

**《网络编程》作品设计**



**基于TCP的QQ聊天工具**

|  |  |
| --- | --- |
| 姓名 | **史 坤** |
| 学院 | **信息与电气工程学院** |
| 专业 | **软件工程** |
| 年级 | **2014级** |
| 学号 | **20142203717** |

2016年01月08日

作 品 真 实 声 明

本人郑重声明：所呈交的作品设计，是本人努力学习所取得的成果。本作品体现了个人学习能力、研究能力。即便有所借鉴他人成果，也都经过千辛万苦和千方百计的再学习再消化和再吸收，作品展示的知识与能力，就是本人真实的知识与能力。

作者签名:

二〇一 七 年 一 月 八 日

作 品 报 告 得 分 表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 评价项目 | 参考得分标准 | 实际得分 |
| 1 | 功能设计 | 功能描述清晰，图文并茂（10分） |  |
| 2 | 逻辑设计 | 逻辑描述清晰，图文并茂（10分） |  |
| 3 | 界面设计 | 界面描述简洁，图文并茂（10分） |  |
| 4 | 模块详细设计 | 模块设计描述详略得当，图文并茂（40分） |  |
| 5 | 系统测试 | 测试步骤清晰，逻辑完整，图文并茂（10分） |  |
| 6 | 主要问题解决过程 | 描述设计过程中困扰自己的一些关键问题及解决过程（10分） |  |
| 7 | 作品总结 | 总结作品创作历程，展望作品未来（10分） |  |
| 合计得分 | | |  |

目 录

[1 功能设计 1](#_Toc471642106)

[1.1功能概述 1](#_Toc471642107)

[1.2功能模块图 1](#_Toc471642108)

[2 逻辑设计 2](#_Toc471642109)

[3 界面设计 4](#_Toc471642112)

[3.1注册界面： 4](#_Toc471642113)

[3.2登录界面 5](#_Toc471642114)

[3.3好友列表页面 5](#_Toc471642115)

[3.4好友聊天页面 6](#_Toc471642116)

[3.5服务器界面 7](#_Toc471642118)

[4 各模块详细设计 7](#_Toc471642119)

[4.1登录模块 7](#_Toc471642120)

[4.2注册模块 9](#_Toc471642124)

[4.3聊天模块 10](#_Toc471642125)

[4.4数据库工具类 12](#_Toc471642126)

[4.5封装的数据交换类 12](#_Toc471642127)

[5 系统测试 13](#_Toc471642128)

[6 主要问题解决过程 15](#_Toc471642130)

[7 作品总结 16](#_Toc471642131)

基于TCP的QQ聊天工具

史 坤

（信息与电气工程学院，软件工程，软工1403,20142203717）

# 1 功能设计

1.1功能概述

该系统是基于JAVA语言开发的一款网络聊天工具，通过Socket实现TCP编程，使用多线程实现了多客户端的连接。模仿腾讯QQ的界面，功能较为简单，但是使用了最基本的网络编程技术，如socket、tcp、I/O阻塞、多线程、MySQL数据库等。

该系统实现了用户的注册、登录以及与好友聊天的功能。

用户注册：系统随机生成账号，用户按照要求填入相应的信息。

用户登录：用户根据自己的账号和密码进行登录。

好友聊天：登录成功后，点击好友列表中的好友进行聊天。

1.2功能模块图

该系统主要分为客户端还有服务器两个模块，具体的功能模块图如下图1.1所示。

基于TCP的QQ聊天系统

服务器模块

客户端模块

发送数据

启动服务器

消息的分发

处理注册请求

处理登录请求

判断是否在线

注册账号

用户登录

接收数据

聊天界面

图1.1功能模块图

# 2 逻辑设计

1、系统整体逻辑：

本系统由客户端响应用户的操作，通过封装对象实例化客户端，将请求发送给客户端，由客户端去执行相对应的数据库操作，将查询结果返回给服务器，服务器再执行响应的请求结果，图2.1所示

客 户 端

服 务 器

数 据 库

发送请求 查询数据库

setData

请求结果 查询结果返回到服务器

getData

图2.1系统整体逻辑图

2、注册功能实现逻辑：

用户通过客户端进行账号的申请，客户端将注册的User对象信息封装为CommandTranser对象，通过实例化一个客户端，经过sendData(CommandTranser msg)方法向服务器发送数据，服务器根据传来的注册请求和username去执行相应的数据库操作进行校验，由msg.isfalg()返回结果，如果已存在该账号，则重新进行注册，未存在该账号，则可以进行登录。如图2.2所示

查询

服务器 查询数据库

返回查询结果

false

注册请求 用户是否存在

setData()

true

客户端注册 允许并完成注册，用户登录

图2.2 用户注册逻辑图

3、登录逻辑图：

用户通过客户端进行登录，客户端将带有账号和密码的登录指令封装为CommandTranser对象，然后实例化一个客户端，经过sendData(CommandTranser msg)方法向服务器发送数据，服务器根据传来的登录请求、username和password去执行相应的数据库操作进行校验，由msg.isfalg()返回结果，存在该用户则允许登录，否则需要重新输入正确的账号密码再次进行登录。如图2.3所示

查询

服务器 查询数据库

返回查询结果

true

登录请求 用户是否存在

setData()

false

客户端登录 打开好友列表页面

图2.3 用户登录逻辑图

4、聊天逻辑图：

服务器

yes

用户是否存在

message

no

客户端登录 好友

图2.4 聊天逻辑图

# 3 界面设计

3.1注册界面：

该界面主要给用户提供注册新账号的功能。注册账号时除了自动生成的账号和输入的密码外，还需要进行身份的验证，其中包括真实姓名、性别 、手机号。注册界面如图3.1.1所示

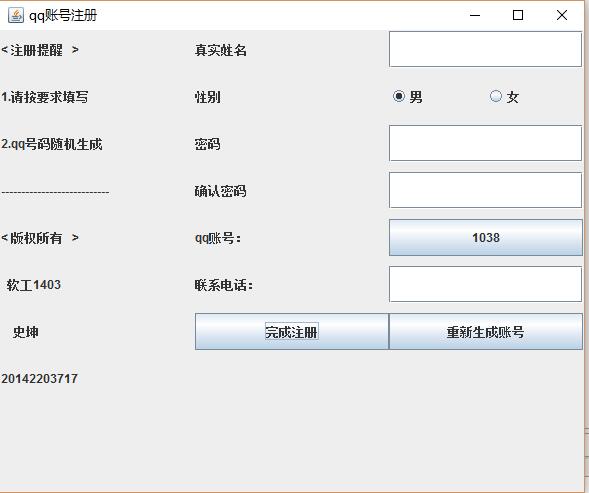


图3.1.1 用户注册页面图

在客户端对输入的信息进行了校验，不允许输入的信息为空，并给出相应的系统提示信息，如图3.1.2所示。通过客户端向服务器发送的注册请求，服务器进行数据库的校验，同样返回系统提示信息。

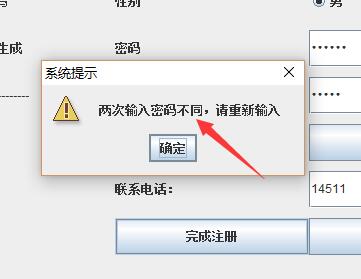
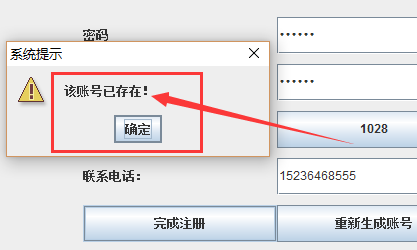


图3.1.2 系统提示图

3.2登录界面

用户输入账号和密码后，服务器会将用户输入的信息进行校验，如果账号和密码输入有误，则会产生系统提示，让用户重新进行登录。如图3.2.1和图 3.2.2所示

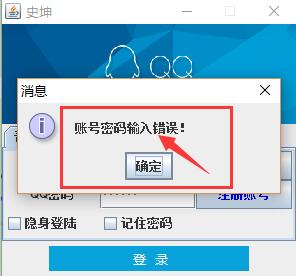
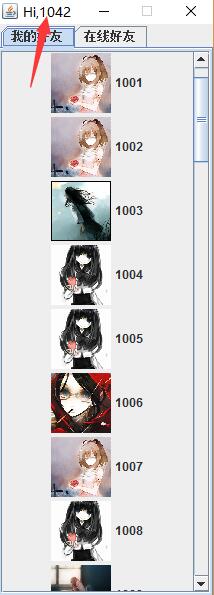
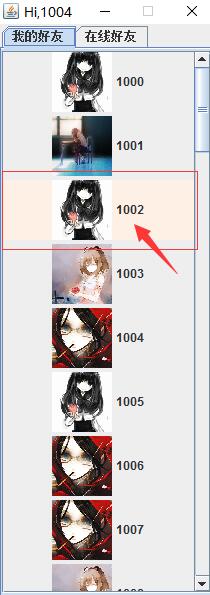


图3.2.1系统登录模块图 图3.2.2登录失败提示图

3.3好友列表页面

用户输入正确的账号和密码后，完成登录请求，并打开好友列表界面，在列表顶部展示欢迎信息，如图3.3.1所示。添加了鼠标移入移出效果，如图3.3.2所示

如图3.3.1好友列表页面 如图3.3.2好友列表页面

3.4好友聊天页面

用户在列表中双击鼠标，打开相应的聊天页面，如图3.4.1所示。在文本框中输入要发送的消息，如果用户不存在，会发出相应的系统提示，如图3.4.2所示。

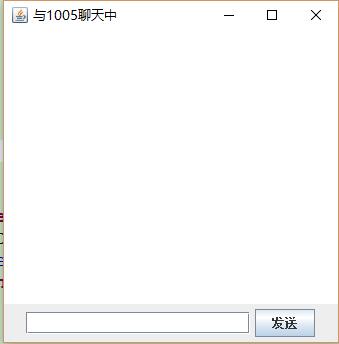
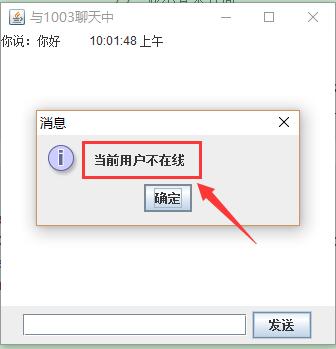
 

图3.4.1聊天页面图 图3.4.2系统提示图

鼠标双击打开两个在线的好友，可以进行私聊功能，如图3.4.3所示。

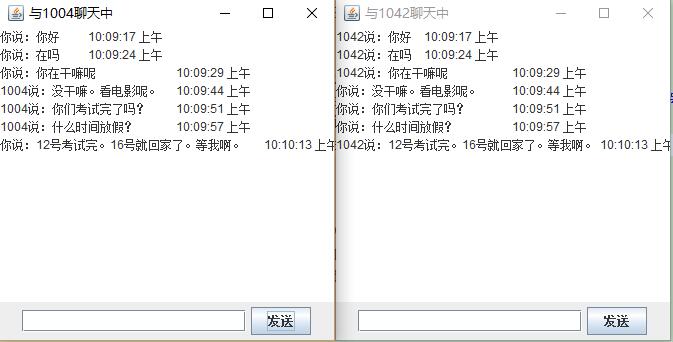


图3.4.3私聊页面图

3.5服务器界面

点击开启服务器后，新建一个线程开启服务器循环监听客户端的连接。如图3.5所示。

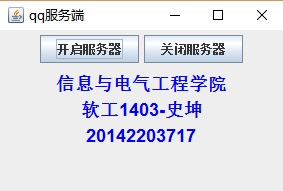


图3.5 服务器页面图

# 4 各模块详细设计

4.1登录模块

点击了登录模块后，首先判断账号和密码是否为空，然后封装为CommandTranser对象，向服务器发送数据，服务器通过与数据库进行比对来验证账号密码。

客户端获取用户输入的信息以及发送请求的代码如下：

if (e.getSource() == login) {

String username = text\_name.getText().trim();

String password = new String(text\_pwd.getPassword()).trim();

if ("".equals(username) || username == null) {

JOptionPane.showMessageDialog(null, "请输入帐号！！");

return;

} if ("".equals(password) || password == null) {

JOptionPane.showMessageDialog(null, "请输入密码！！");

return;

}

User user = new User(username, password);

CommandTranser msg = new CommandTranser();//封装msg对象发送给服务器

msg.setCmd("login");msg.setData(user);

msg.setReceiver(username);msg.setSender(username);

// 实例化客户端，并且发送数据，这个client客户端直到进程死亡，否则一直存在

Client client = new Client();

client.sendData(msg);

msg = client.getData();

if (msg != null) {

if (msg.isFlag()) {

JOptionPane.showMessageDialog(null, "登陆成功！");

new FriendsUI(username, client); // 显示好友列表界面

} else {

JOptionPane.showMessageDialog(this, msg.getResult());

}

}

}

服务器Userservice类操作数据库对登录进行验证，如果账号和密码正确，则返回true，否则返回false。实现代码如下：

public boolean checkUser(User user) {

PreparedStatement stmt = null;

Connection conn = null;

ResultSet rs = null;

conn = DBHelper.getConnection();

String sql = "select \* from user where username=? and password =?";

try {

stmt = conn.prepareStatement(sql);

stmt.setString(1, user.getUsername());

stmt.setString(2, user.getPassword());

rs = stmt.executeQuery();

if (rs.next()) {

return true;

}

}

return false;

}

如果登录成功，则将生成一个SocketThread对象，包括一个socket，一个用户账号，将该SocketThread加入到HashMap<String,Socket>集合中，并且开启此用户的接收线程。代码如下：

private CommandTranser execute(CommandTranser msg) {

if ("login".equals(msg.getCmd())) { //登录请求

UserService userService = new UserService();

User user = (User) msg.getData();

msg.setFlag(userService.checkUser(user));

/\*

\* 如果登陆成功，将该客户端加入已经连接成功的map集合里面 并且开启此用户的接受线程

\*/

if (msg.isFlag()) {

// 将该线程加入连接成功的map集合

SocketThread socketThread = new SocketThread();

socketThread.setName(msg.getSender());

socketThread.setSocket(socket);

SocketList.addSocket(socketThread);

msg.setResult("登陆成功");

} else {

msg.setResult("账号密码输入错误！");

}

}

HashMap<String,Socket>用于记录所有已经成功登录的客户端。代码如下：

public class SocketList {

private static HashMap<String, Socket> map=new HashMap<String, Socket>();

//将SocketThread入集合

public static void addSocket(SocketThread socketThread){

map.put(socketThread.getName(), socketThread.getSocket());

}

public static Socket getSocket(String name){ //通过昵称返回socket

return map.get(name);

}

}

4.2注册模块

客户端获取用户输入的信息后，将用户信息封装为User对象，将请求封装为CommandTranser对象，发送给服务器。代码如下：

User user = new User(username, password,realname,sex,phone);

CommandTranser msg = new CommandTranser();

msg.setCmd("checkregist");

msg.setData(user);

msg.setReceiver(username);

msg.setSender(username);

//实例化客户端 并且发送数据 这个client客户端 直到进程死亡 否则一直存在

Client client = new Client();

client.sendData(msg);

msg = client.getData();

服务器根据发送的请求，在UserService类中checkregistUser(User user)先进行账号的验证操作，查询该账号是否已经存在。代码如下：

//注册时，查询是否已经注册该账号

public boolean checkregistUser(User user) {

PreparedStatement stmt = null;

Connection conn = null;

ResultSet rs = null;

conn = DBHelper.getConnection();

String sql = "select \* from user where username=?";

try {

stmt = conn.prepareStatement(sql);

stmt.setString(1, user.getUsername());

rs = stmt.executeQuery();

if (rs.next()) {

return true;

}

}

return false;

}

如果返回false，则说明该账号未注册，允许用户进行注册，然后UserService类中的register(User user)完成用户的注册。

public boolean registUser(User user) {

PreparedStatement stmt = null;

Connection conn = null;

conn = DBHelper.getConnection();

String sql = "insert into user(username,password,realname,sex,phone) values (?,?,?,?,?)";

try {

stmt = conn.prepareStatement(sql);

stmt.setString(1, user.getUsername());

stmt.setString(2, user.getPassword());

......

stmt.executeUpdate();;

}

return true;

}

4.3聊天模块

一旦聊天功能启动，为每一个用户开启一个线程，通过I/O阻塞，client.getData()接收服务器发送的数据，如果服务器处理数据成功，接收消息，否则弹出该好友不在线的系统提示信息。

客户端获取消息内容、发送方和接收方，通过CommandTranser向服务器发送数据，代码如下：

public void actionPerformed(ActionEvent e)

if (e.getSource() == send\_btn) {// 如果点击了发送按钮

Date date = new Date();

SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss a");

String message = "你说：" + message\_txt.getText() + "\t"

+ sdf.format(date) + "\n";

chat\_txt.append(message);// 在本地文本区追加发送的信息

CommandTranser msg = new CommandTranser();//msg为客户端向服务器发送的数据

msg.setCmd("message");

msg.setSender(owner);

msg.setReceiver(friend);

msg.setData(message\_txt.getText());

client.sendData(msg);

message\_txt.setText(null);// 发送信息完毕 写信息的文本框设空

}

}

服务器时刻监听客户端发送的消息，并调用CommandTranser execute(CommandTranser msg)方法传入message进行处理，如果处理成功，则可以向该好友发送消息，如果服务器处理信息失败，信息发送给自己，代码如下：

public void run() {

ObjectInputStream ois = null;

ObjectOutputStream oos = null;

while (socket != null) {// 时刻监听 客户端发送来的数据

try {

ois = new ObjectInputStream(socket.getInputStream());

CommandTranser msg = (CommandTranser) ois.readObject();

msg = execute(msg);

if ("message".equals(msg.getCmd())) {//服务器处理消息

if (msg.isFlag()) {

oos = new ObjectOutputStream(SocketList.getSocket(

msg.getReceiver()).getOutputStream());

} else {

oos = new ObjectOutputStream(socket.getOutputStream());

}

}

oos.writeObject(msg);

}

}

服务器对发送的消息进行处理，判断好友是否在线，然后将msg返回。代码如下：

// 处理客户端发送的信息

private CommandTranser execute(CommandTranser msg) {

//如果是发送消息的指令，判断当前用户是否在线

if ("message".equals(msg.getCmd())) {

// 如果要发送的用户在线 发送信息

if (SocketList.getSocket(msg.getReceiver()) != null) {

msg.setFlag(true);

} else {

msg.setFlag(false);

msg.setResult("当前用户不在线");

}

}

return msg;

}

服务器将所有的操作做完后，客户端再通过getData()接收从服务器返回的数据，并在聊天界面中进行展示，代码如下：

public class ClientThread extends Thread {

private Client client;//客户端对象

public void run() {

while (isOnline) {

//I/O阻塞，接收服务端发送的数据

CommandTranser msg = client.getData();

if (msg != null) {

if (msg.isFlag()) {

Date date = new Date();

SimpleDateFormat sdf = new SimpleDateFormat(

"hh:mm:ss a");

String message = msg.getSender() + "说："+ (String) msg.getData() + "\t" + sdf.format(date)+ "\n";

chat\_txt.append(message);

} else {

JOptionPane.showMessageDialog(chat\_txt, msg.getResult());

}

}

}

}

}

4.4数据库工具类

Dbhelpher为连接数据库的工具类，用户加载数据库驱动，连接数据库。

public class DBHelper {

private static final String driver="com.mysql.jdbc.Driver";

private static final String url="jdbc:mysql://localhost:3306/qq\_program?useUnicode=true&characterEncoding=utf-8";

private static final String username="root";

private static final String password="";

private static Connection con=null;

static

{//静态块代码负责加载驱动

try {

Class.forName(driver);

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public static Connection getConnection(){

if(con==null){

try {

con=DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return con;

}

}

4.5封装交换数据类

该类主要是客户端与服务器交换的数据，其中包括发送者、接受者、传递的请求数据、服务器需要做的指令以及处理结果。

代码如下：

public class CommandTranser implements Serializable {

private static final long serialVersionUID = 1L;

private String sender = null;// 发送者

private String receiver = null;// 接受者

private Object data = null;// 传递的数据

private boolean flag = false;// 指令的处理结果

private String cmd = null;// 服务端要做的指令

private String result = null;// 处理结果

}

# 5 系统测试

1. 用户注册的测试，如果已经存在该账号，会出现系统提示，并重新生成号码并注册。否则注册成功。

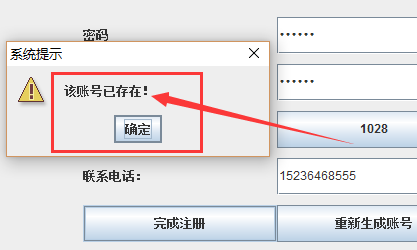
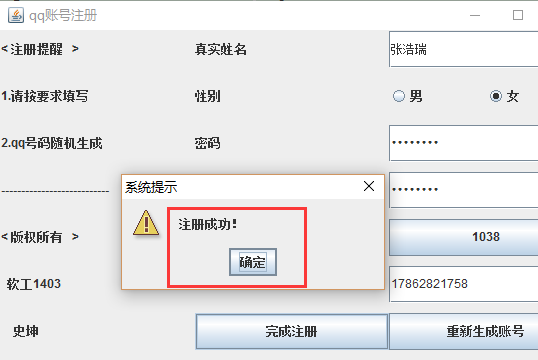


图5.1.1注册失败系统提示图 图5.1.2注册成功系统提示图

1. 用户输入账号和密码后，服务器会将用户输入的信息进行校验，如果账号和密码输入有误，则会产生系统提示，让用户重新进行登录。如图5.2.1和图 5.2.2所示

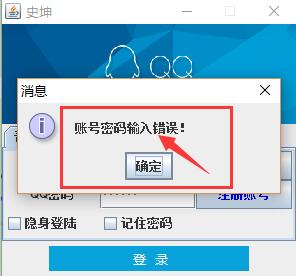
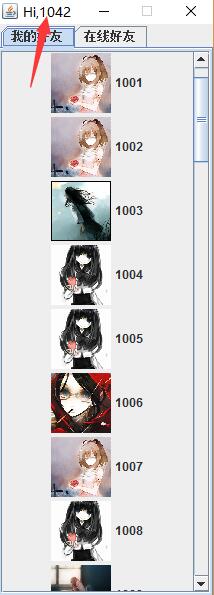
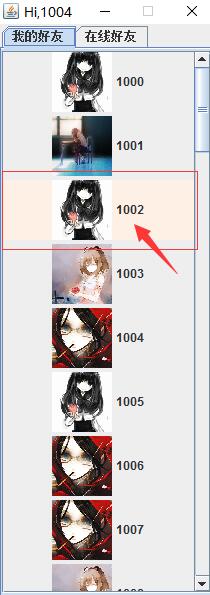


图5.2.1系统登录模块图 图5.2.2登录失败提示图

1. 用户输入正确的账号和密码后，完成登录请求，并打开好友列表界面，在列表顶部展示欢迎信息，如图5.3.1所示。添加了鼠标移入移出效果，如图5.3.2所示

如图5.3.1好友列表页面 如图5.3.2好友列表页面

1. 用户在列表中双击鼠标，打开相应的聊天页面，如图5.4.1所示。在文本框中输入要发送的消息，如果用户不存在，会发出相应的系统提示，如图5.4.2所示。

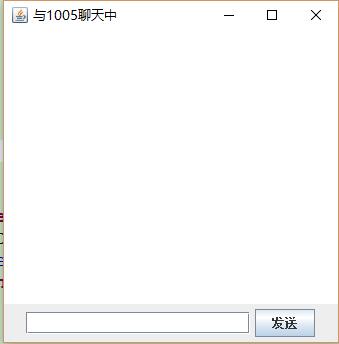
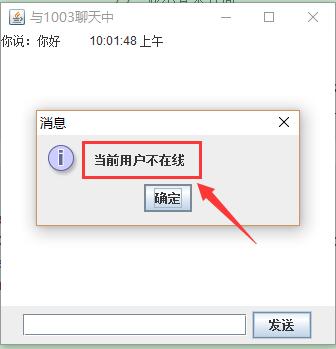
 

图5.4.1聊天页面图 图5.4.2系统提示图

鼠标双击打开两个在线的好友，可以进行私聊功能，如图5.5.1所示。

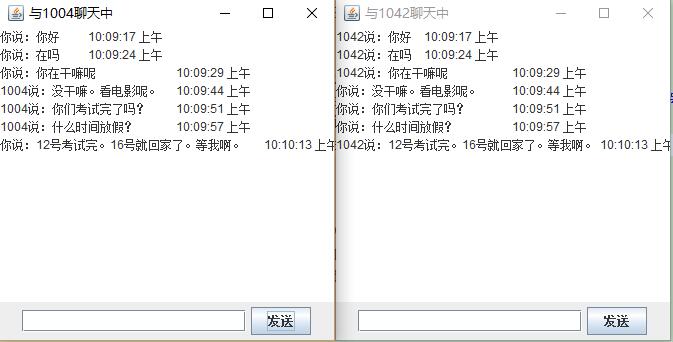


图5.5.1私聊页面图

# 6 主要问题解决过程

1. 如何判断该好友是否在线？

最后一节课的课程分享，5班的同学在他的课程设计中使用了数组对登录的用户进行了记载，0代码不在线，1代表在线。在这里，我使用了相同的处理方式，将所以已经成功登录的用户SocketThread加入到HashMap集合中，SocketThread中记录了所有成功连接的socket实体类。

1. 消息只能接收一次

使用while循环，时刻监听socket发送的数据，I/O阻塞接受客户端的消息。

1. 验证登录和验证注册的逻辑使用了同一个。

一开始写好了登录的验证，后来实现注册时，在想注册和登录不就是返回结果相反吗？登录时是通过服务器查询数据库中存在该账号才允许登录，而注册验证是查询数据库后没有该用户才允许注册，所以直接调用了同一个方法。后来注册时有时可以，有时出现问题，当时没有仔细想，后来才发现，返回结果是相反，但是验证注册不需要密码啊，只需要查询数据库中是否有该用户即可，如果使用登录的验证进行注册验证，那用户账号一样，密码不一样，同样可以允许注册了。

将注册的验证和注册的完成操作单独拿出，bug解决。

1. 如何将不同的请求进行集中操作？

Java中的泛型可以处理不同数据的传入，但是由于平时使用的不多，对于泛型并不能彻底为我所用。最后封装了一个CommandTranser对象，实例化客户端时，通过sendData(CommandTranser msg)方法向服务器发送数据，然后通过getData()方法获得数据库返回的结果。

1. 如果好友不在线，发送的消息怎么处理？

一开始自己想，只要好友不在线，弹出系统提示就足够了，不需要再将消息进行处理，可以在运行时出现了问题。没有想到好的解决办法，自己的处理方式是当好友不在线时，将该消息的发送方和接收方都设置为了自己。其实后来想想，如果实现了在线好友列表，在未发送消息之前，就可以知道好友是否在线，该问题就迎道而解。

# 7 作品总结

1. 系统整体

好几个周的努力，几天晚上的熬夜，该网络聊天工具基本算是完成，实现了用户的注册、登录以及好友之间的聊天功能。

1. 系统亮点

该网络聊天工具虽然简单，但是其中之间夹杂了不少的逻辑。一开始自己想把登录、注册的验证只在客户端与数据库之间进行，仔细一想，那不就是上学期的大作业了嘛，不能体现网络编程的思想，所以将所有的请求处理放在服务器，虽然给自己带来了很大的困难，不过总之算是完成。

1. 系统不足

好友之间的聊天，聊天内容局限性太大，内容太单一，只能进行文本消息的发送和接收，不能发送图片、文件。

网络的安全问题以及数据库的安全问题在该系统中并没有得到体现。

随机生成的账号时就确保该账号未曾注册过，而并不是单击注册之后再进行校验。

1. 系统未来展望

在好友列表界面，有一个在线好友功能，未有实现，是否可以跟随着服务器的启动与关闭，创建和删除一个在线好友列表的数据库表。

增强系统的功能性与实用性，可以实现图片、文件、语音等的发送。

加强系统的安全性，对用户信息进行加密。

本学期的网络编程课算是结束，可是学习的路程还没有结束，课上的一些知识点，还不能让自己很轻松的驾驭。但是这一学期的学习，弥补了上学期java课程中网络一章的不足，收获了很多，也接触到了很多特别有意思的地方，但是还有很多需要自己再继续学习下去，继续深入下去，路漫漫其修远兮，吾将上下而求索。一学期，也感谢董老师带来的激情与付出。