Java 中文乱码问题研究

任平红[1],陈 矗[1],郑秋梅[2]
(1. 曲阜师范大学 计算机科学学院,山东 日照 276826;
2. 中国石油大学 计算机与通信工程学院,山东 青岛 266555)

摘 要:Java 语言的跨平台特性,使其在各种系统环境下能够正确运行,但是在开发 Java Web 应用程序时,因为编码不统一,经常会出现中文乱码问题。针对客户端和服务器端传输数据,客户端显示中文字符编码,以及应用程序与数据库之间的数据交互等问题,分析了 Java 乱码产生的原因,并针对每种情况,结合实际的项目开发经验,给出了设置页面编码方式,修改 Web 服务器编码属性,以及使用过滤器等等方法。采用以上方法,可以有效地解决 Java Web 中的中文乱码问题。

关键词:编码;Java Web 应用;中文乱码;字符集;过滤器
中图分类号:TP311.1 **文献标识码**:A **文章编号**:1673-629X(2013)03-0117-04
doi:10.3969/j.issn.1673-629X.2013.03.030

Research of Character Encoding in Java

REN Ping-hong[1],CHEN Chu[1],ZHENG Qiu-mei[2]
(1. College of Computer Science,Qufu Normal University,Rizhao 276826,China;
2. College of Computer and Communication Engineering,China University of Petroleum,
Qingdao 266555,China)

Abstract:The cross-platform feature of Java language enables Java applications to run correctly in a variety of system environments. However,Chinese characters often garble in the development of Web applications because of non-uniform encoding. For data transmission between client and server,the client's display of Chinese characters and the interaction between Java applications and databases,analyze the cause of garbled characters and present solution for each case according to the actual experience in project development such as setting the encoding of pages,modifying encoding attribute of the Web server and making use of filters. The problem of Chinese garbled in the Java Web can be solved effectively referring to the above methods.

Key words:coding;Java Web application;Chinese character encoding;character set;filter

0 引 言

Java 在设计之初就考虑到了国际化的问题,Java 平台[1]内部的字符编码为通用的国际标准字符集 Unicode,JVM(Java Virtual Machine,Java 虚拟机)屏蔽了与操作系统平台相关的信息,使 Java 程序生成在 JVM 上运行的字节码。JVM 在执行字节码时,还要把字节码解释成具体平台上的机器指令[2]。计算机中存储数据采用二进制,不同的字符对应二进制数据的规则,就是字符的编码,字符编码的集合称为字符集。

Java 中常用的字符集有[3]:

(1)ASCII 码。

ASCII 码是美国国家信息互换标准码,是基于英文字符的一套电脑编码系统。它使用一个字节表示字符,最多可以表示 256 种字符。ASCII 码是计算机中使用最广泛的字符集,用于在不同的计算机硬件和软件系统中实现数据传输标准化,它已被国际标准化组织定为国际标准。

(2)ISO-8859-1。

ASCII 码中只有英文字符,缺少其他语言所需的字符。国际标准化组织基于 ASCII 编码,在其基础上增加其他语言地区的常用字符形成新的编码。其中最常用的是 ISO-8859-1,也叫作 Latin-1 或西欧语言。它以 ASCII 为基础,是单字节编码,是 Java 网络传输使用的标准字符集。

(3)GB2312,GBK。

GB2312 是中国国家标准汉字信息交换用编码,基本能满足汉字的处理需求。GBK 是对 GB2312 的扩充,采用双字节表示,包括了一些生僻字的编码,完全兼容 GB2312。

收稿日期:2012-07-10;**修回日期**:2012-10-16
基金项目:国家自然科学基金资助项目(51006123)
作者简介:任平红(1980-),女,山东德州人,讲师,硕士,研究方向为软件工程、计算机图形图像处理。

(4) Unicode。

Unicode 是统一的在计算机上使用的字符编码标准,使用 0 ~ 65535 的双字节无符号数对字符编码,为每个字符设定唯一的二进制编码。它是由国际标准化组织设计,可以容纳全世界所有语言文字的编码方案。Unicode 编码可以满足跨语言、跨平台进行文本转换及处理的要求,又称为万国码。

(5) UTF-8。

在 Unicode 编码中,一个英文字符要占两个字节,数据量较大。为了减少数据量,可使用 UTF-8 编码。UTF-8 是一种针对 Unicode 的可变长字符编码,也是一种前缀编码,又称为万国码。它可以用来表示 Unicode 标准中的任何字符,且其编码中的第一个字节仍与 ASCII 码兼容,是电子邮件,网页及其他存储应用中,优先采用的编码。UTFF-8 包含全世界所有国家使用的字符。

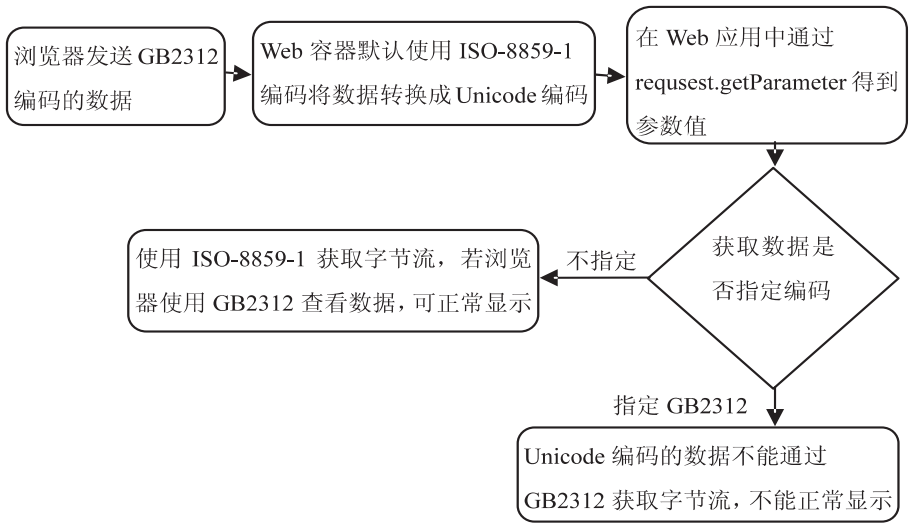

图 1 Web 请求/响应过程中的字符编码转换过程

1 Java 乱码产生的原因

Java 乱码问题产生的根本原因是使用错误的字符集解码字节流或者将给定的字节流用错误的字符集编码。为了使 Java 语言具有跨平台的特性,Java 内部使用统一的字符编码 Unicode 字符集来表示字符,如果 Unicode 字符集与本地字符集相互转换出现问题,就会产生乱码。如果需要在 Java 程序中读取字符类型的数据,要将本地字符集编码转换为 Unicode 编码,同时,如果需要输出字符数据,则将 Unicode 编码转换为本地字符集编码[4]。但是在实际的 Java Web 应用中,可能还会涉及到客户端的浏览器、应用程序、Web 服务器以及数据库等等相关的因素。每一部分都可能会使用不同的字符集,如果字符集之间不兼容或转换出现问题,也会出现乱码问题。Java 乱码问题大部分产生在 Web 应用中。

客户端的浏览器向服务器端提交 HTTP 请求时,会以一定的编码集对传递的数据进行编码。服务器端接受到 HTTP 请求后要解析 HTTP 协议,并且要对参数及 Cookie,URL 等进行解码,可能还需要读取数据库、本地或网络资源中的数据,都可能存在编码问题。当所有请求的数据处理完毕后,需要将这些数据再编码,并发送到浏览器端,再经过浏览器解码成为文本[5]。

假设浏览器发送的数据编码方式为 GB2312,此过程如图 1 所示。

可见,如果在 Java Web 应用中不指定任何的编码方式,并且也不读取其它资源的数据,用 GB2312 编码方式显示网页,可以正常显示。但是在 JSP/Servlet 中,可能需要直接写入中文字符或者从其他的外部资源中读取某些中文字符。如果中文字符对应的 Unicode 编码是由 GB2312 转换而得来的,那么用 ISO-8859-1 编码方输出,会出现乱码。所以应该在 Web 容器内部以及浏览器端使用 GB2312 或 GBK 进行编码。

2 乱码问题的分类及对应的解决方案

2.1 包含中文字符的 JSP 页面乱码问题

JSP 页面不能正常显示中文字符的原因是没有正确地设置页面的 pageEncoding 属性或 charset 属性[6]。pageEncoding 属性指的是 JSP 文件在本地保存时的编码方式,contentType 中的 charset 属性是指服务器把网页内容发送给客户端时所使用的编码。在 JSP 页面里使用 page 指令设置以上两个属性的值为支持中文的字符集编码(例如 UTF-8,GB2312 或 GBK)即可正常显示中文字符:

```
<% pagepageEncoding="GB2312"%>
<% page contentType="text/html;charset= GB2312"%>
```

2.2 提交的表单数据或参数中有中文字符

(1) 以 Post 方式提交表单。

提交表单数据有 Post 和 Get 两种方法。Post 方法提交普通表单时,浏览器根据 contentType 中的 charset 值对表单中的参数进行编码,通过 HTTP 的 BODY 传递到服务器端,在服务器端同样也是使用 charset 值进行解码[7]。例如 Servlet 在接收数据时,可通过 ServletRequest 接口的 setCharacterEncoding 设置正确的编码[8]。由于以 Post 方式提交的表单采用了特殊的加密方式,只要正确地设置了编码,一般不会出现乱码问

题。

如果表单的 type 值为 multipart/form-data,即表单中包含 file 类型的上传文件的控件时,上传的文件编码也是使用 contentType 中的 charset 值进行编码。但是上传文件是用字节流方式传输到服务器的临时目录,如果指定的 charset 不支持字节流编码,将会默认使用 ISO-8859-1 编码。

(2)以 Get 方式提交表单。

当表单或参数的提交方式为 Get 时,在接收数据时通过 setCharacterEncoding 设置字符的编码集是无效的,因为在 Web 服务器 Tomcat5.0 版本以后,默认使用 ISO-8859-1 对 URL 提交的数据和以 GET 方式提交的表单数据进行重新编码和解码[9],因此为了解决乱码问题,需要在 Tomcat 的配置文件 server.xml 中修改服务器的属性,使服务器采用支持中文的编码方式对数据进行编码。

将

```
<ConnectorconnectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>
```

修改为:

```
<ConnectoruseBodyEncodingForURI="true" URIEncoding="UTF-8" connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>
```

以上修改只有重启 Tomcat 服务器才能生效。

(3)使用过滤器解决乱码问题。

无论是以何种方式提交的表单或数据,都可以在接收时对参数进行编码转换[10],例如:

```
strName = new String(strName.getBytes("ISO-8859-1"), "UTF-8");
```

但是这种方法实现起来比较繁琐,在实际的项目开发中,一般推荐使用过滤器解决乱码问题。只要在配置文件中正确地设置过滤资源,无论以何种方式提交表单或数据,都可以解决乱码问题。

过滤器(Filter)的作用是用于过滤、拦截请求或响应信息,可以在 Servlet 或 JSP 页面运行之前和之后被自动调用。Servlet 过滤器能够对 Servlet 容器的请求和响应对象进行检查和修改。当客户端发出资源请求时,Web 服务器可以根据应用程序的配置文件(例如 web.xml)的配置进行检查。如果请求的资源在过滤规则设置的范围之内,则对客户请求/响应进行拦截,并且可以根据过滤器的设置对请求头和请求数据进行检查或修改。

在过滤器中可以设置编码方式或资源的 MIME 类型,例如 CharsetFilter.java:

```
packagecn.edu.qfnu.myFilter;
importjavax.servlet.*;
importjava.io.IOException;
public classMyCharsetFilter implements Filter{
private FilterConfig myFilterConfig=null;
private StringmyEncoding=null;
public void init(FilterConfig filterConfig) throws ServletException {
this.myFilterConfig=filterConfig;
this.myEncoding = filterConfig.getInitParameter("initEncoding");
}
public voiddoFilter(ServletRequest req, ServletResponse res,
FilterChain filterChain) throws ServletException, IOException,
{
if(req.getCharacterEncoding()==null){
req.setCharacterEncoding(encoding);
}
filterChain.doFilter(req,res);
}
public void destroy(){
myFilterConfig=null;
myEncoding=null;
}
}
```

在 web.xml 文件中对参数和过滤器的过滤规则进行设置:

```
<filter>
<filter-name>MyCharsetFilter</filter-name>
<filter-class>cn.edu.qfnu.myFilter.MyCharsetFilter</filter-class>
<init-param>
<param-name>initEcoding</param-name>
<param-value>UTF-8</param-value>
</init-param>
</filter>
<filter-mapping>
<filter-name>MyCharsetFilter</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
```

2.3　Java 程序和数据库之间的乱码问题

数据库本身的编码应该支持中文,否则会产生乱码问题。此外,大部分数据库的 JDBC 驱动程序在 Java 应用程序和数据库之间使用 ISO-8859-1 为默认的数据编码格式来传递数据。Java 应用程序向数据库中写入数据时,JDBC 首先会把程序内部的 Unicode 编码格式转化成 ISO-8859-1 编码格式,然后进行保存[11]。例如 MySQL 数据库的默认编码为 ISO-8859-1,中文值写入数据库时可能会出现乱码。解决方法为把数据库的默认编码修改成为支持中文的编码,例如 UTF-8,GBK 或 GB2312,可以通过修改 my.ini 文件实现编码的修改。

另外大部分数据库都支持 Unicode 编码方式,可以在驱动的 URL 中加入编码信息,例如,连接 MySQL 数据库:

jdbc:mysql://localhost/test? useUnicode=true&character Encoding=UTF-8

2.4 Java 程序与文件流的乱码问题

Java 中读写字节流文件使用 FileInputStream/FileOutputStream,读取字符流文件使用 FileReader/FileWriter,可以避免字节与字符之间的转换[12]。但以上几个类只能使用系统默认的编码方式,假如文件流本身的编码方式和系统编码方式不一致,也会出现乱码问题。此时,可以使用基于字符的类 Input Stream Reader/Output Stream Writer,优点是可以在构造函数中指定文件流的编码类型,Input Stream Reader(InputStream in, Charset cs) 和 Output Stream Writer(Output Stream out, Charset cs)。

3 结束语

在上述分析中,给出了 Java 中乱码问题产生的原因,Java 中常见的几类乱码问题以及对应的解决方法。在实际的 Web 应用中,还会涉及到 Web 服务器,应用服务器以及 JDBC 数据库驱动等等,但只要根据字符编码及转换的原理,在程序的入口和出口采用统一的编码和解码方式,即可以解决 Java 乱码问题。

参考文献:

[1] Gosling J, Joy B, Steele G. The Java Language Specification [M]. MA: Addision-Wesley Publishing Company, 1996.

[2] Lindholm T, Yellin F. The Java Virtual Machine Specification [M]. MA: Addision-Wesley Publishing Company, 1996.

[3] 王子君,范学峰,张志浩. Java 编码问题研究与应用[J]. 计算机工程,2002,28(3):242-245.

[4] 冀振燕,程 虎,梅 嘉. Java 语言国际化的设计与实现[J]. 软件学报,2000,11(11):1541-1546.

[5] 冯金辉,朱良森. Java 编码中文问题研究及解决方案[J]. 计算机系统应用,2005(11):79-81.

[6] 许 晖,李涓子. J2EE 系统国际化问题的解决方案[J]. 计算机工程,2005,31(18):79-81.

[7] 曹 莉,赵文静. Java 中文处理研究[J]. 计算机技术与发展,2006,16(5):100-103.

[8] 金恩华,徐良贤. J2EE Web 应用中汉字编码的研究[J]. 计算机应用与软件,2005(6):55-56.

[9] 包竹苇,李 淼,张 建. Java 网络传输中字符编码问题的研究[J]. 计算机工程与应用,2007(4):93-95.

[10] 彭利民,孙素云. JSP 和 Servlet 网络编程设计中汉字编码的研究[J]. 计算机与现代化,2006(3):43-45.

[11] 刘 冰. Java 编程中中文问题的产生及其解决方案[J]. 现代计算机,2010(3):105-107.

[12] 刘长生,谢 强,丁 林. Java 应用中的汉字乱码问题分析[J]. 计算机技术与发展,2006,16(1):158-161.

(上接第 116 页)

参考文献:

[1] Govert N, Kazai G. Overview of the initiative for the evaluation of XML retrieval (INEX) 2002[C]//Proc of the 1st Workshop of the Initiative for the Evaluation of XML Retrieval (INEX). Schloss Dagstuhl, Germany: European Research Consortium for Informatics and Mathematics, 2002:1-17.

[2] 周 健,孙丽艳. 面向对象 XML 的存储模式的研究[J]. 计算机技术与发展,2009,19(3):114-117.

[3] 况 旭,刘 波. XML 的面向对象语言特性[J]. 计算机技术与发展,2010,20(1):54-57.

[4] 刘喜平,万常选,刘德喜. 有效的 XML 模糊内容与结构检索和计分[J]. 计算机研究与发展,2010,47(6):1070-1078.

[5] Bao Zhifeng, Lu Jiaheng, Ling Tok Wang, et al. An Effective Object-level XML Keyword Search[J]. Computer Science, 2010, 5981:93-109.

[6] 万常选, 鲁 远. 基于权重查询词的 XML 结构查询扩展[J]. 软件学报,2008,19(10):2611-2619.

[7] Sigurbjornsson B, Kamps J, de Rijke M. The University of Amsterdam at INEX 2004[C]//Proc of the 3rd Workshop of the Initiative for the Evaluation of XML Retrieval (INEX). Berlin: Springer, 2004:104-109.

[8] Amer-Yahia S, Lakshmanan L V S, Pandit S. FleXPath: Flexible structure and full-text querying for XML[C]//Proc of the ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2004:83-94.

[9] Liu S, Chu W W, Shahinian R. Vague content and structure (VCAS) retrieval for document centric XML collection[C]//Proc of Int Workshop on the Web and Databases (WebDB). New York: ACM, 2005:79-84.

[10] Liu S, Zou Q, Chu W W. Configurable indexing and ranking for XML information retrieval[C]//Proc of the 27th Annual Int ACM SIGIR Conf on Research and Development in Information Retrieval. New York: ACM, 2004:88-95.

[11] Amer-Yahia S, Koudas N, Marian A, et al. Structure and content scoring for XML[C]//Proc of Int Conf on Very Large Data Bases. New York: ACM, 2005:361-372.

[12] 黄 瑞,史忠植. 一种新的 Web 异构语义信息检索方法[J]. 计算机研究与发展,2008,45(8):1338-1345.

Java中文乱码问题研究

作者： 任平红， 陈矗， 郑秋梅
作者单位： 任平红,陈矗(曲阜师范大学 计算机科学学院,山东 日照 276826)， 郑秋梅(中国石油大学 计算机与通信工程学院,山东 青岛 266555)

刊名： 计算机技术与发展
英文刊名： Computer Technology and Development
年，卷(期)： 2013(3)