

# Java 说明文档

小组成员：

夏搏远 U201916320

张艺凡 U201916261

汪頔 U201916214

---

# 目录

<b>1 配置说明</b>	<b>1</b>
<b>2 主要功能介绍</b>	<b>2</b>
<b>3 数据库设计</b>	<b>2</b>
<b>4 文件结构说明</b>	<b>3</b>
4.1 QQ 端	3
4.2 QQ 空间端	4
<b>5 关键功能说明</b>	<b>4</b>
5.1 服务端说明	4
5.2 发送群通知	7
5.3 聊天说明	8
5.4 发送文件	16
5.5 接收文件	18
5.6 好友列表	21
<b>6 设计界面及最终界面</b>	<b>23</b>
6.1 注册界面	23
6.2 登录界面	23
6.3 主界面	24
6.4 服务器管理界面	24
6.5 服务端用户管理界面	25
6.6 服务端未读消息管理	25
<b>7 文件说明</b>	<b>25</b>
7.1 Src/server/entity/Users.java	25
7.2 Src/server/entity/Msg.java	26
7.3 Src/server/dao/IUser.java	26
7.4 Src/server/dao/IMsg.java	26
7.5 Src/server/dao/UserDao.java	26
7.6 Src/server/dao/MsgDao.java	26
7.7 Src/server/common/JDBC.java	27
7.8 Src/server/common/Page.java	27
7.9 Src/server/common/PageService.java	27
7.10 Src/server/frame/ServerMana.java	27
7.11 Src/server/frame/MainWindow.java	27
7.12 Src/server/frame/ServerActionListen.java	28
7.13 Src/server/frame/UserActionListen.java	28
7.14 Src/server/frame/QQServer.java	28
7.15 Src/com/socket/TCP.java	29
7.16 Src/com/socket/UDP.java	29

---

7.17 Src/com/socket/TCPServer.java.....	29
7.18 Src/client/socket/CS_TCP.java.....	30
7.19 Src/client/socket/CC_TCP.java.....	30
7.20 Src/client/socket/S_TCP.java.....	31
7.21 Src/client/socket/UDPChat.java.....	31
7.22 Src/client/common/MyLabel.java.....	31
7.23 Src/client/common/MyTreeIcon.java.....	31
7.24 Src/client/frame.....	32
7.25 Src/client/control/Login.java.....	33
7.26 Src/client/control/Register.java.....	33
7.27 Src/client/control/Chat.java.....	33
7.28 Src/client/control/RecieveThread.java.....	34
7.29 Src/client/control/Main.java.....	34
<b>8 QQ 空间文件说明.....</b>	<b>34</b>
8.1 数据库连接功能——DBUtil.java.....	34
8.2 对数据库的基本操作——BaseDao.java.....	35
8.3 用户对数据库的操作——UserDao.java.....	37
8.4 日志主页面——index.jsp.....	39
8.5 日志详情页——detail.jsp.....	40
8.6 发布日志——upload.jsp.....	41
8.7 处理相关的提交请求——DoUpload.jsp.....	42
8.8 提示成功——successs.jsp.....	43
<b>9 小组分工.....</b>	<b>43</b>
<b>10 后续改进点.....</b>	<b>45</b>

# 1 配置说明

1.该程序只用作小组童年怀旧，不用作任何商业用途，其中 oldqq 程序主要涉及 java 网络编程+JavaGUI，QQZone 则涉及 HTML 与 JAVA，以及少量的 Javascript、css 语言等。

2.由于 qq 空间以及 qq 的主体我们放在两个项目完成，通过将 qq 空间 tomcat 发布的方式，通过将 qq 聊天窗体的用户名传给页面，页面接收到相应的参数，使用该参数得到对应的用户对象，由此实现一个空间按钮到 qq 空间的跳转。

3.我们使用的是 mysql 数据库，下载我们的 sql 文件，首先修改数据库配置文件 oldqq.src.com.MyTools 中，以及修改 qqzone\src\main\java\util 中的 DBUtil 文件，将数据库数据表、root 以及密码改成自己的账户，便可实现与数据库的连接。
















```
1.      // 数据库地址和密码
2.      public static final String JDBC_URL = "jdbc:mysql://127.0.0.1:3306/oldqq?useUni
      code=true&characterEncoding=utf8
3.      &serverTimezone=GMT%2B8&useSSL=false&allowPublicKeyRetrieval=true";
4.      public static final String JDBC_USER = "root";
5.      public static final String JDBC_PWD = "123456";
```

4.oldqq 的分为服务端和客户端两个入口，服务端入口为 mainWindow.java，客户端入口为 login.java。

5.注：如果数据库连接不上，注意查看 mysql-connector-java-8.0.27.jar 包导入的正确性，另外该项目需要 maven 和 tomcat 的配置，相关配置过程不做详述。

6.QQ 空间构建的是 Maven 项目，主要配置说明如下：

## 相关 jar 包

	Tomcat 8.5.73	Provided ▼
	Maven: junit:junit:4.12	Test ▼
	Maven: org.hamcrest:hamcrest-core:1.3	Test ▼
	Maven: javax.servlet:javax.servlet-api:4.0.1	Provided ▼
	Maven: org.slf4j:slf4j-log4j12:1.7.2	Compile ▼
	Maven: log4j:log4j:1.2.17	Compile ▼
	Maven: org.slf4j:slf4j-api:1.7.2	Compile ▼
	Maven: mysql:mysql-connector-java:8.0.19	Compile ▼
	Maven: com.google.protobuf:protobuf-java:3.6.1	Compile ▼
	Maven: com.alibaba:fastjson:1.2.62	Compile ▼
	Maven: cn.hutool:hutool-all:5.4.7	Compile ▼
	Maven: commons-io:commons-io:2.4	Compile ▼
	Maven: commons-fileupload:commons-fileupload:1.3.1	Compile ▼
	Maven: javax.servlet:jstl:1.2	Compile ▼
	Maven: org.projectlombok:lombok:1.18.12	Provided ▼

- 
- ①mysql-connector-java.jar: 作为最重要的包之一, IDEA 在连接 mysql 数据库的时候要通过 mysql 驱动包进行连接。
  - ②Junit.jar:测试类所需包
  - ③Lombok.jar:为实体类提供所有属性的 get()和 set()方法
  - ④Tomcat: 发布 jsp 界面所需的服务器配置
  - ⑤Cn.hutool.jar: 糊涂工具包

## 2 主要功能介绍

本程序可以实现客户端可以实现用户注册、登录与用户之间互发消息, 其中用户可以在线或离线互发消息, 也可以发送表情和传输文件, 并且, 多个用户可以实现群聊的功能, 在聊天页面还可以跳转到本人的 qq 空间。

服务器端可以实现数据库的连接开启与关闭, 从而控制用户登录, 其次它可以他可以删查改用户信息, 并且对于未读消息也可以进行筛查。

qq 空间具体可以实现查看个人用户以及发布个人日志。

## 3 数据库设计

msg 表		
字段名	类型	描述
msg_id	int	未读消息的 ID
msg_content	varchar	未读消息内容
msg_sendto	int	未读消息接收者
msg_setfrom	int	未读消息发送者
msg_sendtime	datetime	消息发送时间
msg_remark	varchar	备用
msg_type	varchar	备用

tb_note		
字段名	类型	描述
noteID	int	日志的 ID
title	varchar	日志的标题
content	varchar	日志的内容
u_id	int	发送该文章的用户 ID

oldqq		
字段名	类型	描述
u_id	int	用户的 ID
u_name	varchar	用户的昵称
u_pwd	varchar	用户的密码
u_ip	varchar	用户的 IP 地址
u_state	varchar	用户的状态,
u_gender	varchar	用户的性别
u_email	varchar	用户的邮箱地址
u_last_login	datetime	用户最后一次登录时间
u_last_exit	datetime	用户最后一次退出时间
u_remarke	varchar	备用
u_signature	varchar	用户的个人签名
u_head_img	varchar	头像
u_type	varchar	用户的种类
u_birthday	datetime	用户的生日

## 4 文件结构说明

### 4.1 QQ 端

#### 4.1.1 Client 文件夹

Common 文件夹：存放常用的类

Control 文件夹：包含界面中对 JButton、JLabel 的监听等方法

Frame 文件夹：用户端的静态界面

Img 文件夹：包含文件中的所有图片

Socket 文件夹：用户端线程控制

#### 4.1.2 Com 文件夹

Socket 文件夹：包含最基本的服务端 Socket 套接字和客户端 Socket 套接字

MyTools 文件：一些经常调用的方法和枚举，都集中放在这里

#### 4.1.3 Server 文件夹

Common 文件夹：包含数据库连接文件和页数设置文件

Dao 文件夹：数据访问层，包含对数据库操作接口和 dao 文件

Entity 文件夹：对数据库的映射，实体类

Frame 文件夹：服务端页面及监听

QQServer 文件：服务端服务器

---

## 4.2 QQ 空间端

### 4.2.4 src 文件夹

main 文件夹：

- ①java 文件夹：包含程序需要的主要 java 代码
- ②resource 文件夹：包含程序需要的主要资源的代码
- ③webapp 文件夹：包含 web 开发所需的代码

test 文件夹：

包含所有测试所用到的代码

### 4.2.5 .idea 文件夹

IDEA 运行所需的相关配置文件

### 4.2.6 target 文件夹

项目运行配置的文件夹

### 4.2.7 pom.xml

IDEA 项目所需的配置，包括依赖的包，相关插件

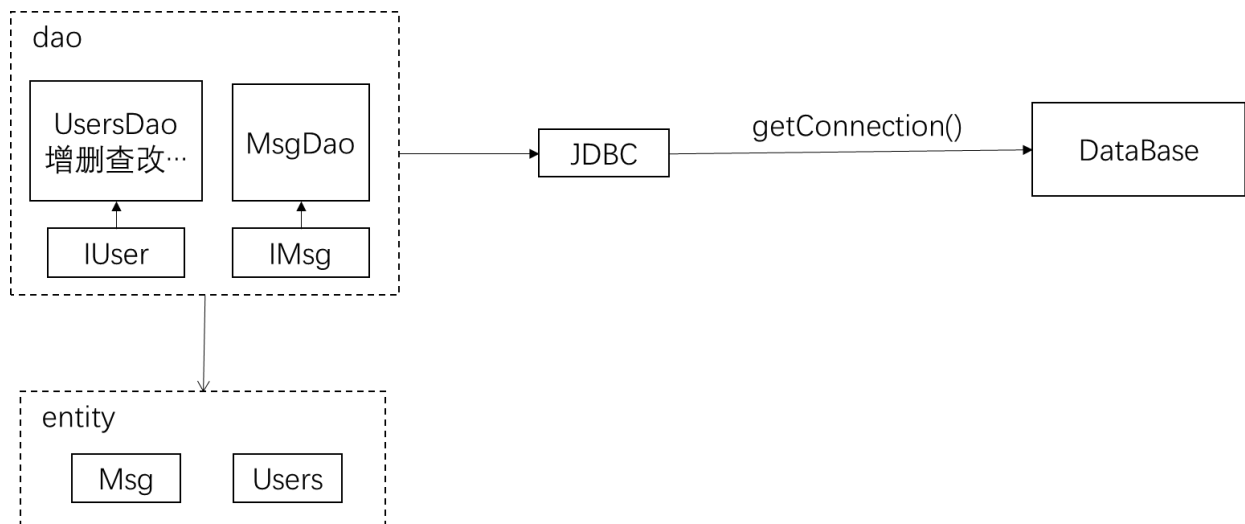
## 5 关键功能说明

### 5.1 服务端说明

#### 5.1.1 开启/关闭服务端服务器

- ①按钮添加监听，执行 startOrCloseServer()方法
- ②在该方法中，根据设置的默认的端口实例化 QQServer，QQServer 继承自 TCPServer。
- ③在 TCPServer 中实例化 ServerSocket，并开启一个线程不断地接收连接。
- ④接收到消息 dealWithMessage () 方法，根据不同的 flag 调用不同的方法处理信息。

这里与数据库连接的方法是：



## ServerMaNa.java

```

1. public void startOrCloseServer()
2. {
3.     if(btnStart.getText().equals("启动服务器"))
4.     {
5.         qqServer=new QQServer(MyTools.QQServerPort);
6.         allTime=0;
7.         startTime=System.currentTimeMillis();
8.         isRun=true;
9.         lblState.setText("服务器正在运行中...");
10.        btnStart.setText("停止服务器");
11.    }
12.    else if(btnStart.getText().equals("停止服务器"))
13.    {
14.        if(qqServer!=null)
15.        {
16.            qqServer.closeServer();
17.            isRun=false;
18.            lblState.setText("服务器已关闭。");
19.            JOptionPane.showMessageDialog(null,"本次服务器总共运行"+allTime+"秒");
20.            btnStart.setText("启动服务器");
21.        }
22.    }
23. }
  
```

QQServer 继承自 TCPServer.java

## TCPServer.java

开辟新线程后台接收消息并处理。



```

1. public void getConnectionNewThread()
2. {
3.     Runnable runnable=new Runnable()
4.     {
5.         @Override
6.         public void run()
7.         {
8.             while (isStart)
9.             {
10.                TCP tcp = new TCP(server);
11.                tcpSockets.add(tcp);
12.                getMessageNewThread(tcp);
13.            }
14.        }
15.    };
16.    new Thread(runnable).start();
17. }
18. /**
19.  * 开辟新线程后台接收消息并处理
20.  * @param tcp 传过来的 TCP 连接
21.  */
22. public void getMessageNewThread(final TCP tcp)
23. {
24.     Runnable runnable = new Runnable()
25.     {
26.         @Override
27.         public void run()
28.         {
29.             boolean exit=false;
30.             while(!exit)
31.             {
32.                 try
33.                 {
34.                     //将获取的消息按#分割，msg[0]的表示标志，msg[1]表示内容
35.                     String[] temp=tcp.getMessage().split(MyTools.FLAGEND);
36.                     String flagHead=temp[0];//获取消息的标志
37.                     String message=temp[1];//获取消息的真实内容
38.                     Flag flag=MyTools.stringToFlagEnum(flagHead);//获取标志
39.                     dealWithMessage(flag,message,tcp);
40.                 }
41.                 catch (Exception e)
42.                 {

```

```
43.             //e.printStackTrace();
44.             if(tcpSockets.remove(tcp))//从 ArrayList 中移除已退出的用户
45.             {
46.                 dealWithExit(tcp);
47.             }
48.             exit=true;
49.         }
50.     }
51. }
52. };
53.     new Thread(runnable).start();
54. }
```

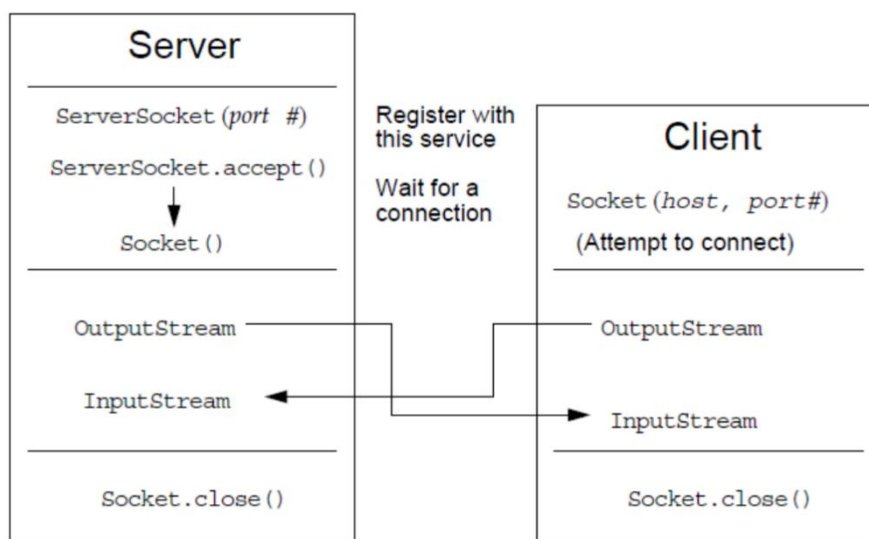
## 5.2 发送群通知

循环存放与客户端连接的 tcpServer，给每个客户端 Socket 都发消息。

```
1. public void sendPublicMessage(String message)
2. {
3.     for(TCP t:tcpSockets)
4.         t.sendMessage(Flag.PUBLIC_MESSAGE+MyTools.FLAGEND+message);
5.     JOptionPane.showMessageDialog(null, "公告发布成功!", "恭喜",
        JOptionPane.INFORMATION_MESSAGE);
6. }
```

## 5.3 聊天说明

### 5.3.2 常见的网络编程



(1) 服务器首先启动监听程序，对指定的端口进行监听，等待接收客户端的连接请求。

(2) 客户端程序启动，请求连接服务器的指定端口。

(3) 服务器收到客户端的连接请求后与客户端建立套接字连接。

(4) 连接成功后，客户端与服务器分别打开两个流，其中客户端的输入流连接到服务器的输出流，服务器的输入流

连接到客户端的输出流，两边的流建立连接后就可以双向的通信了。

(5) 当通信完毕后客户端与服务器端两边各自断开连接。

### 5.3.3 聊天机制

#### 1.1.1.1 在线聊天

每个客户端（QQ 用户）既是客户端又是服务端。

①每个用户在登录进入主界面初始化时，会实例化 `S_TCP`，然后在其中调用 `TCPServer` 的构造方法，启动用户的服务端，也就是构建 `ServerSocket`。

②与其他用户进行聊天（前提为其他用户在线）时，根据好友的 IP 和端口实例化 `CC_TCP`，也就是构建与好友服务端进行连接的客户端 `Socket`。并且会执行 `getMessageNewThread()` 方法，创建不断接收信息的线程。

③发送消息时，直接调用 `sendMessage()` 方法，注意在不同的功能下调用该方法传递的参数不同，这里的参数为 **START\_CHAT#Name**，然后将消息传递给了对应好友的服务器。

④初始化界面->发送框消息传递至接收框中。

⑤如果接收到用户消息，直接插入在接收框中。

#### 实现代码

`Main.java` 的 `init()` 方法内实例化 `S_TCP(serverPort,main)`，`S_TCP` 继承自 `TCPServer`，其构造方法中开启服务端服务器。

---

## TCPServer.java

构建 ServerSocket

```
1. public TCPServer(int serverPort)
2. {
3.     this.serverPort=serverPort;
4.     isStart = true;
5.     if(startServerTCP())//启动 TCP 服务端
6.     {
7.         afterServerStart(); //服务端启动后要做的的事情
8.         getConnectionNewThread();//开辟新线程不断的接受连接
9.     }
10. }

1. public boolean startServerTCP()
2. {
3.     if(serverPort==MyTools.QQServerPort)//如果启动的是 QQ 服务器
4.     {
5.         try
6.         {
7.             server=new ServerSocket(serverPort);
8.             return true;
9.         }
10.        catch (IOException e)
11.        {
12.            System.out.println("启动 QQ 服务器失败！端口"+serverPort+"已被占用，请检查是否已开启了一个 QQ 服务器！");
13.            return false;
14.        }
15.
16.    }
17.    else//如果启动的是普通 TCP 服务端
18.    {
19.        while(true)
20.        {
21.            try
22.            {
23.                server = new ServerSocket(serverPort);
24.                break;
25.            }
26.            catch (IOException e)
```

```

27.         {
28.             serverPort++;
29.         }
30.     }
31.     return true;
32. }
33. }

```

## Main.java

选择在线或离线聊天

```

1. public void startChat(ActionEvent e)
2. {
3.     if(tree.getSelectionPath().getPathCount()==3)
4.     {
5.         String str=tree.getSelectionPath().getLastPathComponent().toString();
6.         String friendName=str.substring(0,str.indexOf("("));
7.         String friendIP=str.substring(str.indexOf("(")+1,str.indexOf(":"));
8.         int friendPort=Integer.parseInt(str.substring(str.indexOf(":")+1,str.indexO
f(")")));
9.         if(!friendIP.equals("下线或隐身"))
10.        {
11.            Chat chat=new Chat(friendIP,friendPort,friendName,this.lbl1 用户
名.getText());
12.            chat.setVisible(true);
13.        }
14.        else
15.        {
16.            Chat chat=new Chat(cs_TCP, friendName);
17.            chat.setVisible(true);
18.        }
19.    }
20.    else
21.    {
22.        JOptionPane.showMessageDialog(null, "对不起，您未选中任何好友！");
23.    }
24. }

```

## Chat.java

实例化 CC\_TCP，根据朋友 IP 和端口构建客户端 Socket，发送消息，初始化界面，初始化接收文件。

```

1. public Chat(String friendIP,int friendTCPPort,String friendName,String myName)

```

```

2. {
3.     this.friendIP=friendIP;
4.     this.friendTCPPort=friendTCPPort;
5.     this.friendName=friendName;
6.     this.myName=myName;
7.     try
8.     {
9.         cc_TCP=new CC_TCP(this.friendIP, this.friendTCPPort,this);
10.        cc_TCP.sendMessage(Flag.START_CHAT+MyTools.FLAGEND+myName);//发起聊天请求
11.    }
12.    catch (Exception e)
13.    {
14.        e.printStackTrace();
15.    }
16.    init();
17.    initGetFileServer();//初始化接收文件的服务
18. }

```

## 发送消息

```

1. public void sendMessage()
2.     {
3.         //发送的是文本信息
4.         if ("".equals(ImgPath) && "".equals(screenCutImgName))
5.         {
6.             if(faceIdx<0)
7.             {
8.                 if(cc_TCP!=null)//如果是发起聊天的一方
9.                     cc_TCP.sendMessage(Flag.MESSAGE+MyTools.FLAGEND+jTextPane 发送
框.getText());
10.                else if(tcp!=null)//如果是接收聊天的一方
11.                    tcp.sendMessage(Flag.MESSAGE+MyTools.FLAGEND+jTextPane 发送
框.getText());
12.                else if(cs_TCP!=null)//发送未读消息
13.                    cs_TCP.sendMessage(Flag.UNDERLINE_MESSAGE+MyTools.FLAGEND+frien
dName+MyTools.SPLIT1+jTextPane 发送框.getText());
14.                setReceivePaneText(true,jTextPane 发送框.getText());//将用户发送的消息
显示在聊天框中
15.                jTextPane 发送框.setText("");// 清空发送框
16.            }
17.            else
18.            {

```

```

19.         if(cc_TCP!=null)//如果是发起聊天的一方
20.             cc_TCP.sendMessage(Flag.FACE+MyTools.FLAGEND+faceIdx);
21.         else if(tcp!=null)//如果是接收聊天的一方
22.             tcp.sendMessage(Flag.FACE+MyTools.FLAGEND+faceIdx);
23.         new MyTextPane(jTextPane 接收
框).addIcon(MyTools.getFaceByIdx(faceIdx), myName);
24.         jTextPane 发送框.setText("");// 清空发送框
25.         faceIdx=-1;//图片表情发完后置为默认
26.     }
27.
28. }
29. //发送的是图片
30. else if(!"".equals(ImgPath))
31. {
32.     try
33.     {
34.         //置空发送框
35.         jTextPane 发送框.setText("");
36.         //接收框显示图片
37.         jTextPane 接收框.insertIcon(new ImageIcon(ImageIO
38.             .read(new FileInputStream(ImgPath))));
39.         //发送图片给对方
40.         cc_TCP.sendImg(ImgPath);
41.
42.
43.         //发送完图片后置空图片路径
44.         ImgPath = "";
45.     }
46.     catch (FileNotFoundException e)
47.     {
48.         // TODO Auto-generated catch block
49.         e.printStackTrace();
50.     }
51.     catch (IOException e)
52.     {
53.         // TODO Auto-generated catch block
54.         e.printStackTrace();
55.     }
56. }
57. else if(!"".equals(screenCutImgName))
58. {
59.     try

```

```

60.         {    //置空发送框
61.             jTextPane 发送框.setText("");
62.             //接收框显示图片
63.             jTextPane 接收框.insertIcon(new ImageIcon(ImageIO
64.                 .read(new FileInputStream("./screenCut/snap.jpg"))));
65.             //发送图片给对方
66.             cc_TCP.sendImg("./screenCut/snap.jpg");
67.
68.
69.             //发送完图片后置空图片路径
70.             screenCutImgName = "";
71.         }
72.         catch (FileNotFoundException e)
73.         {
74.             // TODO Auto-generated catch block
75.             e.printStackTrace();
76.         }
77.         catch (IOException e)
78.         {
79.             // TODO Auto-generated catch block
80.             e.printStackTrace();
81.         }
82.     }
83.
84. }

```

发送按钮绑定了 sendMessage（）方法监听。

## TCP.java

### sendMessage 方法（输出流）

```

1. public void sendMessage(String text)
2. {
3.     try
4.     {
5.         socket.getOutputStream().write(text.getBytes());
6.         System.out.println("TCP 成功发送: " + text);
7.         Thread.sleep(100); //每发送完一个消息线程休息一下，防止连续发送消息重叠在一起
8.     }
9.     catch (Exception e)

```



```

10.    {
11.        e.printStackTrace();
12.    }
13. }

```

### 1.1.1.2 离线聊天

- ①初始化界面，将发送框消息发送至接收框，并清空发送框。
- ②从登录界面获取 CS\_TCP，这个是与根据总服务器端口和服务器 IP 构建的客户端套接字，发送消息至服务端。
- ③服务端在初始化时便开辟新线程不断的接受连接，并在接受连接线程中构建接收消息线程，因此在服务器接收消息之后便处理接收过来的消息。
- ④服务器接收消息的线程中调用 `dealWithMessage(flag,message,tcp)` 方法。
- ⑤在 `dealWithMessage(flag,message,tcp)` 方法中根据不同的 flag 调用不同的处理消息方法，并在好友上线时将消息显示在右下角提示框中。

#### 实现代码

#### Chat.java

```

1. public Chat(CS_TCP cs_TCP,String friendName)
2. {
3.     this.cs_TCP=cs_TCP;
4.     this.friendName=friendName;
5.     init();
6.     setTitle("给"+friendName+"发送离线消息中.....");
7. }

```

#### TCPServer.java

##### 初始化 TCPServer

```

1. public TCPServer(int serverPort)
2. {
3.     this.serverPort=serverPort;
4.     isStart = true;
5.     if(startServerTCP())//启动 TCP 服务端
6.     {
7.         afterServerStart(); //服务端启动后要做的的事情
8.         getConnectionNewThread();//开辟新线程不断的接受连接
9.     }
10. }

```

##### getConnectionNewThread()连接线程

```

1. public void getConnectionNewThread()

```

```

2. {
3.     Runnable runnable=new Runnable()
4.     {
5.         @Override
6.         public void run()
7.         {
8.             while (isStart)
9.             {
10.                TCP tcp = new TCP(server);
11.                tcpSockets.add(tcp);
12.                getMessageNewThread(tcp);
13.            }
14.        }
15.    };
16.    new Thread(runnable).start();
17. }

```

getMessageNewThread(tcp)开启获取信息线程

```

1. public void getMessageNewThread(final TCP tcp)
2.     {
3.         Runnable runnable = new Runnable()
4.         {
5.             @Override
6.             public void run()
7.             {
8.                 boolean exit=false;
9.                 while(!exit)
10.                {
11.                    try
12.                    {
13.                        //将获取的消息按#分割，msg[0]的表示标志，msg[1]表示内容
14.                        String[] temp=tcp.getMessage().split(MyTools.FLAGEND);
15.                        String flagHead=temp[0];//获取消息的标志
16.                        String message=temp[1];//获取消息的真实内容
17.                        Flag flag=MyTools.stringToFlagEnum(flagHead);//获取标志
18.                        dealWithMessage(flag,message,tcp);
19.                    }
20.                    catch (Exception e)
21.                    {
22.                        //e.printStackTrace();
23.                        if(tcpSockets.remove(tcp))//从 ArrayList 中移除已退出的用户
24.                        {

```

```

25.                dealWithExit(tcp);
26.            }
27.                exit=true;
28.            }
29.        }
30.    }
31.    };

```

dealWithMessage(flag,message,tcp)中，根据不同的 flag 执行不同的方法。针对发送的信息，调用 void showMessage(String message,TCP tcp)显示信息方法。

### S\_TCP.java

将消息展示在好友的接受框中。

```

1. public void showMessage(String message,TCP tcp)
2. {
3.     for(Chat chat:chats)
4.     {
5.         if(chat.friendName.equals(tcp.getClientName()))
6.         {
7.             if(!chat.isVisible())
8.             {
9.                 new PublicMessageFrame(tcp.getClientName()+"给您发来消息",message,chat);
10.                //对方未打开聊天窗体时就播放声音
11.                MyTools.playMsgSound();
12.            }
13.            chat.setReceivePaneText(false, message);
14.        }
15.    }
16. }

```

## 5.4 发送文件

- ①界面点击发送文件 label，调用 sendFile()方法；
- ②实例化 sendThread 线程，在该线程中，调用 startSend（）
- ③根据好友 IP 和好友文件接收端口建立 Socket 套接字，根据文件路径实例化文件对象，字节输入流读文件，更新文件传输进度条并且写文件。

### Chat.java

```

1. public void sendFile()
2. {
3.     System.out.println("对方已开启文件接收监听！");
4.     SendFileFrame sendFileFrame=new SendFileFrame();

```

```

5.     sendFileFrame.lblProgress.setText("正在等待文件被接收...");
6.     String filePath = sendFileFrame.showDialog(JFileChooser.FILES_ONLY);
7.     sendFileFrame.setVisible(true);
8.     SendTread sendThread;
9.     //sendThread = new SendTread(sendFileFrame, filePath,message.split(MyTools.SPLI
    T1)[0],Integer.parseInt(message.split(MyTools.SPLIT1)[1]));
10.    sendThread = new SendTread(sendFileFrame, filePath,friendIP,friendGetFilePort);
11.    sendThread.start();
12. }

```

## SendTread.java

发送文件的线程

```

1. public void startSend(String filepath)
2. {
3.     try
4.     {
5. //         ss = new ServerSocket(ConnInfor.CONN_PORT);
6. //         socket = ss.accept();
7.         socket = new Socket(connIP, clientPort);
8. //         JOptionPane.showMessageDialog(null, "发送端已成功建立连接!");
9.         File file = new File(filepath);
10.         out = new DataOutputStream(socket.getOutputStream());
11.         String fileName = file.getName();
12.         long FileLen = file.length();
13.         out.writeUTF(fileName);
14.         out.flush();
15.         out.writeLong(FileLen);
16.         out.flush();
17.
18.         fileIn = new DataInputStream(new FileInputStream(file));
19.         byte[] buff = new byte[1024];
20.         int len = 0;
21.         long passLen = 0;
22.         long startTime = System.currentTimeMillis();
23.         long endTime = 0;
24.         double passTime = 0;
25.         while (true)
26.         {
27.             if (fileIn != null)
28.             {
29.                 len = fileIn.read(buff);

```

```

30.             passLen += len;
31.         }
32.         endTime = System.currentTimeMillis();
33.         passTime = endTime - startTime;
34.         if (len == -1)
35.         {
36.             fram.updateProgressBar(fram.progressBar, FileLen,
37.                                     FileLen, fram.lblProgress, passTime);
38.             //fram.afterSendFile() ;
39.             break;
40.         }
41.
42.         fram.updateProgressBar(fram.progressBar, passLen, FileLen,
43.                                 fram.lblProgress, passTime);
44.         out.write(buff, 0, len);
45.     }
46. }
47. catch (IOException e)
48. {
49.     e.printStackTrace();
50. }
51. finally
52. {
53.     closeSend();
54. }
55. }

```

## 5.5 接收文件

- ①新建聊天窗口即会建立接收文件的 ServerSocket，发送传输文件的信息。
- ②实例化 RecieveThread 线程。
- ③RecieveThread 的 run 方法中执行了 recieveFile() 方法。
- ④字节输入流读取，然后根据路径用字节输出流写入文件，不断更新文件传输进度条，完成文件传输。

### ReceiveThread.java

```

1. public void recieveFile()
2. {
3.     try
4.     {
5.         in = new DataInputStream(socket.getInputStream());
6.         String filePath = null;
7.         // 读取发过来的文件名和文件长度

```

```

8.         String fileName = in.readUTF();
9.         long fileLen = in.readLong();
10.        // 弹出接收对话
11.        int result = JOptionPane.showConfirmDialog(null, "是否接收文件?
    " + fileName,
12.            "提示", JOptionPane.YES_NO_CANCEL_OPTION);
13.        // 选择对话框中的取消按钮的相应操作
14.        if (result == JOptionPane.CANCEL_OPTION)
15.        {
16.
17.        }
18.        // 选择对话框中的否按钮的相应操
19.        else if (result == JOptionPane.NO_OPTION)
20.        {
21.            closeRecieve();
22.        }
23.        // 选择对话框中的是按钮的相应操
24.        else if (result == JOptionPane.OK_OPTION)
25.        {
26.
27.            filePath = fram.showDialog(JFileChooser.DIRECTORIES_ONLY);
28.            fram.setVisible(true);
29.            fileOut = new DataOutputStream(new FileOutputStream(filePath
30.                + "/" + fileName));
31.
32.            byte[] buff = new byte[1024];
33.
34.            int len = 0;
35.
36.            long passLen = 0;
37.
38.            long startTime = System.currentTimeMillis();
39.            long endTime = 0;
40.            long passTime = 0;
41.
42.            while (true)
43.            {
44.                if (in != null)
45.                {
46.                    len = in.read(buff);
47.                    passLen += len;
48.                }

```

```

49.         endTime = System.currentTimeMillis();
50.         passTime = endTime - startTime;
51.         if (len == -1)
52.         {
53.             fram.updateProgressBar(fram.progressBar, fileLen,
54.                                     fileLen, fram.lblProgress, passTime);
55.             break;
56.         }
57.         fram.updateProgressBar(fram.progressBar, passLen,
58.                                 fileLen, fram.lblProgress, passTime);
59.         fileOut.write(buff, 0, len);
60.     }
61.
62.     }
63.
64.     }
65.     catch (UnknownHostException e)
66.     {
67.         // TODO Auto-generated catch block
68.         e.printStackTrace();
69.     }
70.     catch (IOException e)
71.     {
72.         // TODO Auto-generated catch block
73.         e.printStackTrace();
74.     }
75.     finally
76.     {
77.         closeRecieve();
78.     }
79. }

```

## Chat.java

```

1. public void initGetFileServer()
2.     {
3.         while (true)
4.         {
5.             try
6.             {
7.                 getFileServer = new ServerSocket(myGetFilePort);
8.                 System.out.println("已开启文件接收监听!");
9.                 if(cc_TCP!=null)

```

```

10.         cc_TCP.sendMessage(Flag.SENDFILE+MyTools.FLAGEND+myGetFilePort);
11.     else
12.         tcp.sendMessage(Flag.SENDFILE+MyTools.FLAGEND+myGetFilePort);
13.     break;
14. }
15. catch (Exception e)
16. {
17.     myGetFilePort++;
18. }
19. }
20. Runnable runnable=new Runnable()
21. {
22.
23.     @Override
24.     public void run()
25.     {
26.         //tcp.sendMessage(Flag.GETFILE_OK+MyTools.FLAGEND+TCP.getLocalHostI
P()+MyTools.SPLIT1+myPort);
27.         RecieveThread recieveThread;
28.         while (true)
29.         {
30.             try
31.             {
32.                 Socket socket = getFileServer.accept();
33.                 System.out.println("与文件发送方连接成功! ");
34.                 recieveThread = new RecieveThread(new SendFileFrame(), sock
et);
35.                 recieveThread.start();
36.             }
37.             catch (IOException e)
38.             {
39.                 e.printStackTrace();
40.             }
41.         }
42.     }
43. };
44. new Thread(runnable).start();
45. }

```

## 5.6 好友列表

①设置在线用户、不在线用户和我的好友的图标:



未点击前某个图标->点击之后分组图标发生变化。并添加至 nodeImages 中。

② treeModel=new DefaultTreeModel(root); //新建一个用于存放好友树的 Model

③ 根据 groupNames 循环，并为每个分组添加分组节点 DefaultMutableTreeNode  
node=new DefaultMutableTreeNode(groupNames[i]);

④ String[] temp=friendInfos.get(i)[j].split(MyTools.SPLIT3); 循环每个分组的好友信息，获取每个好友的用户名、IP、端口号、好友头像、状态，并且将“用户名+头像”添加至 nodeImages。

⑤ 将每个分组内的好友添加至每个分组中。node.add(new DefaultMutableTreeNode(friendName+"("+ip+": "+port+")"))

⑥ 将分组结点添加至根节点上 root.add(node);

⑦ tree.setCellRenderer(new MyTreeIcon(nodeImages)); tree.setModel(treeModel); 完成好友树的建立。

## Main.java

```
1.  public void init()
2.      {
3.          tree.setRootVisible(false); //设置根节点不可见
4.          tree.setAutoscrolls(true); //设置自动滚动
5.          tree.getSelectionModel().setSelectionMode(TreeSelectionMode.SINGLE_TREE_SELECTION); //设置单选模式
6.
7.          nodeImages=new ArrayList<String>();
8.          nodeImages.add("所有在线用户"+MyTools.SPLIT1+"../img/face/f107.png"+MyTools.SPLIT1+"../img/face/f108.png");
9.          nodeImages.add("所有不在线用户"+MyTools.SPLIT1+"../img/face/f109.png"+MyTools.SPLIT1+"../img/face/f110.png");
10.         nodeImages.add("我的好友"+MyTools.SPLIT1+"../img/face/f111.png"+MyTools.SPLIT1+"../img/face/f111.png");
11.         DefaultMutableTreeNode root=new DefaultMutableTreeNode("root");
12.         treeModel=new DefaultTreeModel(root); //新建一个用于存放好友树的 Model
13.         for(int i=0;i<groupNames.length;i++)
14.             {
15.                 DefaultMutableTreeNode node=new DefaultMutableTreeNode(groupNames[i]);
16.
17.                 for(int j=0;j<friendInfos.get(i).length;j++)
18.                     {
19.                         String[] temp=friendInfos.get(i)[j].split(MyTools.SPLIT3);
20.
21.                         //System.out.println(friendInfos.get(i)[j]);
22.                         String friendName=temp[0]; //好友用户名
23.                         String ip=temp[1]; //好友 IP 地址
```

```

23.         String port=temp[2];//好友的端口号
24.         String headImage=temp[3];//好友头像
25.         String state=temp[4];//状态
26.         nodeImages.add(friendName+MyTools.SPLIT1+"../img/headImage
/small/"+headImage+"_32.jpg");
27.         node.add(new DefaultMutableTreeNode(friendName+"("+ip+": "+
port+"")));
28.     }
29.     root.add(node);
30. }
31. tree.setCellRenderer(new MyTreeIcon(nodeImages));
32. tree.setModel(treeModel);
33. }

```

## 6 设计界面及最终界面

### 6.1 注册界面

注册	
昵称 (label+text)	头像 (选择)
密码	
.....	
注册按钮	

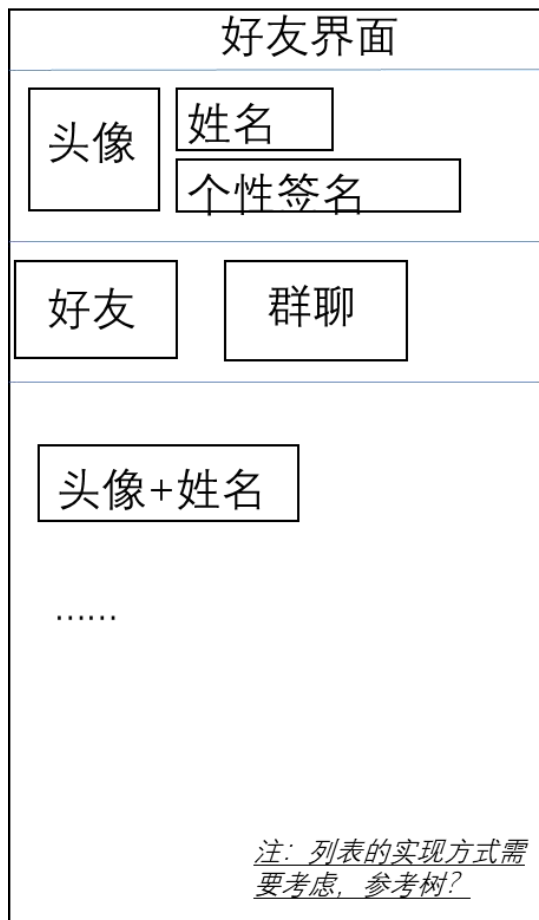


### 6.2 登录界面

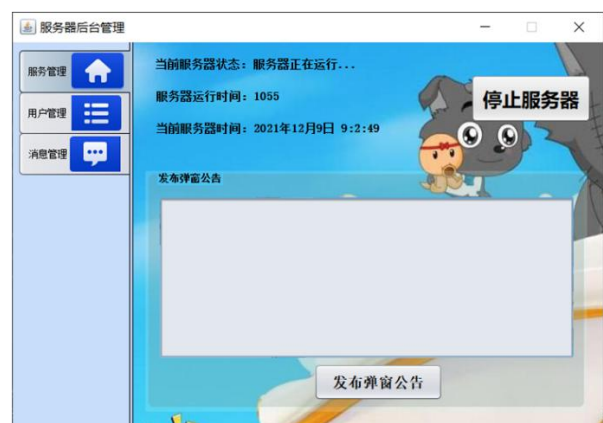
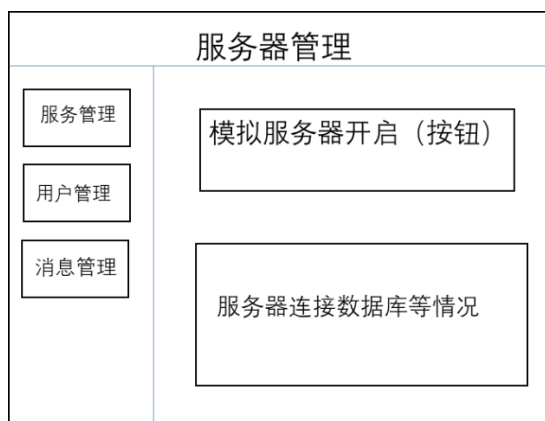
登录	
头像	账号
	密码
注册 (按钮)	登录 (按钮)



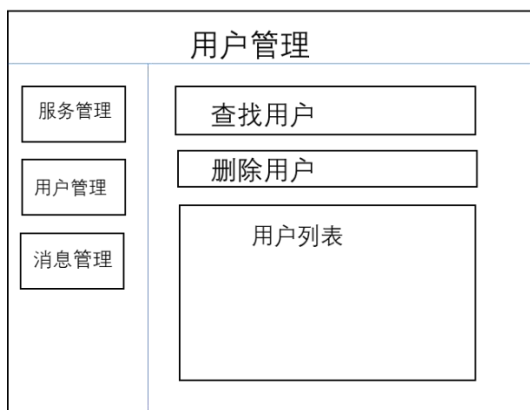
## 6.3 主界面



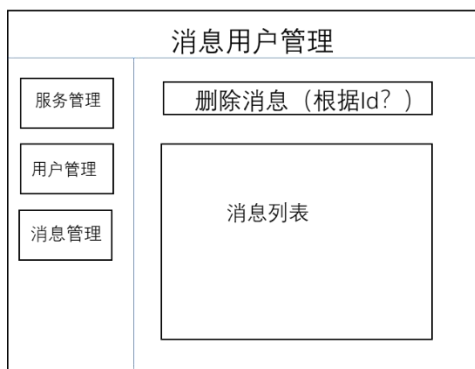
## 6.4 服务器管理界面



## 6.5 服务端用户管理界面



## 6.6 服务端未读消息管理



## 7 文件说明

### 7.1 Src/server/entity/Users.java

用户表的实体类

* @param id	用户 ID
* @param name	用户名
* @param pwd	用户密码
* @param gender	用户性别
* @param email	用户 e-mail
* @param lastLogin	用户最后一次登录时间
* @param lastExit	用户最后一次下线时间
* @param remarke	备注
* @param signature	用户签名
* @param headImg	用户头像
* @param birthday	用户生日
* @param type	用户类型 (管理员/普通用户)

---

## 7.2 Src/server/entity/Msg.java

```
int msgId;
String msgContent;
int sendFrom;//信息发送者
int sendTo;//信息接收者
String sendTime;//信息发送时间
String remark;
String msgTye;//信息类型： 离线消息和群公告
```

## 7.3 Src/server/dao/IUser.java

用户类数据库操作接口

## 7.4 Src/server/dao/IMsg.java

信息类数据库操作接口

## 7.5 Src/server/dao/UserDao.java

连接数据库并对数据库的一些操作

- (1) 新增 JDBC 连接
- (2) 运用 con.update(sql,params)更新，类似的增删查改（sql 是需要执行的预编译语句、params 是预编译语句的参数列表）
- (3) delete(int usersId)中 userid 改为 userId
- (4) ArrayList<Users> queryAll()、Users queryById(int usersid)、Users queryByName(String userName)三种查询方法。
- (5) boolean checkUserIsExit(String name)检查用户是否存在
- (6) boolean checkNameAndPwd(String name, String password)检查用户名密码是否正确
- (7) boolean updateUserState(String name, String state)更新用户状态
- (8) void setLastLogin(String name)、setLastExit(String name)记录用户最后一次登录和退出时间
- (9) setIP(String IP,String name)设置用户 ip

## 7.6 Src/server/dao/MsgDao.java

- (1) new JDBC()连接数据库，例 jdbc.update(sql, strArr)可直接对数据库进行更新
- (2) boolean insertMsg(Msg msg)插入信息
- (3) boolean deleteMsg(int msgId)删除消息
- (4) int updateMsg(Msg msg)更新消息，返回更新操作数
- (5) Msg selectAMsg(int msgId)查看一条信息
- (6) ArrayList<Msg> selectMsgs(String sql)查看多条信息
- (7) boolean deleteMsgBySendTO(int sendto)根据接收人筛选并删除发送至该接收人的所有信息

---

(8) `List<Msg> selectMsgBySendTo(int sendto)`根据接收人筛选查看所有信息

## 7.7 Src/server/common/JDBC.java

- (1) `getConnection()`默认为 MyTools 里面的数据库连接参数->直接连接数据库
- (2) `update(String sql,String[] params)`执行数据库更新操作 (返回操作数)
- (3) `query(String sql,String[] params)` 执行数据库的查询操作

## 7.8 Src/server/common/Page.java

**Page 类，有以下属性：**

```
int totalRecords;    //总记录数
int totalPage;       //总页数
int pageSize;        //每页显示几条记录
static int currentPage; //当前页
以及对应有一些方法。
```

## 7.9 Src/server/common/PageService.java

- (1) `void initPage()`初始化页面
- (2) `List gotoPage(int target)` 跳转到某一页
- (3) `List gotoFirst()`跳转到首页
- (4) `List gotoNext()`跳转到下一页

## 7.10 Src/server/frame/ServerMana.java

- (1) 添加按钮 `btnStart`
- (2) `btnStart` 添加监听 `void startOrCloseServer()`根据按钮文本判断打开/关闭服务器
- (3) 页面主要通过 `groupLayout.setHorizontalGroup()`和 `groupLayout.setVerticalGroup()`来改变整体布局

## 7.11 Src/server/frame/MainWindow.java

### 7.11.1 main 方法

- (1) 主题换色
- (2) 新建窗体

### 7.11.2 MainWindow()构造方法

- (1) 设置大小
- (2) `serverMana=new ServerMana();`服务管理页面->ServerMana 里设置样式
- (3) `tabbedPane.addTab("服务管理", new ImageIcon(MainWindow.class.getResource("/client/img/manager_server.png")), serverMana, null);`

- (4) `UserMana ()`、`MsgMana ()` 同理

- 
- (5) **GroupLayout** 支持两种组:
- 串行组 (sequential group): 按顺序沿指定方向 (水平/垂直) 逐个放置元素。
- 并行组 (parallel group): 沿指定方向 (水平/垂直) 并行排列元素, 能够以四种不同方式对齐其子元素。
- >使其摆放更整齐, 实现边框
- (6) `setDefaultCloseOperation(int operation)`; 用户单击关闭窗口方法中的**参数解释**如下:
- ①为“0”或 **DO\_NOTHING\_ON\_CLOSE**:  
(在 `WindowConstants` 中定义): 不执行任何操作; 要求程序在已注册的 `WindowListener` 对象的 `windowClosing` 方法中处理该操作。
- ②为“1”或 **HIDE\_ON\_CLOSE**  
调用任意已注册的 `WindowListener` 对象后自动隐藏该窗体。此时没有关闭程序, 只是将程序界面隐藏了。
- ③为“2”或 **DISPOSE\_ON\_CLOSE**  
调用任意已注册 `WindowListener` 的对象后自动隐藏并释放该窗体。但继续运行应用程序, 释放了窗体中占用的资源。
- ④为“3”**EXIT\_ON\_CLOSE** (在 `JFrame` 中定义):  
使用 `System exit` 方法退出应用程序。仅在应用程序中使用。结束了应用程序。

## 7.12 Src/server/frame/ServerActionListen.java

- (1) `void actionPerformed(ActionEvent e)` 启动服务器按钮的监听事件

## 7.13 Src/server/frame/UserActionListen.java

- (1) `UserActionListen ()` 初始化, 展示所有用户
- (2) `actionPerformed(ActionEvent e)` 功能按钮监听
- ①如果按钮为“删除”, 根据选择用户的 `userId` 删除用户
- ②如果按钮为“刷新”, 刷新整个页面
- ③如果按钮为“查找”, 分为按用户 ID 查找和按姓名查找
- ④如果按钮为“其他”, 可以跳转至首页、上一页、下一页、尾页, 以及修改页码

## 7.14 Src/server/frame/QQServer.java

- (1) `QQServer extends TCPServer`, 开启服务器
- (2) `dealWithMessage(Flag flag, String message, TCP tcp)` 处理客户端发来的各种信息、flag 信息标志、message 消息内容、tcp TCP 连接
- (3) `void dealWithExit(TCP tcp)` 退出登录、更新用户状态
- (4) `void afterServerStart()` 服务端启动后执行方法, 显示在线人数
- (5) `void doLogin(String message, TCP tcp)` 处理客户端登录
- (6) `void doGetFriendInfo(String message, TCP tcp)` 处理用户发来的“请求好友资料”申请
- (7) `void doRegister(String message, TCP tcp)` 用户注册

---

## 7.15 Src/com/socket/TCP.java

- (1) TCP(String serverIP, int serverPort)构造方法一，创建客户端 Socket  
getMessageNewThread();对于客户端的 TCP，给它启动一个新线程不停的接收消息
- (2) TCP(ServerSocket server)构造方法二，创建服务端 Socket
- (3) void closeSocket()关闭 socket 连接
- (4) void getMessageNewThread()开辟新线程后台接收消息并处理
- (5) void sendMessage(String text) 发送消息
- (6) String getMessage()接收消息
- (7) void sendImg(String imgPath) 根据图片路径发送图片
  - ①先创建字节输入流对象
  - ②利用 sendMessage () 发送文件名
  - ③图片大小大于 0 时，输出
  - ④线程 sleep
- (8) void getImg(Chat chat ,String imgName)获取图片  
参数为聊天框和文件名
  - ①调用系统功能创建文件（文件名拼接得到文件路径）
  - ②字节输入流 read 读取图片，再用文件输出流展示
  - ③new MyTextPane()在接收框显示出别人传递过来的图片
- (9) String getClientIP()获取客户端 IP 地址

## 7.16 Src/com/socket/UDP.java

- (1) void getMyUsefulPort()获取可用的端口号
  - ①实例化一个 DatagramSocket
  - ②或者端口号++
- (2) void sendMessage(String text) 给好友发送消息
- (3) String getMessage()接收消息
  - ①建立 DatagramPacket 自寻址套接字对象
  - ②socket.receive(dp)接收信息
- (4) void newThreadGetMessage()开辟新线程后台获取消息

## 7.17 Src/com/socket/TCPServer.java

- (1) void closeServer()关闭服务器
- (2) boolean startServerTCP()启动 TCP 服务端  
new ServerSocket(serverPort)
- (3) void getConnectionNewThread()开辟新线程不断的接受连接
- (4) void getMessageNewThread(final TCP tcp) 辟新线程后台接收消息并处理，tcp 传过来的 TCP 连接



---

## 7.18 Src/client/socket/CS\_TCP.java

(1) CS\_TCP(String serverIP, int serverPort, Login login, Main main)构造方法, 将登录和主窗体传过来并构造客户端 socket

(2) void dealWithMessage(Flag flag, String message)根据 flag (登录、注册等)不同, 调用不同的方法。

Msg[0]表示标志, msg[1]表示内容。

这里在登录成功后销毁登录窗体并且将登录页面的 cs\_TCP 传递给主界面的 cs\_TCP, 并执行 doUsersInfo()函数, 处理用户信息。

如果登录失败, 返回登录信息。

(3) void doUsersInfo(String message)将获取到的信息置于主界面

(4) void doFriendList(String message)接收好友列表

将传送过来的信息用分隔符进行分开, 并逐个获取好友成员, 最后将好友成员初始化进 main 界面。

在 main 的 initTree 方法中, 调用 MyTree 构造方法。

如:

传送的信息: 所有在线用户,所有不在线用户,我的好友;胡图图&127.0.0.1&5000&0&0,张小丽&127.0.0.1&5001&1&0;蜡笔小新&下线或隐身&0&2&-1;无

首先用 MyTools.SPLIT1 也就是分号“;”分割, 获取 message.split(MyTools.SPLIT1)[0], 也就是“所有在线用户,所有不在线用户,我的好友”分组名称, 再用 MyTools.SPLIT2 进行分组, 将分组名填入 groupNames[]中。

下面再对不同分组内成员进行循环, 将第一个分组内各个用户的信息填入 friendNames[], 后面分组依次类推。

最终将 groupNames, friendNames 传递给 MyTree 构造方法, 构建好友树。

(5) void doQunChat(String message)处理群聊消息。如果有群聊消息, 传给主界面, 如果没有, 则 new PublicMessageFrame (右下角) 弹窗。

(6) void doUnreadMessage(String message)显示未读消息弹窗, 并且响起消息提示音。(调用 MyTools.playMsgSound();方法)

## 7.19 Src/client/socket/CC\_TCP.java

客户端与客户端的 TCP 连接

(1) CC\_TCP(String serverIP, int serverPort, Chat chat)构造方法, 并创建客户端 Socket 连接。

(2) void dealWithMessage(Flag flag, String message)根据 Flag (展示信息, 发送文件)调用不同的方法处理信息。

(3) void showMessage(String message)展示信息, 直接调用 chat.setReceivePaneText(false, message)方法

(4) void doStartSendFile(String message) message 是文件传输端口, 将端口传输给 chat 界面。

---

## 7.20 Src/client/socket/S\_TCP.java

(1) S\_TCP(int serverPort,Main main)构造方法，继承 TCPServer，建立服务端 ServerSocket

(2) void dealWithMessage(Flag flag,String message,TCP tcp)根据客户端发过来的标志（开始聊天、发送信息、发送文件...），调用不同的函数。

(3) void doGetImg(String message,TCP tcp)接收图片

循环现有所有打开的聊天窗体集合，如果聊天好友的名字等于客户端的名字，这个聊天窗口等于当前窗口。并且 TCP 接受图片 getImg () 方法。

(4) void doStartChat(String message,TCP tcp)开始聊天后要做的的事情

Message 是聊天对象的姓名，区分是与自己聊天还是与他人聊天。

(5) void showMessage(String message,TCP tcp)显示信息

(6) void doSendFile(String message,TCP tcp)处理接收的文件，message 是好友传输文件的端口，传给 chat 页面。

(7) void doFace(String message,TCP tcp)处理表情。

## 7.21 Src/client/socket/UDPChat.java

(1) UDPChat(String friendIP, Chat parent)构造方法，根据好友 ip 创建一个聊天

(2) void newThreadGetMessage() 开辟新线程获取消息

## 7.22 Src/client/common/MyLabel.java

MyLabel 构造方法：

JLabel jLabel=null;

String fileName="";

String extension="";

int mode=1;//模式为 1 表示用图片，模式为 2 表示用 Boder

Color backColor=null;//Label 的父容器的背景色

(1) public void addEvent()给 JLabel 添加事件，使其达到规定效果。

## 7.23 Src/client/common/MyTreeIcon.java

主界面上面的分组图标

//如果是：用户名;图片路径，那么就是好友节点

//如果是：用户名;图片路径 1;图片路径 2,那么表示分组节点，

//图片路径 1 表示未展开的分组，图片路径 2 表示展开了的分组

```
1.    public Component getTreeCellRendererComponent(  
2.        JTree tree,  
3.        Object value,  
4.        boolean sel,  
5.        boolean expanded,  
6.        boolean leaf,
```

```

7.             int row,
8.             boolean hasFocus)
9.     {
10.        super.getTreeCellRendererComponent(tree, value, sel, expanded,
11.            leaf, row, hasFocus);
12.        for(String str:nodeImages)
13.        {
14.            String[] temp=str.split(MyTools.SPLIT1);
15.            if(value.toString().startsWith(temp[0])&&!temp[0].equals(""))//注
意 Value 一定要 toString()
16.            {
17.                try
18.                {
19.                    //Image grayImage = GrayFilter.createDisabledImage(ImageIO.
read(Main.class.getResource(temp[1])));
20.                    if(temp.length==2)//如果是: 用户名;图片路径, 那么就是好友节点
21.                        //this.setIcon(new ImageIcon(grayImage));
22.                    this.setIcon(MyTools.getIcon(temp[1]));
23.                    else if(temp.length==3)//如果是: 用户名;图片路径 1;图片路径 2,
那么表示分组节点,
24.                    {
25.                        if(!expanded)
26.                            this.setIcon(MyTools.getIcon(temp[1]));
27.                        else
28.                            this.setIcon(MyTools.getIcon(temp[2]));
29.                    }
30.                }
31.                catch (Exception e)
32.                {
33.                    e.printStackTrace();
34.                }
35.            }
36.        }
37.    }
38.    return this;
39. }

```

根据 nodeImages, 通过分号“;”分割, 并根据上述规则, 判断是好友节点还是分组节点, 并展示不同的图片。

## 7.24 Src/client/frame

全部为前端, 得到静态页面。

## 7.25 Src/client/control/Login.java

(1) void init()初始化页面，设置不同 **Jlabel** 的鼠标监听事件（鼠标放上、点下、松开显示不同图片）。

(2) void initUserStatus()设置用户状态。

(3) void Login () 登录

登录时发送的信息为：

**Flag.LOGIN+MyTools.FLAGEND+name+MyTools.SPLIT1+password+MyTools.SPLIT1+main.getServerPort()+MyTools.SPLIT1+comboBox 状态.getSelectedIndex()**

登录状态#用户名；密码；服务器端口；

如：LOGIN#胡图图;123;5000;0

(4) void mousePress () 追踪鼠标

(5) void mouseDrag(MouseEvent e)处理窗体的拖拽

(6) void addEvent()给窗体添加事件

## 7.26 Src/client/control/Register.java

在注册时也会发送信息，信息为：**Flag.REGISTER+MyTools.FLAGEND//注册标志+txtName.getText()+MyTools.SPLIT1//用户名+new String(pwd.getPassword()+MyTools.SPLIT1//密码+sex+MyTools.SPLIT1//性别+txtEmail.getText()+MyTools.SPLIT1//电子邮件+txtbirthday.getText()+MyTools.SPLIT1//生日+txtSignat.getText()+MyTools.SPLIT1//个性签名+comboBoxHeadImage.getSelectedIndex()//头像索引。**

## 7.27 Src/client/control/Chat.java

(1) 构造方法

根据朋友的 IP 和端口构建 CC\_TCP，实现好友通信。

发送聊天请求时发送的信息为：Flag.START\_CHAT+MyTools.FLAGEND+myName

(2) void init()初始化窗口

设置一些标题、文字、按钮监听事件，并通过 receiveDocument.insertString()将收到的消息展示在聊天框里。（**StyledDocument receiveDocument**）

(3) void initGetFileServer()初始化接收文件服务，默认接收文件端口为 10000。

根据文件传输端口建立服务端 ServerSocket，当发送聊天的 TCP 不为空时，发送消息；当发送聊天的 TCP 为空时，接收聊天的 TCP 发送消息。

再利用 Accept 得到服务端 Socket，开始接收线程。

recieveThread = new **RecieveThread**(new SendFileFrame(), socket)实例化接收文件线程，接收文件。

(4) void setReceivePaneText(boolean isFromMyself, String text)将发送的消息显示在聊天框中。

(5) void **sendMessage()**给好友发送消息，发文本、发表情、发图片。

(6) void setReceivePaneText(boolean isFromMyself, String text)设置接受框文本，如果是自己发信息，则显示自己的名字+时间和信息内容，如果是好友发信息则显示好友姓名+时间和信息内容。

(7) void **sendFile()**发送文件。

根据 String filePath = sendFileFrame.showDialog(JFileChooser.FILES\_ONLY);获取文件路径。

SendFileFrame.java 文件中 showDialog()方法如下:

首先实例化 JfileChooser, 然后调用其 chooser.getSelectedFile().getAbsolutePath()方法获取文件路径。

开始发送文件线程 sendThread () ;。

在 **sendThread** 类的 run 方法中, 调用 startSend 方法。

## 7.28 Src/client/control/RecieveThread.java

接收文件的线程。

- (1) 新建字节输入流
- (2) 读取文件名和文件长度
- (3) 确定接收路径
- (4) Fileout 字节输出流写文件, 并一直更新进度条, 最后将文件 write 完成传输。

## 7.29 Src/client/control/Main.java

- (1) Void StartChat () 开始聊天

根据好友“下线或隐身”或者其他状态, 选择不同的 **Chat 构造方法**。

- (2) void getFriendInfo(ActionEvent e)获取好友资料
- (3) void deleteFriend(ActionEvent e)删除好友
- (4) void gotoChatRoom()进入群聊

# 8 QQ 空间文件说明

## 8.1 数据库连接功能——DBUtil.java

getConnection 函数将尝试连接 My SQL 数据, 并返回一个 connection 对象。

```
1. public static Connection getConnection() {
2.     Connection connection = null;
3.     try {
4.         // 得到数据库连接的相关信息
5.         String dbUrl = "jdbc:mysql://localhost:3306/myblog?useUnicode=true&characterE
ncoding=utf8&serverTimezone=GMT%2B8&useSSL=false&allowPublicKeyRetrieval=tr
ue"
6.     ;
7.         String dbName = "root";
8.         String dbPwd = "123456";
9.         // 得到数据库连接
10.        connection = DriverManager.getConnection(dbUrl, dbName, dbPwd);
11.    } catch (SQLException throwables) {
```

```
12.     throwables.printStackTrace();
13. }
14.
15.     return connection;
16. }
```

## 8.2 对数据库的基本操作——BaseDao.java

1. 执行更新函数 `executeUpdate`: 如果有参数, 则设置参数, 下标从 1 开始 (数组或集合、循环设置参数); 执行更新, 返回受影响的行数

```
1.  public static int executeUpdate(String sql, List<Object> params) {
2.     int row = 0; // 受影响的行数
3.     Connection connection = null;
4.     PreparedStatement preparedStatement = null;
5.
6.     try {
7.         // 得到数据库连接
8.         connection = DBUtil.getConnection();
9.         // 预编译
10.        preparedStatement = connection.prepareStatement(sql);
11.        // 如果有参数, 则设置参数, 下标从 1 开始
12.        if (params != null && params.size() > 0) {
13.            // 循环设置参数, 设置参数类型为 Object
14.            for (int i = 0; i < params.size(); i++) {
15.                preparedStatement.setObject(i+1, params.get(i));
16.            }
17.        }
18.        // 执行更新, 返回受影响的行数
19.        row = preparedStatement.executeUpdate();
20.    } catch (Exception e) {
21.        e.printStackTrace();
22.    } finally {
23.        // 关闭资源
24.        DBUtil.close(null, preparedStatement, connection);
25.    }
26.
27.    return row;
28. }
```

2. `queryRows` 函数, 执行大多数的查询功能, 返回所查询 `sql` 语句所要求的结果。在函数中使用了反射, 从而可以根据数据库中的列名得到对象的值, 更具有普适性。

```

1.  public static List queryRows(String sql, List<Object> params, Class cls) {
2.      List list = new ArrayList();
3.      Connection connection = null;
4.      PreparedStatement preparedStatement = null;
5.      ResultSet resultSet = null;
6.
7.      int fieldNum = 0;
8.      try {
9.          // 得到数据库连接
10.         connection = DBUtil.getConnection();
11.         // 预编译
12.         preparedStatement = connection.prepareStatement(sql);
13.         // 如果有参数，则设置参数，下标从 1 开始
14.         if (params != null && params.size() > 0) {
15.             // 循环设置参数，设置参数类型为 Object
16.             for (int i = 0; i < params.size(); i++) {
17.                 preparedStatement.setObject(i + 1, params.get(i));
18.             }
19.         }
20.         // 执行查询，返回结果集
21.         resultSet = preparedStatement.executeQuery();
22.
23.         // 得到结果集的元数据对象（查询到的字段数量以及查询了哪些字段）
24.         ResultSetMetaData resultSetMetaData = resultSet.getMetaData();
25.         // 得到查询的字段数量
26.         fieldNum = resultSetMetaData.getColumnCount();
27.
28.         // 判断并分析结果集
29.         while (resultSet.next()) {
30.             // 实例化对象
31.             Object object = cls.newInstance();
32.             // 遍历查询的字段数量，得到数据库中查询的每一个列名
33.             for (int i = 1; i <= fieldNum; i++) {
34.                 // 得到查询的每一个列名
35.                 // getColumnLabel(): 获取列名或别名
36.                 // getColumnName(): 获取列名
37.                 String columnName = resultSetMetaData.getColumnLabel(i); // 如果是 tb_user,userId
           字段
38.                 // 通过反射，使用列名得到对应的 field 对象
39.                 Field field = cls.getDeclaredField(columnName);
40.                 // 拼接 set 方法，得到字符串

```

```

41.         String setMethod = "set" + columnName.substring(0, 1).toUpperCase() + columnName.
            substring(1);
42.         // 通过反射，将 set 方法字符串反射成类中对应的 set 方法
43.         Method method = cls.getDeclaredMethod(setMethod, field.getType());
44.         // 得到查询的每一个字段对应的值
45.         Object value = resultSet.getObject(columnName);
46.         // 通过 invoke 方法调用 set 方法
47.         method.invoke(object, value);
48.     }
49.     // 将 Javabean 设置到集合中
50.     list.add(object);
51. }
52.
53. } catch (Exception e) {
54.     e.printStackTrace();
55. } finally {
56.     DBUtil.close(resultSet, preparedStatement, connection);
57. }
58. return list;
59. }

```

3.queryRow 函数：专门服务于用户查询的函数，返回值是一个单独的 Object 对象，避免了后续的处理。

```

1. public static Object queryRow(String sql, List<Object> params, Class cls) {
2.     List list = queryRows(sql, params, cls);
3.     Object object = null;
4.     // 如果集合不为空，则获取查询的第一条数据
5.     if (list != null && list.size() > 0) {
6.         object = list.get(0);
7.     }
8.
9.     return object;
10. }

```

## 8.3 用户对数据库的操作——UserDao.java

1.queryUserByName 函数：通过用户名参数来查询 User 对象，其中调用了 BaseDao 类中的方法。

```

1. public User queryUserByName(String userName) {
2.     // 1. 定义 sql 语句
3.     String sql = "select * from tb_user where uname = ?";

```



```

4.
5.    // 2. 设置参数集合
6.    List<Object> params = new ArrayList<>();
7.    params.add(userName);
8.
9.    // 3. 调用 BaseDao 的查询方法
10.   User user = (User) BaseDao.queryRow(sql, params, User.class);
11.
12.   return user;
13. }

```

2.queryNotesById 函数：通过给定的 id 查找该用户所有的日志。

```

1.  public ArrayList<Note> queryNotesById(int userId) {
2.    // 1. 定义 sql 语句
3.    String sql = "select * from tb_note where userId = ?";
4.
5.    // 2. 设置参数集合
6.    List<Object> params = new ArrayList<>();
7.    List list=new ArrayList();
8.    ArrayList<Note> NoteList=new ArrayList<>();
9.    params.add(userId);
10.
11.   // 3. 调用 BaseDao 的查询方法
12.   list = BaseDao.queryRows(sql, params, Note.class);
13.
14.   //4.转换为 note
15.   for (int i=0;i< list.size();i++){
16.       NoteList.add((Note) list.get(i));
17.   }
18.
19.   return NoteList;
20. }

```

3.InsertIntoNotes 函数：通过给定的 title,content 和 userid, 可以向数据库中插入一行。

```

1.  public int InsertIntoNotes(String title,String Content,int userId){
2.    BaseDao baseDao=new BaseDao();
21.   //1.定义 sql 语言
3.    String sql="insert into tb_note (title,content,userId) values (?,?,?);
4.
5.    //2.配置相关参数
6.    List<Object> params = new ArrayList<>();

```

```

7.    params.add(title);
8.    params.add(Content);
9.    params.add(userId);
10.
11.    //3.调用 BaseDao 中的函数
12.    int row = BaseDao.executeUpdate(sql,params);
13.    return row;
14. }

```

## 8.4 日志主页面——index.jsp

对 JSP 页面 HTML 语言部分不做过多赘述。

### 1. 获取用户参数

```

1.    Object object=request.getParameter("username");
2.
3.    UserDao userDao=new UserDao();
4.    User user=new User();
5.    //调用相关函数，获得 user 对象
6.    user=userDao.queryUserByName(object.toString());

```

2.展现该用户的日志：通过调用 queryNotesById()函数来获取该用户的日志，同时使用 for 循环来创建<li>对象来展示所有的文章标题。

```

1.    <div class="body">
2.        <ul class="clean-list">
3.            <%
4.                ArrayList<Note> NoteList=new ArrayList<>();
5.                //1.调用函数
6.                NoteList=userDao.queryNotesById(user.getUserId());
7.
8.                //2.for 循环来创建对象
9.                for(int i=0;i<NoteList.size();i++){%>
10.                    <li style="font-size: 25px;font-family: 楷体" id="<%=i%>">
11.
12.                        //3.链接到对应的详情页面，并传递日志 id 参数和用户 userId 参数
13.                        <a href="detail.jsp?id=<%=Integer.toString(i)%>&userId=<%=user.
14.                            getUserId%>">
15.                            <%=NoteList.get(i).getTitle()%>
16.                        </a>
17.                    </li>
18.                <%}%>
19.            <%>

```

```
20.  
21.         </ul>  
22. </div>
```

显示效果:

日志列表



## 8.5 日志详情页——detail.jsp

1.通过 request.getParameter()函数来取得对应的 userId，随后查询到对应的 user。

```
1. <%  
2.         //获取 url 传送过来的 userId 参数  
3.         Object userId=request.getParameter("userId");  
4.  
5.         UserDao userDao=new UserDao();  
6.         User user=new User();  
7.         user=userDao.queryUserByName("admin");  
8.  
9.     %>
```

2.通过得到 index.java 传递而来的参数来得到对应用户，随后调用函数来调取对应的文章。

```
1. <div class="col-md-9">  
2.  
3.     <%  
4.         //获取 url 传送过来的 userId 参数  
5.         Object id =request.getParameter("id");  
6.         ArrayList<Note> NoteList=new ArrayList<>();  
7.         NoteList=userDao.queryNotesById(user.getUserId());  
8.     %>
```

```
9.         //展示所取的日志中对应的日志
10.     <div style="font-family: 楷体;font-size: 20px">
11.         <%=NoteList.get(Integer.valueOf(id.toString())).getContent()%>
12.
13.     </div>
```

## 8.6 发布日志——upload.jsp

1. 获取 url 传递过来的 userId 参数，并查询到对应的 user
2. 采用提交表单技术，将内容传递到 Doupload.jsp 中进行处理。

```
1.     //设置 form action 为跳转至 DoUpload.jsp 页面进行处理，并传递 userId 参数
2.     <form action="doUpload.jsp?userId=<%=getuserId%>" method="post">
3.
4.         <table style="margin-left: 10%">
5.             <span>发表个人志</span>
6.
7.             <tr>
8.                 <td>标题: </td>
9.                 <td><input type="text" size="40" name="title"/></td>
10.
11.             </tr>
12.             <tr>
13.                 <td>文章内容: </td>
14.                 <td><textarea type="text" name="content" size="1000" style="margin: 0px; width: 399px; height: 581px;">
15. </textarea></td>
16.             </tr>
17.         </table>
```

展示效果（后续会对页面进行优化）：

发表个人志

标题:

文章内容: 

采薇采薇，薇亦作止。日归日归，岁亦莫止。靡室靡家，玁狁之故。不遑启居，玁狁之故。

采薇采薇，薇亦柔止。日归日归，心亦忧止。忧心烈烈，载饥载渴。我戍未定，靡使归聘。

采薇采薇，薇亦刚止。日归日归，岁亦阳止。王事靡盬，不遑启处。忧心孔疚，我行不来。

## 8.7 处理相关的提交请求——DoUpload.jsp

1. 获取 url 传递过来的 userId,title 和 content 参数
2. 调用 UserDao 中的 InsertIntoNotes 函数，尝试将表单内容插入数据库
3. 倘若提交成功，则转至 success.jsp 页面，提示用户成功
4. 倘若提交失败，则提示用户失败后，返回 upload.jsp

```
1. <div class="col-md-9">
2.
3.     这里是处理界面
4.     <%
5.         //1.设置 request 参数
6.         request.setCharacterEncoding("UTF-8");
7.         response.setContentType("text/html;charset=UTF-8");
8.
9.         //2.获取相关参数
10.        String title = request.getParameter("title");
11.        String content = request.getParameter("content");
12.        int userId= Integer.parseInt(request.getParameter("userId"));
13.        int rows=0;
14.
15.        //3.尝试插入
16.        rows=userDao.InsertIntoNotes(title,content,userId);
17.
18.        //4.判断是否插入成功
19.        if(rows != 0){
20.            request.setAttribute("login",userId);
21.            request.getRequestDispatcher("success.jsp").forward(request,response);
22.        }
```

```

23.         else {
24.             session.setAttribute("errors", "用户名密码不正确，请从新输入！");
25.             response.sendRedirect("upload.jsp?userId=<%=user.getUserId()%>");
26.         }
27.         %>
28.     用户名: <%=user.getUsername() %>
29. </div>

```

## 8.8 提示成功——successs.jsp

```

1. <div class="col-md-9">
2.
3.     <h2>欢迎来到我的世界!!By<%=request.getAttribute("login") %></h2>
4.     <button>
5.         <a href="index.jsp">
6.             回到主页！
7.         </a>
8.     </button>>
9.
10. </div>

```

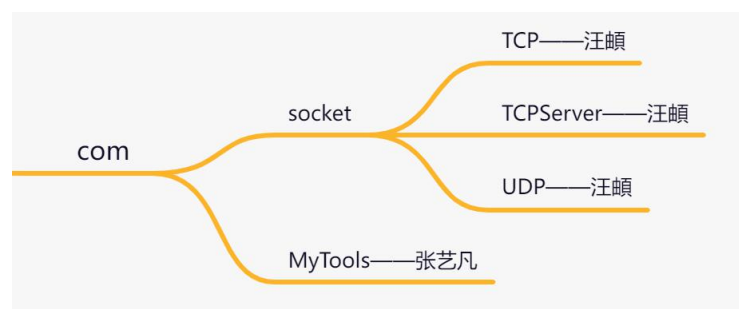
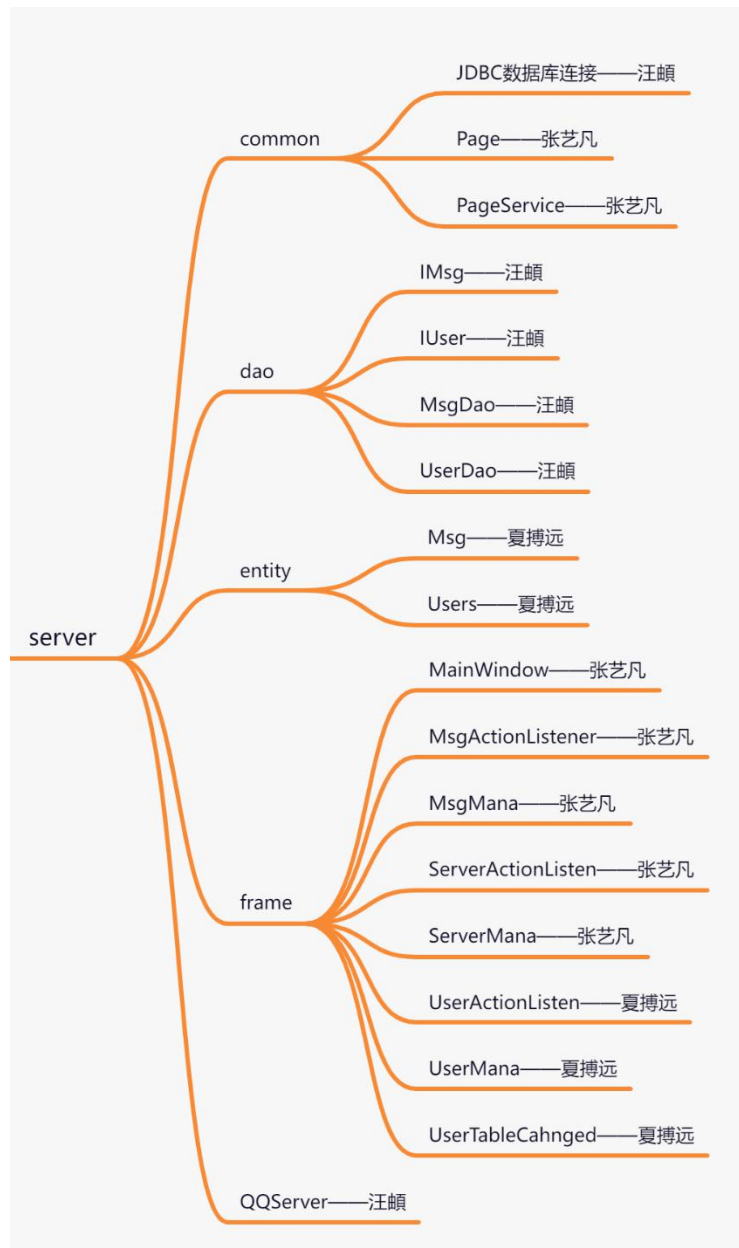
发布成功效果：

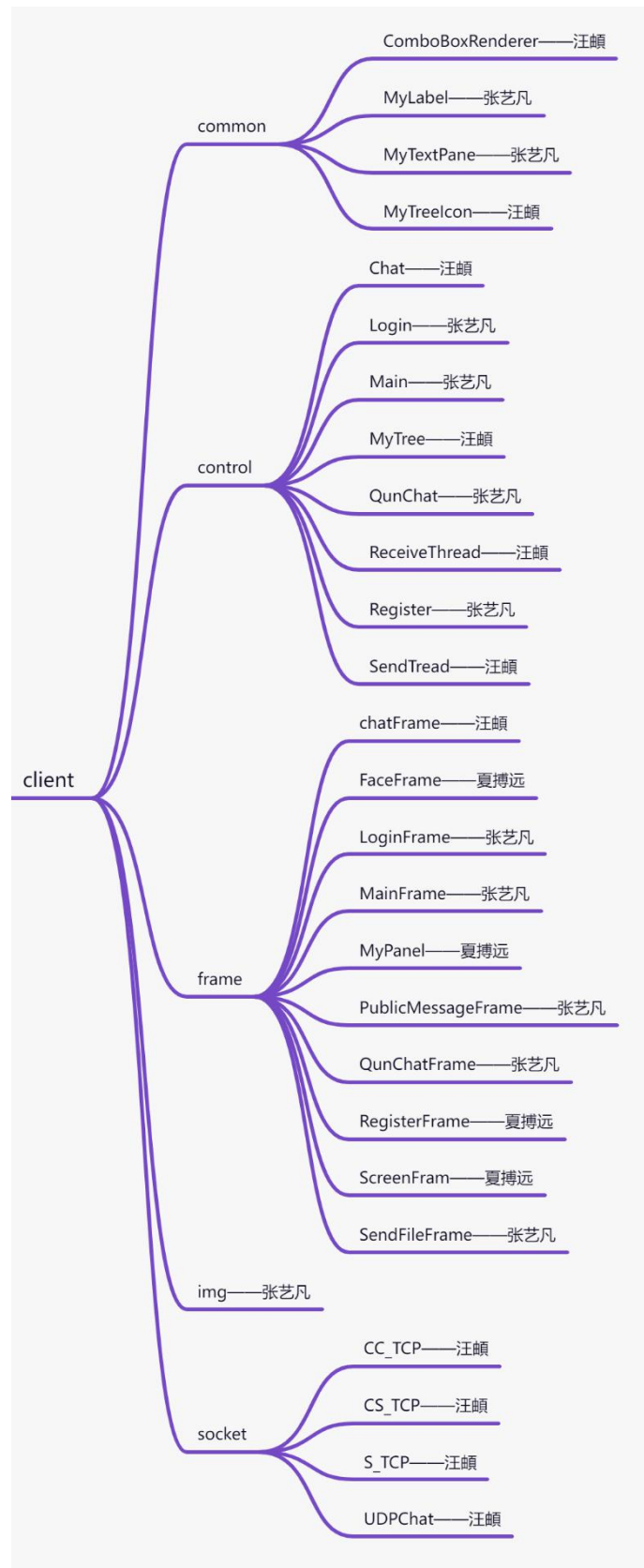
欢迎来到我的世界!!By124

[回到主页!](#) >

## 9 小组分工

1. QQ 空间部分：夏搏远
2. QQ 主体部分：





## 10 后续改进点

1.注重版权问题，避免使用其他软件的图标；



- 
- 2.实现离线传文件功能;
  - 3.页面更加精简化, 实现更多功能的互通互用;
  - 4.实现 qq 空间和 qqGUI 更好的结合