聊天室用戶端ex327

ChatClientFrame

package com.zzk;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.EventQueue;

import java.awt.Font;

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.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.net.URL;

import java.net.UnknownHostException;

import java.text.DateFormat;

import java.util.Date;

import java.util.Vector;

import javax.swing.DefaultComboBoxModel;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JList;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JSplitPane;

import javax.swing.JTextArea;

import javax.swing.JTextField;

public class ChatClientFrame extends JFrame {

private JTextField tf\_newUser;

private JList user\_list;

private JTextArea ta\_info;

private JTextField tf\_send;

private ObjectOutputStream out;// 宣告輸出流對像

private boolean loginFlag = false;// 為true時表示已經登入，為false時表示未登入

public static void main(String args[]) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

ChatClientFrame frame = new ChatClientFrame();

frame.setVisible(true);

frame.createClientSocket();// 呼叫方法建立套接字對像

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public void createClientSocket() {

try {

Socket socket = new Socket("localhost", 1982);// 建立套接字對像

out = new ObjectOutputStream(socket.getOutputStream());// 建立輸出流對像

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

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

class ClientThread extends Thread {

Socket socket;

public ClientThread(Socket socket) {

this.socket = socket;

}

public void run() {

try {

BufferedReader in = new BufferedReader(new InputStreamReader(

socket.getInputStream()));// 建立輸入流對像

DefaultComboBoxModel model = (DefaultComboBoxModel) user\_list

.getModel();// 獲得列表框的模型

while (true) {

String info = in.readLine().trim();// 讀取資訊

if (!info.startsWith("MSG:")) {// 接收到的不是訊息

if (info.startsWith("退出：")) {// 接收到的是退出訊息

model.removeElement(info.substring(3));// 從使用者列表中移除使用者

} else {// 接收到的是登入使用者

boolean itemFlag = false;// 標記是否為列表框增加列記錄，為true不增加，為false增加

for (int i = 0; i < model.getSize(); i++) {// 對使用者列表進行檢查

if (info.equals((String) model.getElementAt(i))) {// 如果使用者列表中存在該使用者名稱

itemFlag = true;// 設定為true，表示不增加到使用者列表

break;// 結束for循環

}

}

if (!itemFlag) {

model.addElement(info);// 將登入使用者增加到使用者列表

}

}

} else {// 如果獲得的是訊息，則在純文字域中顯示接收到的訊息

DateFormat df = DateFormat.getDateInstance();// 獲得DateFormat實例

String dateString = df.format(new Date()); // 格式化為日期

df = DateFormat.getTimeInstance(DateFormat.MEDIUM);// 獲得DateFormat實例

String timeString = df.format(new Date()); // 格式化為時間

String sendUser = info.substring(4,info.indexOf("：發送給："));// 獲得發送資訊的使用者

String receiveInfo = info.substring(info.indexOf("：的資訊是：")+6);// 獲得接收到的資訊

ta\_info.append(" "+sendUser + " " +dateString+" "+timeString+"\n "+receiveInfo+"\n");// 在純文字域中顯示資訊

ta\_info.setSelectionStart(ta\_info.getText().length()-1);// 設定選擇起始位置

ta\_info.setSelectionEnd(ta\_info.getText().length());// 設定選擇的結束位置

tf\_send.requestFocus();// 使發送資訊純文字框獲得焦點

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

private void send() {

if (!loginFlag) {

JOptionPane.showMessageDialog(null, "請先登入。");

return;// 如果使用者沒登入則傳回

}

String sendUserName = tf\_newUser.getText().trim();// 獲得登入使用者名稱

String info = tf\_send.getText();// 獲得輸入的發送資訊

if (info.equals("")) {

return;// 如果沒輸入資訊則傳回，即不發送

}

Vector<String> v = new Vector<String>();// 建立向量對象，用於儲存發送的訊息

Object[] receiveUserNames = user\_list.getSelectedValues();// 獲得選擇的使用者陣列

if (receiveUserNames.length <= 0) {

return;// 如果沒選擇使用者則傳回

}

for (int i = 0; i < receiveUserNames.length; i++) {

String msg = sendUserName + "：發送給：" + (String) receiveUserNames[i]

+ "：的資訊是： " + info;// 定義發送的資訊

v.add(msg);// 將資訊增加到向量

}

try {

out.writeObject(v);// 將向量寫入輸出流，完成資訊的發送

out.flush();// 更新輸出緩衝區

} catch (IOException e) {

e.printStackTrace();

}

DateFormat df = DateFormat.getDateInstance();// 獲得DateFormat實例

String dateString = df.format(new Date()); // 格式化為日期

df = DateFormat.getTimeInstance(DateFormat.MEDIUM);// 獲得DateFormat實例

String timeString = df.format(new Date()); // 格式化為時間

String sendUser = tf\_newUser.getText().trim();// 獲得發送資訊的使用者

String receiveInfo = tf\_send.getText().trim();// 顯示發送的資訊

ta\_info.append(" "+sendUser + " " +dateString+" "+timeString+"\n "+receiveInfo+"\n");// 在純文字域中顯示資訊

tf\_send.setText(null);// 清空純文字框

ta\_info.setSelectionStart(ta\_info.getText().length()-1);// 設定選擇的起始位置

ta\_info.setSelectionEnd(ta\_info.getText().length());// 設定選擇的結束位置

tf\_send.requestFocus();// 使發送資訊純文字框獲得焦點

}

/\*\*

\* Create the frame

\*/

public ChatClientFrame() {

super();

setTitle("聊天室客戶端");

setBounds(100, 100, 385, 288);

final JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.SOUTH);

final JLabel label = new JLabel();

label.setText("輸入聊天內容：");

panel.add(label);

tf\_send = new JTextField();

tf\_send.addActionListener(new ActionListener() {

public void actionPerformed(final ActionEvent e) {

send();// 呼叫方法發送資訊

}

});

tf\_send.setPreferredSize(new Dimension(180, 25));

panel.add(tf\_send);

final JButton button = new JButton();

button.addActionListener(new ActionListener() {

public void actionPerformed(final ActionEvent e) {

send();// 呼叫方法發送資訊

}

});

button.setText("發 送");

panel.add(button);

final JSplitPane splitPane = new JSplitPane();

splitPane.setDividerLocation(100);

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

final JScrollPane scrollPane = new JScrollPane();

splitPane.setRightComponent(scrollPane);

ta\_info = new JTextArea();

ta\_info.setFont(new Font("", Font.BOLD, 14));

scrollPane.setViewportView(ta\_info);

final JScrollPane scrollPane\_1 = new JScrollPane();

splitPane.setLeftComponent(scrollPane\_1);

user\_list = new JList();

user\_list.setModel(new DefaultComboBoxModel(new String[] { "" }));

scrollPane\_1.setViewportView(user\_list);

final JPanel panel\_1 = new JPanel();

getContentPane().add(panel\_1, BorderLayout.NORTH);

final JLabel label\_1 = new JLabel();

label\_1.setText("使用者名稱稱：");

panel\_1.add(label\_1);

tf\_newUser = new JTextField();

tf\_newUser.setPreferredSize(new Dimension(140, 25));

panel\_1.add(tf\_newUser);

final JButton button\_1 = new JButton();

button\_1.addActionListener(new ActionListener() {

public void actionPerformed(final ActionEvent e) {

if (loginFlag) {// 已登入標記為true

JOptionPane.showMessageDialog(null, "在同一視窗只能登入一次。");

return;

}

String userName = tf\_newUser.getText().trim();// 獲得登入使用者名稱

Vector v = new Vector();// 定義向量，用於儲存登入使用者

v.add("使用者：" + userName);// 增加登入使用者

try {

out.writeObject(v);// 將使用者向量發送到服務器

out.flush();// 更新輸出緩衝區

} catch (IOException ex) {

ex.printStackTrace();

}

tf\_newUser.setEnabled(false);// 禁用使用者純文字框

loginFlag = true;// 將已登入標記設定為true

}

});

button\_1.setText("登 錄");

panel\_1.add(button\_1);

final JButton button\_2 = new JButton();

button\_2.addActionListener(new ActionListener() {

public void actionPerformed(final ActionEvent e) {

String exitUser = tf\_newUser.getText().trim();

Vector v = new Vector();

v.add("退出：" + exitUser);

try {

out.writeObject(v);

out.flush();// 更新輸出緩衝區

} catch (IOException ex) {

ex.printStackTrace();

}

System.exit(0); // 退出系統

}

});

button\_2.setText("退 出");

panel\_1.add(button\_2);

//工作列

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

URL url=ChatClientFrame.class.getResource("client.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) {

String exitUser = tf\_newUser.getText().trim();

Vector v = new Vector();

v.add("退出：" + exitUser);

try {

out.writeObject(v);

out.flush();// 更新輸出緩衝區

} catch (IOException ex) {

ex.printStackTrace();

}

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);

}

}