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

**作业管理系统详细设计**

**V1.0**

**广西民族大学**

评审日期： 2018年6月28日

目录

[1．导言](#_Toc8771)

[1.1 目的](#_Toc32116)

[1.2 范围](#_Toc8337)

[1.3 引用标准](#_Toc26843)

[1.4 参考资料](#_Toc19471)

[1.5 版本更新信息](#_Toc15214)

[2．详细设计简述](#_Toc31438)

[2.1 设计简介](#_Toc19174)

[2.2 模块简介](#_Toc19434)

[3、界面详细设计](#_Toc24870)

[3.1、页面设计说明](#_Toc21512)

[4、客户端模块详细设计](#_Toc12619)

[4.1、公共模块-登录模块](#_Toc20152)

[4.1.1 表现层](#_Toc4611)

[4.1.2 控制层](#_Toc991)

[4.1.3 业务逻辑层](#_Toc4034)

[4.1.4 数据持久层](#_Toc18849)

[4.1.5 域模型层](#_Toc15770)

[4.2、管理员模块](#_Toc3678)

[4.2.1 表现层](#_Toc8630)

[4.2.2 控制层](#_Toc29624)

[4.2.3 业务逻辑层](#_Toc17095)

[4.2.4 数据持久层](#_Toc2757)

[4.2.5 域模型层](#_Toc17909)

[4.3、教师模块](#_Toc730)

[4.3.1 表现层](#_Toc18685)

[4.3.2 控制层](#_Toc26508)

[4.3.3 业务逻辑层](#_Toc17067)

[4.3.4 数据持久层](#_Toc12249)

[4.3.5 域模型层](#_Toc17370)

[4.4、学生模块](#_Toc12196)

[4.4.1 表现层](#_Toc1425)

[4.4.2 控制层](#_Toc17882)

[4.4.3 业务逻辑层](#_Toc21759)

[4.4.4 数据持久层](#_Toc13568)

[4.4.5 域模型层](#_Toc13410)

# 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
* 用户

## 1.2 范围

该文档内容涵盖作业管理系统的所有功能模块的详细设计描述。

## 1.3 引用标准

[1] 《软件工程案例教程 第2版》 韩万江等 机械工业出版社

## 1.4 参考资料

[1] 《软件项目管理案例教程 第3版》 韩万江等 机械工业出版社

## 1.5 版本更新信息

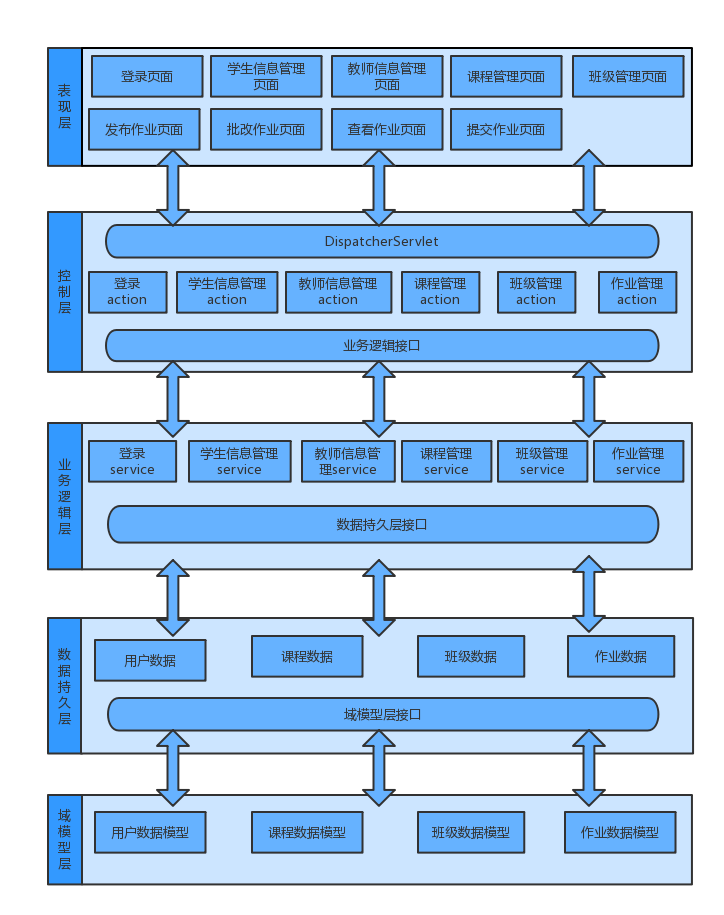
本文档的更新记录如下表1-1所示。

表1-1 文档更新记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改编号 | 修改日期 | 修改后版本 | 修改位置 | 修改内容概述 |
| 001 | 2018.6.25 | 0.1 | 全部 | 初始发布版本 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 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2-1所示。 图 2-1 作业管理系统分层结构图示意图

## 2.2 模块简介

系统功能模块图如图2-2所示。

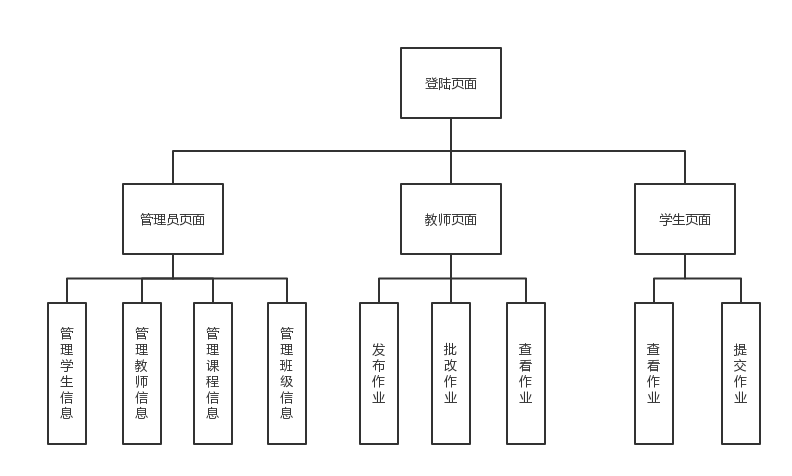


图 2-2 作业管理系统功能模块示意图

# 3、界面详细设计

## 3.1、页面设计说明

首页:



管理员页面：



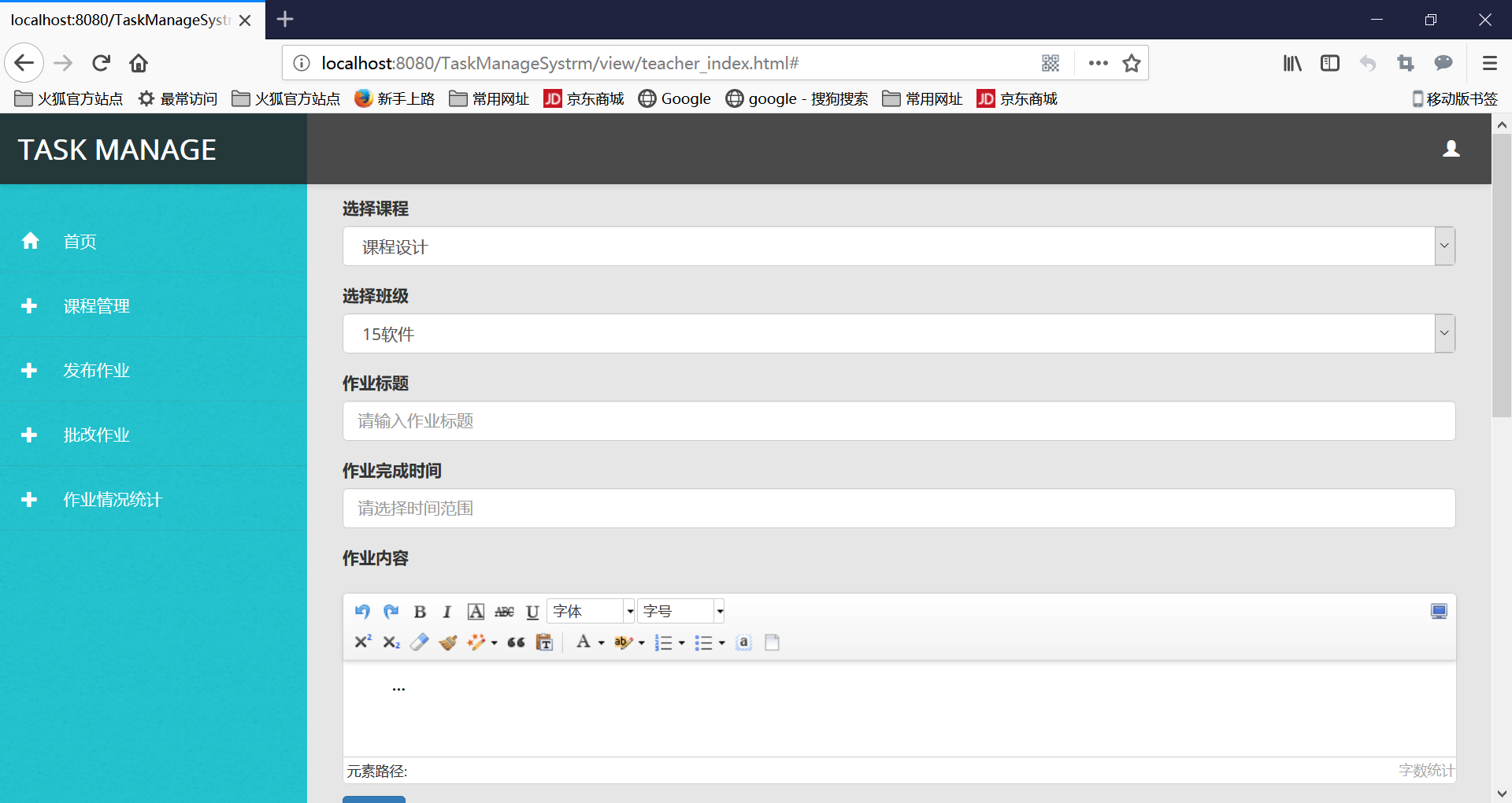




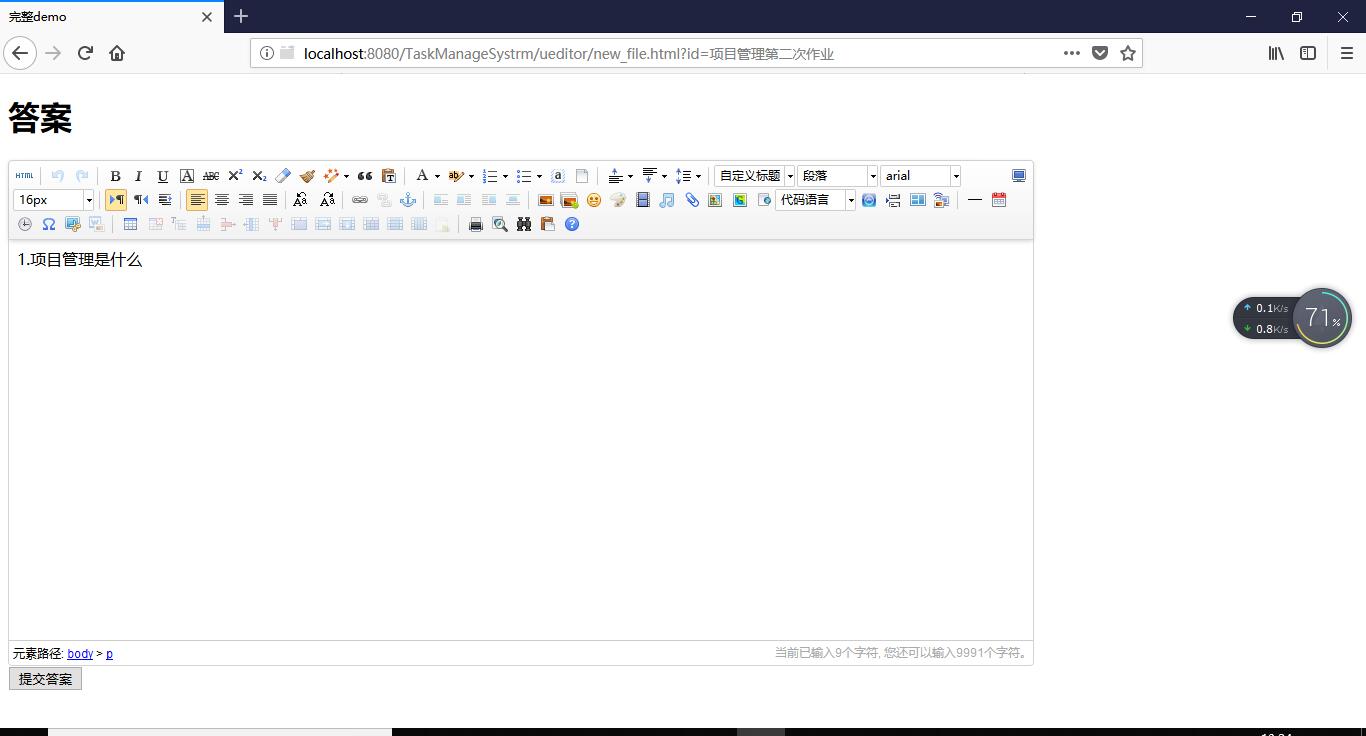


教师页面：





学生页面：

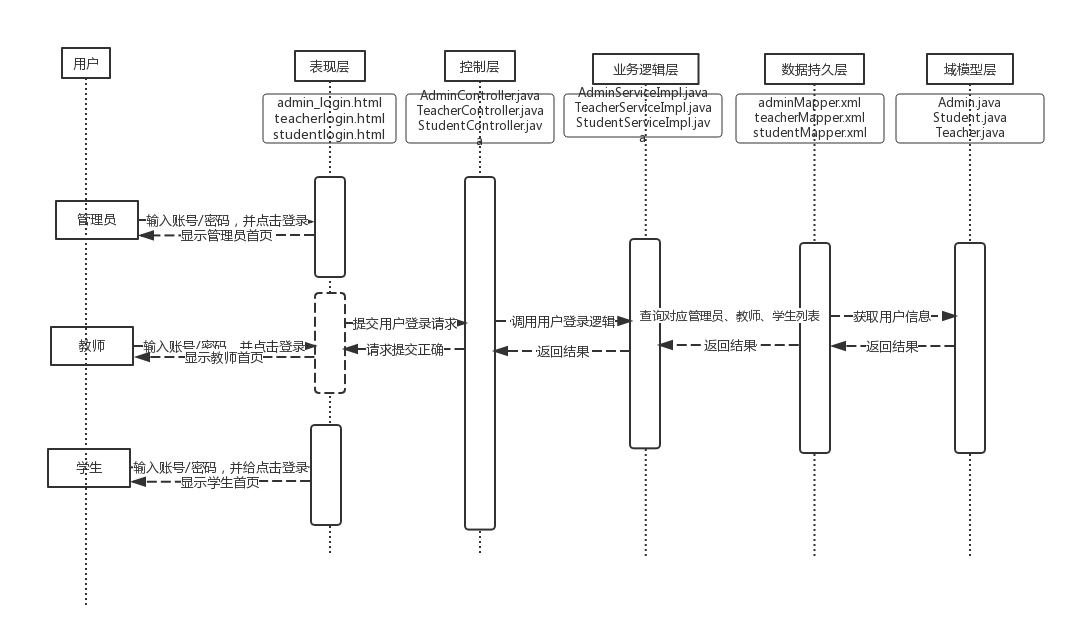


# 4、客户端模块详细设计

客户端模块主要包括管理员模块、学生模块、教师模块。

## 4.1、公共模块-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4-1所示。

图 4-1 系统响应示意图L

### 4.1.1 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HTML页面列表见表4-3所示。

表 4-3 登入模块表现层HTML列表

|  |  |  |
| --- | --- | --- |
| 界面 | HTML | 功能描述 |
| 管理员登录页面 | admin\_login.html | 管理员登入功能，当登入出错时给出提示。 |
| 教师登录页面 | teacherlogin.html | 教师登入功能，当登入出错时给出提示。 |
| 学生登录页面 | studentlogin.html | 学生登入功能，当登入出错时给出提示。 |

登录的流程图如图4-3所示。

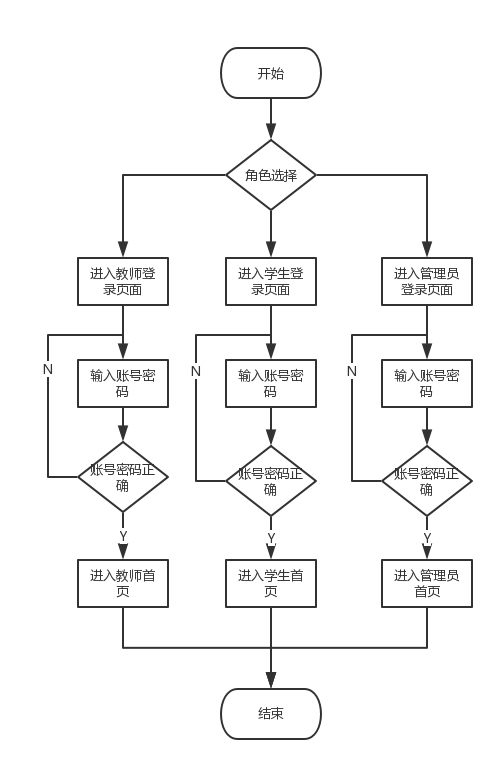


图 4-3 登录模块详细设计流程图

### 4.1.2 控制层

登录模块的控制层负责接受来自admin\_login.html、teacherlogin.html、studentlogin.html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4-4所示。

表 4-4 登入模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 登录 | AdminController.java  TeacherController.java  StudentController.java | SUCCESS | admin\_home\_page.html  student\_homepage.html  teacher\_homepage.html分别进入管理员首页、教师首页、学生首页。 |
| ERROR | admin\_login.html  teacherlogin.html  studentlogin.html---登入失败，显示提示窗口。 |

在控制层中AdminController.java、TeacherController.java、StudentController.java的描述分别如下所示：

AdminController.java：

package com.gxun.task.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

import com.gxun.task.service.IAdminService;

@Controller

@RequestMapping("/view")

public class AdminController {

@Autowired

IAdminService ias;

@RequestMapping("/a\_login")

public String login(String adm\_account, String adm\_password){

boolean result = ias.isValidUser(adm\_account, adm\_password);

if (result){

return "admin\_home\_page";

}

return "admin\_login";

}

}

TeacherController.java、StudentController.java：

package com.gxun.task.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

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

import com.gxun.task.service.IUserService;

@Controller

@RequestMapping("/user")

public class UserController {

@Autowired

IUserService usvr;

@RequestMapping("/lg")

public String login(String account, String pwd,HttpServletRequest reqest){

boolean result = usvr.isValidUser(account, pwd);

if (result){

reqest.getSession().setAttribute("account", account);

return "student\_homepage";

}

return "ng";

}

@RequestMapping("/tealogin")

public String tealogin(String account, String pwd,HttpServletRequest reqest){

boolean result = usvr.isValidteacher(account, pwd);

if (result){

reqest.getSession().setAttribute("techer\_sno", account);

return "teacher\_index";

}

return "ng";

}

}

### 4.1.3 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 4-5登入模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 登录 | AdminServiceImpl.java  TeacherServiceImpl.java  StudentServiceImpl.java | 调用IAdminService.java  ITeacherService.java  IStudentService.java | 返回给Controller |

在登入模块的业务逻辑层是调用了公用的IAdminService.java、ITeacherService.java、IStudentService.java接口，同时在实现该接口。

IAdminService.java接口主要方法：

**public** **interface** IAdminService {

**boolean** isValidUser(String adm\_account, String adm\_password);

}

ITeacherService.java接口主要方法：

**public** **interface** ITeacherService {

**boolean** isInsertTeacher(String t\_no,String t\_password,String t\_name,String t\_phonenum,String t\_college);

String isSearchTeacher();

}

}

}

IStudentService.java接口主要方法：

**public** **interface** IUserService {

**boolean** isValidUser(String account, String pwd);

**boolean** isValidteacher(String account, String pwd);

}

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

AdminServiceImpl.java主要实现属性与方法:

**package com.gxun.task.service.impl;**

**import org.springframework.beans.factory.annotation.Autowired;**

**import org.springframework.stereotype.Service;**

**import com.gxun.task.entity.Admin;**

**import com.gxun.task.mapper.AdminMapper;**

**import com.gxun.task.service.IAdminService;**

**@Service**

**public class AdminServiceImpl implements IAdminService{**

**@Autowired**

**AdminMapper adminMapper;**

**@Override**

**public boolean isValidUser(String adm\_account, String adm\_password) {**

**Admin u = adminMapper.qryByAnameAndPwd(adm\_account, adm\_password);**

**if (null!=u){**

**return true;**

**}**

**return false;**

**}**

**}**

### 4.1.4 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4-6所示。

表 4-6 登入模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Mapper | 调用数据模型 | 说明 |
| 登录 | AdminMapper.java  TeacherMapper.java  StudentMapper.java | 调用 user.Java | 对用户信息进行增删改查操作 |

AdminMapper.java定义了管理员信息进行验证的接口：

**package** com.gxun.task.mapper;

**import** com.gxun.task.entity.Admin;

**public** **interface** AdminMapper {

Admin qryByAnameAndPwd(String adm\_account, String adm\_password);

}

### 4.1.5 域模型层

登录模块用到域模型层中Admin.java、Teacher.java、Student.Java，Admin.java、Teacher.java、Student.Java是一个公用域模型，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4-7所示。

表 4-7登入模块域模型层列表

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

Admin.java主要属性与方法：

**package com.gxun.task.entity;**

**public class Admin {**

**private String adm\_account;**

**private String adm\_password;**

**private String adm\_name;**

**public String getAdm\_account() {**

**return adm\_account;**

**}**

**public void setAdm\_account(String adm\_account) {**

**this.adm\_account = adm\_account;**

**}**

**public String getAdm\_password() {**

**return adm\_password;**

**}**

**public void setAdm\_password(String adm\_password) {**

**this.adm\_password = adm\_password;**

**}**

**public String getAdm\_name() {**

**return adm\_name;**

**}**

**public void setAdm\_name(String adm\_name) {**

**this.adm\_name = adm\_name;**

**}**

}

Teacher.java主要属性与方法：

**package com.gxun.task.entity;**

**public class Student {**

**private int id;**

**private String stuNo;**

**private String stuPwd;**

**private String stuName;**

**private String stuCollege;**

**private String stuClass;**

**public int getId() {**

**return id;**

**}**

**public void setId(int id) {**

**this.id = id;**

**}**

**public String getStuNo() {**

**return stuNo;**

**}**

**public void setStuNo(String stuNo) {**

**this.stuNo = stuNo;**

**}**

**public String getStuPwd() {**

**return stuPwd;**

**}**

**public void setStuPwd(String stuPwd) {**

**this.stuPwd = stuPwd;**

**}**

**public** String getStuName() {

**return** stuName;

}

**public** **void** setStuName(String stuName) {

**this**.stuName = stuName;

}

**public** String getStuCollege() {

**return** stuCollege;

}

**public** **void** setStuCollege(String stuCollege) {

**this**.stuCollege = stuCollege;

}

**public** String getStuClass() {

**return** stuClass;

}

**public** **void** setStuClass(String stuClass) {

**this**.stuClass = stuClass;

}

}

Student.Java主要属性与方法：

package com.gxun.task.entity;

public class Student {

private int id;

private String stuNo;

private String stuPwd;

private String stuName;

private String stuCollege;

private String stuClass;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getStuNo() {

return stuNo;

}

public void setStuNo(String stuNo) {

this.stuNo = stuNo;

}

public String getStuPwd() {

return stuPwd;

}

public void setStuPwd(String stuPwd) {

this.stuPwd = stuPwd;

}

public String getStuName() {

return stuName;

}

public void setStuName(String stuName) {

this.stuName = stuName;

}

public String getStuCollege() {

return stuCollege;

}

public void setStuCollege(String stuCollege) {

this.stuCollege = stuCollege;

}

public String getStuClass() {

return stuClass;

}

public void setStuClass(String stuClass) {

this.stuClass = stuClass;

}

}

## 4.2、管理员模块

管理员模块的系统内部的相应响应操作示意图如图4-2所示。

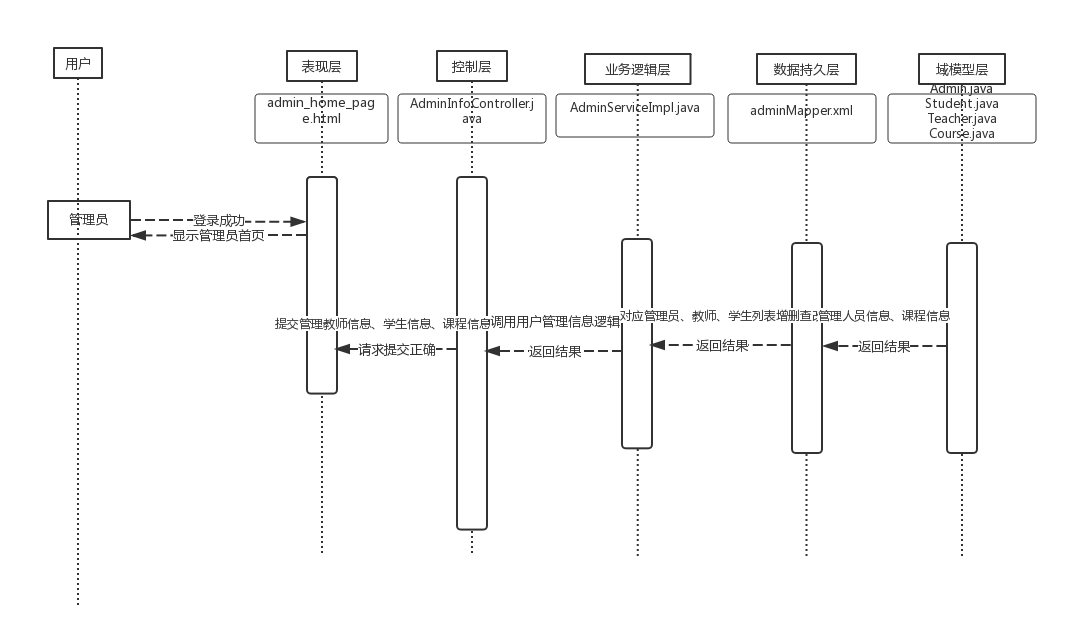


图 4-2 系统响应操作示意图

### 4.2.1 表现层

管理员模块的表现层主要完成对教师信息、学生信息、课程信息的增删查改功能，对相应教师信息、学生信息、课程信息添加后，显示信息在相应的页面上表现层对应的HTML页面列表，见表4-4所示。

表 4-4 管理员模块表现层HTML列表

|  |  |  |
| --- | --- | --- |
| 界面 | HTML | 功能描述 |
| 教师信息管理页面 | add\_teacher.html | 管理员对教师的信息进行增删查改 |
| 学生信息管理页面 | add\_class.html | 管理员对班级（学生）的信息进行增删查改 |
| 课程信息管理页面 | add\_lesson.html | 管理员对课程的信息进行增删查改 |

管理员模块的流程图如图4-4所示。

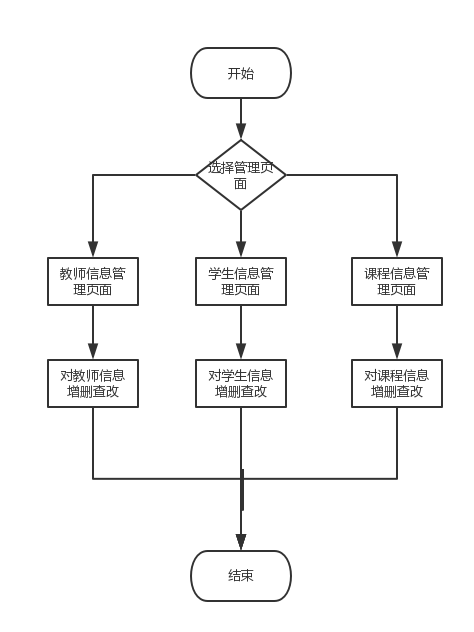


图 4-4 管理员模块详细设计流程图

### 4.2.2 控制层

管理员模块的控制层负责接受来自add\_teacher.html、add\_class.html、add\_lesson.html的信息管理，同时调用管理员模块的业务逻辑接口，将相对应的教师信息、学生信息、课程信息传递到业务逻辑层进行操作。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。管理员模块控制层列表见表4-5所示。

表 4-5 管理员模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 管理员管理学生信息、教师信息、课程信息 | StudentInfoController.java  TeacherInfoController.java  CourseInfoController.java | SUCCESS | 相应信息显示在页面上 |
| ERROR |  |

在控制层中StudentInfoController.java、TeacherInfoController.java、CourseInfoController.java的描述分别如下所示：

StudentInfoController.java：

package com.gxun.task.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

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

import com.gxun.task.service.IStudentService;

@Controller

@RequestMapping("/studentinfo")

public class StudentInfoController {

@Autowired

IStudentService iss;

//插入学生信息

@RequestMapping("/s\_insert")

public String insertStudentInfo(String stuNo,String stuPwd,String stuName,String stuCollege,String stuClass) {

boolean result = iss.isInsertStudent(stuNo, stuPwd, stuName, stuCollege, stuClass);

if (result){

return "add\_class";

}

return "add\_class";

}

//查询所有学生信息

@RequestMapping("/s\_search")

@ResponseBody

public String searchTeacherInfo() {

String json = iss.isSearchStudent();

return json;

}

}

TeacherInfoController.java

package com.gxun.task.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

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

import com.gxun.task.service.ITeacherService;

@Controller

@RequestMapping("/teacher")

public class TeacherInfoController {

@Autowired

ITeacherService its;

@RequestMapping("/t\_insert")

//插入教师信息

public String insertTeacherInfo(String t\_no,String t\_password,String t\_name,String t\_phonenum,String t\_college){

boolean result =its.isInsertTeacher(t\_no, t\_password, t\_name, t\_phonenum, t\_college);

if (result){

return "add\_teacher";

}

return "add\_teacher";

}

//查询所有教师信息

@RequestMapping("/t\_search")

@ResponseBody

public String searchTeacherInfo() {

String json = its.isSearchTeacher();

return json;

}

}

### 4.2.3 业务逻辑层

管理员模块的业务逻辑层主要完成管理员对教师信息、学生信息、课程信息的增删查改逻辑的判定，同时调用管理员模块的业务逻辑接口。比如：管理员选择管理教师，对教师进修增删查改操作。管理员模块业务逻辑层列表如图4-5所示。

表 4-5管理员模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 管理员管理学生、教师、课程信息 | CourseServiceImpl.java  TeacherServiceImpl.java  StudentServiceImpl.java | 调用ICourseService.java  ITeacherService.java  IStudentService.java | 返回给Controller |

在管理员模块的业务逻辑层是调用了公用的ICourseService.java、ITeacherService.java、IStudentService.java接口，同时在实现该接口。

ICourseService.java接口主要方法：

**public** **interface** ICourseService {

**boolean** isInsertCourse(String c\_no,String c\_name,String c\_class,String c\_teacher);

String isSearchCourse();

}

ITeacherService.java接口主要方法：

**public** **interface** ITeacherService {

**boolean** isInsertTeacher(String t\_no,String t\_password,String t\_name,String t\_phonenum,String t\_college);

String isSearchTeacher();

}

}

}

IStudentService.java接口主要方法：

**public** **interface** IStudentService {

**boolean** isInsertStudent(String stuNo,String stuPwd,String stuName,String stuCollege,String stuClass);

String isSearchStudent();

}

CourseServiceImpl.java实现了ICourseService.java接口，StudentServiceImpl.java实现了IStudentService.java接口，TeacherServiceImpl.java实现了ITeacherService.java接口，同时需要调用数据持久层的CourseMapper.Java/StudentMapper.Java/TeacherMapper.Java，利用@Resource来实现对数据持久层接口的调用。

CourseServiceImpl.java主要实现属性与方法:

package com.gxun.task.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.alibaba.fastjson.JSON;

import com.gxun.task.entity.Course;

import com.gxun.task.entity.CourseTask;

import com.gxun.task.entity.Teacher;

import com.gxun.task.mapper.CourseMapper;

import com.gxun.task.mapper.TaskMapper;

import com.gxun.task.service.ICourseService;

@Service

public class CourseServiceImpl implements ICourseService{

@Autowired

CourseMapper courseMapper;

//插入课程信息

@Override

public boolean isInsertCourse(String c\_no,String c\_name,String c\_class,String c\_teacher){

Course c = courseMapper.insertCourseInfo(c\_no, c\_name, c\_class, c\_teacher);

if (c!=null){

return true;

}

return false;

}

//查询所有课程信息

public String isSearchCourse() {

List<Course> c=courseMapper.searchCourseInfo();

String json=JSON.toJSONString(c);

return json;

}

}

StudentServiceImpl.java主要实现属性与方法:

package com.gxun.task.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.alibaba.fastjson.JSON;

import com.gxun.task.entity.Student;

import com.gxun.task.entity.Teacher;

import com.gxun.task.mapper.StudentMapper;

import com.gxun.task.service.IStudentService;

@Service

public class StudentServiceImpl implements IStudentService{

@Autowired

StudentMapper studentMapper;

@Override

public boolean isInsertStudent(String stuNo, String stuPwd, String stuName, String stuCollege, String stuClass) {

Student s = studentMapper.insertStuInfo(stuNo, stuPwd, stuName, stuCollege, stuClass);

if (s!=null){

return true;

}

return false;

}

@Override

public String isSearchStudent() {

List<Student> t=studentMapper.searchStuInfo();

String json=JSON.toJSONString(t);

return json;

}

}

TeacherServiceImpl.java主要实现属性与方法:

package com.gxun.task.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.alibaba.fastjson.JSON;

import com.gxun.task.entity.Teacher;

import com.gxun.task.mapper.TeacherMapper;

import com.gxun.task.service.ITeacherService;

@Service

public class TeacherServiceImpl implements ITeacherService{

@Autowired

TeacherMapper teacherMapper;

@Override

//插入教师信息

public boolean isInsertTeacher(String t\_no,String t\_password,String t\_name,String t\_phonenum,String t\_college){

Teacher t = teacherMapper.insertTeaInfo(t\_no, t\_password, t\_name, t\_phonenum, t\_college);

if (t!=null){

return true;

}

return false;

}

//查询所有教师信息

public String isSearchTeacher() {

List<Teacher> t=teacherMapper.searchTeaInfo();

String json=JSON.toJSONString(t);

return json;

}

}

### 4.2.4 数据持久层

管理员模块的是数据持久层能对课程信息，教师信息，学生信息进行增删改查。管理员模块数据持久层列表见表4-7所示。

表 4-7 管理员模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Mapper | 调用数据模型 | 说明 |
| 管理员管理学生、教师、课程信息 | CourseMapper.java  TeacherMapper.java  StudentMapper.java | 调用 Course.Java  Student.java  Teacher.java | 对学生信息、教师信息、课程信息进行增删改查操作 |

CourseMapper.java定义了对课程的增删查改的接口：

**package com.gxun.task.mapper;**

**import java.util.List;**

**import com.gxun.task.entity.Course;**

**import com.gxun.task.entity.CourseTask;**

**public interface CourseMapper {**

**Course insertCourseInfo(String c\_no,String c\_name,String c\_class,String c\_teacher);**

**List<Course> searchCourseInfo();**

**}**

TeacherMapper.java定义了对教师的增删查改的接口：

**package com.gxun.task.mapper;**

**import java.util.List;**

**import com.gxun.task.entity.Teacher;**

**public interface TeacherMapper {**

**Teacher insertTeaInfo(String t\_no,String t\_password,String t\_name,String t\_phonenum,String t\_college);**

**List<Teacher> searchTeaInfo();**

**}**

StudentMapper.java定义了对学生的增删查改的接口：

**package com.gxun.task.mapper;**

**import java.util.List;**

**import com.gxun.task.entity.Student;**

**public interface StudentMapper {**

**Student insertStuInfo(String stuNo,String stuPwd,String stuName,String stuCollege,String stuClass);**

**List<Student> searchStuInfo();**

**}**

### 4.2.5 域模型层

管理员模块用到域模型层中Admin.java、Teacher.java、Student.Java，Admin.java、Teacher.java、Student.Java是一个公用域模型，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4-7所示。

表 4-7登入模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Course.java | 对课程进行增删查改 |
| Teacher.java | 对教师进行增删查改 |
| Student.Java | 对学生进行增删查改 |

## 4.3、教师模块

教师模块的系统内部的相应响应操作示意图如图4-2所示。

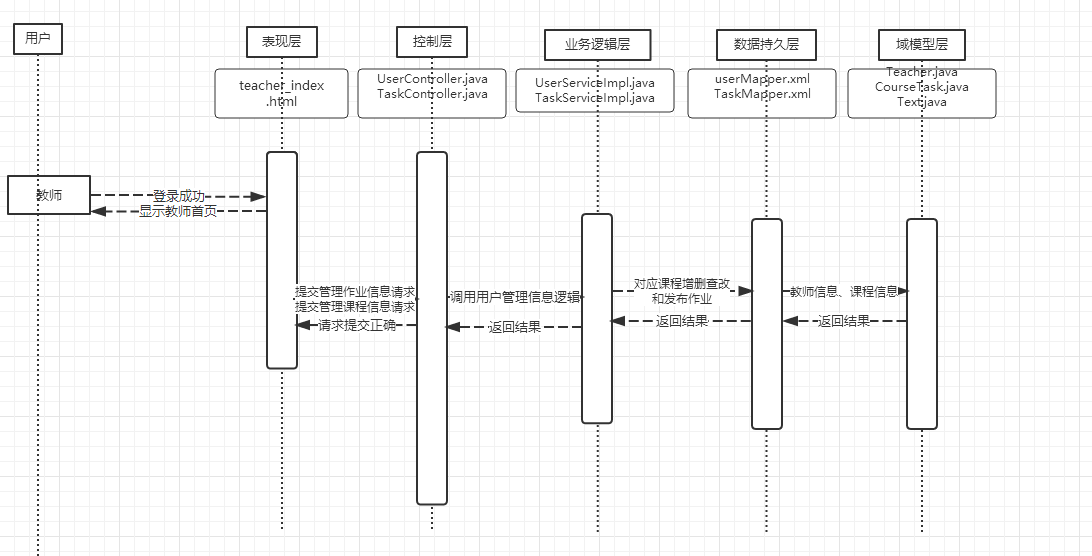


图 4-2 系统响应操作示意图

### 4.3.1 表现层

教师模块的表现层主要完成对课程信息的增删查改、发布作业功能，对相应课程信息添加后，显示信息在相应的页面上表现层对应的HTML页面，发布作业后，对应班级的学生收到作业，列表见表4-4所示。

表 4-4 教师模块表现层HTML列表

|  |  |  |
| --- | --- | --- |
| 界面 | HTML | 功能描述 |
| 作业管理页面 | add\_task.html | 教师发布有关课程作业到班级 |
| 课程信息管理页面 | course\_manage.html | 教师对课程的信息进行增删查 |

教师模块的流程图如图4-4所示

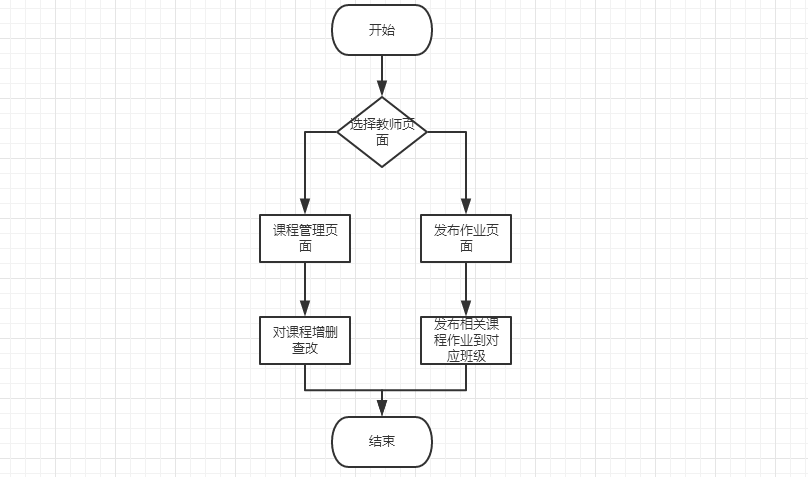


图 4-4 教师模块详细设计流程图

### 4.3.2 控制层

教师模块的控制层负责接受来自add\_task.html、course\_manage.html的信息管理，同时调用教师模块的业务逻辑接口，将相对应的作业信息、课程信息传递到业务逻辑层进行操作。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。教师模块控制层列表见表4-5所示。

表 4-5 教师模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 教师管理课程信息、发布作业 | TaskController.java  UserController.java | SUCCESS | 相应信息显示在页面上 |
| ERROR |  |

在控制层中TaskController.java、UserController.java的描述分别如下所示：

TaskController.java

package com.gxun.task.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

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

import com.alibaba.fastjson.JSON;

import com.gxun.task.service.TaskService;

@Controller

@RequestMapping("/course")

public class TaskController {

@Autowired

TaskService cs;

@RequestMapping("/addCourse")

@ResponseBody

public String addCourse(String courseName) {

if(cs.addCourse(courseName)) {

return "y";

}

return "n";

}

//判断课程是否存在

@RequestMapping("/judgeCourseExist")

@ResponseBody

public String judgeCourseExist(String courseName) {

if(cs.isExist(courseName)) {

return "y";

}

return "n";

}

@RequestMapping("/showCourse")

@ResponseBody

public String showCourse() {

String j = cs.getCourses();

return j;

}

@RequestMapping("/showClasss")

@ResponseBody

public String showClasss(){

String json = cs.showClasss();

return json;

}

@RequestMapping("/delCourse")

@ResponseBody

public String delCourse(String cond) {

boolean r = cs.delCourse(cond);

System.out.println(cond);

if(r) {

return "ok";

}

return "ng";

}

}

UserController.java

package com.gxun.task.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

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

import com.gxun.task.service.IUserService;

@Controller

@RequestMapping("/user")

public class UserController {

@Autowired

IUserService usvr;

@RequestMapping("/lg")

public String login(String account, String pwd,HttpServletRequest reqest){

boolean result = usvr.isValidUser(account, pwd);

if (result){

reqest.getSession().setAttribute("account", account);

return "student\_homepage";

}

return "ng";

}

@RequestMapping("/tealogin")

public String tealogin(String account, String pwd,HttpServletRequest reqest){

boolean result = usvr.isValidteacher(account, pwd);

if (result){

reqest.getSession().setAttribute("techer\_sno", account);

return "teacher\_index";

}

return "ng";

}

@RequestMapping("/getSession")

@ResponseBody

public String getSession(String sessionName, HttpServletRequest request) {

if (request.getSession().getAttribute(sessionName) == null) {

return "ng";

}

return request.getSession().getAttribute(sessionName).toString();

}

@RequestMapping("/addTask")

@ResponseBody

public String addTask(String tid, String taskName,String start, String end, String context,

String cName, String cla) {

boolean i=usvr.insertTask(tid, taskName, start, end, context, cName, cla);

if(i) {

return "ok";

}

return "ng";

}

@RequestMapping("/getTask")

@ResponseBody

public String getTask(String taskName) {

String json = usvr.getTask(taskName);

return json;

}

@RequestMapping("/upTask")

@ResponseBody

public String upTask(String taskName,String task) {

boolean r = usvr.upTask(taskName,task);

if(r) {

return "ok";

}

return "ng";

}

}

### 4.3.3 业务逻辑层

教师模块的业务逻辑层主要完成教师对课程信息增删查改、发布作业逻辑的判定，同时调用教师模块的业务逻辑接口。比如：管理员选择管理课程信息，对课程信息进修增删查改操作。教师模块业务逻辑层列表如图4-5所示。

表 4-5教师模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 教师管理课程信息、发布作业 | TaskServiceImpl.java  UserServiceImpl.java | 调用TaskService.java  IUserService.java | 返回给Controller |

在管理员模块的业务逻辑层是调用了公用的TaskService.java、IUserService.java接口，同时在实现该接口。

TaskService.java接口主要方法：

public interface TaskService {

//添加课程

public boolean addCourse(String courseName);

//判断课程是否存在

public boolean isExist(String courseName);

public String getCourses();

public String showClasss();

public boolean delCourse(String cond);

}

IUserService.java接口主要方法

public interface IUserService {

boolean isValidUser(String account, String pwd);

boolean isValidteacher(String account, String pwd);

boolean insertTask(String tid,String taskName,String start,String end,

String context,String cName,String cla);

String getTask(String taskName);

boolean upTask(String taskName, String task);

}

TaskServiceImpl.java实现了TaskService.java接口，UserServiceImpl实现了IUserService.java接口，同时需要调用数据持久层的CourseMapper.Java/StudentMapper.Java/TeacherMapper.Java，利用@Resource来实现对数据持久层接口的调用。

TaskServiceImpl.java主要实现属性与方法:

package com.gxun.task.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.alibaba.fastjson.JSON;

import com.gxun.task.entity.CourseTask;

import com.gxun.task.mapper.TaskMapper;

import com.gxun.task.service.TaskService;

@Service

@Transactional

public class TaskServiceImpl implements TaskService{

@Autowired

TaskMapper cm;

@Override

public boolean addCourse(String courseName) {

int i = cm.addOneCourse(courseName);

if(i>=1) {

return true;

}

return false;

}

@Override

public boolean isExist(String courseName) {

CourseTask course =cm.qryCourseByName(courseName);

if(null!=course) {

return true;

}

return false;

}

@Override

public String getCourses() {

List<CourseTask> list = cm.qryCourses();

return JSON.toJSONString(list);

}

@Override

public String showClasss() {

List<String> list = cm.showClasss();

return JSON.toJSONString(list);

}

@Override

public boolean delCourse(String cond) {

int i = cm.delCourse(cond);

if(i>=1) {

return true;

}

return false;

}

}

UserServiceImpl.java主要实现属性与方法:

package com.gxun.task.service.impl;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.scheduling.config.Task;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.alibaba.fastjson.JSON;

import com.gxun.task.entity.Student;

import com.gxun.task.entity.Teacher;

import com.gxun.task.entity.Text;

import com.gxun.task.mapper.UserMapper;

import com.gxun.task.service.IUserService;

@Service

@Transactional

public class UserServiceImpl implements IUserService {

@Autowired

UserMapper userMapper;

@Override

public boolean isValidUser(String account, String pwd) {

Student u = userMapper.qryByUnameAndPwd(account, pwd);

if (null!=u){

return true;

}

return false;

}

@Override

public boolean isValidteacher(String account, String pwd) {

Teacher u = userMapper.qryTeacher(account, pwd);

if (null!=u){

return true;

}

return false;

}

@Override

public boolean insertTask(String tid, String taskName, String start, String end, String context,

String cName, String cla) {

Date d=new Date();

SimpleDateFormat f=new SimpleDateFormat("yyyy-MM-dd hh:mm");

String now=f.format(d);

int i=userMapper.insertTask(tid, taskName, now, start, end, context, cName, cla);

if(i>=1) {

return true;

}

return false;

}

@Override

public String getTask(String taskName) {

Text task = userMapper.getTask(taskName);

return JSON.toJSONString(task);

}

@Override

public boolean upTask(String taskName, String task) {

int r = userMapper.upTask(taskName,task);

if(r>=1) {

return true;

}

return false;

}

}

### 4.3.4 数据持久层

教师模块的是数据持久层能对课程信息进行增删改查、发布作业。教师模块数据持久层列表见表4-7所示。

表 4-7 教师模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Mapper | 调用数据模型 | 说明 |
| 教师管理课程信息、发布作业 | TaskMapper.java  userMapper.java | 调用CourseTask.Java  Text.java、Teacher.java | 教师对课程信息进行增删改查操作,发布作业 |

TaskMapper.java定义了对课程的增删查改的接口：

package com.gxun.task.mapper;

import java.util.List;

import com.gxun.task.entity.CourseTask;

public interface TaskMapper {

int addOneCourse(String courseName);

CourseTask qryCourseByName(String courseName);

List<CourseTask> qryCourses();

List<String> showClasss();

int delCourse(String cond);

}

userMapper.java定义了发布作业的接口：

package com.gxun.task.mapper;

import java.util.List;

import org.springframework.scheduling.config.Task;

import com.gxun.task.entity.Student;

import com.gxun.task.entity.Teacher;

import com.gxun.task.entity.Text;

public interface UserMapper {

int insertTask(String tid,String taskName,String now,String start,String end,

String context,String cName,String cla);

Text getTask(String taskName);

int upTask(String taskName, String task);

}

### 4.3.5 域模型层

教师模块用到域模型层中CourseTask.java、Teacher.java、Text.Java，CourseTask.java、Teacher.java、Text.Java是一个公用域模型，在涉及到用户信息查询等操作时，就会调用到该模型，教师模块域模型层列表如表4-7所示。

表 4-7教师模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| CourseTask.java | 对课程信息的增、删、改、查操作 |
| Teacher.java | 登录时调用 |
| Text.Java | 发布作业 |

CourseTask.java的主要属性和方法：

package com.gxun.task.entity;

public class CourseTask {

private int id;

private String cName;

private String note;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getcName() {

return cName;

}

public void setcName(String cName) {

this.cName = cName;

}

public String getNote() {

return note;

}

public void setNote(String note) {

this.note = note;

}

}

Text.java的主要属性和方法：

package com.gxun.task.entity;

import java.util.Date ;

import com.mysql.fabric.xmlrpc.base.Data;

public class Text {

private String id;

private String name;

private String publictime;

private String starttime;

private String endtime;

private String requirement;

private String anwser;

private String course;

private String score;

private String cla;

public String getCla() {

return cla;

}

public void setCla(String cla) {

this.cla = cla;

}

public String getScore() {

return score;

}

public void setScore(String score) {

this.score = score;

}

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPublictime() {

return publictime;

}

public void setPublictime(String publictime) {

this.publictime = publictime;

}

public String getStarttime() {

return starttime;

}

public void setStarttime(String starttime) {

this.starttime = starttime;

}

public String getEndtime() {

return endtime;

}

public void setEndtime(String endtime) {

this.endtime = endtime;

}

public String getRequirement() {

return requirement;

}

public void setRequirement(String requirement) {

this.requirement = requirement;

}

public String getAnwser() {

return anwser;

}

public void setAnwser(String anwser) {

this.anwser = anwser;

}

public String getCourse() {

return course;

}

public void setCourse(String course) {

this.course = course;

}

}

## 4.4、学生模块

学生模块的系统内部的相应响应操作示意图如图4-2所示。

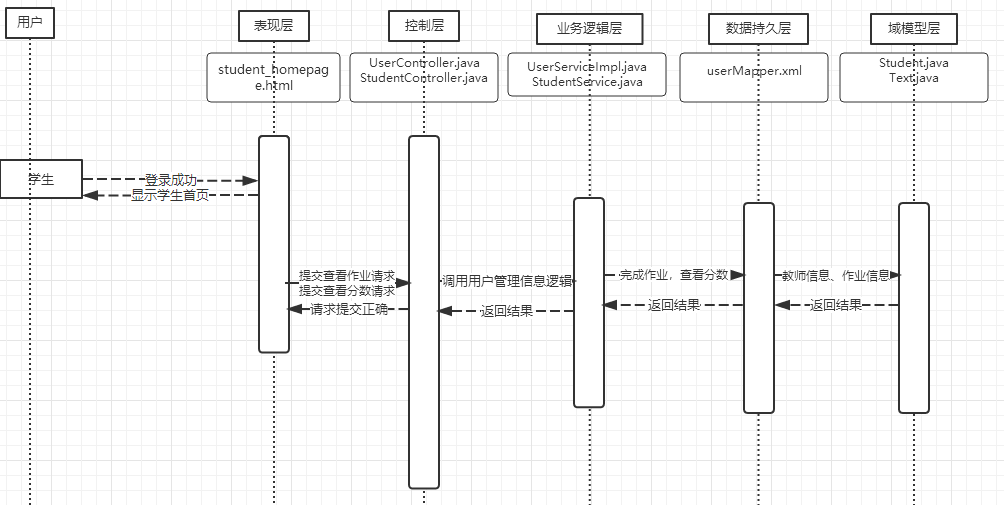


图 4-2 系统响应操作示意图

### 4.4.1 表现层

学生模块的表现层主要完成对作业信息的查看、完成作业、查看已完成的作业的分数，对相应的作业后，显示完成作业的信息在相应的页面上，表现层对应的HTML页面列表见表4-4所示。

表 4-4 学生模块表现层HTML列表

|  |  |  |
| --- | --- | --- |
| 界面 | HTML | 功能描述 |
| 查看作业页面 | student\_homepage.html中JavaScript实现 | 学生对本班的作业进行查看 |
| 查看分数页面 | student\_homepage.html中JavaScript实现 | 学生查看自己做过的作业的分数和查看本次作业的答案 |
| 完成作业页面 | new\_file.html | 选择完成作业 |
| 查看答案页面 | student\_homepage.html中JavaScript实现 | 查看已完成的作业的答案 |

查看作业的JavaScript：

<script>

$(document).ready(function(){

$("#cha").click(function(){

$.post("../user/getSession",{sessionName:"account"} ,

function(msg){

$.ajax({

url:"../stu/select",

type:"post",

data:{msg:msg},

dataType:"json",

success:function(job){

$("tbody").empty();

$("thead").empty();

var tb= $("<tr>");;

tb.append("<td>"+"作业序号"+"</td>");

tb.append("<td>"+"作业名称"+"</td>");

tb.append("<td>"+"作业发布时间"+"</td>");

tb.append("<td>"+"作业结束时间"+"</td>");

tb.append("<td>"+""+"</td>");

$("thead").append(tb);

var k=1;

$.each(job,function(i){

var trObj = $("<tr>");

trObj.append("<td>"+k+"</td>");

trObj.append("<td>"+job[i].name+"</td>");

trObj.append("<td>"+job[i].starttime+"</td>");

trObj.append("<td>"+job[i].endtime+"</td>");

trObj.append("<td><a href='#' id='do\_task'>完成作业</a>"+"</td>");

$("tbody").append(trObj);

k++;

})

}

});

})

})

});

</script>

查看分数的JavaScript:

<script>

$(document).ready(function(){

$("#score").click(function(){

$.post("../user/getSession",{sessionName:"account"} ,

function(msg){

$.ajax({

url:"../stu/selectscore",

type:"post",

data:{msg:msg},

dataType:"json",

success:function(score){

$("tbody").empty();

$("thead").empty();

var tb= $("<tr>");;

tb.append("<td>"+"作业序号"+"</td>");

tb.append("<td>"+"作业名称"+"</td>");

tb.append("<td>"+"作业分数"+"</td>");

tb.append("<td>"+"查看答案"+"</td>");

$("thead").append(tb);

var k=1;

$.each(score,function(i){

var trObj = $("<tr>");

trObj.append("<td>"+k+"</td>");

trObj.append("<td>"+score[i].name+"</td>");

trObj.append("<td>"+score[i].score+"</td>");

trObj.append("<td><a href='#' id='viewAnswer'>查看答案</a>"+"</td>");

$("tbody").append(trObj);

k++;

})

}

});

})

})

});

</script>

查看答案的JavaScript：

<script>

$(document).ready(function(){

$("#wrapper").on("click","#viewAnswer",function(){

var taskName= $(this).parent().siblings().eq(1).text();

$.post("../user/getTask",{taskName:taskName} ,

function(msg){

$("#ans").html(msg.anwser);

$("#ans").toggle();

},"json");

})

});

</script>

学生模块的流程图如图4-4所示。

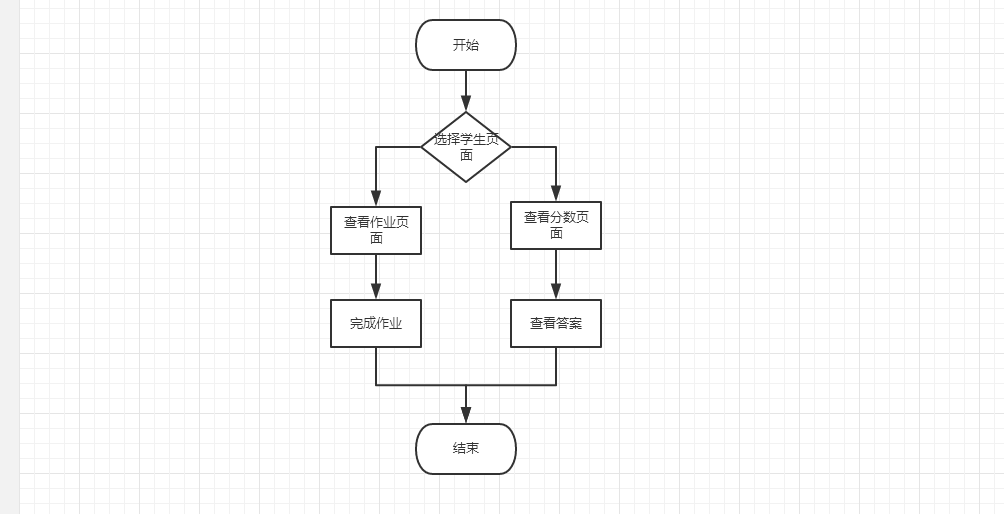


图 4-4 学生模块详细设计流程图

### 4.4.2 控制层

学生模块的控制层负责接受来自student\_homepage.html的完成作业和查看分数，同时调用学生模块的业务逻辑接口，将相对应的作业信息和分数信息传递到业务逻辑层进行操作。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。学生模块控制层列表见表4-5所示。

表 4-5 学生模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 学生查看作业、查看分数、完成作业、查看答案 | StudentController.java  UserController.java | SUCCESS | 相应信息显示在页面上 |
| ERROR |  |

在控制层中StudentController.java、UserController.java的描述分别如下所示：

StudentController.java的主要方法：

package com.gxun.task.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

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

import com.alibaba.fastjson.JSON;

import com.gxun.task.entity.Text;

import com.gxun.task.service.StudentService;

@Controller

@RequestMapping("/stu")

public class StudentController {

@Autowired StudentService studentService;

@ResponseBody

@RequestMapping("/select")

public String select(String msg){

List<Text> list = studentService.selectJob(msg);

return JSON.toJSONString(list);

}

@ResponseBody

@RequestMapping("/selectscore")

public String selectscore(String msg){

List<Text> list = studentService.selectScore(msg);

return JSON.toJSONString(list);

}

}

UserController.java的主要方法：

package com.gxun.task.controller;

import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

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

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

import com.gxun.task.service.IUserService;

@Controller

@RequestMapping("/user")

public class UserController {

@Autowired

IUserService usvr;

@RequestMapping("/lg")

public String login(String account, String pwd,HttpServletRequest reqest){

boolean result = usvr.isValidUser(account, pwd);

if (result){

reqest.getSession().setAttribute("account", account);

return "student\_homepage";

}

return "ng";

}

@RequestMapping("/tealogin")

public String tealogin(String account, String pwd,HttpServletRequest reqest){

boolean result = usvr.isValidteacher(account, pwd);

if (result){

reqest.getSession().setAttribute("techer\_sno", account);

return "teacher\_index";

}

return "ng";

}

@RequestMapping("/getSession")

@ResponseBody

public String getSession(String sessionName, HttpServletRequest request) {

if (request.getSession().getAttribute(sessionName) == null) {

return "ng";

}

return request.getSession().getAttribute(sessionName).toString();

}

@RequestMapping("/addTask")

@ResponseBody

public String addTask(String tid, String taskName,String start, String end, String context,

String cName, String cla) {

boolean i=usvr.insertTask(tid, taskName, start, end, context, cName, cla);

if(i) {

return "ok";

}

return "ng";

}

@RequestMapping("/getTask")

@ResponseBody

public String getTask(String taskName) {

String json = usvr.getTask(taskName);

return json;

}

@RequestMapping("/upTask")

@ResponseBody

public String upTask(String taskName,String task) {

boolean r = usvr.upTask(taskName,task);

if(r) {

return "ok";

}

return "ng";

}

}

### 4.4.3 业务逻辑层

学生模块的业务逻辑层主要完成学生对查看作业、查看分数、完成作业、查看答案的逻辑的判定，同时调用学生模块的业务逻辑接口。比如：学生选择查看作业，对作业进行完成。学生模块业务逻辑层列表如图4-5所示。

表 4-5学生模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 学生查看作业、查看分数、完成作业、查看答案 | UserServiceImpl.java  StudentService.java | 调用IUserService.java  StudentService.java | 返回给Controller |

在学生模块的业务逻辑层是调用了公用的IUserService.java、StudentService.java接口，同时在实现该接口。

IUserService.java接口主要方法：

**public interface IUserService {**

**boolean isValidUser(String account, String pwd);**

**boolean isValidteacher(String account, String pwd);**

**boolean insertTask(String tid,String taskName,String start,String end,**

**String context,String cName,String cla);**

**String getTask(String taskName);**

**boolean upTask(String taskName, String task);**

**}**

StudentService.java接口的主要方法：

import java.util.List;

import com.gxun.task.entity.Text;

public interface StudentService {

public List<Text> selectJob(String msg);

public List<Text> selectScore(String msg);

}

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

UserServiceImpl.java主要属性和方法：

package com.gxun.task.service.impl;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.scheduling.config.Task;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.alibaba.fastjson.JSON;

import com.gxun.task.entity.Student;

import com.gxun.task.entity.Teacher;

import com.gxun.task.entity.Text;

import com.gxun.task.mapper.UserMapper;

import com.gxun.task.service.IUserService;

@Service

@Transactional

public class UserServiceImpl implements IUserService {

@Autowired

UserMapper userMapper;

@Override

public boolean isValidUser(String account, String pwd) {

Student u = userMapper.qryByUnameAndPwd(account, pwd);

if (null!=u){

return true;

}

return false;

}

@Override

public boolean isValidteacher(String account, String pwd) {

Teacher u = userMapper.qryTeacher(account, pwd);

if (null!=u){

return true;

}

return false;

}

@Override

public boolean insertTask(String tid, String taskName, String start, String end, String context,

String cName, String cla) {

Date d=new Date();

SimpleDateFormat f=new SimpleDateFormat("yyyy-MM-dd hh:mm");

String now=f.format(d);

int i=userMapper.insertTask(tid, taskName, now, start, end, context, cName, cla);

if(i>=1) {

return true;

}

return false;

}

@Override

public String getTask(String taskName) {

Text task = userMapper.getTask(taskName);

return JSON.toJSONString(task);

}

@Override

public boolean upTask(String taskName, String task) {

int r = userMapper.upTask(taskName,task);

if(r>=1) {

return true;

}

return false;

}

}

StudentService.java的主要属性和方法：

package com.gxun.task.service.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.gxun.task.entity.Text;

import com.gxun.task.mapper.UserMapper;

@Service

@Transactional

public class StudentService implements com.gxun.task.service.StudentService {

@Autowired

UserMapper userMapper;

@Override

public List<Text> selectJob(String msg) {

List<Text> text = userMapper.selectText(msg);

return text;

}

@Override

public List<Text> selectScore(String msg) {

List<Text> text=userMapper.selectScore(msg);

return text;

}

}

### 4.4.4 数据持久层

学生模块的数据持久层能对需要完成的作业进行查看、已经完成的作业的分数进行查看、需要完成的作业进行完成，对已完成的作业的答案进行查看。学生模块数据持久层列表见表4-7所示。

表 4-7 学生模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Mapper | 调用数据模型 | 说明 |
| 学生查看作业、查看分数、完成作业、查看答案 | UserMapper.java | 调用 Student.Java  Text.java | 学生查看作业、查看分数、完成作业、查看答案 |

UserMapper.java定义了查看作业、查看分数接口：

**package com.gxun.task.mapper;**

**import java.util.List;**

**import org.springframework.scheduling.config.Task;**

**import com.gxun.task.entity.Student;**

**import com.gxun.task.entity.Teacher;**

**import com.gxun.task.entity.Text;**

**public interface UserMapper {**

**Student qryByUnameAndPwd(String account, String pwd);**

**Teacher qryTeacher(String account, String pwd);**

**List<Text> selectText(String msg);**

**List<Text> selectScore(String msg);**

**int insertTask(String tid,String taskName,String now,String start,String end,**

**String context,String cName,String cla);**

**Text getTask(String taskName);**

**int upTask(String taskName, String task);**

**}**

### 4.4.5 域模型层

学生模块用到域模型层中Student.java、Text.Java，Student.java、Text.Java是一个公用域模型，在涉及到用户信息查询等操作时，就会调用到该模型，学生模块域模型层列表如表4-7所示。

表 4-7学生模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Student.java | 登录时调用 |
| Text.Java | 查看作业、查看分数、完成作业、查看答案 |