DOI: 10.13657/j.cnki.gxkxyxb.2005.s1.038

Java/JSP编译器对汉语编码的处理方式
Treatment of Java/JSP Compiler for Chinese Coding

李睿妍,归伟夏
Li Ruiyan, Gui Weixia

(广西大学计算机与电子信息学院,广西南宁 530004)
(School of Comp., Elec. and Info., Guangxi Univ., Nanning, Guangxi, 530004, China)

摘要:分析 Java语言在英文字符和 GB2312码、字节和 Unicode、UTF-8、String和 byte[]等编码方面存在的问题,从 Java编译器和 JSP编译器两个方面提出 Java编程中解决汉语编码的处理方式。
关键词:编译器 Java JSP 编码 处理方式
中图法分类号:TP391.2 文献标识码:A 文章编号:1002-7378(2005)S0-0114-04

Abstract This paper analyses how to use Java language to resolve the coding problem in English character, GB2312, bytes, Unicode, UTF-8, String and byte[]. A new technique to resolve the treatment of Chinese coding in Java programming with Java compiler and JSP compiler is proposed.
Key words compiler, Java, JSP, coding, solving method

网络通讯技术的飞速发展,使得企业仅购买一套基于 Web的企业级应用软件,通过网络就能满足集团公司的需要。但由于应用系统面向多个国家,为多种语言界面,所以在设计软件时就不得不解决国际化问题,如支持多国语言显示,提供表示不同格式的日期、时间、货币和其它值,为不同国家企业定制所需的个性化界面,等等。Java语言是一种面向对象的语言,可以用于生成一个国际化的软件包,系统可以根据区域及语言自动设置相关的属性[1]。本文主要针对 Java编程中汉语编码方式存在的问题,提出相应的解决方法。

1 Java编程中存在的问题

1.1 英文字符和 GB2312码[2]

英文字符的编码方法是 ASCII码,一个字节最多只能区分 256个字符,而汉字几万个,所以通常用双字节来表示汉字,为了能够与英文字符区分,每个字节的最高位一定为 1 在 ISO8859-1(非压缩的拉丁英文字符串编码)中,每个字节的最高位为 0 在 GB2312中,每个字节的最高位为 1 GB2312码的 2个字节中,第一个字节(高字节)的值为 32(20H),第二个字节(低字节)的值为位号值加 32(20H),用这两个值来表示一个汉字的编码,可在 GB平台上使用。

1.2 字节和 Unicode码[2]

Unicode码是微软公司提出的解决多国字符问题的多字节等长编码,它对英文字符采取前面加“0”字节的策略实现等长兼容。如“A”的 ASCII码为 0x41, Unicode就为 0x00, 0x41 各种编码之间可以利用特殊的工具互相转换。Java内核是使用 Unicode码,class文件也一样,但是很多媒体,包括文件 流的保存方式使用字节流。因此 Java要对这些字节流进行转化。在 Java中的 sun.io包里有 byte/char互换函数,其中 ByteToCharConverter类起调度作用,包含有两个很常用的静态函数:

```
public static ByteToCharConverter getDefault();
public static ByteToCharConverter getConverter(String encoding);
```

如果不指定 converter,则系统会自动使用当前的 Encoding,其中 GB平台上用 GBK, EN平台上用 ISO8859-1编码。

下面的例子说明:汉字的“你”的 GB码是: 0xC4E3,而 Unicode码是 0x4F60[1]。

```
……
encoding = "GB2312";
byte b[] = {(byte)'\u00C4', (byte)'\
```

收稿日期:2005-08-23
作者简介:李睿妍(1983-),女,广西梧州人,本科生。

```
u00E3 };
        convertor = ByteToCharConverter.
getConverter(encoding);
    char[]c= convertor.convertAll(b);
    for(int i= 0; i<c.length; i++ ) {
        System.out.println( Integer.toHexString
(c[i]));
    }
    ……
```

则结果为: 0x4F60

但是如果使用 ISO8859-1的编码,打印出来的是: 0x00C4, 0x00E3

如果把上例的结果作为输入,则

```
    ……
    encoding= "gb2312";
    char c[]= { \u4F60 };
        convertor = ByteToCharConverter.
getConverter(encoding);
    byte []b= convertor.convertAll(c);
    for(int i= 0; i<b.length; i++ ) {
        System.out.println( Integer.toHexString
(b[i]));
    }
    ……
```

而结果为: 0xC4, 0xE3

如果使用 ISO8859-1就是 0x3F,?。其中? 号表示无法转化。因为 ISO8859-1是字节编码,会有两个字,而 4F60只有一个字节,所以后面还有一个字无法转化。

很多中文问题从这两个最简单的类派生出来,但却有很多类不支持直接把 encoding输入,这会带来诸多不便。很多程序直接用 default的 encoding,这就给移植带来了很大困难。

1.3 UTF-8

UTF (Unicode/UCS Transformation Format)推荐使用 UTF-8和 UTF-16两种格式,其中 8和 16指的是 Bits数。UTF-16基本上就是 Unicode双字节的实现,或再加上一个应付未来需要的扩充编码机制。UTF-8是一种不等幅的编码方式, ASCII字码可以保持原状不受影响,因此不需要做转换[3]。而其他汉字资料须通过程序来转换,会增加长度,因为每个字需要额外 1个或 2个 Bytes来进行编码。UTF-8是和 Unicode码一一对应的,实现方法如下:。

7位的 Unicode 0__

11位的 Unicode 110__ 10__

16位的 Unicode 1110__ 10__ 10__

21位的 Unicode 11110__ 10__ 10__ 10__

大多数情况是只使用到 16位以下的 Unicode码。每个以 UTF-8编码的字符,不管是以一、二、三个 bytes出现,第一个 byte前端都清楚地标示了该字符的 byte总数。如 110种有两个 1,代表这种字符是以第二种方式出现,由两个 bytes组成。而 1110有三个 1,表示这种字符的第三种方式出现,由三个字节组成。每个多重 byte的 UTF-8编码有一个共同的通性,即其中的第二个、第三个 byte,一律以 10两个 bits开头。由于其中的最高位总设成 1,可以很容易和那些在 UTF-8中只用一个 Byte的 ASCII字元区分开来,方便侦错。

由于上述设计特点, UTF-8和 Unicode码之间,可以很容易做双向自由转换,而不会丢失任何资料。如汉字“你”的 GB码是: 0xC4E3, Unicode码是 0x4F60。显然, 0xC4E3的二进制是: 1100010011100011 由于只有两个字节,则按照两个字节的编码来排是行不通的,因为第 7位不是 0,因此返回“?”。显然, UTF-8编码和 GB不是一一对应的。

1.4 String和 byte[][2]

String的核心实际上是 char[],要把 byte转化成 String必须经过编码。String.length()其实就是 char数组的长度,如果使用不同的编码,很可能会错分,造成散字和乱码。例如:

```
    ……
    Byte[]b= {(byte) \u00c4,(byte) \u00e3 };
    String str= new String(b, encoding);
    ……
```

如果 encoding= 8859-1,则会有两个字,但是 encoding= gb2312只有一个字,而这个问题在处理分页时经常发生。

2 Java/JSP编译器对汉语编码的处理方式

Java编译器和 JSP编译器在对汉语编码进行编译时采用不同的处理方式[3]。

2.1 Java编译器对汉语编码的处理方式

Java编译器在对源文件编译前,会先把源文件转换为 Unicode编码,因而在编译时一定要把源文件的编码方式正确无误地“告诉”编译器。如果没有指定 encoding,则按照系统的默认 encoding, GB平台上是 GB2312,英文平台上是 ISO8859-1。Java的编译器是调用 sun.tools.javac.Main的类,对文件进行编译,这个类有一个 compile函数,其中有一个 encoding的变量, encoding的参数直接传给 encoding变量。

编译器就是用这个变量读取 Java文件,然后用 UTF-8形式编译成 class文件。例如:

```
public void test()
{
    String str= "你";
    FileWrite write= new FileWriter("test.txt");
    write.write(str);
    write.close();
}
```

如果用 GB2312编译,就会找到 E4 BD A0的字段;而如果用 ISO8859-1编译,则会有 00C4 00E3的字段,它的二进制码为: 00000000 11000100 00000000 11100011

因为每个字符都大于 7位,因此用 11位编码:

11000001 10000100 11000011 10100011

C1—— 84—— C3—— A3就会找到 C1 84 C3 A3 但是往往会忽略掉这个参数,因此出现跨平台的问题

上面例子在中文平台上编译运行后,生成 ZhClass,而在英文平台上编译运行,则会输出 EnClass,其中, ZhClass在中文平台上执行正常,但是在英文平台上会出现乱码;而 EnClass在英文平台上执行正常,但是在中文平台上会出现乱码。出现这些问题的原因主要是: (1)在中文平台上编译后, str在运行状态为 char[]的是 0x4F60;在中文平台上运行, FileWriter的默认编码是 GB2312,因此 CharToByteConverter会自动调用 GB2312的 converter,把 str转化成 byte输入到 FileOutputStream中,于是 0xC4, 0xE3被放进了文件。如果是在英文平台上, CharToByteConverter的默认值是 ISO8859-1, FileWriter会自动调用 ISO8859-1去转化 str,但是却无法解释,因此会输出"?"。 (2)在英文平台上编译后, str在运行状态为 char[]的是 0x00C4 0x00E3,在中文平台上运行,中文无法识别,因此会出现"?"。在英文平台上, 0x00C4- > 0xC4, 0x00E3- > 0xE3,? 因此 0xC4, 0xE3被放进了文件。因此,如果在各种语言的平台上编译时采用 encoding指定与源文件的编码相同的编码方式,就可杜绝国际化问题。

2.2 JSP编译器对汉语编码的处理方式

对于 JSP,编译器会根据设定的字符集来判断 JSP文件使用的是何种编码方式,进而将其转换成 Unicode后进行编译。若 JSP中未指定,编译器则会把 JSP文件看作是按照系统默认的编码保存。在 JSP2.0里新增了一个指令来通知编译器这个源文件所使用的编码方式。但是在 JSP中, JSP会被 JSP编译器编译为 Servlet来运行。在 Servlet中,除了一定要把源文件用的是何种编码方式"告诉"编译器外,还要注意实际提交的 URL数据、表单数据的编码格式和 request中声明的编码格式一致。客户端浏览器在通过表单和 URL提交数据时,容器和 JVM会将 request中的数据看作是按照 request所声明的编码方式来编码的,将数据由这种编码方式转换为 unicode后再送入 servlet servlet输出的 unicode数据会由容器根据 response中声明的编码方式进行转换,再送到客户端浏览器上。

在接收客户端输入时,用 request.setCharacterEncoding()声明请求中数据的编码方式。在向客户端输出时用 response.setContentType("text/html; charset= ")声明响应的数据的编码方式,告知浏览器以哪种编码方式显示。这两个 JSP指令声明了请求和响应的编码方式。只要确保 URL参数或表单中数据的编码方式和所声明的编码方式一致,再通过告知 JSP编译器本 JSP文件采用的编码方式及含有何种字符,即可解决 JSP的汉语的编码问题。

解决 request对象读取时出现乱码情况的函数为: request.getParameter("变量名"),它读取客户请求中指定的一个变量的值。变量名一般是表单元素名。值一般是字串型。若按以下代码去接收表单的变量,会出现乱码:

```
<%
    String mm= request.getParameter("login");
    if(mm.length()> 0) { out.println(mm);  }
%>
```

表单传出去的数据本身没有问题,但关键是接收端的 JSP网页是用 request对象读取,而此时的字串已经被 JDK用 Unicode序列化,出现乱码。Java默认的字符集是压缩双字节的 Unicode码。所以会将收到字串统统作 Unicode码处理。导致中文出现乱码。解决方法是,将压缩的双字节 Unicode字符串转成非压缩的拉丁英文字符串 ISO8859-1编码。然后把 ISO8859-1串转成中文国标码 GB2312或 GBK串。相关代码如下:

```
<%!
//解决 request对象读取时出现乱码的函数
public String toChinese(String s1)  {
   try{
   //把压缩的 unicode转成 ISO8859-1
     byte st1[]= s1.getBytes("ISO8859-1");
     //把 ISO8859-1转成中文国标码
```

```
    return new String(st1,"gb2312");
   }
    catch(Exception e)
    {   return s1;}
   }
%>
 <%
    String   mm= request.getParameter("login");
    mm= toChinese(mm);        //调用中文转化函数
   if(mm.length()   >    0)   {    out.println
(mm);    }
%>
```

以上方法很好地解决了用 request对象读取时出现乱码的情况。具体使用时,在存回数据库前,把 Java变量中的串转成 GB2312,然后用 SQL语句将这些支持中文的 SQL传到数据库服务器端执行。为方便起见,也可用 package将这个类做一个类库 jar包。在 TOMCAT的 JSP中要用到它时,将 JAR包复制到 c:\tomcat\common\lib下,在 JSP中用 import导入类库,然后用 new 类名生成一个对象后使用。

3 结束语

Java编程语言作为国际化语言,必然要求其对多国字符有很好的支持,而且 JSP基于 Java语言,两者在汉语编码中的处理有所不同。本文介绍 Java编程中汉语编码问题及其 Java和 JSP编译器解决方法,但未能对其他语言的编码问题做进一步的研究。Java国际化语言问题的解决,将更能适应计算的网络化的需求,为其能够在网络世界进一步发展奠定基础。

参考文献:

[1] 光 军,胡 波. JSP应用开发实例详解[M].北京:北京航空航天大学出版社,2002.

[2] Decoder. JSP技术揭秘[M].北京:清华大学出版社,2001.

[3] 杨学瑜,王志军,刘同利. JSP入门与提高[M].北京:清华大学出版社,2002.

(责任编辑:黎贞崇)

(上接第 113页)

步骤 2用查询分析器修改 SQL Server数据库 owner为 SA

步骤 3:修改 SQL Server对象命名长度。

步骤 4:配置 SQL的 ODBC数据源。

步骤 5:用 Oracle Migration workbench迁移数据:

(1)启动 ORACLE Migration Workbench,选择“Default Repository”;

(2)在“File”菜单的“Select Migration Source”选择“Microsoft SQLServer 2000…”;

(3)在“Action”选择“Capture Data Source…”按照提示一步步从中 SQL Server获取数据;

(4)修改原数据库中数据类型,保证数据导入时无警告和错误;

(5)选择“Action”中的“Migration Data”用于向 Oracle导入数据,按照提示将数据导入到 Oracle。

3.2 问题分析

迁移过程中 2个触发器和 2个存储过程都发生迁移错误,对此可以忽略错误,然后到 Oracle中进行修改。在迁移的过程中,SQL Server 2000中时间类型数据中秒的小数部分没有迁移成功,但是并不影响数据的使用。

4 结束语

将旧数据库系统中的数据迁移到新的数据库中是一个很常见的重要的问题。这里我们介绍了几个数据迁移的工具和方法,最后通过一个实例,介绍如何使用 Oracle Migrate Workbench数据迁移工具将数据从 SQL Server迁移到 Oracle,在迁移过程中虽然有些地方发生迁移错误,但是经过人工修改后,这些错误并不影响对迁移过来的数据的正确使用。

参考文献:

[1] 微软公司.把 Oracle数据库移植到 SQL Server 7.0[EB/OL]. http://www.southtalent.com/5i58/index.asp, 2005-04-09.

[2] 岳鹏飞.如何将一个数据库从 sql server7.0(desktop)移植到 oracle8i[EB/OL]. http://www.china-askpro.com/index.html, 2002-12-08.

(责任编辑:黎贞崇)