JAVA 聊天室

——运用 socket 实作

B0529002

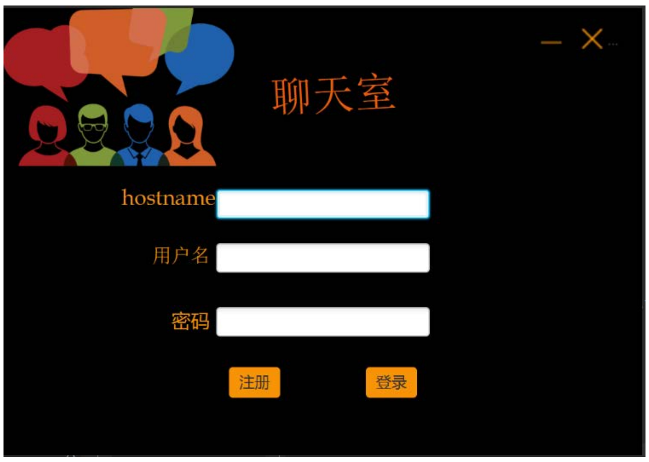
钱泓瑞

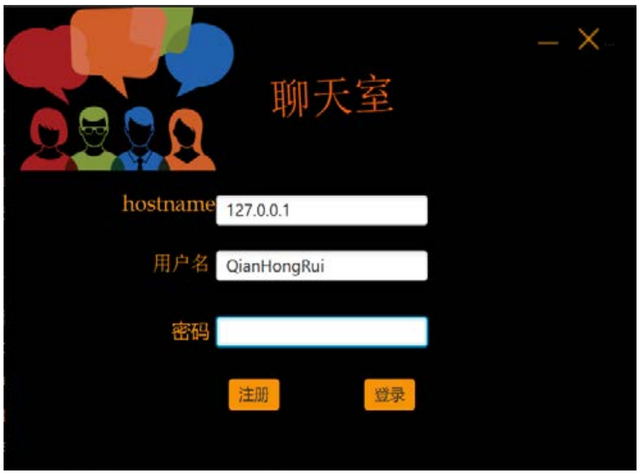
2018/12/10

一、操作说明

1. 登入&注册界面

众所周知，使用者第一次进入时要进行注册，在注册完成之后才能进行登录操作，进入聊天室。

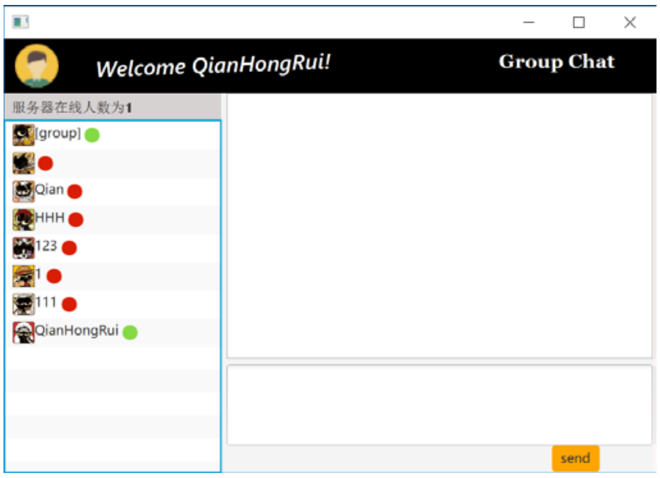


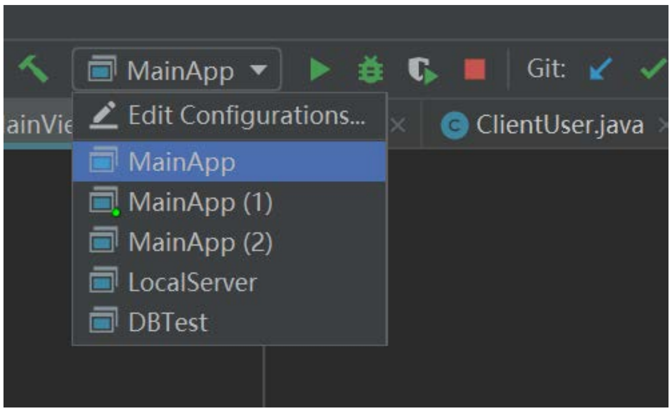


hostname 127.0.0.1
用户名 QianHongRui

2. 在线使用者列表

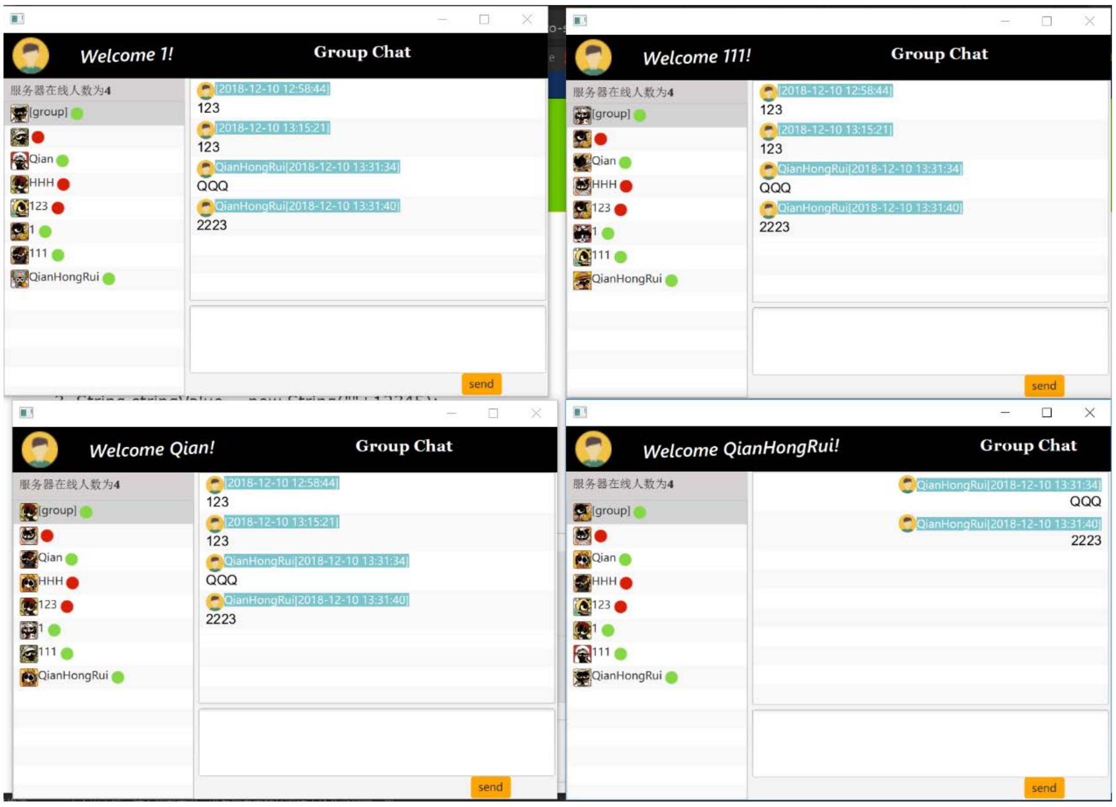
本软体采用 Java 进行撰写，所以运用了 intelliJ 作为开发工具，右上为 APP 多开工具。





3. 多人聊天室——Group

可以提供多人同时进行聊天的选择视窗，每条发言的发言人以及时间都在上面明确显示，以方便使用者辨识。



Welcome 1! Group Chat

服务器在线人数为4

[group]
Qian
HHH
123
1
111
QianHongRui

```
[2018-12-10 12:58:44]
123
[2018-12-10 13:15:21]
123
QianHongRui[2018-12-10 13:31:34]
QQQ
QianHongRui[2018-12-10 13:31:40]
2223
```

Welcome 111! Group Chat

服务器在线人数为4

[group]
Qian
HHH
123
1
111
QianHongRui

```
[2018-12-10 12:58:44]
123
[2018-12-10 13:15:21]
123
QianHongRui[2018-12-10 13:31:34]
QQQ
QianHongRui[2018-12-10 13:31:40]
2223
```

Welcome Qian! Group Chat

服务器在线人数为4

[group]
Qian
HHH
123
1
111
QianHongRui

```
[2018-12-10 12:58:44]
123
[2018-12-10 13:15:21]
123
QianHongRui[2018-12-10 13:31:34]
QQQ
QianHongRui[2018-12-10 13:31:40]
2223
```

Welcome QianHongRui! Group Chat

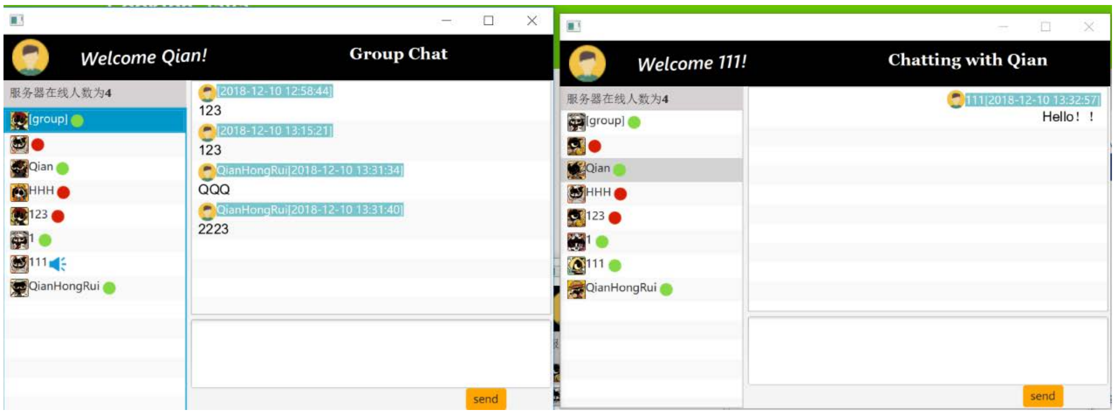
服务器在线人数为4

[group]
Qian
HHH
123
1
111
QianHongRui

```
QianHongRui[2018-12-10 13:31:34]
QQQ
QianHongRui[2018-12-10 13:31:40]
2223
```

4. 单人 P2P 进行私讯

私讯效果类似群聊时的场景。



Welcome Qian! Group Chat

服务器在线人数为4

[group]
Qian
HHH
123
1
111
QianHongRui

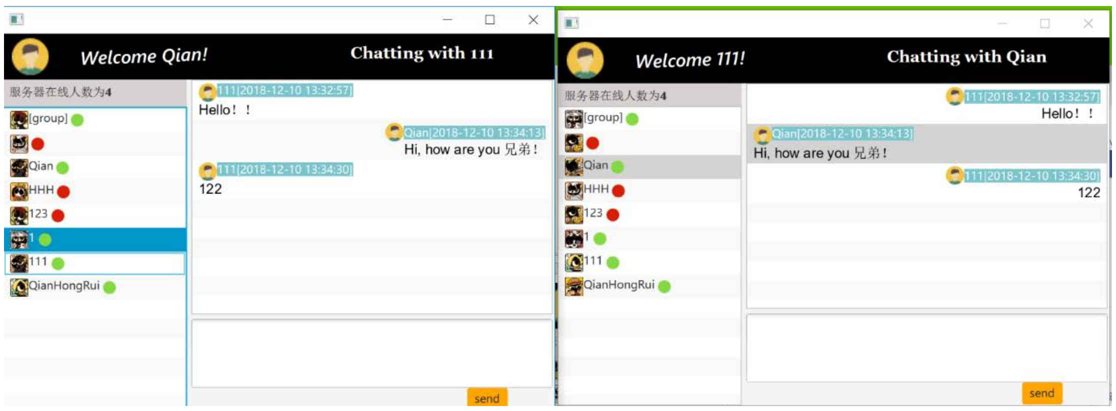
```
[2018-12-10 12:58:44]
123
[2018-12-10 13:15:21]
123
QianHongRui[2018-12-10 13:31:34]
QQQ
QianHongRui[2018-12-10 13:31:40]
2223
```

Welcome 111! Chatting with Qian

服务器在线人数为4

[group]
Qian
HHH
123
1
111
QianHongRui

```
111[2018-12-10 13:32:57]
Hello！！
```


Welcome Qian! Chatting with 111

服务器在线人数为4

[group]
Qian
HHH
123
1
111
QianHongRui

111[2018-12-10 13:32:57]
Hello！！

Qian[2018-12-10 13:34:13]
Hi, how are you 兄弟！

111[2018-12-10 13:34:30]
122

send

Welcome 111! Chatting with Qian

服务器在线人数为4

[group]
Qian
HHH
123
1
111
QianHongRui

111[2018-12-10 13:32:57]
Hello！！

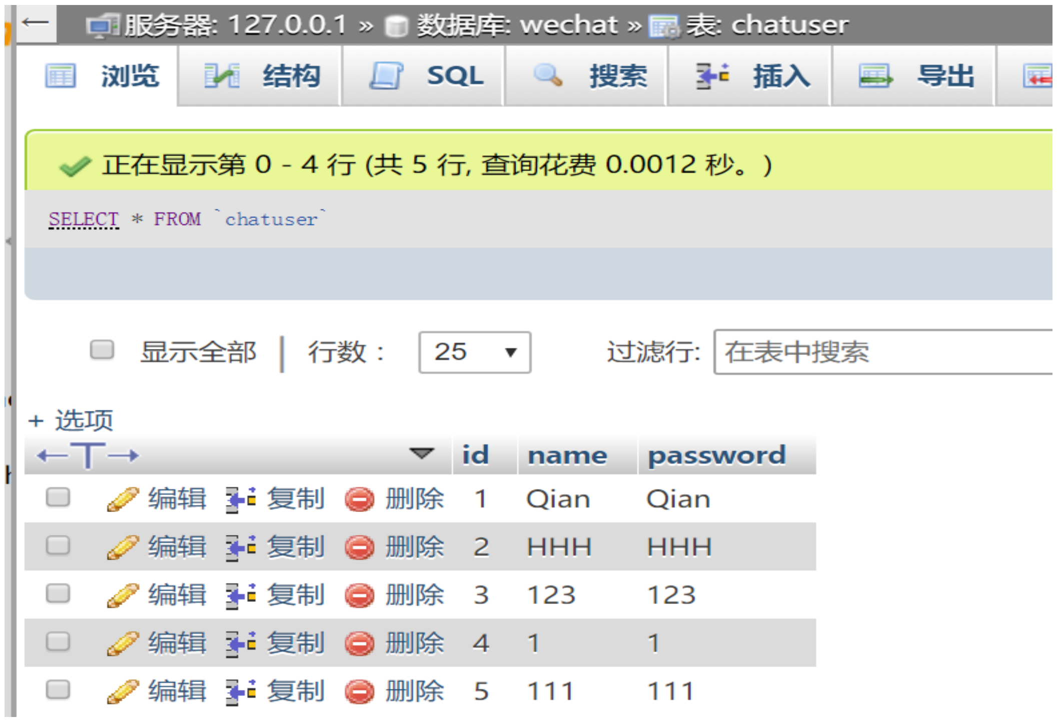
Qian[2018-12-10 13:34:13]
Hi, how are you 兄弟！

111[2018-12-10 13:34:30]
122

send

5. MySQL 资料库

存放使用者个人账号以及密码的资料库。


服务器: 127.0.0.1 » 数据库: wechat » 表: chatuser

浏览 | 结构 | SQL | 搜索 | 插入 | 导出

正在显示第 0 - 4 行 (共 5 行, 查询花费 0.0012 秒。)

```
SELECT * FROM `chatuser`
```

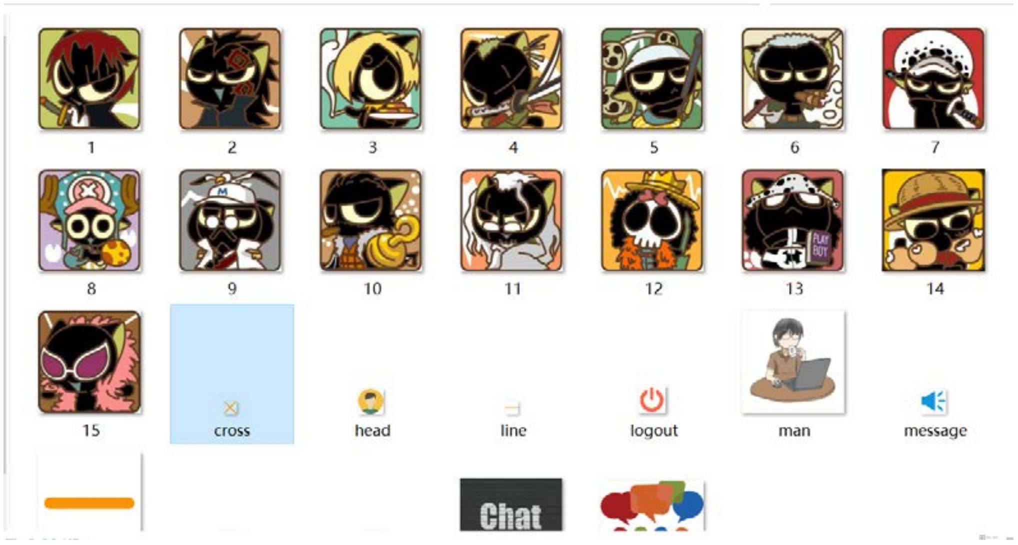
显示全部 | 行数： 25 | 过滤行: 在表中搜索

+ 选项

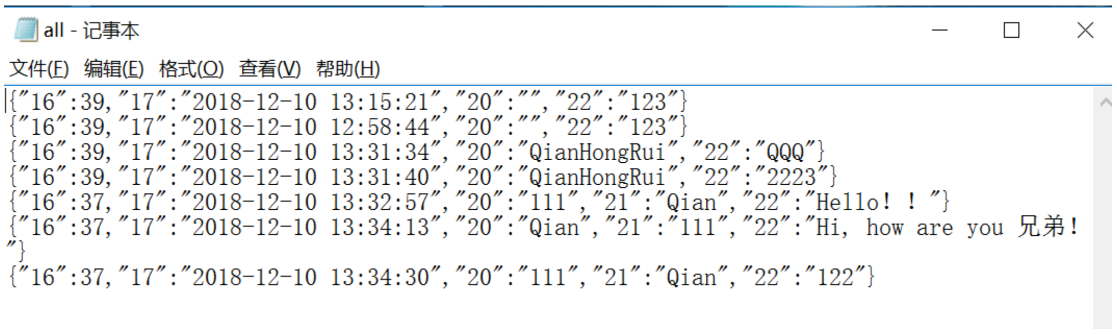
	id	name	password
编辑 复制 删除	1	Qian	Qian
编辑 复制 删除	2	HHH	HHH
编辑 复制 删除	3	123	123
编辑 复制 删除	4	1	1
编辑 复制 删除	5	111	111

6. 大头贴以及历史聊天记录

大头贴让使用者传送图片或者是 URL，使得头像得以改变。



把使用者的聊天记录存放在 TXT 中，想要的时候直接读取 TXT 档案即可

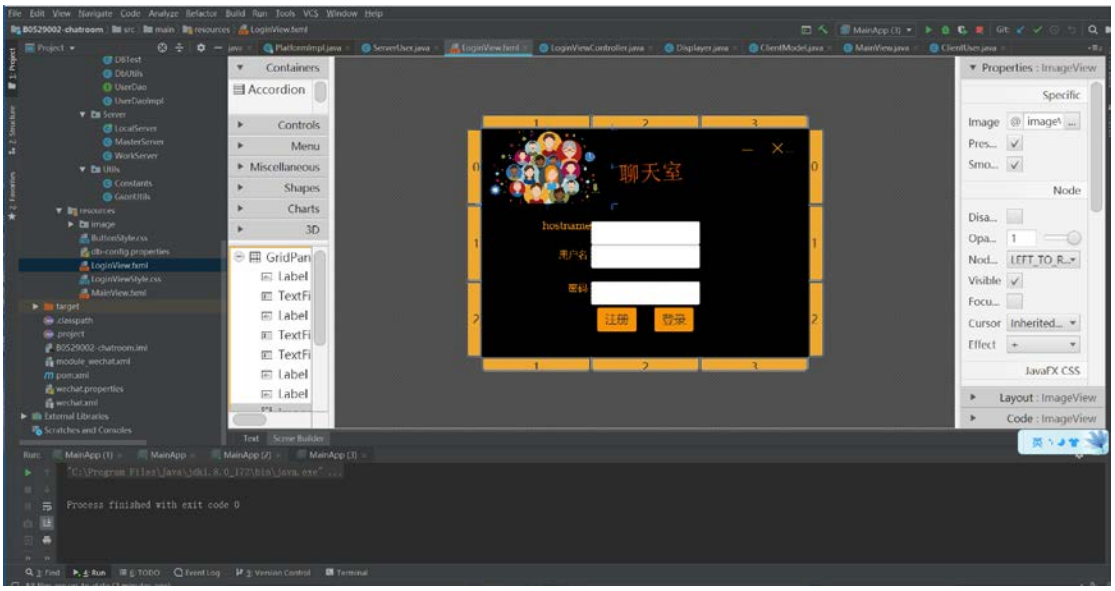


```
{"16":39,"17":"2018-12-10 13:15:21","20":"","22":"123"}
{"16":39,"17":"2018-12-10 12:58:44","20":"","22":"123"}
{"16":39,"17":"2018-12-10 13:31:34","20":"QianHongRui","22":"QQQ"}
{"16":39,"17":"2018-12-10 13:31:40","20":"QianHongRui","22":"2223"}
{"16":37,"17":"2018-12-10 13:32:57","20":"111","21":"Qian","22":"Hello！！"}
{"16":37,"17":"2018-12-10 13:34:13","20":"Qian","21":"111","22":"Hi, how are you 兄弟！
"}
{"16":37,"17":"2018-12-10 13:34:30","20":"111","21":"Qian","22":"122"}
```

二、程式说明

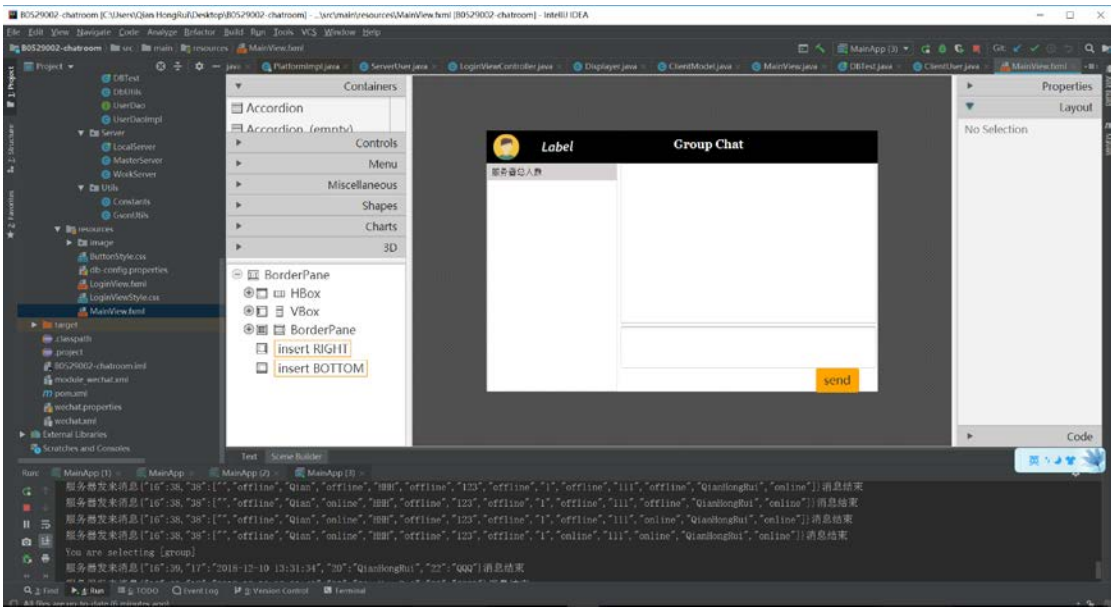
1. 界面：

运用IntelliJ的fxml进行UI界面的构筑，效果甚佳，可视化构图方便快捷。



```
"C:\Program Files\Java\jdk1.8.0_172\bin\java.exe" ...

Process finished with exit code 0
```



```
服务器发来消息{"16":38,"38":["","offline","Qian","offline","HHH","offline","123","offline","1","offline","111","offline","QianHongRui","online"]}消息结束
服务器发来消息{"16":38,"38":["","offline","Qian","online","HHH","offline","123","offline","1","offline","111","offline","QianHongRui","online"]}消息结束
服务器发来消息{"16":38,"38":["","offline","Qian","online","HHH","offline","123","offline","1","offline","111","online","QianHongRui","online"]}消息结束
服务器发来消息{"16":38,"38":["","offline","Qian","online","HHH","offline","123","offline","1","online","111","online","QianHongRui","online"]}消息结束
You are selecting [group]
服务器发来消息{"16":39,"17":"2018-12-10 13:31:34","20":"QianHongRui","22":"QQQ"}消息结束
```

2. Client and Server

把 APP 分成两个部分, 使用者和伺服器, 伺服器主要功能是作为一个交通枢纽去连接传输使用者所传递的 messages，利用多个 Threads 进行运行, 设置看门狗, 当有使用者传送资料时产生中断，伺服器接受所传递的讯息再传送给讯息的接受者。使用者就简明易懂，把讯息，接收人等资料通过 socket 进行传送，并通过 socket 对接收其他使用者的讯息进行接收。

3. Json

运用 Json 对资料进行封包後再进行传输, 更加有效率，且易于对资料的储存。

4. MySQL

通过 JAVA 的 jdbc 与 mysql 进行连接, 把使用者的关键数据存储到资料库中。

5. I/O

历史资料的储存于回传需要 I/O 的进行工作。

三、程式

Bean.ClientUser

```
package com.B0529002.bean;

import java.io.Serializable;

public class ClientUser implements Serializable{
	private String userName;//define username as userID
	private String status;//user status outLine, inLine
	private boolean notify;

	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}
	public boolean isNotify() {
		return notify;
	}
	public void setNotify(boolean notify) {
		this.notify = notify;
	}
	public String getUserName() {
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
	}
}
```

Bean.Message

```
package com.B0529002.bean;
import java.io.Serializable;
import java.util.ArrayList;

public class Message implements Serializable{

	private String content = null;
	private String speaker = null;
	private String timer = null;
	private ArrayList<String>imageList = null;

	public ArrayList<String> getImageList() {
		return imageList;
	}
	public void setImageList(ArrayList<String> imageList) {
		this.imageList = imageList;
	}
	public String getTimer() {
		return timer;
	}
	public void setTimer(String timer) {
		this.timer = timer;
	}
	public String getSpeaker() {
		return speaker;
	}
	public void setSpeaker(String speaker) {
		this.speaker = speaker;
	}
	public String getContent() {
		return content;
	}
	public void setContent(String content) {
		this.content = content;
	}
}
```

Bean.ServerUser

```
package com.B0529002.bean;

import java.util.Queue;
import java.util.concurrent.ConcurrentLinkedQueue;

public class ServerUser {
	private String userName;// define username as userID
	private String status; //user status outLine, inLine
	public Queue<String> session; //user message queue
	public String password;
	public int id;
}

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public String getUserName() {
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}
	public ServerUser(int id,String userName,String password) {
		super();
		this.userName = userName;
		this.id = id;
		this.password = password;
		//Ensure thread concurrent security
		session = new ConcurrentLinkedQueue();
	}
	public ServerUser() {
		super();
		new ServerUser(0, null,null);
	}
	public void addMsg(String message) {
		session.offer(message);
	}
	public String getMsg() {
		if (session.isEmpty())
			return null;
		return session.poll();
	}

}
```

Chatroom.MainView

```
package com.B0529002.Client.chatroom;
import com.B0529002.Client.display.Displayer;
import com.B0529002.Client.model.ClientModel;
import com.B0529002.Client.stage.ControlledStage;
import com.B0529002.Client.stage.StageController;
import com.B0529002.bean.ClientUser;
import com.B0529002.bean.Message;
import com.google.gson.Gson;
import javafx.application.Platform;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.geometry.Pos;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.TextAlignment;
import javafx.scene.text.TextFlow;
import javafx.util.Callback;

import java.net.URL;
import java.util.HashMap;
import java.util.Random;
import java.util.ResourceBundle;

import static com.B0529002.Utils.Constants.*;
import static com.B0529002.Utils.Constants.CONTENT;
public class MainView implements ControlledStage, Initializable {

    @FXML public Button btnEmoji;
    @FXML public TextArea textSend;
    @FXML public Button btnSend;
    @FXML public ListView chatWindow;
    @FXML public ListView userGroup;
    @FXML public Label labUserName;
    @FXML public Label labChatTip;
    @FXML public Label labUserCoumter;

    private Gson gson = new Gson();
    private StageController stageController;
    private ClientModel model;
    private static MainView instance;
    private boolean pattern = GROUP; //chat model
    private String seletUser = "[group]";
    private static String thisUser;
    private ObservableList<ClientUser> uselist;
    private ObservableList<Message> chatReccder;

    public MainView() {
        super();
        instance = this;
    }
    public static MainView getInstance() {
        return instance;
    }
    @Override
    public void setStageController(StageController stageController) {
        this.stageController = stageController;
    }
    @Override
    public void initialize(URL location, ResourceBundle resources) {

        model = ClientModel.getInstance();
        uselist = model.getUserList();
        chatReccder = model.getChatRecoder();
        userGroup.setItems(uselist);
        chatWindow.setItems(chatReccder);
        thisUser = model.getThisUser();
        labUserName.setText("Welcome " + model.getThisUser() + "!");
        btnSend.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                if (pattern == GROUP) {
                    HashMap map = new HashMap();
                    map.put(COMMAND, COM_CHATALL);
                    map.put(CONTENT, textSend.getText().trim());
                    model.sentMessage(gson.toJson(map));
                } else if (pattern == SINGLE) {
                    HashMap map = new HashMap();
                    map.put(COMMAND, COM_CHATWITH);
                    map.put(RECEIVER, seletUser);
                    map.put(SPEAKER, model.getThisUser());
                    map.put(CONTENT, textSend.getText().trim());
                    model.sentMessage(gson.toJson(map));
                }
                textSend.setText("");
            }
        });


userGroup.getSelectionModel().selectedItemProperty().addListener((observable,
oldValue, newValue) -> {
```

```
                    ClientUser user = (ClientUser) newValue;
                    System.out.println("You are selecting " + user.getUserName());
                    if (user.getUserName().equals("[group]")) {
                        pattern = GROUP;
                        if (!seletUser.equals("[group]")) {
                            model.setChatUser("[group]");
                            seletUser = "[group]";
                            labChatTip.setText("Group Chat");
                        }
                    } else {
                        pattern = SINGLE;
                        if (!seletUser.equals(user.getUserName())) {
                            model.setChatUser(user.getUserName());
                            seletUser = user.getUserName();
                            labChatTip.setText("Chatting with " + seletUser);
                            // TODO: 2017/11/29
                        }
                    }
                });

                chatWindow.setCellFactory(new Callback<ListView, ListCell>() {
                    @Override
                    public ListCell call(ListView param) {
                        return new ChatCell();
                    }
                });

                userGroup.setCellFactory(new Callback<ListView, ListCell>() {
                    @Override
                    public ListCell call(ListView param) {
                        return new UserCell();
                    }
                });
            }
            public TextArea getMessageBoxTextArea() {
                return textSend;
            }
            public Label getLabUserCoumter() {
                return labUserCoumter;
            }
            public void readFile_history(ActionEvent actionEvent) {
                //ClientModel.readFile_all();
            }
            public static class UserCell extends ListCell<ClientUser> {
                @Override
                protected void updateItem(ClientUser item, boolean empty) {
                    super.updateItem(item, empty);
                    Platform.runLater(new Runnable() {
                        @Override
                        public void run() {
                            if (item != null) {
                                HBox hbox = new HBox();
                                Random random = new Random();
                                int a=random.nextInt(14)+1;
                                ImageView    imageHead    =    new    ImageView(new
Image("image/"+Integer.toString(a)+".jpg"));
                                imageHead.setFitHeight(20);
                                imageHead.setFitWidth(20);
                                ClientUser user = (ClientUser) item;
                                ImageView imageStatus;
                                if(user.getUserName().equals("[group]")){
                                    imageStatus       =       new       ImageView(new
Image("image/online.png"));
                                } else if(user.isNotify()==true){
                                    imageStatus       =       new       ImageView(new
Image("image/message.png"));
                                }else {
                                    if(user.getStatus().equals("online")){
                                        imageStatus     =     new     ImageView(new
Image("image/online.png"));
                                    }else{
                                        imageStatus     =     new     ImageView(new
Image("image/offline.png"));
                                    }
                                }
                                imageStatus.setFitWidth(20);
                                imageStatus.setFitHeight(20);
                                Label label = new Label(user.getUserName());
                                hbox.getChildren().addAll(imageHead,
label,imageStatus);
                                setGraphic(hbox);
                            } else {
                                setGraphic(null);
                            }
                        }
                    });
                }
            }
            public static class ChatCell extends ListCell<Message> {
                @Override
                protected void updateItem(Message item, boolean empty) {
                    super.updateItem(item, empty);
                    Platform.runLater(new Runnable() {
                        @Override
                        public void run() {
                            //inorder to avoid the
                            if (item != null) {
                                VBox box = new VBox();
                                HBox hbox = new HBox();
                                TextFlow        txtContent        =        new
TextFlow(Displayer.createtheTextNode(item.getContent()));
                                Label labUser = new Label(item.getSpeaker() + "[" +
item.getTimer() + "]");
                                labUser.setStyle("-fx-background-color: #7bc5cd; -
fx-text-fill: white;");
                                ImageView    image    =    new    ImageView(new
Image("image/head.png"));
                                image.setFitHeight(20);
                                image.setFitWidth(20);
                                hbox.getChildren().addAll(image, labUser);
                                if (item.getSpeaker().equals(thisUser)) {

txtContent.setTextAlignment(TextAlignment.RIGHT);
                                    hbox.setAlignment(Pos.CENTER_RIGHT);
                                    box.setAlignment(Pos.CENTER_RIGHT);
                                }
                                box.getChildren().addAll(hbox, txtContent);
                                setGraphic(box);
                            } else {
                                setGraphic(null);
                            }
                        }
                    });
                }
            }

}
```

Display.display

```
package com.B0529002.Client.display;

import java.util.LinkedList;
import java.util.Queue;


import javafx.scene.Node;
import javafx.scene.text.Font;
import javafx.scene.text.Text;

public class Displayer {

    public static Node[] createtheTextNode(String input) {

        Queue<Object> queue = new LinkedList<>();
        queue.add(input);
        Node[] nodes = new Node[queue.size()];
        int i = 0;
        while (!queue.isEmpty()) {
            Object ob = queue.poll();
            if (ob instanceof String) {
                String text = (String) ob;
                nodes[i++] = createTextNode(text);
            }
        }
        return nodes;
    }

    private static Node createTextNode(String text) {
        Text textNode = new Text(text);
        textNode.setFont(Font.font("Arial", 15));//  字体样式和大小
        return textNode;
    }


}
```

Login.LoginViewController

```
package com.B0529002.Client.Login;
import com.B0529002.Client.model.ClientModel;
import com.B0529002.Client.stage.ControlledStage;
import com.B0529002.Client.MainApp;
import com.B0529002.Client.stage.StageController;
import javafx.application.Platform;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;


import java.net.URL;
import java.util.ResourceBundle;

public class LoginViewController implements ControlledStage, Initializable {
	@FXML TextField textPassword;
	@FXML TextField txtUsername;
	@FXML TextField txtHostName;
	@FXML Button btn_login;
	@FXML ImageView imageView;
	@FXML Button btn_signIn;
	StageController myController;
	ClientModel model;

	public LoginViewController() {
		super();
	}
	public void setStageController(StageController stageController) {
		this.myController = stageController;
		model = ClientModel.getInstance();
	}
	public void initialize(URL location, ResourceBundle resources) {
	}

	public void goToMain() {
		myController.loadStage(MainApp.mainViewID,MainApp.mainViewRes);
		myController.setStage(MainApp.mainViewID,MainApp.loginViewID);
		myController.getStage(MainApp.mainViewID).setOnCloseRequest(new
EventHandler<WindowEvent>() {
			@Override
			public void handle(WindowEvent event) {
				model.disConnect();
				//myController.unloadStage(MainApp.EmojiSelectorID);
			}
		});
	}

	public void logIn() {
		StringBuffer result = new StringBuffer();
		if                                   (model.CheckLogin(txtUsername.getText(),
txtHostName.getText(),textPassword.getText(), result, 0)) {
			goToMain();
		} else {
			showError(result.toString());
		}
	}

	// 最小化窗口

	@FXML public void minBtnAction(ActionEvent event){
		Stage stage = myController.getStage(MainApp.loginViewID);
		stage.setIconified(true);
	}
	//关闭窗口，关闭程序
```

```
    @FXML public void closeBtnAction(ActionEvent event){
        Platform.exit();
        System.exit(0);
    }

    public void showError(String error) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("聊天室");
        alert.setContentText("登录失败  " + error);
        alert.show();
    }

    public void signUp(ActionEvent actionEvent) {
        StringBuffer result = new StringBuffer();
        if                          (model.CheckLogin(txtUsername.getText(),
txtHostName.getText(),textPassword.getText(), result, 1)) {
            goToMain();
        } else {
            showError(result.toString());
        }
    }
}
```

Model.ClintModel

```
package com.B0529002.Client.model;
import com.B0529002.Client.chatroom.MainView;
import com.B0529002.Utils.GsonUtils;
import com.B0529002.bean.ClientUser;
import com.B0529002.bean.Message;
import com.google.gson.Gson;
import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

import java.io.*;
import java.net.ConnectException;
import java.net.Socket;
import java.util.*;

import static com.B0529002.Utils.Constants.*;

public class ClientModel {

    static int history=0;
```

```
private BufferedReader reader;
private PrintWriter writer;
private Socket client;
private final int port = 8888;
private String IP;
private boolean isConnect = false;
//连接标志
private boolean chatChange = false;
private String chatUser = "[group]";
private String thisUser;
private Gson gson;

private LinkedHashMap<String, ArrayList<Message>> userSession;
//用户消息队列存储用
private Thread keepalive = new Thread(new KeepAliveWatchDog());
private Thread keepreceive = new Thread(new ReceiveWatchDog());

private ObservableList<ClientUser> userList;
private ObservableList<Message> chatRecoder;

private ClientModel() {
	super();
	gson = new Gson();
	ClientUser user = new ClientUser();
	user.setUserName("[group]");
	user.setStatus("");
	userSession = new LinkedHashMap<>();
	userSession.put("[group]", new ArrayList<>());
	userList = FXCollections.observableArrayList();
	chatRecoder = FXCollections.observableArrayList();
	userList.add(user);
}

private static ClientModel instance;

public static ClientModel getInstance() {
	if (instance == null) {
		synchronized (ClientModel.class) {
			if (instance == null) {
				instance = new ClientModel();
			}
		}
	}
	return instance;
}


public void setChatUser(String chatUser) {
	if (!this.chatUser.equals(chatUser))
```

```
                chatChange = true;
            this.chatUser = chatUser;
            //消除未读信息状态
            for (int i = 0; i < userList.size(); i++) {
                ClientUser user = userList.get(i);
                if (user.getUserName().equals(chatUser)) {
                    if (user.isNotify()) {
                        System.out.println("更改消息目录");
//                                                  user.setStatus(user.getStatus().substring(0,
user.getStatus().length() - 3));
                        userList.remove(i);
                        userList.add(i, user);
                        user.setNotify(false);
                    }
                    chatRecoder.clear();
                    chatRecoder.addAll(userSession.get(user.getUserName()));
                    break;
                }
            }
        }

        public ObservableList<Message> getChatRecoder() {
            return chatRecoder;
        }

        public ObservableList<ClientUser> getUserList() {
            return userList;
        }

        public String getThisUser() {
            return thisUser;
        }

        class KeepAliveWatchDog implements Runnable {
            @Override
            public void run() {
                HashMap<Integer, Integer> map = new HashMap<>();
                map.put(COMMAND, COM_KEEP);

                try {
                    System.out.println("keep alive start" + Thread.currentThread());
                    //heartbeat detection
                    //readFile_all();
                    while (isConnect) {

                        Thread.sleep(500);
//                        System.out.println("500ms keep");
                        writer.println(gson.toJson(map));
                    }
            } catch (InterruptedException e) {
                e.printStackTrace();
            } /*catch ( IOException e) {
               // e.printStackTrace();
            }*/
        }
    }

    class ReceiveWatchDog implements Runnable {
        @Override
        public void run() {
            try {
                System.out.println(" Receieve start" + Thread.currentThread());
                String message;
                while (isConnect) {
                    message = reader.readLine();
                    //System.out.println("读取服务器信息" + message);
                    handleMessage(message);
                }
            } catch (IOException e) {

            }
        }
    }

    /**
     * disconnect
     */
    public void disConnect() {
        System.out.println("disconnected");
        isConnect = false;
        keepalive.stop();
        keepreceive.stop();
        if (writer != null) {
            writer.close();
            writer = null;
        }
        if (client != null) {
            try {
                client.close();
                client = null;
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }


    private void handleMessage(String message) throws IOException {
            Map<Integer, Object> gsonMap = GsonUtils.GsonToMap(message);
            Integer        command        =        GsonUtils.Double2Integer((Double)
gsonMap.get(COMMAND));
            Message m;
            switch (command) {
                case COM_GROUP:
                    HashSet<String> recoder = new HashSet<>();
                    for (ClientUser u : userList) {
                        if (u.isNotify()) {
                            recoder.add(u.getUserName());
                        }
                    }
                    ArrayList<String>       userData       =       (ArrayList<String>)
gsonMap.get(COM_GROUP);
                    userList.remove(1, userList.size());
                    int onlineUserNum = 0;
                    for (int i = 0; i < userData.size(); i++) {
                        ClientUser user = new ClientUser();
                        user.setUserName(userData.get(i));
                        user.setStatus(userData.get(++i));
                        if (user.getStatus().equals("online"))
                            onlineUserNum++;
                        if (recoder.contains(user.getUserName())) {
                            user.setNotify(true);
                            user.setStatus(user.getStatus() + "(*)");
                        }
                        userList.add(user);
                        userSession.put(user.getUserName(), new ArrayList<>());
                    }
                    int finalOnlineUserNum = onlineUserNum;
                    Platform.runLater(new Runnable() {
                        @Override
                        public void run() {

MainView.getInstance().getLabUserCoumter().setText("服务器在线人数为" +
finalOnlineUserNum);
                        }
                    });
                    break;
                case COM_CHATALL:
                    //if(history ==1)
                        writeFile_all(message);
                    //readFile();
                    m = new Message();
                    m.setTimer((String) gsonMap.get(TIME));
                    m.setSpeaker((String) gsonMap.get(SPEAKER));
                    m.setContent((String) gsonMap.get(CONTENT));
                    if (chatUser.equals("[group]")) {
                        chatRecoder.add(m);
                    }
                    userSession.get("[group]").add(m);
                    break;
                case COM_CHATWITH:
                    //if(history==1)
                        writeFile_all(message);
                    String speaker = (String) gsonMap.get(SPEAKER);
                    String receiver = (String) gsonMap.get(RECEIVER);
                    String time = (String) gsonMap.get(TIME);
                    String content = (String) gsonMap.get(CONTENT);
                    m = new Message();
                    m.setSpeaker(speaker);
                    m.setContent(content);
                    m.setTimer(time);
                    if (thisUser.equals(receiver)) {
                        if (!chatUser.equals(speaker)) {
                            for (int i = 0; i < userList.size(); i++) {
                                if (userList.get(i).getUserName().equals(speaker))
{
                                    ClientUser user = userList.get(i);
                                    if (!user.isNotify()) {

//user.setStatus(userList.get(i).getStatus() + "(*)");
                                        user.setNotify(true);
                                    }
                                    userList.remove(i);
                                    userList.add(i, user);
                                    break;
                                }
                            }
                            System.out.println("标记未读");
                        }else{
                            chatRecoder.add(m);
                        }
                        userSession.get(speaker).add(m);
                    }else{
                        if(chatUser.equals(receiver))
                            chatRecoder.add(m);
                        userSession.get(receiver).add(m);
                    }
                    break;
                default:
                    break;
            }
            //if(history==1)
                System.out.println("服务器发来消息" + message + "消息结束");
        }

        //sent json string   message to server
```

```
	public void sentMessage(String message) {
		writer.println(message);
	}


	//检查是否正确登入
	public boolean CheckLogin(String username, String IP, String password,
StringBuffer buf, int type) {
		this.IP = IP; //bind server IP
		Map<Integer, Object> map;
		try {
			//针对多次尝试登录
			if (client == null || client.isClosed()) {
				client = new Socket(IP, port);
				reader = new BufferedReader(new
InputStreamReader(client.getInputStream()));
				writer = new PrintWriter(client.getOutputStream(), true);
			}
			map = new HashMap<>();
			if (type == 0)
				map.put(COMMAND, COM_LOGIN);
			else
				map.put(COMMAND, COM_SIGNUP);
			map.put(USERNAME, username);
			map.put(PASSWORD, password);
			writer.println(gson.toJson(map));
			String strLine = reader.readLine(); //readline 是线程阻塞的
			System.out.println(strLine);
			map = GsonUtils.GsonToMap(strLine);
			Integer result = GsonUtils.Double2Integer((Double)
map.get(COM_RESULT));
			if (result == SUCCESS) {
				isConnect = true;
				//request group
				map.clear();
				map.put(COMMAND, COM_GROUP);
				writer.println(gson.toJson(map));
				thisUser = username;
				keepalive.start();
				keepreceive.start();
				return true;
			} else {
				String description = (String) map.get(COM_DESCRIPTION);
				buf.append(description);
				return false;
			}
		} catch (ConnectException e) {
			buf.append(e.toString());
		} catch (IOException e) {
			e.printStackTrace();
			buf.append(e.toString());
		}
		return false;
	}


	public void writeFile_all(String message) throws IOException {
		//写入中文字符时解决中文乱码问题
		FileOutputStream  fos=new  FileOutputStream(new  File("C:/Users/Qian
HongRui/Desktop/all.txt"),true);
		OutputStreamWriter osw=new OutputStreamWriter(fos, "UTF-8");
		BufferedWriter    bw=new BufferedWriter(osw);

		bw.write(message);
		bw.newLine();
		//注意关闭的先后顺序，先打开的后关闭，后打开的先关闭
		bw.close();
		osw.close();
		fos.close();
	}


	public void readFile_all() throws IOException {
		//BufferedReader 是可以按行读取文件
		FileInputStream  inputStream  =  new  FileInputStream("C:/Users/Qian
HongRui/Desktop/all.txt");
		BufferedReader    bufferedReader    =    new    BufferedReader(new
InputStreamReader(inputStream));

		//System.out.println("歷史信息：/n");

		String str = null;
		while((str = bufferedReader.readLine()) != null)
		{
			handleMessage(str);
		   // System.out.println(str);
		}

		//close
		inputStream.close();
		bufferedReader.close();


		history =1;
	}

}
```

MainApp

```
package com.B0529002.Client;
import com.B0529002.Client.stage.StageController;
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.stage.StageStyle;

public class MainApp extends Application {
	public static String mainViewID = "MainView";
	public static String mainViewRes = "/MainView.fxml";

	public static String loginViewID = "LoginView";
	public static String loginViewRes = "/LoginView.fxml";


	private StageController stageController;


	@Override
	public void start(Stage primaryStage) {
		//新建一个 StageController 控制器
		stageController = new StageController();

		//将主 stage 交给控制器处理
		stageController.setPrimaryStage("primaryStage", primaryStage);

		//加载两个 stage，每个界面一个 stage
		stageController.loadStage(loginViewID,                loginViewRes,
StageStyle.UNDECORATED);

		//显示 MainView stage
		stageController.setStage(loginViewID);
	}


	public static void main(String[] args) {
		launch(args);
	}
}
```

Dao.DbTest

```
package com.B0529002.Dao;
import com.B0529002.bean.ServerUser;
import java.sql.SQLException;

public class DBTest {
	public static void main(String[] args){
		ServerUser p = new ServerUser(3,"liming","student" );
		UserDaoImpl peronDao = UserDaoImpl.getInstance();
		try {
			peronDao.add(p);
			System.out.println("success");
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
}
```

Dao.DbUtils

```
package com.B0529002.Dao;
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

public class DbUtils {
	public static String URL;//数据库连接地址
	public static String USERNAME;//用户名
	public static String PASSWORD;//密码
	//public static String DRIVER;//mysql 的驱动类
	//获取配置信息的内容
	private static Properties properties;
	private DbUtils(){}

	//使用静态块加载驱动程序
	static{
		try {
			InputStream                    input                    =
Class.forName(DbUtils.class.getName()).getResourceAsStream("/db-
config.properties");
			properties = new Properties();
			properties.load(input);
			URL = properties.getProperty("jdbc.url");
			USERNAME = properties.getProperty("jdbc.username");
			PASSWORD = properties.getProperty("jdbc.password");
			//DRIVER = properties.getProperty("jdbc.driver");
			//Class.forName(DRIVER);
			System.out.println(URL);
			System.out.println("DBUTILs success");
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
			System.out.println("DBUTILs error");
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	//定义一个获取数据库连接的方法
	public static Connection getConnection(){
		Connection conn = null;
		try {
			conn = DriverManager.getConnection(URL, USERNAME,
PASSWORD);
		} catch (SQLException e) {
			e.printStackTrace();
			System.out.println("获取连接失败");
		}
		return conn;
	}

	//关闭数据库连接
	public static void close(ResultSet rs,Statement stat,Connection conn){
		try {
			if(rs!=null)rs.close();
			if(stat!=null)stat.close();
			if(conn!=null)conn.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

}
```

Server.WorkServer

```
package com.B0529002.Server;

import com.B0529002.Dao.UserDaoImpl;
import com.B0529002.Utils.GsonUtils;
import com.B0529002.bean.ServerUser;
import com.google.gson.Gson;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.SocketException;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.*;

import static com.B0529002.Utils.Constants.*;

public class WorkServer extends Thread {
	private ServerUser workUser; //the user is connected
	private Socket socket;
	private ArrayList<ServerUser> users;
	private BufferedReader reader;
	private PrintWriter writer;
	private boolean isLogOut = false;
	private long currentTime = 0;
	private Gson gson;

	public WorkServer(Socket socket, ArrayList users) {
		super();
		gson = new Gson();
		this.socket = socket; //bind socket
		this.users = users;	//get the common user resource
	}

	@Override
	public void run() {
		//todo server's work
		try {
			currentTime = new Date().getTime();
			reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
			writer = new PrintWriter(socket.getOutputStream(), true);
			String readLine;
			while (true) {
				//heart check
				long newTime = new Date().getTime();
				if (newTime - currentTime > 2000) {
					logOut();
				} else {
					currentTime = newTime;
				}
				readLine = reader.readLine();
				if (readLine == null)
```

```
					logOut();
				handleMessage(readLine);
				sentMessageToClient();
				if (isLogOut) {
					// kill the I/O stream
					reader.close();
					writer.close();
					break;
				}
			}
		} catch (SocketException e) {
			e.printStackTrace();
			logOut();
		} catch (IOException e) {
			e.printStackTrace();
			logOut();
		}
	}


	//对传递的讯号做分析处理
	private void handleMessage(String readLine) {
		System.out.println("handle message" + readLine);
		Map<Integer, Object> gsonMap = GsonUtils.GsonToMap(readLine);
		Integer	command	=	GsonUtils.Double2Integer((Double)
gsonMap.get(COMMAND));
		HashMap map = new HashMap();
		String username, password;
		switch (command) {
			case COM_GROUP:
				writer.println(getGroup());
				System.out.println(workUser.getUserName() + "请求获得在线
用户详情");
				break;
			case COM_SIGNUP:
				username = (String) gsonMap.get(USERNAME);
				password = (String) gsonMap.get(PASSWORD);
				map.put(COMMAND, COM_RESULT);
				if (createUser(username, password)) {
					//需要马上变更心跳
					currentTime = new Date().getTime();
					//存储信息
					map.put(COM_RESULT, SUCCESS);
					map.put(COM_DESCRIPTION, "success");
					writer.println(gson.toJson(map));
					broadcast(getGroup(),COM_SIGNUP);
					System.out.println("用户" + username + "注册上线了");
				} else {
					map.put(COM_RESULT, FAILED);
```

```
        map.put(COM_DESCRIPTION, username + "已经被注册");
        writer.println(gson.toJson(map)); //返回消息给服务器
        //System.out.println(username + "该用户已经被注册");
    }
    break;
case COM_LOGIN:
    username = (String) gsonMap.get(USERNAME);
    password = (String) gsonMap.get(PASSWORD);
    boolean find = false;
    for (ServerUser u : users) {
        if (u.getUserName().equals(username)) {
            if (!u.getPassword().equals(password)) {
              map.put(COM_DESCRIPTION,  "账号密码输入有误");
                break;
            }
            if (u.getStatus().equals("online")) {
               map.put(COM_DESCRIPTION, "该用户已经登录");
                break;
            }
            currentTime = new Date().getTime();
            map.put(COM_RESULT, SUCCESS);
            map.put(COM_DESCRIPTION, username + "success");
            u.setStatus("online");
            writer.println(gson.toJson(map));
            workUser = u;
            broadcast(getGroup(), COM_SIGNUP);
            find = true;
            System.out.println("用户" + username + "上线了");
            break;
        }
    }
    if (!find) {
        map.put(COM_RESULT, FAILED);
        if (!map.containsKey(COM_DESCRIPTION))
            map.put(COM_DESCRIPTION, username + "未注册");
        writer.println(gson.toJson(map)); //返回消息给服务器
    }
    break;
case COM_CHATWITH:
    String receiver = (String) gsonMap.get(RECEIVER);
    map = new HashMap();
    map.put(COMMAND, COM_CHATWITH);
    map.put(SPEAKER, gsonMap.get(SPEAKER));
    map.put(RECEIVER, gsonMap.get(RECEIVER));
    map.put(CONTENT, gsonMap.get(CONTENT));
    map.put(TIME, getFormatDate());
    for (ServerUser u : users) {
        if (u.getUserName().equals(receiver)) {
            u.addMsg(gson.toJson(map));
```

```
                        break;
                    }
                }
                workUser.addMsg(gson.toJson(map));
                break;
            case COM_CHATALL:
                map = new HashMap();
                map.put(COMMAND, COM_CHATALL);
                map.put(SPEAKER, workUser.getUserName());
                map.put(TIME, getFormatDate());
                map.put(CONTENT, gsonMap.get(CONTENT));
                broadcast(gson.toJson(map), COM_MESSAGEALL);
                break;
            default:
                //System.out.println("");
                break;
        }
    }
    //current time the formatDate String
    public String getFormatDate() {
        Date date = new Date();
        long times = date.getTime();//时间戳
        SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
        String dateString = formatter.format(date);
        return dateString;
    }

    //broadcast the message to all user
    private void broadcast(String message, int type) {
        System.out.println(workUser.getUserName() + " 开始广播 broadcast " +
message);

        switch (type) {
            case COM_MESSAGEALL:
                for (ServerUser u : users) {
                    u.addMsg(message);
                }
                break;
            case COM_LOGOUT:
            case COM_SIGNUP:
                for (ServerUser u : users) {
                    if (!u.getUserName().equals(workUser.getUserName())) {
                        u.addMsg(message);
                    }
                }
                break;
        }
```

```
	}

	//send the message to com.B0529002.Client

	private void sentMessageToClient() {
		String message;
		if (workUser != null)
			while ((message = workUser.getMsg()) != null) {
				writer.println(message); //write it will    auto flush.
				System.out.println(workUser.getUserName()  +  "的数据仓发送
message: " + message + "剩余  size" + workUser.session.size());
			}
	}

	//the    method to release socket's resource.
	private void logOut() {
		if (workUser == null)
			return;
		System.out.println("用户  " + workUser.getUserName() + "  已经离线");
		// still hold this user and change it's status
		workUser.setStatus("offline");
		for (ServerUser u : users) {
			if (u.getUserName().equals(workUser.getUserName()))
				u.setStatus("offline");
		}
		broadcast(getGroup(), COM_LOGOUT);
		isLogOut = true;
	}

	// get a random name
	private String getRandomName() {
		String[] str1 = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j",
				"k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v",
				"w", "x", "y", "z", "1", "2", "3", "4", "5", "6", "7", "8",
				"9", "0", "A", "B", "C", "D", "E", "F", "G", "H", "I", "J",
				"K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V",
				"W", "X", "Y", "Z"};
		StringBuilder name = new StringBuilder();
		String userName = name.toString();
		Random ran = new Random();
		boolean success = false;
		do {
			for (int i = 0; i < 6; i++) {
				int n = ran.nextInt(str1.length);
				String str = str1[n];
				name.append(str);
				System.out.println(name);
			}
			success = true;
```

```
            userName = name.toString();
            for (ServerUser user : users) {
                if (userName.equals(user.getUserName())) {
                    success = false;
                    break;
                }
            }
        } while (!success);
        return userName;
    }

    //create username and bind userName . if failed it will return failed.
    //if success it will add to users.
    private boolean createUser(String userName, String password) {
        for (ServerUser user : users) {
            if (user.getUserName().equals(userName)) {
                return false;
            }
        }
        //add user to userList
        ServerUser   newUser   =   new   ServerUser(users.size(),   userName,
password);
        newUser.setStatus("online");
        users.add(newUser);
        //   add user to db
        try {
            UserDaoImpl.getInstance().add(newUser);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        workUser = newUser;
        return true;
    }

    //return the json of group

    private String getGroup() {
        String[] userlist = new String[users.size() * 2];
        int j = 0;
        for (int i = 0; i < users.size(); i++, j++) {
            userlist[j] = users.get(i).getUserName();
            userlist[++j] = users.get(i).getStatus();
        }
        HashMap map = new HashMap();
        map.put(COMMAND, COM_GROUP);
        map.put(COM_GROUP, userlist);
        return gson.toJson(map);
    }
}
```

Server.MasterServer

```
package com.B0529002.Server;

import com.B0529002.Dao.UserDaoImpl;
import com.B0529002.bean.ServerUser;

import java.io.IOException;
import java.net.ServerSocket;
import java.sql.SQLException;
import java.util.ArrayList;

public class MasterServer {

	//用户列表
	private ArrayList<ServerUser> users;

	public ServerSocket masterServer;
	public WorkServer workServer;

	private int port = 8898;

	public void start() {
		users = new ArrayList<ServerUser>();
		try {
			masterServer = new ServerSocket(port);
			try {
				users                    =                    (ArrayList<ServerUser>)
UserDaoImpl.getInstance().findAll();
				for (ServerUser u:users) {
					u.setStatus("offline");
				}
				System.out.println("get user"+users.size());
			} catch (SQLException e) {
				System.out.println("userList init failed");
				e.printStackTrace();
			}
			System.out.println("server loading");
		} catch (IOException e) {
			e.printStackTrace();
		}

		while (true) {
			try {
				workServer = new WorkServer(masterServer.accept(), users);
```

```
                        workServer.start();
                        System.out.println("workServer product");
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
        }
}
```

GsonUilts

```
package com.B0529002.Utils;

import com.google.gson.*;
import com.google.gson.reflect.TypeToken;

import java.lang.reflect.Type;
import java.util.Map;

public class GsonUtils {
        //      private static Gson gson = new Gson();
        private static Gson gson = new GsonBuilder().
                    registerTypeAdapter(Double.class, new JsonSerializer<Double>() {
                        @Override
                        public JsonElement serialize(Double src, Type typeOfSrc,
JsonSerializationContext context) {
                            if (src == src.intValue())
                                return new JsonPrimitive(src.intValue());
                            return new JsonPrimitive(src);
                        }
                    }).create();

        public static Map<Integer, Object> GsonToMap(String data) {
                Map<Integer, Object> map = gson.fromJson(data, new
TypeToken<Map<Integer, Object>>() {
                }.getType());
                return map;
        }

        public static Integer Double2Integer(Double number){
                return new Integer(number.intValue());
        }
}
```

四、参考资料

http://cooking-java.blogspot.com/2010/03/java-int-to-string.html

https://bbs.csdn.net/topics/390254158

https://m.imooc.com/wenda/detail/351413

http://www.importnew.com/7015.html

https://www.jb51.net/article/132578.htm

https://blog.csdn.net/u010889616/article/details/51477037

https://blog.csdn.net/He11o_Liu/article/details/50776735

https://github.com/sontx/chat-socket

https://www.cnblogs.com/Dyleaf/p/7955145.html

https://www.jianshu.com/p/15c55af10003

https://github.com/playframework/play-java-chatroom-example

https://github.com/vinnyoodles/react-native-socket-io-example

https://github.com/substack/chatwizard

https://github.com/kingston-csj/im

https://lrs.itsa.org.tw/pluginfile.php/18369/mod_resource/content/0/Java_Socket_%E9%80%A3%E7%B7%9A%E7%9A%84%E5%BB%BA%E7%AB%8B%E8%AA%B2%E7%A8%8B_%E6%9F%AF%E5%BF%97%E4%BA%A8%E8%80%81%E5%B8%AB_%E9%9B%BB%E5%AD%90%E6%9B%B8.pdf

https://blog.yslifes.com/archives/652