[**孤傲苍狼**](http://www.cnblogs.com/xdp-gacl/)

**只为成功找方法，不为失败找借口！**

[**javaweb学习总结(三十一)——国际化(i18n)**](http://www.cnblogs.com/xdp-gacl/p/3945800.html)

**一、国际化开发概述**

　　软件的国际化：软件开发时，要使它能同时应对世界不同地区和国家的访问，并针对不同地区和国家的访问，提供相应的、符合来访者阅读习惯的页面或数据。  
　　**国际化(internationalization)又称为 i18n**(读法为i 18 n，据说是因为internationalization(国际化)这个单词从i到n之间有18个英文字母，i18n的名字由此而来)

**二、合格的国际化软件**

　　软件实现国际化，需具备以下两个特征：  
　　1、**对于程序中固定使用的文本元素**，例如菜单栏、导航条等中使用的文本元素、或错误提示信息，状态信息等，**需要根据来访者的地区和国家，选择不同语言的文本为之服务**。  
　　2、**对于程序动态产生的数据**，例如(日期，货币等)，**软件应能根据当前所在的国家或地区的文化习惯进行显示**。

**三、固定文本元素的国际化**

　　对于软件中的菜单栏、导航条、错误提示信息，状态信息等这些固定不变的文本信息，可以把它们写在一个properties文件中，并根据不同的国家编写不同的properties文件。这**一组properties文件称之为一个资源包**。

**3.1、创建资源包和资源文件**

**一个资源包中的每个资源文件都必须拥有共同的基名。除了基名，每个资源文件的名称中还必须有标识其本地信息的附加部分**。例如：一个资源包的基名是“myproperties”，则与中文、英文环境相对应的资源文件名则为: "myproperties\_zh.properties"  "myproperties\_en.properties"

　　每个资源包都应有一个默认资源文件，这个文件不带有标识本地信息的附加部分。若ResourceBundle对象在资源包中找不到与用户匹配的资源文件，它将选择该资源包中与用户最相近的资源文件，如果再找不到，则使用默认资源文件。例如：myproperties.properties

**3.2、资源文件的书写格式**

**资源文件的内容通常采用"关键字＝值"的形式，软件根据关键字检索值显示在页面上**。**一个资源包中的所有资源文件的关键字必须相同，值则为相应国家的文字**。  
并且资源文件中采用的是**properties**格式文件，所以**文件中的所有字符都必须是ASCII字码，属性(properties)文件是不能保存中文的，对于像中文这样的非ACSII字符，须先进行编码。**

例如：

　　国际化的中文环境的properties文件

　　国际化的英文环境的properties文件

　　java提供了一个native2ascII工具用于将中文字符进行编码处理，native2ascII的用法如下所示：

**3.3、编程实现固定文本的国际化**

　　在JavaAPI中提供了一个ResourceBundle 类用于描述一个资源包，并且 ResourceBundle类提供了相应的方法getBundle，这个方法可以根据来访者的国家地区自动获取与之对应的资源文件予以显示。

　　ResourceBundle类提供了一个静态方法getBundle，该方法用于装载资源文件，并创建ResourceBundle实例：

1 Locale currentLocale = Locale.getDefault();

2 ResourceBundle myResources =ResourceBundle.getBundle(basename, currentLocale);

　　basename为资源包基名(且必须为完整路径)。  
　　如果与该locale对象匹配的资源包子类找不到。一般情况下，则选用默认资源文件予以显示。   
　　加载资源文件后， 程序就可以调用ResourceBundle 实例对象的 getString 方法获取指定的资源信息名称所对应的值。

1 String value = myResources.getString(“key");

范例：**根据国家地区自动获取与之对应的资源文件**

[复制代码](javascript:void(0);)

1 package me.gacl.i18n;

2

3 import java.util.Locale;

4 import java.util.ResourceBundle;

5 /\*\*

6 \* @ClassName: I18NTest

7 \* @Description: 编程实现固定文本的国际化

8 \* @author: 孤傲苍狼

9 \* @date: 2014-8-29 下午9:34:05

10 \*

11 \*/

12 public class I18NTest {

13

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

15 //资源包基名(包名+myproperties)

16 String basename = "me.gacl.i18n.resource.myproperties";

17 //设置语言环境

18 Locale cn = Locale.CHINA;//中文

19 Locale us = Locale.US;//英文

20 //根据基名和语言环境加载对应的语言资源文件

21 ResourceBundle myResourcesCN = ResourceBundle.getBundle(basename,cn);//加载myproperties\_zh.properties

22 ResourceBundle myResourcesUS = ResourceBundle.getBundle(basename,us);//加载myproperties\_en.properties

23

24 //加载资源文件后， 程序就可以调用ResourceBundle实例对象的 getString方法获取指定的资源信息名称所对应的值。

25 //String value = myResources.getString(“key");

26 String usernameCN = myResourcesCN.getString("username");

27 String passwordCN = myResourcesCN.getString("password");

28

29 String usernameUS = myResourcesUS.getString("username");

30 String passwordUS = myResourcesUS.getString("password");

31

32 System.out.println(usernameCN+"--"+passwordCN);

33 System.out.println(usernameUS+"--"+passwordUS);

34 }

35 }

[复制代码](javascript:void(0);)

　　运行结果：

**3.4、在WEB应用中实现固定文本的国际化**

如下所示：

[复制代码](javascript:void(0);)

1 <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

2 <!DOCTYPE HTML>

3 <html>

4 <head>

5 <title>国际化(i18n)测试</title>

6 </head>

7 <%

8 //加载i18n资源文件，request.getLocale()获取访问用户所在的国家地区

9 ResourceBundle myResourcesBundle = ResourceBundle.getBundle("me.gacl.i18n.resource.myproperties",request.getLocale());

10 %>

11 <body>

12 <form action="" method="post">

13 <%=myResourcesBundle.getString("username")%>：<input type="text" name="username"/><br/>

14 <%=myResourcesBundle.getString("password")%>：<input type="password" name="password"/><br/>

15 <input type="submit" value="<%=myResourcesBundle.getString("submit")%>">

16 </form>

17 </body>

18 </html>

[复制代码](javascript:void(0);)

运行结果：

　　浏览器语言是中文环境下的显示效果：

　　浏览器语言是英文环境下的显示效果：

　　同样一个页面，在不同语言环境的浏览器下显示出了不同的语言文字效果，这样就实现了固定文本的国际化。

　　IE浏览器切换使用语言：工具→Internet选项

**四、动态数据的国际化**

　　数值，货币，时间，日期等数据由于可能在程序运行时动态产生，所以无法像文字一样简单地将它们从应用程序中分离出来，而是需要特殊处理。Java 中提供了解决这些问题的 API 类(位于 java.util 包和 java.text 包中)

**4.1、Locale 类**

　　Locale 实例对象代表一个特定的地理，政治、文化区域。  
　　一个 Locale 对象本身不会验证它代表的语言和国家地区信息是否正确，只是向本地敏感的类提供国家地区信息，与国际化相关的格式化和解析任务由本地敏感的类去完成。(**若JDK中的某个类在运行时需要根据 Locale 对象来调整其功能，这个类就称为本地敏感类**)

**4.2、DateFormat类(日期格式化)**

　　DateFormat 类可以将一个日期/时间对象格式化为表示某个国家地区的日期/时间字符串。  
　　DateFormat 类除了可按国家地区格式化输出日期外，它还定义了一些用于描述日期/时间的显示模式的 int 型的常量，包括FULL, LONG, MEDIUM, DEFAULT, SHORT，实例化DateFormat对象时，可以使用这些常量，控制日期/时间的显示长度。

**4.2.1、实例化DateFormat类**

　　实例化DateFormat类有九种方式，以下三种为带参形式，下面列出的三种方式也可以分别不带参，或只带显示样式的参数。  
　　getDateInstance(int style, Locale aLocale)：以指定的日期显示模式和本地信息来获得DateFormat实例对象，该实例对象不处理时间值部分。  
　　getTimeInstance(int style, Locale aLocale)：以指定的时间显示模式和本地信息来获得DateFormat实例对象，该实例对象不处理日期值部分。  
　　getDateTimeInstance(int dateStyle, int timeStyle, Locale aLocale)：以单独指定的日期显示模式、时间显示模式和本地信息来获得DateFormat实例对象。

**4.2.2、DateFormat 对象的方法**

　　format：将date对象格式化为符合某个本地环境习惯的字符串。  
　　parse：将字符串解析为日期/时间对象  
　　注意：parse和format完全相反，一个是把date时间转化为相应地区和国家的显示样式，一个是把相应地区的时间日期转化成date对象，该方法在使用时，解析的时间或日期要符合指定的国家、地区格式，否则会抛异常。  
　　DateFormat 对象通常不是线程安全的，每个线程都应该创建自己的 DateFormat  实例对象

**4.2.3、DateFormat使用范例**

[复制代码](javascript:void(0);)

1 package me.gacl.i18n;

2

3 import java.text.DateFormat;

4 import java.text.ParseException;

5 import java.util.Date;

6 import java.util.Locale;

7

8 /\*\*

9 \* @ClassName: DateFormatTest

10 \* @Description: DateFormat类测试

11 \* DateFormat类可以将一个日期/时间对象格式化为表示某个国家地区的日期/时间字符串

12 \* @author: 孤傲苍狼

13 \* @date: 2014-8-29 下午10:03:26

14 \*

15 \*/

16 public class DateFormatTest {

17

18 public static void main(String[] args) throws ParseException {

19 Date date = new Date(); // 当前这一刻的时间（日期、时间）

20

21 // 输出日期部分

22 DateFormat df = DateFormat.getDateInstance(DateFormat.FULL,Locale.GERMAN);

23 String result = df.format(date);

24 System.out.println(result);

25

26 // 输出时间部分

27 df = DateFormat.getTimeInstance(DateFormat.FULL, Locale.CHINA);

28 result = df.format(date);

29 System.out.println(result);

30

31 // 输出日期和时间

32 df = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.LONG,Locale.CHINA);

33 result = df.format(date);

34 System.out.println(result);

35

36 // 把字符串反向解析成一个date对象

37 String s = "10-9-26 下午02时49分53秒";

38 df = DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.LONG,Locale.CHINA);

39 Date d = df.parse(s);

40 System.out.println(d);

41 }

42 }

[复制代码](javascript:void(0);)

**4.3、NumberFormat类(数字格式化)**

　　NumberFormat类可以将一个数值格式化为符合某个国家地区习惯的数值字符串，也可以将符合某个国家地区习惯的数值字符串解析为对应的数值  
　　NumberFormat类的方法：  
　　　　format 方法：将一个数值格式化为符合某个国家地区习惯的数值字符串  
　　　　parse 方法：将符合某个国家地区习惯的数值字符串解析为对应的数值。  
　　实例化NumberFormat类时，可以使用locale对象作为参数，也可以不使用，下面列出的是使用参数的。

1. getNumberInstance(Locale locale)：以参数locale对象所标识的本地信息来获得具有多种用途的NumberFormat实例对象
2. getIntegerInstance(Locale locale)：以参数locale对象所标识的本地信息来获得处理整数的NumberFormat实例对象
3. getCurrencyInstance(Locale locale)：以参数locale对象所标识的本地信息来获得处理货币的NumberFormat实例对象
4. getPercentInstance(Locale locale)：以参数locale对象所标识的本地信息来获得处理百分比数值的NumberFormat实例对象

范例：

[复制代码](javascript:void(0);)

1 package me.gacl.i18n;

2

3 import java.text.NumberFormat;

4 import java.text.ParseException;

5 import java.util.Locale;

6

7 /\*\*

8 \* @ClassName: NumberFormatTest

9 \* @Description: NumberFormat类测试

10 \* @author: 孤傲苍狼

11 \* @date: 2014-8-29 下午10:25:29

12 \*

13 \*/

14 public class NumberFormatTest {

15

16 public static void main(String[] args) throws ParseException {

17 int price = 89;

18

19 NumberFormat nf = NumberFormat.getCurrencyInstance(Locale.CHINA);

20 String result = nf.format(price);

21 System.out.println(result);

22

23 String s = "￥89.00";

24 nf = NumberFormat.getCurrencyInstance(Locale.CHINA);

25 Number n = nf.parse(s);

26 System.out.println(n.doubleValue() + 1);

27

28 double num = 0.5;

29 nf = NumberFormat.getPercentInstance();

30 System.out.println(nf.format(num));

31 }

32 }

[复制代码](javascript:void(0);)

　　运行结果：

**4.4、MessageFormat(文本格式化)**

　　如果一个字符串中包含了多个与国际化相关的数据，可以使用MessageFormat类对这些数据进行批量处理。  
　　例如：At 12:30 pm on jul 3,1998, a hurricance destroyed 99 houses and caused $1000000 of damage  
　　以上字符串中包含了时间、数字、货币等多个与国际化相关的数据，对于这种字符串，可以使用MessageFormat类对其国际化相关的数据进行批量处理。  
　　MessageFormat 类如何进行批量处理呢？  
　　　　1.MessageFormat类允许开发人员用占位符替换掉字符串中的敏感数据（即国际化相关的数据）。  
　　　　2.MessageFormat类在格式化输出包含占位符的文本时，messageFormat类可以接收一个参数数组，以替换文本中的每一个占位符。

**4.4.1、模式字符串与占位符**

模式字符串：

　　At {0} on {1},a destroyed {2} houses and caused {3} of damage

　　字符串中的{0}、{1}、{2}、{3}就是占位符

**4.4.2、格式化模式字符串**

　　1、实例化MessageFormat对象，并装载相应的模式字符串。

　　2、使用format(object obj[])格式化输出模式字符串，参数数组中指定占位符相应的替换对象。

范例：

[复制代码](javascript:void(0);)

1 package me.gacl.i18n;

2

3 import java.text.MessageFormat;

4 import java.util.Date;

5 import java.util.Locale;

6

7 /\*\*

8 \* @ClassName: MessageFormatTest

9 \* @Description: MessageFormat类测试

10 \* @author: 孤傲苍狼

11 \* @date: 2014-8-29 下午10:29:19

12 \*

13 \*/

14 public class MessageFormatTest {

15

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

17 //模式字符串

18 String pattern = "On {0}, a hurricance destroyed {1} houses and caused {2} of damage.";

19 //实例化MessageFormat对象，并装载相应的模式字符串

20 MessageFormat format = new MessageFormat(pattern, Locale.CHINA);

21 Object arr[] = {new Date(), 99, 100000000};

22 //格式化模式字符串，参数数组中指定占位符相应的替换对象

23 String result = format.format(arr);

24 System.out.println(result);

25 }

26 }

[复制代码](javascript:void(0);)

运行结果：

**4.4.3、占位符的三种书写方式**

　　{argumentIndex}: 0-9 之间的数字，表示要格式化对象数据在参数数组中的索引号  
　　{argumentIndex,formatType}: 参数的格式化类型  
　　{argumentIndex,formatType,FormatStyle}: 格式化的样式，它的值必须是与格式化类型相匹配的合法模式、或表示合法模式的字符串。

范例：

[复制代码](javascript:void(0);)

1 package me.gacl.i18n;

2

3 import java.text.MessageFormat;

4 import java.util.Date;

5 import java.util.Locale;

6

7 /\*\*

8 \* @ClassName: MessageFormatTest

9 \* @Description: MessageFormat类测试

10 \* @author: 孤傲苍狼

11 \* @date: 2014-8-29 下午10:29:19

12 \*

13 \*/

14 public class MessageFormatTest {

15

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

17 //模式字符串

18 String pattern = "At {0, time, short} on {0, date}, a destroyed {1} houses and caused {2, number, currency} of damage.";

19 //实例化MessageFormat对象，并装载相应的模式字符串

20 MessageFormat format = new MessageFormat(pattern, Locale.US);

21 Object arr[] = {new Date(), 99, 100000000};

22 //格式化模式字符串，参数数组中指定占位符相应的替换对象

23 String result = format.format(arr);

24 System.out.println(result);

25 }

26 }

[复制代码](javascript:void(0);)

运行结果：

**五、在WEB应用中使用国际化标签库实现固定文本的国际化**

[复制代码](javascript:void(0);)

1 <%@ page language="java" import="java.util.\*" pageEncoding="UTF-8"%>

2 <%--导入国际化标签库 --%>

3 <%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

4 <!DOCTYPE HTML>

5 <html>

6 <head>

7 <title>国际化(i18n)测试</title>

8 </head>

9 <%--

10 //加载i18n资源文件，request.getLocale()获取访问用户所在的国家地区

11 ResourceBundle myResourcesBundle = ResourceBundle.getBundle("me.gacl.i18n.resource.myproperties",request.getLocale());

12 --%>

13 <body>

14 <%--

15 <form action="" method="post">

16 <%=myResourcesBundle.getString("username")%>：<input type="text" name="username"/><br/>

17 <%=myResourcesBundle.getString("password")%>：<input type="password" name="password"/><br/>

18 <input type="submit" value="<%=myResourcesBundle.getString("submit")%>">

19 </form>

20 --%>

21

22 <fmt:setBundle var="bundle" basename="me.gacl.i18n.resource.myproperties" scope="page"/>

23 <form action="">

24 <fmt:message key="username" bundle="${bundle}"/><input type="text" name="username"><br/>

25 <fmt:message key="password" bundle="${bundle}"/><input type="password" name="password"><br/>

26 <input type="submit" value="<fmt:message key="submit" bundle="${bundle}"/>">

27 </form>

28 </body>

29 </html>

[复制代码](javascript:void(0);)

 　　以上就是JavaWeb开发中国际化的总结内容。