Строки - детальнее

Интернационализация приложения

Класс **java.util.Locale** позволяет учесть особенности региональных представлений алфавита, символов, чисел, дат и проч. Автоматически виртуальная машина использует текущие региональные установки операционной системы, но при необходимости их можно изменять. Для некоторых стран региональные параметры устанавливаются с помощью констант, например: **Locale.US**, **Locale.FRANCE**. Для всех остальных объект **Locale** нужно создавать с помощью конструктора, например:

```
Locale rus = new Locale("ru", "RU");
```

Определить текущий вариант региональных параметров можно следующим образом:

```
Locale current = Locale.getDefault();
```

А можно и изменить для текущего экземпляра (instance) JVM:

```
Locale.setDefault(Locale.CANADA);
```

ResourceBundle

Класс **ResourceBundle** предназначен для взаимодействия с текстовыми файлами свойств (**расширение.properties**).

Например, файл **text_it_CH.properties** соответствует объекту Locale, заданному кодом итальянского языка (it) и кодом страны Швейцарии (CH).

```
text.properties
text_ru_RU.properties
text_it_CH.properties
text_fr_CA.properties
```

В файлах свойств информация должна быть организована по принципу:

```
#Комментарий
group1.key1 = value1
group1.key2 = value2
group2.key1 = value3…
```

Например:

```
label.button = submit
label.field = login
message.welcome = Welcome!
```

ResourceBundle

```
Locale current = new Locale(language, country);
ResourceBundle rb = ResourceBundle.getBundle("property.text", current);
          String s1 = rb.getString("str1");
          System.out.println(s1);

          String s2 = rb.getString("str2");
          System.out.println(s2);
    }
```

Все файлы следует разместить в каталоге **property** в корне проекта на одном уровне с пакетами приложения.

Файл **text_en_US.properties** содержит следующую информацию:

str1 = To be or not to be?
str2 = This is a question.

Файл **text_be_BY.properties**:

str1 = Быць або не быць?
str2 = Вось у чым пытанне.

Файл **text.properties**:

str1 = Быть или не быть?
str2 = Вот в чём вопрос.

Интернационализация чисел

Стандарты представления дат и чисел в различных странах могут существенно отличаться. Например, в **Германии** строка **«1.234,567»** воспринимается как **«одна тысяча двести тридцать четыре целых пятьсот шестьдесят семь тысячных»**, **для русских и французов** данная строка просто **непонятна** и не может представлять число. Чтобы сделать такую информацию конвертируемой в различные региональные стандарты, применяются возможности класса **java.text.NumberFormat.**

1
```
NumberFormat nf = NumberFormat.getInstance(new Locale("RU"));
```

или

```
NumberFormat.getInstance();
```

2 Далее для преобразования строки в число и обратно используются методы **Number parse(String source)** и **String format(double number)** соответственно.

Пример,

```
NumberFormat nfGe = NumberFormat.getInstance(Locale.GERMAN);
String str = "1.234,5"; double iGe = 0;
// преобразование строки в германский стандарт
iGe = nfGe.parse(str).doubleValue();
```

Интернационализация дат

Учитывая исторически сложившиеся способы отображения даты и времени в различных странах и регионах мира, в языке создан механизм поддержки всех национальных особенностей. Эту задачу решает класс **java.text.DateFormat**

Процесс получения объекта, отвечающего за обработку регионального стандарта даты, похож на создание объекта, отвечающего за национальные представления чисел, а именно:

```
DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM, new Locale("BY"));
```

или по умолчанию:

```
DateFormat.getDateInstance();
```

Константа **DateFormat.MEDIUM** указывает на то, что будут представлены только дата и время без указания часового пояса. Для указания часового пояса используются константы класса **DateFormat** со значением **LONG** и **FULL**. Константа **SHORT** применяется для сокращенной записи даты, где месяц представлен в виде своего порядкового номера.

Для получения даты в виде строки для заданного региона используется метод **String format(Date date)** в виде:

```
String s = df.format(new Date());
```

Интернационализация дат

Метод **Date parse(String source)** преобразовывает переданную в виде строки дату в объектное представление конкретного регионального формата, например:

```
String str = "April 9, 2012";
Date d = df.parse(str);

DateFormat df = DateFormat.getDateInstance(DateFormat.MEDIUM, Locale.US);
Date d = null;
String str = "April 9, 2012";
   try {
          d = df.parse(str);
          System.out.println(d);
   } catch (ParseException e) {
          System.err.print("Error position: " + e.getErrorOffset());
   }
   df = DateFormat.getDateInstance(DateFormat.LONG, new Locale("ru","RU"));
   System.out.println(df.format(d));

   df = DateFormat.getDateInstance(DateFormat.FULL,Locale.GERMAN);
   System.out.println(df.format(d));
```

Результат работы программы:

Mon Apr 9 00:00:00 EEST 2012
9 Апрель 2012 г.
Montag, 9. April 2012

Статья