案例编程MOOK系列

# Java

## 编程精选集锦


《电脑编程技巧与维护》杂志社 主编

科学出版社
www.sciencep.com

案例编程 Book 系列

# Java 编程精选集锦

《电脑编程技巧与维护》杂志社　主编

科学出版社

北　京

## 内 容 简 介

Java 具有平台独立、面向对象、多线程等许多优点，是目前最为优秀的编程语言。所以，得到广大编程人员的喜爱，因而在网络编程方面 Java 已成为首选的编程语言。

本书的特色体现如下几点：第一，每一章都是通过一个个的实例来介绍 Java 应用编程方法和技巧，避免枯燥、空洞的理论，并且每一个实例都具有很强的实用性和代表性。在实例的讲解上一般是先给设计目标，接着介绍实现该目标的基本思想和方法，然后详细给出其核心程序的源代码，对程序的关键部分进行讲解并给出程序的运行效果。第二，所选的每一个实例都是从事 Java 编程人员的经验总结，具有很强的实用性，其中很多编程技巧值得借鉴。第三，每一个实例的程序源代码都是经过上机调试并通过，给开发人员移植源代码带来了方便，从而加快应用编程的步伐。第四，选取一些老版本开发环境的经典实例加以点评分析，使之能够起到触类旁通的作用。

本书定位于有 Java 应用基础的编程人员和开发人员，对初学 Java 编程的新手也有一定的参考价值。

**图书在版编目(CIP)数据**

Java 编程精选集锦/《电脑编程技巧与维护》杂志社主编. —北京：科学出版社，2003
(案例编程 MOOK 系列)
ISBN 7-03-011489-2

Ⅰ.J… Ⅱ.电… Ⅲ.JAVA 语言-程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2003)第 036157 号

策划编辑：孟战龙 责任编辑：于 娜
责任印制：吕春珉 封面设计：三函设计

科学出版社 出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com
新蕾印刷厂 印刷
科学出版社发行 各地新华书店经销

2003 年 6 月第 一 版 开本：B5(720×1000)
2003 年 6 月第一次印刷 印张：26 1/2
印数：1—5 000 字数：667 000

**定价：45.00 元(含光盘)**

(如有印装质量问题，我社负责调换〈路通〉)

# 出版说明

MOOK 又称杂志书，是杂志(Magazine)和图书(Book)相结合的新型出版物，它集合了二者的优点，既有图书专业性、权威性和内容深入全面的特点，又有杂志的贴近生活，轻松、休闲、通俗易懂、时效性强的特点。

MOOK 代表了一种出版趋势，即图书的杂志化和杂志的图书化，在这里可以发表专家学者的真知灼见，它营造的是一种轻松、休闲的氛围，专家学者用读者更易于理解的语言，阐释理论、知识和信息。

“案例编程 MOOK 系列”是科学出版社策划出版的一套 MOOK 丛书，包括《Visual C/C + + 编程精选集锦》(三册)、《Visual Basic 编程精选集锦》、《Java 编程精选集锦》、《Delphi 编程精选集锦》、《ASP 编程精选集锦》共七本。每本书有几十个编程实例，并依不同的编程应用分成了若干章，条目清晰可查、使用极为方便。这套书选编了《电脑编程技巧与维护》杂志近一两年发表的精彩编程文章，并根据读者要求，组织收入了更具价值的编程案例。这套书遵循 MOOK 的特点：讲究内容的深入性、专业性和权威性，同时兼顾轻松、通俗易懂、时效性强等特点。实例讲解部分先给出设计目标，然后介绍实现目标的基本思想和方法，最后详细给出其核心程序的源代码，对程序的关键部分进行讲解并给出程序的运行效果。

“案例编程 MOOK 系列”的与众不同，在于它的理念。这里少了教科书上的说教，少了尘世浮华的喧嚣；这里带给你的是一份清新、纯粹的体验感受；把它作为生活的一部分，用心经营，仔细体会。“案例编程 MOOK 系列”的与众不同，在于它的思想。汇集众多顶级程序员、项目经理的成功经验，告诉你最新的创意和最实用的方法；力求为你建造一个真正的知识整合，思想交流的平台；你可以习得编程高手的诀窍、丰富你的编程技巧，你可以与高手切磋编程实战技巧，你可以积累更多的经验。

思想、智慧、观念、技巧无处不在……

# 前　言

Java具有平台独立、面向对象、多线程等许多优点，是目前最为优秀的编程语言，受到广大编程人员的喜爱，因而Java已成为网络编程方面的首选编程语言。

本书精选了《电脑编程技巧与维护》杂志近一两年发表的精彩编程文章，并根据读者要求组织收入了更具价值的编程案例。《电脑编程技巧与维护》杂志是为从事电脑编程、系统应用和开发人员创办的专业性和实用性都很强的技术刊物，它从1994年创刊以来，八年多始终遵循着"实用第一，智慧密集"的办刊宗旨，紧紧跟踪计算机软硬件技术发展和应用趋势，不断求变创新。针对软件开发过程中许多要点和技巧问题，着重提供各类解决方案。对编程人员来说，程序开发能力的提高，除了对语言和算法的学习，还要集思广益，充分借鉴参考别人的长处，深入透彻地理解其中的精髓，然后融入到自己的设计能力中去，这样无论是对于自身和整体都有莫大的提高，这正是编写此书的初衷。根据Java的不同应用对象，将精选的66个实例分列在5章中：第1章Java编程基础与应用，是为初学Java编程和应用的读者提供一些入门的实例；第2章数据库应用编程，为编程人员提供使用Java进行数据库编程方法和技巧；第3章网络应用编程，介绍Java在网络编程方面的应用实例和技巧；第4章图形图像应用编程，介绍Java实现图形图像处理的编程技巧的一些实例；第5章Java应用编程专家指点，介绍一些Java开发中关键技术问题的解决实例。全书每一章都本着实用第一的原则，紧紧围绕一个主题展开，循序渐进，由浅入深地介绍了使用Java进行应用程序开发的思想方法与编程技巧。

本书的特色体现如下几点：其一，每一章都是通过一个个的实例来介绍Java编程方法和技巧，避免枯燥、空洞的理论，并且每一个实例都具有很强的实用性和代表性。第二，所选的每一个实例都是从事Java编程人员的经验总结，具有很强的实用性，其中很多编程技巧值得借鉴。第三，每一个实例的程序源代码都是经过上机调试并通过，给程序开发人员移植源代码带来了方便，加快编程的步伐。第四，对老版本经典实例进行点评，选取一些老版本开发环境的经典实例加以点评分析，使之能够起到触类旁通的作用。

本书是《电脑编程技巧与维护》资源的二次开发，浓缩了Java程序设计的精华，其目的是提升Java程序开发能力，把应用Java进行编程的心得体会、经验与读者共享。本书定位于有Java应用基础的编程人员和应用开发人员，对初学Java编程的新手也有一定的参考价值。本书内容全面、概念清晰、层次分明，例题典型而实用，但不足甚至疏漏之处在所难免，恳请广大读者批评指正。

《电脑编程技巧与维护》杂志社

# 目　录

## 第1章　Java 编程基础与应用


实例 1　Java 和 ASP 的结合：Applet 小程序从 Form 表单中动态提取参数 …… 2
实例 2　Servlet 中汉字处理浅析 …… 7
实例 3　用 Java Servlet 替代 CGI …… 13
实例 4　Java 中的 Hashtable 类及其应用 …… 16
实例 5　Linux 下 Java 程序的编译与调试 …… 22
实例 6　Java RMI 的原理和实现方法 …… 28
实例 7　Java、Java Applet 与 JavaScript 间的通信 …… 31
实例 8　用 Java 做数字签名的方法 …… 35
实例 9　用 Java Applet 实现树型导航控件 …… 39
实例 10　用 JBuilder 开发 Java 小应用程序 …… 53
实例 11　Java 程序与 Java-Servlet、ASP、PHP、CGI 等的通信 …… 57
实例 12　用 Java 小应用程序实现正则表达式下字符串的查找 …… 63
实例 13　Java 与 JavaScript 的相互访问 …… 68
实例 14　纯 Java 网页的编程技巧 …… 73


## 第2章　数据库应用编程


实例 15　JavaServlet 驱动 SQL Server 中的数据库 …… 86
实例 16　JDBC 查询结果的表格方式显示 …… 90
实例 17　用 Java JDBC 技术连接数据库 …… 94
实例 18　在 Applet 中应用 JDBC 访问数据库 …… 98
实例 19　用 Java Applet 访问 Oracle 7 数据库 …… 102
实例 20　Java 对象系列化技术在分布式数据库中的应用 …… 112


## 第3章　网络应用编程


实例 21　利用 Java 实现网络通信 …… 123
实例 22　Java 编程搜索网络服务器 …… 132
实例 23　用 Java 实现底层网络通信 …… 140
实例 24　用 JBuilder 4 创建基于 InternetBeans Express 的 Servlet …… 148
实例 25　用 Java 编写 Web 服务器 …… 152
实例 26　用 Java 编程收发电子邮件 …… 156

实例 27　网络象棋原理及 Java 实现 …… 164
实例 28　使用 Java 访问 POP3 邮件服务器 …… 195
实例 29　Java 的网络功能与编程 …… 199
实例 30　Applet 编程技巧 …… 212
实例 31　使用 Java Swing 轻松编制浏览器 …… 221

## 第4章　图形图像应用编程

实例 32　Java 中如何消除动画闪烁 …… 226
实例 33　使用 Java 实现带滚动条的图像缩放 …… 229
实例 34　Java 3D 中实现 VRML 虚拟场景的调用和显示 …… 245
实例 35　Java 中实现图像切换特效 …… 250
实例 36　Visual J++6.0 中读取图像的灰度与进行灰度变换 …… 255
实例 37　应用 Java 语言进行 AutoCAD 2000 二次开发 …… 259
实例 38　使用 Java 定制点亮式按钮 …… 270
实例 39　利用 Java 制作特效文字 …… 274
实例 40　用 Java 制作广告轮换条 …… 276

## 第5章　Java 应用编程专家指点

实例 41　如何让 Java 容器中的组件动起来 …… 281
实例 42　Java 中利用管道实现线程间的通信 …… 286
实例 43　基于 Java 语言的声音播放 …… 293
实例 44　Java 中的名字目录服务及其管理实现 …… 296
实例 45　优化 Java 程序设计 …… 300
实例 46　在 Java 程序不中断运行的情况下动态更换元件 …… 305
实例 47　用 Java 语言实现经典的同步—互斥问题 …… 310
实例 48　用 Java 自己动手编制网络搜索软件 …… 319
实例 49　利用 Java RMI 实现分布式应用 …… 324
实例 50　Java 程序中的多线程实现 …… 328
实例 51　在 Java 程序中运行外部类文件 …… 332
实例 52　在 Java 中执行外界程序 …… 335
实例 53　Java 类库中的设计模式 …… 337
实例 54　用 Visual J++ 实现 COM 组件 …… 343
实例 55　用 C/C++ 实现 Java 的本地方法 …… 348
实例 56　利用 RMI 实现 Java 分布式应用的方法与实例 …… 356
实例 57　在 Java 程序中处理异常 …… 362
实例 58　Java 程序的多线程机制 …… 365
实例 59　灵活运用 VRML-Java 创建三维世界 …… 369

实例 60　利用 Java 的多线程技术实现并发多任务的管理 …… 373
实例 61　Java 实现跨平台的代理服务器及其计费 …… 380
实例 62　实现 Java 的动态类载入机制 …… 384
实例 63　Java Servlet 中的 Cookie 及其在口令验证中的应用 …… 389
实例 64　深入 Java 编程 - Java 的字节代码 …… 396
实例 65　Java Servlet 中对模板文件的处理 …… 404
实例 66　基于 Visual J++6.0 的网络浏览器开发技术 …… 408

# 第 1 章

# Java 编程基础与应用

# 实例 1

# Java 和 ASP 的结合:Applet 小程序从 Form 表单中动态提取参数

## 一、Java 及 Java Applet 概述

Java 程序主要有两种:应用程序(Application)和小程序(Applet)。应用程序直接由 Java 解释器解释执行,而小程序则要嵌入到 Web 页面中,当浏览器显示 Web 页面时,自动解释并运行 Java 小程序。Java 小程序是 Java 的主要应用之一,它是 WWW 上神通广大的多面手,可以用来实现 Internet 上的各种应用,而且与平台无关。

Java Applet 是嵌入 HTML 语言并通过主页发布到 Internet 的 。网络用户访问服务器的 Applet 时,这些 Applet 从网络上进行传输,然后在支持 Java 的浏览器中运行。由于 Java 语言的安全机制,用户一旦载入 Applet,就可以放心地来生成多媒体的用户界面或完成复杂的计算而不必担心病毒的入侵。

## 二、Java Applet 的"静态"参数提取过程

我们已经知道,Java Applet 是嵌入 HTML 语言并通过主页发布到 Internet 的。将小程序嵌入到 HTML 文档中,必须使用 < Applet > 标记。这个标记定义了一个在网络文档中运行的小程序,Java Applet 就靠此 < Applet > 标记中定义的内容来启动。不支持 Java 小程序的浏览器将忽略这个标记及有关的 < Param > 标记,支持 Java 小程序的浏览器则运行此处指定的 Java 小程序。

表 1-1 所列为 Applet 标记的主要参数,它们的用法在此不多加详述。

**表 1-1 Applet 标记的格式**

| 标记 | 说 明 |
|---|---|
| CODEBASE | 小程序的 URL。当小程序的主类文件的位置不同于 HTML 文档的位置时,浏览器会根据 CODEBASE 提供的路径来确定小程序的主类文件的位置,这个位置不一定在 HTML 文档所在的机器上 |
| CODE | 小程序的主类文件名,该文件名带有 .class 扩展名。如果这个主类文件和 HTML 文档不在同一个目录下,就需要用 CODEBASE 指定主类文件的位置 |
| WIDTH | 小程序在浏览器中的初始宽度,以像素为单位 |
| HEIGHT | 小程序在浏览器中的初始高度,以像素为单位 |

续表

| 标记 | 说　明 |
| --- | --- |
| ALT | 替换的信息。当浏览器不支持 Java 小程序但是能够识别 < Applet > 标记时，将在 Web 页面上显示此处定义的替换信息 |
| NAME | 小程序的实例名。在同一 Web 页上显示的其他小程序通过这个名字和小程序进行通信 |
| ALIGN | 定义小程序在 Web 页面上的对齐方式。它的可能值为 left，right，top，texttop，middle，absmiddle，baseline，bottom，absbottom。它的默认值为 bottom。这些值的作用和 < IMG > 标记中的一样 |
| VSPACE | 小程序在 Web 页面中上下保留的空白像素数。该属性仅当 Align 标志为 top 或 bottom 时使用 |
| HSPACE | 小程序在 Web 页面中左右留出的空白像素数。该属性仅当 Align 标志为 left 或 right 时使用 |
| PARMA，NAME，VALUE | 带有 NAME 和 VALUE 属性值的 < Param > 标记定义了一个参数的名称和值。小程序内部可以从 Web 页面中得到此参数。NAME 表示参数名，VALUE 表示参数值 |

之所以称 Java Applet 程序常用的参数传递为“静态”传递，完全是因为以往的应用都是在 Web 页面中直接向 Java Applet 传递参数，即使用 HTML 的标记 < Applet Param > 参数直接设定。可见这样的初始化工作是预先设计好的。

Java Applet 的“静态”参数传递过程如图 1-1 所示：在 Web 页面这一端，通过使用 < Param > 参数，设置参数。而在 Java Applet 这一端则通过调用 getParameter( )函数来获取参数值。

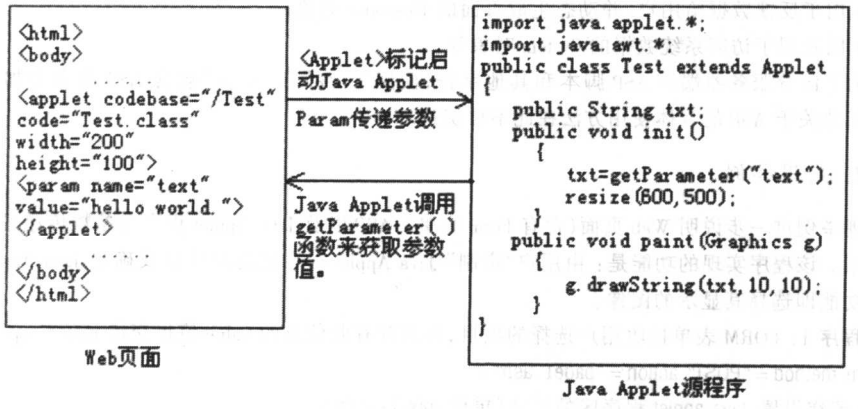


图 1-1

## 三、Java Applet 的“动态”参数提取

所谓“动态”，也就是说在 Web 页面中并没有预先对 Java Applet 程序所需要的参数进行设置，甚至连 < Applet > 标记都找不到。

CGI，ASP，Java 等都是 Internet 网上编程的语言，它们各有各的优势。但你是否尝试过将其综合起来应用，更大限度地发挥它们的潜力呢？本例就主要介绍了一种 Java Apllet 如何与 ASP 相结合，从 Form 表单中动态提取参数 Param 的方法，使 Java Applet 在其初始化时就能够与用户进行交互。

### 1. ASP 的引入

在 Microsoft 的 Web 服务器 IIS(Internet Information System )的众多组件中，有一个称为 Active Server Page(以下简称 ASP)的服务器端的脚本引擎，用于解释执行运行于服务器端的 VBScipt 或 JavaScript 脚本。ASP 不是编程语言，它是一套服务器端的对象模型，用户可以通过代码访问 ASP 对象的各种方法和属性，来操作服务器端的数据。

使用 ASP 技术可以将用户提交的数据在服务器端操作处理，并且将操作的结果以 HTML 代码的格式，发送到用户的浏览器上，由浏览器解释、显示操作结果。

正是由于“ASP 可以将用户提交的数据在服务器端操作处理，再将操作的结果以 HTML 代码的格式发送到用户的浏览器上，由浏览器解释、显示操作结果”，我们才可以使得 Java Applet 在其初始化过程中与用户交互，由 Form 表单动态地提取参数。

ASP 中包含了五个内建的对象：

- 用于在一个用户的多个页面之间共享数据的 Session 对象；
- 在同一应用的多个用户之间共享信息的 Application 对象；
- 用于操作用户提交数据的 Request 对象；
- 用于反馈数据给用户，并动态生成页面的 Response 对象；
- 以及用于访问系统数据的 System 对象等。

为了区分服务器端的 ASP 脚本和其他字符，使用“ < % ”和“ % > ”包含 ASP 的命令加以区别。其他关于 ASP 的具体使用方法在此不再赘述。

### 2. 应用举例

现举例进一步说明 Web 页面(含有 Form 表单)，ASP 以及 Java Applet 程序在参数传递过程中的联系。该程序实现的功能是：由用户“定制” Java Applet 程序区的尺寸以及确定 Java Applet 程序的功能即选择其显示的图像。

**程序 1**：FORM 表单接收用户选择的项目，并将存有此信息的 value 值提交给 page1.asp 文件

```
< form method = "POST" action = "page1.asp" >
< p > 请您设置 Java applet 程序区的尺寸(单位:pix ): < /p >
< p > Width: < input type = "text" name = "T1" size = "20" > < /p >
< p > Height: < input type = "text" name = "T2" size = "20" > < /p >
< p > 请选择 Java Applet 程序显示的图像: < /p >
< p > < input type = "radio" value = "V1" name = "R1" > Picture1 < /p >
```

```
< p > < input type = "radio" value = "V2" checked name = "R1" > Picture2
   < input type = "submit" value = "Submit" name = "B1" >
   < input type = "reset" value = "Reset" name = "B2" > < /p >
< /form >
```

**程序 2**：ASP 文件接收 value 值，并借此“定制”HTML 文件的 < applet > 标记，实现“动态“设置 Java Applet 程序的参数

```
< % @ language = VBScript % >
< html >
< % str1 = request.form("T1")
  str2 = request.form("T2")
  pic1 = request.form("R1")
  if pic1 = "V1" then
     pic = "1.gif"
  else
     pic = "2.gif"
  end if
% >
< body >
< p > your select picture is < % = pic% > < /p >
< p >
< applet codebase = "/Test" code = "Test.class" Width = " < % = T1% > " height = " < % = T2% > " >
< param name = "picture" value = " < % = pic% > " >
< /applet >
< /p >
< /body >
< /html >
```

**程序 3**：Java Apple 程序调用 getParameter()函数来获取参数值

```
import java.applet.*;
import java.awt.*;
public class Test extends Applet
{
  public String num;
  Image img = null;
  public void init()
  {
    num = getParameter("picture");
    img = getImage(getDocumentBase(),num);
  }
  public void paint(Graphics g)
```

```
    g.drawImage(img,10,10,this);
  }
}
```

（李艳梅　朱秋萍　方志豪）

# 实例 2

## Servlet 中汉字处理浅析

目前 Java 的应用状况是服务器端的应用比客户端普及，而在服务器端的一项重要应用就是用 Servlet 代替 CGI 开发应用服务器，比如访问数据库等。Servlet 与 CGI 相比优势更为明显，这主要体现在两个方面：一是 Servlet 支持多线程，而不是像 CGI 一样以多进程的方式运行（虽然目前 fast CGI 正在改进这一问题），能够节省更大的系统资源，提高系统的运行效率；而是 Java 的平台独立性，这在系统平台复杂的情况下具有无可比拟的优势，比如用 Solaris 下的 Apache 作为 Web 服务器，而数据库服务器是建立在 NT 的 SQL Server 上。此时不用 Servlet 实现就有很大的不便。

然而，目前 Servlet 对汉字的支持还有很多问题，可能是 Servlet 内部对多语言的支持上存在问题，也可能是 Servlet 引擎（Servlet Engine）对多语言的至此存在问题，（笔者在 JRUN 2.3.3 免费版和 Sun JSWDK 1.0 下多发现对汉字支持的异常），笔者在工作中遇到过 Servlet 中汉字显示和传递的不正常的问题。笔者通过尝试，找到了一些克服的办法，也有一些问题没有彻底解决。

Servlet 对汉字的支持异常主要体现在如下 3 个方面：Servlet 向浏览器传递汉字信息，浏览器向 Servlet 传递汉字信息；servlet 通过 JDBC 向数据库服务器传统信息。

分析发现，Servlet（包括 Java）对多语言的支持是通过设定编码方式（Character Encoding）实现的，不同的语言对应不同的编码方式。因为 Java 中的字符类型（Char）和字符串类（String）是以独立于特定平台的双字节 Unicode 码表示的，而输入输出流（Stream）是以二进制字节为单位的，二者的转换都要针对特定的编码方式，相同的字节流（或 Byte 数组）按照不同的编码方式转换成的字符或字符串一般并不相同，反之亦然。这样，如果系统没有正确地识别系统的编码方式，就会造成这种转换异常，导致汉字无法正常显示。这种 Byte 和 Char 转换主要在两种情况下出现，①用字节数据构造字符串或从字符串对象获取字节数组时；②发生在字节流（Input/OutputStream）和字符流（Reader/Writer）的转换过程中，涉及到 InputStreamReader/OutputSreamWriter 类。两种转换过程中，都可以使用系统默认的字符编码方式，或显示指定。第一种情况，比如：

```
String msg = "汉字显示";
byte[] buf = msg.getBytes("gb2312");
String isomsg = new String(buf, "iso - 8859 - 1");
```

上面的三行代码将汉字字符串转换为西文编码的字符串，自然，是一串乱码。后一种情况比如：

```
FileInputStream fin = new FileInputStream("./myfile.txt");
InputStreamReader inr = new InputStreamReader(fin, "gb2312");
```

上面两行代码从文件中取出字节流，然后转换为字符流，采用汉字编码。InputStreamReader 和 OutputStreamWriter 的字符编码方式可以通过 getEncoding()函数得到。

Java 虚拟机一般能够正确地识别出当前系统的字符编码方式，因此一般的 Java 应用程序和

Applet 都不会出现汉字乱码的问题；而 Servlet 是通过 Servlet 引擎运行的，而有的 Servlet 引擎不能很好地识别操作系统的运行环境，因此出现汉字显示的异常。笔者发现 JRun 和 Sun JSWDK 都存在这种问题。Java 系统的默认字符编码方式可以通过系统的 file.encoding 属性设置，但设置确不能影响 Servlet 的运行。系统的默认编码方式可以如下设定：

```
System.setProperty("file.encoding", "gb2312");
```

基于上面的分析，下面介绍 Servlet 汉字传递异常的解决办法。

### 1.从 Servlet 向浏览器传递中文信息

采用默认的字符编码，向浏览器输出中文信息会不能正确显示。这个问题最容易解决，只要向相应对象(Http ServletResponse)中指定汉字编码信息即可，即

```
public void doGet/doPost(HttpServletRequest req, HttpServletResponse res){
  ...
  res.setContentType("text/html; charset = gb2312");
// 或者
// res.setContentType("text/html");
// res.setHeader("charset", gb2312);
  ...
}
```

如此设定以后，如果用相应对象(HttpServletResponse)的 getEncoding()方法查看系统的编码方式，就会发现系统已经正确设置了。

### 2. 在 Servlet 得到浏览器的中文输入

这种情况较为复杂。采用 HttpServletRequest 对象的默认方法，比如 getParameters()，getParameterValues()得到的从表单输入中文信息，都是经过系统默认的字符编码方式转换的字符串。由于转换没有基于汉字编码，所以通过上述函数无法得到正常的汉字信息。笔者发现，无法通过设定系统的默认编码方式解决这一问题(即设置 file.encoding 属性)，因此只能通过获取 Http 请求对象(HttpServletRequest)的字节流信息，然后按照汉字编码方式转换为字符串，然后解码，并从中抽取出输入参数。对于 post 方式传递的表单信息，以 ServletStream 的 getContent()方法获取输入流；对于 get 方式传递的参数，直接用请求对象的 getQueryString()方法获取。采用这种方法，其缺点是不如 getParameter()和 getParameterValues()函数获取参数方便，为克服这种缺点，笔者编写了一个表单中文信息提取程序(ChnRequestBroker)，实现了和系统类似的界面(代码附后)。使用过程中只要将 Http 请求对象作为参数生成一个 ChnRequestBroker 实例，然后调用其 getParameter()或 getParameterValues()方法提出参数即可。比如：

```
public void doGet(HttpServletRequest req, HttpServletResponse res){
  ...
  ChnRequestBroker chnBrk = new ChnRequestBroker(req);
  String parm = chnBrk.getParameter("Name");
  ...
}
```

### 3.JDBC 查询语句中汉字传输的异常

数据库查询是 Servelt 的一个重要应用领域,但是 Sun 开发的 JDBC API 中的一个 Bug 导致无法正常传递汉字信息。这个 Bug 存在于 sun.jdbc.odbc.JdbcOdbcObject 类中,在 Sun JDK 1.2 和 1.1 版本中都有表现。Inspris 公司开发的 JBuilder 中, JdbcOdbcObject 类的实现和 Sun 公司的实现不同,不存在这一问题。该 Bug 表现在通过 Statement 类的 executeQuery(sql)和 executeUpdate(sql)执行数据库操作时,如果 SQL 语句中包含中文信息,比如字段名和字符串常量,将会导致 SQL 执行失败。这一 Bug 的是由于 JdbcOdbcObject 类将 SQL 语句从字符向字节转换过程中,调用了错误的计算字节数的函数引起的(据 Sun 网站 http://developer.java.sun.com/developer/bugParade/index.html)。Bug 的排除只要将 JdbcOdbcObject 类的 CharsToBytes(String, char[])中计算字节数的两条语句修改即可实现:即将 CharToByteConverter 实例的 netCharIndex()调用修改为 nextByteIndex()。代码如下所示:

```
// Bug removed according SUN's suggestion
//byte abyte2[] = new byte[abyte0.length + chartobyteconverter.nextCharIndex()];
byte abyte2[] = new byte[abyte0.length + chartobyteconverter.nextByteIndex()];
System.arraycopy(abyte0, 0, abyte2, 0, abyte0.length);
//System.arraycopy(abyte1, 0, abyte2, abyte0.length, hartobyteconverter.nextCharIndex());
System.arraycopy(abyte1, 0, abyte2, abyte0.length, chartobyteconverter.nextByteIndex());
abyte0 = new byte[abyte2.length];
System.arraycopy(abyte2, 0, abyte0, 0, abyte2.length);
```

实际操作过程中,只要从类压缩包 classes.zip(JDK 1.1x)或 rt.jar(JDK 1.2x)中将 sun.jdbc.odbc.JdbcOdbcObject.class 释放出来(用 JDK 的 jar 或 winzip),如果有 Java 反编译工具,将该类编译、排除错误、重新编译,然后将原来的类替换即可(采用 jar uvf)。

```
import java.io.*;
import java.net.*;
import java.util.*;
import javax.servlet.http.*;
import javax.servlet.*;
public class ChnRequestBroker
{
  private HttpServletRequest request = null;
  private boolean ready = false;
  private int numItems = 0;
  private String parameters[], values[];
  public ChnRequestBroker(ServletRequest req){
    setRequest(req);
  }
  public void setRequest(ServletRequest req){
    request = (HttpServletRequest)req;
    ready = false;
```