[如何用ResourceBundle来读取配置文件](http://blog.csdn.net/haiyan0106/article/details/2257725)

对于java基础很好的人来说，这个应该是简单的不能再简单的了。不过估计一些ja va新手不一定会知道（比如我，上次为别人写一个东西，需要从外存读文件来设置对象的值，因为不知道有ResourceBundle这种东西，就自己用FileReader在那搞，折腾了半天，最后还不能把配置文件和jar包打在一起发布，郁闷）。

我们可以在每个对象中用ResourceBundle来读配置文件设置自己的值，也可以用一个固定的对象去读取然后保存下来以便以后使用。在每个class中都去读配置文件会导致代码散乱，所以，只用一个class来进行读取是比较合理的做法。另外，由于I/O的速度比较慢，如果负责读取配置文件的class是在每次用到这些配置项的时候去读文件，就容易成为性能上的瓶颈。为避免这样的情况，可以在初始化的时候就把配置项一次全部读入，并保存在静态成员变量中。不过不排除会有对配置项进行动态读取的需求（因为某些应用是不能停掉的，比如应用服务器。在这些应用运行期间更新了配置文件，就需要在不不关闭应用的情况下重新读入配置项）。以下的例子只考虑了静态读取的情况，如果是动态读取，则可以把读取配置文件的代码放到某个方法中，通过对某个事件的响应来调用该方法更新配置项。

假设我们用来读取配置文件的class叫TestResourceBundle，配置项的值来自一个叫property\_en.properties的文件（该文件应该放到TestResourceBundle所对应的CLASSPATH的目录），有两个值需要配置：name和value。首先，需要在该class中定义一些字符串常量，如下：

public static final String PROPERTIES\_FILE\_NAME = "property";  
public static final String MY\_NAME\_KEY = "name";  
public static final String MY\_VALUE\_KEY = "value";

其中PROPERTIES\_FILE\_NAME指出了文件的名字。实际读取的文件应该是property\_en.properties，但是只需要告诉ResourceBundle文件名是"property"就足够了。下划线和后面的"en"表示的是本地化信息。这里的en代表"ENGLISH"，后缀properties是默认的。MY\_NAME\_KEY和MY\_VALUE\_KEY表示配置项在配置文件中的名字，用ResourceBundle的getString方法根据这些名字去读取相应的值。

然后，定义需要配置的变量。这些变量应该是静态的：  
private static String myName;  
private static String myValue;

然后进行静态初始化：  
static {  
 try {  
   ResourceBundle bundle = ResourceBundle  
     .getBundle(PROPERTIES\_FILE\_NAME, Locale.ENGLISH);  
   myName = bundle.getString(MY\_NAME\_KEY).trim();  
   myValue = bundle.getString(MY\_VALUE\_KEY).trim();  
 }  
 catch(Exception ex) {  
   System.err.println(  "[Property]:Can't Load property.properties");  
   myName = "default name";  
   myValue = "default value";  
   System.out.println(  "myName will use the default value: " + myName);  
   System.out.println(  "myValue will use the default value: " + myValue);  
 }  
}  
 **ResourceBundle bundle = ResourceBundle  
  .getBundle(PROPERTIES\_FILE\_NAME, Locale.ENGLISH);**这行代码初始化了一个ResourceBundle，Locale.ENGLISH用于指明本地化情况，因此会从"property\_en.properties"中去读取配置项。如果是Locale.CHINA，则会从property\_zh.properties中读取。这种机制使得程序的本地化变得简单。  
 myName = bundle.getString(MY\_NAME\_KEY).trim();  
这行代码读入配置文件中名为"name"的变量的值，并赋给静态变量myName。  
此外这段代码还包含了例外处理，当读取失败的时候，配置项会使用缺省值。  
这样，该类就通过ResourceBundle读取外存上的配置文件对数据进行了配置。

property\_en.properties文件的内容如下  
# properties sample  
#  
name=sega  
value=100

其中以'#'开头的行为注释，ResourceBundle在遇到这些行的时候会忽略掉。

以下为完整的[**测试**](http://lib.csdn.net/base/softwaretest)代码：

public class TestResourceBundle {  
 public static final String PROPERTIES\_FILE\_NAME = "property";  
 public static final String MY\_NAME\_KEY = "name";  
 public static final String MY\_VALUE\_KEY = "value";  
   
 private static String myName;  
 private static String myValue;  
 static {  
   try {  
     ResourceBundle bundle = ResourceBundle  
  .getBundle(PROPERTIES\_FILE\_NAME, Locale.ENGLISH);  
     myName = bundle.getString(MY\_NAME\_KEY).trim();  
     myValue = bundle.getString(MY\_VALUE\_KEY).trim();  
   }  
   catch(Exception ex) {  
  System.err.println(    "[Property]:Can't Load property.properties");  
  myName = "default name";  
  myValue = "default value";  
  System.out.println(    "myName will use the default value: " + myName);  
  System.out.println(    "myValue will use the default value: " + myValue);  
   }  
 }  
   
 public void print() {  
  System.out.println("My name is: " + myName);  
  System.out.println("My value is: " + myValue);  
 }  
 public static void main(String[] args) {  
  TestResourceBundle test = new TestResourceBundle();  
  test.print();  
 }  
}

关于ResourceBundle更详细的信息，可以参考以下链接：  
http://ja va.sun.com/docs/books/tutorial/i18n/resbundle/index.html (English)  
[http://www.leftworld.net/online/j2sedoc/ja varef/ja va.util.resourcebundle\_dsc.htm](http://www.leftworld.net/online/j2sedoc/ja%20varef/ja%20va.util.resourcebundle_dsc.htm) (中文)