Java程序设计中对 Unicode 的支持

王莺,彭侠夫,何金花
(厦门大学 信息科学与技术学院,福建 厦门 361005)

摘要:在信息国际化的今天,正确处理不同的语言文字已经成为绝大多数软件系统的必备功能,该文介绍了在 Java 程序设计中如何处理和显示不同的语言文字,尤其是非英文字符的处理。本文从基本的编码知识入手,详细介绍了 Java 语言中编码转换的技术,并给出了可以用于实际开发的实例代码,为构建国际化的 Java 软件系统打下良好的基础。

关键词:Java 程序设计;Unicode 字符集;编码/解码
中图分类号:TP391　　**文献标识码**:A　　**文章编号**:1009-3044(2007)03-10773-01

Unicode Charset Support in Java Programming

WANG Ying, PENG Xia-fu, HE Jin-hua
(Information and Science College, Xiamen University, Xiamen 361005, China)

Abstract: Currently, Multi languages support is a key feature of software product. It is necessary to process and display the characters in different languages correctly. This article give a brief introduction of encoding first, then show you how to handle the encoding/decoding in Java, provide some demo codes could be used in development at last. All of these make contribution on building software with international, multi languages support.

Key words: Java programming Unicode charset encoding/decoding

1 引言

在基于 Java 程序设计中,我们经常碰到非英文字符处理及显示的问题。一大堆看不懂的乱码肯定不是任何人所希望的结果,怎样才能够让那些基于特定字符集编码的非英文字符正确显示呢?怎样才能够恰当地选择字符集编码方式并正确地处理?本文从基本的字符集知识出发,通过对 Java 程序设计中有关编码转换的问题进行的分析并结合编程实例,详细介绍 Java 的多语言支持(Unicode 编码)特性。

2 关于字符集的知识

正如大家所知,计算机的信息处理建立在二进制之上,可以这么说,计算机只理解 0/1,为了便于阅读和理解,需要将这些 0/1 的串映射为人可以阅读的字符,这些映射称之为字符集;而字符在字符集中的编码称为字符编码[4]。例如“A”在 ASCII 字符集中的编码为“0x41”。早期的计算机系统使用了 7 位二进制的 ASCII 编码作为缺省的编码方式,于是,在计算机中一切处理程序最初都是以单字节编码为准进行处理。对于英文来说,ASCII 码 0-127 就足以代码所有字符,对于其它语言,如中文,日文等,则必须使用两个字节 (byte) 来代表一个字符,在习惯上称为双字节(即 DBCS: Double-Byte Character Set),相对的,英文的字符编码就称为单字节 SBCS(Single-Byte Character Set)。按照这种使用双字节来编码的思路,发展出了简体中文的 GB2312,繁体中文的 BIG5 和日文的 SJIS 等针对特定语言的字符集。当不同语言的字符混合在一起的时候,必须经过字符集的转换,非常麻烦。例如:中英文混合情况。为解决这个问题,国际标准组织提出了容纳全世界所有语言文字的 Unicode 字符集。

Unicode 的学名是“Universal Multiple-Octet Coded Character Set”,简称为 UCS[2]。UCS 可以看作是“Unicode Character Set”的缩写。UCS 有两种格式:UCS-2 和 UCS-4。顾名思义,UCS-2 就是占用双字节的 Unicode 编码,UCS-4 就是占用 4 个字节的 Unicode 编码[4]。本文中所有的内容均以 UCS-2 为准。

在 Unicode 出现之前,许多的信息系统和数据都是基于特定字符集编码开发的,例如,大量的中文电子文档,企业遗留的财务系统,服务部门的计费系统等等。当需要对旧的信息系统和数据进行升级时,系统对多语言支持往往是很重要的部分,在信息国际化的大环境中非常必要的。但很多开发人员没有认识到 Unicode 编码是多语言支持的核心,或者不熟悉 Java 语言中正确的编码转换方法,经常写出这样的代码:

```
new String(source.getBytes("GB2312"), "ISO8859-1")
new String(source.getBytes("ISO8859-1"), "GB2312")
```

这样的代码存在如下的问题:(1)可读性差,不便于维护;(2)将系统和 GB2312 字符集紧紧绑在一起,多语言支持难以实现;(3)每次都要进行 new 操作,频繁的产生不必要的对象,消耗系统资源。

事实上,Java 语言在设计之初,就考虑到对多语言的支持,默认在 Java 程序内部使用 Unicode 字符集,在本文接下来的部分就 Java 语言的这一特性进行详细的介绍。

3 Java 程序设计中的编码转换

在 Java 程序内部,所有的字符都是按照 Unicode 字符集来编码,所以在 Java 程序运行时,就存在由一个本地字符集(与特定语言相关的操作系统)编码向 Java 内部 Unicode 字符集编码的转换过程,称为解码(Decoder);同样的,由 Java 程序内部输出信息,是从 Unicode 字符集编码向本地字符集转换的过程就是编码(Encoder)[1]。

Java 程序中保证非英文字符能够被正确显示的条件只有一个:解码过程和编码过程使用相同或者兼容的字符集编码。

从 JDK1.4 起,Java 提供了 java.nio.charset 包,其中有三个类用于解码和编码:Charset、CharsetEncoder 和 CharsetDecoder 来帮助我们完成编码转换[3],下面我们就用这三个类来演示 Java 程序设计中的编码转换。

在 Java 程序设计中默认支持下列字符集编码[3]:

US-ASCII: 7 位 ASCII;
ISO-8859-1: ISO 拉丁字母;
UTF-8: 8 位 UCS 转换格式;
UTF-16BE: 16 位 UCS 转换格式,大尾数法字节顺序;
UTF-16LE: 16 位 UCS 转换格式,小尾数法字节顺序;
UTF-16: 16 位 UCS 转换格式,用标记(marker)识别的字节顺序。

然后,不同的平台可能支持特定于该平台的额外字符集(例如,在 Windows 平台上,您会发现它支持 Windows-1252 字符集)。如果您需要支持其他的字符集,您可以创建自己的字符集。请参阅 java.nio.charset.spi 包中的 CharsetProvider API[1]。

(下转第 847 页)

收稿日期:2006-11-01

作者简介:王莺(1979-),女,内蒙人,硕士研究生,主要研究方向:虚拟现实技术及计算机信息系统集成;彭侠夫(1963-),男,江西人,教授,博士生导师,主要研究方向:船舶运动、控制理论、控制工程等;何金花(1981-),女,福建人,硕士研究生,主要研究方向:虚拟现实、视景仿真等。

实践课比例,兼顾文化休养和专业提高设置选修课等措施,着眼于学生的素质和能力培养,以科学合理地制(修)适合于高职高专计算机专业的教学计划。

参考文献:

[1] 谭浩强. 高等学校计算机基础教育改革的新阶段[J]. 计算机教育,2003.11.

[2] 杨近. 高职教育软件人才培养中存在的问题与对策[J]. 中国高教研究, 2006.6.

[3] 杨建立,杨京楼. 高职院校教学改革的必要性及其实践探讨[J]. 教育与职业, 2006.9.

[4] 王冬. 以教师的视角探究两年制高职教学改革[J]. 教育与职业,2006.5.

[5] 李晓明,陈平,张铭,朱敏悦.关于计算机人才需求的调研报告[J]. 计算机教育, 2004(08).

[6] 陈道蓄. 计算机专业教育改革[R]. 江苏省信息学科首届高峰论坛, 2006.7.

[7] 方锦明,朱闻亚. 高职计算机专业俱乐部模式的实践探索[J]. 江西:职教论坛, 2006.4(教研).

[8] 翟玉庆. 计算机学科研究型优秀人才培养模式探索与实践[R]. 江苏省信息学科首届高峰论坛, 2006.7.

[9] 教育部计算机科学与技术专业教学指导分委员会.中国计算机本科专业发展战略研究报告[J]. 中国大学教学,2005.5.

[10] 周远清. 我国高等教育的改革与发展[J]. 中国大学生就业,2005.22.

[11] 专业教学方案设计与实施建议.
http://www.yd56.com.cn/news/default-61.htm 2006.8.24.

[12] 教学计划编制说明.
http://www.fesky.com.cn/dzcm/xkjs/bianz.htm,2006.8.24.

(上接第 773 页)

在得到一个解码器或编码器之前,需要获得用于特定字符集的 Charset。例如,GB2312 是用于 GB2312 字符集的名称。您只需象下面这样把该名称传递到 Charset 的 forName() 方法中即可:

```
Charset charset = Charset.forName("GB2312");
```

一旦有了 Charset, 只需按如下所示请求 CharsetDecoder 和 CharsetEncoder:

```
CharsetDecoder decoder = charset.newDecoder();
CharsetEncoder encoder = charset.newEncoder();
```

有了解码器和编码器后,您就可以在特定字符集编码和 Unicode 字符集合之间进行转换了,如下所示:

```
ByteBuffer bytes = …;
CharBuffer chars = decoder.decode(bytes);
bytes = encoder.encode(chars);
```

以字符‘中’为例,可通过以下方式得到其在 GB2312 字符集中编码为“D6D0”。这是一个编码过程。

```
String str = "中";
String CHARSET = "GB2312";
char nativeChars[] = str.toCharArray();
Charset nativeCharset = Charset.forName(CHARSET);
CharBuffer nativeCharBuffer = CharBuffer.wrap(nativeChars);
CharsetEncoder encoder = nativeCharset.newEncoder();
ByteBuffer nativeBytebuffer = encoder.encode(nativeCharBuffer);
byte[] nativeBytes = nativeBytebuffer.array();
System.out.println("\n#----- " + CHARSET + "encoding output -----#");
for (int i = 0; i < nativeBytes.length; i++)
{System.out.print(Integer.toHexString(' \u00FF' & nativeBytes[i]).toUpperCase());}
```

GB2312 字符集编码的输出结果:

```
#----- GB2312 encoding output -----#
D6D0
```

而后,我们将字符‘中’的 GB2312 字符集编码转换为 Unicode 字符集编码,这是一个编码过程。

```
CharsetDecoder unicodeDecoder = nativeCharset.newDecoder();
CharBuffer unicodeCharbuffer = unicodeDecoder.decode(nativeBytebuffer);
char unicodeChars[] = unicodeCharbuffer.array();
System.out.println ("\n#----- Unicode encoding output -----#");
for (int i = 0; i < unicodeChars.length; i++)
{System.out.print (Integer.toHexString (unicodeChars [i]).toUpperCase());}
System.out.println("\n" + String.valueOf(unicodeChars));
```

Unicode 字符集编码的输出结果:

```
#----- Unicode encoding output -----#
4E2D
中
```

字符‘中’的 Unicode 字符集编码为“4E2D”,并可以被正确的显示。

类似的,我们可以获得字符‘中’在日文字符集 SHIFT-JIS 中的编码为“9286”,从 SHIFT-JIS 字符集向 Unicode 字符集编码转换的结果仍为“4E2D”。

```
String str = "中";
String CHARSET = "SHIFT-JIS";
char nativeChars[] = str.toCharArray();
… …
```

输出结果:

```
#----- SHIFT-JIS encoding output -----#
9286
#----- Unicode encoding output -----#
4E2D
中
```

通过以上的程序实例,可以得出结论:无论字符原来用何种本地字符集表示,在 Unicode 字符集中都被表示成相同的编码。或者说,Unicode 字符集和语言的种类无关。

可以推断出, 无论输入的数据最初由何种字符集编码表示,只要在进入 Java 程序后,进行 Unicode 字符集编码转换,并按照 Unicode 字符集编码输出,都可以被正确的显示。Unicode 字符集为不同语言中的所有字符都提供了唯一的编码。

那么,我们现在就可以回答本文开篇提出的两个问题。

第一, 借助于 java.nio.charset 包中的 Charset、CharsetEncoder 和 CharsetDecoder 将特定字符集编码的字符转换为 Unicode 字符集编码;

第二,选择 Unicode 字符集编码将字符输出。

做到了以上两点,处理任何语言的字符都不会遇到类似于乱码的显示问题。

4 结束语

本文从 Unicode 字符集编码角度分析了 Java 的多语言支持特性,并对由字符集引起的显示问题进行了分析和解答,相信大家能够更好的理解和运用这一技术,从长远角度出发,设计,实现符合国际化规范的应用系统。

参考文献:

[1](美)Bruce Eckel. Java 编程思想 [M]. 机械工业出版社, 2002.9.P170-174.

[2](美)The Unicode Consortium. The Unicode Standard, Version 4.0[M]. Addison-Wesley 出版社,1996.

[3]Unicode 官方网站. http://www.unicode.org/.

[4]Java API 文档. http://java.sun.com/.