

**信息科学与工程学院**



**计算机网络原理课程设计**

课程设计题目： 一个网络聊天系统的设计与实现

学 生 姓 名 ： 樊 列 龙

学 号 ： 0909113224

班 级 ： 物联网工程1103班

指 导 老 师 ： 桂 劲 松

完 成 时 间 ： 2014年01月13日

[第一节 实验平台和器材准备 1](#_Toc30892)

[1.1 硬件配置 1](#_Toc11050)

[1.2 实验平台 1](#_Toc17595)

[第二节 需求分析 2](#_Toc13338)

[2.1 实验要求 2](#_Toc1078)

[2.1.1 基本要求 2](#_Toc484)

[2.1.2 基本功能要求 2](#_Toc9417)

[2.1.3 详细功能要求 2](#_Toc30337)

[三、概要设计 4](#_Toc26517)

[3.1 JICQ部署和原理图 4](#_Toc24917)

[3.1.1 软件简介和工作原理 4](#_Toc19985)

[3.2 JICQ基本用例图、交互图和序列图 5](#_Toc30484)

[3.2.1 JICQ用例图 5](#_Toc30422)

[3.2.2 用户注册活动图 6](#_Toc28129)

[3.2.3 用户登陆活动图 6](#_Toc12731)

[3.2.4 客户机之间通信活动图 7](#_Toc1905)

[3.2.5 添加好友活动图 8](#_Toc3570)

[3.2.6 删除好友活动图 9](#_Toc24885)

[3.3 数据库模型 10](#_Toc31244)

[3.3.1 JICQ存储设计 10](#_Toc25299)

[3.4 系统类包 11](#_Toc3336)

[3.5 JICQ线程控制 11](#_Toc9597)

[3.6 网页客户端设计 12](#_Toc2258)

[3.7 工程架构设计 12](#_Toc17124)

[3.7.1 JavaEE MVC框架简介 12](#_Toc27114)

[3.7.2 项目用用MVC框架部署 13](#_Toc12503)

[第四节 详细设计 14](#_Toc13571)

[4.3 数据库详细设计 14](#_Toc14035)

[4.3.1 数据库SQL代码： 14](#_Toc31451)

[4.2 Java 后端控制实现 17](#_Toc13576)

[4.2.1 服务器端架构实施文件 17](#_Toc2977)

[4.2.2 Socket核心代码 17](#_Toc28839)

[4.6 Android客户端实现 25](#_Toc24215)

[4.6.1 实现关键代码 25](#_Toc30575)

[第五节 实验结果分析 27](#_Toc6112)

[5.1 实验结果展示 27](#_Toc31955)

[5.1.1 Web用户注册功能 27](#_Toc9620)

[5.1.1 服务器后端 28](#_Toc25911)

[5.1.2 客户端 28](#_Toc30347)

[第六节 错误调试 35](#_Toc6154)

[6.1 数据库连接错误 35](#_Toc8328)

[6.1.1 解决方案 35](#_Toc14288)

[6.2 POST乱码解决方式 36](#_Toc14500)

[6.2.1 错误表现 36](#_Toc5972)

[6.2.2解决方案 36](#_Toc30763)

[6.3 Socket连接错误 36](#_Toc15775)

[6.3.1 错误表现 36](#_Toc22859)

[6.3.2产生错误的原因 37](#_Toc5730)

[6.3.3 解决方式 37](#_Toc29246)

[第八节 实验总结 38](#_Toc6900)

[8.1 对于计算机网络方面的总结： 38](#_Toc19168)

[8.2 对于Java方面的总结 38](#_Toc24606)

[第九节 参考资料 39](#_Toc1996)

[附件： 40](#_Toc5949)

# 第一节 实验平台和器材准备

## 1.1 硬件配置

|  |  |  |
| --- | --- | --- |
| 服务器 | CPU | PIII1G以上 |
| 内存 | 512M以上 |
| 硬盘 | 60G以上，双IDE硬盘,做镜像 |
| 客户端 | CPU | PIII300 以上 |
| 内存 | 512M以上 |
| 硬盘 | 120G以上 |

## 1.2 实验平台

|  |  |
| --- | --- |
| 软件开发环境 | MyEclipse9.0 |
| 数据库 | SQL Server2008 |
| JICQ-WWW服务器环境 | Tomcat 7.0 |
| JICQ 测试环境 | FireFox2.5, IE 9, Google Chrome，Windows7 |
| 数据库开发环境 | SQL Server Management Studio |
| 数据库开发环境 | Power Designer 16 |
| 其他工具 | Visual Paradigm |

# 第二节 需求分析

## 2.1 实验要求

### 2.1.1 基本要求

设计中要求综合运用所学知识，上机解决一些与实际应用结合紧密的、规模较大的问题，通过分析、设计、编码、调试等各环节的训练，深刻理解、牢固掌握网络原理的基本技术，掌握分析、解决实际问题的能力。

## 2.1.2 基本功能要求

|  |  |
| --- | --- |
| 1、服务模式 | 实现网络聊天的功能，采用Windows Socket（或Java Socket）编程，服务器与客户端采用了TCP/IP连接方式，在设计聊天方案时，可将所有信息发往服务器端，再由服务器进行处理，服务器端是所有信息的中心。 |
| 2、保存信息 | 服务器端要保存用户信息，可利用数据库来实现这一功能，需要建立用户信息数据库。在客户端可采用文件系统保存聊天纪录和用户号码。 |
| 3、提示上线 | 可通过闪动托盘图标和播放不同的音乐提示信息到来及好友上线等信息。可通过建立消息链表来保存用户接收的各种消息。 |

## 2.1.3 详细功能要求

表 2.1.3 详细功能要求

|  |  |
| --- | --- |
| 1、登录功能。 | 建立与服务器的连接并登陆，能显示登陆错误信息。 |
| 2、一对一和多对多聊天 | 客户可以通过服务器转发，实现一对一和多对多聊天。 |
| 3、用户的状态 | 客户端程序应该可以实时显示目前其它用户的状态。 |
| 4、界面显示 | 将在线好友显示在好友列表中，并实现系统托盘，加入工具栏。 |
| 5、聊天记录 | 能保存聊天纪录，并能察看聊天纪录。 |
| 6、友好的界面 | 应该具有易用、美观的图形界面 |
| 7、信息提示 | 闪动托盘图标提示到来信息，并播放不同音乐来提示。 |
| 8、其他功能 | 用户登陆成功，将保存其号码，以便下次登陆时，不必再输入而可以直接选择，显示登陆时间。 |

# 三、概要设计

## 3.1 JICQ部署和原理图

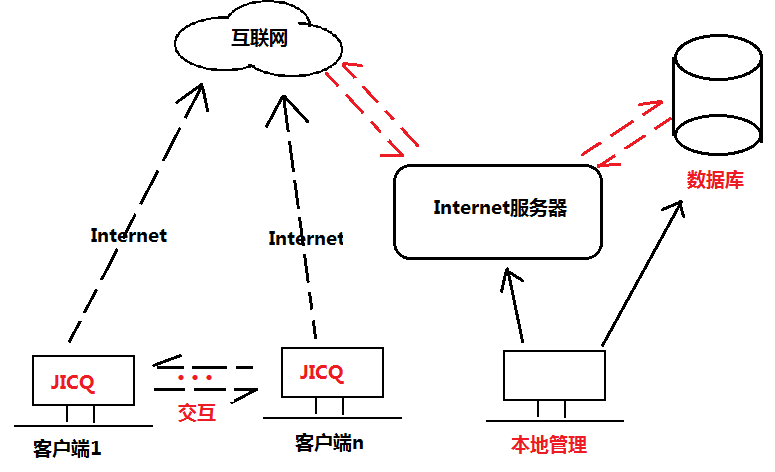


图3.1 JICQ工作原理和部署图

### 3.1.1 软件简介和工作原理

#### 1、软件简介

Java即时通信系统的服务器端包括JICQ服务器程序（其监听9001端口）和数据库服务器（本系统采用MS SQL Server2008）；客户端为JICQ即时通信工具软件和网页服务软件。

Java即时通信系统（JICQ）是以Java语言为实现的网络即时通信工具，包括服务器端和客户端，可以在企业内部网络和互联网上运行。使用JICQ可以和好友进行交流，信息即时发送，即时回复，传输文件等功能。

本系统充分利用了Java语言的平台无关性等众多优点，主要采用了Socket网络编程接口、多线程、输入输出流、Swing界面编程技术，为我们构建一个跨平台的真正畅通的网络信息交换工具 。

#### 工作原理简述

本聊天软件采用Internet Socket通信，实现两个客户端用户的聊天基本功能，包括一对一，一对多，群聊、好友列表、上线提示等。

客户端采用JavaSwing及Socket技术实现，服务器端采用JavaEE和JavaSocket实现，客户端连上Internet后向指定好友（制定IP）发送消息，服务器端搜到消息后按照客户端的需求将消息转发给相应的用户，另一用户能接收到来自服务器的数据。

服务器端使用JDBC连接数据库，并操作数据库。

本地管理员可以在后台看到数据流向并且能对数据库进行管理。

## 3.2 JICQ基本用例图、交互图和序列图

### 3.2.1 JICQ用例图

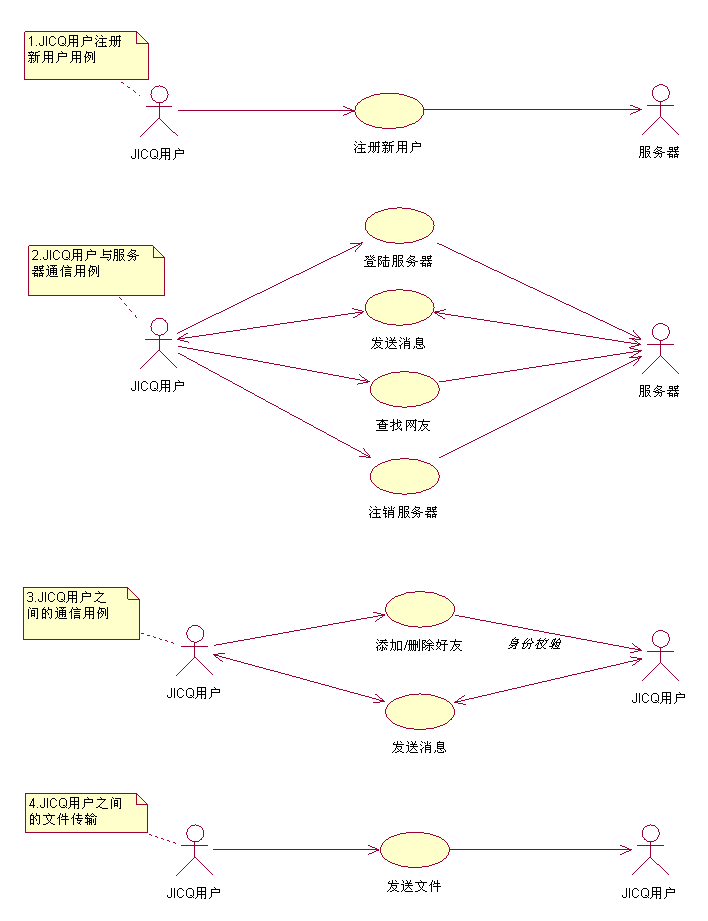


图3.2.1 用户用例图

### 3.2.2 用户注册活动图

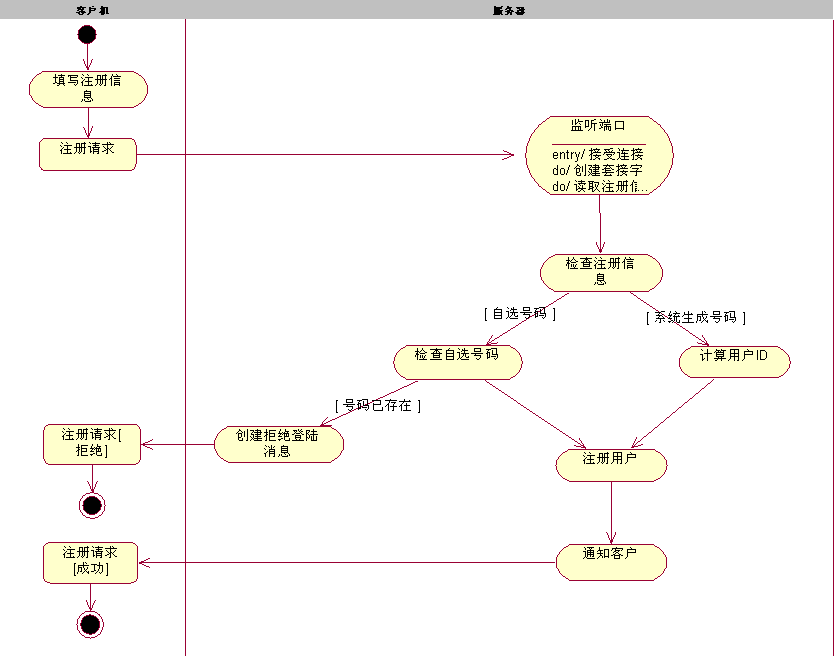


图3.2.2 用户注册活动图

### 3.2.3 用户登陆活动图

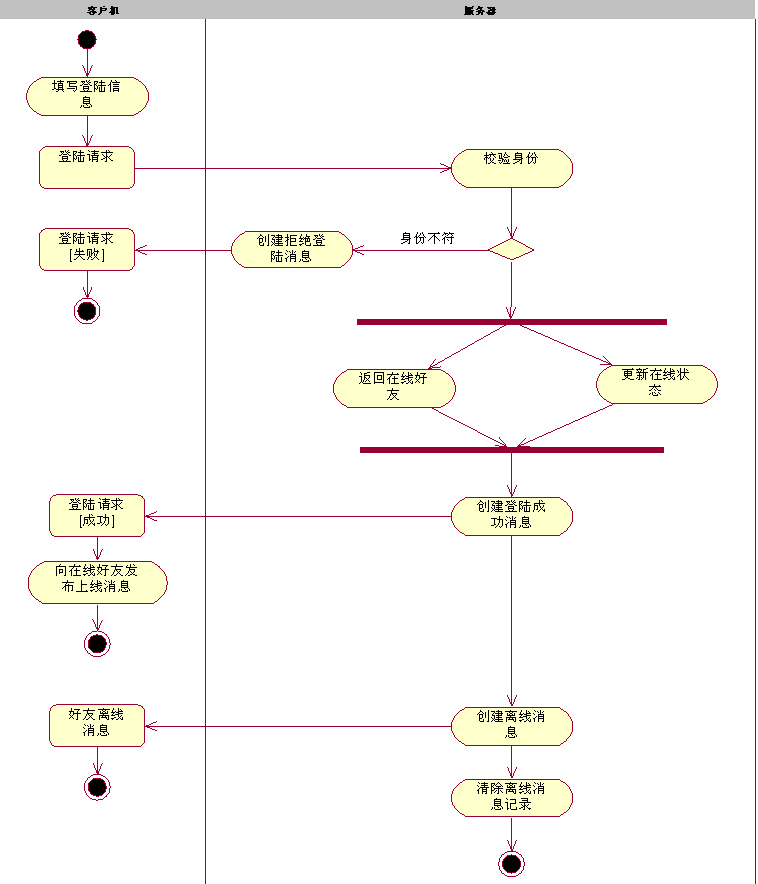


图3.2.3 用户登陆活动图

### 3.2.4 客户机之间通信活动图

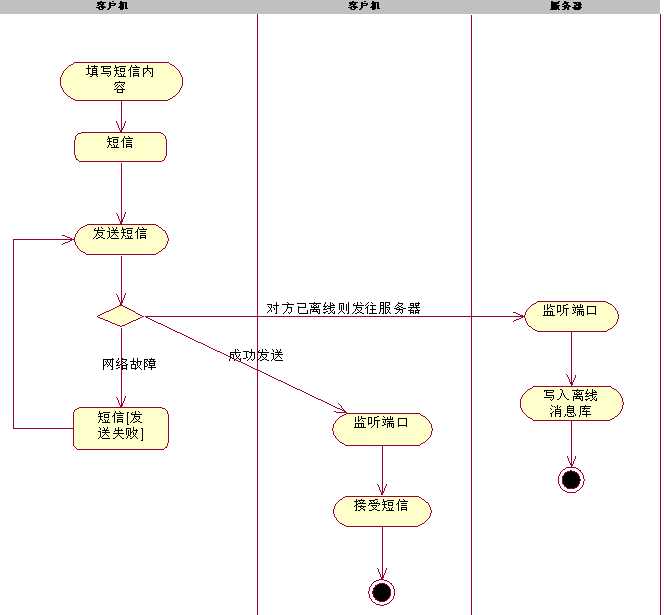


图3.2.4 用户登陆序列图

### 3.2.5 添加好友活动图

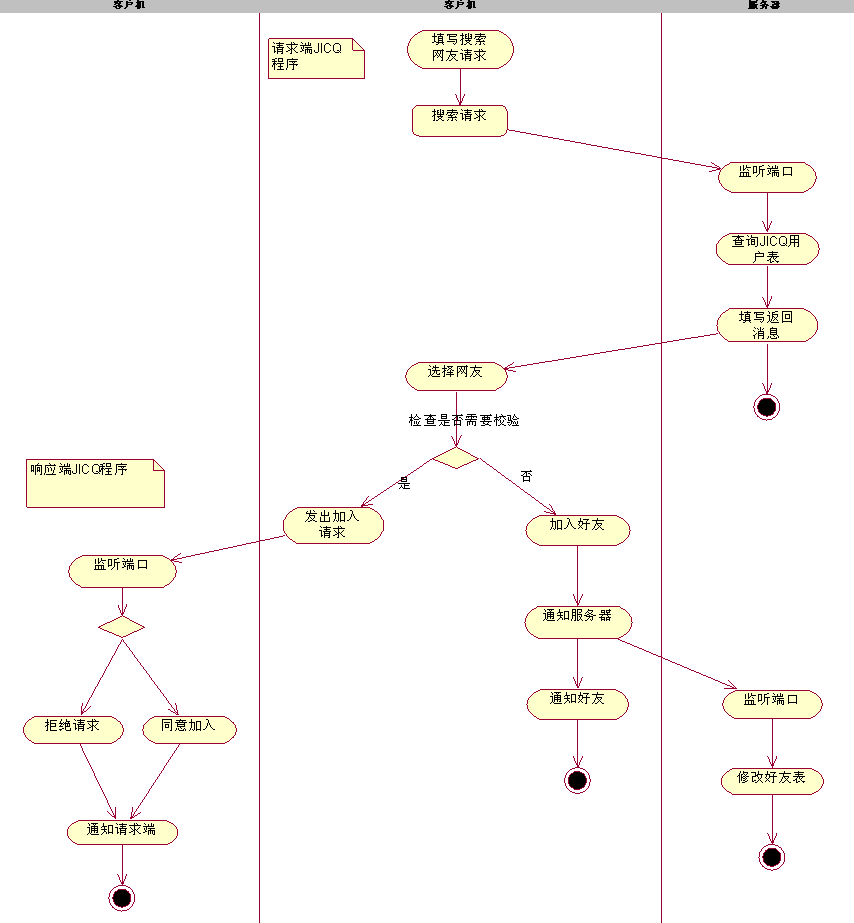


图3.2.5 添加好友活动图

## 3.2.6 删除好友活动图

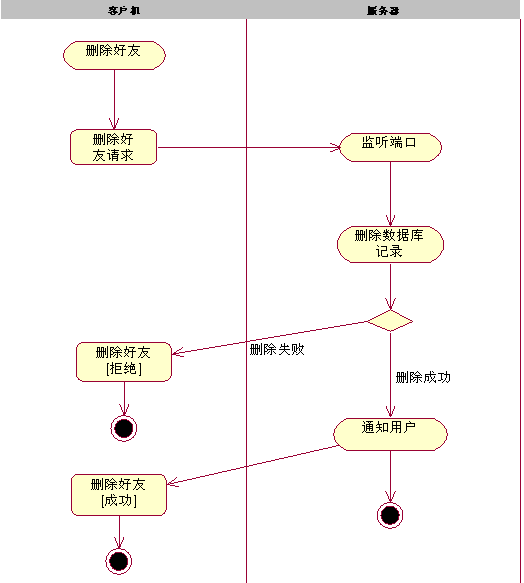


图3.2.6 删除好友活动图

## 3.3 数据库模型

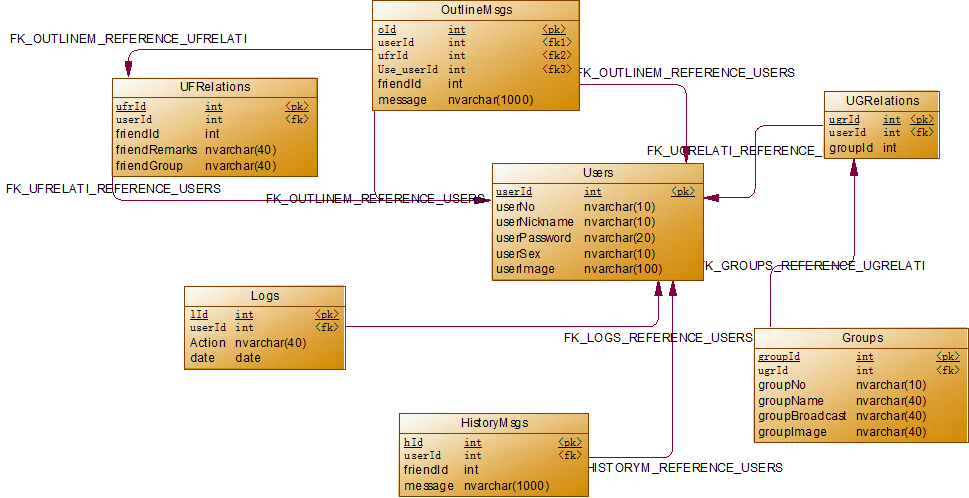


图3.3 数据库模型设计

### 3.3.1 JICQ存储设计

客户端JICQ除了必备的程序文件之外，还有相关的数据文件，以辅助程序运行。主要包括用户信息，好友信息，头像文件，声音文件等信息。

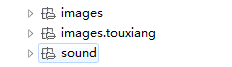


图3.2.1 存储结构设计

## 3.4 系统类包

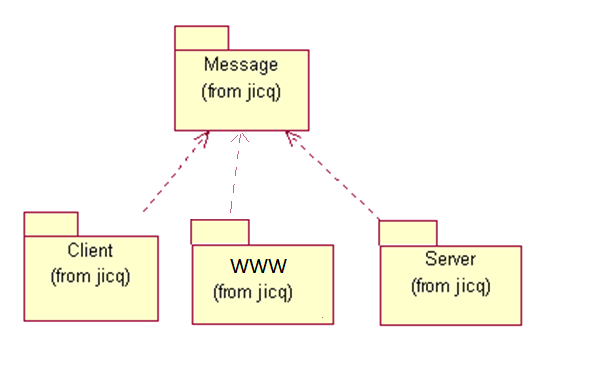


图3.4 系统类包

Message包为JICQ系统的消息类包，其中包含有JICQ系统定义的短消息类型，为服务器与客户端所共用；Server包为系统的服务器端程序类包，运行在服务器端机器；Client包为系统的客户端程序类包，运行在客户端机器。 www为网页服务类包，为系统提供网页在线注册的功能。

## 3.5 JICQ线程控制

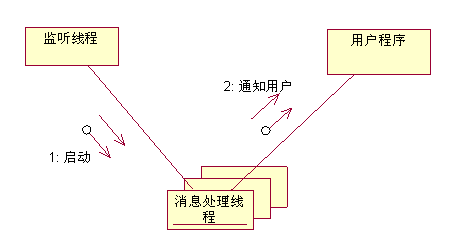


图3.5 JICQ线程控制

客户端JICQ主要由监听线程、消息处理线程控制。首先由监听线程监听消息，当有消息来到时，启动消息处理线程处理消息，最后由消息线程将消息通知用户边界对象。

另外，客户端还有一个能与服务器保持联络的线程，每两分钟向服务器通知一次，表明该客户端仍然在线。

## 3.6 网页客户端设计

使用HTML5和CSS3做前端网页的基本显示，并使用JavaScript和JQuery库对网页进行全局的优化，提供良好的用户体验。



图3.5 网页客户端设计

## 3.7 工程架构设计

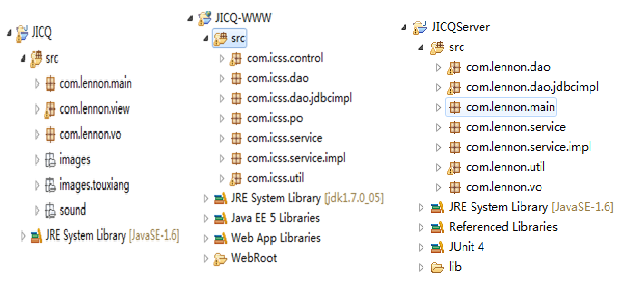


图3.6 JICQ工程架构图

### 3.7.1 JavaEE MVC框架简介

mvc是一种源于桌面程序的架构模式，它的基本思想是把程序界面和业务逻辑分开，这样便于软件的后期维护，同时也方便开发时期分工及管理，mvc有很多有点所以现在已经被广泛的应用与web开发中。

三层架构分为**表示层，逻辑层，持久层**；其中表示层是属于web方面的开发，对应与mvc的视图和控制器，控制器根据需要调用业务逻辑和视图层显示给用户；逻辑层和持久层是为了程序的可移植性我们把mvc中的模型层分为专门用于计算的逻辑层和专门访问数据的持久层（包括访问数据库、访问xml、以及访问其他可以永久保存数据的文件等），业务逻辑层主要是javabean实现；而持久层最常见的就是dao，封装了数据库的所有操作。其实三层架构从结构来说是纵向的分层，上层依赖于下层，而下层不依赖于上层，即单项依赖。

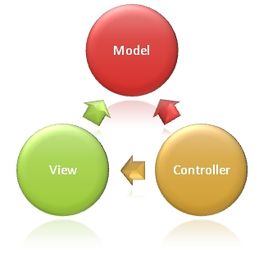


图3.7.1 MVC框架原理图

### 3.7.2 项目用用MVC框架部署

1. 前端：前端显示界面，利用jsp页面显示
2. 后端：后端控制程序，分为四个层，domain层、dao层、service层、servlet层
3. damain层：存放数据库中原始对象
4. dao层：实现对数据库的增、删、改、查操作
5. service层：数据的持久化层
6. servlet层：用于前端和后端的接口

# 第四节 详细设计

## 4.3 数据库详细设计

### 4.3.1 数据库SQL代码：

|  |
| --- |
| SQLServer2008 数据库SQL代码 |
| USE [iCoze]  GO  /\*\*\*\*\*\* Object: Table [dbo].[Users] Script Date: 01/16/2014 14:34:00 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [dbo].[Users](  [userId] [int] IDENTITY(1,1) NOT NULL,  [userNo] [nvarchar](10) NULL,  [userNickname] [nvarchar](10) NULL,  [userPassword] [nvarchar](20) NULL,  [userSex] [nvarchar](10) NULL,  [userImage] [nvarchar](100) NULL,  PRIMARY KEY CLUSTERED  (  [userId] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO  /\*\*\*\*\*\* Object: Table [dbo].[UGRelations] Script Date: 01/16/2014 14:34:00 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [dbo].[UGRelations](  [ugrId] [int] IDENTITY(1,1) NOT NULL,  [userId] [int] NULL,  [groupId] [int] NULL,  PRIMARY KEY CLUSTERED  (  [ugrId] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO  /\*\*\*\*\*\* Object: Table [dbo].[UFRelations] Script Date: 01/16/2014 14:34:00 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [dbo].[UFRelations](  [ufrId] [int] IDENTITY(1,1) NOT NULL,  [userId] [int] NULL,  [friendId] [int] NULL,  [friendRemarks] [nvarchar](40) NULL,  [friendGroup] [nvarchar](40) NULL,  PRIMARY KEY CLUSTERED  (  [ufrId] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO  /\*\*\*\*\*\* Object: Table [dbo].[OutlineMsgs] Script Date: 01/16/2014 14:34:00 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [dbo].[OutlineMsgs](  [oId] [int] IDENTITY(1,1) NOT NULL,  [userId] [int] NULL,  [friendId] [int] NULL,  [message] [nvarchar](1000) NULL,  PRIMARY KEY CLUSTERED  (  [oId] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO  /\*\*\*\*\*\* Object: Table [dbo].[Logs] Script Date: 01/16/2014 14:34:00 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [dbo].[Logs](  [lId] [int] IDENTITY(1,1) NOT NULL,  [userId] [int] NULL,  [Action] [nvarchar](40) NULL,  [date] [date] NULL,  PRIMARY KEY CLUSTERED  (  [lId] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO  /\*\*\*\*\*\* Object: Table [dbo].[HistoryMsgs] Script Date: 01/16/2014 14:34:00 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [dbo].[HistoryMsgs](  [hId] [int] IDENTITY(1,1) NOT NULL,  [userId] [int] NULL,  [friendId] [int] NULL,  [message] [nvarchar](1000) NULL,  PRIMARY KEY CLUSTERED  (  [hId] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO  /\*\*\*\*\*\* Object: Table [dbo].[Groups] Script Date: 01/16/2014 14:34:00 \*\*\*\*\*\*/  SET ANSI\_NULLS ON  GO  SET QUOTED\_IDENTIFIER ON  GO  CREATE TABLE [dbo].[Groups](  [groupId] [int] IDENTITY(1,1) NOT NULL,  [groupNo] [nvarchar](10) NULL,  [groupName] [nvarchar](40) NULL,  [groupBroadcast] [nvarchar](40) NULL,  [groupImage] [nvarchar](40) NULL,  PRIMARY KEY CLUSTERED  (  [groupId] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO |

## 4.2 Java 后端控制实现

### 4.2.1 服务器端架构实施文件

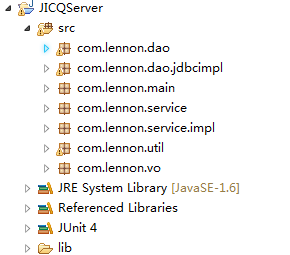


图4.5.1 架构实施文件

### 4.2.2 Socket核心代码

|  |
| --- |
| Com.lennon.util.TCPServer .java --- TCP 服务类 |
| package com.lennon.util;  import java.io.IOException;  import java.net.ServerSocket;  import java.net.Socket;  import java.util.ArrayList;  import java.util.HashSet;  import java.util.List;  import java.util.Set;  import java.util.concurrent.ExecutorService;  import java.util.concurrent.Executors;  public class TCPServer {  private final static int PORT = 9001;  private final static int MAX\_THREAD = 100;  private ServerSocket server;  private Socket socket;  public static Set<Socket> clients = new HashSet<Socket>();  public static boolean flag = false;  public static List<String> message = new ArrayList<String>();  private ServerThread serverThread = null;  public TCPServer() {  ExecutorService threadPool = Executors.newFixedThreadPool(MAX\_THREAD);  while (true) {  try {  server = new ServerSocket(PORT);  System.out.println("...Waiting...");  } catch (IOException e) {  }  try {  socket = server.accept();  System.out.println("上线客户端IP：" + socket.getInetAddress());  clients.add(socket);  serverThread = new ServerThread(socket);  threadPool.execute(serverThread);    } catch (IOException e) {  }  //  // if (socket != null) {  // new ServerThread(socket).start();// 为每个客户端启动一个线程  // }  }  }  }  public void setUid(int uid) {  this.uid = uid;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getUrl() {  return url;  }  public void setUrl(String url) {  this.url = url;  }  public String getRolename() {  return rolename;  }  public void setRolename(String rolename) {  this.rolename = rolename;  }  @Override  public String toString() {  return "User [uid=" + uid + ", username=" + username + ", password="  + password + ", email=" + email + ", url=" + url  + ", rolename=" + rolename + "]";  }  } |

|  |
| --- |
| Com.lennon.util.ServerThread.java --- 线程类 |
| package com.lennon.util;  import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStream;  import java.io.InputStreamReader;  import java.io.OutputStream;  import java.io.PrintWriter;  import java.net.Socket;  import java.util.HashMap;  import java.util.Iterator;  import java.util.List;  import java.util.Map;  import com.lennon.service.FriendService;  import com.lennon.service.UserService;  import com.lennon.service.impl.FriendServiceImpl;  import com.lennon.service.impl.UserServiceImpl;  import com.lennon.vo.Friend;  import com.lennon.vo.User;  public class ServerThread implements Runnable {  public static Map<String, String> toFriendIP = new HashMap<String, String>();  Socket socket;  int port;  UserService userService = new UserServiceImpl();  FriendService friendService = new FriendServiceImpl();  String userNo;  String userPassword;  String str = null;  InputStream input;  InputStreamReader isReader = null;  BufferedReader reader = null;  OutputStream socketOut;  PrintWriter pw = null;  ServerThread(Socket t) {  socket = t;  try {  input = socket.getInputStream();  isReader = new InputStreamReader(input);  reader = new BufferedReader(isReader);  socketOut = socket.getOutputStream();  pw = new PrintWriter(socketOut, true);  } catch (IOException e) {  }  }  public void run() {  try {  while ((str = reader.readLine()) != null) {  System.out.println("str=" + str);  if (str.equals("Register")) {  // 用户注册  String userNickname = reader.readLine();  String userPassword = reader.readLine();  String userSex = reader.readLine();  int n = userService.addUserByReg(userNickname, userPassword, userSex);  pw.println(n);  }  if (str.equals("Login")) {  // 用户登录  userNo = reader.readLine();  userPassword = reader.readLine();  if (!userService.isValidateUser(userNo, userPassword)) {  pw.println("0");// 登录失败  } else {// 登录成功  pw.println("1");  toFriendIP.put(userNo, socket.getInetAddress().toString());  }  }  if (str.equals("iCoze")) {  /\*\*  \* 显示好友列表  \*/  userNo = reader.readLine();  User user = userService.findUserByUserNo(userNo);  String userMessage = "";  userMessage += user.getUserId() + ":" + user.getUserNo() + ":" + user.getUserNickname() + ":" + user.getUserSex() + ":" + user.getUserImage() + ":" + user.getUserStat();  pw.println(userMessage);// 回传登录用户信息  List<Friend> friends = friendService.findAllFriendsByUserId(user.getUserId());  Friend friend = new Friend();  String sum = "";  Iterator<Friend> iterator = friends.iterator();  while (iterator.hasNext()) {  friend = iterator.next();  String friendStat = (toFriendIP.get(friend.getFriendNo()) == null) ? "[离线]" : "[在线]";  String s = friend.getFriendId() + ":" + friend.getFriendNo() + ":" + friend.getFriendNickname() + ":" + friend.getFriendSex() + ":" + friend.getFriendImage() + ":"  + friendStat + ";";  sum += s;  }  pw.println(sum);// 返回当前登录用户的好友列表  while ((str = reader.readLine()) != null) {  System.out.println("str=" + str);  if (str.equals("Chat")) {  String friendNo = reader.readLine();// 获取聊天对象  String friendIP = toFriendIP.get(friendNo);  if (friendIP == null) {  // 如果好友不在线，返回本人IP  pw.println("OffLine");  } else {  // 如果好友在线，返回好友IP  pw.println(friendIP);  System.out.println("好友IP是：" + friendIP);  }  }  if (str.equals("Show")) {  System.out.println("------------");  userNo = reader.readLine();  System.out.println(userNo);  User user1 = userService.findUserByUserNo(userNo);  System.out.println(user1);  if (user1 != null) {  pw.println(user1.getUserNickname());  pw.println(user1.getUserNo());  // pw.println(user.getUserSex());  // pw.println(user.getUserImage());  }  }  if (str.equals("findUser")) {  String userNo1;  userNo1 = reader.readLine(); // userNo1 要查找的好友的QQ号  System.out.println(userNo1);  user = new User();  user = userService.findUserByUserNo(userNo1);  if (user != null) {  System.out.println("Success");  pw.println("查找成功");  pw.println(user.getUserNickname());  pw.println(user.getUserNo());  pw.println(user.getUserSex());  pw.println(user.getUserImage());  // 弹出新窗口  // pw.println(user.getUserImage());  str = reader.readLine(); // addFriend  // String friendNo = reader.readLine();  System.out.println("------>" + str);  // System.out.println(friendNo);  System.out.println(userNo);  int friendId = userService.findUserByUserNo(reader.readLine()).getUserId();  int userID = userService.findUserByUserNo(userNo).getUserId();  System.out.println("friendId=" + friendId + ", userId = " + userID);  if (str.equals("addFriend")) {  int n = friendService.addFriendByFriendId(userID, friendId, "");  System.out.println("返回值：" + n);  if (n < 0) {  pw.println("friendHasInList");  } else {  pw.println("addFriendSuccess");  }  }  } else {  pw.println("查找失败");  }  }  if (str.equals("findGroup")) {  userNo = reader.readLine();  System.out.println(userNo);  }  }  }  }  } catch (IOException e) {  e.printStackTrace();  return;  }  }  } |

|  |
| --- |
| LoginService.java -- Service 层数据持久化 |
| public class LoginService {  public User checLogin(String username, String password) {  User user = null;  try {  user = UserDAO.*getUser*(username, password);  } catch (SQLException e) {  e.printStackTrace();  }  return user;  }  } |

|  |
| --- |
| LoginServlet.java -- Web层实现前端和后端的桥梁 |
| public class LoginServlet extends HttpServlet {  public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  String username = request.getParameter("username");  String password = request.getParameter("password");  LoginService service = new LoginService();  User user = service.checLogin(username, password);  if (user == null) {  ErrorMsg msg = new ErrorMsg(ErrorMsg.*NAMEPASSWORD*);  request.getSession().setAttribute("msg", msg);  response.sendRedirect(getServletContext().getContextPath()  + "/index.jsp");  return;  }  request.getSession().setAttribute("user", user);  request.getRequestDispatcher("/showinfo").forward(request, response);  }  public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  doGet(request, response);  }  } |

## 4.6 Android客户端实现

### 4.6.1 实现关键代码

|  |
| --- |
| ManiActvity.java |
| …  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  button = (Button)findViewById(R.id.button);  editText = (EditText) findViewById(R.id.editText);  button.setOnClickListener(new OnClickListener() {  public void onClick(View v) {  Intent intent = new Intent(Intent.ACTION\_VIEW, Uri.parse(editText.getText().toString()));  startActivity(intent);  }  });  }  … |

# 第五节 实验结果分析

## 5.1 实验结果展示

### 5.1.1 Web用户注册功能

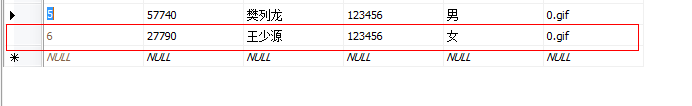
#### 1.用户注册首页展示



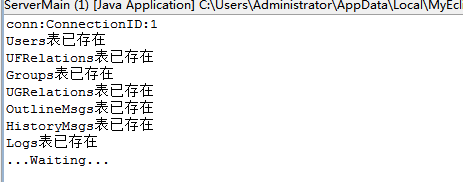
#### 2.用户注册成功并返回账号



#### 3.数据库查看



### 5.1.1 服务器后端

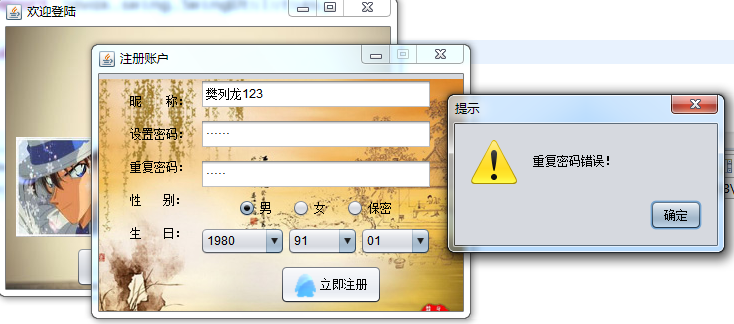


### 5.1.2 客户端

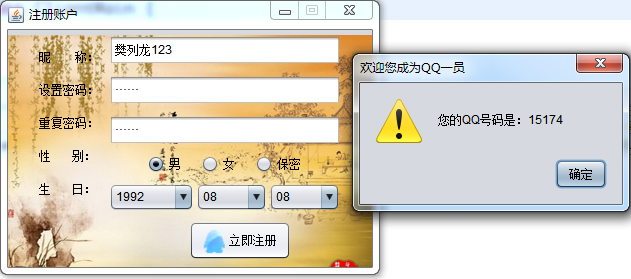
#### 1、登陆主界面



#### 2.注册功能错误处理



#### 3.用户注册成功返回QQ号



#### 7.登陆刚注册的号



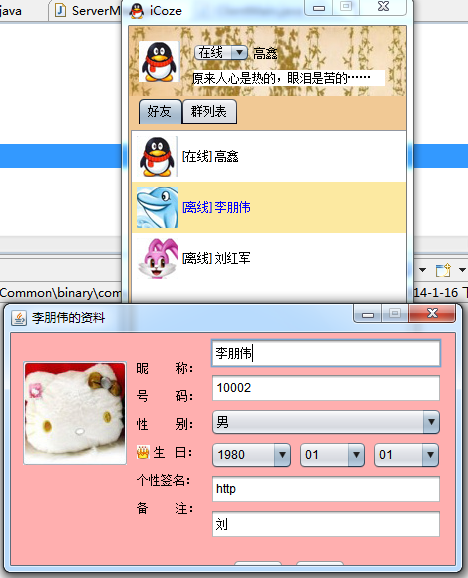
#### 8.查找好友



#### 9.添加好友



#### 10.查看好友资料



#### 11.发起聊天



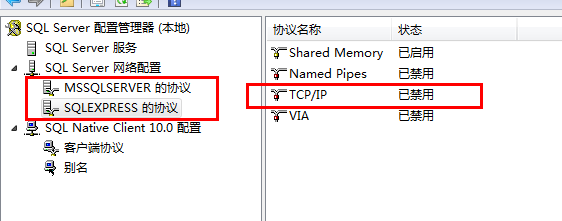
# 第六节 错误调试

## 6.1 数据库连接错误

### 6.1.1 解决方案



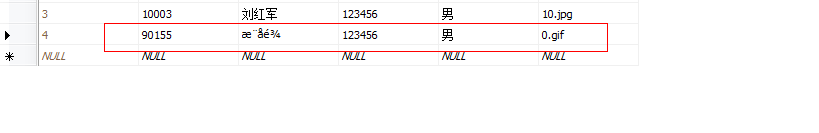
发现数据库TCPIP没开



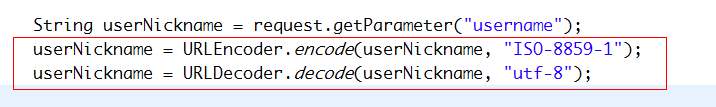
打开启用后正常

## 6.2 POST乱码解决方式

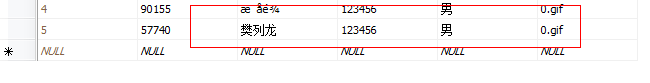
### 6.2.1 错误表现



### 6.2.2解决方案

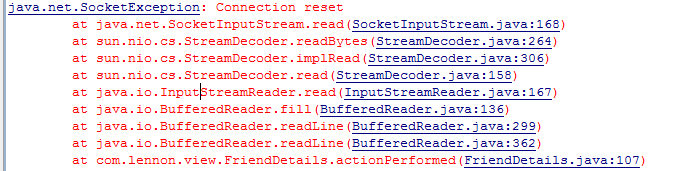


加上上面的代码后正常

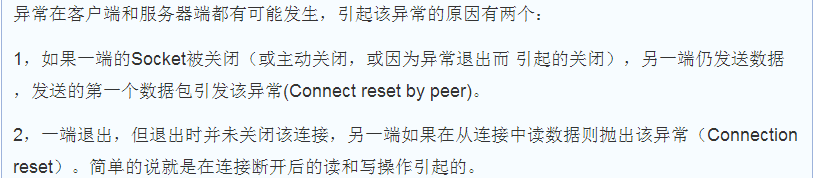


## 6.3 Socket连接错误

### 6.3.1 错误表现



### 6.3.2产生错误的原因



### 6.3.3 解决方式

优化Socket连接，并且在数据发送接收完毕后关闭流。

# 

# 第八节 实验总结

## 8.1 对于计算机网络方面的总结：

（1）通过学习，了解到了计算机在通信领域的应用意义，同时也认识到学习计算机技术能为我们以后在通信行业的发展打下一定的基础；

（2）通过课程设计对Socket编程和通信原理等专业知识进行了复习和巩固，从而对软件开发和计算机网络应用过程有了更清晰的认识；

（3）在调试之前应理清思路，对实验系统的工作原理和过程有一个全面准确的地把握，了解每一个工作部件的正常工作情况，这样才能有条不紊地进行调试；

（4）调试的过程中应仔细观察实际结果同理论结果之间的差异，便于准确地查找到错误点；

（5）当实验结果与理论情况不符实，应一级一级向前检查，直到找到问题所在。

## 8.2 对于Java方面的总结

在课程设计过程中遇到盲点时，应虚心地向老师同学请教，这样可以提高知识点掌握的速率。虚心求教在程序设计过程中非常重要，有时候由于自身思维模式的限制难以找到软件设计中的错误和不足，相互检查程序可以提高找到错误和缺陷准确率。

在实验过程中，体会到一个方案从无到有的过程。困难是有的，但是想法和对策也总会有的，一旦思路理清楚了，抓住了问题的关键，找到突破口后就能做到对症下药，直到问题解决。

# 第九节 参考资料

[1]Andrew S.Tanenbaum著 潘爱民,徐明伟译.计算机网络(第四版).清华大学出 版社.2006.03.

[2]Bruce Eckel Chuck Allison, 陈昊鹏 译. Java 高级编程(第四版). 机械工业出版社 社.2007.06

# 附件：

