网络技术 NETWORK TECHNOLOGY

基于 Java 的 Web 聊天室系统

余海燕 郑笑飞

摘 要 本文讨论了用 Java 开发 Web 聊天室系统的优点，并给出了一个 Java 聊天室系统的实例。作者解决了其中的中文传输、显示及输入的问题。该聊天室系统具有高效性、健壮性和灵活性，达到了预期的设计目标。

关键字 WEB 聊天室，JAVA，UTF－8，异常，哈希表，广播

一、概述

聊天室（Chat room）是 Web 站点提供的常用服务之一，它给网络用户带来了在线实时交流的机会，而且使用起来不需要安装专门的聊天软件，只需要浏览器即可。Web 聊天室系统由于其方便、灵活和易于使用的特点而广受欢迎。目前开发 Web 聊天室系统的方法主要有以下几种：CGI，Java，ActiveX，ASP 等等。相比较起来，Java 语言具有其优越之处[1]：一是跨平台和可移植性好。Java Application 和 Java Applet 程序几乎能在所有平台上编译、执行，而象含 ActiveX 的页面主要针对 x86 的 Win32 系统，且在 Netscape 浏览器中运行时需额外安装 plug－in。二是使用 Java 语言编写的聊天室能够做到真正的实时聊天。常见的 CGI、ASP 等方法一般是通过无连接的 HTTP 协议来传输数据，需要靠 HTML 页面的自动定时刷新来模拟聊天过程。而 Java Applet 能够与服务器端建立永久的 TCP/IP 连接，用户的发言能够被马上传输和广播，而且也不需要传输额外的 HTML 内容。三是 Java 语言本身的功能非常适合于编写网络应用程序，如鲁棒性（Robust），完善的 Net 类库和多线程支持等等。本文基于 Java 语言来开发一个完整的 Web 聊天室系统。

二、结构与目标

基于 Java 的 Web 聊天室系统包括聊天服务器和客户端两部分。聊天服务器是一个 Java Application，与 Web 服务器程序运行在同一机器上。客户端部分即是一个含 Java Applet 的 HTML 页面，它由 Web 服务器传送给客户端浏览器，交由浏览器的 Java 虚拟机（VM）解释执行。该 Applet 初始化后与聊天服务器进行连接，聊天服务器对于每个连接请求产生一个连接线程（Connection Thread），来维护和管理与该客户端的会话。客户端的发言被传送到服务器端后由其向其他客户进行广播（Broadcast），达到相互聊天的目的。Java 聊天室系统结构如图 1 所示：

为了让这个聊天室系统能够真正实用，必须达到以下要求：

1. 完善的支持中文。由于 Java 编译器版本及运行环境的差异等原因，在 Java 语言的中文处理中常出现乱码等现象，表现在中文显示与网络传输不正常、中文不能输入等方面。在这个聊天室系统中我们将彻底解决中文的兼容性问题。

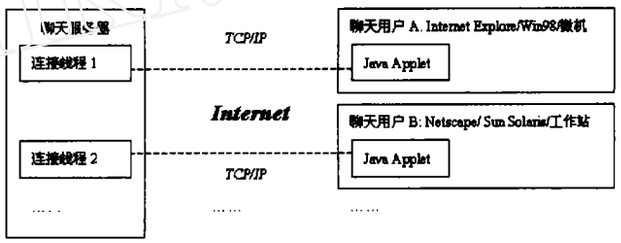

图 1 Java 聊天室的结构

2. 具有健壮性。即聊天室系统能够处理各种异常，能够识别和控制客户端的各种行为，能够返回清理不正常退出后所分配的系统资源，能够踢出超时连接用户以减轻服务器负载等。虽然 Java 语言本身能够自动收集处理无用的对象[2]，但我们仍然需要作一定的清理工作。

3. 广泛的适应性。因为我们不能要求聊天用户必须使用某种浏览器或操作系统，因此所编写的 Java 程序，尤其是 Applet，必须能在各种平台的各个版本的浏览器上都能正常运行。考虑到网络用户的使用情况，我们定的标准是能适应以下版本的浏览器：Netscape 3.x，Netscape Communicator 4.x，Internet Explorer 3.x、4.x、5.x 中英文版。

三、聊天服务器

在聊天服务器中，我们使用哈希表（Hashtable）来存储所有的连接线程。主线程为 ChatServer，对于每个新的客户连接请求产生一个 Connection 线程。同时我们还运行了一个检查线程 CheckActiveTimer，它相当于一个定时器，每隔一定时间就扫描所有的客户连接线程（即扫描 Hashtable），检查每个客户连接是否超时（例如很长时间没有发言或者死机），并给出警告或直接踢出（Kick）用户。

在网络传输过程中，我们使用字节输入输出流 DataInputStream 和 DataOutputStream 的 writeUTF、readUTF 方法进行传送接收，这两个方法以 UTF－8 编码方式来对 Unicode 字符串进行编码和解码，这样我们就能正确的进行中英文的传送。虽然在 JDK 1.1 以上版本编译器中，我们可以使用基于字符的流，如 BufferdReader、PrintWriter 等类进行网络传输，但是经过试

验，在某些旧的 Java VM 上（如 IE4 所带的 VM）仍然出现一些汉字传输后变成?字符的现象，所以我们不采用这种方式。

聊天服务器的主要程序段代码及其解释如下：

(1)主线程 ChatServer 的定义及初始化

```
public class ChatServer extends Thread {//主线程 ChatServer
    public Hashtable chatusers;
//存储与所有聊天用户的连接线程
    protected ServerSocket listen_socket; //监听 Socket
    protected CheckActiveTimer check;
//定时检查用户活动情况的线程
    public final static int PORT = 6543; //默认端口号
    public final static long WARNTIMEOUT = 180000l;
//超过 3 分钟用户无反应则警告
    public final static long KICKTIMEOUT = 240000l;
//超过 4 分钟用户无反应则被踢出
    //类初始化过程
    public ChatServer()    {
        chatusers = new Hashtable(); //分配哈希表
     check = new CheckActiveTimer(this); //启动检查线程
        try  { //创建监听 Socket
            listen_socket = new ServerSocket(PORT);  }
    catch(IOException e) {  fail(" 创建监听 Socket 失败");
}
        this.start();  //启动线程的执行
    }
```

(2)主线程的执行过程 run()

```
public void run(){
    try{ while(true){
        Socket client_socket = listen_socket.accept();
                //创建与客户端连接线程
        Connection c = new Connection(client_socket, this);
            }}
        catch(IOException e)
        {   error("与新登录用户连接失败");    }
    }
```

(3)检查连接用户活动情况的方法 checkit()

实际上，ChatServer 的 checkit()方法是在 CheckActiveTimer 线程里被定时调用的。

```
public void checkit()  {
    Vector kill = new Vector();
//用 Vector 存储要被踢出的超时用户
    long now = (new Date()).getTime();
//取现在的时间,单位:毫秒
    Enumeration e = chatusers.keys();
// 遍历连接线程,检查每个用户的上次发言时间
    while(e.hasMoreElements()){
        String key = (String)e.nextElement();
        Connection c = (Connection)chatusers.get(key);
        long inactive = now - c.lasttime;
    if(inactive> KICKTIMEOUT){//连接超时大于踢出时间
            kill.addElement(c);
        }else if(inactive> WARNTIMEOUT)
//连接超时大于警告时间
            {
            try{  //向该用户发出警告消息
        c.out.writeUTF("注意:您将在" + ((KICKTIMEOUT -
inactive)/1000) + "秒后被踢出聊天室");
        c.out.flush();
        }
    catch(IOException  e1)  {  error("警告用户"+
c.logname + "失败");  }}}
// 检查完毕后,踢出超时用户
    for(int i = 0; i < kill.size(); i + +){
        Connection c = (Connection)kill.elementAt(i);
        log("准备踢出用户" + c.logname);
        try{ c.client_socket.close();    }
        catch(IOException e2)    {
            error("无法踢出用户:" + c.logname);  }}}
```

(4)检查用户是否超时活动的线程 CheckActive

```
class CheckActiveTimer extends Thread{
    ...
public void run()  {
    while(true)  {
        if(server! = null) server.checkit();
        try{ sleep(PERIOD); }//线程暂时中止 PERIOD 时间
    catch(InterruptedException e) { System.err.println(" 错
误:检查线程被中止"); }
    }}}
```

(5)处理同客户端连接的线程 Connection

```
class Connection extends Thread{
public Socket client_socket;
public long lasttime; //上次活动时间
protected DataInputStream in;    // 读缓冲流
protected DataOutputStream out;  // 写缓冲流
public String hashkey; // 标志此 Connection 的字符串
public String username; // 聊天用户代号
public String logname;
protected Hashtable chatusers; //引用 ChatServer 的 chatusers
public Connection(Socket client_socket, ChatServer server) {
// 初始化
    this.client_socket = client_socket;
    chatusers = server.chatusers;
    lasttime = (new Date()).getTime();
    username = "未知";
    try{
 in = new DataInputStream(client_socket.getInputStream());
out = new DataOutputStream(client_socket.getOutputStream());
    }catch(IOException e){
System.err.println("错误:获取客户端 Socket 流时发生错误");
        try{ client_socket.close();}
        catch(IOException e2)    {
    System.err.println("错误:无法关闭客户端 Socket");
        }
        this.stop(); //停止线程
        return;
        }
     hashkey = client_socket.getInetAddress() + ":" + client_
socket.getPort();
    logname = "[" + hashkey + "/" + username + "]";
    chatusers.put(hashkey, this); // 加入用户列表
    this.start();
}
public void run()  {
    String line;
    boolean command;
    try{ for(;;){
            command = false;
            line = in.readUTF();    //读入一行
            if(line = = null) break;
```

```
                lasttime = (new Date()).getTime();
        if(line.startsWith("$username")) { //用户登录名字
                username = line.substring(9);
        logname = "[" + hashkey + "/" + username + "]";
            broadcast(username + "进入聊天室，目前聊天室
用户人数为" + chatusers.size());
                command = true;
            }
            if(line.startsWith("list ")) { // 列用户命令
                Enumeration e = chatusers.keys();
                String msg = "目前在聊天室的用户有: \n";
                while(e.hasMoreElements()) {
        String key = (String)e.nextElement();
        Connection c = (Connection)chatusers.get(key);
                    msg = msg + c.username + "\n";
                }
                out.writeUTF(msg); out.flush();
                command = true;
            }
if(!command) broadcast(username + ": " + line); //进行广播
        }}
        catch(IOException e){
System.err.println("错误: 与" + logname + "通信时发生错误");
        }
        finally { // 与客户端通信失败后，资源清理工作
            try{
                client_socket.close(); //关闭 Socket
      System.out.println("关闭与" + logname + "的连接");
            }
            catch(IOException e){
System.err.println("错误: 不能关闭与" + logname + "的连接");
            }
            finally{ //最终删除线程
                client_socket = null;
        chatusers.remove(hashkey); //从哈希表中删除
                try{
        System.out.println(logname + "离开了聊天室。");
            broadcast(username + "离开了聊天室。");
                }catch(IOException e){
                    System.err.println("错误: 广播错误");
                }
                this.stop();
            }}}
public void broadcast(String s) throws IOException//广播过程
    {
        Enumeration e = chatusers.keys();
        while(e.hasMoreElements()) {
            String key = (String)e.nextElement();
        Connection c = (Connection)chatusers.get(key);
            c.out.writeUTF(s);
            c.out.flush();
        }}}
```

四、客户端的 Java Applet

当含有 Applet 的页面传送到客户的浏览器时，Applet 被解释执行。Applet 有一个参数（PARAM）为 username，代表聊天用户的代号。当 Applet 初始化同服务器进行连接时，发送一条形式如“$username...”的消息，告诉服务器聊天用户的名字。聊天用户可以输入 list 命令来列出当前聊天室的用户（参见前面的 Connection 线程的代码）。客户端 Applet 启动了一个 StreamListener 的线程专门用于从服务器获取消息并显示在 Applet 的发言区。

虽然我们可以正确的传输与显示中文，但在 Java Applet 的 Text 类组件中不能输入中文的现象仍然存在，尤其是在 IE 浏览器上。这个问题产生的根源不是 Java 语言本身或者我们所编写的代码不正确，而完全是 Microsoft VM for Java 的 Bug。对于使用 IE 的用户来说，在 Web 上的 Java 聊天室输入中文是麻烦的，需要通过拷贝和粘贴方式把汉字贴入 Applet 的 TextField。对于这个问题，我们可以使用一个基本的技巧来解决。那就是不使用 Java 的 TextField 或 TextArea 类进行输入，而是依靠 HTML 页面中的表格（Form）的 Text 元素进行输入，然后把输入结果传递给 Applet。传递的方法是在 HTML 页面里加入 JavaScript 程序，与该页面的 Applet 进行通讯。

对于客户端 Applet 来说，进行必要的资源清理工作尤为重要。在 HTML 页面的生存期内，Applet 执行的顺序一般是：init()-->start()-->stop()-->destroy()。所以我们必须在 Applet 的 stop()方法中进行资源清理工作，此外，当与聊天服务器连接中断（如服务器 down 机或因超时被服务器踢出时），也要进行相应的清理工作。这样，客户端的系统资源不会丢失（如内存泄漏 leak），而且服务器端也能够迅速掌握客户端的状态。

客户端的主要代码段及解释如下：

1. 登录页面 default.asp

该页面将登录用户的用户名动态地传递给 Applet，这是通过插入 ASP 脚本[3]实现的。如果不用 ASP，也可以使用 JavaScript 编写一个弹出的输入对话框让用户输入名字。

```
<HTML> <HEAD> <TITLE> 聊天室 </TITLE>
<script language = "javascript">
function SendIt() //将 Form 的输入传递给 Applet
{
document.AppletClient.SendToServer
(document.CHATFORM.textField.value);
document.CHATFORM.textField.value = "";
}
function ClearDisplay()
//调用 Applet 的 Clear 过程清除发言显示区
{ document.AppletClient.Clear(); }
</script> </HEAD> <BODY>
<%
If trim(request("username")) = "" then
'显示用户登录 Form
%>
<form name = "LOGINFORM" action = "default.asp">
<font style = "font-size: 14"> 输入你的代号: </font>
<input type = "text" name = "username" size = 20 style =
"font-size: 14">
<input type = " submit" value = " 进入聊天室" style = " font-
size: 14">
</form>
<%
```

```
else ´否则显示聊天 Form
%>
<APPLET CODE = " ChatClient.class" name = AppletClient
WIDTH = 500 HEIGHT = 300>
<PARAM name = "username" value = " <% = request("user-
name")%> ">
</APPLET>
<form name = " CHATFORM" onSubmit = " SendIt(); return
false; ">
<input type = " text" name = " textField" size = 40" style = "
font - size: 14">
<input type = " button" value = " 发送" onClick = " SendIt()"
style = "font - size: 14">
<input type = " button" value = " 清除显示区" onClick = "
ClearDisplay()" style = "font - size: 14">
</form>
<% end if%> </BODY> </HTML>
```

2. Applet 的定义及初始化

```
public class ChatClient extends Applet{
    public final static int PORT = 8543;
    public Socket s = null;
    protected DataInputStream in = null; // 读缓冲
    protected DataOutputStream out = null; // 写缓冲
    protected TextArea outputarea; // 显示发言的区域
    protected StreamListener listener = null;
//监听线程用于显示服务器发回的消息
    protected String username; //聊天用户名
    public void init() { //Applet 的启动过程
        super.init();
username = getParameter("username"); //获取用户名参数
        outputarea = new TextArea();
        outputarea.setEditable(false); //仅用于显示
        this.setLayout(new BorderLayout());
        this.add("Center", outputarea);
        try{ //进行连接
    //由于 Applet 的安全限制,只能同下载服务器进行连接
    s = new Socket(this.getCodeBase().getHost(), PORT);
        in = new DataInputStream(s.getInputStream());
    out = new DataOutputStream(s.getOutputStream());
        listener = new StreamListener(in, outputarea, this);
outputarea.appendText(" 连接到服务器 " + s.getInetAddress
().getHostName() + ": " + s.getPort() + "\n");
outputarea.appendText(" 使用 list 命令列出当前聊天室的用
户\n");
    out.writeUTF("$username" + username); //登录用户名
            }
        catch(IOException e){
outputarea.appendText("错误:不能与聊天服务器进行连接\n");
        }}
```

3. 提供给 JavaScript 调用的过程

```
public void SendToServer(String s) {
//此过程供网页内的 JavaScript 程序调用,将发言 s 发送到服务器
try{
Date nowTime = new Date();
//将当前时间插入发言,时:分:秒
String chattime = nowTime.getHours() + ": " + now-
Time.getMinutes() + ": " + nowTime.getSeconds();
out.writeUTF(s + " (" + chattime + ")"); }
catch(IOException e){ outputarea.appendText(" 错误: 不
能发送消息到聊天服务器\n"); }}
public void Clear()
//此过程供网页内的 JavaScript 程序调用,清除显示区
{ outputarea.setText(""); }
```

4. 线程中止 Stop()过程

```
public void stop() {
    //释放相关资源
    if(listener! = null) { //中止输入流监听线程
        listener.stop();
        listener = null; }
    try{ if(s! = null) s.close(); } //关闭 Socket
    catch(IOException e)
    { outputarea.appendText("错误:无法关闭 Socket\n"); }
    s = null;
    outputarea.appendText("释放资源\n");
    super.stop(); }
```

5. 输入流监听线程 StreamListener

```
class StreamListener extends Thread{
  ......
public void run(){
    String line;
    try {
        for(;;){
            line = in.readUTF(); //读输入
            if(line = = null) break;
            //将服务器发回信息添加到 textarea 显示区
                output.appendText(line + "\n");
        }
    catch(IOException e){}
    finally{ //清理工作
        try{
        if(app.s! = null) app.s.close(); //关闭 socket
        }catch(IOException e2)
    { output.appendText("错误:无法关闭 Socket\n"); }
        app.s = null;
output.appendText("与服务器连接断开,释放资源\n");
    }}}
```

结论

整个 Java 聊天室系统的程序在 Visual J++ 1.1 下编译运行通过，服务器端安装 NT Server 4.0 和 IIS 4.0，运行聊天服务器。客户端使用 Netscape 或 IE 浏览器进行连接。试验证明，该聊天室系统解决了在 Java 语言中的中文传输、显示及输入问题。该聊天室系统具有高效性、健壮性和灵活性，达到了预期的设计目标。读者可以在此基础上进一步增加功能，如建立私有聊天房间、用户间互送消息甚至将聊天室设为虚拟图形版等等，Java 语言将会带给程序员更多的想像与发挥余地。

参考文献

1. Introduction. JDK 1.1.6 Online Document. Sun，1997

2. Java Language Specification. Visual J++ Online Help. Microsoft，1997

3. Active Server Pages 帮助. Microsoft Developer Network. Microsoft，1998

（收稿日期：2000 年 2 月 21 日）