# 信息绑定

程序中的一些文字信息可以将之定义在一个属性文件中，而不一定要写在程序代码中，这样在日后想要更改文字信息时，只要更改文本文件的内容而不用重新编译程序，就可以在程序运行时显示不同的信息。

# 使用ResourceBundle

使用java.util.ResourceBundle来做信息绑定，首先要准备一个.properties文件，例如messages.properties,而文件内容如下：

onlyfun.caterpillar.welcome=Hello

onlyfun.caterpillar.name=World

.properties文件必须放置在Classpath的路径下，文件中编写的是键、值配对，之后在程序中可以使用键来取得对应的值，

范例18.15

**public** **class** ResourceBundleDemo {

**public** **static** **void** main(String[] args) {

//绑定messages.properties

ResourceBundle resource = ResourceBundle.*getBundle*("messages");

//取得对应信息System.*out*.print(resource.getString("onlyfun.caterpillar.welcome")+"!"); System.*out*.println(resource.getString("onlyfun.caterpillar.name")+"!");

}

}

ResourceBundle的静态getBundle()方法会取得一个ResourceBundle的实例，所给定的参数名称是信息文件的主文件名，getBundle()会自动找到对应的.properties文件。取得ResourceBundle实例后，可以根据getString()指定键来取得文件中对应的值。

结果如下：

Hello!World!

如果日后想要改变显示的信息，只要改变.properties文件的内容就可以了。

# 格式化信息

程序在运行的过程中，有些信息可能必须动态决定，而之前介绍信息绑定时，.properties文件中的信息则是静止的，也就是固定的文字内容，除非修改.properties文件内容并重新启动程序，否则一些信息内容无法随着程序动态显示。

可以使用java.text.MessageFormat类来辅助信息的格式化，MessageFormat接受一个字符串模式(Pattern)指定。对于文字信息中可能变动的部分，可以使用参数索引（Argument Index）先占住文字位置，参数索引是{0}～{9}的非负整数，之后在使用MessageFormat实例的format()方法时，可以提供真正的参数来填充参数索引处的信息。

举个例子：

String message = "Hello ! {0} ! This is your firse {1}";

Object[] params = **new** Object[]{"caterpillar","Java"};

MessageFormat formatter = **new** MessageFormat(message);

//显示格式化后的信息

System.*out*.println(formatter.format(params));

MessageFormat实例的format()方法会使用params数组中对象的toString()方法，将取得的String信息依索引位置分别填入{0}到{9}的对应位置，执行format()方法后返回的就是格式化后的信息内容。就上面的程序片段而言，会显示：

Hello ! caterpillar ! This is your firse Java

依照这样的方法，如果想在信息绑定时也能进行信息的格式化，让一些信息在程序运行过程中动态决定，则可以在.properties中如下编写，例如编写messages2.properties:

onlyfun.caterpillar.greeting=Hello! **{0}**! This is your first **{1}**!

接着可以绑定messages2.properties,并在程序中进行信息的格式化，如范例18.16所示.

**import** java.text.MessageFormat;

**import** java.util.ResourceBundle;

**public** **class** MessageFormatDemo {

**public** **static** **void** main(String[] args) {

**try** {

ResourceBundle resource = ResourceBundle.*getBundle*("messages2");

String message = resource.getString("onlyfun.caterpillar.greeting");

Object[] params = **new** Object[] { args[0], args[1] };

MessageFormat formatter = **new** MessageFormat(message);

System.*out*.println(formatter.format(params));

} **catch** (ArrayIndexOutOfBoundsException e) {

System.*out*.println("没有指定参数");

}

}

}

程序运行时必须给它两个参数，分别作为{0}与{1}的信息填充。执行时的一个例子如下：

java onlyfun.caterpillar.MessageFormatDemo caterpillar Java

Hello! caterpillar! This is your first Java!

# 国际化信息

国际化的英文是Internationalization，简称I18N .国际化的目的是让应用程序可以依地区不同而显示不同的信息，最基本的就是让不同语系的用户可以看到属于自己语系的信息，像英文语系的看到英文的内容，而中文语系的看到中文的内容。

为了在应用程序中表示一个区域，Java提供有java.util.Local类，一个 Locale实例包括了语系信息和区域信息。例如en表示英文语系的国家，这个字母组合是在ISO 639中定义的，而区域信息则是像US表示美国，这个字母组合是在ISO 3166中定义的。

可以这么新增一个Locale实例，用以表示中文语系zh 、中国 cn :

Locale locale = **new** Locale("zh","cn");

如何将Locale用于信息绑定呢？当使用ResourceBundle.getBundle()方法时，默认就会自动取得计算机上的语系与区域信息，而事实上信息文件的名称由basename加上语系与地区来组成。例如：

basename.properties(默认)

basename\_en.properties

basename\_zh.properties

没有指定语言与地区的basename是默认的资源文件名称，当没有提供专用的语系、区域信息文件时，就会寻找默认的资源文件。

如果想要提供中文的信息，由于信息资源文件必须是ISO-8859-1编码，所以对于非西方语系的处理，必须先将之转换为 Java Unicode Escape 格式 。

例如：可以在messages\_zh\_CN中编写以下的内容：

onlyfun.caterpillar.welcome=哈啰

onlyfun.caterpillar.name=世界

然后使用JDK的工具程序native2ascii来转换。例如：

native2ascii –encoding GB2312 messages3\_zh\_cn.txt messages3\_zh\_cn.properties

转换后的messages3\_zh\_cn.properties文件内容会如下：

onlyfun.caterpillar.welcome=\u54c8\ufffd

onlyfun.caterpillar.name=\u4e16\u754c

将这个文件放于Classpath可以存取到的路径位置，也可以提供默认的信息文件messages3.properties

onlyfun.caterpillar.welcome=Hello

onlyfun.caterpillar.name=World

下面测试一下信息文件。我所使用的操作系统为语系设置是中文，区域设置是中国，来编写范例18.17进行测试。

**public** **class** I18NDemo {

**public** **static** **void** main(String[] args) {

ResourceBundle resource = ResourceBundle.*getBundle*("messages3"); System.*out*.print(resource.getString("onlyfun.caterpillar.welcome")+"!"); System.*out*.println(resource.getString("onlyfun.caterpillar.name")+"!");

}

}

根据操作系统的设置，执行程序时会使用默认的语系zh与区域设置中国，所以就会寻找messages3\_zh\_cn.properties的内容。结果会显示以下的信息：

哈啰!世界!

在使用ResourceBundle.*getBundle*()时可以给定Locale实例作为参数。例如若想提供messages\_en\_US.properties文件，并想要ResourceBundle.*getBundle*()取得这个文件的内容，则可以像下面这样编写：

Locale locale = **new** Locale("en","US");

ResourceBundle resource = ResourceBundle.*getBundle*("messages3",locale);

根据Locale对象的设置，这个程序片段将会取得messages3\_en\_US.properties文件中的信息内容。