聊天室伺服器端ex326

ChatServerFrame

package com.zzk;

import java.awt.BorderLayout;

import java.awt.Frame;

import java.awt.Image;

import java.awt.MenuItem;

import java.awt.PopupMenu;

import java.awt.SystemTray;

import java.awt.TrayIcon;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.URL;

import java.util.Hashtable;

import java.util.Iterator;

import java.util.Set;

import java.util.Vector;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JScrollPane;

import javax.swing.JTextArea;

public class ChatServerFrame extends JFrame {

private JTextArea ta\_info;

private ServerSocket server; // 宣告ServerSocket對像

private Socket socket; // 宣告Socket對像socket

private Hashtable<String, Socket> map = new Hashtable<String, Socket>();// 用於儲存連接到服務器的使用者和客戶端套接字對像

public void createSocket() {

try {

server = new ServerSocket(1982);// 建立服務器套接字對像

while (true) {

ta\_info.append("等待新客戶連接......\n");

socket = server.accept();// 獲得套接字對像

ta\_info.append("客戶端連接成功。" + socket + "\n");

new ServerThread(socket).start();// 建立並啟動線程對像

}

} catch (IOException e) {

e.printStackTrace();

}

}

class ServerThread extends Thread {

Socket socket;

public ServerThread(Socket socket) {

this.socket = socket;

}

public void run() {

try {

ObjectInputStream ins = new ObjectInputStream(socket

.getInputStream());

while (true) {

Vector v = null;

try {

v = (Vector) ins.readObject();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

if (v != null && v.size() > 0) {

for (int i = 0; i < v.size(); i++) {

String info = (String) v.get(i);// 讀取資訊

String key = "";

if (info.startsWith("使用者：")) {// 增加登入使用者到客戶端列表

key = info.substring(3, info.length());// 獲得使用者名稱並作為鍵使用

map.put(key, socket);// 增加鍵值對

Set<String> set = map.keySet();// 獲得集合中所有鍵的Set檢視

Iterator<String> keyIt = set.iterator();// 獲得所有鍵的迭代器

while (keyIt.hasNext()) {

String receiveKey = keyIt.next();// 獲得表示接收資訊的鍵

Socket s = map.get(receiveKey);// 獲得與該鍵對應的套接字對像

PrintWriter out = new PrintWriter(s

.getOutputStream(), true);// 建立輸出流對像

Iterator<String> keyIt1 = set.iterator();// 獲得所有鍵的迭代器

while (keyIt1.hasNext()) {

String receiveKey1 = keyIt1.next();// 獲得鍵，用於向客戶端增加使用者列表

out.println(receiveKey1);// 發送資訊

out.flush();// 更新輸出緩衝區

}

}

} else if (info.startsWith("退出：")) {

key = info.substring(3);// 獲得退出使用者的鍵

map.remove(key);// 增加鍵值對

Set<String> set = map.keySet();// 獲得集合中所有鍵的Set檢視

Iterator<String> keyIt = set.iterator();// 獲得所有鍵的迭代器

while (keyIt.hasNext()) {

String receiveKey = keyIt.next();// 獲得表示接收資訊的鍵

Socket s = map.get(receiveKey);// 獲得與該鍵對應的套接字對像

PrintWriter out = new PrintWriter(s

.getOutputStream(), true);// 建立輸出流對像

out.println("退出：" + key);// 發送資訊

out.flush();// 更新輸出緩衝區

}

} else {// 轉發接收的訊息

key = info.substring(info.indexOf("：發送給：") + 5,

info.indexOf("：的資訊是："));// 獲得接收方的key值,即接收方的使用者名稱

String sendUser = info.substring(0, info

.indexOf("：發送給："));// 獲得發送方的key值,即發送方的使用者名稱

Set<String> set = map.keySet();// 獲得集合中所有鍵的Set檢視

Iterator<String> keyIt = set.iterator();// 獲得所有鍵的迭代器

while (keyIt.hasNext()) {

String receiveKey = keyIt.next();// 獲得表示接收資訊的鍵

if (key.equals(receiveKey) && !sendUser.equals(receiveKey)) {// 與接受使用者相同，但不是發送使用者

Socket s = map.get(receiveKey);// 獲得與該鍵對應的套接字對像

PrintWriter out = new PrintWriter(s.getOutputStream(), true);// 建立輸出流對像

out.println("MSG:" + info);// 發送資訊

out.flush();// 更新輸出緩衝區

}

}

}

}

}

}

} catch (IOException e) {

ta\_info.append(socket + "已經退出。\n");

}

}

}

public static void main(String args[]) {

ChatServerFrame frame = new ChatServerFrame();

frame.setVisible(true);

frame.createSocket();

}

/\*\*

\* Create the frame

\*/

public ChatServerFrame() {

super();

addWindowListener(new WindowAdapter() {

public void windowIconified(final WindowEvent e) {

setVisible(false);

}

});

setTitle("聊天室服務器端");

setBounds(100, 100, 385, 266);

final JScrollPane scrollPane = new JScrollPane();

getContentPane().add(scrollPane, BorderLayout.CENTER);

ta\_info = new JTextArea();

scrollPane.setViewportView(ta\_info);

//工作列

if (SystemTray.isSupported()){ // 判斷是否支援系統工作列

URL url=ChatServerFrame.class.getResource("server.png"); // 獲得圖片所在的URL

ImageIcon icon = new ImageIcon(url); // 實例化圖形對像

Image image=icon.getImage(); // 獲得Image對像

TrayIcon trayIcon=new TrayIcon(image); // 建立工作列圖標

trayIcon.addMouseListener(new MouseAdapter(){ // 為工作列增加鼠標介面卡

public void mouseClicked(MouseEvent e){ // 鼠標事件

if (e.getClickCount()==2){ // 判斷是否雙擊了鼠標

showFrame(); // 呼叫方法顯示窗體

}

}

});

trayIcon.setToolTip("系統工作列"); // 增加工具提示純文字

PopupMenu popupMenu=new PopupMenu(); // 建立出現選單

MenuItem exit=new MenuItem("退出"); // 建立選單項

exit.addActionListener(new ActionListener() { // 增加事件監聽器

public void actionPerformed(final ActionEvent arg0) {

System.exit(0); // 退出系統

}

});

popupMenu.add(exit); // 為出現選單增加選單項

trayIcon.setPopupMenu(popupMenu); // 為工作列圖標加出現菜彈

SystemTray systemTray=SystemTray.getSystemTray(); // 獲得系統工作列對像

try{

systemTray.add(trayIcon); // 為系統工作列加工作列圖標

}catch(Exception e){

e.printStackTrace();

}

}

}

public void showFrame(){

this.setVisible(true); // 顯示窗體

this.setState(Frame.NORMAL);

}

}