客户端

View

```
public class QQView {

  public static void main(String[] args) throws UnknownHostException,
ClassNotFoundException, IOException {
    new QQView().mainMenu();
  }

  private boolean loop = true;
  private String key = "";
  // 用户登录连接服务
  private UserClientService userClientService = new UserClientService();
  // 用户发送信息服务
  private MessageClientService messageClientService = new MessageClientService();
  // 用户发送文件服务
  private FileClientService fileClientService = new FileClientService();

  private void mainMenu() throws UnknownHostException, ClassNotFoundException,
IOException {
    while (loop) {
      System.out.println("=================一级菜单");
      System.out.println("1 登录系统");
      System.out.println("9 退出系统");
      System.out.print("您的选择");
      key = Utility.readString(1);
      switch (key) {
        case "1":
          System.out.print("用户名: ");
          String userId = Utility.readString(50);
          System.out.print("密码: ");
          String pwd = Utility.readString(50);
          // 需要到服务端验证是否正确
          if (userClientService.checkUser(userId, pwd)) {
            System.out.println("========欢迎" + userId);
            while (loop) {
              System.out.println("============二级菜单");
              System.out.println("1 在线用户列表");
              System.out.println("2 群发消息");
              System.out.println("3 私聊消息");
              System.out.println("4 发送文件");
              System.out.println("9 退出系统");
              System.out.print("您的选择");
              key = Utility.readString(1);
              switch (key) {
                case "1":
                  userClientService.onlineFriendList();
                  break;
                case "2":
                  System.out.println("请输入要发送的信息: ");
                  String content1 = Utility.readString(50);
                  messageClientService.sendMessageToAll(content1, userId);
                  break;
                case "3":
                  System.out.println("请输入一个在线用户号: ");
                  String getterId = Utility.readString(50);
                  System.out.println("请输入要发送的信息: ");
                  String content = Utility.readString(50);
                  messageClientService.sendMessageToOne(content, userId, getterId);
                  break;
                case "4":
                  System.out.println("发送文件的路径: ");
                  String src = Utility.readString(50);
                  System.out.println("发送文件的名称: ");
                  String fileName = Utility.readString(50);
                  System.out.println("发送给谁: ");
                  getterId = Utility.readString(50);
                  System.out.println("发送文件的类型: ");
                  String fileType = Utility.readString(50);
                  fileClientService.sendFileToOne(src, fileName, userId, getterId,
fileType);

                  break;
                case "9":
                  userClientService.clientExit();
                  loop = false;
                  break;

                default:
                  break;
              }
            }

          } else {
            System.out.println("登陆失败");
          }
          break;
        case "9":
          loop = false;
          break;

        default:
          break;
      }

    }
  }
}
```

UserClientService 登录连接等服务

```
public class UserClientService {
  private User user = new User();

  private Socket socket;

  public boolean checkUser(String userId, String pwd) throws UnknownHostException,
IOException, ClassNotFoundException {
    boolean b = false;
    user.setUserId(userId);
    user.setPwd(pwd);
    socket = new Socket(InetAddress.getByName("127.0.0.1"), 9999);
    ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
    oos.writeObject(user);

    ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
    Message msg = (Message) ois.readObject();
    if (msg.getMessageType().equals(MessageType.MESSAGE_LOGIN_SUCCESSED)) {// 登陆
成功

      // 创建一个和服务端保持通信的线程，主要是监听接收的消息
      ClientConnectServerThread clientConnectServer = new
ClientConnectServerThread(socket, user.getUserId());
      clientConnectServer.start();
      //放入模拟线程池中

ManageClientConnectServerThread.addClientConnectServerThread(user.getUserId(),
clientConnectServer);
      b = true;

    }else{
      //登陆失败
      socket.close();
    }
    return b;
  }

  // 获取在线用户列表
  public void onlineFriendList() {
    //发送一个message MESSAGE_GET_ONLINE_FRIEND
    Message msg = new Message(user.getUserId(), null, null, null,
MessageType.MESSAGE_GET_ONLINE_FRIEND);
    try {
      //从线程Hashmap拿取
      ObjectOutputStream oos = new ObjectOutputStream(

ManageClientConnectServerThread.getClientConnectServerThread(user.getUserId()).getS
ocket().getOutputStream());
      oos.writeObject(msg);

    } catch (IOException e) {
      e.printStackTrace();
    }
  }

  //无异常退出
  public void clientExit(){
    Message msg = new Message(user.getUserId(), null, null, null,
MessageType.MESSAGE_CLIENT_EXIT);
    try {
      ObjectOutputStream oos = new ObjectOutputStream(

ManageClientConnectServerThread.getClientConnectServerThread(user.getUserId()).getS
ocket().getOutputStream());
      oos.writeObject(msg);
      System.out.println("结束进程");
      System.exit(0);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

ClientConnectServerThread 客户端监听服务

```
public class ClientConnectServerThread extends Thread {
  private Socket socket;
  private String userId;

  // public ClientConnectServerThread(Socket socket) {
  // this.socket = socket;
  // }

  public ClientConnectServerThread(Socket socket, String userId) {
    this.socket = socket;
    this.userId = userId;
  }

  @Override
  public void run() {
    while (true) {
      // 接收做多线程 因为要一直和等待接收信息，则需要while循环
      try {
        ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
        Message msg = (Message) ois.readObject();

        // 服务器返回的在线用户列表
        if (msg.getMessageType().equals(MessageType.MESSAGE_RET_ONLINE_FRIEND)) {
          String[] onlineUsers = msg.getContent().split(" ");
          System.out.println("==========当前用户列表===========");
          for (String string : onlineUsers) {
            System.out.println("用户: " + string);
          }
        } else if (msg.getMessageType().equals(MessageType.MESSAGE_COMM_MES)) {//接
收私法消息
          System.out.println(msg.getSender() + "向你发来了一条" + msg.getContent() +
"信息。日期为" + msg.getSendTime());
        } else if (msg.getMessageType().equals(MessageType.MESSAGE_TOALL_MES)) {//
接收群发消息
          System.out.println(msg.getSender() + "向你群发来了一条" + msg.getContent()
+ "信息。日期为" + msg.getSendTime());
        } else if (msg.getMessageType().equals(MessageType.MESSAGE_FILE_MES)) {//接
收文件
          String path = "E:/";
          FileOutputStream fileOutputStream = new FileOutputStream(path +
msg.getFileName() + "." + msg.getFileType());
          fileOutputStream.write(msg.getFileBytes());
          fileOutputStream.flush();
          fileOutputStream.close();
          System.out.println("接收文件成功");
        }

      } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        try {
          ManageClientConnectServerThread.removeClientConnectServerThread(userId);
          socket.close();
          break;
        } catch (IOException e1) {
          // TODO Auto-generated catch block
          e1.printStackTrace();
        }
      }
    }
  }

  public Socket getSocket() {
    return socket;
  }
```

```
    public void setSocket(Socket socket) {
        this.socket = socket;
    }

}
```

ManageClientConnectServerThread 模拟监听线程池

```
public class ManageClientConnectServerThread {
    // key 用户id
    // 存放线程,目的是为了实现多用户登录
    private static HashMap<String, ClientConnectServerThread> hm = new HashMap<>();

    // 添加到hashmap
    public static void addClientConnectServerThread(String userId,
ClientConnectServerThread clientConnectServerThread) {
        hm.put(userId, clientConnectServerThread);
    }

    // 从中取出
    public static ClientConnectServerThread getClientConnectServerThread(String
userId) {
        return hm.get(userId);
    }

    public static void removeClientConnectServerThread(String userId) {
        hm.remove(userId);
    }

}
```

MessageClientService 客户端发送消息服务

```
public class MessageClientService {

    //发送消息给某人
    public void sendMessageToOne(String content, String senderId, String getterId) {
        Message msg = new Message();
        msg.setSender(senderId);
        msg.setGetter(getterId);
        msg.setContent(content);
        msg.setSendTime(new Date());
        msg.setMessageType(MessageType.MESSAGE_COMM_MES);
        System.out.println(msg.getSender() + "发给" + msg.getGetter() + "的消息" +
msg.getContent());

        try {
          ObjectOutputStream objectOutputStream = new ObjectOutputStream(

ManageClientConnectServerThread.getClientConnectServerThread(senderId).getSocket().
getOutputStream());
          objectOutputStream.writeObject(msg);
        } catch (IOException e) {
          e.printStackTrace();
        }
    }

    //群发消息
    public void sendMessageToAll(String content, String senderId) {
        Message msg = new Message();
        msg.setSender(senderId);
        msg.setContent(content);
        msg.setSendTime(new Date());
        msg.setMessageType(MessageType.MESSAGE_TOALL_MES);

        System.out.println(msg.getSender() + "群发的消息" + msg.getContent());
        try {
          ObjectOutputStream objectOutputStream = new ObjectOutputStream(

ManageClientConnectServerThread.getClientConnectServerThread(senderId).getSocket().
getOutputStream());
          objectOutputStream.writeObject(msg);
        } catch (IOException e) {
          e.printStackTrace();
        }
    }

}
```

FileClientService 客户端发送文件服务

```
public class FileClientService {
  public void sendFileToOne(String src, String fileName, String senderId, String
getterId, String fileType) {
    Message msg = new Message();
    msg.setSender(senderId);
    msg.setGetter(getterId);
    msg.setFileName(fileName);
    msg.setFileType(fileType);
    msg.setSendTime(new Date());
    msg.setMessageType(MessageType.MESSAGE_FILE_MES);
    System.out.println(msg.getSender() + "发给" + msg.getGetter() + "的文件" + "路径
为" + src);

    try {
      int len = (int) new File(src).length();
      byte[] b = new byte[len];
      FileInputStream fileInputStream = new FileInputStream(src);
      fileInputStream.read(b);
      msg.setFileBytes(b);
      msg.setFileLen(len);
      ObjectOutputStream objectOutputStream = new ObjectOutputStream(

ManageClientConnectServerThread.getClientConnectServerThread(senderId).getSocket().
getOutputStream());
      objectOutputStream.writeObject(msg);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

服务端

服务端主服务

```
public class QQserver {

  public static void main(String[] args) throws ClassNotFoundException, IOException
{
    QQserver qQserver = new QQserver();
  }

  // 可以使用并发的集合ConcurrentHashMap,没有线程安全问题
  //模拟用户数据库
  static HashMap<String, User> userList = new HashMap<>();

  static {
    userList.put("100", new User("100", "123456"));
    userList.put("200", new User("200", "123456"));
    userList.put("300", new User("300", "123456"));
    userList.put("400", new User("400", "123456"));
  }

  private ServerSocket serverSocket;
  private boolean loop = true;
  private SendToAllService sendToAllService = new SendToAllService();

  public QQserver() throws IOException, ClassNotFoundException {
    System.out.println("服务端在9999进行监听");
    // 端口可以写在配置文件
    serverSocket = new ServerSocket(9999);
    //等待连接线程
    new Thread(new UserServiceThread(serverSocket)).start();
    while (loop) {
      //服务器额外功能
      System.out.println("服务器界面");
      System.out.println("1 新闻推送");
      String key = Utility.readString(50);
      switch (key) {
        case "1":
          sendToAllService.sendNewsToAllService();
          break;

        default:
          break;
      }
    }
  }
}
```

UserServiceThread 用户连接验证服务

```
public class UserServiceThread implements Runnable {
  private ServerSocket serverSocket;

  public UserServiceThread(ServerSocket serverSocket) {
    this.serverSocket = serverSocket;
  }

  @Override
  public void run() {
    try {
      while (true) {
        Socket socket = serverSocket.accept();
        ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
        User user = (User) ois.readObject();

        ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
        System.out.println(user.getUserId() + "尝试连接");

        // 验证用户是否有效

        if (checkUser(user.getUserId(), user.getPwd())) {
          // 有用户连接
          System.out.println(user.getUserId() + "已连接");
          oos.writeObject(new Message(null, null, null, new Date(),
MessageType.MESSAGE_LOGIN_SUCCESSED));
```

```
          ServerConnectClientThread serverConnectClientThread = new
ServerConnectClientThread(socket, user.getUserId());
          serverConnectClientThread.start();

ManageServerConnectClientThread.addServerConnectClientThread(user.getUserId(),
serverConnectClientThread);
        } else {
          oos.writeObject(new Message(null, null, null, new Date(),
MessageType.MESSAGE_LOGIN_FAIL));
          socket.close();
        }

      }
    } catch (Exception e) {
    } finally {
      try {
        serverSocket.close();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }

  //模拟验证用户账号密码
  private static boolean checkUser(String userId, String pwd) {
    User user = QQserver.userList.get(userId);
    if (user == null) {
      return false;
    }
    if (!(user.getPwd().equals(pwd))) {
      return false;
    }
    return true;
  }
}
```

ServerConnectClientThread 服务端接收用户请求分发监听服务

```
public class ServerConnectClientThread extends Thread {
  private Socket socket;
  private String userId;

  public ServerConnectClientThread(Socket socket, String userId) {
    this.socket = socket;
    this.userId = userId;
  }

  @Override
```

```
    public void run() {
      // 可以发送或接收消息
      while (true) { //主要负责用户与用户的请求分发
        try {
          ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
          Message msg = (Message) ois.readObject();

          // 用户在线列表信息
          if (msg.getMessageType().equals(MessageType.MESSAGE_GET_ONLINE_FRIEND)) {
            String onlineUsers = ManageServerConnectClientThread.onlineUsers();
            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(socket.getOutputStream());
            Message message_online = new Message(null, msg.getSender(), onlineUsers,
null,
                MessageType.MESSAGE_RET_ONLINE_FRIEND);
            objectOutputStream.writeObject(message_online);
          } else if (msg.getMessageType().equals(MessageType.MESSAGE_CLIENT_EXIT))
{//无异常退出
            System.out.println(msg.getSender() + "将要退出系统");

ManageServerConnectClientThread.removeServerConnectClientThread(msg.getSender());
            socket.close();
            break;
          } else if (msg.getMessageType().equals(MessageType.MESSAGE_COMM_MES)) {//私
发消息
            System.out.println(msg.getSender() + "发给" + msg.getGetter() + "的消息" +
msg.getContent());
            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(ManageServerConnectClientThread

.getServerConnectClientThread(msg.getGetter()).getSocket().getOutputStream());
            objectOutputStream.writeObject(msg);//
          } else if (msg.getMessageType().equals(MessageType.MESSAGE_TOALL_MES)) {//
群发消息
            HashMap<String, ServerConnectClientThread> hm =
ManageServerConnectClientThread.getHm();
            Iterator<String> iterator = hm.keySet().iterator();
            while (iterator.hasNext()) {
              String userId = iterator.next().toString();
              if (!(userId.equals(msg.getSender()))) {
                ObjectOutputStream objectOutputStream = new
ObjectOutputStream(ManageServerConnectClientThread

.getServerConnectClientThread(userId).getSocket().getOutputStream());
                objectOutputStream.writeObject(msg);
              }
            }
          } else if (msg.getMessageType().equals(MessageType.MESSAGE_FILE_MES)) {//用
户间文件传输

            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(ManageServerConnectClientThread

.getServerConnectClientThread(msg.getGetter()).getSocket().getOutputStream());
            objectOutputStream.writeObject(msg);

          }
          // 对msg处理
        } catch (Exception e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
          try {
            ManageServerConnectClientThread.removeServerConnectClientThread(userId);
            socket.close();
            break;
          } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
          }
        }
      }
    }

    public Socket getSocket() {
      return socket;
    }

    public void setSocket(Socket socket) {
      this.socket = socket;
    }

    public String getUserId() {
      return userId;
    }

    public void setUserId(String userId) {
      this.userId = userId;
    }

}
```

ManageServerConnectClientThread 服务端模拟监听线程池

```
public class ManageServerConnectClientThread {
  private static HashMap<String, ServerConnectClientThread> hm = new HashMap<>();

  public static void addServerConnectClientThread(String userId,
ServerConnectClientThread serverConnectClientThread) {
    hm.put(userId, serverConnectClientThread);
  }

  // 从中取出
  public static ServerConnectClientThread getServerConnectClientThread(String
userId) {
    return hm.get(userId);
  }

  public static void removeServerConnectClientThread(String userId) {
    hm.remove(userId);
  }

  public static String onlineUsers() {
    Set<String> keySet = hm.keySet();
    StringBuffer stringBuffer = new StringBuffer();
    for (String string : keySet) {
      stringBuffer.append(string + " ");
    }
    return stringBuffer.toString();
  }

  public static HashMap<String, ServerConnectClientThread> getHm() {
    return hm;
  }

}
```

SendToAllService 服务端新闻推送服务

```
public class SendToAllService{

  public void sendNewsToAllService() {
    while (true) {
      System.out.println("请输入需要推出的新闻,exit退出");
      String news = Utility.readString(50);
      if ("exit".equals(news)) {
        break;
      }
      Message msg = new Message();
      msg.setContent(news);
      msg.setSendTime(new Date());
      msg.setSender("服务器");
      msg.setMessageType(MessageType.MESSAGE_TOALL_MES);

      HashMap<String, ServerConnectClientThread> hm =
ManageServerConnectClientThread.getHm();
      Iterator<String> iterator = hm.keySet().iterator();
      while (iterator.hasNext()) {
        String userId = iterator.next().toString();
          try {
            ObjectOutputStream objectOutputStream = new
ObjectOutputStream(ManageServerConnectClientThread

.getServerConnectClientThread(userId).getSocket().getOutputStream());
            objectOutputStream.writeObject(msg);
          } catch (IOException e) {
            // TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }
        }
    }
  }
}
```