# Java Socket 网络通信

有客户端与服务器端，采用 socket 通信。服务器转发客户端的消息。

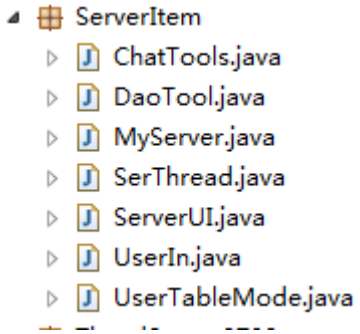
## 协议

第一个字符，表示消息类型。

1：注册，2：登录，3：私聊，4：群聊，5：文件

第二个字符起，表示消息内容。

## 服务器




首先创建服务器，写在 MySever 类中继承 Thread：

```
ServerSocket server = new ServerSocket(port);
// 绑定在指定端口号的服务器
```

然后等待客户机连接：

```
client = server.accept();// 阻塞，连上后返回 Socket 对象
```

写好服务器界面，在界面中输入端口好，并启动服务器：

```
    public void actionPerformed(ActionEvent e) {

        String text = jt_port.getText();
        if (text != null && !text.equals("")) {
            int port = Integer.parseInt(text);
            MyServer ms = new MyServer(port, textShow);
```

```
                ms.start();
            } else {
                JOptionPane.showMessageDialog(ServerUI.this, "请输入端口号！");
            }
        }
    });
```

将对客户端消息进行处理的操作封装在 SerThread 类中，在构造函数中获得输入输出流：

```
public SerThread(Socket client, JTextArea textShow) {
    this.client = client;
    this.textShow = textShow;
    try {
        os = client.getOutputStream();
        is = client.getInputStream();
        br = new BufferedReader(new InputStreamReader(is));
    } catch (IOException e) {
        e.printStackTrace();
    }

}
```

并读第一个字符判断消息类型，进行处理：

注册：

```
case "1":// 注册
String userinfo = s.substring(1);// 从第二个字符开始读起
String us[] = userinfo.split("#");// 用#号分割
user = new UserIn(us[0], us[1]);//创建用户对象
boolean flag = DaoTool.join(us[0], us[1]);// 将账号密码存储，并返回存储情况
if (flag) {
    os.write("true\r\n".getBytes());// 发送给客户机注册成功与否
    } else {
    os.write("false\r\n".getBytes());
    }
    break;
```

存储，在 DaoTool 类中使用静态块，hashMap 来存用户账号和密码。 用户类，封装用户信息。

登录：

```
case "2":// 登录
            s = s.substring(1);
            String str[] = s.split("#");
            user = new UserIn(str[0], str[1]);
            flag = DaoTool.check(user);
            if (flag) {
                os.write("true\r\n".getBytes());
            } else {
                os.write("false\r\n".getBytes());
            }
            break;
```

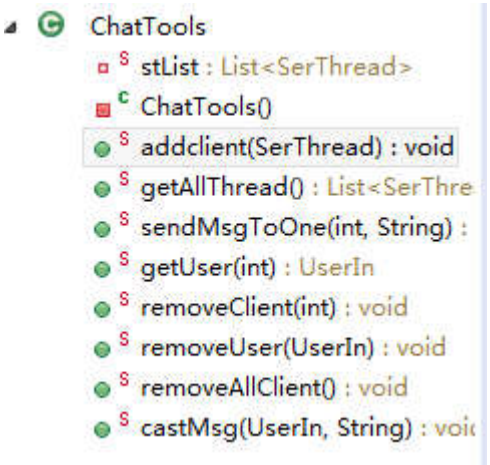
群聊：

```
case "4":// 群聊
            s = s.substring(1);
            ChatTools.castMsg(user, s);
            textShow.append(user.getname()+"对所有人说"+s);
            break;
```

使用 ChatTools 类，管理用户对应的线程对象，并发送消息。

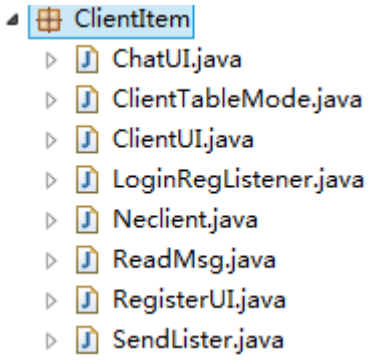
```
private static List<SerThread> stList =
 new ArrayList(); // 保存处理线程的对象队列
```

方法如下：




其中 castMsg 方法是发送一条消息到所有客户机。

## 客户端



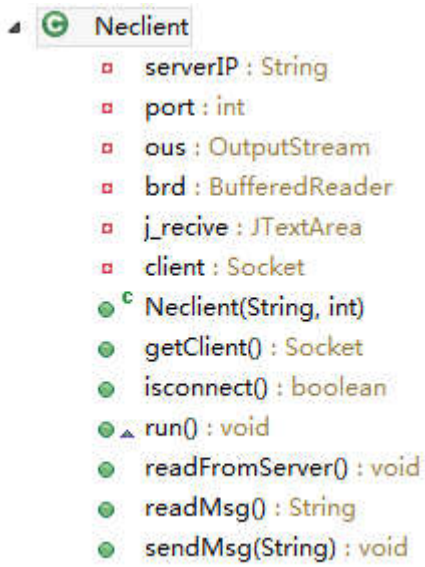

需要做的有注册，登录，聊天界面的实现及跳转，以及连接服务器和读取及发送消息。

连接服务器操作如下：

```
// 创建连接对象
    nc = new Neclient("localhost", 1111);
    // 连接上服务器
    if (nc.isconnect()) { // 连接上
        String msg = 2 + name + "#" + psd;
        nc.sendMsg(msg);// 向服务器发送登录消息
        msg = nc.readMsg();// 读取服务器发给客户机用户登录是否成功
        boolean flag = Boolean.parseBoolean(msg);
        if (flag) {
//              JOptionPane.showMessageDialog(frame, "登录成功");
            ChatUI cui = new ChatUI(nc);//显示聊天界面,传入连接对象
            cui.showChatUI();
            frame.dispose();//关闭登陆界面
        } else {
            JOptionPane.showMessageDialog(frame, "登录失败");
        }
    } else {
        JOptionPane.showMessageDialog(frame,
                "客户机无法连接上服务器，请检查是否有网络或者防火墙是否关闭！");
    }
}
```

从从服务器读取消息的操作写在一个线程类 Neclient 中：

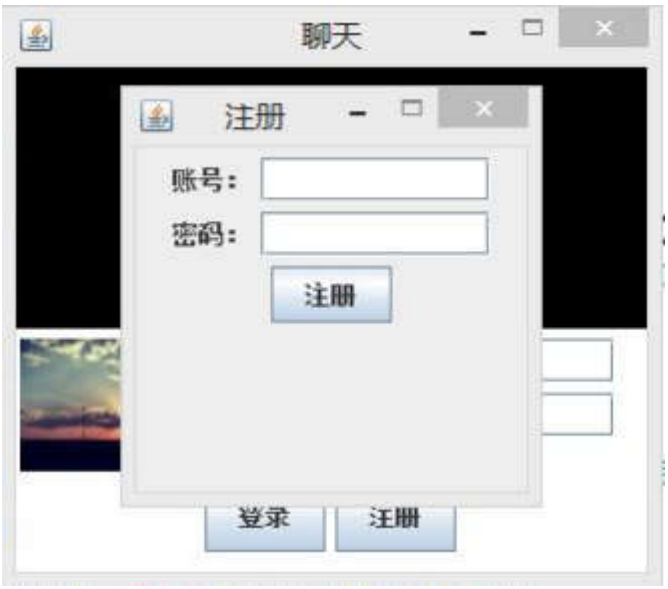
方法如下：

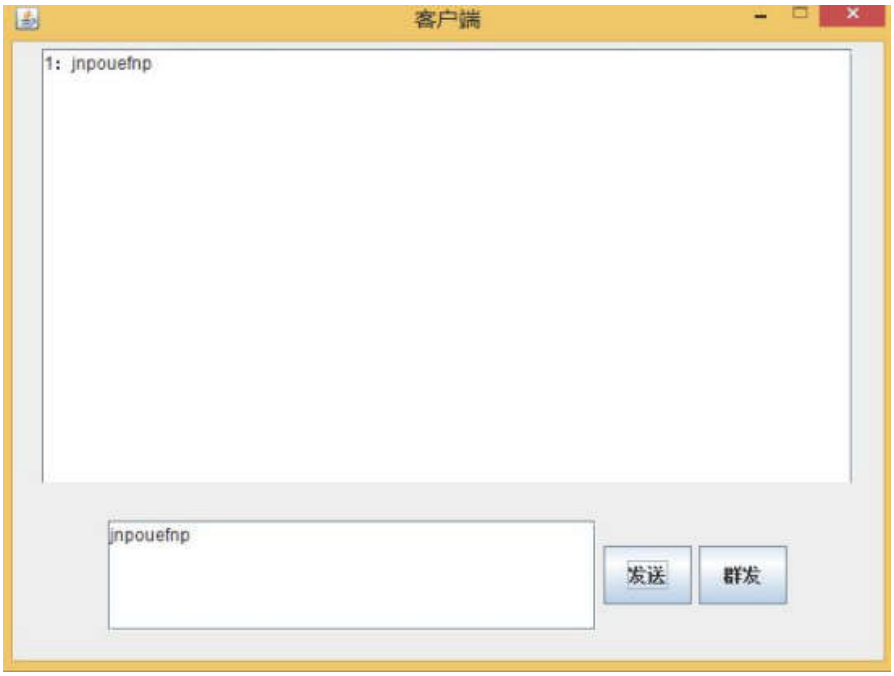


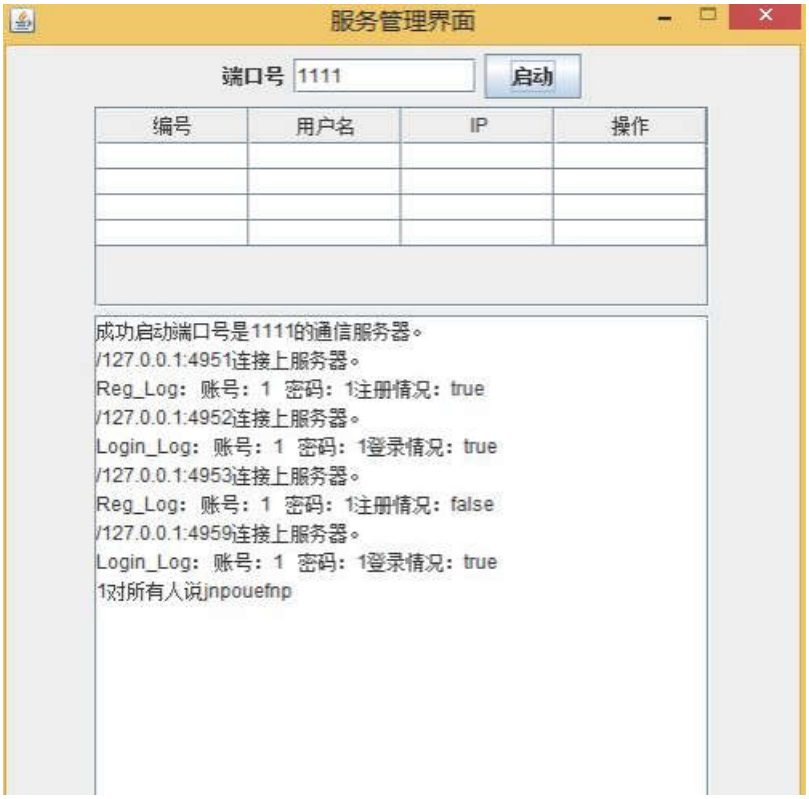

```
/** 连上服务器，判断是否成功 */
public boolean isconnect() {
    try {
        client = new Socket(this.serverIP, this.port);// 创建服务器连接对象
        InputStream ins = client.getInputStream(); // 得到输入输出流对象
        ous = client.getOutputStream();
        brd = new BufferedReader(new InputStreamReader(ins)); // 可以读取一行字符串
        return true;
    } catch (IOException e) {
        e.printStackTrace();
    }
    return false;
}

// 线程中读取服务器发来的消息
public void run() {
    while (true) {
        readFromServer();
}
}
```

## 实现效果：


聊天
注册
账号：
密码：
注册
登录
注册


客户端
1：jnpouefnp
jnpouefnp
发送
群发


服务管理界面
端口号
1111
启动
编号
用户名
IP
操作
成功启动端口号是1111的通信服务器。
/127.0.0.1:4951连接上服务器。
Reg_Log：账号：1 密码：1注册情况：true
/127.0.0.1:4952连接上服务器。
Login_Log：账号：1 密码：1登录情况：true
/127.0.0.1:4953连接上服务器。
Reg_Log：账号：1 密码：1注册情况：false
/127.0.0.1:4959连接上服务器。
Login_Log：账号：1 密码：1登录情况：true
1对所有人说jnpouefnp