型
res/anim/	XML 文件，被编译成逐帧动画或者补间动画的对象
res/dreamable/	.png,.9.png 和.jpg 文件被编译成图表资源列表 为了获取图像类型的资源文件，使用 Resource.getDrawable(id) • 位图文件 • 9-patches(可调整大小的位图)

res/layout/	可被编译成屏幕布局（部分布局）的 XML 文件，查看布局的宣告
res/values/	可被编译成多种类型的资源的 XML 文件。 注意：不像其他的 res/文件夹，它可以容纳任何数量的文件，但只是描述其创建而不是资源本身。XML 元素的类型决定了这些资源在 R 类的什么位置被替换了。 文件可以被命名为任何名字，这个文件夹有一些经典的文件（一般约定以文件中定义的元素名称来给文件命名）： • Arrays.xml 定义数组 • Colors.xml 定义颜色和颜色字符串值。分别用 Resources.getDrawable()和 Resources.getColor()获取这些资源 • Dimens.xml 定义尺寸数据，用 Resources.getDimension()取得这些资源 • Strings.xml 定义字符串值（用 Resources.getString 或者 Resources.getText()来获取这些资源。getText()将保留任何丰富的通常在 UI 中描述的文字样式。） • Styles.xml 定义样式对象。
res/xml	任何 XML 文件，在运行时可通过 Resources.getXML 被编译和读取
res/raw	任何被直接拷贝到设备上的文件。在程序被编译时，它们直接加到压缩文件中。在应用程序中可以通过用 Resources.openRawResource(id)来获取资源，id 的形式为：R.raw.filename

资源被编译到最终的 APK 文件里。Android 创建了一个包装——R，这样你在代码中可以通过它关联到对应的资源文件。R 包含的子类的命名由资源的路径和文件的名称决定。

3.2 全局资源的声明（Global Resource Notes）

- 一些资源允许你定义颜色。Android 能接受多种网络类型的颜色值——任何一种一下的十六进制：#RGB,#ARGB,#RRGGBB,#AARRGGBB
- 所有的颜色都可以设置阿尔法值，开始的两个十六进制的数字指定透明度。0 阿尔法值意味着透明，默认值是不透明的。

3.3 使用资源（Using Resource）

这一小节描述如何使用你创建的资源。包含如下的内容：

- 在代码中使用资源——在代码中如何调用资源并实例化它们。
- 从其他资源里引用资源——可以从资源里引用其他资源，这样可以重复使用公共资源
- 支持替代资源和替代配置——根据语言和主机的配置，可以指定加载不同的资源

在编译时，android 会产生一个 R 类，它包含项目中所有的资源的资源标识符。这个类包含一些子类，这些子类对应的资源都是 android 的支持的，并且是你提供的资源文件。每个类提供一个或多个编译后的资源的标志符，你可以在代码中使用他们。下面这个小的资源文件包含字符串，布局（屏幕或者屏幕的部分）和图像资源。

注意：R 类是一个自动生成的文件不能手动更改，当资源有变更时它会自动更新。

> **(编者提示：如果你发现你的 R 文件中没能包含你列出的资源文件，那就说明你的资源文件设置有问题，当出问题的时，R 是不会自动更新的。另外在 res 下不支持二级目录，只能用官方定义的那几个目录，多的目录都会出错。)**

```
package com.android.samples;
public final class R {
    public static final class string {
        public static final int greeting=0x0204000e;
        public static final int start_button_text=0x02040001;
        public static final int submit_button_text=0x02040008;
        public static final int main_screen_title=0x0204000a;
    };
    public static final class layout {
        public static final int start_screen=0x02070000;
        public static final int new_user_pane=0x02070001;
        public static final int select_user_list=0x02070002;
    };
    public static final class drawable {
        public static final int company_logo=0x02020005;
        public static final int smiling_cat=0x02020006;
        public static final int yellow_fade_background=0x02020007;
        public static final int stretch_button_1=0x02020008;
    };
};
```

在代码中使用资源（Using Resources in Code）

只要知道资源的 ID 和你编译的资源文件的类型就可以在代码中使用资源了。下面是资源引用的语法：

R.*resource_type.resource_name*

或者

android.R.*resource_type.resource_name*

其中 *resource_type* 是 R 子类的一种类型，它对应一种特定的资源；

resource_name 是 XML 文件或者其他文件的名字属性（不含后缀）。每种资源文件对应 R 类中特定的子类，子类的类型取决于资源的类型；为了学习哪种 R 子类索引你编译的资源类型，参考下小节的资源引用文档。被你的应用程序编译的资源文件可以不通过包名引用（如：R.resource_type.resource_name）。Android 包含许多标准的资源，例如屏幕的风格和按钮的背景。如果要想在代码中指向这些资源，必须引入 android，如 android.R.drawable.button_background。

下面是一些好的或者糟糕的在代码中使用资源的例子：

```
// Load a background for the current screen from a drawable
resource.
this.getWindow().setBackgroundDrawableResource(R.drawable.my_backgr
```

```
ound_image);

// WRONG Sending a string resource reference into a
// method that expects a string.
this.getWindow().setTitle(R.string.main_title);

// RIGHT Need to get the title from the Resources wrapper.
this.getWindow().setTitle(Resources.getText(R.string.main_title));

// Load a custom layout for the current screen.
setContentView(R.layout.main_screen);

// Set a slide in animation for a ViewFlipper object.
mFlipper.setInAnimation(AnimationUtils.loadAnimation(this,
        R.anim.hyperspace_in));

// Set the text on a TextView object.
TextView msgTextView = (TextView)findViewByID(R.id.msg);
msgTextView.setText(R.string.hello_message);
```

3.5 引用资源（References to Resources）

一个在属性（或者资源）里的值也可以指向一个资源。这常常被使用在布局文件中的字符串（可以被本地化）和图片（存在于其他文件中的），通过引用可以是任何的资源类型，包括颜色和整数。

例如，如果有颜色资源，我们可以写一个布局文件，它的文本的颜色值包含在一个资源文件中：

```
<?xml version="1.0" encoding="utf-8"?>
<EditText id="text"
    xmlns:android="http://schemas.android.com/apk/res/android"
                                    android:layout_width="fill_parent"
android:layout_height="fill_parent"
    android:textColor="@color/opaque_red"
    android:text="Hello, World!" />
```

注意：这里使用'@'前缀说明引用资源——后面的文本是资源的名字，格式为@[package:]type/name。在这种情况下，我们不需要指定特定的包名，因为我们指向的引用资源在我们自己的包中。如果是指向系统的资源，你必须写包名：

```
<?xml version="1.0" encoding="utf-8"?>
<EditText id="text"
    xmlns:android="http://schemas.android.com/apk/res/android"
                                    android:layout_width="fill_parent"
android:layout_height="fill_parent"
    android:textColor="@android:color/opaque_red"
    android:text="Hello, World!" />
```

另外一个例子，当在布局文件中提供字符串时，你应该经常使用资源引用，这样可以进行本地化：

```
<?xml version="1.0" encoding="utf-8"?>
<EditText id="text"
    xmlns:android="http://schemas.android.com/apk/res/android"
                                    android:layout_width="fill_parent"
android:layout_height="fill_parent"
    android:textColor="@android:color/opaque_red"
    android:text="@string/hello_world" />
```

这段代码也可以被用来创建资源间的引用。例如，我们可以创建新的图像资源，它指向已经存在的图像资源：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
                                                        <drawable
id="my_background">@android:drawable/theme2_background</drawable>
</resources>
```

3.6 引用主题属性（References to Theme Attributes）

另外一种资源值允许你引用当前的主题属性。这种属性引用只能被使用在风格资源和 XML 属性中；通过改变当前标准的主题，你可以自定义 UI 元素，而不需要提供大量的值。

在下面的例子中，我们通过使用主题将布局中中文本的颜色设置成定义在基本的系统中的一种：

```
<?xml version="1.0" encoding="utf-8"?>
<EditText id="text"
    xmlns:android="http://schemas.android.com/apk/res/android"
                                    android:layout_width="fill_parent"
android:layout_height="fill_parent"
    android:textColor="?android:textDisabledColor"
    android:text="@string/hello_world" />
```

注意：这和资源引用和相识，唯一不同的是使用'?'代替'@'前缀了。当你使用'?'标记名字属性时，将使用系统主题属性——因为资源工具知道一个属性资源被引用，你不需要明确说明类型（类似?android:attr/android:text DisabledColor）。

使用资源标识符到主题去寻找相应的数据而不是使用原始数据，其语法和'@'类似：?[namespace:]type/name，这里的 type 是可选的。

3.7 使用系统资源（Using System Resources）

许多系统的资源应用程序可以使用。这些资源被定义在 android.R 类下面。例如，你可以在屏幕上显示标准的应用程序图标，通过使用下面的代码：

```
public class MyActivity extends Activity
{
    public void onStart()
```

```
    {
        requestScreenFeatures(FEATURE_BADGE_IMAGE);

        super.onStart();

        setBadgeResource(android.R.drawable.sym_def_app_icon);
    }
}
```

相似的方法，下面的代码将使你的屏幕变成系统定义的标准的"绿色背景"：

```
public class MyActivity extends Activity
{
    public void onStart()
    {
        super.onStart();

        setTheme(android.R.style.Theme_Black);
    }
}
```

3.8 替代资源（替代语言和配置）Alernate Resouces (for alternate languages and configurations)

你可以为你的产品提供不同的资源，根据 UI 界面语言或者设备的硬件配置。注意，你可以包含不同的字符，布局和其他的资源，SDK 不提供显示的方法让你指定加载的资源。Android 通过检测硬件和语言的设置，去加载特定的资源文件。用户可以在设定面板上设定语言的种类。
为了包含替代资源，在同一目录下创建并行的文件夹，在每个文件夹名字后面加上相应的指定，说明它使用的配置（如语言，屏幕的方面等等）。例如，下面是一个工程中包含一个字符串资源，一个是英文版的，一个是法文版的：

```
MyApp/
    res/
        values-en/
            strings.xml
        values-fr/
            strings.xml
```

Android 支持很多种不同类型的修饰，把修饰加在资源文件夹名字的后面，通过破折号隔开。并可以多个修饰加在文件夹的后面，但是它们必须按它们出现的顺序。例如，一个文件夹包含图像资源并且完全指定配置，如下所示：

```
MyApp/
    res/
        drawable-en-rUS-port-160dpi-finger-keysexposed-qwerty-dpad-480x320/
```

更典型的是，你只会指定资源的一些特定的配置项。你可能会从整个属性列表中删除任何的一些属性值，只要剩下的属性值保留正确的顺序就可以：

```
MyApp/
    res/
```

```
drawable-en-rUS-finger/
drawable-port/
drawable-port-160dpi/
drawable-qwerty/
```

修饰语	值
语言	两个小写字母参照 ISO 639-1 标准。例如：en，fr，es
地区	一个小写字母加两个大写字母，参照 ISO 3166-1-alpha-2 标准。例如：rUS，rFR，rES
屏幕方向	Port(竖屏)，land(横屏)，square
屏幕像素	92dpi，108dpi，等等
触摸屏类型	Notouch(不支持触屏)，stylus(手写笔)，finger(手指触摸)
键盘是否可用	Keysexposed(可用)，keyshidden(不可用)
主要文本输入模式	Nokeys(无键盘)，qwerty(标准传统键盘)，12key(12 个键)
主要的无触摸的导航模式	Nonav，dpad，trackball，wheel
屏幕分辨率	320×240，640×480，等等。大分辨率需要开始声明。

上面的这个列表不包含设备的特殊参数，如载体，商标，设备/硬件，制造商。任何应用程序需要知道的设备信息都在上面表格中的资源修饰语里。

下面是一些通用的关于资源目录的命名指导：

- 各个变量用破折号分开（每个基本的目录名后跟一个破折号）
- 变量是区分大小写的（所有变量名的大小写必须保持一致）

  例如，
- 一个 drawable 的目录必须命名为 drawable-port，而不是 drawable-PORT
- 你不能有两个目录命名为 drawable-port 和 drawable-PORT，即使故意将“port”和“PORT”指向不同的参数值。
- 同一类型的修饰语在一个文件名中只能出现一次（你不能这样命名 drawable-rEN-rFR/）
- 你可以指定多个参数来定义具体的配置，但参数必须保持上面表格中的顺序。例如，drawable-en-rUS-land 指在 US-English 的设备上应用全景。
- Android 会试图找到最适合当前配置的目录，关于这一点将在下面详细讲述
- 表格里所有的参数的顺序是用来防止多个合格目录的事件发生（可看下面的例子）
- 所有的目录，无论是否合格，都是放在 res/目录下的。合格的目录是不能嵌套的（你不能这样写 res/drawable/drawable-en）
- 所以的资源将在代码或简单的资源引用语法，不加修饰的名字。因此，如果一个资源被命名为：

  MyApp/res/drawable-port-92dp/myimage.png

  那么它可以这样被引用：

  R.drawable.myimage（在代码中）

  @drawable/myimage (在 XML 文件中)

3.9Android 如何找到最佳目录（How Android finds the best mathching directory）

Android 将挑选哪些资源文件在运行时会被使用，这取决于当前的配置。选择过程如下：

1. 清除所有的不符合当前设备配置的资源。例如，如果当前屏幕的像素是 108dpi，那么将清除 MyApp/res/drawable-port-92dpi/。

```
MyApp/res/drawable/myimage.png
MyApp/res/drawable-en/myimage.png
MyApp/res/drawable-port/myimage.png
MyApp/res/drawable-port-92dpi/myimage.png
```

2. 挑选出和配置最多匹配的资源文件。例如，如果地球是 en-GB，屏幕方向是 port，那么有两个符合配置的选项：MyApp/res/drawable-en/和 MyApp/res/drawable-port/。目录 MyApp/res/drawable/将被清除，因为它和当前的配置有 0 个属性相同，而其他两个文件有一个和配置文件相同的属性。

```
MyApp/res/drawable/myimage.png
MyApp/res/drawable-en/myimage.png
MyApp/res/drawable-port/myimage.png
```

3. 根据配置的优先级选取最终的资源文件，上面表格的修饰语就是按照优先级来写的。也就是说，语言的优先级比屏幕方向的优先级高，所以，我们通过语言来选取配置文件 MyApp/res/drawable-en/。

```
MyApp/res/drawable-en/myimage.png
MyApp/res/drawable-port/myimage.png
```

3.10 术语（Terminology）

资源系统将一些不同的分散的集合在一起从而形成最终的完整的资源功能。为了帮助我们了解整个系统，下面有一些你在使用中会遇到的简单的关于核心概念和部件的定义：

Asset: 应用程序的独立的数据包 ,这包含所有从 java 程序编译成的目标文件 ,图像 (例如 PNG 图片)， XML 文件等等。这些文件以一种特定的方式组织在一起，在程序打包最后时，它们被打包成 一个独立的 ZIP 文件。

aapt: Android 最终文件打包工具。这个工具产生最终程序的 ZIP 文件。除了将最终的元数据文件打包在一起，它也解析资源定义到最终的二进制数据里。

资源表（Resource Table）： aapt 工具产生的特殊的文件，描述了所有在程序/包里的资源。这个文件可以通过 资源类来访问；它不能直接和应用程序接触。

资源（Resource）： aapt 工具产生的特殊的文件，描述了所有在程序/包里的资源。这个文件可以通过 资源类来访问；它不能直接和应用程序接触。

资源标识符（Resource Identifier）： 在资源表里所有的资源都被唯一的整数标识着。 所有的代码中（资源描述，XML 文件，Java 源代码）你可以直接使用符号名代替真实的整数数值。

基本资源（Primitive Resource）： 所有基本资源都可以被写成一个简单的字串，使用一定的格式可以描述资源 系统里各种不同的基本类型： 整数，颜色，字串，其他资源的引用，等等。像图片以及 XML 描述文件这些复杂资源， 被以基本字串资源储存，它们的值就是相关最终数据文件的路径。

包装资源（Bag Resource）： 有一种特殊类型的资源，不是简单的字符串，而是有一个随意的名字/数值配对列表。 每个数值可以对应它本身的资源标识，每个值可以持相同类型的字符串格式的数据作为一个正常的资源。包装资源支持 继承：一个包里的数据能从其他包里继承，有选择地替换或者扩展能产生你自己需要的内容。

种类（Kind）： 资源种类是对于不同需求的资源标识符而言的。例如，绘制资源类常常实例化绘制类的对象，所以这些包含颜色以及指向图片或 XML 文件的字符串路径数据是原始数据。其它常见资源类型是字符串（本地化字符串），颜色（基本颜色），布局（一个指向 XML 文件的字串路径，它描述的是一个用户界面）以及风格（一个描述用户接口属性的包装资源）。还有一个标准的"attr"资源类型，它定义了命名包装数据以及 XML 属性的资源标识符。

风格（Style）： 包含包装资源类型的名字常常用来描述一系列用户接口属性。例如，一个 TextView 的类可能会有一个描述界面风格的类来定义文本大小，颜色以及对齐方式。在一个界面布局的 XML 文件中，可以使用"风格" 属性来确定整体界面风格，它的值就是风格资源的名字。

风格类（Style Class）： 这里将详述一些属性资源类。其实数据不会被放在资源表本身，通常在源代码里它以常量的形式出现，这也可以使你在风格类或者 XML 的标签属性里方便找到它的值。例如，Android 平台里定义了一个"视图"的风格类，它包含所有标准视图的属性：画图区域，可视区域，背景等。这个视图被使用时，它就会借助风格类去从 XML 文件取得数据并将其载入到实例中。

配置（Configuration）： 对许多特殊的资源标识符，根据当前的配置，可以有多种不同的值。配置包括地区（语言和国家），屏幕方向，屏幕分辨率，等等。当前的配置用来选择当资源表载入 时哪个资源值生效。

主题（Theme）：一个标准类型的资源能为一个特殊的上下文提供全局的属性值。 例如，当应用工程师写一个活动时，他能选择一个标准的主题去使用，白色的或者黑色的；这个类型 能提供很多信息，如屏幕背景图片/颜色，默认文本颜色，按钮类型，文本编辑框类型，文本大小，等。 当布置一个资源布局时，控件（文本颜色，选中后颜色，背景）的大部分设置值取自当前主题； 如果需要，布局中的风格以及属性也可以从主题的属性中获得。

覆盖层（Overlay）： 资源表不能定义新类型的资源，但是你可以在其他表里替换资源 值。就像配置值，这可以在装载时候进行；它能加入新的配置值（例如，改变字串到新的位置）， 替换现有值（例如，将标准的白色背景替换成"Hello Kitty"的背景图片），修改资源包（例如修改主题 的字体大小。白色主题字体大小为 18pt）。这实际上允许用户选择设备不同的外表，或者下载新的外表 文件。

资源引用（Resource Reference）

资源引用这份文档提供了各类资源的详细列表，并提供了如何在 Java 代码中使用资源和资源中引用资源。

国际化和本地化（Internationalization and Localization）

即将推出：国际化和本地化是非常重要的，但是在当前的 SDK 中还没有完成。随着 SDK 的成熟，将来本节将会包含国际化和本地化的信息。同时，从外部和在创建和使用资源时实践良好的结构将会更理想。

4.本地化你的 Android 应用程序

[英文版还是草稿，这篇暂以草稿来翻译](1.5 的正式版都快出了，可是这篇文章依然还是草稿)

Android 可以运行在不同国家的各种硬件上，并别有很多语言不同的用户在使用 Android。为了能够用最好的方法接触到我们的用户，开发者必须考虑让他们的应用程序使用用户本地化的语言，这样才是明智的。这篇文件就解释了如何来本地化一个 Android 应用程序。

在此之前你需要知道什么

这里是介绍一下如何来建立一个 Android 应用程序(译者注:相比关注 eoe 特刊许久的朋友，对这个应该很熟悉的)；如果想了解更多的信息，你可以到这个访问 eoeAndroid 社区，这篇文稿简单介绍了一下 Android 资源的装载、通过 Android 的 XML 系统来派生一个 UI 界面，以及简单介绍一下 Activity 的生命周期。

4.1 介绍一下 L10nDemo 这个应用程序

为了说明这个讨论的核心内容，我们需要创造一个 L10nDemo 应用程序。这个应用程序非常简单，并且特别系统地讲解了 Android 本地化的一些关键要素。

这个应用程序是有一个简单的 UI 界面以及一个 ImageButton 组成，页面上有三个 TextViews。这 3 个 Textviews 会根据 Android 的本地化后的系统有所改变，而按钮上的图片，则会相应的变成这个国家的国旗。点击的时候，button 按钮将会调出一个对话窗口，显示一些额外的信息 。UI 界面有两种:第一种竖屏幕，第二种横屏。每种模式下，显示的内容都是一样的，但是在结构组织上会有一点微小的差别。

这里有 L10nDemo 的源码，这个是运行在 1.0_r2 SDK 的 emulator 中的，下载代码请点击这个链接 https://android-developers.googlegroups.com/web/L10nDemo.zip?gda=VkhgDT4AAABbwPBSOzVTOhnH8QAAJRGkc_eKk8aavG57jinPV4YF8hrrnv28OuiyeMcIjswJCkvjsKXVs-X7bdXZc5buSfmx&gsc=gg7xFCEAAAB2IECM3xdqRKAbz0TqF_32sQtvHZRPiJNW1GKaHu4ct0zfKN-m9S9niuHrq-IEXAE

4.2Android 本地化的原理

在我们深入到 L10nDemo 的细节前，让我从一个 Android 如何进行本地化的概述开始。

Android，可以允许开发人员将他们的 UI 分解。大体上来说，Android 将一个应用程序的 UI 分成了 3 个大的核心部分:

1. The layout of a UI -- UI 的布局文件，比如说 Button 在哪儿，Label 标签还有数据放哪里，等等，Android 允许开发者通过 XML 文件来解析这些 UI 布局文件。
2. 被装进 UI 的资源，比如图像和声音。

3. Java 写的应用程序的代码，他们可以驱使 UI 不断发生变化，并且让用户在进行输入操作的时候使屏幕发生变化。

开发者需要利用一下这些资源；在这个实例，开发者可以简单的将他们的图片文件放到'raw/'这个文件夹中，并且从 Java 的资源代码中程序化地构建用户的界面。然而，这些设计对于我们要"国际化"的目的来说，很贫弱(意思就是你很难用使用多种语言).但是 Android 的系统，则可以很容易地进行本地化，所以我们强烈建议我们的开发人员都来使用这个系统。

想必你也很清楚，Android 可以直接装载 XML 为形式的布局文件，可以使用在项目文件夹中的'res/'目录下的资源。然后，这里还需要提醒一下，Android 还可以自己选择到底要装载那些资源，而这个选择则是依据用户当前的设置以及他的地区。
在这个实例中，你的应用程序代码也许会涉及到'R.string.title' 这样一段字符串。Android 将会在运行的时候在装载'res/values'中相匹配的字符文件。示例中，如果你的设置的语言和地区是'en-US', 然后 Android will 会寻找一段适合当前语言的 R.string.title 值 如果在你的 res/中只有一个 Value 文件，那么 Android 就会直接找到默认的 strings.xml。但是如果在你的 res 文件中出现了 value-en-US 这个文件夹，那么 Android 就会在这个里面找到对应的 strings.xml。Android 首先会在这个文件夹中找文件，知道最后找不到了再返回给默认的。(注意，无论是什么语言'res/values/strings.xml'中的就是默认的语言，很多开发者使用英语，所以也就不需要在设置一个 vale-en 文件夹了)
除了本地化，Android 还会根据不同的设备设置还有状态，以及他们的屏幕状况来选择不同的资源文件。这里，我们在一个 T-Moblie G1 来展示一下屏幕的开启以及键盘的出现。Android 会寻找想匹配的'R.layout.main'在 'res/layout-land/main.xml'和'res/layout/main.xml'中。你可以混合他们将他们搭配，比如这样 'res/values-fr-rCA-land/strings.xml' 这里支持了一个法国设备的横屏状态。当然有的时候，这会看起来很不自然，那就需要你来做一下这里的衔接工作，如果你在 UI 上下了很大功夫，这里就对你很有利，你可以自己为不同的屏幕状态设置不同的 labels 的长度。

更多资料详见
http://code.google.com/android/devel/resources-i18n.html#AlternateResources。
正如你看到的，资源文件升级时很正常且合理的，这让我们的用户可以是各种不同语言的使用者，剩下的部分，让我们聚集到到如何使用我们的矿建来建立一个本地化的应用程序。

4.3 L10nDemo 的资源。

下面是直接从 L10nDemo 项目文件中提取的文件列表，还有一个附带的解释。

- Layouts:
  res/layout-land -- 横屏模式
  res/layout -- 默认模式，当没有任何相匹配的模式时使用这个(提示，默认情况下是竖屏)
- Strings
  res/values-fr -- 法语(加拿大和法国)
  res/values-de – 德语(德国和奥地利)

res/values-en-rUS --美式英语
res/values --英国英语，这里作为默认的

- Drawables (i.e. national flag images):图片资源，这里就是我们的国旗资源了
  res/drawable-de-rDE -- used when the current locale is Germany 德国
  res/drawable-fr-rCA -- used when the current locale is French-Canadian (fr-CA)法语-加拿大
  res/drawable-fr-rFR -- used when the current locale is France (fr-FR)法国
  res/drawable-en-rCA -- used when the current locale is English Canadian (en-CA)英语-加拿大
  res/drawable-en-rAU -- used when the current locale is Australia 澳大利亚
  res/drawable-en-rUS -- used when the current locale is the United States 美国
  res/drawable -- used when no other more specific locale is matched. In this case, the default content is in English, and consists of the UK flag.没有任何其他的国家设定的时候，默认是英语，这里用的将是英国国旗。

最后，我们这里还有一个列表，让你准确了解一下，不同的地区模式下，使用的是哪些资源。

Language	**Country**	**Locale Code (/data/locale)**	**Strings**	**Layouts [1]**	**Drawables**
German	Germany	de-DE	res/values-de/strings.xml	res/layout (or res/layout-land)	res/drawable-de-rDE/flag.png
German	Austria	de-AT	res/values-de/strings.xml	res/layout (or res/layout-land)	res/drawable/flag.png [2]
French	France	fr-FR	res/values-fr/strings.xml	res/layout (or res/layout-land)	res/drawable-fr-rFR/flag.png
French	Canada	fr-CA	res/values-fr/strings.xml	res/layout (or res/layout-land)	res/drawable-fr-rCA/flag.png
English	Canada	en-CA	res/values/strings.xml [3]	res/layout (or res/layout-land)	res/drawable-en-rCA/flag.png
English	United Kingdom	en-GB	res/values/strings.xml [3]	res/layout (or res/layout-	res/drawable/flag.png

				land)	
English	Australia	en-AU	res/values/strings.xml [3]	res/layout (or res/layout-land)	res/drawable-en-rAU/flag.png
English	United States	en-US	res/values-en-rUS/strings.xml	res/layout (or res/layout-land)	res/drawable-en-rUS/flag.png

- [1] -因为这个 demo 是建立在不同的语言和不同的布局下的，同样的布局文件可以放到所有的语言中去。只有在设备发生变化的时候，他们才会发生变化。然而，如果你的应用程序在当前的测试阶段，实例化了例如日语和汉语，我们可能需要为这些语言特定一个不同的布局。所以你需要使用诸如'res/layout-jp', 'res/layout-land-jp', 和 'res/layout-port-jp'这样的布局。
  (译者注：本文卸载 1.0r2 的 SDK 中，而在 1.5 的 SDK 中已经对中文有了很好的支持，日语也是如此这里的担心大可不必)
- [2] -因为项目中并没有为澳大利亚匹配其国旗，系统将会回到默认的选项，放一张美国的国旗。在真实世界中，我们一定要避免这样的错误，这是政治问题。如果要从 'res/drawable-de-rDE' 转换到 'res/drawable-de'，这里就会有更多的属性了，因为此时是德语国家的德国和奥地利共有，这里要注意一下奥地利和德国。这里可能在奥地利的国旗上会出现错误，但是他会比默认的 UK 国旗好
- [3] - 加拿大、澳大利亚和应够都共享一个字符串，既然有这三个国家，他们都会被返回给默认值，美式英语有单独的匹配项(美式英语在具体的品写上有差别)。然而，我们在工作的时候，必须要考虑给 res/values-en/strings.xml 更好的”语言风格”。

4.4 Java 代码中的本地化技术

有些时候，可能需要你从代码中来获得资源。最显眼的例子就是在运行的时候填入模板。幸运的是，Android 本地化的架构也适合于通过代码重新获得资源。比如，下面这一小段代码来自于 L10nDemo，使用一个标准的 Java messageFormat，这里的功能使用了 Android 的本地化结构来转换这些模板：

```
String msg = getResources().getString(R.string.text_c); // contains {0}, {1}, and {2} in the XML file
msg = java.text.MessageFormat.format(msg, "foo", "bar", myStringVariable);
```

有些情况下，开发者可能希望能找到一些更加精确的资源。比如，一个项目中，我们可能会因为没有做好一个具体的本地化，而丢失了很多的准备好的资源，比如'raw/'。这里我们希望开发人员能够使用转而使用 Android 的自己的国际化架构，而不是手动地通过自己寻找。这里可以通过标注的 Java API 来完成，或者使用 Android Context。下面两句话的效力相当。

```
String locale = context.getResources().getConfiguration().locale.getDisplayName
();
String locale = java.util.Locale.getDefault().getDisplayName();
```

4.5 测试一下我们已经本地化的应用程序

现在这里需要的是我们的设备，所有船运过去的设备一开始都没有调整好他们的的本地换。这就意味着，在实际的系统上，改变系统的地区与语言将不是很轻松，因此我们也没有一个很好的方法来测试我们的一个用程序的本地话，除非我们用那些使用原版语言的设备。我们希望这里在将来能够轻松一下(1.5 版的 SDK 基本上已经实现了)。

幸运的是，这里已经可以再 emulator 上进行测试了；你可以很轻松的改变默认的语言，ibng 重新启动 emulator 中的系统(这里不是直接重启 Emulator)

1. 使用你想要选择的语言，假设我们这里要设置法语'fr'，国家加拿大'CA'。
2. 启动 Emulator，这里我们使用命令行，而不是 Eclipse
3. "adb shell" 命令行语句。
4. 我们使用这样的命令行语句，首先"setprop persist.sys.language [[language code]];setprop persist.sys.country [[country code]];stop;sleep 5;start" 执行完毕以后，继续开始第一步的步骤。

为了检验加拿大英语，使用这段命令行

```
"setprop persist.sys.language en;setprop persist.sys.country CA;stop;sleep 5;star
t"
```

这里会造成 emulator 的重新开始(和电脑上的重启类似，但又不完全相同)Home 显现以后，再次重新载入你的应用程序(简单点击即可)这个应用程序将会以新的马甲面试了。注意这里默认的系统预言是英语，所以选择一个非应用的程序，这里将不会被本地化，系统自身依然显示为英文。然而在这里需要注意，如果系统的地区与语言被改变了，那你的应用程序也会相应的本地化 。

4.6 发布你本地化好的应用程序.

大部分开发者都想要把自己的程序发布到 Android Market，并且能让程序适应用户的手机。发布过程简单，但是这里有需要额外考虑一下。这个部分将重点讲解这一部分，应该如何管理你的 apk 文件。

如果你的应用程序值需要做一些简单的国际化，你可以做一个 apk 文件包括这所有的国际化的内容。然而如果你要做的很极致，应用程序需要继承很多语言(比如你要继承 Google 支持的 40 中语言在你的产品中)，这样就会有多达 40 中不同的字符串、布局、资源文件。一个人是很难做到这点的，所以不要把时间浪费在这个上面，我们可以将他们分组。这里不会有对每一个开发者都好的解决方案；有一些改动小的语言可以放在一组，比如说，我们可以欧洲放一组、北美放一组，亚洲放一组等等。

如果你决定好了要割裂你的应用程序，你就需要重建，不行的是，在单枪的 SDK 工具中，'aapt -c'并不能多次执行。最简单的方法是在 Eclipse 创建一个临时目录。然后重新建立项目，的确很繁琐。最后再把移出的文件弄回去。

5.在 Android 中轻松实现横竖屏的布局

by
Iceskysl
http://iceskysl.1sters.com/

在 android 中是非常容易实现对横屏和竖屏的支持的,这得益于其灵活的布局模式，在 android 中，推荐采用 XML 文件的布局模式。正因为此，其对横屏和竖屏的支持非常简单，我们这里用一个小例子演示。

5.1 第一步：创建一个 android 工程

这步没啥好说的了，大家 5.1 都会，创建的工程名字为 layoutsDemo。创建后其项目的 res/layout 目录下存放的是针对竖屏的布局模板文件，大家可以看到，默认已经生成一个名字为 main.xml 的布局模板文件，看下其代码，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"
    />
</LinearLayout>
```

比较简单，就显示一个 TextView，其内容是@string/hello

5.2 第二步：新建一个 layout-land 文件夹

在 res 目录下新建一个名字为 layout-land 的文件夹，这个就是用来存放横屏的布局模板文件，为了个竖屏的想比较，我们也创建一个叫 main.xml 的布局模板文件，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_land"
    />
</LinearLayout>
```

看到其代码和 layout 目录下的 main.xml 几乎一致，只是为了区别这个是横屏的，我让其显示的文字是@string/hello_land。

5.3 第三步：编写 string.xml

将上面两个模板中用到的 string，在这里定义下，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Main!</string>
    <string name="hello_land">Hello World, Land Main!</string>
    <string name="app_name">LayoutsDemo</string>

</resources>
```

5.4 第四步：运行程序查询信息

运行程序，首先显示的竖屏的效果，如下图 1

图 1 竖屏的效果

可以看到，其显示的是“Hello World, Main!”，也就是使用的是竖版的布局。

紧接着，我们推开 G1 的键盘，也就是让他切换到横屏模式下（在模拟器可以使用 Ctrl+F11 键快速切换），可以看到其显示效果如下图

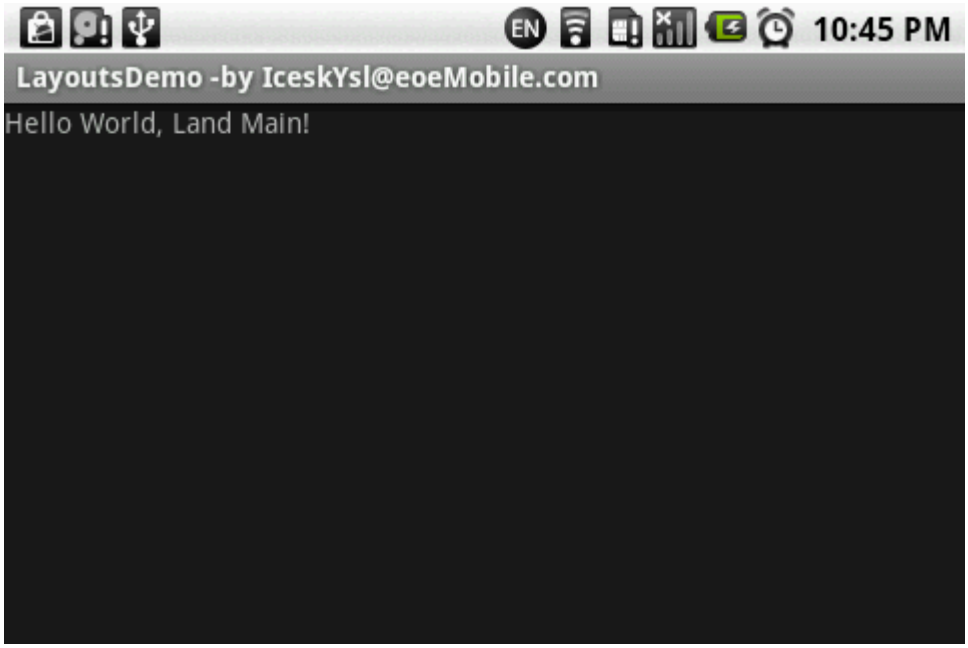


图 1 横屏的效果

可以看到，其显示的是“Hello World, Land Main!”，也就是使用的是横屏的布局。

如上文章演示了如何在 android 中通过 layout 实现横屏和竖屏的布局，希望对您有用。

注：Layouts 规划
res/layout-land -- used when the device is in landscape mode.
res/layout -- the default, used when a more specific layout mode is not matched (i.e. when device is in portrait mode)

6.如何获取当前 Locale ， 设定 Locale

By pcr

——How to make my application know where I am, in country level.

如果您的应用已经支持或打算支持，两种或两种以上的语言；那么，在应用中获取当前系统的 Locale 就变得很重要。只有通过它，我们才能确定，用户所使用的语言是不是我们的应用所支持的语言。进而，我们可以进行一些选择，挑选一种与用户语言最为接近的，我们所支持的语言，显示在用户面前。当然，如果您不去获取并设定当前应用的 Locale，也是可以的。只是这样一来，当系统的语言是您的应用所支持的语言时，系统会自动替您选择显示的语言；而如果不支持时，系统将使用默认的和区域无关的字符（String）及值（Value）。

下面我就来简单的为您介绍一下几种 Android 中常见的正确或不正确的获取 Locale 的方法。(有关如何设定将再下一章中详细介绍）。

第一种，是 Java 语言中常见的一种：直接访问 Default Locale。
示例如下：

获取： Locale locale = Locale.*getDefault*();
设置： Locale.*setDefault*(locale);

这种方法，在 Java 中可以说是通用的，但这里我们不推荐。之所以不推荐，是因为这种方法并不安全。正像您看到的，它只是一个静态变量，它在运行中极有可能被外界的环境干扰、改变；因此通过它来获取和设置 Local 是极不准确的。同时，用这种方法设置后，你的应用也只是在底层上能获取到你所设置的 Locale，界面上不会有什么效果。所以既不推荐通过它来获取，也不建议随便更改它的值。

第二种，通过 Context 获取。
在 Android 中，Context 几乎无处不在。最常见的 Activity 本身就是一个 Context，而所有的 View 在构造时，都必须传入 Context；而且，同一个 App，使用的 BaseContext 是共享的。所以通过 Context 来确定当前应用的 Locale 可以说是最安全，也最有效的。这里，我们以在 Activity 中 onCreate(Bundle savedInstanceState)方法中获取 Locale 来示例，代码如下:

```
Resources res = this.getResources();
Configuration cfg = res.getConfiguration();
Locale locale = cfg.locale;
```

这一示例中，
第一行，getResources()是 Context 类所提供的方法，用于获取资源文件读取类(Resources)的实例；
第二行， getConfiguration() 是从 Resources 实例中获取系统为当前应用所创建的 Configuration，里面储存了屏幕方向、文字拉伸等很多信息，其中就有我们这里所关心的 Locale。
第三行，不用多废话了，从 Configuration 里获取到了当前的 Locale 设定。

值得注意的是，这个方法所获得的 Locale 如果我们自己没有改过的话，就是系统为我们分配的 Locale，也就是用户在系统设置中指定的 Locale。但如果我们自己用代码更改过的话，那么在整个应用的任意一处再获得时，都会一起发生变化。有关如何更改这一设置，将在下一章详细介绍，不只是为 cfg.locale 赋一下值就完的。

第三种，通过注册网络获取

这种方法获取的值，意义上与前两种完全不同。笔者也只是在网上有见过，并没有亲身尝试，这里只是简单介绍一下。有想看原文的朋友，可以直接访问 google 的讨论组查看，地址如下：

https://groups.google.com/group/android-developers/browse_thread/thread/4d70d1a48e23ce0c/a15cb7453ffa2b1f?lnk=gst&q=+Localizing+#a15cb7453ffa2b1f

这种方法，所获取的 Locale 不是我们前面所说的用户为系统所设置的语言区域，而是手机连接到网络后，由当地网络所提供的 Locale。所以它是真正意义上的"我在哪？"当然，如果用户当时没有接入网络，就得不到了。

代码如下：

```
TelephonyManager manager = (TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
String networkOperator = manager.getNetworkOperator();
String locationCode = null;
if (networkOperator != null && networkOperator.length() >= 3) {
        locationCode = networkOperator.substring(0, 3);
}
```

这段代码，实际上是从系统服务上获取了当前网络的 MCC (移动国家号)，进而确定所处的国家和地区，比如中国大陆就是(460)。另有一种方法也能获得此代码，方法如下：

```
Resources res = this.getResources();
Configuration cfg = res.getConfiguration();
int mcc = cfg.mcc;
```

您一定，看着觉得眼熟。是的，它和我们前面获取 locale 的代码基本一样。用这种方法来获取的 MCC，是 int 型的，在没有联网时为 0。得到的结果和前面的方法完全一样。只是它没有前一种方法的实时性强；即，如果启动应用后 MCC 发生变化，这里一般还是原来的值。

以上，就是几种我们常见的，获取和 Locale 相关的信息的方法。下一章里，我们将详细为您介绍如何更改自己应用的 Locale，以实现在我们的应用中，更改显示语言的功能。

7. 如何在代码中强行指定自己 App 的 locale

By pcr

Android 系统作为一个操作系统，他所支持的地区及语言种类是相当多的。而作为我们一般开发者而言，我们不可能有那么雄厚的财力物力，让我们制作的应用和 Android 上面支持的语言一样多。因此分歧就出现了，如果用户当前系统设置的默认语言，我们的应用里没有怎么办？如果我们希望用户在应用中能自行选择支持的语言，如何实现呢？下面我们就通过几段代码来解答这个貌似复杂的问题。

7.1 如何来简单的获取当前的 locale

首先，如果您希望用户能自行选择支持的语言，那么肯定，您不会再继续让系统替您选择语言了。要实现这一功能，其实很简单，只需要在重载 Activity 的 onCreate(Bundle savedInstanceState) 时，实现下面这样一段代码就可以了：

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Locale locale = getLocaleFormPreference(this);
        Resources res = getResources();
        Configuration cfg = res.getConfiguration();
        cfg.locale = locale;
        res.updateConfiguration(cfg, null);

        mLocale = locale;
        setContentView(R.layout.xxxx);
        ……
}
```

下面对这段代码，进行一下简单的说明。前两行就不提了，大家重载时都会写。

- 第 4 行：Locale locale = getLocaleFormPreference(**this**);
  这一行实际上是我单独封装了一个方法，用于将 Preference 里保存的用户选择的语言所对应地区提取出来。它的返回值一般是 Locale.SIMPLIFIED_CHINESE 或别的 Locale 实例。因为会用到 Preference，所以传入了 Activity.this。
- 之后，第 5、6 行我们分别获得了当前 Activity 的 Resources 和 resources 里的 Configuration。
- 并在第 7 行更改了 Configuration 里的 locale 设置，但这并不算完。
- 必须调用第 8 行的 res.updateConfiguration(cfg, **null**)，来通知 Resources，更改读取 assets 等的位置。至此，一个 Activity 的语言区域就被我们单独指定了。

除了刚才已经描述过的部分，这里还有两点注意事项：

1. 如果你会调用类似 setContentView(resID)的方法，那么最好像我们示例中一样，将它放在 updateConfiguration(cfg, **null**)之后。这样在填充 view 的内容时您刚才指定的 Locale 就会立刻发生作用。
2. 可能您在您的第一个 Activity 上做了上面的操作后，您会发现，后续启动的所有 Activity 的语言都已经换过来了。不错，因为这些 Activity 的 Resources 实际上是共用的。但

千万别偷懒，一定要在每一个处在根位置的 Activity 里都加上相同功能的代码。因为 Activity 是有生命周期的，当你的 Activity 全被系统回收了，而你唤醒它时，被重建的不是一个做了 locale 指定的 Activity，那你的语言设定，就又被系统设定所取代了。而且，我们后面讲解的功能实际上也依赖于此。所以为了方便，您也可以考虑将那段代码，整体封装成一个外部的静态方法。

7.2 如何在运行时动态改变 locale

在上一节中，我们已经成功让我们的应用拥有自己指定的 locale 了，接下来，您可能会希望用户在使用时可以通过 Preference 或其他的设置方法，动态的改变当前显示的语言了。那么接下来的代码，将为您给出这个功能的一种实现方法。

在正式讲解代码之前，请允许我稍微描述一下，在实现这一功能时，我们可能遇到的问题，再由此来说明我为什么会建议这样一种实现方法。
首先，如果您已经作过尝试，您会发现，在 Activity 已经显示在屏幕上以后，您再运行我们前面讲过的那段代码，屏幕上并不会有什么变化；而且退回之前的 Activity 只要不是因为系统回收或屏幕方向改变等原因，造成 onCreate(Bundle savedInstanceState)被重新调用；这些之前的 Activity 的语言一样没有变化；只有新建的 Activity 的语言才是新的。又因为 Activity 之间是不太容易互相访问的，所以我们需要解决的问题就变成了：1. 如何动态更新当前显示的 Activity;和 2. 怎样在返回前面的 Activity 时，那个 Activity 也能自动更新。

7.3 首先，我们来说 1——如何动态更新当前显示的 Activity

如果是传统的开发，或许您会说那很简单，把每个屏幕上元素的字符都重设一遍就好了。是的，这么做是很简单，但我不得不说，它的工作量和维护成本实在是不可预知的；而且，除了字符，在其它方面，如 layout、drawable 等，都可能受 Locale 影响。所以，我们需要一种更通用、更简洁的方法。
应该注意，下面我介绍的方法，是即使在 Google 的 android 讨论组里也搜不到的。那么这个方法是什么呢？那就是重新启动一个 Activity。
或许您会觉得这个方法消耗过大，但如果您想到，其实 android 屏幕方向改变都能导致 Activity 重起也就释然了。而且，这个方法我其实借用了一个假设，那就是大家都已经了解、考虑并且为了 Activity 的生命周期做出了一定的努力，已经良好的实现了 onRestoreInstanceState(savedInstanceState) 和 onSaveInstanceState(bundle)。当然，如果您的应用根本不担心生命周期，那么你就更没问题了。但如果您的应用在生命周期这一问题上，还有问题，那很遗憾，您最好还是先把生命周期这个严重的问题搞好吧。下面我们就来看代码，实现方法如下。
首先，在用户改变语言区域后，调用如下代码

```
Locale locale = getLocaleFormPreference(this);
Configuration cfg = getResources().getConfiguration();

if(locale!=null && !locale.equals(cfg.locale)) {
        Bundle bundle = new Bundle();
        onSaveInstanceState(bundle);
```

```
            Intent intent = getIntent();
            intent.putExtra("InstanceState", bundle);
            intent.addFlags(Intent.FLAG_ACTIVITY_FORWARD_RESULT);
            startActivity(intent);
            finish();
    }
```

之后，在被启动的 Activity 的 onCreate(Bundle savedInstanceState)的最后位置运行下面的代码：

```
    if(savedInstanceState == null) {
            savedInstanceState =

                    getIntent().getBundleExtra("InstanceState");
            if(savedInstanceState != null) {
                    onRestoreInstanceState(savedInstanceState);
            }
    }
```

下面，我再为大家一行一行的，讲解一下这两段代码：

- 首先，我们在第 1 段代码的前 4 行对 Locale 进行了一下判断，使得只在 Locale 发生变化的情况下，才重起 Activity。
- 之后，5~8 行，我们手动调了的 onSaveInstanceState(bundle)方法，获得了恢复 Activity 所需要的信息，并以"InstanceState"为 Key 存入了 Intent。
- 第 9 行，我们为 Intent 添加了一个 flag——*FLAG_ACTIVITY_FORWARD_RESULT*。这一步，是可选的，目的是如果当前的 Activity 需要给启动它的 Activity 一个回复的 code 时，由它新启动的 Activity 来回复。
- 最后 10~11 行是启动新的 Activity 并终止当前的。
- 而第 2 段代码则是通过判断 savedInstanceState 来选择和模拟系统 restart 一个 Activity 的过程。因为默认新启动 Activity 时，savedInstanceState 是空的，而重启动时 savedInstanceState 会带有信息，因此我们可以这样去模拟。

这里，同样有两点需要注意：

1. 这段代码需要在根 Activity 上执行，如果您应用了 ActivityGroup 并需要在 child 上执行更新操作。请相应更改第一段代码的第 6、7 行，调用根 Activity 上的方法。而第 2 段代码，仍然放在根 Activity 的 onCreate(Bundle savedInstanceState)里。
2. 如果您处理生命周期的操作，除了在 onRestoreInstanceState()里，其他地方，也有操作，请自行想办法，按照您需要的顺序模拟 restart 操作。

7.4 最后，我们介绍一下 2——怎样在返回前面的 Activity 时，那个 Activity 也能自动更新。

方法和动态更新时差不多。在每一个可能在改变语言区域的 Activity 之前显示出来的 Activity 中，重载 onResume()并加入如下代码：

```
    oldLocale = mLocale;
    Locale locale = getLocaleFormPreference(this);

    if(oldLocale!=null && !oldLocale.equals(locale)) {
            Bundle bundle = new Bundle();
```

```
            activity.onSaveInstanceState(bundle);
            Intent intent = new Intent(this, this.getClass());
            intent.putExtra("InstanceState", bundle);
            intent.addFlags(Intent.FLAG_ACTIVITY_FORWARD_RESULT);
            startActivity(intent);
            finish();
    }
```

并在 onCreate(Bundle savedInstanceState)最后加入和动态更新时完全相同的代码。

因为代码相似度很高，就不再细说了。唯一要说明的是 mLocale 实际上是在 onCreate(Bundle savedInstanceState)中，指定的 Locale 时(也就是上一节最初时那段代码)记住的。

关于这段代码，仍然有一点注意事项：

那就是这一行：Intent intent = **new** Intent(**this**, **this**.getClass());

这里我们没有使用 getIntent()或 **new** Intent(getIntent())来获取 Intent。主要的原因是，如果我们重启的 Activity 恰好是这个 Application 的初始页面，将导致新启动的 Activity 没有焦点而直接关到后台去，其它情况下，用哪个都没问题。而如果，你的 Activity 启动需要要一些原有 Intent 的数据，别忘了自行导入。

好了，这就是我给您关于这个功能的一种，个人认为比较简单的实现建议。如果您还有更理想的方案，希望您能来 www.eoeandroid.com 与我们共同分享。

8.Android Applications Localization Helper (Android 本地化助手)

By 游利卡
前言:

在看这期任务的时候，就感觉这期任务一定特别困那。但是翻译了第一篇文章以后，居然发现，这个国际化在 Android 上居然这么简单。简直让人无法想象。
第一篇就是量多了一些，美国人的废话多了一些(启示自己也不少)

第二篇来介绍一下这个 Android Application Localization Helper（以下简称 AALH）。因为知道国际化很容易了，自然也就觉得这个工具应该也不是太负责。拿到了以后的确，和我想的一样。而且还提供了源码的下载。刚一打开一个 sln 文件，咦，怎么这么熟悉。剩下的都是 cs 文件。cs 文件，难道~！
God!God！God!，他居然是用 C#写的！他居然是用 C#写的！他居然是用 C#写的！他居然是用 C#写的！他居然是用 C#写的！他居然是用 C#写的！

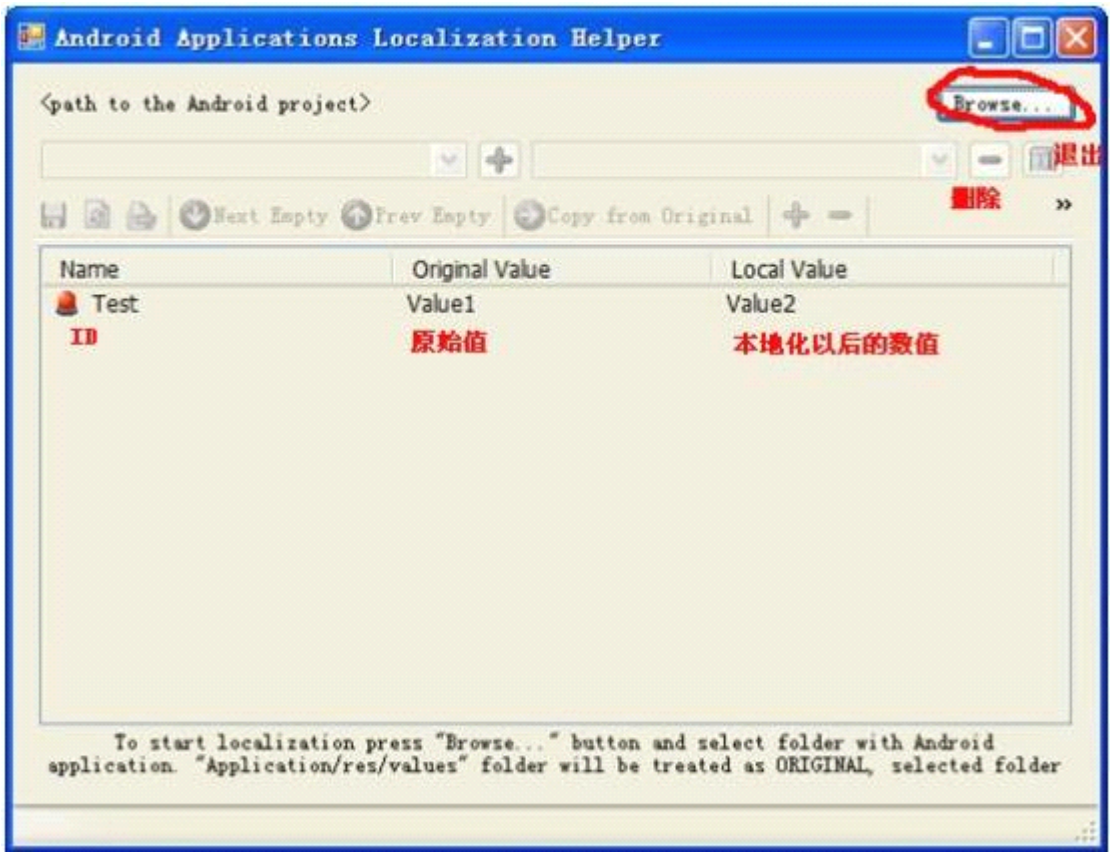
我那个感动啊，真是一把鼻涕一把泪。因为我就是从.NET 平台出身的。所以 C#自然也是非常熟悉了。因为机器在上次恢复了以后没装 VS，只能用技术本来打开源码了。都是熟悉的句子啊。特别是刚开始的 using 每次的 namespace 命名空间。当时真恨不得写一个源码分析来了。

开始正题；
大家可以到这里下载 AALH，包括源代码。
http://www.artfulbits.com/Android/Localizer.aspx

看过源码了。因为使用的是 C#开发，所以理论上只有 Windows 用户才有使用这个的机会了，而且一定要保证你的机器上安装了.NET framework。至少 2.0 版本以上，那是微软的一套库文件。C#就是靠这个来运行的，他相当于 Java 平台的 JRE。如果你想深入了解这个，那就需要到.NET 的社区去看了，C#的语法和 Java 很像。所以转换起来非常方便，不过要晋级开发还需要一些功夫。
Linux 用户我不是很清楚，但是肯定的是 wine 肯定是不够的，在 linux 上，微软也放出了 linux 上的.NET framework 叫做 mono，两者配合能不能使用这个，只能说大家试试了。

C#奉行的是快速开发，提供了非常丰富的组件，因为国际化只涉及 xml 文件，所以 C#的确是不二之选。

介绍一下界面。



AALH 的界面非常简单 最右上角的可以用来导入你的项目文件。下面两个功能一个删除一个退出

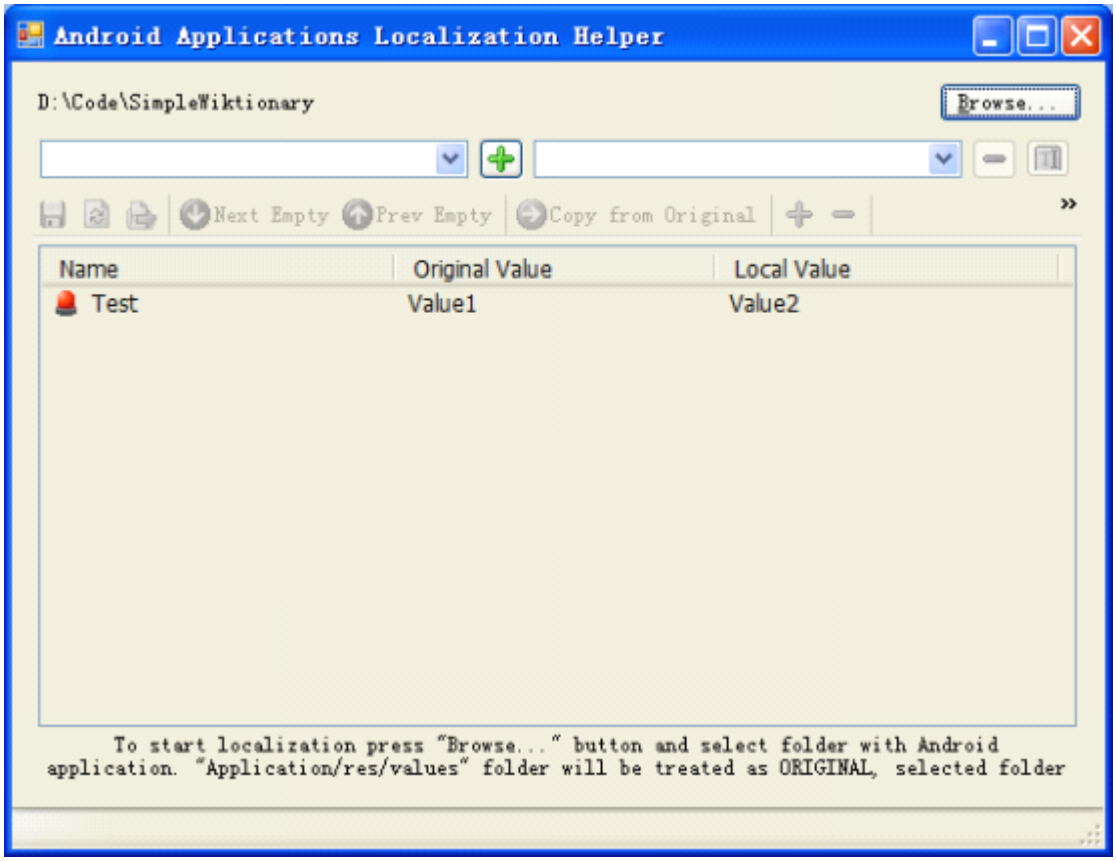
而中间大的框框，就是现实相应的文字信息的 ID 原始值 本地化后的值

如何使用:

其实 Android 的国际化工作之需要把所有的文字处理好就可以了。当然其实还有很多其他的工作，只不过文字工作量最大。而且有的时候国际化的时候有一些防止在 Value 的文件夹的文件并不需要改变。

而当你导入到了你的项目文件的时候，AALH 会自动搜索到你的 Value 文件夹。如果没有设定国际化内容。在左边第一个框中就不会显示出任何内容。

好我们现在来导入上一期的维基词典。



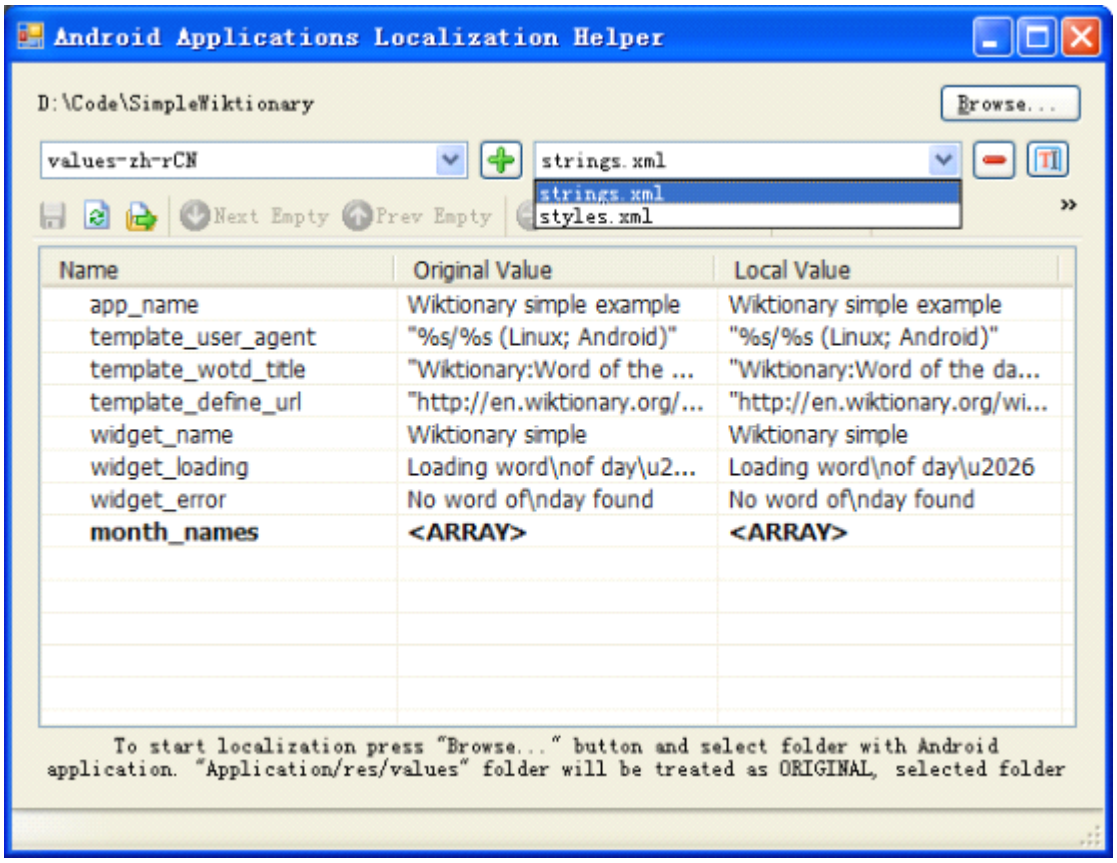
如果你的文件中，全都是默认值，没有任何国际化的内容那么左边的第一个方框就会空，点开下来菜单也不会有东西。

但是这时，你就可以来添加你的国际化文件夹了。点击那个绿色的加号，然后。

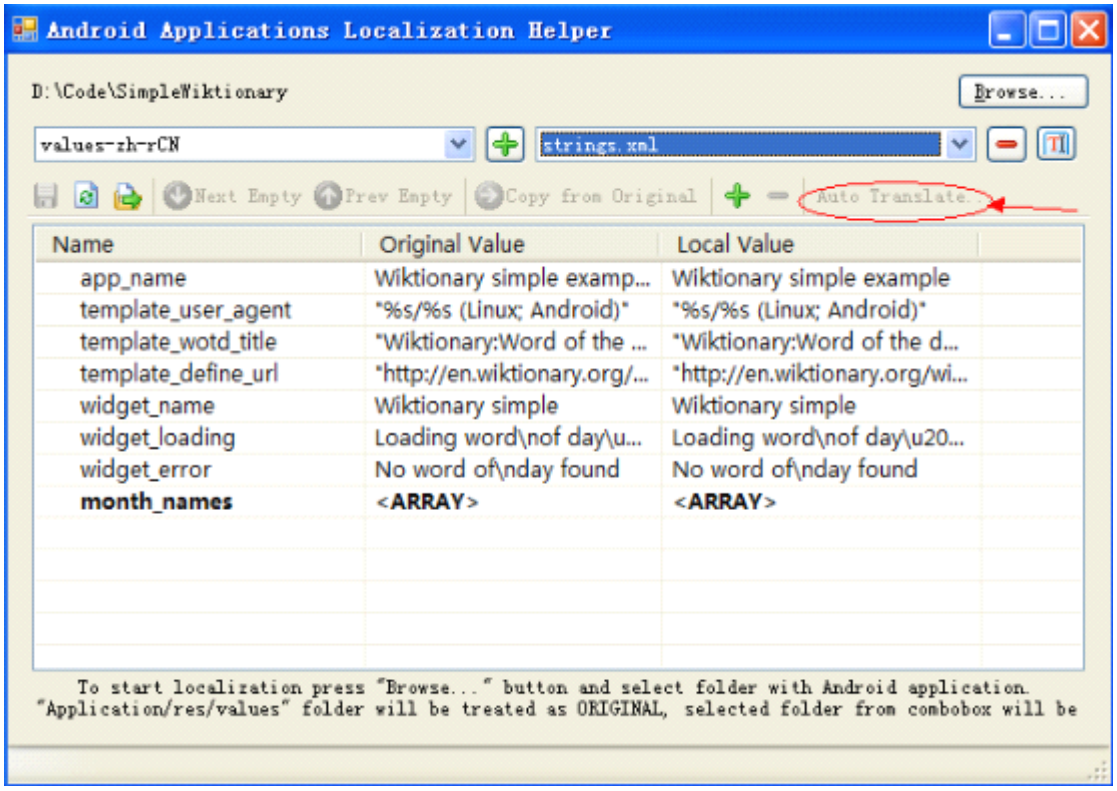
我们直接添加我们要国际化的字符就可以了。下面还提供了很多的提示，支持的语言，非常方便。这里的中文的 zh-CN 可不要直接写上去。因为国庆，所以 zh 分还多。而第二个代表地区，地区必须加上 r 的前缀，正确应该是 zh-rCN

好咱们用中文吧

当添加好了以后 AALH 会自动把原有的 Value 文件夹里的文件复制新创建的 Value-** 我们这里是 Value-zh-CN



我们添加了 String 字符串。然后他就会把 xml 文件中所有的项目按照 id 还有数值的形式保存下来。
虽然才 40K 但是他居然可以使用 Google 的翻译工具。



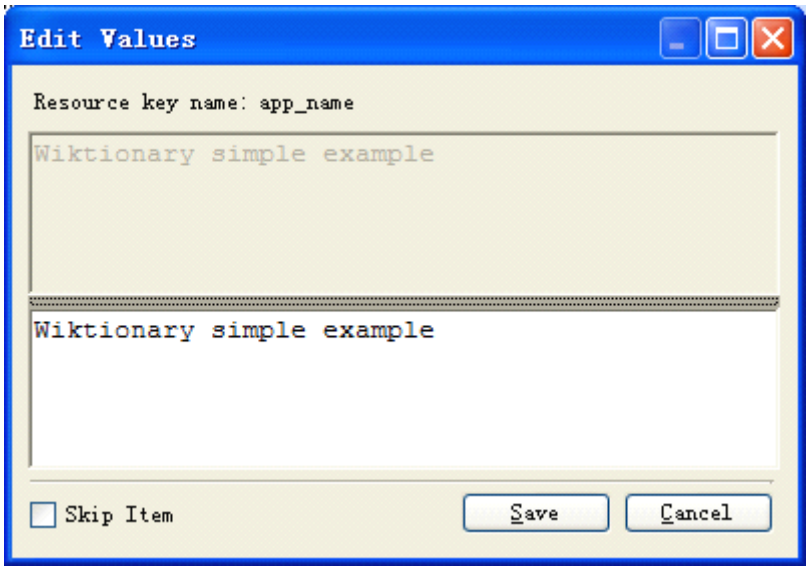
看右上角

可是对于我们博大精深的汉语来说，这个还不是很好用。而且这个 Widget 还有很多的其他字符，所以当前翻译功能被禁止了。

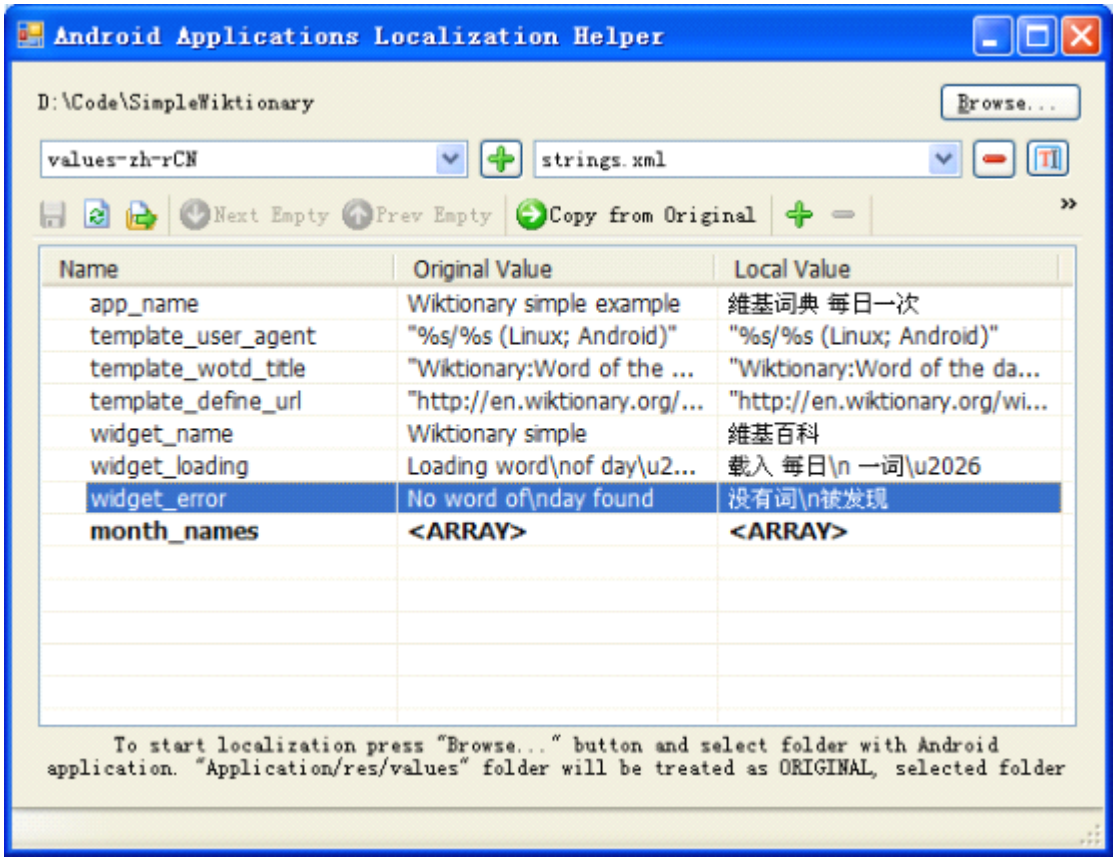
咱们只能手动来写了。

其实右边的就是让我们来编辑的了。这里我们可以看到因为 xml 都格式化好了，所以呢之需要改变数值就可以了，这比直接编辑 XML 文件要方便不少。好现在咱们开始进行中文化了。

双击你要本地化的项目。然后就会



一个对话框，直接在里面编辑就可以了。



现在我们编辑好了。

不过在我们实际使用的时候，需要把文件夹中的一个 bak 文件删掉，不然会报错。

现在已经变成中文的了

点击进去看看

本来 Android 的国际化就比较简单，有了 AALH 更是如鱼得水了。

9. 编后语

可
By 游利卡

首先在这里想说的还是抱歉，因为我之前没有安排好时间，让这次特刊迟了这么长时间才与大家见面。第五期的特刊是 ice 参与最少的特刊，所以很多东西都承担在了我的身上，在发布之前的两个小时，本来正在排版的时候，居然中途睡着了。罪过罪过！

但是本期内容同样十分丰富，绝对不辜负大家对于 eoe 特刊的期待。

关于国际化，大家可能想着这会是十分复杂的东西，但是事实证明，Android 平台的 framework 让我们这方面的工作变得十分轻松而有趣。

如果还认为国际化的工作会很难很复杂，那看完本期特刊再下定论吧！

此外本期特刊还帮助你如何获取横竖屏实际那、如何获取当前的 Locale 等等，总之，尽情享受吧！

10. 其他

BUG 提交

如果你发现文档中翻译不妥的地方，请到如下地址反馈，我们会定期更新、发布更新后的版本
http://www.eoeandroid.com/viewthread.php?tid=753

资源下载:

本期文章中包含的源码请在如下地址下载:
http://www.eoeandroid.com/viewthread.php?tid=753

参加翻译

如果你有兴趣参加我们的特刊，请参考如下链接
http://www.eoeandroid.com/forumdisplay.php?fid=39

关于 eoeAndroid

当前，3G 商业，传统互联网与移动互联网也呈现出全业务发展的融合趋势，电信与互联网行业已经踏入继单机计算时代、传统互联网时代之后的第三个纪元。
由于看好移动互联网和 Android 手机平台的商业前景，同时也拥有专业而独特的产品、技术服务能力，我们聚集了一群热爱 Android 的技术英才，组建了 eoeMobile 团队。
eoeMobile 是一支专注于 Android 平台应用开发、产品运营和相关商业与技术服务的团队，立志于建立中国最大的 Android 应用开发专业社区 eoeAndroid.com，想为 Android 在中国的发展尽自己的微薄之力。