**武汉大学计算机学院**

**本科生实验报告**

**网络及分布式实验**

专 业 名 称 ：软件工程

课 程 名 称 ：网络及分布式实验

指 导 教 师 ：胡XX 教授

学 生 学 号 ：201730258XX

学 生 姓 名 ：XXX

二○二○年六月

**郑 重 声 明**

本团队呈交的实验报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本实验报告不包含他人享有著作权的内容。对本实验报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本实验报告的知识产权归属于培养单位。

团队成员签名： 日期： 2020-06-01

摘 要

本实验为基于Java + CORBA构建的分布式待办事项列表系统。其中基于JavaFX编写的客户端能够连接到管理各种待办事项列表对象的服务器，实现了查询列表项、添加列表项、删除选定列表项、清除待办事项列表。服务器还应具有允许成员注册的管理功能。每个待办事项列表项都有一个说明、一个开始时间和一个结束时间，用户在此时间内完成其项。用户能够将任意数量的项目添加到其待办事项列表中。

**关键词：**CORBA；分布式应用

**目 录**

[1 实验目的和意义 1](#_Toc42807517)

[1.1 实验内容 1](#_Toc42807518)

[1.2.实验意义 1](#_Toc42807519)

[2 实验设计 2](#_Toc42807520)

[2.1 概述 2](#_Toc42807521)

[2.2 功能模块 2](#_Toc42807522)

[2.3主要功能模块详细设计 2](#_Toc42807523)

[2.4主要代码 4](#_Toc42807524)

[2.5运行方式 17](#_Toc42807525)

[结论 18](#_Toc42807526)

[1 程序主要界面及结果 18](#_Toc42807527)

[1.1客户端注册登录界面： 18](#_Toc42807528)

[1.2客户端待办事项列表主界面： 19](#_Toc42807529)

[1.3 服务端运行结果截图 19](#_Toc42807530)

[1.4 orbd运行截图 19](#_Toc42807531)

[参考资料 21](#_Toc42807532)

# 1 实验目的和意义

## 1.1 实验内容

使用Java和CORBA构建分布式待办事项列表系统。各种客户端应该能够连接到管理各种待办事项列表对象的服务器，以便查询列表项、添加列表项、删除列表项和清除待办事项列表。服务器还应具有允许成员注册的管理功能。每个待办事项列表项都有一个说明、一个开始时间和一个结束时间，用户必须在此时间内完成其项。用户应该能够将任意数量的项目添加到其待办事项列表中。

## 1.2.实验意义

本实验是理论知识和动手能力的综合体现。通过本实验，让学生对计算机网络与分布式系统这门课程更加熟悉和了解。

# 2 实验设计

## 2.1 概述

基于Java+CORBA构建的分布式待办事项列表系统。

项目代码开源地址为：<https://github.com/myHomeworks/WHU-TO_DO_LIST>

## 2.2 功能模块

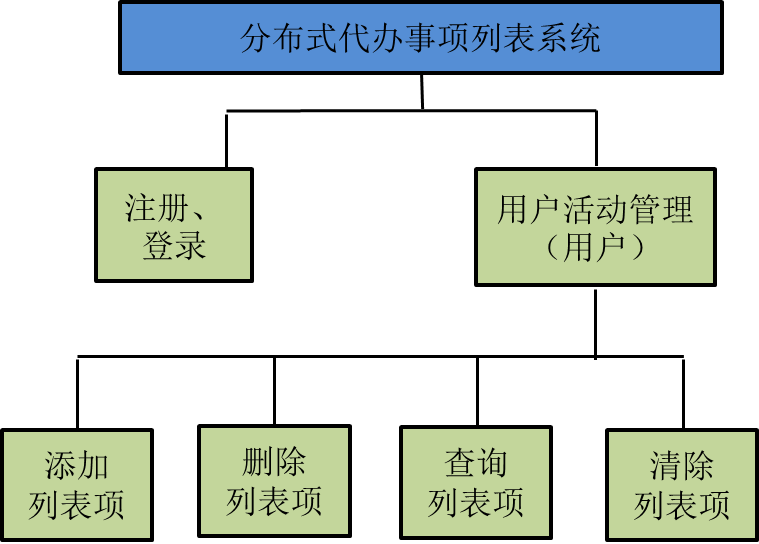
****

图2.1 功能模块

## 2.3主要功能模块详细设计

**2.3.1 注册模块**

i、功能描述

用户提供账户名、密码等个人信息，完成注册功能。

ii、输入

|  |  |
| --- | --- |
| 输入项 | 要求 |
| 账户名 | 文本输入，不能为空 |
| 密码 | 不能为空 |

iii、输出

若注册成功，则显示绿色“注册成功”提示字样；失败则显示红色“账户已存在”、“注册失败”等提示字样。

**2.3.2 登录模块**

i、功能描述

用户输入账户名、密码等个人信息，完成登录功能。

ii、输入

|  |  |
| --- | --- |
| 输入项 | 要求 |
| 账户名 | 文本输入，不能为空 |
| 密码 | 不能为空 |

iii、输出

身份验证通过后，系统将跳转至待办事项主页面，并初始化主界面；失败则显示红色“账户或密码错误”等提示字样。

**2.3.3 新增代办项模块**

i、功能描述

用户输入开始时间、结束时间、备注等信息，完成新增待办事项功能。

ii、输入

|  |  |
| --- | --- |
| 输入项 | 要求 |
| 开始时间 | 文本输入“yyyy-MM-dd HH:mm”格式的时间，不能为空 |
| 结束时间 | 文本输入“yyyy-MM-dd HH:mm”格式的时间，不能为空 |
| 备注说明 | 文本输入，不能为空 |

iii、输出

若输入合法，并成功添加，将清除输入项，并刷新已有代办事项列表；若失败，则需继续编辑输入。

**2.3.4 删除选定待办项模块**

i、功能描述

用户删除选定的待办事项。

ii、输入

用户鼠标选定某一待办事项后，同时点击“删除”按钮。

iii、输出

刷新删除目标待办事项后的待办事项列表。

**2.3.5按时间查询选定待办项模块**

i、功能描述

用户输入开始时间、结束时间等信息，完成查询待办事项功能。

ii、输入

|  |  |
| --- | --- |
| 输入项 | 要求 |
| 开始时间 | 文本输入“yyyy-MM-dd HH:mm”格式的时间，不能为空 |
| 结束时间 | 文本输入“yyyy-MM-dd HH:mm”格式的时间，不能为空 |

iii、输出

若输入合法，并成功查询，将刷新待办事项列表；若失败，则需继续编辑输入。

**2.3.6清除全部选定待办项模块**

i、功能描述

用户清除所有的待办事项。

ii、输入

用户点击“清除”按钮

iii、输出

刷新列表内容为空的待办事项列表。

## 2.4主要代码

**2.4.1 idl**

**user.idl代码如下:**

**module** UserModule{  
 **interface** User{  
 **boolean** add(**in string** startTime,**in string** endTime,**in string** label);  
 **string** query(**in string** startTime,**in string** endTime);  
 **boolean** delete(**in string** key);  
 **boolean** clear();  
 **string** show();  
 };  
};

add()、query()、delete()、clear()、show()等方法分别为客户端对to-do-list中的item进行添加、查询、删除、清除、全部显示等操作。

creator.idl

**module** CreatorModule{  
 **interface** Creator{  
 **boolean** login(**in string** name,**in string** password);  
 **boolean** register(**in string** name,**in string** password);  
 };  
};

login()、register()方法依次为提供用户登录、注册的功能。

在实现上述.idl文件后，在shell中相关文件夹依次执行>idlj –fall user.idl、>idlj –fall creator.idl两条命令。

**2.4.2 idl相关的其他类**

**Item.java:**

**package** model;  
  
**import** java.io.Serializable;  
**import** java.util.Date;  
  
**public class** Item **implements** Serializable{  
 **private** Date **startTime**;  
 **private** Date **endTime**;  
 **private** String **label**;  
  
 **public** Item(Date startTime, Date endTime, String label) {  
 **this**.**startTime** = startTime;  
 **this**.**endTime** = endTime;  
 **this**.**label** = label;  
 }  
  
 **public** Date getStartTime() {  
 **return startTime**;  
 }  
  
 **public** Date getEndTime() {  
 **return endTime**;  
 }  
  
 **public** String getLabel() {  
 **return label**;  
 }  
}

Person.java:

**package** model;  
  
**import** java.io.Serializable;  
  
**public class** Person **implements** Serializable{  
 **private** String **name**;  
 **private** String **password**;  
  
 **public** Person(String name, String password) {  
 **this**.**name** = name;  
 **this**.**password** = password;  
 }  
  
 **public** String getPassword() {  
 **return password**;  
 }  
}

**UserImpl.java 实现user.idl中定义的接口**

**package** UserModule;  
  
**import** model.Item;  
**import** org.omg.CORBA.ORB;  
  
**import** java.io.FileInputStream;  
**import** java.io.FileOutputStream;  
**import** java.io.ObjectInputStream;  
**import** java.io.ObjectOutputStream;  
**import** java.text.DateFormat;  
**import** java.util.\*;  
**import** com.alibaba.fastjson.JSON;  
  
**public class** UserImpl **extends** UserPOA {  
 **private** Date **date**;  
 **private** DateFormat **dateFormat**;  
 **private** String **dateRegex**;  
 **private** List<Item> **personalList**; *//单个用户的to-do-list* **private** Map<String, List<Item>> **allPersonListMap**; *//所有用户的to-do-list* **private** ORB **orb**;  
  
 **public** UserImpl(String name) {  
 init();  
 **personalList** = **allPersonListMap**.get(name);  
 **if** (**personalList** == **null**) {  
 **personalList** = **new** ArrayList<Item>();  
 **allPersonListMap**.put(name, **personalList**);  
 }  
  
 **date** = **new** Date();  
 **dateFormat** = DateFormat.*getDateInstance*();  
 *//日期格式规定为：xxxx-xx-xx,xx:xx* **dateRegex** = **"\\d{4}-\\d{1,2}-\\d{1,2} \\d{2}:\\d{2}"**;  
 }  
  
 **private void** init() {  
 *//从文件中读取to-do-list列表,转化为HashMap* **try** {  
 FileInputStream fin = **new** FileInputStream(**"item.file"**);  
 ObjectInputStream oin = **new** ObjectInputStream(fin);  
 **try** {  
 Object object = oin.readObject();  
 **allPersonListMap** = (HashMap<String, List<Item>>) object;  
 } **catch** (ClassNotFoundException e) {  
 System.***out***.println(**"object cast error"**);  
 **allPersonListMap** = **new** HashMap<String, List<Item>>();  
 }  
 oin.close();  
 fin.close();  
 } **catch** (Exception e) {  
 **allPersonListMap** = **new** HashMap<String, List<Item>>();  
 }  
 }  
  
 **public void** setORB(ORB orb) {  
 **this**.**orb** = orb;  
 }  
  
 *//将to-do-list表保存到本地文件中* **private void** saveData() {  
 **try** {  
 FileOutputStream fout = **new** FileOutputStream(**"item.file"**);  
 ObjectOutputStream oout = **new** ObjectOutputStream(fout);  
 oout.writeObject(**allPersonListMap**);  
 oout.flush();  
 fout.flush();  
 oout.close();  
 fout.close();  
 } **catch** (Exception e) {  
 System.***out***.println(**"save error."**);  
 e.printStackTrace();  
 }  
 }  
  
 *//判断输入日期是否符合格式要求* **private boolean** isFormatMatch(String dateStr) {  
 **return** dateStr.matches(**dateRegex**);  
 }  
  
 *//将字符串转化为日期* **private** Date turnToDate(String dateStr) {  
 String[] str = dateStr.split(**"[,|:]"**);  
 **try** {  
 **date** = **dateFormat**.parse(str[0]);  
 **date**.setHours(Integer.*parseInt*(str[1]));  
 **date**.setMinutes(Integer.*parseInt*(str[2]));  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
  
 **return date**;  
 }  
  
 @Override  
 **public boolean** add(String startTime, String endTime, String label) {  
 Date startDate, endDate;  
 **if** (isFormatMatch(startTime) && isFormatMatch(endTime)) {  
 startDate = turnToDate(startTime);  
 endDate = turnToDate(endTime);  
 Item item = **new** Item(startDate, endDate, label);  
 **personalList**.add(item);  
 saveData();  
 **return true**;  
 } **else** {  
 **return false**;  
 }  
 }  
  
 @Override  
 **public** String query(String startTime, String endTime) {  
 Date startDate, endDate;  
 List<Item> items = **new** ArrayList<>();  
 **if** (isFormatMatch(startTime) && isFormatMatch(endTime)) {  
 startDate = turnToDate(startTime);  
 endDate = turnToDate(endTime);  
 **for** (Item item : **personalList**) {  
 **if** (item.getStartTime().after(startDate)  
 && item.getEndTime().before(endDate)) {  
 items.add(item);  
 }  
 }  
 } **else** {  
 **return null**;  
 }  
 **return** JSON.*toJSON*(items).toString();  
 }  
  
 @Override  
 **public** String show() {  
 **if** (**personalList**.size() > 0) {  
 **return** JSON.*toJSON*(**personalList**).toString();  
 } **else** {  
 **return null**;  
 }  
 }  
  
 @Override  
 **public boolean** delete(String key) {  
 **int** index = Integer.*parseInt*(key);  
 **if** (index <= **personalList**.size() && index >= 1) {  
 **personalList**.remove(index - 1);  
 saveData();  
 **return true**;  
 } **else** {  
 **return false**;  
 }  
 }  
  
 @Override  
 **public boolean** clear() {  
 **int** index = **personalList**.size() - 1;  
 **if** (index < 0) {  
 **return false**;  
 } **else** {  
 **while** (index >= 0) {  
 **personalList**.remove(index);  
 index--;  
 }  
 saveData();  
 **return true**;  
 }  
 }  
}

**2.4.3 JavaFX相关类**

**Main.java 包括注册登录的界面**

**package** sample;  
  
**import** UserModule.UserHelper;  
**import** client.ToDoListClient;  
**import** com.alibaba.fastjson.JSONObject;  
**import** javafx.application.Application;  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.event.ActionEvent;  
**import** javafx.event.EventHandler;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.geometry.Insets;  
**import** javafx.geometry.Pos;  
**import** javafx.scene.Group;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.scene.control.\*;  
**import** javafx.scene.control.cell.PropertyValueFactory;  
**import** javafx.scene.layout.GridPane;  
**import** javafx.scene.layout.HBox;  
**import** javafx.scene.layout.VBox;  
**import** javafx.scene.paint.Color;  
**import** javafx.scene.text.Font;  
**import** javafx.scene.text.FontWeight;  
**import** javafx.scene.text.Text;  
**import** javafx.stage.Stage;  
**import** model.Item;  
**import** org.omg.CosNaming.NamingContextPackage.CannotProceed;  
**import** org.omg.CosNaming.NamingContextPackage.InvalidName;  
**import** org.omg.CosNaming.NamingContextPackage.NotFound;  
  
**import** java.text.ParseException;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Date;  
**import** java.util.List;  
  
**public class** Main **extends** Application {  
 SimpleDateFormat **simpleDateFormat** = **new** SimpleDateFormat(**"yyyy-MM-dd HH:mm"**);*//注意月份是MM* **private final** TableView<Item> **tableView** = **new** TableView<>();  
 **private final** ObservableList<Item> **data** = FXCollections.*observableArrayList*();  
  
 **public** Main() **throws** ParseException {  
 }  
  
 @Override  
 **public void** start(**final** Stage primaryStage) **throws** Exception {  
 ToDoListClient.*init*();  
  
*// Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));* primaryStage.setTitle(**"Login"**);  
*// primaryStage.setScene(new Scene(root, 300, 275));* primaryStage.show();  
  
 *// gridPane before login* GridPane loginGridPane = **new** GridPane();  
 loginGridPane.setAlignment(Pos.***CENTER***);  
 loginGridPane.setHgap(10);  
 loginGridPane.setVgap(10);  
 loginGridPane.setPadding(**new** Insets(25, 25, 25, 25));  
  
 *// loginGridPane row 0, welcome msg* Text sceneTitle = **new** Text(**"Welcome to TO-DO-LIST system!"**);  
 sceneTitle.setFont(Font.*font*(**"Microsoft YaHei"**, FontWeight.***NORMAL***, 20));  
 loginGridPane.add(sceneTitle, 0, 0, 2, 1);  
  
 *// loginGridPane row 1, username* **final** Label username = **new** Label(**"账号"**);  
 loginGridPane.add(username, 0, 1);  
 **final** TextField usernameTextField = **new** TextField();  
 loginGridPane.add(usernameTextField, 1, 1);  
  
 *// loginGridPane row 2, password* Label pwd = **new** Label(**"密码"**);  
 loginGridPane.add(pwd, 0, 2);  
 **final** PasswordField pwdPwdField = **new** PasswordField();  
 loginGridPane.add(pwdPwdField, 1, 2);  
  
 *// loginGridPane row 3, register-btn* Button registerButton = **new** Button(**"注册"**);  
 HBox hBoxReg = **new** HBox(1);  
 hBoxReg.setAlignment(Pos.***BOTTOM\_LEFT***);  
 hBoxReg.getChildren().add(registerButton);  
 loginGridPane.add(hBoxReg, 0, 4);  
  
 *// loginGridPane row 3, login-btn* Button loginButton = **new** Button(**"登录"**);  
 HBox hBoxLog = **new** HBox(1);  
 hBoxLog.setAlignment(Pos.***BASELINE\_RIGHT***);  
 hBoxLog.getChildren().add(loginButton);  
 loginGridPane.add(hBoxLog, 1, 4);  
  
 *// loginGridPane row 6, tips* **final** Text tipText = **new** Text();  
 loginGridPane.add(tipText, 1, 6);  
  
 *// scene of login* Scene loginScene = **new** Scene(loginGridPane, 400, 275);  
 primaryStage.setScene(loginScene);  
  
 **final** Scene tableScene = **new** Scene(**new** Group());  
  
 **tableView**.setEditable(**false**);  
  
 **final** TableColumn startTimeCol = **new** TableColumn(**"Start Time"**);  
 startTimeCol.setMinWidth(300);  
 startTimeCol.setCellValueFactory(**new** PropertyValueFactory<>(**"startTime"**));  
  
 TableColumn endTimeCol = **new** TableColumn(**"End Time"**);  
 endTimeCol.setMinWidth(300);  
 endTimeCol.setCellValueFactory(**new** PropertyValueFactory<>(**"endTime"**));  
  
 TableColumn markCol = **new** TableColumn(**"Mark"**);  
 markCol.setMinWidth(300);  
 markCol.setCellValueFactory(**new** PropertyValueFactory<>(**"label"**));  
  
 **tableView**.setItems(**data**);  
 **tableView**.getColumns().addAll(startTimeCol, endTimeCol, markCol);  
  
 **final** Text mainSceneTitle = **new** Text(**"Welcome, "**);  
 mainSceneTitle.setFont(Font.*font*(**"Tahoma"**, FontWeight.***NORMAL***, 20));  
  
 **final** TextField addStartTime = **new** TextField();  
 addStartTime.setPromptText(**"start time"**);  
 addStartTime.setMaxWidth(startTimeCol.getPrefWidth());  
 addStartTime.setMinWidth(200);  
  
 **final** TextField addEndTime = **new** TextField();  
 addEndTime.setPromptText(**"end time"**);  
 addEndTime.setMaxWidth(endTimeCol.getPrefWidth());  
 addEndTime.setMinWidth(200);  
  
 **final** TextField addMark = **new** TextField();  
 addMark.setPromptText(**"marks"**);  
 addMark.setMaxWidth(markCol.getPrefWidth());  
 addMark.setMinWidth(250);  
  
 **final** Button addBtn = **new** Button(**"添加"**);  
 **final** Button searchBtn = **new** Button(**"查询"**);  
 **final** Button delBtn = **new** Button(**"删除"**);  
 **final** Button clearBtn = **new** Button(**"清除"**);  
  
 **final** HBox addHBox = **new** HBox();  
 addHBox.getChildren().addAll(addStartTime, addEndTime, addMark, addBtn, searchBtn, delBtn, clearBtn);  
 addHBox.setSpacing(3);  
  
 **final** VBox vbox = **new** VBox();  
 vbox.setSpacing(5);  
 vbox.setPadding(**new** Insets(10, 0, 0, 10));  
 vbox.getChildren().addAll(mainSceneTitle, **tableView**, addHBox);  
  
 ((Group) tableScene.getRoot()).getChildren().addAll(vbox);  
  
 clearBtn.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent event) {  
 ToDoListClient.*user*.clear();  
 **data**.clear();  
 }  
 });  
  
 delBtn.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent event) {  
*// Item itemToDel = tableView.getSelectionModel().getSelectedItem();* **int** index = **tableView**.getSelectionModel().getSelectedIndex();  
 ToDoListClient.*user*.delete(String.*valueOf*(index));  
 **data**.remove(index);  
 }  
 });  
  
 searchBtn.setOnAction((ActionEvent e) -> {  
 **try** {  
 String startTime = addStartTime.getText();  
 String endTime = addEndTime.getText();  
  
 String jsonStr = ToDoListClient.*user*.query(startTime, endTime);  
 List lists = JSONObject.*parseArray*(jsonStr);  
 **data**.clear();  
 **for** (**int** i = 0; i < lists.size(); i++) {  
 JSONObject jsonObject = (JSONObject) lists.get(i);  
 String startTime1 = jsonObject.getString(**"startTime"**);  
 String endTime1 = jsonObject.getString(**"endTime"**);  
 String mark = jsonObject.getString(**"label"**);  
 **data**.add(**new** Item(**new** Date(Long.*parseLong*(startTime1)),  
 **new** Date(Long.*parseLong*(endTime1)), mark));  
 }  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 });  
  
 addBtn.setOnAction((ActionEvent e) -> {  
 **try** {  
 String startTime = addStartTime.getText();  
 String endTime = addEndTime.getText();  
 String mark = addMark.getText();  
 **if** (ToDoListClient.*user*.add(startTime, endTime, mark)) {  
 **data**.add(**new** Item(**simpleDateFormat**.parse(startTime),  
 **simpleDateFormat**.parse(endTime), mark));  
 addStartTime.clear();  
 addEndTime.clear();  
 addMark.clear();  
 }  
 } **catch** (ParseException ex) {  
 ex.printStackTrace();  
 }  
 });  
  
 *// handle register event* registerButton.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent event) {  
 **if** (ToDoListClient.*creator*.register(usernameTextField.getText(), pwdPwdField.getText())) {  
 tipText.setFill(Color.***GREEN***);  
 tipText.setText(**"注册成功"**);  
 } **else** {  
 tipText.setFill(Color.***FIREBRICK***);  
 tipText.setText(**"注册失败"**);  
 }  
 }  
 });  
  
 *// handle login even --> will open new page if login successfully* loginButton.setOnAction(**new** EventHandler<ActionEvent>() {  
 @Override  
 **public void** handle(ActionEvent event) {  
 **try** {  
  
 **if** (ToDoListClient.*creator*.login(usernameTextField.getText(), pwdPwdField.getText())) {  
 **try** {  
 ToDoListClient.*user* = UserHelper.*narrow*(  
 ToDoListClient.*ncRef*.resolve\_str(usernameTextField.getText()));  
 } **catch** (NotFound | InvalidName | CannotProceed e) {  
 e.printStackTrace();  
 }  
 tipText.setFill(Color.***GREEN***);  
 tipText.setText(**"登录成功"**);  
 primaryStage.setTitle(**"To-DO-LIST Homepage"**);  
 primaryStage.setWidth(950);  
 primaryStage.setHeight(600);  
 mainSceneTitle.setText(mainSceneTitle.getText() + usernameTextField.getText() + **"!"**);  
 primaryStage.setScene(tableScene);  
  
 String jsonStr = ToDoListClient.*user*.show();  
  
 List lists = JSONObject.*parseArray*(jsonStr);  
 **for** (**int** i = 0; i < lists.size(); i++) {  
 JSONObject jsonObject = (JSONObject) lists.get(i);  
 String startTime = jsonObject.getString(**"startTime"**);  
 String endTime = jsonObject.getString(**"endTime"**);  
 String mark = jsonObject.getString(**"label"**);  
 **data**.add(**new** Item(**new** Date(Long.*parseLong*(startTime)),  
 **new** Date(Long.*parseLong*(endTime)), mark));  
 }  
  
 } **else** {  
 tipText.setFill(Color.***FIREBRICK***);  
 tipText.setText(**"账户或密码错误，登录失败"**);  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 tipText.setFill(Color.***FIREBRICK***);  
 tipText.setText(**"登录失败"**);  
 }  
 }  
 });  
 }  
  
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
}

## 2.5运行方式

**2.5.1 运行环境**

操作系统：Windows 10

所用语言：JDK8

IDE: IntelliJ IDEA 2019.2.4 (Ultimate Edition)

引用依赖：fastjson-1.1.36.jar

**2.5.2 orbd**

在命令行终端运行-> start orbd -ORBInitialPort 1050 -ORBInitialHost localhost

**2.5.3 server**

在IDEA中导入ToDoListServer项目，并运行src/server/ToDoListServer.

java中main方法。

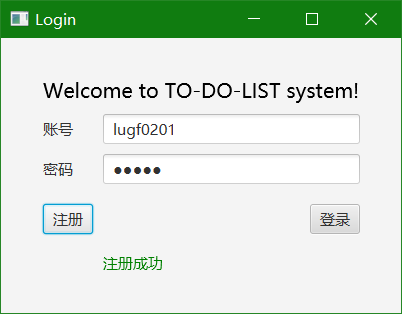
**2.5.4 client**

在IDEA中导入Client项目，并运行src/sample/Main.Java中main方法。

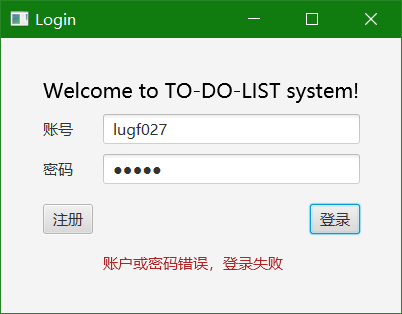
# 结论

## 1 程序主要界面及结果

### 1.1客户端注册登录界面：

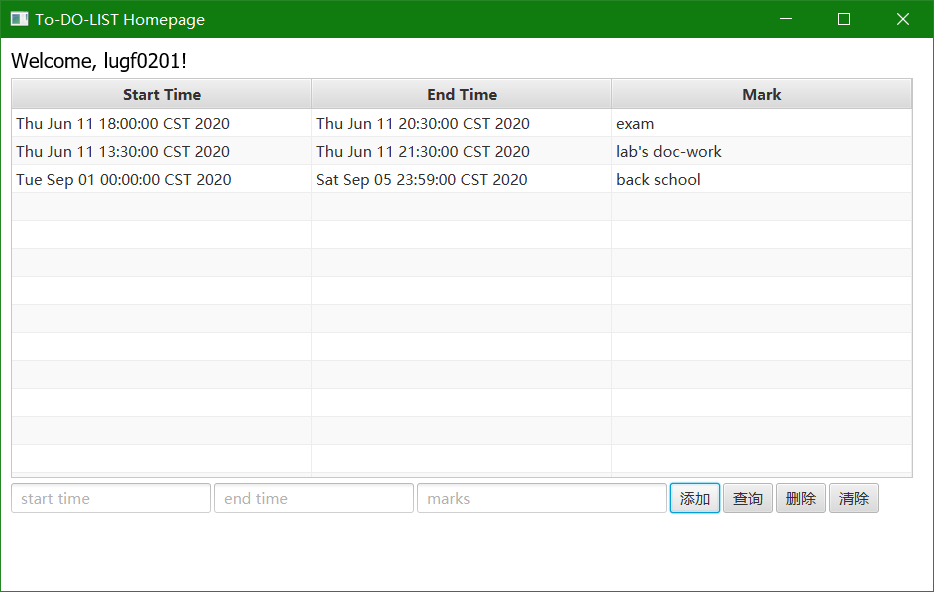


附图1 注册成功页面



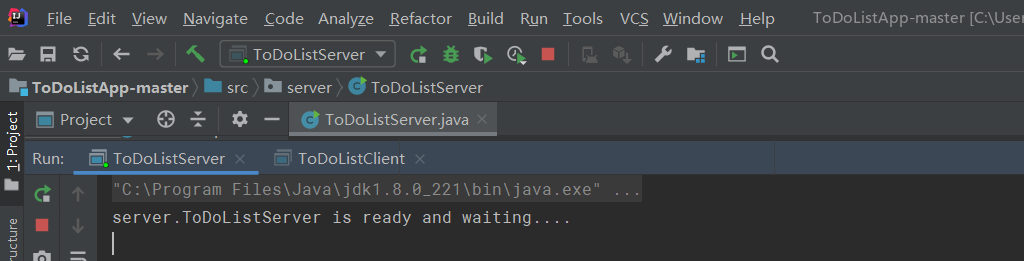
附图2 登录失败页面

### 1.2客户端待办事项列表主界面：

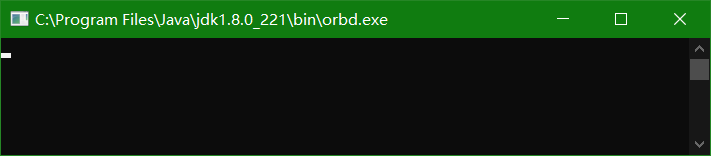


附图3 代办事项主页面

### 1.3 服务端运行结果截图



### 1.4 orbd运行截图



# 参考资料

[1] JavaTM IDL Technology Documentation at <http://java.sun.com/j2se/1.5/docs/guide/idl/index.html>.

[2] Java语言程序设计（基础班 原书第10版） 梁勇（Y.Daniel Liang） 机械工业出版社

|  |
| --- |
| 【结论】： |
| 【小结】： |
| 指导老师评语及成绩 |
| 【评语】：  成 绩： 指导老师签名：  批阅日期： |