|  |  |
| --- | --- |
| **项目编号** |  |
| **文档编号** |  |
| **密级** |  |



**作业管理系统的设计与实现**

**详细设计**

学 院 **人工智能学院**

专 业 **软件工程**

年级班级 **2018级**

学生姓名 **作业管理系统小组**

撰写日期：2021年6月22日

## 版本更新信息

本文档的更新记录如表所示。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 修改编号 | 修改日期 | 修改人 | 修改后版本 | 修改位置 | 修改内容概述 |
| 001 | 2021.6.14 | 龙超伟 | 0.1 | 全部 | 初始发布版本 |
| 002 | 2021.6.16 | 龙超伟 | 1.0 | 3章节 | 增加 |
| 003 | 2021.6.18 | 龙超伟、李明辉、托朝健 | 2.0 | 4章节 | 修改 |
| 004 | 2021.6.21 | 龙超伟、李明辉、托朝健 | 3.0 | 4、5章节 | 修改 |
| 005 | 2021.6.24 | 龙超伟、李明辉、托朝健 | 4.0 | 5、6章节 | 修改、增加 |

目录

[1．导言 3](#_Toc75734133)

[1.1 目的 4](#_Toc75734134)

[1.2 范围 4](#_Toc75734135)

[1.3 引用标准 4](#_Toc75734136)

[1.4 参考资料 4](#_Toc75734137)

[1.5 版本更新信息 4](#_Toc75734138)

[2．详细设计简述 4](#_Toc75734139)

[2.1 设计简介 4](#_Toc75734140)

[2.2 模块简介 4](#_Toc75734141)

[3、界面详细设计 4](#_Toc75734142)

[3.1、页面设计说明 4](#_Toc75734143)

[3.2、静态页面详细设计 4](#_Toc75734144)

[4、客户端模块详细设计 4](#_Toc75734145)

[4.2、登录模块 4](#_Toc75734146)

[4.2.1 表现层 4](#_Toc75734147)

[4.2.2 控制层 4](#_Toc75734148)

[4.2.3 业务逻辑层 4](#_Toc75734149)

[4.2.4 数据持久层 4](#_Toc75734150)

[4.2.5 域模型层 4](#_Toc75734151)

[4.2、学生作业提交模块模块 4](#_Toc75734152)

[4.2.1 表现层 4](#_Toc75734153)

[4.2.2 控制层 4](#_Toc75734154)

[4.2.3 业务逻辑层 4](#_Toc75734155)

[4.2.4 数据持久层 4](#_Toc75734156)

[4.2.5 域模型层 4](#_Toc75734157)

[4.3、教师下载作业提交模块模块 4](#_Toc75734158)

[4.2.1 表现层 4](#_Toc75734159)

[4.2.2 控制层 4](#_Toc75734160)

[4.2.3 业务逻辑层 4](#_Toc75734161)

[4.2.4 数据持久层 4](#_Toc75734162)

[4.2.5 域模型层 4](#_Toc75734163)

[5、管理端模块详细设计 4](#_Toc75734164)

[5.1、管理员模块详细设计 4](#_Toc75734165)

[5.1.1 表现层 4](#_Toc75734166)

[5.1.2 控制层 4](#_Toc75734167)

[5.1.3 业务逻辑层 4](#_Toc75734168)

[5.1.4 数据持久层 4](#_Toc75734169)

[5.1.5 域模型层 4](#_Toc75734170)

[5.2、教师管理模块详细设计 4](#_Toc75734171)

[5.2.1 表现层 4](#_Toc75734172)

[5.2.2 控制层 4](#_Toc75734173)

[5.2.3 业务逻辑层 4](#_Toc75734174)

[5.2.4 数据持久层 4](#_Toc75734175)

[5.2.5 域模型层 4](#_Toc75734176)

[5.3、学生管理模块详细设计 4](#_Toc75734177)

[5.3.1 表现层 4](#_Toc75734178)

[5.3.2 控制层 4](#_Toc75734179)

[5.3.3 业务逻辑层 4](#_Toc75734180)

[5.3.4 数据持久层 4](#_Toc75734181)

[5.3.5 域模型层 4](#_Toc75734182)

# 

# 1．导言

## 1.1 目的

该文档是对作业管理系统概要设计进行的详细说明，是为了指导和规范作业管理系统开发而制定的详细开发设计文档

本文档的预期读者包括：

* 设计开发人员
* 项目管理人员
* 测试人员
* 学生、教师、管理员

## 1.2 范围

该文档内容涵盖作业管理系统系统的所有功能模块的详细设计描述

## 1.3 引用标准

[1] 《软件设计模式》 朱洪军等 “十三五”规范计划

## 1.4 参考资料

[1] 《软件设计模式》 朱洪军等 “十三五”规范计划

## 1.5 版本更新信息

本文档的更新记录如下表所示。

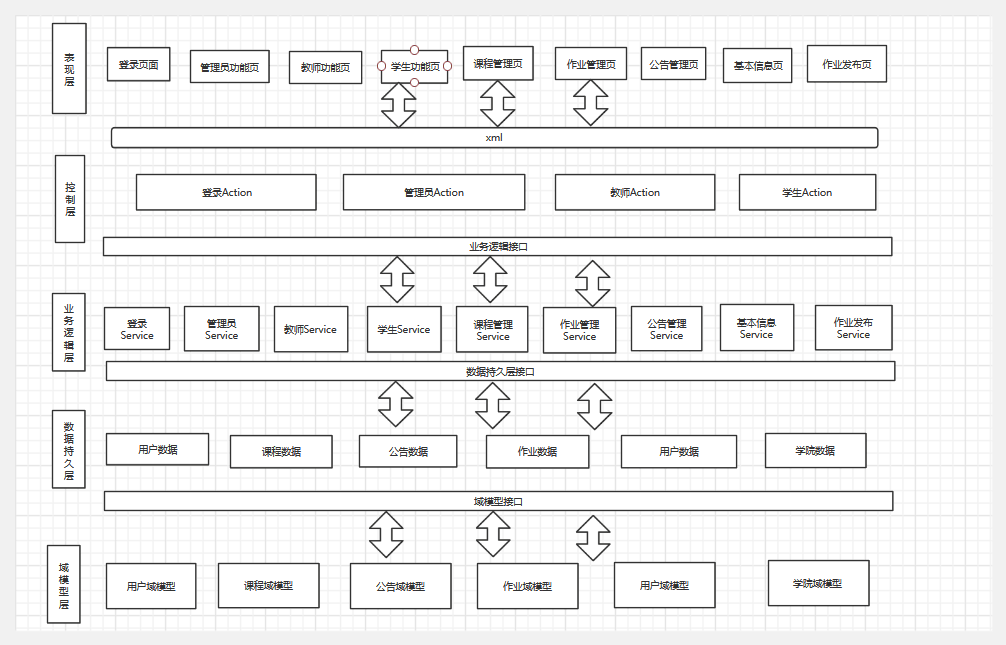
表1-1 文档更新记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改编号 | 修改日期 | 修改人 | 修改后版本 | 修改位置 |
| 001 | 2021.6.14 | 龙超伟 | 0.1 | 全部 |
| 002 | 2021.6.16 | 龙超伟 | 1.0 | 3章节 |
| 003 | 2021.6.18 | 龙超伟、李明辉、托朝健 | 2.0 | 4章节 |
| 004 | 2021.6.21 | 龙超伟、李明辉、托朝健 | 3.0 | 4、5章节 |
| 005 | 2021.6.24 | 龙超伟、李明辉、托朝健 | 4.0 | 5、6章节 |

# 2．详细设计简述

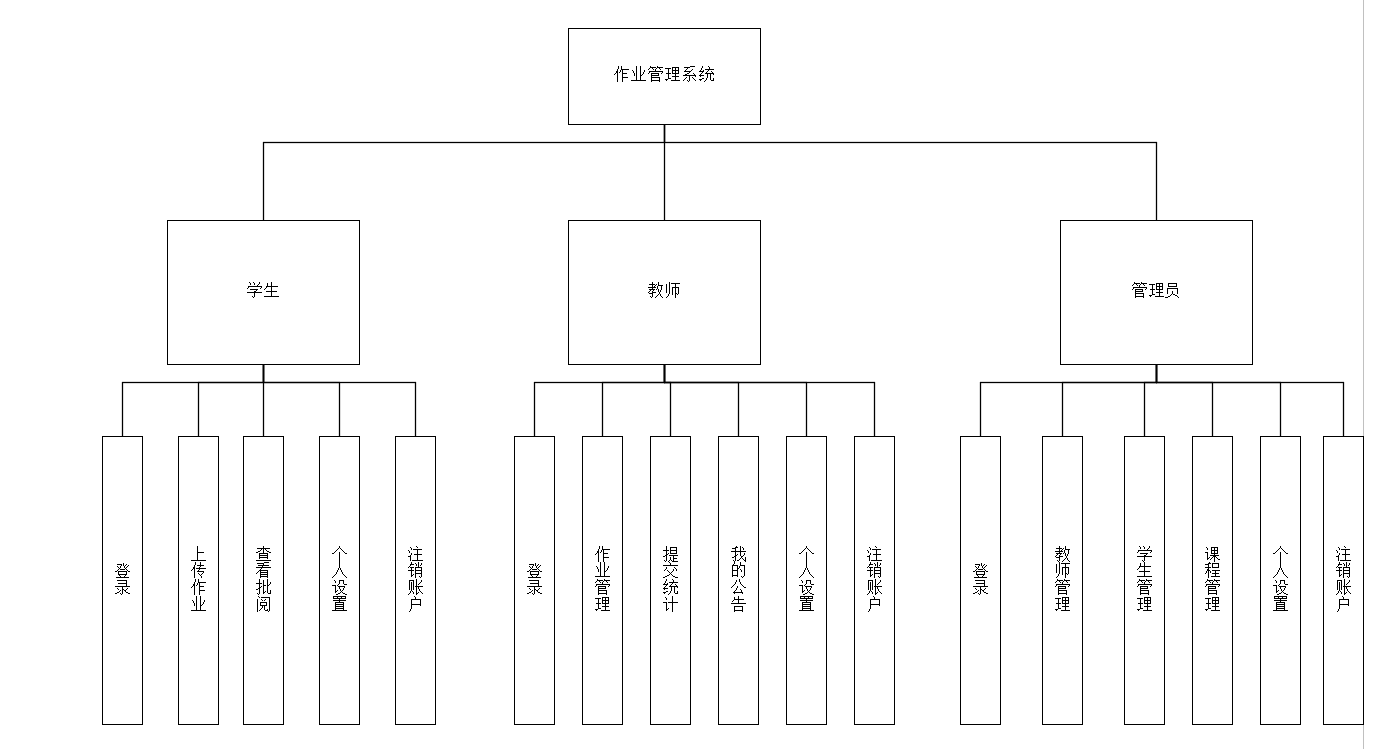
## 2.1 设计简介

作业管理系统分层结构图如图所示。



## 2.2 模块简介

系统功能模块图如图所示。



# 3、界面详细设计

## 3.1、页面设计说明

界面设计技术：JSP

## 3.2、静态页面详细设计

表3-1 页面表现层设计

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 登录界面 | login.jsp | 实现登录界面 |
| 学生主界面 | Student.jsp | 实现学生选择功能界面 |
| 老师主界面 | Teacher.jsp | 实现老师选择功能页面 |
| 管理员主页面 | Manager.jsp | 实现管理员选择功能界面 |
| 学生公告，作业查看界面 | S\_home.jsp | 实现在主界面下查看公告和作业 |
| 学生下载作业界面 | S\_downloadwork.jsp | 实现下载作业功能 |
| 学生信息界面 | S\_personalseting.jsp | 实现学生个人信息界面 |
| 学生批阅界面 | S\_showcheck.jsp | 实现查看老师批阅内容 |
| 学生提交作业界面 | S\_uploadwork.jsp | 实现学生提交作业功能 |
| 管理员增加课程界面 | M\_addcourse.jsp | 实现管理员增加课程功能 |
| 管理员增加学生界面 | M\_addstudent.jsp | 实现管理员增加学生功能 |
| 管理员增加老师界面 | M\_addteacher.jsp | 实现管理员增加老师功能 |
| 管理员课程管理界面 | M\_coursemanage.jsp | 实现课程的查删功能 |
| 管理员个人信息界面 | M\_personalseting.jsp | 实现管理员个人信息增删查改功能 |
| 管理员管理学生界面 | M\_studentmange.jsp | 实现管理员删查学生功能 |
| 管理员管理教师界面 | m\_teachermanage.jsp | 实现管理员删查教师功能 |
| 管理员课程修改界面 | m\_updatecourse.jsp | 实现管理员修改课程功能 |
| 管理员学生修改界面 | m\_updatestudent.jsp | 实现管理员修改学生功能 |
| 管理员教师修改界面 | m\_updateteacher.jsp | 实现管理员修改教师功能 |
| 教师界面总览界面 | t\_allnotice.jsp | 实现教师删除，查看公告功能 |
| 教师作业总览界面 | t\_allworks.jsp | 实现教师删除，查看作业功能 |
| 教师个人信息界面 | t\_personalseting.jsp | 实现教师查看，修改个人信息功能 |
| 教师发布公告界面 | t\_publishnotice.jsp | 实现教师发布公告功能 |
| 教师发布作业界面 | t\_publishwork.jsp | 实现教师发布作业功能 |
| 教师提交统计界面 | t\_statistics.jsp | 实现教师提交统计功能 |
| 教师下载学生作业界面 | t\_statistics\_details.jsp | 实现教师下载学生作业功能 |

# 4、客户端模块详细设计

客户端模块主要包括登录模块，教师主界面模块，学生主界面模块，管理员主界面模块，学生作业提交模块，教师下载作业模块，管理员添加模块。

## 4.2、登录模块

在登录模块时，系统内部的相应响应操作示意图如图4-2所示。

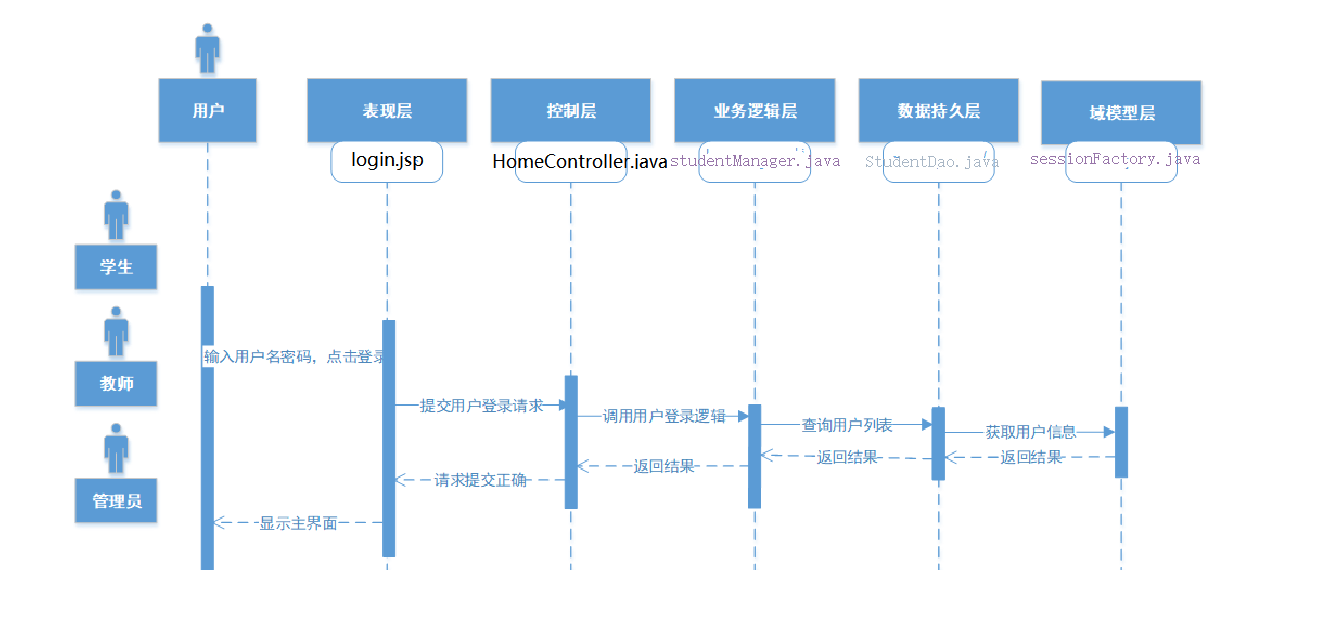


图 4-1系统响应示意图

### 4.2.1 表现层

登录模块的表现层主要完成不同用户的登录功能，登录页面时要求用户输入账号密码的基本信息，确认后页面给出响应消息，提示登录成功或失败的提示。表现层对应的JSP页面列表见表?-?所示。

表 4-2 登入模块表现层JSP列表

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 登录页面 | login.jsp | 用户（教师、学生、游客）登入功能，当登入出错时给出提示。 |

login.jsp的流程图如图4-1所示。

进入游首页



图 4-1 登录模块详细设计流程图

### 4.2.2 控制层

登录模块的控制层负责接受来自login.jsp的用户输入，同时调用登入模块的业务逻辑接口，将用户名与密码等用户关键信息传递到业务逻辑层进行判定。等到业务逻辑处理完成之后，将来自业务逻辑层的相应信息传到表现层，并决定显示页面。登入模块控制层列表见表4-3所示。

表 4-3登入模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 登录 | Login.java | SUCCESS | Student.jsp或teacher.jsp或manger.jsp—登入成功，显示提示窗口。 |
| ERROR | login.jsp---登入失败，显示提示窗口。 |

在控制层中HomeController.java的描述如下所示

package wms.controller;

import java.util.ArrayList;

import java.util.List;

import javax.annotation.Resource;

import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import wms.entity.Manager;

import wms.entity.Student;

import wms.entity.Teacher;

import wms.service.IManagerService;

import wms.service.IStudentManager;

import wms.service.ITeacherManager;

@Controller

public class HomeController {

/\*\*

\* 用于实现登录版块功能

\* @Resource注解用于实现依赖注入

\* @requestmapping注解用于映射地址

\*/

@Resource(name="managerManager")

private IManagerService managerManager;

@Resource(name="teacherManager")

private ITeacherManager teacherManager;

@Resource(name="studentManager")

private IStudentManager studentManager;

/\*\*

\* 请求登录页面

\*/

@RequestMapping("/")

public String index(){

return "login";

}

/\*\*

\* 登录操作

\*/

@RequestMapping(value="/login",method=RequestMethod.GET)

public String login(String account,String password,String type,HttpSession session){

System.out.println("输入的账号和姓名："+account+","+password);

List<String> listNav = new ArrayList<String>();

listNav.add("首页");//设置面包屑导航

session.setAttribute("listNav", listNav);

if( type.equals("Manager") ){//如果类型等于管理员

//通过账号得到一个教师对象，判断 (账号是否为空)-和-(密码是否正确)

Manager manager = managerManager.getManagerLogin(account);

if( manager != null ){ //如果不等于空说明账号正确

if( manager.getM\_password().equals(password) ){

session.setAttribute("manager",manager);

return "manager/manager";

}else{

session.setAttribute("errMsg","密码错误！！");

return "login";

}

}else{//账号错误

session.setAttribute("errMsg","账号或密码错误！！"); //账号不存在！

return "login";

}

}else if( type.equals("Teacher") ){//如果类型等于教师

//通过账号得到一个教师对象，判断 (账号是否为空)-和-(密码是否正确)

Teacher teacher = teacherManager.getTeacherLogin(account);

if( teacher != null ){ //如果不等于空说明账号正确

if( teacher.getT\_password().equals(password) ){

session.setAttribute("teacher",teacher);

return "teacher/teacher";

}else{

session.setAttribute("errMsg","密码错误！！");

return "login";

}

}else{//账号错误

session.setAttribute("errMsg","账号或密码错误！！");

return "login";

}

}else if( type.equals("Student") ){//如果类型等于学生

//通过账号得到一个学生对象，判断 (账号是否为空)-和-(密码是否正确)

Student student = studentManager.getStudentLogin(account);

if( student != null ){ //如果不等于空说明账号正确

if( student.getS\_password().equals(password) ){

session.setAttribute("student",student);

return "student/student";

}else{

session.setAttribute("errMsg","密码错误！");

return "login";

}

}

else{//账号错误

session.setAttribute("errMsg","账号或密码错误！");

return "login";

}

}else

session.setAttribute("errMsg","未选择类型错误！！");

return "login";

}

/\*\*

\* 注销操作

\*/

@RequestMapping("/logout")

public String logout(HttpSession session){

//session.invalidate();

return "login";

}

}

### 4.2.3 业务逻辑层

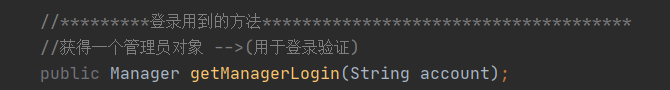
登录模块的业务逻辑层主要完成对用户登录逻辑的判定，同时调用登录模块的业务逻辑接口。比如：用户到登录是输入的用户名是否存在、密码是否正确，同时在对用户的身份进行判定。登录模块业务逻辑层列表如图4-5所示。

表 4-4登入模块业务逻辑层列表

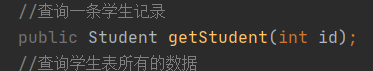
|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Dao | 调用说明 | 出口 |
| 登录 | ImangerDao.java  MangerDao.java  IstudentDao.java  studentDao.java  IteacherDao.java  teacherDao.java |  | 返回给HomeController |

在登入模块的业务逻辑层是调用了公用的IUserService接口，同时在实现该接口。

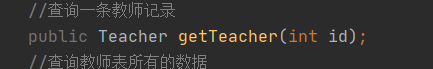
ImangerDao.java接口主要方法:



IstudentDao.java接口主要方法:



IteacherDao.java接口主要方法:



### 4.2.4 数据持久层

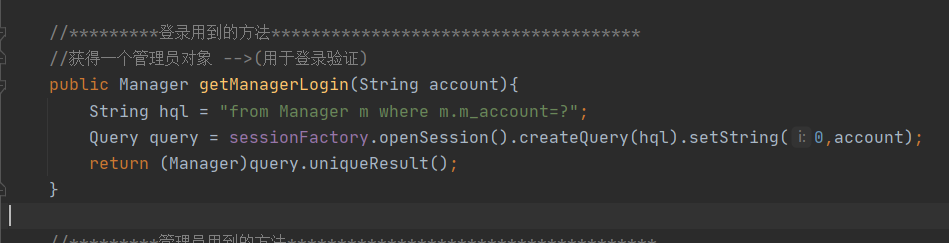
登录模块的是数据持久层能对用户数据进行增删改查。登入模块数据持久层列表见表4-5所示。

表 4-5 登入模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 登录 | ImangerDao.java  MangerDao.java  IstudentDao.java  studentDao.java  IteacherDao.java  teacherDao.java |  | 对用户信息进行增删改查操作 |

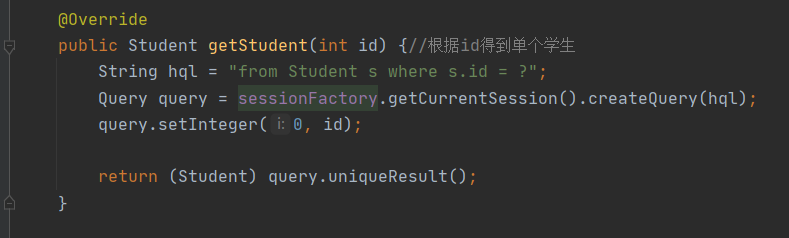
IMangerDao.java定义了用户信息进行增、删、改、查的接口：

MangerDao.java主要实现属性与方法:



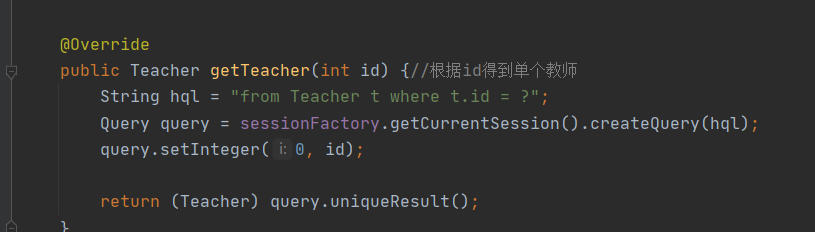
IStudentDao.java定义了用户信息进行增、删、改、查的接口：

StudentDao.java主要实现属性与方法:



ITeacherDao.java定义了用户信息进行增、删、改、查的接口：

TeacherDao.java主要实现属性与方法:



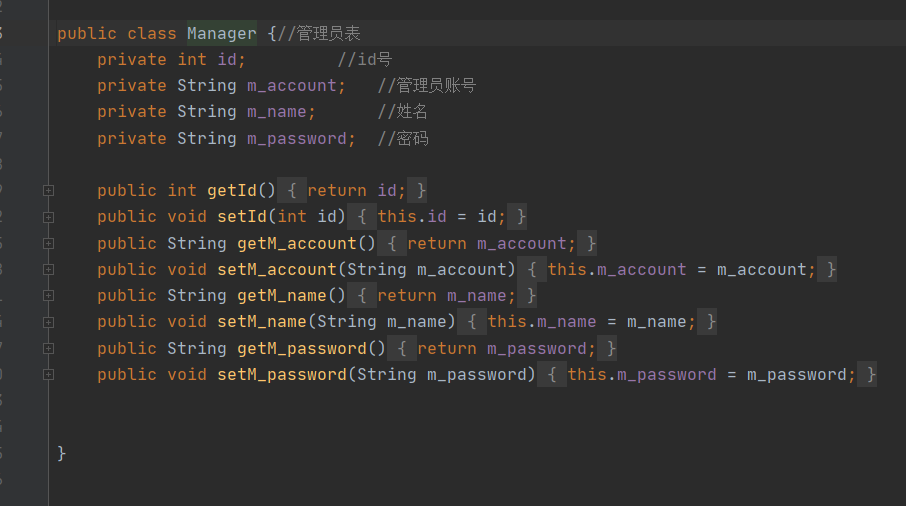
### 4.2.5 域模型层

登录模块用到域模型层中，在涉及到用户信息查询等操作时，就会调用到该模型，登录模块域模型层列表如表4-6所示。

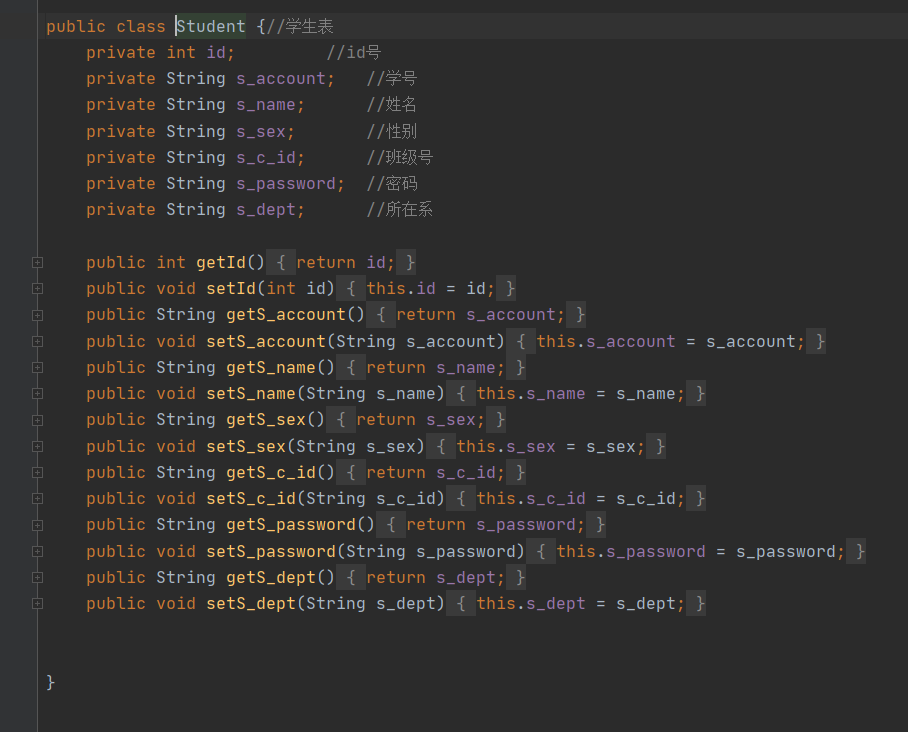
表 4-6登入模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Manager.java  Student.java  Teacher.java | 对用户信息的增、删、改、查操作 |

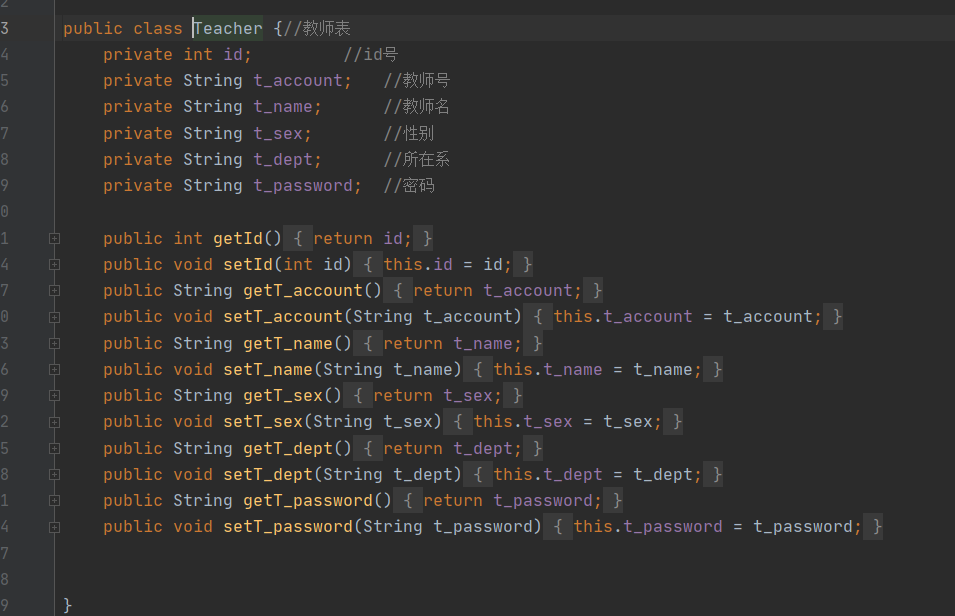
Manager.java:主要属性与方法



Student.java:主要属性与方法



Teacher.java：主要属性与方法



## 4.2、学生作业提交模块模块

在学生作业提交时，系统内部的相应响应操作示意图如图4-2所示。

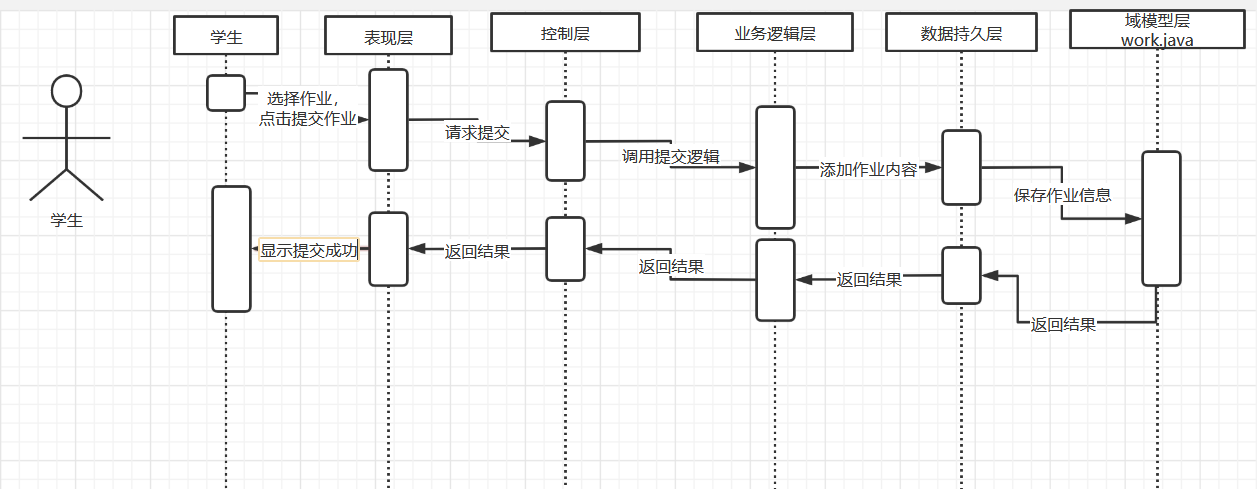
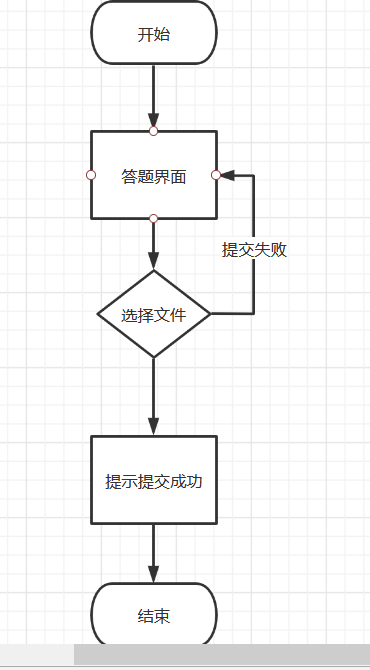


图 4-3 系统响应示意图

### 4.2.1 表现层



学生作业提交模块的表现层主要完成不同学生的作业提交功能，显示页面信息。表现层对应的JSP页面列表见表4-7-所示。

表 4-7 学生作业提交模块表现层JSP列表

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 学生作业提交页面 | S\_uploadwork.jsp | 提供作业提交功能 |

### 4.2.2 控制层

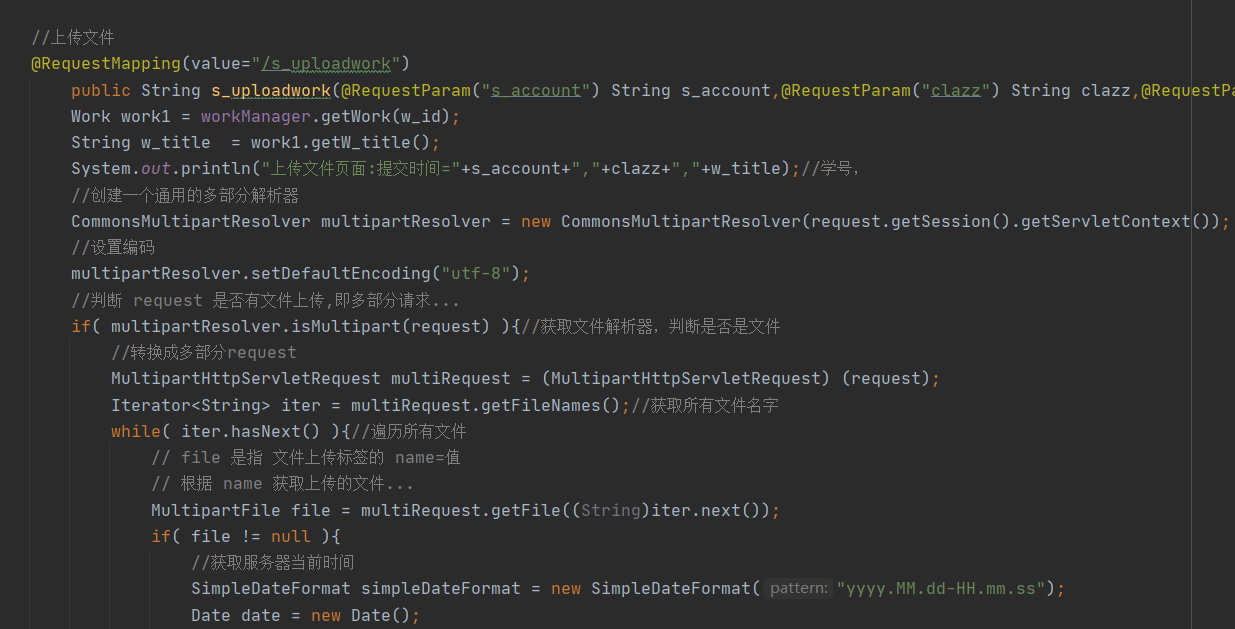
学生作业提交模块的控制层负责调用学生作业提交的业务逻辑接口，将学生名与作业等学生关键信息传递到业务逻辑层进行判定。等到业务逻辑处理完成之后，将来自业务逻辑层的相应信息传到表现层，并决定显示页面。

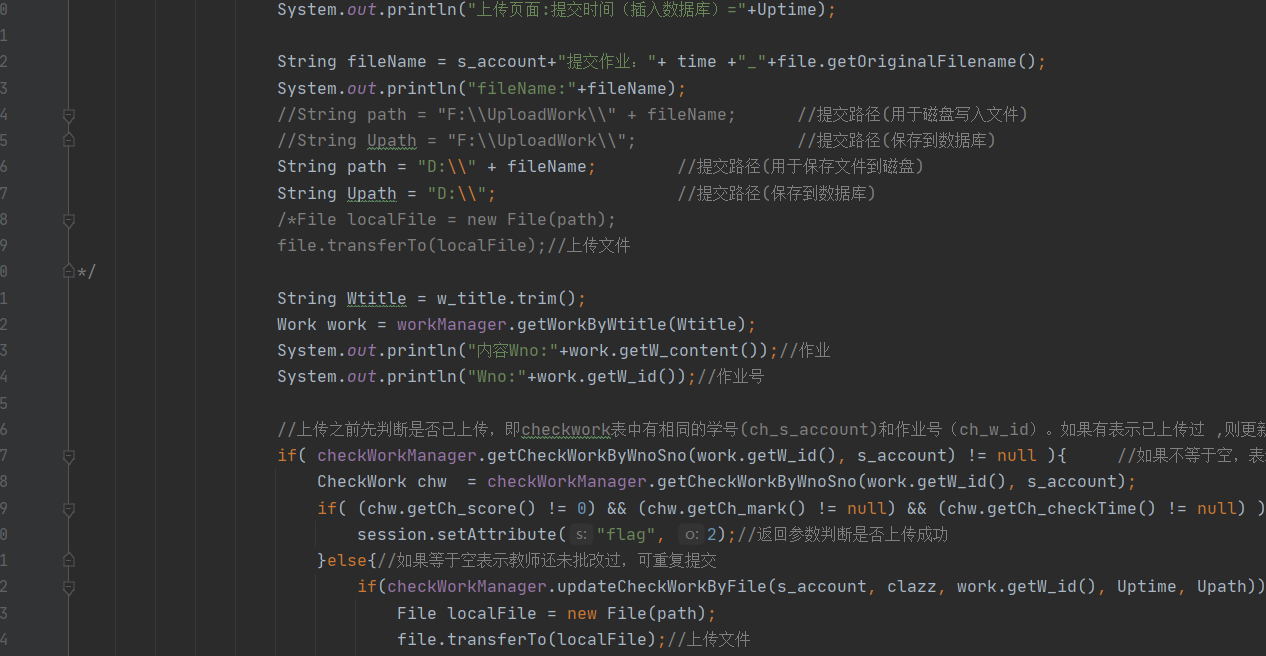
层列表见表4-8所示。

表 4-8学生作业提交模块模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 学生作业提交 | Studentcontroller.java | SUCCESS | S\_uploadwork.jsp—提交成功，显示提示窗口。 |
| ERROR | S\_uploadwork.jsp---提交失败，显示提示窗口。 |

在控制层中Studentcontroller.java的描述如下所示：







### 4.2.3 业务逻辑层

学生作业提交模块的业务逻辑层主要完成对学生作业提交逻辑的判定，业务逻辑接口。学生作业提交模块业务逻辑层列表如图4-12所示。

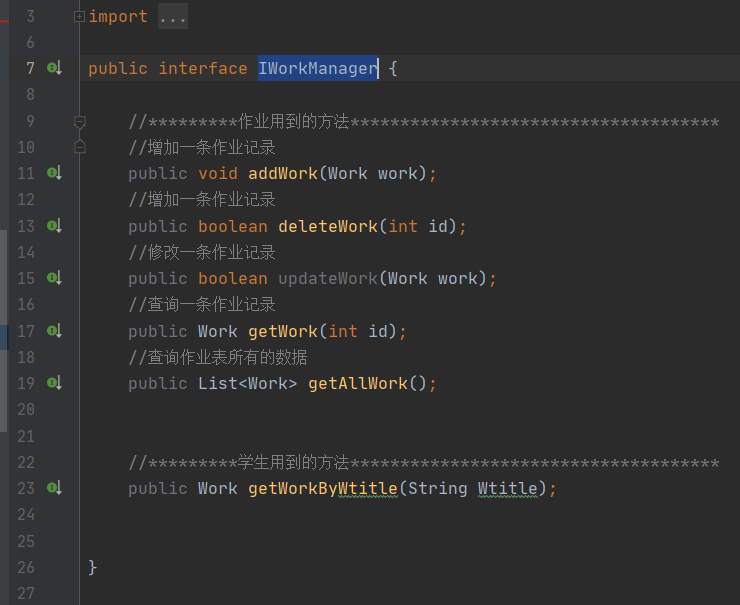
表 4-9登入模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 学生作业提交 | IWorkManage.java  WorkManage.java | 调用 work.Java | 返回给Studentcontroller |

在学生作业提交的业务逻辑层是调用了的IWorkManage.java

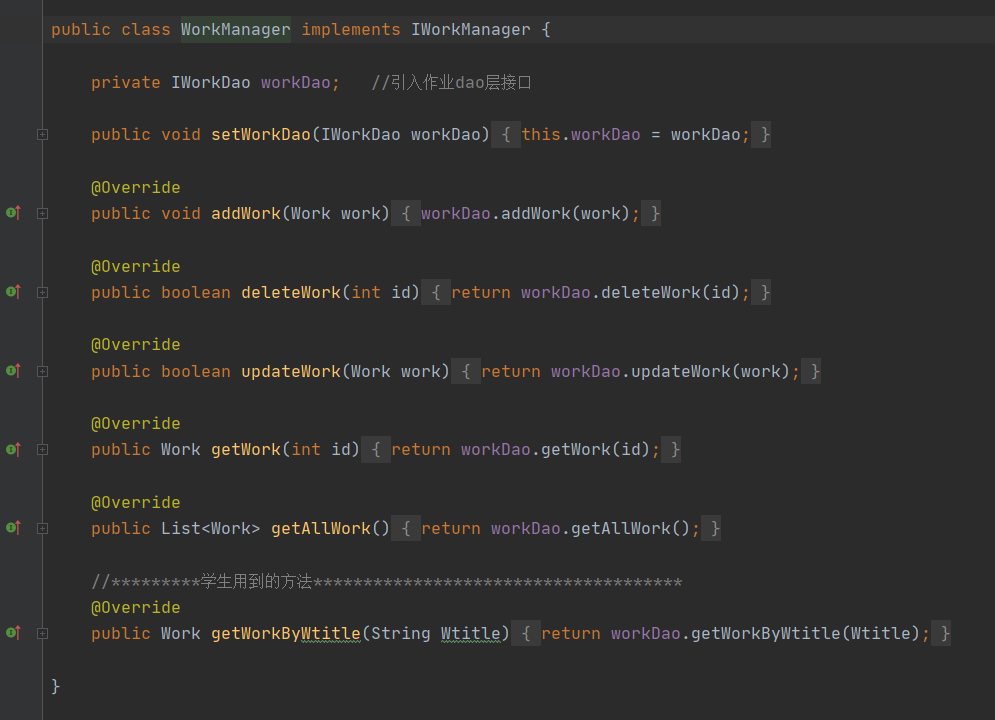
接口，同时在实现该接口。

IWorkManage.java接口主要方法：



WorkManage.java实现了IWorkManage.java接口， 同时需要调用数据持久层的WorkManageDao.Java，利用@Resource来实现对数据持久层接口的调用。

WorkManage.java主要实现属性与方法:



### 4.2.4 数据持久层

学生作业提交模块的是数据持久层能对学生作业提交数据进行增改查。学生作业提交模块数据持久层列表见表4-10所示。

表 4-10 学生作业提交模块数据持久层列表

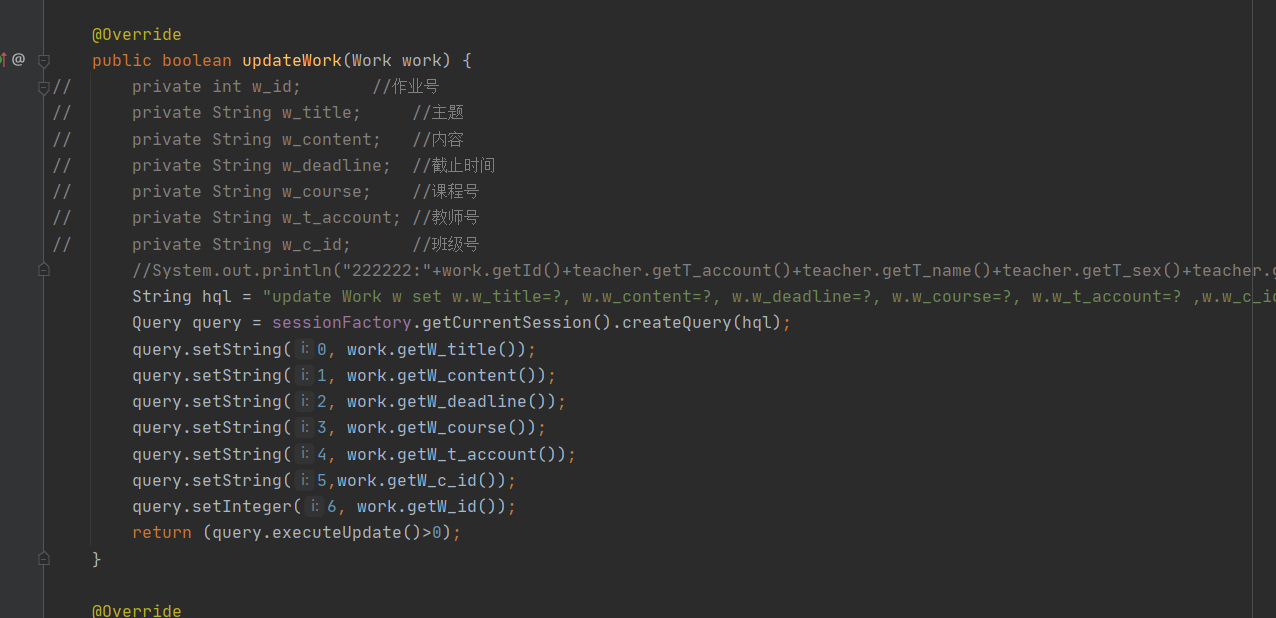
|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 学生作业提交 | IWorkDao.java  WorkDao.java | 调用 work.Java | 对作业信息进行增改查操作 |

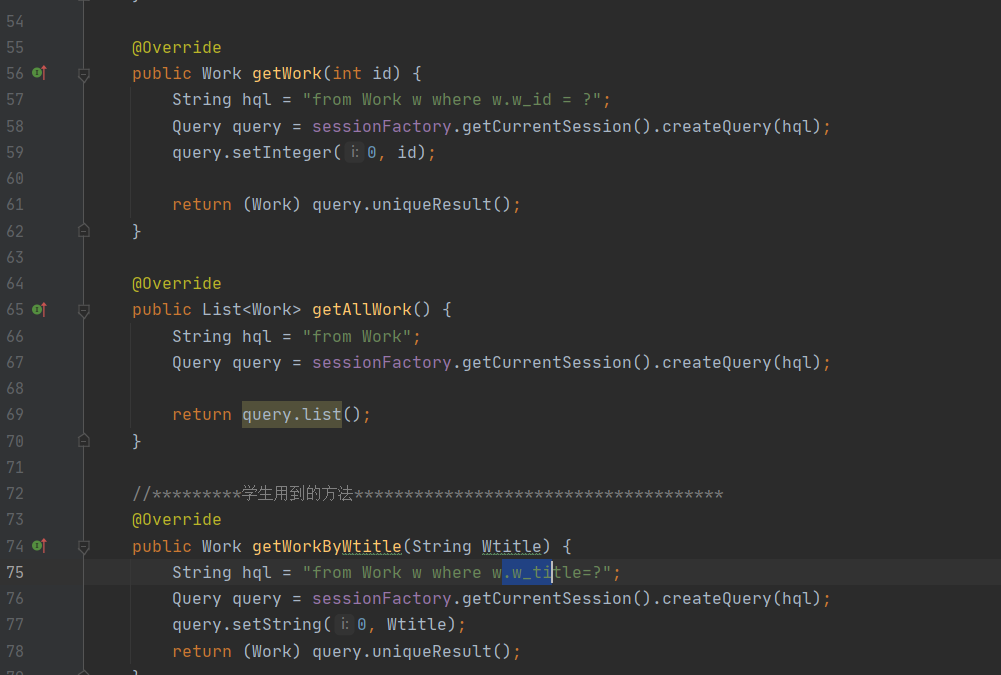
IWorkDao.java定义了用户信息进行增、改、查的接口：



WorkDao.java是IWorkDao.java接口的实现：







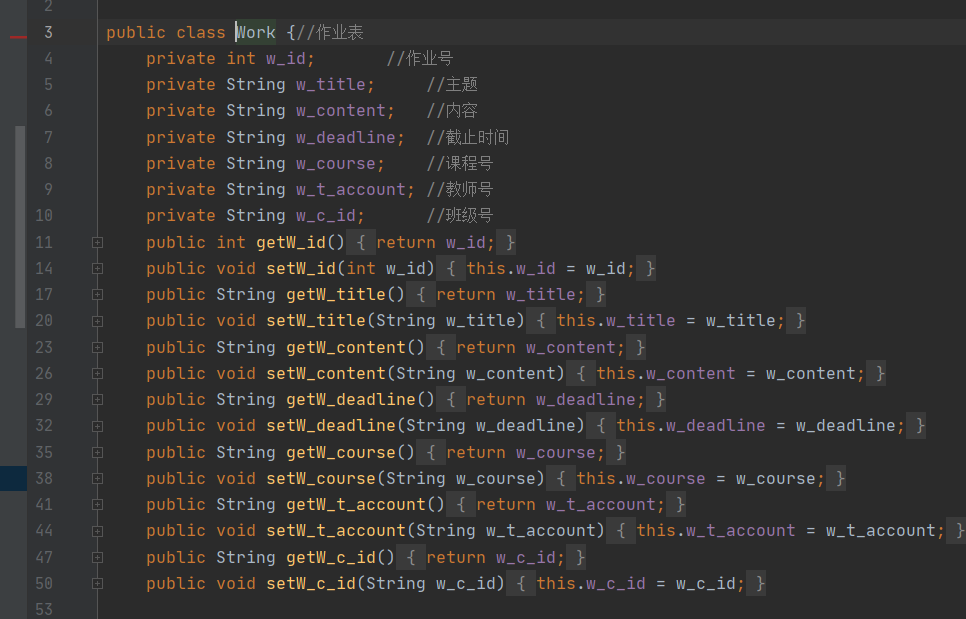
### 4.2.5 域模型层

学生作业提交模块用到域模型层中Work.java，Work.java是一个公用域模型，在涉及到作业的增查改等操作时，就会调用到该模型，学生作业提交模块域模型层列表如表4-7所示。

表 4-11学生作业提交模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Work.java | 对作业信息的增、改、查操作 |

Work.java主要属性与方法：



## 4.3、教师下载作业提交模块模块

在教师下载作业模块时，系统内部的相应响应操作示意图如图4-6所示。

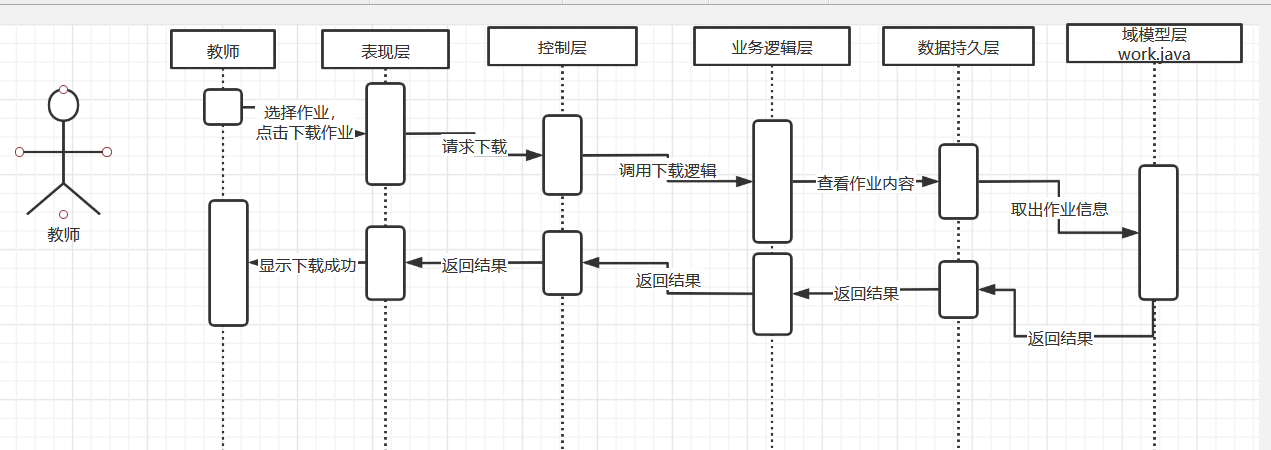


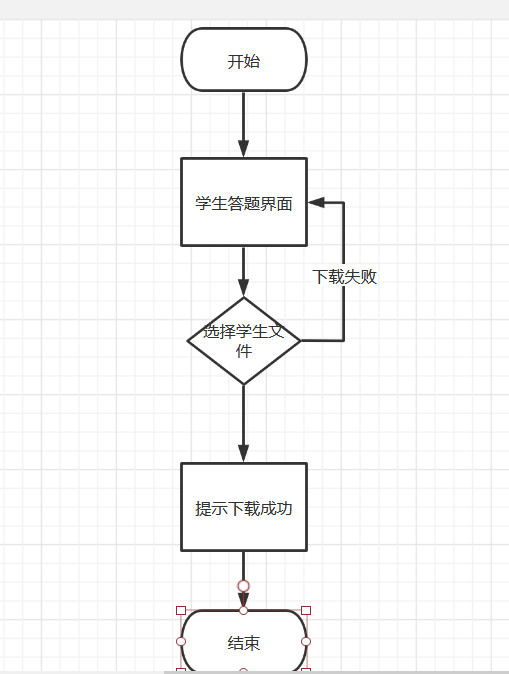
图 4-6 系统响应示意图

### 4.2.1 表现层

教师下载作业模块的表现层主要完成不老师教师下载作业功能，显示页面信息。表现层对应的JSP页面列表见表4-12-所示。

表 4-12 教师下载作业模块表现层JSP列表

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 教师作业下载页面 | S\_allwork.jsp | 提供教师下载作业功能 |



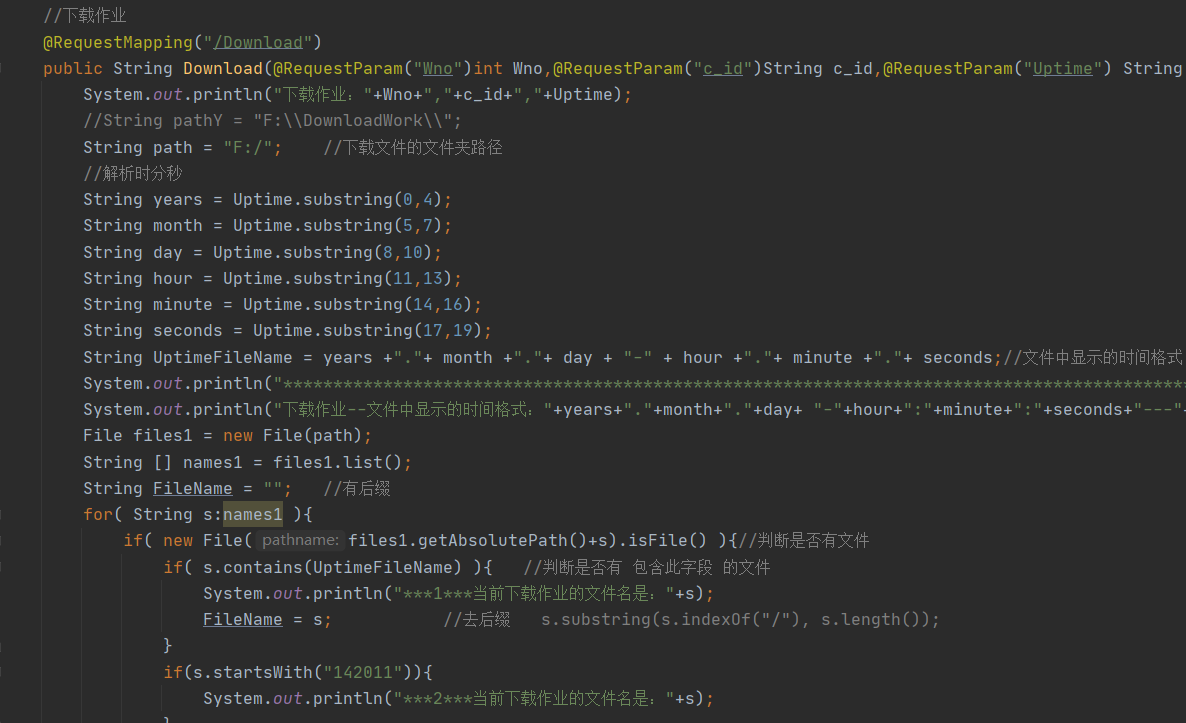
### 4.2.2 控制层

教师下载作业模块的控制层负责调用教师下载作业的业务逻辑接口，将学生关键信息传递到业务逻辑层进行判定并提供老师下载。等到业务逻辑处理完成之后，将来自业务逻辑层的相应信息传到表现层，并决定显示页面。

表 4-14 教师下载作业模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 教师下载作业 | Teachercontroller.java | SUCCESS | T\_allwork.jsp—下载成功，显示提示窗口。 |
| ERROR | T\_allwork.jsp.jsp---下载失败，显示提示窗口。 |

在控制层中Teachercontroller.java的描述如下所示：





### 4.2.3 业务逻辑层

教师下载作业模块的业务逻辑层主要完成对教师下载作业逻辑的判定，业务逻辑接口。教师下载作业模块业务逻辑层列表如图4-15所示。

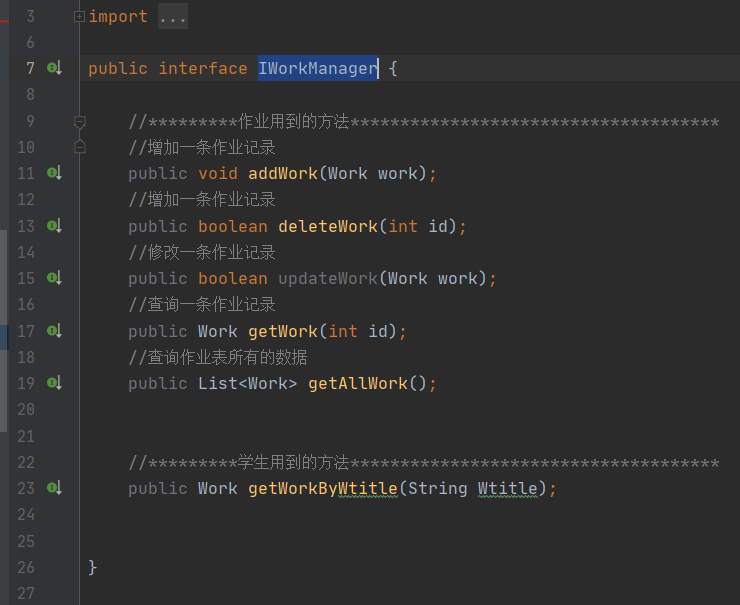
表 4-15教师下载作业模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 教师下载作业 | IWorkManage.java  WorkManage.java | 调用 work.Java | 返回给Teachercontroller |

在教师下载作业的业务逻辑层是调用了的IWorkManage.java

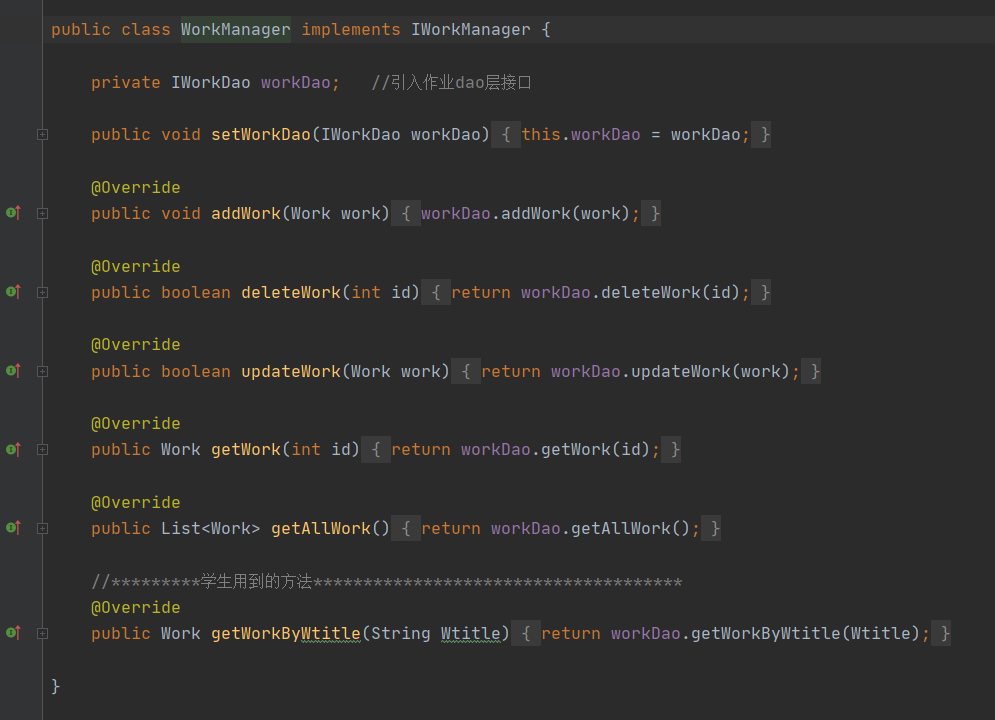
接口，同时在实现该接口。

IWorkManage.java接口主要方法：



WorkManage.java实现了IWorkManage.java接口， 同时需要调用数据持久层的WorkManageDao.Java，利用@Resource来实现对数据持久层接口的调用。

WorkManage.java主要实现属性与方法:



### 4.2.4 数据持久层

教师下载作业模块的是数据持久层能对学生作业数据下载。教师下载作业模块数据持久层列表见表4-16所示。

表 4-16 教师下载作业模块数据持久层列表

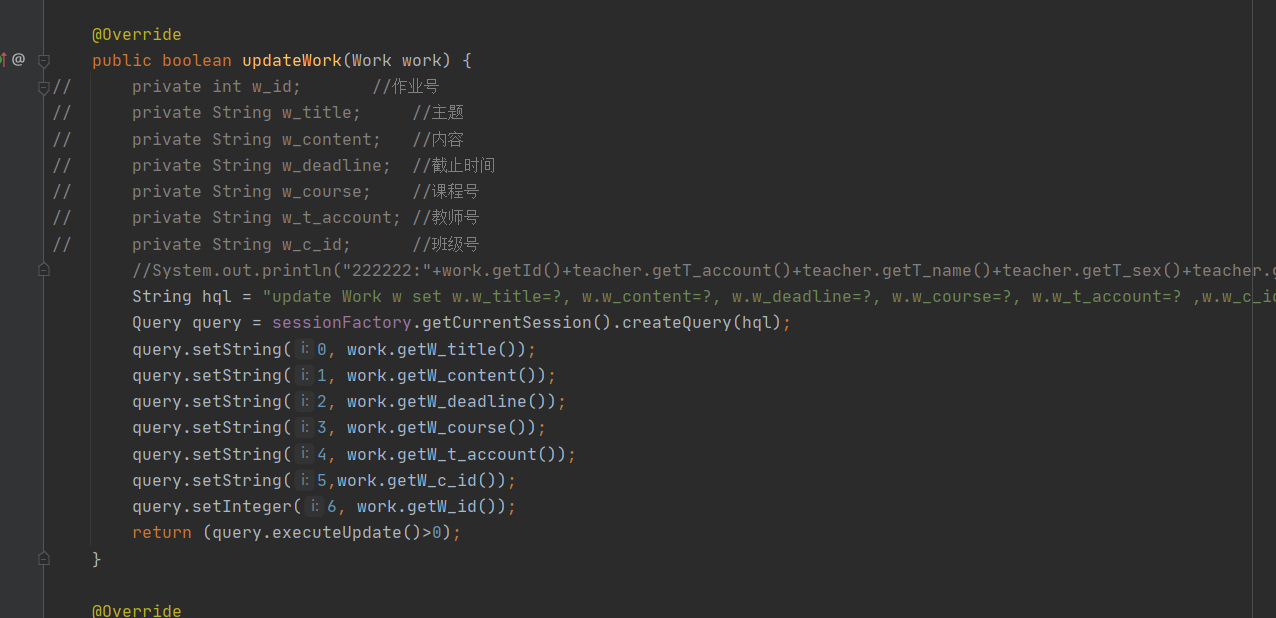
|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 教师下载作业 | IWorkDao.java  WorkDao.java | 调用 work.Java | 对作业信息进行下载 |

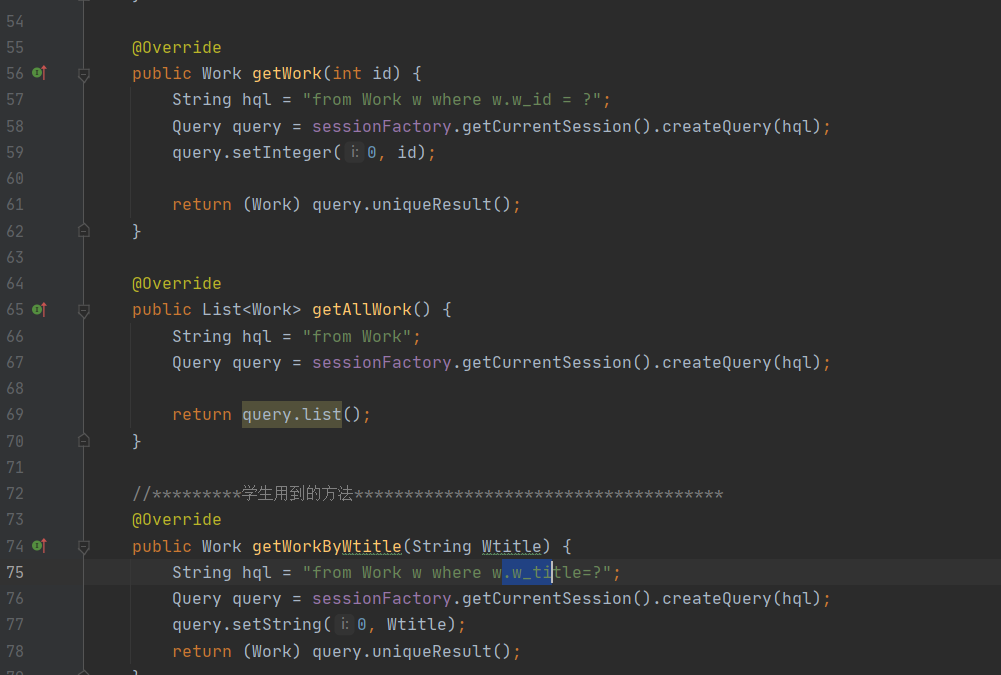
IWorkDao.java定义了用户信息进行增、改、查的接口：



WorkDao.java是IWorkDao.java接口的实现：







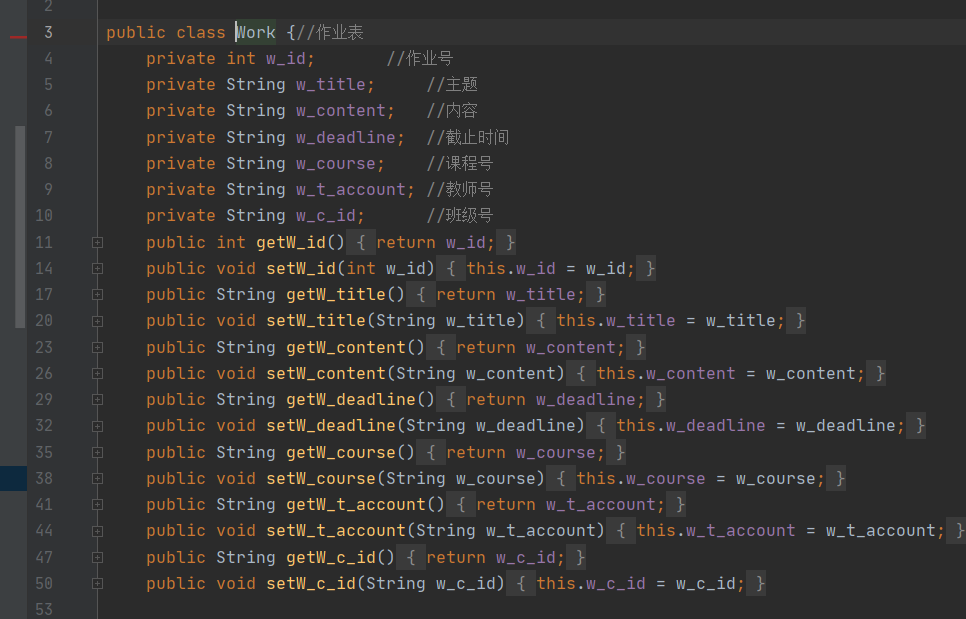
### 4.2.5 域模型层

教师下载作业模块用到域模型层中Work.java，Work.java是一个公用域模型，在涉及到作业的增查改等操作时，就会调用到该模型，教师下载作业模块域模型层列表如表4-17所示。

表 4-17登入模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Work.java | 对作业信息的增、改、查操作 |

Work.java主要属性与方法：



# 5、管理端模块详细设计

管理端模块主要分为管理员管理模块，教师管理模块，学生管理模块

## 5.1、管理员模块详细设计

在管理员管理模块，系统内部的相应操作示意图5.1.1

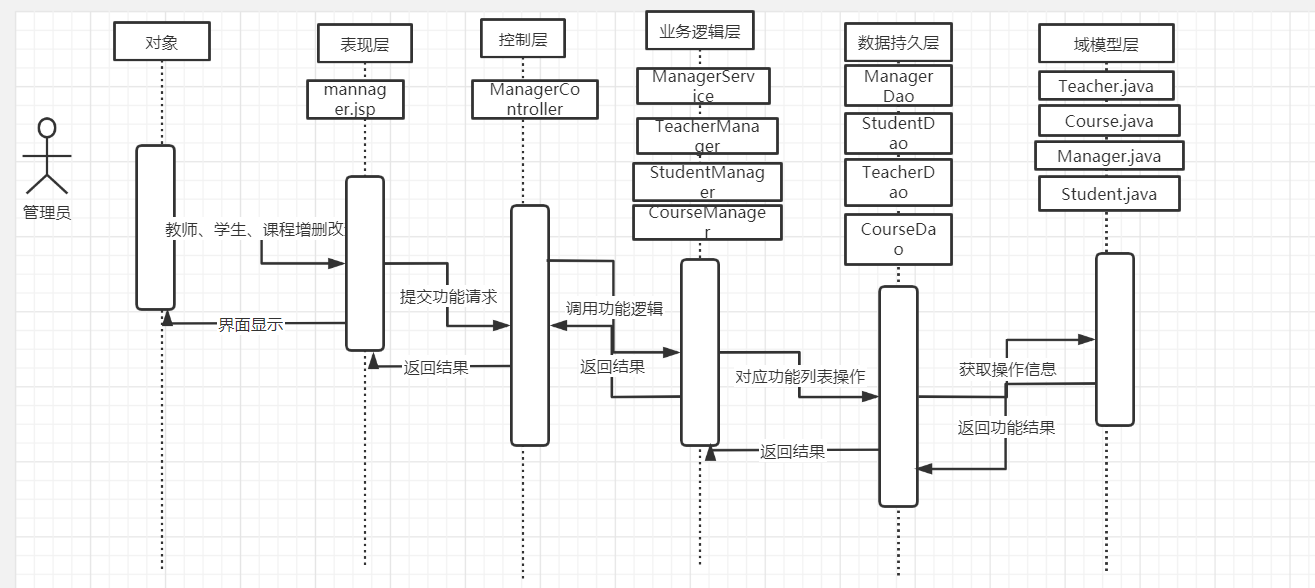


图5.1

### 5.1.1 表现层

管理员管理的表现层主要完成管理员不同的操作功能，管理页面需要管理员提交相应的操作及其需要的信息，确认后页面给出相应的响应，提示操作结果，表现层对应的jsp页面见表

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 管理员页面 | mannager.jsp | 管理员对（学生、个人、教师）进行各种操作功能 |

流程图如图5.1.2

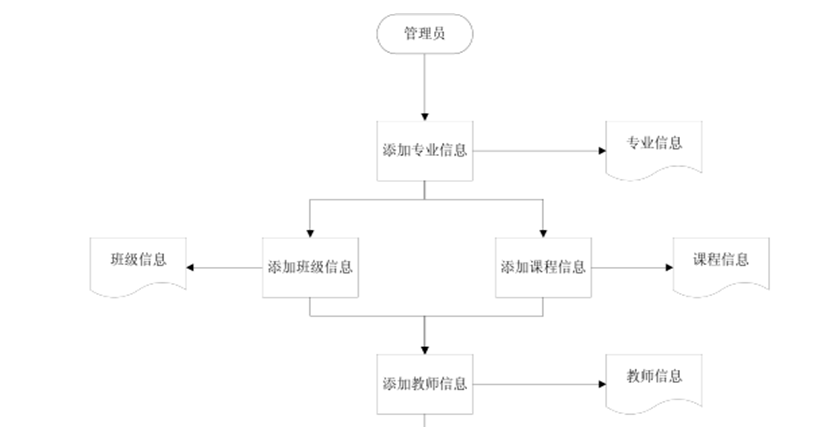


图5.1.2

### 5.1.2 控制层

管理员管理模块的控制层负责接受来自manager.jsp的管理员功能输入，同时调用管理员模块的业务逻辑接口，将管理员提交的请求传递到业务逻辑层进行判定。等到业务逻辑处理完成之后，将来自业务逻辑层的相应信息传到表现层，并决定显示页面。管理员管理模块控制层列表见表5.1.3所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 管理员管理 | ManagerController.java | SUCCESS | m\_coursemanage.jsp—对课程管理，显示提示窗口。  m\_studentmanage.jsp—对学生管理，显示提示窗口  m\_teachermannage.jsp—对老师管理，显示提示窗来  m\_personalseting.jsp—管理个人信息，显示提示窗口 |
| ERROR | manager.jsp---操作失败，显示提示窗口。 |

在控制层中ManagerController.java的描述如下所示：

@Controller

@RequestMapping("/manager")

public class ManagerController {

@Resource(name="managerManager")

private IManagerService managerManager; //引入管理员services层接口

@Resource(name="teacherManager")

private ITeacherManager teacherManager; //引入教师services层接口

@Resource(name="studentManager")

private IStudentManager studentManager; //引入学生services层接口

@Resource(name="courseManager")

private ICourseManager courseManager; //引入课程services层接口

/\*-------------------------------framework(导航栏)页面操作---------------------------------------------------------------------------------------\*\*/

/\*

\* 管理员页面(导航栏)------教师总览-------页面（内嵌）

\*/

@RequestMapping("/m\_teachermanage")

public String m\_teachermanage(Model model,HttpSession session){

//设置面包屑导航

List<String> listNav = (List)session.getAttribute("listNav");

listNav.add("教师总览");

session.setAttribute("listNav", listNav);

List<Teacher> teacher1 = teacherManager.getAllTeacher();

System.out.println("teacher教师总览页面:"+teacher1.get(0).getT\_account()+","+teacher1.get(0).getT\_name()+","+teacher1.get(0).getT\_password());

model.addAttribute("teacher", teacher1);

return "manager/m\_teachermanage";

}

/\*

\* 管理员页面(导航栏)------教师添加------按钮(导航栏跳转)

\*/

@RequestMapping("/m\_addteacher")

public String m\_addteacher(){

return "manager/m\_addteacher";

}

/\*

\* 管理员页面(导航栏)------学生总览-------页面（内嵌）

\*/

@RequestMapping("/m\_studentmanage")

public String m\_studentmanage(Model model){

List<Student> student1 = studentManager.getAllStudent();

System.out.println("student:"+student1.get(0).getS\_account()+","+student1.get(0).getS\_name()+","+student1.get(0).getS\_password());

model.addAttribute("student", student1);

return "manager/m\_studentmanage";

}

/\*

\* 管理员页面(导航栏)------学生添加------按钮(导航栏跳转)

\*/

@RequestMapping("/m\_addstudent")

public String m\_addstudent(){

return "manager/m\_addstudent";

}

/\*

\* 管理员页面(导航栏)------课程总览-------页面（内嵌）

\*/

@RequestMapping("/m\_coursemanage")

public String m\_coursemanage(Model model){

List<Course> course1 = courseManager.getAllCourse();

System.out.println("管理员页面----课程总览:"+course1.get(0).getC\_id()+","+course1.get(0).getC\_name()+","+course1.get(0).getC\_credit());

model.addAttribute("course", course1);

return "manager/m\_coursemanage";

}

/\*

\* 管理员页面(导航栏)------课程添加------按钮(导航栏跳转)

\*/

@RequestMapping("/m\_addcourse")

public String m\_addcourse(){

return "manager/m\_addcourse";

}

/\*

\* 管理员页面(导航栏)------个人设置------按钮(导航栏跳转)

\*/

@RequestMapping("/m\_personalseting")

public String m\_personalseting(Model model){//@RequestParam("m\_name") String m\_name,

/\*System.out.println("管理员页面---个人设置---按钮(导航栏跳转)----m\_name:"+m\_name);

Manager manager = managerManager.getManagerByName(m\_name);\*/

//System.out.println("点击个人设置--->管理员信息:"+manager.getId()+","+manager.getM\_name()+","+manager.getM\_account()+","+manager.getM\_password());

//model.addAttribute("m", manager);

return "manager/m\_personalseting";

}

/\*---------------------------------------------iframe内嵌页面操作-----------------------------------------------------------------------\*\*\*/

/\*

\* 管理员页面（内嵌页面）------添加教师

\*/

@RequestMapping("/addTeacher")

public String addTeacher(Teacher teacher,Model model){

System.out.println("管理员页面（内嵌页面）------添加教师:"+teacher.getT\_account()+","+teacher.getT\_name()+","+teacher.getT\_password());

teacherManager.addTeacher(teacher);

model.addAttribute("teacher", teacher);

return "redirect:/manager/m\_teachermanage"; //重定向到-->教师总览页面（内嵌）

}

/\*

\* 管理员页面（内嵌页面）------添加学生

\*/

@RequestMapping("/addStudent")

public String addStudent(Student student,Model model){

System.out.println("管理员页面（内嵌页面）------添加学生:"+student.getS\_account()+","+student.getS\_name()+","+student.getS\_password());

studentManager.addStudent(student);

model.addAttribute("student", student);

return "redirect:/manager/m\_studentmanage"; //重定向到-->学生总览页面（内嵌）

}

/\*

\* 管理员页面（内嵌页面）------添加课程

\*/

@RequestMapping("/addCourse")

public String addCourse(Course course,Model model){

System.out.println("管理员页面（内嵌页面）------添加课程:"+course.getC\_id()+","+course.getC\_name()+","+course.getC\_credit());

courseManager.addCourse(course);

model.addAttribute("course", course);

return "redirect:/manager/m\_coursemanage"; //重定向到-->课程总览页面（内嵌）

}

/\*

\* 管理员页面（内嵌页面）------得到教师表

\*/

@RequestMapping("/getAllTeacher")

public String getAllTeacher(Model model){

List<Teacher> teacher = teacherManager.getAllTeacher();

model.addAttribute("teacher", teacher);

return "/manager/manager";

}

/\*

\* 管理员页面（内嵌页面）------删除教师

\*/

@RequestMapping("/delTeacher")

public void delTeacher(String id,HttpServletResponse response){

String result = "{\"result\":\"error\"}";

if( teacherManager.deleteTeacher(id) ){

result = "{\"result\":\"success\"}";

}

response.setContentType("application/json");

PrintWriter out = null;

try {

out = response.getWriter();

out.write(result);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

/\*

\* 管理员页面（内嵌页面）------删除学生

\*/

@RequestMapping("/delStudent")

public void delStudent(String id,HttpServletResponse response){

String result = "{\"result\":\"error\"}";

if( studentManager.deleteStudent(id) ){

result = "{\"result\":\"success\"}";

}

response.setContentType("application/json");

PrintWriter out = null;

try {

out = response.getWriter();

out.write(result);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

/\*

\* 管理员页面（内嵌页面）------删除课程

\*/

@RequestMapping("/delCourse")

public void delCourse(int id,HttpServletResponse response){

String result = "{\"result\":\"error\"}";

if( courseManager.deleteCourse(id) ){

result = "{\"result\":\"success\"}";

}

response.setContentType("application/json");

PrintWriter out = null;

try {

out = response.getWriter();

out.write(result);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

/\*

\* 管理员页面（内嵌页面）------(点击修改教师按钮)-->挑转到教师修改页面

\*/

@RequestMapping("/updateTeacher\_jsp")

public String updateTeacher\_jsp(int id ,Model model){

System.out.println("教师："+id);

Teacher teacher = teacherManager.getTeacher(id);

model.addAttribute("t", teacher);

return "manager/m\_updateteacher";

}

/\*

\* 管理员页面（内嵌页面）------修改教师信息

\*/

@RequestMapping("/updateTeacher")

public String updateTeacher(Teacher teacher,Model model){

System.out.println("管理员页面（内嵌页面）------修改教师信息:"+teacher.getId()+","+teacher.getT\_account()+","+teacher.getT\_name()+","+teacher.getT\_sex()+","+teacher.getT\_dept()+","+teacher.getT\_password());

if( teacherManager.updateTeacher(teacher) ){

teacher = teacherManager.getTeacher(teacher.getId());

model.addAttribute("teacher", teacher);

return "redirect:/manager/m\_teachermanage"; //修改成功后--重定向到-->教师总览页面（内嵌）

}else{

model.addAttribute("error", "教师信息修改出错了！！");

return "redirect:/manager/m\_teachermanage"; //修改出错后--重定向到-->教师总览页面（内嵌）

}

}

/\*

\* 管理员页面（内嵌页面）------(点击修改学生按钮)-->挑转到学生修改页面

\*/

@RequestMapping("/updateStudent\_jsp")

public String updateStudent\_jsp(int id ,Model model){

System.out.println("学生："+id);

Student student = studentManager.getStudent(id);

model.addAttribute("s", student);

return "manager/m\_updatestudent";

}

/\*

\* 管理员页面（内嵌页面）------修改学生信息

\*/

@RequestMapping("/updateStudent")

public String updateStudent(Student student,Model model){

System.out.println("管理员页面（内嵌页面）------修改学生信息:"+student.getId()+","+student.getS\_account()+","+student.getS\_name()+","+student.getS\_sex()+","+student.getS\_c\_id()+","+student.getS\_password()+","+student.getS\_dept());

if( studentManager.updateStudent(student) ){

student = studentManager.getStudent(student.getId());

model.addAttribute("student", student);

return "redirect:/manager/m\_studentmanage"; //修改成功后--重定向到-->学生总览页面（内嵌）

}else{

model.addAttribute("error", "学生信息修改出错了！！");

return "redirect:/manager/m\_studentmanage"; //修改出错后--重定向到-->学生总览页面（内嵌）

}

}

/\*

\* 管理员页面（内嵌页面）------(点击修改课程按钮)-->挑转到课程修改页面

\*/

@RequestMapping("/updateCourse\_jsp")

public String updateCourse\_jsp(int id ,Model model){

System.out.println("修改课程的课程号："+id);

Course course = courseManager.getCourse(id);

System.out.println("修改课程的课程名："+course.getC\_name());

model.addAttribute("c", course);

return "manager/m\_updatecourse";

}

/\*

\* 管理员页面（内嵌页面）------修改课程信息

\*/

@RequestMapping("/updateCourse")

public String updateCourse(Course course,Model model){

System.out.println("管理员页面（内嵌页面）------修改课程信息:"+course.getC\_id()+","+course.getC\_name()+","+course.getC\_credit());

if( courseManager.updateCourse(course) ){

course = courseManager.getCourse(course.getC\_id());

model.addAttribute("course", course);

return "redirect:/manager/m\_coursemanage"; //修改成功后--重定向到-->课程总览页面（内嵌）

}else{

model.addAttribute("error", "课程信息修改出错了！！");

return "redirect:/manager/m\_coursemanage"; //修改出错后--重定向到-->课程总览页面（内嵌）

}

}

/\*

\* 管理员页面（内嵌页面）------个人设置-->修改账号-------（信息修改）

\*/

@RequestMapping("/updateManagerAccount")

public String updateManagerAccount(int id, String account, HttpSession request){

System.out.println("管理员个人设置修改账号:"+id+","+account);

if( managerManager.updateManagerAccount(id , account) ){

Manager manager = managerManager.getManagerById(id);

request.setAttribute("manager", manager);

request.setAttribute("msg", "账号修改成功");

/\*String name = manager.getM\_name();

model.addAttribute("m\_name", manager.getM\_name());\*/

//return "redirect:/manager/m\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

//model.addAttribute("msg", "账号修改成功");

return "redirect:/manager/m\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

}else{

request.setAttribute("msg", "管理员账号修改出错了！！");

return "redirect:/manager/m\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

}

}

/\*

\* 管理员页面（内嵌页面）------个人设置-->修改姓名-------（信息修改）

\*/

@RequestMapping("/updateManagerName")

public String updateManagerName(int id, String name, HttpSession request){

System.out.println("管理员个人设置修改姓名:"+id+","+name);

if( managerManager.updateManagerName(id , name) ){

Manager manager = managerManager.getManagerByName(name);

/\*model.addAttribute("m\_name", manager.getM\_name());

return "redirect:/manager/m\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

\*/ request.setAttribute("manager", manager);

request.setAttribute("msg", "姓名修改成功");

return "redirect:/manager/m\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

}else{

/\*model.addAttribute("error", "管理员姓名修改出错了！！");

return "redirect:/manager/m\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

\*/ request.setAttribute("msg", "管理员姓名修改出错了！！");

return "redirect:/manager/m\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

}

}

/\*

\* 管理员页面（内嵌页面）------个人设置-->修改密码-------（信息修改）

\*/

@RequestMapping("/updateManagerPassword")

public String updateManagerPassword(int id, String password, HttpSession request){ //RedirectAttributes model--->重定向传递参数

System.out.println("管理员个人设置修改密码:"+id+","+password);

if( managerManager.updateManagerPassword(id, password) ){

Manager manager = managerManager.getManagerById(id);

/\*model.addAttribute("m\_name", manager.getM\_name());

return "redirect:/manager/m\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

\*/ request.setAttribute("manager", manager);

request.setAttribute("msg", "密码修改成功");

return "redirect:/manager/m\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

}else{

/\*model.addAttribute("error", "管理员密码修改出错了！！");

return "redirect:/manager/m\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

\*/ request.setAttribute("msg", "管理员密码修改出错了！！");

return "redirect:/manager/m\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

}

}

}

### 5.1.3 业务逻辑层

管理员管理模块的业务逻辑层主要完成对管理员操作逻辑的判定，同时调用管理模块的业务逻辑接口。登录模块业务逻辑层列表如图5.1.3所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 管理员管理 | ManagerService.java  ImanagerService.java | 调用managerDao | 返回给Action |

ManagerService.java主要代码

public class ManagerService implements IManagerService{

private IManagerDao managerDao; //引入学生dao层接口

public void setManagerDao(IManagerDao managerDao) {

this.managerDao = managerDao;

}

//\*\*\*\*\*\*\*\*\*登录用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Override

public Manager getManagerLogin(String account) { //获得一个管理员对象 -->(用于登录验证)

return managerDao.getManagerLogin(account);

}

//\*\*\*\*\*\*\*\*\*管理员用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Override

public List<Manager> getAllManager() { //查询管理员表所有的数据

return managerDao.getAllManager();

}

@Override

public Manager getManagerByName(String name) { //查询一条管理员记录----通过名字

return managerDao.getManagerByName(name);

}

@Override

public Manager getManagerById(int id) { //查询一条管理员记录----通过id

return managerDao.getManagerById(id);

}

@Override

public boolean updateManagerAccount(int id, String account) { //修改管理员账号

return managerDao.updateManagerAccount(id, account);

}

@Override

public boolean updateManagerName(int id, String name) { //修改管理员姓名

return managerDao.updateManagerName(id, name);

}

@Override

public boolean updateManagerPassword(int id, String password) { //修改管理员密码

return managerDao.updateManagerPassword(id, password);

}

}

ImanagerService.java是managerservice的接口定义

public interface IManagerService {

//\*\*\*\*\*\*\*\*\*登录用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//获得一个管理员对象 -->(用于登录验证)

public Manager getManagerLogin(String account);

//\*\*\*\*\*\*\*\*\*管理员用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//查询管理员表所有的数据

public List<Manager> getAllManager();

//查询一条管理员记录----通过名字

public Manager getManagerByName(String name);

//查询一条管理员记录----通过id

public Manager getManagerById(int id);

//修改管理员账号

public boolean updateManagerAccount(int id, String account);

//修改管理员姓名

public boolean updateManagerName(int id, String name);

//修改管理员密码

public boolean updateManagerPassword(int id, String password);

}

### 5.1.4 数据持久层

管理员管理模块的是数据持久层能对用户数据进行增删改查。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 登录 | ManagerDao.java  IManagerDao.java | 调用 Student.Java  Course.java  Teacher.java  Mannage.java | 对用户信息进行增删改查操作 |

IManagerDao定义了管理员对信息进行增、删、改、查的接口：

public interface IManagerDao {

//\*\*\*\*\*\*\*\*\*登录用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//获得一个管理员对象 -->(用于登录验证)

public Manager getManagerLogin(String account);

//\*\*\*\*\*\*\*\*\*管理员用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//查询管理员表所有的数据

public List<Manager> getAllManager();

//查询一条管理员记录----通过名字

public Manager getManagerByName(String name);

//查询一条管理员记录----通过id

public Manager getManagerById(int id);

//修改管理员账号

public boolean updateManagerAccount(int id, String account);

//修改管理员姓名

public boolean updateManagerName(int id, String name);

//修改管理员密码

public boolean updateManagerPassword(int id, String password);

}

ManagerDao.java是对IManagerDao.java接口的实现

public class ManagerDao implements IManagerDao{

//引入SessionFactory变量,SessionFactory接口负责初始化Hibernate。它充当数据存储源的代理，并负责创建Session对象

private SessionFactory sessionFactory;

public void setSessionFactory(SessionFactory sessionFactory) {

this.sessionFactory = sessionFactory;

}

//\*\*\*\*\*\*\*\*\*登录用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//获得一个管理员对象 -->(用于登录验证)

public Manager getManagerLogin(String account){

String hql = "from Manager m where m.m\_account=?";

Query query = sessionFactory.openSession().createQuery(hql).setString(0,account);

return (Manager)query.uniqueResult();

}

//\*\*\*\*\*\*\*\*\*管理员用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Override

public List<Manager> getAllManager() { //查询管理员表所有的数据

String hql = "from Manager";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

return query.list();

}

@Override

public Manager getManagerByName(String name) { //查询一条管理员记录----通过名字

String hql = "from Manager m where m.m\_name = ?";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, name);

return (Manager) query.uniqueResult();

}

@Override

public Manager getManagerById(int id) { //查询一条管理员记录----通过id

String hql = "from Manager m where m.id = ?";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setInteger(0, id);

return (Manager) query.uniqueResult();

}

@Override

public boolean updateManagerAccount(int id, String account) { //修改管理员账号

String hql = "update Manager m set m.m\_account=? where m.id=? ";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, account);

query.setInteger(1, id);

return (query.executeUpdate()>0);

}

@Override

public boolean updateManagerName(int id, String name) { //修改管理员姓名

String hql = "update Manager m set m.m\_name =? where m.id=? ";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, name);

query.setInteger(1, id);

return (query.executeUpdate()>0);

}

@Override

public boolean updateManagerPassword(int id, String password) { //修改管理员密码

String hql = "update Manager m set m.m\_password=? where m.id=? ";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, password);

query.setInteger(1, id);

return (query.executeUpdate()>0);

}

}

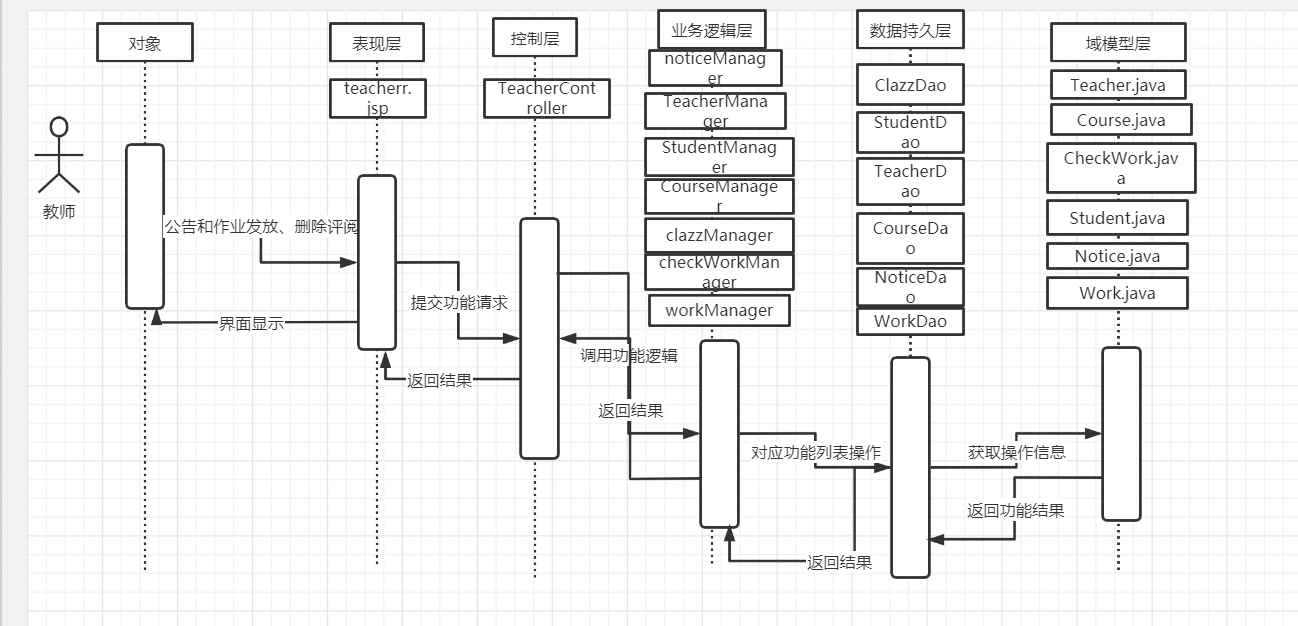
### 5.1.5 域模型层

管理员管理模块用到域模型层Student.Java,Course.java,Teacher.java,Mannage.java，在涉及到信息管理等操作时，就会调用到这些模型，管理模块域模型层列表。

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Student.Java  Course.java  Teacher.java  Mannage.java | 管理员对所需要的功能信息的增、删、改、查操作 |

## 5.2、教师管理模块详细设计

在教师管理模块，系统内部的相应操作示意图5.2.1

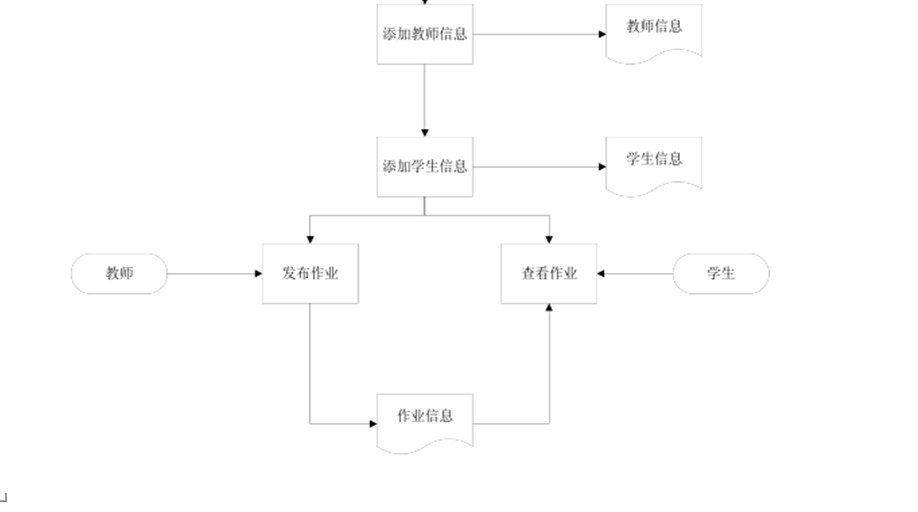


### 5.2.1 表现层

教师模块的表现层主要完成发布和批改作业，发布公告等功能，管理页面要求教师进行相应的操作，确认后页面给出响应消息，提示操作成功或失败的提示。表现层对应的JSP页面列表见表所示

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 教师管理页面 | teacher.jsp | 教师模块的表现层主要完成发布和批改作业，发布公告等功能 |

流程图：



### 5.2.2 控制层

教师管理模块的控制层主要完成发布和批改作业，发布公告等功能，同时调用管理模块的业务逻辑接口，将教师提交的操作关键信息传递到业务逻辑层进行判定。等到业务逻辑处理完成之后，将来自业务逻辑层的相应信息传到表现层，并决定显示页面。教师管理模块控制层列表见表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 教师管理 | TeacherController.java | SUCCESS | t\_allworks.jsp—对作业进行管理，显示提示窗口。  t\_allnotice.jsp—对班级公告管理，显示提示窗口  m\_teachermannage.jsp—对老师管理，显示提示窗来  t\_personalseting.jsp—管理个人信息，显示提示窗口  t\_statistics.jsp-对作业进行批改 |
| ERROR | teacher.jsp---操作失败，显示提示窗口。 |

在控制层中TeacherController.java的描述如下所示：

@Controller

@RequestMapping("/teacher")

public class TeacherController {

@Resource(name="teacherManager")

private ITeacherManager teacherManager; //引入教师services层接口

@Resource(name="workManager")

private IWorkManager workManager;

@Resource(name="noticeManager")

private INoticeManager noticeManager;

@Resource(name="clazzManager")

private IClazzManager clazzManager;

@Resource(name="courseManager")

private ICourseManager courseManager;

@Resource(name="checkWorkManager")

private ICheckWorkManager checkWorkManager;

@Resource(name="studentManager")

private IStudentManager studentManager;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*framework(导航栏)页面操作\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* 教师界面(导航栏)----我的作业---总览--页面（内嵌）

\*/

@RequestMapping("/t\_allworks")

public String t\_allworks(@RequestParam("t\_account") String t\_account,Model model){

List<Work> work = workManager.getAllWork();

List<Work> resuletWork = new ArrayList<Work>();

for(int i=0; i<work.size(); i++){

if(work.get(i).getW\_t\_account().equals(t\_account)){

resuletWork.add(work.get(i));

}

}

model.addAttribute("work", resuletWork);

return "teacher/t\_allworks";

}

/\*

\* 教师界面(导航栏)----我的作业---发布作业--页面（内嵌）

\*/

@RequestMapping("/t\_publishwork")

public String t\_publishwork(Model model){

List<Clazz> clazz = clazzManager.getAllClazz();

List<Course> course = courseManager.getAllCourse();

/\*List<Teacher> teacher1 = teacherManager.getAllTeacher();

System.out.println("teacher教师总览页面:"+teacher1.get(0).getT\_account()+","+teacher1.get(0).getT\_name()+","+teacher1.get(0).getT\_password());

model.addAttribute("teacher", teacher1);\*/

model.addAttribute("clazz", clazz);

model.addAttribute("course", course);

return "teacher/t\_publishwork";

}

/\*

\* 教师界面(导航栏)----我的公告---总览--页面（内嵌）

\*/

@RequestMapping("/t\_allnotice")

public String t\_allnotice(@RequestParam("t\_account") String t\_account, Model model){

System.out.println("t\_account:"+t\_account);

/\*List<Notice> notice = notice teacherManager.getAllNotice();

System.out.println("teacher教师总览页面:"+teacher1.get(0).getT\_account()+","+teacher1.get(0).getT\_name()+","+teacher1.get(0).getT\_password());

model.addAttribute("teacher", teacher1);\*/

List<Notice> notice = noticeManager.getNoticeByTid(t\_account);

model.addAttribute("n", notice);

return "teacher/t\_allnotice";

}

/\*

\* 教师界面(导航栏)----我的公告---发布公告--页面（内嵌）

\*/

@RequestMapping("/t\_publishnotice")

public String t\_publishnotice(Model model,HttpSession session){

List<Clazz> clazz = clazzManager.getAllClazz();

model.addAttribute("clazz", clazz);

return "teacher/t\_publishnotice";

}

/\*

\* 教师界面(导航栏)----提交统计---按钮页面（内嵌）

\*/

@RequestMapping("/t\_statistics")

public String t\_statistics(@RequestParam("t\_account") String t\_account, Model model,HttpSession session){

//获取所有作业

List<Work> work = workManager.getAllWork();

List<Work> resultWork = new ArrayList<Work>();

List<Work> result = new ArrayList<Work>();

List<String> clazz = new ArrayList<String>();

//获取该账号对应的作业

for(int i=0,j=0; i<work.size(); i++){

if(work.get(i).getW\_t\_account().equals(((Teacher)session.getAttribute("teacher")).getT\_account())){

resultWork.add(work.get(i));

System.out.println("提交统计:"+resultWork.get(j).getW\_content()+resultWork.get(j++).getW\_c\_id());

}

}

//传递该账号对应的作业

for(int i=0; i<resultWork.size(); i++){

result.add(resultWork.get(i));

}

//result = resultWork;

model.addAttribute("resultWork", result);

//删除作业表中该教师所对应的班级的重复值

for ( int i = 0 ; i < resultWork.size(); i ++ ) {

for ( int j = resultWork.size() - 1 ; j > i; j -- ) {

if (resultWork.get(j).getW\_c\_id().equals(resultWork.get(i).getW\_c\_id())) {

resultWork.remove(j);

}

} System.out.println(resultWork.get(i).getW\_content()+resultWork.get(i).getW\_c\_id());

}

//获取该账号对应的班级

for(int i=0; i<resultWork.size(); i++){

clazz.add(resultWork.get(i).getW\_c\_id());

System.out.println(clazz.get(i));

}

//传递该账号对应的班级

model.addAttribute("clazz", clazz);

session.setAttribute("c", clazz);

session.setAttribute("w", result);

//将作业按班级分类传递到页面

for(int i=0; i<clazz.size(); i++){

List<Work> workByClazz = new ArrayList<Work>();

for(int j=0; j<result.size(); j++){

if(result.get(j).getW\_c\_id().equals(clazz.get(i).toString())){

workByClazz.add(result.get(j));

}

}

model.addAttribute("workByClazz",workByClazz);

}

return "teacher/t\_statistics";

}

/\*

\* 教师界面(导航栏)----个人设置---按钮页面（内嵌）

\*/

@RequestMapping("/t\_personalseting")

public String t\_personalseting(Model model){

/\*List<Teacher> teacher1 = teacherManager.getAllTeacher();

System.out.println("teacher教师总览页面:"+teacher1.get(0).getT\_account()+","+teacher1.get(0).getT\_name()+","+teacher1.get(0).getT\_password());

model.addAttribute("teacher", teacher1);\*/

return "teacher/t\_personalseting";

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*iframe内嵌页面操作\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//作业总览删除操作

@RequestMapping("/t\_deletework")

public void t\_deletework(int id,HttpServletResponse response){

String result = "{\"result\":\"error\"}";

if( workManager.deleteWork(id) ){

result = "{\"result\":\"success\"}";

}

response.setContentType("application/json");

PrintWriter out = null;

try {

out = response.getWriter();

out.write(result);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

//公告总览删除操作

@RequestMapping("/t\_deletenotice")

public void t\_deletenotice(int id,HttpServletResponse response){

String result = "{\"result\":\"error\"}";

if( noticeManager.deleteNotice(id) ){

result = "{\"result\":\"success\"}";

}

response.setContentType("application/json");

PrintWriter out = null;

try {

out = response.getWriter();

out.write(result);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

//发布作业点击发布按钮

@RequestMapping("t\_publishWork")

public String t\_publishWork(@RequestParam("w\_title") String w\_title,@RequestParam("w\_content") String w\_content,

@RequestParam("w\_deadline") String w\_deadline,@RequestParam("w\_course") String w\_course,@RequestParam("w\_c\_id") String w\_c\_id,HttpSession session){

Teacher teacher = (Teacher)session.getAttribute("teacher");

Work work = new Work();

work.setW\_c\_id(w\_c\_id);

work.setW\_content(w\_content);

work.setW\_course(w\_course);

work.setW\_deadline(w\_deadline);

work.setW\_t\_account(teacher.getT\_account());

work.setW\_title(w\_title);

workManager.addWork(work);

return "redirect:/teacher/t\_allworks?t\_account="+teacher.getT\_account();

}

//发布公告点击发布按钮

@RequestMapping("t\_publishNotice")

public String t\_publishNotice(@RequestParam("nt\_title") String nt\_title,@RequestParam("nt\_content") String nt\_content,

@RequestParam("nt\_time") String nt\_time,@RequestParam("nt\_c\_id") String nt\_c\_id,HttpSession session){

Teacher teacher = (Teacher)session.getAttribute("teacher");

Notice notice = new Notice();

notice.setNt\_title(nt\_title);

notice.setNt\_content(nt\_content);

notice.setNt\_c\_id(nt\_c\_id);

notice.setNt\_t\_account(teacher.getT\_account());

notice.setNt\_time(nt\_time);

noticeManager.addNotice(notice);

return "redirect:/teacher/t\_allnotice?t\_account="+teacher.getT\_account();

}

//提交统计班级作业点击

@RequestMapping("/checkwork")

public String checkwork(@RequestParam("w\_id")int w\_id,@RequestParam("c\_id")String c\_id,HttpSession session,Model model){

System.out.println("作业号,班级号:"+w\_id+","+c\_id);

int Wno = w\_id;

String clazz = c\_id;

//根据w\_id和c\_id查询批改表中的对应记录

List<CheckWork> checkwork = checkWorkManager.getCheckWorkByWnoClno(w\_id, c\_id);

for(int i=0; i<checkwork.size(); i++){

System.out.println("提交统计班级作业点击123:"+checkwork.get(i).getCh\_s\_account());

}

//根据c\_id查询学生表中的对应记录

List<Student> student = studentManager.getStudentByCLno(c\_id);

Work work = workManager.getWork(w\_id);

session.setAttribute("checkwork", checkwork);

session.setAttribute("student",student );

session.setAttribute("Wno",Wno );

session.setAttribute("c\_id",clazz );

System.out.println("提交统计班级作业点击--班级号:"+clazz);

System.out.println("提交统计班级作业点击--:作业号"+Wno);

session.setAttribute("w\_title",work.getW\_title() );

return "teacher/t\_statistics\_details";

}

/\*

\* 管理员页面（内嵌页面）------个人设置-->修改账号-------（信息修改）

\*/

@RequestMapping("/updateTeacherAccount")

public String updateTeacherAccount(int id, String account, HttpSession request){

System.out.println("教师个人设置修改账号:"+id+","+account);

if( teacherManager.updateTeacherAccount(id , account) ){

Teacher teacher = teacherManager.getTeacher(id);

request.setAttribute("teacher", teacher);

request.setAttribute("msg", "账号修改成功");

return "redirect:/teacher/t\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

}else{

request.setAttribute("msg", "教师账号修改出错了！！");

return "redirect:/teacher/t\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

}

}

/\*

\* 管理员页面（内嵌页面）------个人设置-->修改姓名-------（信息修改）

\*/

@RequestMapping("/updateTeacherName")

public String updateTeacherName(int id, String name, HttpSession request){

System.out.println("教师个人设置修改姓名:"+id+","+name);

if( teacherManager.updateTeacherName(id , name) ){

Teacher teacher = teacherManager.getTeacher(((Teacher)request.getAttribute("teacher")).getId());

/\*model.addAttribute("m\_name", manager.getM\_name());

return "redirect:/manager/m\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

\*/ request.setAttribute("teacher", teacher);

request.setAttribute("msg", "姓名修改成功");

return "redirect:/teacher/t\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

}else{

/\*model.addAttribute("error", "管理员姓名修改出错了！！");

return "redirect:/manager/m\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

\*/ request.setAttribute("msg", "管理员姓名修改出错了！！");

return "redirect:/teacher/t\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

}

}

/\*

\* 教师页面（内嵌页面）------个人设置-->修改密码-------（信息修改）

\*/

@RequestMapping("/updateTeacherPassword")

public String updateTeacherPassword(int id, String password, HttpSession request){ //RedirectAttributes model--->重定向传递参数

System.out.println("教师个人设置修改密码:"+id+","+password);

if( teacherManager.updateTeacherPassword(id, password) ){

Teacher teacher = teacherManager.getTeacher(id);

/\*model.addAttribute("m\_name", manager.getM\_name());

return "redirect:/manager/m\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

\*/ request.setAttribute("teacher", teacher);

request.setAttribute("msg", "密码修改成功");

return "redirect:/teacher/t\_personalseting"; //修改成功后--重定向到-->个人设置页面（内嵌）

}else{

/\*model.addAttribute("error", "管理员密码修改出错了！！");

return "redirect:/manager/m\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

\*/ request.setAttribute("msg", "教师密码修改出错了！！");

return "redirect:/teacher/t\_personalseting"; //修改出错后--重定向到-->个人设置页面（内嵌）

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*iframe内嵌iframe内嵌页面操作\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//提交批改操作

@RequestMapping("/t\_check")

public String t\_check(@RequestParam("ch\_mark") String ch\_mark,@RequestParam("ch\_score") int ch\_score,@RequestParam("ch\_id") int ch\_id,@RequestParam("c\_id") String c\_id,@RequestParam("Wno") int Wno,RedirectAttributes model){

System.out.println("提交批改操作："+ch\_id+","+ch\_mark+","+ch\_score+","+Wno+","+c\_id);

checkWorkManager.Check(ch\_mark, ch\_score, ch\_id);

model.addAttribute("w\_id", Wno);

model.addAttribute("c\_id", c\_id);

return "redirect:/teacher/checkwork"; //重定向到-->提交统计班级作业点击

}

//下载作业

@RequestMapping("/Download")

public String Download(@RequestParam("Wno")int Wno,@RequestParam("c\_id")String c\_id,@RequestParam("Uptime") String Uptime,RedirectAttributes model,HttpServletResponse response) throws IOException{

System.out.println("下载作业："+Wno+","+c\_id+","+Uptime);

//String pathY = "F:\\DownloadWork\\";

String path = "D:\\"; //下载文件的文件夹路径

//解析时分秒

String years = Uptime.substring(0,4);

String month = Uptime.substring(5,7);

String day = Uptime.substring(8,10);

String hour = Uptime.substring(11,13);

String minute = Uptime.substring(14,16);

String seconds = Uptime.substring(17,19);

String UptimeFileName = years +"."+ month +"."+ day + "-" + hour +"."+ minute +"."+ seconds;//文件中显示的时间格式

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("下载作业--文件中显示的时间格式："+years+"."+month+"."+day+ "-"+hour+":"+minute+":"+seconds+"---"+UptimeFileName);

File files1 = new File(path);

String [] names1 = files1.list();

System.out.println(files1.isDirectory());

String FileName = ""; //有后缀

for( String s:names1 ){

if( new File(files1.getAbsolutePath()+s).isFile() ){//判断是否有文件

if( s.contains(UptimeFileName) ){ //判断是否有 包含此字段 的文件

System.out.println("\*\*\*1\*\*\*当前下载作业的文件名是："+s);

FileName = s; //去后缀 s.substring(s.indexOf("/"), s.length());

}

if(s.startsWith("142011")){

System.out.println("\*\*\*2\*\*\*当前下载作业的文件名是："+s);

}

}

}

String fileNameTemp = path + FileName; //文件路径+名称+文件类型

//download(response,fileNameTemp);

//获取所要下载文件的路径

String path1 = fileNameTemp;

String realPath = path1.substring(path1.lastIndexOf(":\\")+1); //真正的文件名+扩展名

//告诉浏览器是以下载的方法获取到资源

//告诉浏览器以此种编码来解析URLEncoder.encode(realPath, "utf-8"))

response.setHeader("content-disposition","attachment; filename="+URLEncoder.encode(realPath, "utf-8"));

//获取到所下载的资源

FileInputStream fis = new FileInputStream(path1);

OutputStream out = response.getOutputStream();

//FileInputStream fis = new FileInputStream("F:\\temp\\myfile.txt");

int len = 0;

byte [] buf = new byte[1024];

while((len=fis.read(buf))!=-1){

out.write(buf,0,len);

}

//关闭输入流

fis.close();

//关闭输出流

out.close();

File files = new File("D:/"); //创建File对象,指向F盘根目录

String[] names = files.list(); //获取F盘根目录所有文件和路径,并以字符串数组返回

for(String s:names){ //遍历字符串数组

boolean a = s.startsWith("ja"); //文件名前缀带有ja的返回true,没有则返回false

boolean b = (new File(files.getAbsolutePath()+s)).isFile(); //判断本次循环的字符串所指向的内容是否是文件,是则返回true.否则返回false

boolean c = s.contains("va"); //文件名是否包含"va",包含则返回true,否则false

if(a&&b){ //此处条件根据需要进行修改

System.out.println(s); //打印出符合条件的文件

}

}

model.addAttribute("w\_id", Wno);

model.addAttribute("c\_id", c\_id);

return "redirect:/teacher/checkwork"; //重定向到-->提交统计班级作业点击

}

public void download(HttpServletResponse response,String fileNameTemp ) throws IOException{

//获取所要下载文件的路径

String path1 = fileNameTemp;

System.out.println(path1);

//String path = this.getServletContext().getRealPath(/WEB-INF/web.xml); //从项目路径下找

String realPath = path1.substring(path1.lastIndexOf("\\")+1);

//告诉浏览器是以下载的方法获取到资源

//告诉浏览器以此种编码来解析URLEncoder.encode(realPath, "utf-8"))

response.setHeader("content-disposition","attachment; filename="+URLEncoder.encode(realPath, "utf-8"));

//获取到所下载的资源

FileInputStream fis = new FileInputStream(path1);

OutputStream out = response.getOutputStream();

//FileInputStream fis = new FileInputStream("F:\\temp\\myfile.txt");

int len = 0;

byte [] buf = new byte[1024];

while((len=fis.read(buf))!=-1){

out.write(buf,0,len);

}

//关闭输入流

fis.close();

//关闭输出流

out.close();

}

}

### 5.2.3 业务逻辑层

教师管理模块的业务逻辑层主要完成发布和批改作业，发布公告等功能操作进行判定，同时调用管理模块的业务逻辑接口，将教师提交的操作关键信息传递到业务逻辑层进行判定。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 管理员管理 | TeacherService.java  ITeacherService.java | 调用TeacherDao | 返回给Action |

ITeacherService.java定义了接口

public interface ITeacherManager {

//\*\*\*\*\*\*\*\*\*管理员用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//增加一条教师记录

public void addTeacher(Teacher teacher);

//删除一条教师记录

public boolean deleteTeacher(String id);

//修改一条教师记录

public boolean updateTeacher(Teacher teacher);

//查询一条教师记录

public Teacher getTeacher(int id);

//查询教师表所有的数据

public List<Teacher> getAllTeacher();

//修改教师账号

public boolean updateTeacherAccount(int id, String account);

//修改教师姓名

public boolean updateTeacherName(int id, String name);

//修改教师密码

public boolean updateTeacherPassword(int id, String password);

//\*\*\*\*\*\*\*\*\*教师用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//修改一条教师姓名

public boolean updateTeacherName(Teacher teacher);

//获得一个教师对象 -->(用于登录验证)

public Teacher getTeacherLogin(String account);

}

TeacherManager实现了接口功能，同时需要调用数据持久层的，利用@Resource来实现对数据持久层接口的调用

public class TeacherManager implements ITeacherManager{

private ITeacherDao teacherDao; //引入教师dao层接口

public void setTeacherDao(ITeacherDao teacherDao) {

this.teacherDao = teacherDao;

}

@Override

public void addTeacher(Teacher teacher) {

teacherDao.addTeacher(teacher);

}

@Override

public boolean deleteTeacher(String id) {

return teacherDao.deleteTeacher(id);

}

@Override

public boolean updateTeacher(Teacher teacher) {

return teacherDao.updateTeacher(teacher);

}

@Override

public Teacher getTeacher(int id) {

return teacherDao.getTeacher(id);

}

@Override

public List<Teacher> getAllTeacher() {

return teacherDao.getAllTeacher();

}

@Override

public boolean updateTeacherName(Teacher teacher) {

return teacherDao.updateTeacherName(teacher);

}

@Override

public Teacher getTeacherLogin(String account) {

return teacherDao.getTeacherLogin(account);

}

@Override

public boolean updateTeacherAccount(int id, String account) {

// TODO Auto-generated method stub

return teacherDao.updateTeacherAccount(id, account);

}

@Override

public boolean updateTeacherName(int id, String name) {

// TODO Auto-generated method stub

return teacherDao.updateTeacherName(id, name);

}

@Override

public boolean updateTeacherPassword(int id, String password) {

// TODO Auto-generated method stub

return teacherDao.updateTeacherPassword(id, password); }

}

### 5.2.4 数据持久层

教师管理模块的是数据持久层能对作业、公告和个人呢信息数据进行增删改查。教师管理模块数据持久层列表见表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 教师管理 | ITeacherDao.java  TeacherDao.java | 调用  Teacher.Java  Course.java  Notice.java  Work.java  Clazz.java | 对作业发布和批阅以及公告和个人信息进行增删改查操作 |

ITeacherDao.java定义了功能接口；

public interface ITeacherDao {

//\*\*\*\*\*\*\*\*\*管理员用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//增加一条教师记录

public void addTeacher(Teacher teacher);

//删除一条教师记录

public boolean deleteTeacher(String id);

//修改一条教师记录

public boolean updateTeacher(Teacher teacher);

//查询一条教师记录

public Teacher getTeacher(int id);

//查询教师表所有的数据

public List<Teacher> getAllTeacher();

//修改教师账号

public boolean updateTeacherAccount(int id, String account);

//修改教师姓名

public boolean updateTeacherName(int id, String name);

//修改教师密码

public boolean updateTeacherPassword(int id, String password);

//\*\*\*\*\*\*\*\*\*登录用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//获得一个教师对象 -->(用于登录验证)

public Teacher getTeacherLogin(String account);

//\*\*\*\*\*\*\*\*\*教师用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//修改一条教师姓名

public boolean updateTeacherName(Teacher teacher);

}

TeacherDao对接口定义的方法进行实现，并调用域模型层进行操作的实现。

public class TeacherDao implements ITeacherDao{

//引入SessionFactory变量,SessionFactory接口负责初始化Hibernate。它充当数据存储源的代理，并负责创建Session对象

private SessionFactory sessionFactory;

public void setSessionFactory(SessionFactory sessionFactory) {

this.sessionFactory = sessionFactory;

}

@Override

public void addTeacher(Teacher teacher) {//添加教师

sessionFactory.getCurrentSession().save(teacher);

}

@Override

public boolean deleteTeacher(String id) {//删除教师

String hql = "delete Teacher t where t.id = ?";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, id);

return (query.executeUpdate()>0); //大于0，删除成功

}

@Override

public boolean updateTeacher(Teacher teacher) {//修改教师信息,hql语句

System.out.println("管理员--教师信息--修改:"+teacher.getId()+teacher.getT\_account()+teacher.getT\_name()+teacher.getT\_sex()+teacher.getT\_dept()+teacher.getT\_password());

String hql = "update Teacher t set t.t\_account=?, t.t\_name=?, t.t\_sex=?, t.t\_dept=?, t.t\_password=? where t.id=?";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, teacher.getT\_account());

query.setString(1, teacher.getT\_name());

query.setString(2, teacher.getT\_sex());

query.setString(3, teacher.getT\_dept());

query.setString(4, teacher.getT\_password());

query.setInteger(5, teacher.getId());

return (query.executeUpdate()>0);

}

@Override

public Teacher getTeacher(int id) {//根据id得到单个教师

String hql = "from Teacher t where t.id = ?";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setInteger(0, id);

return (Teacher) query.uniqueResult();

}

@Override

public List<Teacher> getAllTeacher() {//查找所有教师信息

String hql = "from Teacher";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

return query.list();

}

@Override

public boolean updateTeacherAccount(int id, String account) { //修改教师账号

// TODO Auto-generated method stub

String hql = "update Teacher t set t.t\_account=? where t.id=? ";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, account);

query.setInteger(1, id);

return (query.executeUpdate()>0);

}

@Override

public boolean updateTeacherName(int id, String name) { //修改教师姓名

// TODO Auto-generated method stub

String hql = "update Teacher t set t.t\_name =? where t.id=? ";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, name);

query.setInteger(1, id);

return (query.executeUpdate()>0);

}

@Override

public boolean updateTeacherPassword(int id, String password) { //修改教师密码

// TODO Auto-generated method stub

String hql = "update Teacher t set t.t\_password=? where t.id=? ";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, password);

query.setInteger(1, id);

return (query.executeUpdate()>0);

}

@Override

public boolean updateTeacherName(Teacher teacher) {//修改一条教师姓名

String hql = "update Teacher t set t.t\_name=? where t.t\_account=?";

Query query = sessionFactory.openSession().createQuery(hql);

query.setString(0, teacher.getT\_name());

query.setString(1, teacher.getT\_account());

return (query.executeUpdate() > 0);

}

@Override

public Teacher getTeacherLogin(String account) {//获得一个教师对象 -->(用于登录验证)

String hql = "from Teacher t where t.t\_account=?";

Query query = sessionFactory.openSession().createQuery(hql).setString(0,account);

return (Teacher)query.uniqueResult();

}

}

### 5.2.5 域模型层

教师管理模块用到域模型层中teacher.Java,course.java,notice.java,work.java，这些都是公用域模型，在涉及到作业、公告等操作时，就会调用到该模型，教师管理模块域模型层列表如表所示。

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Teacher.Java  Course.java  Notice.java  Work.java  Clazz.java | 对作业、公告、教师个人信息信息的增、删、改、查操作 |

Teacher.Java

public class Teacher {//教师表

private int id; //id号

private String t\_account; //教师号

private String t\_name; //教师名

private String t\_sex; //性别

private String t\_dept; //所在系

private String t\_password; //密码

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getT\_account() {

return t\_account;

}

public void setT\_account(String t\_account) {

this.t\_account = t\_account;

}

public String getT\_name() {

return t\_name;

}

public void setT\_name(String t\_name) {

this.t\_name = t\_name;

}

public String getT\_sex() {

return t\_sex;

}

public void setT\_sex(String t\_sex) {

this.t\_sex = t\_sex;

}

public String getT\_dept() {

return t\_dept;

}

public void setT\_dept(String t\_dept) {

this.t\_dept = t\_dept;

}

public String getT\_password() {

return t\_password;

}

public void setT\_password(String t\_password) {

this.t\_password = t\_password;

}

}

Course.java

public class Course {//课程表

private int c\_id; //id号

private String c\_name; //课程名

private int c\_credit; //学分

public int getC\_id() {

return c\_id;

}

public void setC\_id(int c\_id) {

this.c\_id = c\_id;

}

public String getC\_name() {

return c\_name;

}

public void setC\_name(String c\_name) {

this.c\_name = c\_name;

}

public int getC\_credit() {

return c\_credit;

}

public void setC\_credit(int c\_credit) {

this.c\_credit = c\_credit;

}

}

Work.java

public class Work {//作业表

private int w\_id; //作业号

private String w\_title; //主题

private String w\_content; //内容

private String w\_deadline; //截止时间

private String w\_course; //课程号

private String w\_t\_account; //教师号

private String w\_c\_id; //班级号

public int getW\_id() {

return w\_id;

}

public void setW\_id(int w\_id) {

this.w\_id = w\_id;

}

public String getW\_title() {

return w\_title;

}

public void setW\_title(String w\_title) {

this.w\_title = w\_title;

}

public String getW\_content() {

return w\_content;

}

public void setW\_content(String w\_content) {

this.w\_content = w\_content;

}

public String getW\_deadline() {

return w\_deadline;

}

public void setW\_deadline(String w\_deadline) {

this.w\_deadline = w\_deadline;

}

public String getW\_course() {

return w\_course;

}

public void setW\_course(String w\_course) {

this.w\_course = w\_course;

}

public String getW\_t\_account() {

return w\_t\_account;

}

public void setW\_t\_account(String w\_t\_account) {

this.w\_t\_account = w\_t\_account;

}

public String getW\_c\_id() {

return w\_c\_id;

}

public void setW\_c\_id(String w\_c\_id) {

this.w\_c\_id = w\_c\_id;

}

}

Clazz.java

public class Clazz {//班级表

private int id; //id号

private String c\_id; //班级号

private int c\_count; //人数

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getC\_id() {

return c\_id;

}

public void setC\_id(String c\_id) {

this.c\_id = c\_id;

}

public int getC\_count() {

return c\_count;

}

public void setC\_count(int c\_count) {

this.c\_count = c\_count;

}

}

Notice.java

public class Notice {//公告表

private int nt\_id; //公告号

private String nt\_title; //主题

private String nt\_content; //内容

private String nt\_time; //发布时间

private String nt\_t\_account;//教师号

private String nt\_c\_id; //班级号

public int getNt\_id() {

return nt\_id;

}

public void setNt\_id(int nt\_id) {

this.nt\_id = nt\_id;

}

public String getNt\_title() {

return nt\_title;

}

public void setNt\_title(String nt\_title) {

this.nt\_title = nt\_title;

}

public String getNt\_content() {

return nt\_content;

}

public void setNt\_content(String nt\_content) {

this.nt\_content = nt\_content;

}

public String getNt\_time() {

return nt\_time;

}

public void setNt\_time(String nt\_time) {

this.nt\_time = nt\_time;

}

public String getNt\_t\_account() {

return nt\_t\_account;

}

public void setNt\_t\_account(String nt\_t\_account) {

this.nt\_t\_account = nt\_t\_account;

}

public String getNt\_c\_id() {

return nt\_c\_id;

}

public void setNt\_c\_id(String nt\_c\_id) {

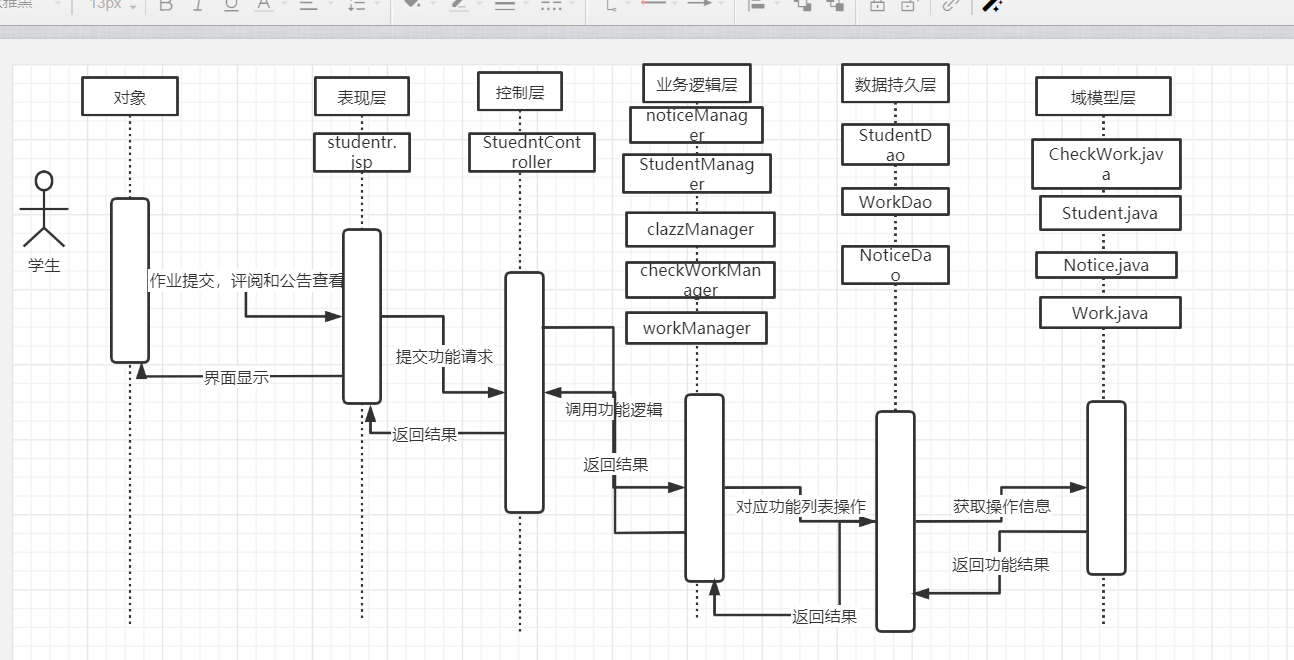
this.nt\_c\_id = nt\_c\_id;

}

}

## 5.3、学生管理模块详细设计

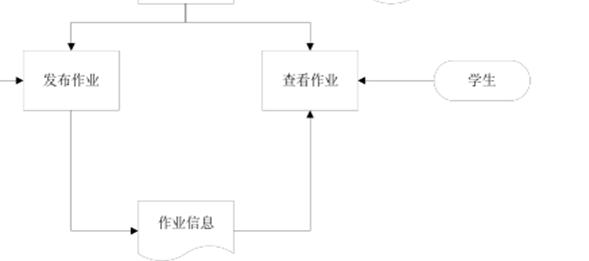
在学生管理模块，系统内部的相应操作示意图



### 5.3.1 表现层

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 教师管理页面 | student.jsp | 学生管理的表现层主要完成作业查看、提交，评阅、公告查看以及个人信息的增删改查。 |

流程图如图所示：



### 5.3.2 控制层

学生管理的控制层负责接受来自sudent.jsp的学生操作输入，同时调用学生管理模块的业务逻辑接口，将学生操作关键信息传递到业务逻辑层进行判定。等到业务逻辑处理完成之后，将来自业务逻辑层的相应信息传到表现层，并决定显示页面。学生管理模块控制层列表见表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 学生管理 | StudentController.java | SUCCESS | s\_showcheck.jsp—对作业进行查看，显示提示窗口。  s\_personalseting.jsp—对个人信息进程管理  s\_uploadwork.jsp—对作业进行提交  s\_home.jsp—对公告进行显示 |
| ERROR | studentr.jsp---操作失败，显示提示窗口。 |

在控制层中StudentController.java的描述如下所示：

@Controller

@RequestMapping("/student")

public class StudentController {

@Resource(name="studentManager")

private IStudentManager studentManager;

@Resource(name="workManager")

private IWorkManager workManager;

@Resource(name="checkWorkManager")

private ICheckWorkManager checkWorkManager;

@RequestMapping("/updateStudentName")

public String updateStudentName(Student student,HttpSession request){

if (studentManager.updateStudentName(student)){

student = studentManager.getStudentLogin(student.getS\_account());

request.setAttribute("student", student);

}

return "/student/s\_personalseting";

}

@RequestMapping("/updateStudentPassword")

public String updateStudentPassword(Student student,HttpSession request){

if (studentManager.updateStudentPassword(student)){

student = studentManager.getStudentLogin(student.getS\_account());

request.setAttribute("student", student);

}

return "/student/s\_personalseting";

}

@RequestMapping("/getAllNotice")

public String getAllNotice(Student student,HttpSession request){

if (studentManager.updateStudentPassword(student)){

student = studentManager.getStudentLogin(student.getS\_account());

request.setAttribute("student", student);

}

return "/student/s\_personalseting";

}

@RequestMapping(value="/s\_home")

public String s\_home(String c\_id,String s\_account,HttpSession request){

List<Notice> notice = studentManager.getAllNotice(c\_id);

List<Work> work = studentManager.getAllWork(c\_id);

boolean[] status = new boolean[work.size()];

String[] teachername = new String[work.size()];

String[] ntTName = new String[notice.size()];

for (int i=0;i<ntTName.length;i++){

ntTName[i] = studentManager.getTeacherNameByAccount(notice.get(i).getNt\_t\_account());

}

for (int n=0;n<status.length;n++){

status[n] = studentManager.getCheckWorkStatus(work.get(n).getW\_id(),s\_account);

}

for (int m=0;m<teachername.length;m++){

teachername[m] = studentManager.getWork\_TeacherName(work.get(m).getW\_t\_account());

}

request.setAttribute("ntTName", ntTName);

request.setAttribute("notice", notice);

request.setAttribute("work", work);

request.setAttribute("status", status);

request.setAttribute("teachername", teachername);

return "student/s\_home";

}

@RequestMapping(value="/s\_personalseting")

public String s\_personalseting(){

return "student/s\_personalseting";

}

@RequestMapping(value="/s\_downloadwork")

public String s\_downloadwork(){

return "student/s\_downloadwork";

}

@RequestMapping(value="/s\_showcheck")

public String s\_showcheck(String s\_account,HttpSession request){

List<CheckWork> checkwork = studentManager.getAllCheckWork(s\_account);

List<Work> work = new ArrayList<Work>(checkwork.size());

for (int n=0;n<checkwork.size();n++){

work.add(n, studentManager.getWorkBycheckwork(checkwork.get(n).getCh\_w\_id()));

}

request.setAttribute("work", work);

request.setAttribute("checkwork", checkwork);

return "student/s\_showcheck";

}

//到达上传页面

@RequestMapping("/s\_uploadwork\_jsp")

public String s\_uploadwork\_jsp(String s\_account,HttpSession session){

String clazz = studentManager.getClazzBySno(s\_account);

List<Work> work = studentManager.getAllWork(clazz);

String[] w\_title = new String[work.size()];

HashMap<Integer,String> map = new HashMap<>();

int[] w\_id=new int[work.size()];

for (int m=0;m<w\_title.length;m++){

w\_title[m] = work.get(m).getW\_title();

w\_id[m] = work.get(m).getW\_id();

map.put(w\_id[m],w\_title[m]);

System.out.println(map.keySet());

System.out.println(map.values());

}

session.setAttribute("map",map);

session.setAttribute("w\_id", w\_id);

session.setAttribute("w\_title", w\_title);

session.setAttribute("s\_account", s\_account);

session.setAttribute("clazz", clazz);

session.setAttribute("flag", 0);

return "student/s\_uploadwork";

}

//上传文件

@RequestMapping(value="/s\_uploadwork")

public String s\_uploadwork(@RequestParam("s\_account") String s\_account,@RequestParam("clazz") String clazz,@RequestParam("w\_id") int w\_id, HttpServletRequest request,HttpServletResponse response,HttpSession session) throws IllegalStateException, IOException{

Work work1 = workManager.getWork(w\_id);

String w\_title = work1.getW\_title();

CommonsMultipartResolver multipartResolver = new CommonsMultipartResolver(request.getSession().getServletContext());

//设置编码

multipartResolver.setDefaultEncoding("utf-8");

//判断 request 是否有文件上传,即多部分请求...

if( multipartResolver.isMultipart(request) ){//获取文件解析器，判断是否是文件

//转换成多部分request

MultipartHttpServletRequest multiRequest = (MultipartHttpServletRequest) (request);

Iterator<String> iter = multiRequest.getFileNames();//获取所有文件名字

while( iter.hasNext() ){//遍历所有文件

// file 是指 文件上传标签的 name=值

// 根据 name 获取上传的文件...

MultipartFile file = multiRequest.getFile((String)iter.next());

if( file != null ){

//获取服务器当前时间

SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy.MM.dd-HH.mm.ss");

Date date = new Date();

String time = simpleDateFormat.format(date); //当前提交时间 (做文件名)

System.out.println(time);

String Uptime = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(date);//（插入数据库）

String fileName = s\_account+"提交作业："+ time +"\_"+file.getOriginalFilename();

System.out.println(fileName);

//String path = "F:\\UploadWork\\" + fileName; //提交路径(用于磁盘写入文件)

//String Upath = "F:\\UploadWork\\"; //提交路径(保存到数据库)

String path = "D:\\Work" + fileName; //提交路径(用于保存文件到磁盘)

String Upath = "D:\\Work"; //提交路径(保存到数据库)

/\*File localFile = new File(path);

file.transferTo(localFile);//上传文件

\*/

String Wtitle = w\_title.trim();

Work work = workManager.getWork(w\_id);

//上传之前先判断是否已上传，即checkwork表中有相同的学号(ch\_s\_account)和作业号（ch\_w\_id）。如果有表示已上传过 ,则更新当前表

if( checkWorkManager.getCheckWorkByWnoSno(work.getW\_id(), s\_account) != null ){ //如果不等于空，表示已提交过

CheckWork chw = checkWorkManager.getCheckWorkByWnoSno(work.getW\_id(), s\_account);

if( (chw.getCh\_score() != 0) && (chw.getCh\_mark() != null) && (chw.getCh\_checkTime() != null) ){//判断教师是否批改过,如果不等于空表示教师已批改过，不可重复提交

session.setAttribute("flag", 2);//返回参数判断是否上传成功

}else{//如果等于空表示教师还未批改过，可重复提交

if(checkWorkManager.updateCheckWorkByFile(s\_account, clazz, work.getW\_id(), Uptime, Upath)){

File localFile = new File(path);

file.transferTo(localFile);//上传文件

}

session.setAttribute("flag", 3);//返回参数判断是否上传成功

}

}else{//表示未上传过 ,即第一次上传次作业

if(checkWorkManager.addCheckWorkByFile(s\_account, clazz, work.getW\_id(), Uptime, Upath)){

File localFile = new File(path);

file.transferTo(localFile);//上传文件

}

session.setAttribute("flag", 1);//返回参数判断是否上传成功

}

/\*if(checkWorkManager.addCheckWorkByFile(s\_account, clazz, work.getW\_id(), Uptime, Upath)){

System.out.println("上传成功，上传数据成功插入数据库表");

}

session.setAttribute("flag", 1);//返回参数判断是否上传成功

\*/

}

}

}

return "student/s\_uploadwork";

}

}

### 5.3.3 业务逻辑层

学生管理的业务逻辑层主要完成对用户登录逻辑的判定，同时调用登录模块的业务逻辑接口。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 学生管理 | IStudentManager.java  StudentManager.java | 调用StudentDao | 返回给Action |

在学生管理的业务逻辑层是调用了IStudentService接口，同时在实现该接口。

IstudentManager：

public interface IStudentManager {

//\*\*\*\*\*\*\*\*\*管理员用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//增加一条学生记录

public void addStudent(Student student);

//删除一条学生记录

public boolean deleteStudent(String id);

//修改一条学生记录

public boolean updateStudent(Student student);

//查询一条学生记录

public Student getStudent(int id);

//查询学生表所有的数据

public List<Student> getAllStudent();

//\*\*\*\*\*\*\*\*\*登录用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//获得一个学生对象 -->(用于登录验证)

public Student getStudentLogin(String account);

//\*\*\*\*\*\*\*\*\*学生用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//修改一条学生姓名

public boolean updateStudentName(Student student);

//修改一个学生的密码

public boolean updateStudentPassword(Student student);

//获得学生的公告

public List<Notice> getAllNotice(String c\_id);

//获得学生的所有作业

public List<Work> getAllWork(String w\_c\_id);

//获得学生的作业批改状态

public boolean getCheckWorkStatus(int ch\_w\_id,String s\_account);

//获得作业的教师姓名

public String getWork\_TeacherName(String t\_account);

//获得批阅列表

public List<CheckWork> getAllCheckWork(String s\_account);

//获得已提交作业

public Work getWorkBycheckwork(int ch\_w\_id);

public String getTeacherNameByAccount(String t\_account);

//根据学号获取班级号(上传作业，学号-->班级号-->作业主题)

public String getClazzBySno(String Sno);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*教师用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//通过班级号查询一条学生记录

public List<Student> getStudentByCLno(String CLno);

}

StudentManager实现了接口的功能，并且调用数据持久层StudenDao

public class StudentManager implements IStudentManager{

private IStudentDao studentDao; //引入学生dao层接口

public void setStudentDao(IStudentDao studentDao) {

this.studentDao = studentDao;

}

@Override

public void addStudent(Student student) {

studentDao.addStudent(student);

}

@Override

public boolean deleteStudent(String id) {

return studentDao.deleteStudent(id);

}

@Override

public boolean updateStudent(Student student) {

return studentDao.updateStudent(student);

}

@Override

public Student getStudent(int id) {

return studentDao.getStudent(id);

}

@Override

public List<Student> getAllStudent() {

return studentDao.getAllStudent();

}

@Override

public boolean updateStudentName(Student student) {

return studentDao.updateStudentName(student);

}

@Override

public Student getStudentLogin(String account) {

return studentDao.getStudentLogin(account);

}

@Override

public boolean updateStudentPassword(Student student) {

return studentDao.updateStudentPassword(student);

}

@Override

public List<Notice> getAllNotice(String c\_id) {

return studentDao.getAllNotice(c\_id);

}

@Override

public List<Work> getAllWork(String w\_c\_id) {

return studentDao.getAllWork(w\_c\_id);

}

@Override

public boolean getCheckWorkStatus(int ch\_w\_id,String s\_account) {

return studentDao.getCheckWorkStatus(ch\_w\_id,s\_account);

}

@Override

public String getWork\_TeacherName(String t\_account) {

return studentDao.getWork\_TeacherName(t\_account);

}

@Override

public List<CheckWork> getAllCheckWork(String s\_account) {

return studentDao.getAllCheckWork(s\_account);

}

@Override

public Work getWorkBycheckwork(int ch\_w\_id) {

return studentDao.getWorkBycheckwork(ch\_w\_id);

}

@Override

public List<Student> getStudentByCLno(String CLno) {

return studentDao.getStudentByCLno(CLno);

}

@Override

public String getTeacherNameByAccount(String t\_account) {

return studentDao.getTeacherNameByAccount(t\_account);

}

@Override

public String getClazzBySno(String Sno) {

return studentDao.getClazzBySno(Sno);

}

}

### 5.3.4 数据持久层

学生管理模块的是数据持久层能对数据进行增删改查。学生管理模块数据持久层列表见表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 登录 | IStudenDao.java  StudentDao.java | 调用 Student.Java  Notice.java  Course.java  Work.java | 对作业、公告进行查看，提交作业 |

IStudenDao.java

public interface IStudentDao {

//\*\*\*\*\*\*\*\*\*管理员用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//增加一条学生记录

public void addStudent(Student student);

//删除一条学生记录

public boolean deleteStudent(String id);

//修改一条学生记录

public boolean updateStudent(Student student);

//查询一条学生记录

public Student getStudent(int id);

//查询学生表所有的数据

public List<Student> getAllStudent();

//\*\*\*\*\*\*\*\*\*登录用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//获得一个学生对象 -->(用于登录验证)

public Student getStudentLogin(String account);

//\*\*\*\*\*\*\*\*\*学生用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//修改一个学生姓名

public boolean updateStudentName(Student student);

//修改一个学生的密码

public boolean updateStudentPassword(Student student);

//获得学生的公告

public List<Notice> getAllNotice(String c\_id);

//获得学生的所有作业

public List<Work> getAllWork(String w\_c\_id);

//获得学生的作业批改状态

public boolean getCheckWorkStatus(int ch\_w\_id,String s\_account);

//获得作业的教师姓名

public String getWork\_TeacherName(String t\_account);

//获得批阅列表

public List<CheckWork> getAllCheckWork(String s\_account);

//获得已提交作业

public Work getWorkBycheckwork(int ch\_w\_id);

//根据学号获取班级号(上传作业，学号-->班级号-->作业主题)

public String getClazzBySno(String Sno);

public String getTeacherNameByAccount(String t\_account);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*教师用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//通过班级号查询一条学生记录

public List<Student> getStudentByCLno(String CLno);

}

StudentDao.java

public class StudentDao implements IStudentDao{

//引入SessionFactory变量,SessionFactory接口负责初始化Hibernate。它充当数据存储源的代理，并负责创建Session对象

private SessionFactory sessionFactory;

public void setSessionFactory(SessionFactory sessionFactory) {

this.sessionFactory = sessionFactory;

}

//\*\*\*\*\*\*\*\*\*管理员用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Override

public void addStudent(Student student) { //增加学生

sessionFactory.getCurrentSession().save(student);

}

@Override

public boolean deleteStudent(String id) {//删除学生

String hql = "delete Student s where s.id = ?";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, id);

return (query.executeUpdate()>0); //大于0，删除成功

}

@Override

public boolean updateStudent(Student student) {//修改学生信息,hql语句

System.out.println("管理员--学生信息--修改:"+student.getId()+student.getS\_account()+student.getS\_name()+student.getS\_sex()+student.getS\_c\_id()+student.getS\_password()+student.getS\_dept());

String hql = "update Student s set s.s\_account=?, s.s\_name=?, s.s\_sex=?, s.s\_c\_id=?, s.s\_password=?, s.s\_dept=? where s.id=?";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setString(0, student.getS\_account());

query.setString(1, student.getS\_name());

query.setString(2, student.getS\_sex());

query.setString(3, student.getS\_c\_id());

query.setString(4, student.getS\_password());

query.setString(5, student.getS\_dept());

query.setInteger(6, student.getId());

return (query.executeUpdate()>0);

}

@Override

public Student getStudent(int id) {//根据id得到单个学生

String hql = "from Student s where s.id = ?";

Query query = sessionFactory.getCurrentSession().createQuery(hql);

query.setInteger(0, id);

return (Student) query.uniqueResult();

}

@Override

public List<Student> getAllStudent() {//查找所有学生信息

String hql = "from Student"; //------针对实体表Student

Query query = sessionFactory.getCurrentSession().createQuery(hql);

return query.list();

}

//\*\*\*\*\*\*\*\*\*登录用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Override

public Student getStudentLogin(String account) { //获得一个学生对象 -->(用于登录验证)

String hql = "from Student s where s.s\_account=?";

Query query = sessionFactory.openSession().createQuery(hql).setString(0,account);

return (Student)query.uniqueResult();

}

//\*\*\*\*\*\*\*\*\*学生用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Override

public boolean updateStudentName(Student student) { //修改一条学生姓名

String hql = "update Student s set s.s\_name=? where s.s\_account=?";

Query query = sessionFactory.openSession().createQuery(hql);

query.setString(0, student.getS\_name());

query.setString(1, student.getS\_account());

return (query.executeUpdate() > 0);

}

@Override

public boolean updateStudentPassword(Student student) {

String hql = "update Student s set s.s\_password=? where s.s\_account=?";

Query query = sessionFactory.openSession().createQuery(hql);

query.setString(0, student.getS\_password());

query.setString(1, student.getS\_account());

return (query.executeUpdate() > 0);

}

@Override

public List<Notice> getAllNotice(String c\_id) {

String hql = "from Notice n where n.nt\_c\_id=?";

Query query = sessionFactory.openSession().createQuery(hql);

query.setString(0, c\_id);

return query.list();

}

@Override

public List<Work> getAllWork(String w\_c\_id) {

String hql = "from Work w where w.w\_c\_id=?";

Query query = sessionFactory.openSession().createQuery(hql);

query.setString(0, w\_c\_id);

return query.list();

}

@Override

public boolean getCheckWorkStatus(int ch\_w\_id,String s\_account) {

String hql = "from CheckWork cw where cw.ch\_w\_id=? and cw.ch\_s\_account=?";

Query query = sessionFactory.openSession().createQuery(hql);

query.setInteger(0, ch\_w\_id);

query.setString(1, s\_account);

return (query.uniqueResult()!=null);

}

@Override

public String getWork\_TeacherName(String t\_account) {

String sql = "select t\_name from teacher where t\_account=?";

SQLQuery sqlquery = sessionFactory.openSession().createSQLQuery(sql);

sqlquery.setString(0, t\_account);

return (String)sqlquery.uniqueResult();

}

@Override

public List<CheckWork> getAllCheckWork(String s\_account) {

String hql = "from CheckWork cw where cw.ch\_s\_account=?";

Query query = sessionFactory.openSession().createQuery(hql);

query.setString(0, s\_account);

return query.list();

}

@Override

public Work getWorkBycheckwork(int ch\_w\_id) {

String hql = "from Work w where w.w\_id=?";

Query query = sessionFactory.openSession().createQuery(hql);

query.setInteger(0, ch\_w\_id);

return (Work)query.uniqueResult();

}

//根据学号获取班级号(上传作业，学号-->班级号-->作业主题)

@Override

public String getClazzBySno(String Sno){

String sql = "select s\_c\_id from Student where Student.s\_account=?";

SQLQuery query = sessionFactory.openSession().createSQLQuery(sql);

query.setString(0, Sno);

return (String)query.uniqueResult();

}

//教师用到的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@Override

public List<Student> getStudentByCLno(String CLno) {

String hql = "from Student s where s.s\_c\_id = ?";

Query query = sessionFactory.openSession().createQuery(hql);

query.setString(0, CLno);

return query.list();

}

@Override

public String getTeacherNameByAccount(String t\_account) {

String sql = "select t\_name from teacher where t\_account=?";

SQLQuery sqlquery = sessionFactory.openSession().createSQLQuery(sql);

sqlquery.setString(0, t\_account);

return (String)sqlquery.uniqueResult();

}

}

### 5.3.5 域模型层

学生管理模块用到域模型层中Student.Java,Notice.java,Course.java,Work.java域模型，在涉及到作业、公告等操作时，就会调用到该模型，学生管理模块域模型层列表如表所示。

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Course.java  Notice.java  Work.java  Clazz.java | 对作业、公告进行查询和提交，对学生个人信息进行修改等操作。 |

Course.java

public class Clazz {//班级表

private int id; //id号

private String c\_id; //班级号

private int c\_count; //人数

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getC\_id() {

return c\_id;

}

public void setC\_id(String c\_id) {

this.c\_id = c\_id;

}

public int getC\_count() {

return c\_count;

}

public void setC\_count(int c\_count) {

this.c\_count = c\_count;

}

}

Course.java

public class Course {//课程表

private int c\_id; //id号

private String c\_name; //课程名

private int c\_credit; //学分

public int getC\_id() {

return c\_id;

}

public void setC\_id(int c\_id) {

this.c\_id = c\_id;

}

public String getC\_name() {

return c\_name;

}

public void setC\_name(String c\_name) {

this.c\_name = c\_name;

}

public int getC\_credit() {

return c\_credit;

}

public void setC\_credit(int c\_credit) {

this.c\_credit = c\_credit;

}

}

Notice.java

public class Notice {//公告表

private int nt\_id; //公告号

private String nt\_title; //主题

private String nt\_content; //内容

private String nt\_time; //发布时间

private String nt\_t\_account;//教师号

private String nt\_c\_id; //班级号

public int getNt\_id() {

return nt\_id;

}

public void setNt\_id(int nt\_id) {

this.nt\_id = nt\_id;

}

public String getNt\_title() {

return nt\_title;

}

public void setNt\_title(String nt\_title) {

this.nt\_title = nt\_title;

}

public String getNt\_content() {

return nt\_content;

}

public void setNt\_content(String nt\_content) {

this.nt\_content = nt\_content;

}

public String getNt\_time() {

return nt\_time;

}

public void setNt\_time(String nt\_time) {

this.nt\_time = nt\_time;

}

public String getNt\_t\_account() {

return nt\_t\_account;

}

public void setNt\_t\_account(String nt\_t\_account) {

this.nt\_t\_account = nt\_t\_account;

}

public String getNt\_c\_id() {

return nt\_c\_id;

}

public void setNt\_c\_id(String nt\_c\_id) {

this.nt\_c\_id = nt\_c\_id;

}

}

Student.java

public class Student {//学生表

private int id; //id号

private String s\_account; //学号

private String s\_name; //姓名

private String s\_sex; //性别

private String s\_c\_id; //班级号

private String s\_password; //密码

private String s\_dept; //所在系

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getS\_account() {

return s\_account;

}

public void setS\_account(String s\_account) {

this.s\_account = s\_account;

}

public String getS\_name() {

return s\_name;

}

public void setS\_name(String s\_name) {

this.s\_name = s\_name;

}

public String getS\_sex() {

return s\_sex;

}

public void setS\_sex(String s\_sex) {

this.s\_sex = s\_sex;

}

public String getS\_c\_id() {

return s\_c\_id;

}

public void setS\_c\_id(String s\_c\_id) {

this.s\_c\_id = s\_c\_id;

}

public String getS\_password() {

return s\_password;

}

public void setS\_password(String s\_password) {

this.s\_password = s\_password;

}

public String getS\_dept() {

return s\_dept;

}

public void setS\_dept(String s\_dept) {

this.s\_dept = s\_dept;

}

}

Work.java

public class Work {//作业表

private int w\_id; //作业号

private String w\_title; //主题

private String w\_content; //内容

private String w\_deadline; //截止时间

private String w\_course; //课程号

private String w\_t\_account; //教师号

private String w\_c\_id; //班级号

public int getW\_id() {

return w\_id;

}

public void setW\_id(int w\_id) {

this.w\_id = w\_id;

}

public String getW\_title() {

return w\_title;

}

public void setW\_title(String w\_title) {

this.w\_title = w\_title;

}

public String getW\_content() {

return w\_content;

}

public void setW\_content(String w\_content) {

this.w\_content = w\_content;

}

public String getW\_deadline() {

return w\_deadline;

}

public void setW\_deadline(String w\_deadline) {

this.w\_deadline = w\_deadline;

}

public String getW\_course() {

return w\_course;

}

public void setW\_course(String w\_course) {

this.w\_course = w\_course;

}

public String getW\_t\_account() {

return w\_t\_account;

}

public void setW\_t\_account(String w\_t\_account) {

this.w\_t\_account = w\_t\_account;

}

public String getW\_c\_id() {

return w\_c\_id;

}

public void setW\_c\_id(String w\_c\_id) {

this.w\_c\_id = w\_c\_id;

}

}