第 15 章

应用本地化 15

本地化是一个基于设备语言设置，为应用提供合适资源的过程。本章我们将对HelloMoon应用进行本地化，并为其提供中文版本的字符串资源。当设备的语言被设置为中文时，Android会自动找到并使用相应的中文资源，如图15-1所示。

图15-1　你好, 月球

15.1　本地化资源

语言设置是设备配置的一部分。Android提供了用于不同语言的配置修饰符。本地化处理因而变得简单：首先创建带有目标语言配置修饰符的资源子目录，然后将可选资源放入其中。Android资源系统会为我们处理其他后续工作。

语言配置修饰符来自于ISO 639-1标准代码。中文的修饰符为-zh。在HelloMoon项目中，新建两个res子目录：res/raw-zh/和res/values-zh/。

可查阅随书代码（http://www.bignerdranch.com/solutions/AndroidProgramming.zip），查找所需的资源。在文件包中，找到以下文件：

15_Localization/HelloMoon/res/raw/one_small_step.wav

15_Localization/HelloMoon/res/values/strings.xml

将以上文件复制到刚才新建的对应目录中。（中文资源请读者自行处理，随书代码包只包含英文和西班牙版本的资源。）

如此一来，便完成了为应用提供本地化资源的任务。如需验证成果，可打开Settings应用，找到语言设置选项，然后将设备语言改为简体中文即可。尽管Android系统版本繁多，但语言设置选项通常被标为Language and input、Language and Keyboard或其他类似名称。

运行HelloMoony应用。单击播放按钮，欣赏阿姆斯特朗的著名讲话。

默认资源

中文语言的配置修饰符为-zh。本地化处理时，我们的第一反应可能是直接重命名原来的raw和values目录为raw-zh和values-zh。

这可不是个好主意。这些没有配置修饰符的资源是Android的默认资源。默认资源的提供非常重要。如果Android无法找到匹配设备配置的资源，而又没有默认资源可用时，应用将会崩溃。

例如，如果在values-en/以及values-zh/目录下分别有一个strings.xml文件，而在values/目录下并没有准备string.xml文件，那么运行在语言设置既非英文也非中文的设备上时，HelloMoon将会崩溃。因此，即便未提供匹配的设备配置资源，默认资源仍能保证应用的正常运行。

例外的屏幕显示密度

Android默认资源使用规则并不适用于屏幕显示密度。项目的drawable目录通常按屏幕显示密度要求，具有三类修饰符：-mdpi、-hdpi和-xhdpi。不过，Android决定使用哪一类drawable资源并不是简单地匹配设备的屏幕显示密度，也不是在没有匹配的资源时直接使用默认资源。

最终的选择取决于对屏幕尺寸和显示密度的综合考虑。Android甚至可能会选择低于或高于当前设备屏幕密度的drawable资源，然后通过缩放去适配设备。可访问网页http://developer.android.com/guide/practices/ screens_support.html，了解更多相关信息。尽管Android的drawable资源使用稍显复杂，但请记住一点：无需在res/drawable/目录下放置默认的drawable资源。

15.2　配置修饰符

目前为止，我们已经见识并使用过几个用于提供可选资源的配置修饰符，如语言（values-zh/）、屏幕方位（layout-land/）、屏幕显示密度（drawable-mdpi/）以及API级别（values-v11）。

表15-1列出了一些具有配置修饰符的设备配置特征，Android系统通过这些配置修饰符识别并定位资源。

表15-1 具有配置修饰符的设备特征

1	移动国家码，通常附有移动网络码
2	语言代码，通常附有地区代码
3	布局方向
4	最小宽度
5	可用宽度
6	可用高度
7	屏幕尺寸
8	屏幕纵横比
9	屏幕方位
10	UI模式
11	夜间模式
12	屏幕显示密度
13	触摸屏类型
14	键盘可用性
15	首选输入法
16	导航键可用性
17	非文本导航方法
18	API级别

可访问网页http://developer.android.com/guide/topics/resources/providing-resources.html，查看表中所有设备配置特征描述及其对应配置修饰符的使用例子。

15.2.1 可用资源优先级排定

考虑到有那么多定位资源的配置修饰符，有时可能会出现设备配置与好几个可选资源都匹配的情况。假设发生这种状况，Android会基于表15-1所示的顺序确定修饰符的使用优先级。

为实际了解这种优先级排定，我们来为HelloMoon应用再添加一种可选资源，即水平模式的app_name字符串资源。app_name资源将作为activity的标题显示在操作栏上。设备处于水平模式时，操作栏上可显示更多的文字。我们可以尝试设置一段对话，而不只是“你好，月球”这几个字。

创建一个values-land目录，然后将默认的字符串资源文件复制进去。

水平模式下，只需改变app_name字符串资源即可。删除其他无需变化的字符串资源，然后对照代码清单15-1修改app_name的值。

代码清单15-1 创建水平模式的字符串资源（values-land/strings.xml）

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

```
    <string name="app_name">Hello, Moon! How are you? Is it cold up there?</string>
    <string name="app_name">HelloMoon</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="hellomoon_play">Play</string>
    <string name="hellomoon_stop">Stop</string>
    <string name="hellomoon_image_description">Neil Armstrong stepping
            onto the moon</string>
</resources>
```

字符串备选资源（也包括其他values资源）都是基于每一个字符串提供，因此，字符串资源相同时，无需再复制一份。重复的字符串资源只会导致未来的维护噩梦。

现在总共有三个版本的app_name资源：values/strings.xml文件中的默认版本、values-zh/strings.xml文件中的中文备选版本以及values-land/strings.xml文件中的水平模式备选版本。

在设备语言设置为简体中文的前提下，运行HelloMoon应用，然后旋转设备至水平模式。因为中文备选版本资源优先级最高，所以我们看到的是来自于values-zh/strings.xml文件的字符串资源，如图15-2所示。

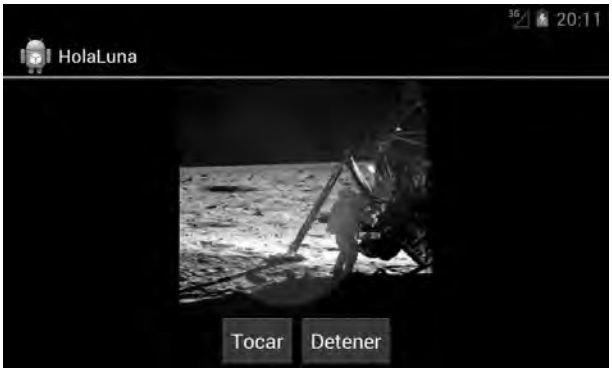

图15-2　Android排定语言优先级高于屏幕方位

也可以将设备的语言重新设置为英语语言，然后再次运行应用，确认水平模式下的备选字符串资源如期出现。

15.2.2　多重配置修饰符

可以在同一资源目录上使用多个配置修饰符。创建一个名为values-zh-land的资源目录，为HelloMoon应用准备水平模式的中文字符串资源。

在同一资源目录上使用多个配置修饰符，各配置修饰符必须按照优先级别顺序排列。因此，values-zh-land是一个有效的资源目录名，而values-land-zh目录名则无效。

注意，语言区域修饰符，如-es-rES，看上去像是由两个不同的配置修饰符组成的多重配置修饰符，但实际上它仅是一个独立的配置修饰符（区域部分本身不能构成一个有效的配置修饰符）。

将values-land/strings.xml文件复制到values-zh-land/目录中，然后按照代码清单15-2修改相应内容。

代码清单15-2 创建水平模式下的中文版字符串资源（values-zh-land/strings.xml）

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">Hello, Moon! How are you? Is it cold up there?</string>
  <string name="app_name">¡Hola, Luna! ¿Como estás? ¿Hace frío ahí arriba?</string>
</resources>
```

在设备语言已设置为中文简体的前提下，运行HelloMoon应用，确认新的备选资源如期出现，如图15-3所示。

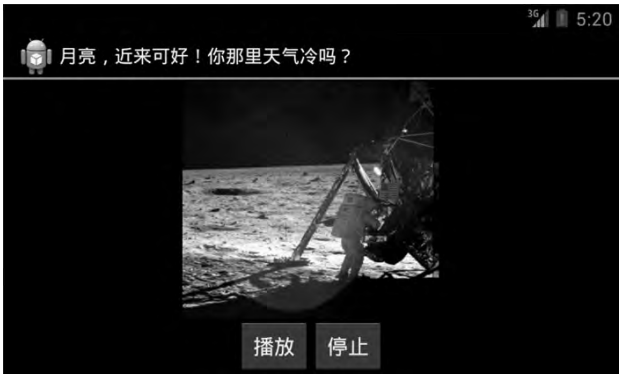

图15-3 水平模式下的中文字符串资源出现了

15.2.3 寻找最匹配的资源

我们来看看Android是如何确定app_name资源版本的。首先，记得当前设备有以下三个版本的app_name字符串备选资源：

values-zh/strings.xml

values-land/strings.xml

values-zh-land/strings.xml

其次，在设备配置方面，当前的设备配置包括中文简体语言以及水平屏幕方向。

1. 排除不兼容的目录

要找到最匹配的资源，Android首先要做的就是将不兼容当前设备配置的资源目录排除掉。

结合备选资源和设备配置来看，三个版本的备选资源均兼容于设备的当前配置。如果将设备旋转至竖直模式，设备配置则会发生改变。此时，values-land/与values-zh-land/资源目录由于不兼容当前配置，因此可以被过滤掉。

有些配置修饰符的兼容性并不具有非黑即白的排他性。例如，API级别就不是一个严格匹配的修饰符。修饰符-v11可兼容运行API 11级及更高级别的系统版本设备。

有些配置修饰符被添加到较新的API级别中。例如，API 17级中引入的布局方向配置修饰符。该配置修饰符有两种选择：-ldltr（自左向右）和-ldrtl（自右向左）。比较晚添加的配置修饰符一般带有隐含的API级别修饰符，因此，-ldltr也可看作是-ldltr-v17。（这些附加的隐含配置修饰符也

从另一方面解释了为何要以防万一在项目中准备默认资源。）

Android对屏幕显示密度采取了不同的处理方式。因此，通常的配置修饰符与设备配置兼容匹配的原则不适用于它。Android会选择其认为最合适的资源来匹配设备配置，这些资源可能是，也可能不是对应设备配置的配置修饰符目录下准备的资源。

2. 按优先级表筛选不兼容目录

筛选掉不兼容的资源目录后，自优先级别最高的MCC开始，Android逐项查看并按优先级表继续筛选不兼容的目录（表15-1）。如有任何以MCC为配置修饰符的资源目录存在，那么所有不带有MMC修饰符的资源目录都会被排除掉。如果仍有多个目录匹配，则Android将继续按次高优先级进行筛选，如此反复，直至找到唯一满足兼容性的目录。

本例中没有目录包含MCC修饰符，因此无法筛选掉任何目录。接着，Android查看到次高优先级的设备语言修饰符。三个资源目录中，values-zh/和values-zh-land/目录包含语言修饰符，这样，不包含语言修饰符的values-land/即可被排除。

Android继续查看优先级表，接下来是屏幕方向。此时，Android会找到一个带屏幕方位的修饰符目录以及一个不带屏幕方向的目录，由此，values-zh/目录也被排除在外。就这样，values-zh-land/成了唯一满足兼容的目录。因而，Android最终确定使用values-zh-land/目录下的资源。

15.3　更多资源使用原则及控制

现在，我们已经对Android资源系统有了进一步的了解。不过，为将来开发自己的应用做准备，还应了解以下资源使用方面的要求。

15.3.1　资源命名

资源的名字只能由小写字母组成并且不能包含空格，一些正确命名的例子有：one_small_step.wav, app_name, armstrong_on_moon.jpg。

无论是在XML还是在代码中引用资源，引用都不应包括文件的扩展名。例如，在布局文件中引用的@drawable/armstrong_on_moon以及在代码中引用的R.drawable.armstrong_on_moon。这也意味着，在同一子目录下，不能以文件的扩展名为依据，来区分命名相同的资源文件。

15.3.2　资源目录结构

所有资源都必须保存在res/目录的子目录下。尝试在res/目录的根目录下保存资源将会导致编译错误。

res子目录的名字直接与Android编译过程绑定，因此无法随意进行更改。我们已看到过的子目录有drawable/、layout/、menu/、raw/以及values/等。也可访问网页http://developer.android.com/guide/topics/resources/available-resources.html，查看系统支持的（无修饰符的）res子目录的完全清单。

Android会无视res/目录下的其他子目录。创建res/my_stuff可能不会导致错误发生，但Android不会使用放置在其中的任何资源。

此外，我们也无法在res/目录下创建多级子目录。这种限制会给开发带来些许麻烦。要知道，实际开发项目中往往有几百个drawable资源。因此，创建多级子目录来管理这些资源文件是很自然的想法，但Android不允许这样做。既然这样，我们唯一能做的就是有意识的对资源命名，使其可按文件名进行排序，以便于查找某个特定文件。

Android对布局文件的命名约定就是按文件名排序的典型例子。布局文件通常以其定义的视图类型名作为前缀，如`activity_`、`dialog_`，以及`list_item_`等。例如，在CriminalIntent应用的res/layout/目录下，一些布局命名有：`activity_crime_pager`、`activity_fragment`、`dialog_date`、`fragment_crime`和`list_item_crime`。可以看到，activity布局按照名称进行排序，这样查找某个activity布局文件就容易多了。

15.4 测试备选资源

应用开发时，布局以及其他资源的测试非常重要。针对不同尺寸的屏幕、屏幕方位等设备配置，布局测试可提前让我们知道所需的备选资源数量。我们可以在虚拟设备上测试，也可以在实体设备上测试，甚至还可以使用图形布局工具进行测试。

图形布局工具提供了很多选项，用以预览布局在不同配置下的显示效果。这些选项有屏幕尺寸、设备类型、API级别以及设备语言等。

要查看这些选项，可在图形布局工具中打开fragment_hello_moon.xml文件。然后参照图15-4，尝试使用各种选项对布局进行预览。

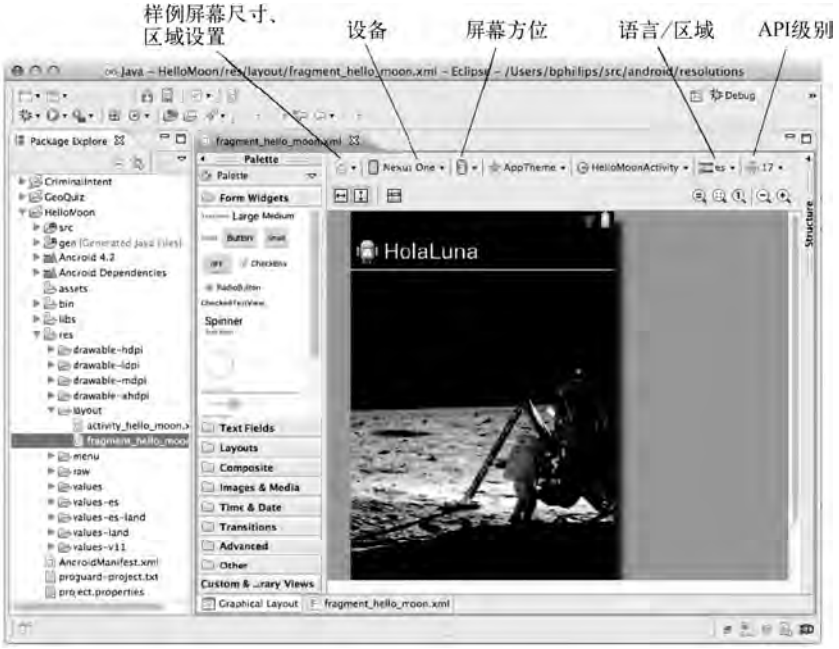

图15-4 使用图像布局工具预览资源

通过设置设备语言为未提供本地化资源的语言，可确保项目已包括所有必需的默认资源。运行应用进行测试，查看所有视图界面并旋转设备。如应用发生崩溃，请查看LogCat中“Resource not found...”错误信息，确认缺少哪些默认资源。