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

**作业批改系统详细设计**

**V1.0**

**广西民族大学**

评审日期： 2019年6月24日

目录

[1．导言 3](#_Toc13209147)

[1.1 目的 3](#_Toc13209148)

[1.2 范围 4](#_Toc13209149)

[1.3 引用标准 4](#_Toc13209150)

[1.4 参考资料 4](#_Toc13209151)

[1.5 版本更新信息 5](#_Toc13209152)

[2．详细设计简述 5](#_Toc13209153)

[2.1 设计简介 5](#_Toc13209154)

[2.2 模块简介 5](#_Toc13209155)

[3、界面详细设计 6](#_Toc13209156)

[3.1、页面设计说明 6](#_Toc13209157)

[3.2、静态页面详细设计 6](#_Toc13209158)

[4、客户端模块详细设计 7](#_Toc13209159)

[4.1、登录模块（公用模块） 7](#_Toc13209160)

[4.1.1 表现层 7](#_Toc13209161)

[4.1.2 控制层 9](#_Toc13209162)

[4.1.3 业务逻辑层 12](#_Toc13209163)

[4.1.4 数据持久层 13](#_Toc13209164)

[4.1.5 域模型层 14](#_Toc13209165)

[4.2、教师端 15](#_Toc13209166)

[4.2.1发布作业 16](#_Toc13209167)

[4.2.2 题库管理 24](#_Toc13209168)

[4.2.3 批改作业 32](#_Toc13209169)

[4.3、学生端 39](#_Toc13209170)

[4.3.1 写作业、查看分数 39](#_Toc13209171)

[5、管理端模块详细设计 47](#_Toc13209172)

[5.1学生管理 47](#_Toc13209173)

[5.1.1 表现层 47](#_Toc13209174)

[5.1.2 控制层 47](#_Toc13209175)

[5.1.3 业务逻辑层 49](#_Toc13209176)

[5.1.4 数据持久层 51](#_Toc13209177)

[5.1.5 域模型层 53](#_Toc13209178)

[5.2教师管理 54](#_Toc13209179)

[5.2.1 表现层 54](#_Toc13209180)

[5.2.2 控制层 55](#_Toc13209181)

[5.2.3 业务逻辑层 56](#_Toc13209182)

[5.2.4 数据持久层 58](#_Toc13209183)

[5.2.5 域模型层 59](#_Toc13209184)

[5.3 班级管理 61](#_Toc13209185)

[5.3.1 表现层 61](#_Toc13209186)

[5.3.2 控制层 61](#_Toc13209187)

[5.3.3 业务逻辑层 63](#_Toc13209188)

[5.3.4 数据持久层 66](#_Toc13209189)

[5.3.5 域模型层 68](#_Toc13209190)

[6、签字 69](#_Toc13209191)

# 1．导言

## 1.1 目的

该文档是对作业批改系统概要设计进行的详细说明，是为了指导和规范作业批改系统开发而制定的详细开发设计文档。把本系统划分为若干个模块、决定各个模块之间传递的信息，以及数据结构、模块结构的设计等。在以下的详细设计报告中将对本阶段中对系统所做的所有详细设计进行说明。详细设计文档将用文字和程序流程图、体系结构图、实体关系图、时序图、用例图、界面设计以及类图相结合的方式来描述作业批改系统的功能、性能、用户界面、系统结构设计、系统接口设计，模板设计以及针对用户操作的界面设计给出的各种响应。

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

1.2.1 系统目标

开发一个支持我们在校全体师生通过作业批改系统达到方便快捷的完成作业的布置、批改、提交等相关操作的一个网站，改善传统手工收集作业浪费的人力物力。

1.2.2 主要软件需求

开发一个提供以下功能的系统。

1、题库管理

2、作业布置

3、作业提交

4、作业批改

5、分数统计

6、作业情况查阅

7、作业范例学习和点评

## 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

[2] Project2010企业项目管理实践 张会斌 人民邮电出版社。

[3] 需求规格说明书范例

https://wenku.baidu.com/view/b6e5cf0303d8ce2f00662368.html

<https://wenku.baidu.com/view/8962a721b42acfc789eb172ded630b1c59ee9b97.html?>rec\_flag=default&sxts=1529052120879

## 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 | 2019.6.26 | v0.1 | 全部 | 生成目录以及模板 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 2．详细设计简述

## 2.1 设计简介

系统主要是采用B/S模型的方式开发，浏览端使用chrome/ie/firefox，后台使用mysql做为数据库支持。业务处理主要使用后台数据和前台程序配合操作，前后端分离，业务性较强，需要和后台数据进行交互的业务处理编写成存储过程置于后台运行，从而增强整个系统的性能和交互性。

## 2.2 模块简介

系统主要分为前端和后端两个模块，前端是用于界面的展示以及使用，后端则是用来处理用户数据，实现业务逻辑功能，并维护系统等。

# 3、界面详细设计

## 3.1、页面设计说明

3.1.1登录界面

1. 描述：首先通过使用数据库表格中已录入的用户账号和密码登录系统，系统根据账号检索出用户是管理员、教师、学生，然后跳转到改用户对应的界面。
2. 运行结果如下图3-1登录界面，3-2管理员界面，3-3教师界面，3-4学生界面所示

例如，在账号栏输入用户账号“116263000316”，在密码栏中输入密码“000317”，点击登录即可。

3. 在登录界面输入用户账号和密码之后，loginservice会对所输入的用户信息进行正确性验证，若和数据库中“ 。。”表中的数据核对无误则可以进入相应界面，否则显示登录失败。

## 3.2、静态页面详细设计

作业批改系统静态页面见表3-1

表3-1 页面表现层设计

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 4、客户端模块详细设计

客户端模块主要包括登陆模块、教师端、学生端。

## 4.1、登录模块（公用模块）

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。

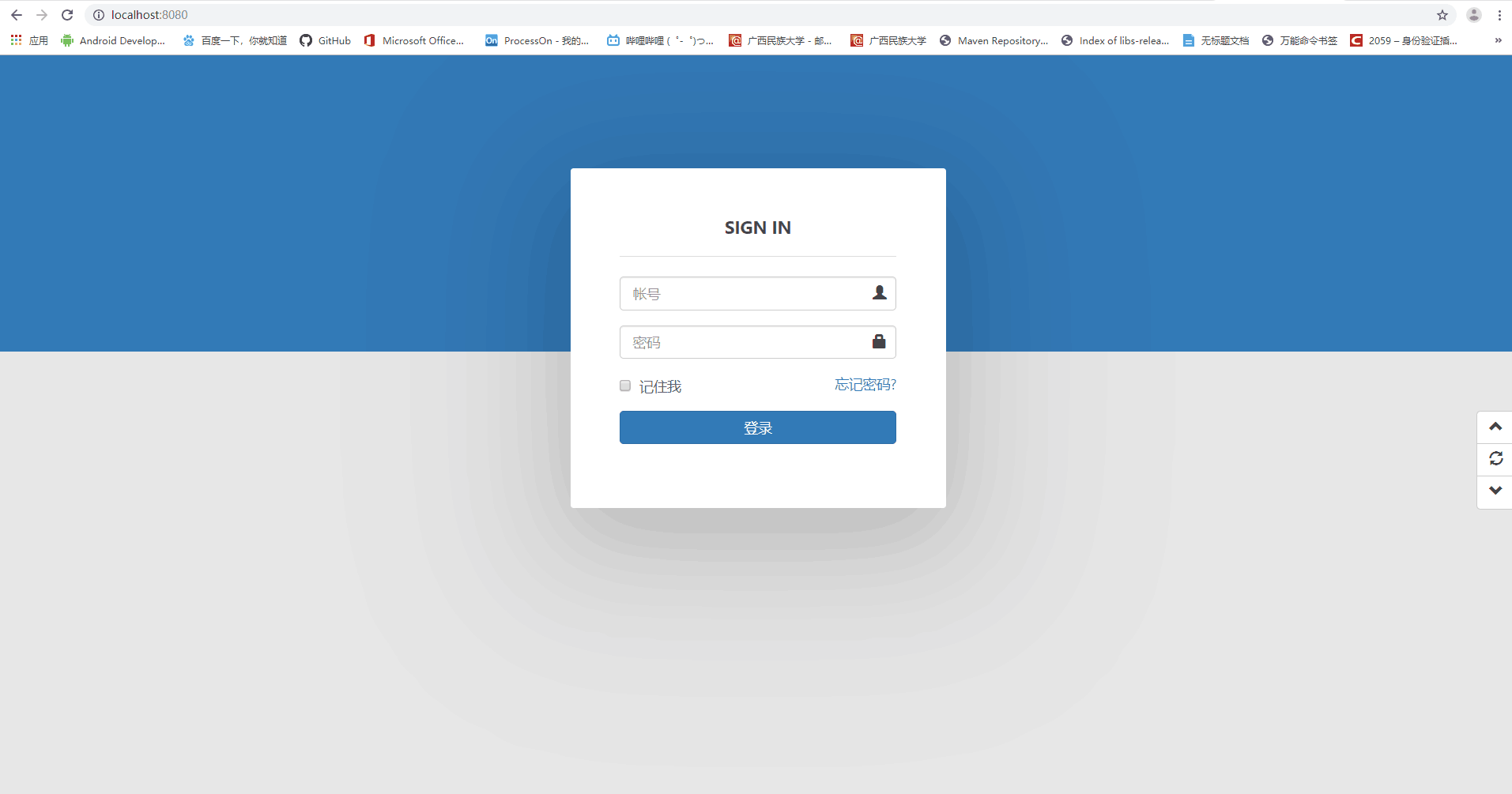


图 4-1 系统响应示意图

### 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JSP页面列表见表4-3所示。

表 4-2登录模块表现层html列表

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

login.html的流程图如图4-2所示。

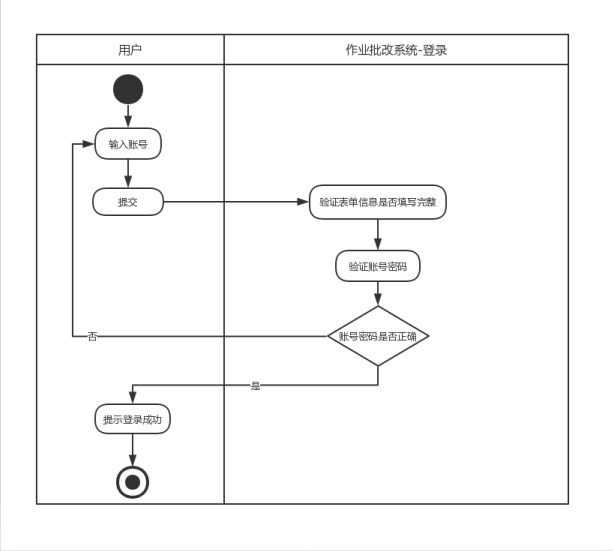
进入游首页

图 4-2 登录模块流程图

login.html的时序图如图4-2所示。

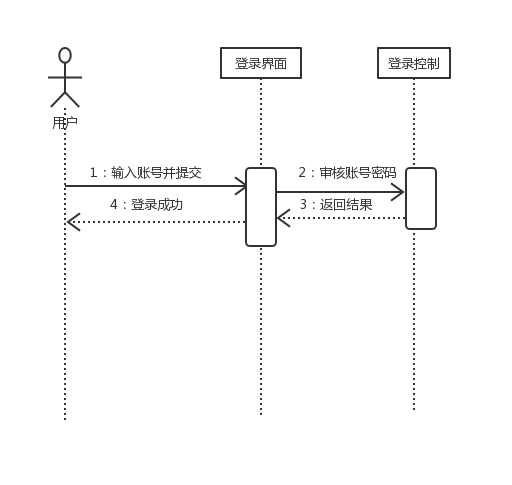


图 4-2 登录模块时序图

### 4.1.2 控制层

登录模块的控制层负责接受来自页面的用户输入，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 4-3 登入模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 登录 | LogController.java | SUCCESS | 登入成功，跳转相应界面。 |
| ERROR | 登入失败，显示提示窗口。 |

在控制层中LogController.java的描述如下所示：

|  |
| --- |
| package gxun.soft.homework\_system.controller;  import gxun.soft.homework\_system.domain.Account;  import gxun.soft.homework\_system.service.LoginService;  import gxun.soft.homework\_system.service.impl.LoginServiceImp;  import io.swagger.annotations.Api;  import io.swagger.annotations.ApiOperation;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.http.HttpRequest;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.RequestParam;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpSession;  @Api(value = "登录")  @Controller  public class LogController {  @Autowired  private LoginService loginService;  */\*\**  *\* 登录*  *\** ***@param***  *\** ***@param***  *\** ***@param***  *\** ***@return***  *\*/*  @ApiOperation(value = "登录")  @PostMapping(value = {"/login"})  public String login(@RequestParam("userId") String userId,  @RequestParam("password") String password,  HttpSession session) {  Account account = new Account();  Integer id = Integer.*parseInt*(userId);  String psw = password;  account.setUserId(id);  account.setPassword(psw);  account = loginService.accountLogin(account);  if (account != null) {  session.setAttribute("userId", userId);  session.setAttribute("password",password);  Integer accountType = account.getAccountType();  switch (accountType){  case 0:  // 获取用户姓名  // String stuName = ...  return "student/workList";  case 1:  return "teacher/workList";  case 2:  return "admin/admin";  default: return "index";  }  } else {  System.*out*.println("账号不存在>>>>>>>重新登录");  return "index";  }  }  @GetMapping(value = {"/logout"})  public String loginout(HttpServletRequest request){  if(request.getSession().getAttribute("user")!=null){  request.getSession().removeAttribute("user");  }  return "login";  }  } |

### 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表4-4所示。

表 4-4登入模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 登录 | LoginServiceImp.java  LoginService.java | 调用LoginMapper.java | 返回给LogController |

LoginService.java接口主要方法：

|  |
| --- |
| package gxun.soft.homework\_system.service;  import gxun.soft.homework\_system.domain.Account;  import org.apache.ibatis.annotations.Param;  import org.springframework.stereotype.Service;  public interface LoginService {  */\*\**  *\* 用户登录*  *\** ***@param*** *account*  *\** ***@return***  *\*/*  Account accountLogin(Account account);  } |

LoginServiceImp.java实现了LoginService.java接口， 同时需要调用数据持久层的LoginMapper.Java，利用@Service来实现对数据持久层接口的调用。

LoginServiceImp.java主要实现属性与方法:

|  |
| --- |
| package gxun.soft.homework\_system.service.impl;  import gxun.soft.homework\_system.domain.Account;  import gxun.soft.homework\_system.mapper.LoginMapper;  import gxun.soft.homework\_system.service.LoginService;  import org.apache.ibatis.annotations.Param;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  @Service  public class LoginServiceImp implements LoginService {  @Autowired  private LoginMapper loginMapper;  */\*\**  *\* 用户登录*  *\** ***@param*** *account*  *\** ***@return***  *\*/*  @Override  public Account accountLogin(@Param("account") Account account){  System.*out*.println("account>>>>>>>>>!!!!!"+account.toString());  account = loginMapper.accountLogin(account);  return account;  }  } |

### 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 4-5登入模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 登录 | loginMapper.java  loginMapper.xml | 调用 account.Java | 对用户信息进行增删改查操作 |

loginMapper.java定义了用户信息进行增、删、改、查的接口：

|  |
| --- |
| package gxun.soft.homework\_system.mapper;  import gxun.soft.homework\_system.domain.Account;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Param;  @Mapper  public interface LoginMapper {  */\*\**  *\* 账户登录*  *\** ***@param*** *account*  *\** ***@return***  *\*/*  Account accountLogin(@Param("account") Account account);  } |

loginMapper.xml是loginMapper.java接口的实现：

|  |
| --- |
| <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "<http://mybatis.org/dtd/mybatis-3-mapper.dtd>">  <mapper namespace="gxun.soft.homework\_system.mapper.LoginMapper">  <!-- 匹配账号密码登录-->  <select id="accountLogin" parameterType="gxun.soft.homework\_system.domain.Account" resultType="gxun.soft.homework\_system.domain.Account">  select \* from account where userId = #{account.userId} and password = #{account.password}  </select>  </mapper> |

### 4.1.5 域模型层

登录模块用到域模型层中User.java，User.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 4-6登入模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Admin.java | 用户登录 |

Admin.java主要属性与方法：

|  |
| --- |
| package gxun.soft.homework\_system.domain;  //import com.baomidou.mybatisplus.annotations.TableName;  //  //@TableName("admin")  public class Admin {  private Integer adminId;  private String adminName;  public Integer getAdminId() {  return adminId;  }  public void setAdminId(Integer adminId) {  this.adminId = adminId;  }  public String getAdminName() {  return adminName;  }  public void setAdminName(String adminName) {  this.adminName = adminName;  }  } |

## 4.2、教师端

教师登录后，系统内部的相应响应操作示意图如图4-7所示。



图4-7 教师模块示意图

### 4.2.1发布作业

#### 4.2.1.1表现层

教师模块的发布作业功能的表现层见表4-8所示。

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 发布作业页面 | workAdd.html | 教师发布作业 |

表4-8

#### 4.2.1.2控制层

控制层是系统提供给外界的功能接口，对外部的请求，首先过滤并校验请求参数，然后才调用业务逻辑接口，执行相应操作，对外部请求做出响应。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 发布作业 |  | SUCCESS | 跳转到。。界面。 |
| ERROR | 跳转页面失败 |

在控制层中StudentManageController.java的描述如下所示：

#### 4.2.1.3 业务逻辑层

业务逻辑层是为控制层提供完成系统功能的接口，主要处理一个业务的逻辑。（批改作业）

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 发布作业 | WorkServiceImp.java  WorkService.java | 调用  WorkMapper.java | 返回给WorkController |

WorkService.java接口主要方法：

|  |
| --- |
| package gxun.soft.homework\_system.service;  import gxun.soft.homework\_system.domain.Work;  import org.apache.ibatis.annotations.Param;  import java.util.Date;  import java.util.List;  public interface WorkService {  */\*\**  *\* 新增作业*  *\** ***@param*** *work*  *\** ***@return***  *\*/*  int addWork(Work work);  int updateWorkNameByWorkId(Integer workId, String workName);  */\*\**  *\* 按workId更改作业结束时间*  *\** ***@param*** *workId*  *\** ***@param*** *endTime*  *\** ***@return***  *\*/*  int updateWorkEndTimeByWorkId(Integer workId,Date endTime);  */\*\**  *\* 按workId删除作业*  *\** ***@param*** *workId*  *\** ***@return***  *\*/*  */\*\**  *\* 按workId删除作业*  *\** ***@param*** *workId*  *\** ***@return***  *\*/*  int deleteWorkByWorkId(Integer workId);  */\*\**  *\* 按workId查找作业*  *\** ***@param*** *workId*  *\** ***@return***  *\*/*  Work findWorkByWorkId(Integer workId);  */\*\**  *\*按teacherId查找作业*  *\** ***@param*** *teaId*  *\** ***@return***  *\*/*  List<Work> finWorkByTeacherId(Integer teaId);  */\*\**  *\* 按workName查找作业*  *\** ***@param*** *workName*  *\** ***@return***  *\*/*  List<Work> findWorkByWorkName(String workName);  */\*\**  *\* 获取所有作业*  *\** ***@return***  *\*/*  List<Work> getAllWorks();  }  } |

WorkServiceImp.java实现WorkService.java接口， 同时需要调用数据持久层的WorkMapper.Java，利用@Service来实现对数据持久层接口的调用。

WorkServiceImp.java主要实现属性与方法:

|  |
| --- |
| package gxun.soft.homework\_system.service.impl;  import gxun.soft.homework\_system.domain.Work;  import gxun.soft.homework\_system.mapper.WorkMapper;  import gxun.soft.homework\_system.service.WorkService;  import org.springframework.beans.factory.annotation.Autowired;  import java.util.Date;  import java.util.List;  public class WorkServiceImp implements WorkService {  @Autowired  WorkMapper workMapper;  @Override  public int addWork(Work work){  return workMapper.addWork(work);  }  @Override  public int updateWorkNameByWorkId(Integer workId, String workName){  return workMapper.updateWorkNameByWorkId(workId,workName);  }  @Override  public int updateWorkEndTimeByWorkId(Integer workId, Date endTime){  return workMapper.updateWorkEndTimeByWorkId(workId,endTime);  }  @Override  public int deleteWorkByWorkId(Integer workId){  return workMapper.deleteWorkByWorkId(workId);  }  @Override  public Work findWorkByWorkId(Integer workId){  return workMapper.findWorkByWorkId(workId);  }  @Override  public List<Work> finWorkByTeacherId(Integer teaId){  return workMapper.finWorkByTeacherId(teaId);  }  @Override  public List<Work> findWorkByWorkName(String workName){  return workMapper.findWorkByWorkName(workName);  }  @Override  public List<Work> getAllWorks(){  return workMapper.getAllWorks();  }  } |

#### 4.2.1.4 数据持久层

数据是一个应用系统的核心，应用系统的所有功能的运行都以来与数据的支持。持久化是指应用系统中运行数据和数据库中的数据之间的交互，主要包括数据的查询、添加、修改、删除等操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 发布作业 | WorkMapper.java  WorkMapper.xml | 调用 Work.Java | 进行作业发布 |

WorkMapper.java定义了教师对作业发布的接口：

|  |
| --- |
| package gxun.soft.homework\_system.mapper;  import gxun.soft.homework\_system.domain.Work;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Param;  import java.util.Date;  import java.util.List;  @Mapper  public interface WorkMapper {  //作业操作  */\*\**  *\* 新增作业*  *\** ***@param*** *work*  *\** ***@return***  *\*/*  int addWork(@Param("work") Work work);  */\*\**  *\* 按workId更改作业名*  *\** ***@param*** *workId*  *\** ***@return***  *\*/*  int updateWorkNameByWorkId(@Param("workId") Integer workId, @Param("workName") String workName);  */\*\**  *\* 按workId更改作业结束时间*  *\** ***@param*** *workId*  *\** ***@param*** *endTime*  *\** ***@return***  *\*/*  int updateWorkEndTimeByWorkId(@Param("workId") Integer workId, @Param("endTime")Date endTime);  */\*\**  *\* 按workId删除作业*  *\** ***@param*** *workId*  *\** ***@return***  *\*/*  int deleteWorkByWorkId(@Param("workId") Integer workId);  */\*\**  *\* 按workId查找作业*  *\** ***@param*** *workId*  *\** ***@return***  *\*/*  Work findWorkByWorkId(@Param("workId") Integer workId);  */\*\**  *\*按teacherId查找作业*  *\** ***@param*** *teaId*  *\** ***@return***  *\*/*  List<Work> finWorkByTeacherId(@Param("teaId") Integer teaId);  */\*\**  *\* 按workName查找作业*  *\** ***@param*** *workName*  *\** ***@return***  *\*/*  List<Work> findWorkByWorkName(@Param("workName") String workName);  */\*\**  *\* 获取所有作业*  *\** ***@return***  *\*/*  List<Work> getAllWorks();  } |

WorkMapper.xml是WorkMapper.java接口的实现：

|  |
| --- |
| <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "<http://mybatis.org/dtd/mybatis-3-mapper.dtd>">  <mapper namespace="gxun.soft.homework\_system.mapper.WorkMapper">  <insert id="addWork" parameterType="gxun.soft.homework\_system.domain.Work">  insert into work (workId, teaId, workName, startTime, endTime, correctAnswerList) values(#{work.workId},#{work.teaId},#{work.workName},#{work.startTime},#{work.endTime},#{work.correctAnswerList})  </insert>  <update id="updateWorkNameByWorkId">  update work set workName = #{workName} where workId = #{workId}  </update>  <update id="updateWorkEndTimeByWorkId">  update work set endTime = #{endTime} where workId = #{workId}  </update>  <delete id="deleteWorkByWorkId" parameterType="int">  delete from work where workId = #{workId}  </delete>  <select id="findWorkByWorkId" parameterType="int" resultType="gxun.soft.homework\_system.domain.Work">  select \* from work where workId = #{workId}  </select>  <select id="finWorkByTeacherId" parameterType="int" resultType="gxun.soft.homework\_system.domain.Work">  select \* from work where teaId = #{teaId}  </select>  <select id="findWorkByWorkName" parameterType="String" resultType="gxun.soft.homework\_system.domain.Work">  select \* from work where workName = #{workName}  </select>  <select id="getAllWorks" resultType="gxun.soft.homework\_system.domain.Work">  select \* from work  </select>  </mapper> |

#### 4.2.1.5 域模型层

发布作业的域模型层是由Work.java组成，Work.java拥有了作业部分的实体been类:

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Work.java | 发布作业的实体been类 |

Work.java的主要属性及方法：

|  |
| --- |
| package gxun.soft.homework\_system.domain;  import springfox.documentation.spring.web.json.Json;  import java.util.Date;  //@TableName("work")  public class Work {  private Integer workId;  private Integer teaId;  private String workName;  private Date startTime;  private Date endTime;  private Json correctAnswerList;  public Integer getWorkId() {  return workId;  }  public void setWorkId(Integer workId) {  this.workId = workId;  }  public Integer getTeaId() {  return teaId;  }  public void setTeaId(Integer teaId) {  this.teaId = teaId;  }  public String getWorkName() {  return workName;  }  public void setWorkName(String workName) {  this.workName = workName;  }  public Date getStartTime() {  return startTime;  }  public void setStartTime(Date startTime) {  this.startTime = startTime;  }  public Date getEndTime() {  return endTime;  }  public void setEndTime(Date endTime) {  this.endTime = endTime;  }  public Json getCorrectAnswerList() {  return correctAnswerList;  }  public void setCorrectAnswerList(Json correctAnswerList) {  this.correctAnswerList = correctAnswerList;  }  } |

### 4.2.2 题库管理

#### 4.2.2.1表现层

教师模块的题库管理功能的表现层见表4-8所示。

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 题库管理 | questionAdd.html  questionList.html | 教师管理题库，对题目进行增删查改 |

表4-8

#### 4.2.2.2 控制层

控制层是系统提供给外界的功能接口，对外部的请求，首先过滤并校验请求参数，然后才调用业务逻辑接口，执行相应操作，对外部请求做出响应。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 题库管理 |  | SUCCESS | 跳转到。。界面。 |
| ERROR | 跳转页面失败 |

在控制层中StudentManageController.java的描述如下所示：

#### 4.2.2.3 业务逻辑层

业务逻辑层是为控制层提供完成系统功能的接口，主要处理一个业务的逻辑。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 题库管理 | QuestionServiceImp.java  QuestionService.java | 调用QuestionMapper.java | 返回给QuestionController |

QuestionService.java接口主要方法：

|  |
| --- |
| package gxun.soft.homework\_system.service;  import gxun.soft.homework\_system.domain.Question;  import java.util.List;  public interface QuestionService {  */\*\**  *\* 为题库新增题目*  *\** ***@param*** *question*  *\** ***@return***  *\*/*  int addQuestion(Question question);  */\*\**  *\* 按questionId删除题目*  *\** ***@param*** *questionId*  *\** ***@return***  *\*/*  int deleteQuestionByQuestionId(Integer questionId);  */\*\**  *\* 按questionId查找题目*  *\** ***@param*** *questionId*  *\** ***@return***  *\*/*  Question findQuestionByQuestionId(Integer questionId);  */\*\**  *\* 按questionType查找题目 0：选择题 1：填空题 2：简答题*  *\** ***@param*** *questionType*  *\** ***@return***  *\*/*  List<Question> getQuestionsByQuestionType(Integer questionType);  */\*\**  *\* 获取所有题目*  *\** ***@return***  *\*/*  List<Question> getAllQuestions();  } |

QuestionServiceImp.java实现了QuestionService.java接口， 同时需要调用数据持久层的QuestionMapper.Java，利用@Service来实现对数据持久层接口的调用。

QuestionServiceImp.java主要实现属性与方法:

|  |
| --- |
| package gxun.soft.homework\_system.service.impl;  import gxun.soft.homework\_system.domain.Question;  import gxun.soft.homework\_system.mapper.QuestionMapper;  import gxun.soft.homework\_system.service.QuestionService;  import org.springframework.beans.factory.annotation.Autowired;  import java.util.List;  public class QuestionServiceImp implements QuestionService {  @Autowired  private QuestionMapper questionMapper;  @Override  public int addQuestion(Question question) {  return questionMapper.addQuestion(question);  }  @Override  public int deleteQuestionByQuestionId(Integer questionId) {  return questionMapper.deleteQuestionByQuestionId(questionId);  }  @Override  public Question findQuestionByQuestionId(Integer questionId) {  return questionMapper.findQuestionByQuestionId(questionId);  }  @Override  public List<Question> getQuestionsByQuestionType(Integer questionType) {  return questionMapper.getQuestionsByQuestionType(questionType);  }  @Override  public List<Question> getAllQuestions() {  return questionMapper.getAllQuestions();  }  } |

#### 4.2.2.4 数据持久层

数据是一个应用系统的核心，应用系统的所有功能的运行都以来与数据的支持。持久化是指应用系统中运行数据和数据库中的数据之间的交互，主要包括数据的查询、添加、修改、删除等操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 题库管理 | QuestionMapper.java  QuestionMapper.xml | 调用 Question.Java | 教师对题库进行管理 |

QuestionMapper.java定义了教师对题库管理的接口：

|  |
| --- |
| package gxun.soft.homework\_system.mapper;  import gxun.soft.homework\_system.domain.Question;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Param;  import java.util.List;  @Mapper  public interface QuestionMapper {  //题目操作  */\*\**  *\* 为题库新增题目*  *\** ***@param*** *question*  *\** ***@return***  *\*/*  int addQuestion(@Param("question") Question question);  */\*\**  *\* 按questionId删除题目*  *\** ***@param*** *questionId*  *\** ***@return***  *\*/*  int deleteQuestionByQuestionId(@Param("questionId") Integer questionId);  */\*\**  *\* 按questionId查找题目*  *\** ***@param*** *questionId*  *\** ***@return***  *\*/*  Question findQuestionByQuestionId(@Param("questionId") Integer questionId);  */\*\**  *\* 按questionType查找题目 0：选择题 1：填空题 2：简答题*  *\** ***@param*** *questionType*  *\** ***@return***  *\*/*  List<Question> getQuestionsByQuestionType(@Param("questionType") Integer questionType);  */\*\**  *\* 按questionType，以及模糊questionName查找题目 0：选择题 1：填空题 2：简答题*  *\** ***@param*** *questionType*  *\** ***@return***  *\*/*  List<Question> getQuestionsByQuestionNameAndType(@Param("questionName") String questionName,@Param("questionType") Integer questionType);  */\*\**  *\* 获取所有题目*  *\** ***@return***  *\*/*  List<Question> getAllQuestions();  } |

QuestionMapper.xml是QuestionMapper.java接口的实现：

|  |
| --- |
| <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "<http://mybatis.org/dtd/mybatis-3-mapper.dtd>">  <mapper namespace="gxun.soft.homework\_system.mapper.QuestionMapper">  <insert id="addQuestion" parameterType="gxun.soft.homework\_system.domain.Question">  insert into question(questionId, questionType, score, details, answer, correctAnswer)  values (#{question.questionId},#{question.questionType},#{question.score},#{question.details},#{question.answer},#{question.correctAnswer})  </insert>  <delete id="deleteQuestionByQuestionId" parameterType="int">  delete from question where questionId = #{questionId}  </delete>  <select id="findQuestionByQuestionId" parameterType="int" resultType="gxun.soft.homework\_system.domain.Question">  select \* from question where questionId = #{questionId}  </select>  <select id="getQuestionsByQuestionType" parameterType="int" resultType="gxun.soft.homework\_system.domain.Question">  select \* from question where questionType = #{questionType}  </select>  <select id="getQuestionsByQuestionNameAndType" resultType="gxun.soft.homework\_system.domain.Question">  select \* from question where questionType = #{questionType} and questionName = "%"#{questionName}"%"  </select>  <select id="getAllQuestions" resultType="gxun.soft.homework\_system.domain.Question">  select \* from question;  </select>  </mapper> |

#### 4.2.2.5 域模型层

作业解答的域模型层是由Question.java组成，Question.java拥有了学生部分的实体been类:

表 4-5学生模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Question.java | 题库的实体been类 |

Question.java的主要属性及方法：

|  |
| --- |
| package gxun.soft.homework\_system.domain;  import springfox.documentation.spring.web.json.Json;  //import com.baomidou.mybatisplus.annotations.TableName;  //  //@TableName("question")  public class Question {  private Integer questionId;  private String questionType;  private Integer score;  private Json details;  private Json answer;  private Json correctAnswer;  public Integer getQuestionId() {  return questionId;  }  public void setQuestionId(Integer questionId) {  this.questionId = questionId;  }  public String getQuestionType() {  return questionType;  }  public void setQuestionType(String questionType) {  this.questionType = questionType;  }  public Integer getScore() {  return score;  }  public void setScore(Integer score) {  this.score = score;  }  public Json getDetails() {  return details;  }  public void setDetails(Json details) {  this.details = details;  }  public Json getAnswer() {  return answer;  }  public void setAnswer(Json answer) {  this.answer = answer;  }  public Json getCorrectAnswer() {  return correctAnswer;  }  public void setCorrectAnswer(Json correctAnswer) {  this.correctAnswer = correctAnswer;  }  } |

### 4.2.3 批改作业

#### 4.2.3.1表现层

教师模块的批改作业功能的表现层见表4-8所示。

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 批改作业 |  | 教师批改作业 |

表4-8

#### 4.2.3.2 控制层

控制层是系统提供给外界的功能接口，对外部的请求，首先过滤并校验请求参数，然后才调用业务逻辑接口，执行相应操作，对外部请求做出响应。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 发布作业 |  | SUCCESS | 跳转到。。界面。 |
| ERROR | 跳转页面失败 |

在控制层中StudentManageController.java的描述如下所示：

#### 4.2.3.3 业务逻辑层

业务逻辑层是为控制层提供完成系统功能的接口，主要处理一个业务的逻辑。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 批改作业 | Stu\_workServiceImp.java  Stu\_workService.java | 调用Stu\_workMapper.java | 返回给Stu\_workController |

Stu\_workService.java接口主要方法：

|  |
| --- |
| package gxun.soft.homework\_system.service;  import gxun.soft.homework\_system.domain.Stu\_work;  import java.util.List;  public interface Stu\_workService {  */\*\**  *\* 新增答题记录*  ***@param*** *stu\_work*  ***@return***  *\*/*  int addStu\_Work(Stu\_work stu\_work);  */\*\**  *\* 更新答题记录,可更新 分数、答题状态、答题答案*  ***@param*** *stu\_work*  ***@return***  *\*/*  int updateStu\_work(Stu\_work stu\_work);  */\*\**  *\*以workId查找学生答题记录*  ***@param*** *workId*  ***@return***  *\*/*  List<Stu\_work> findStu\_workByWorkId(Integer workId);  */\*\**  *\* 按stuId查找答题记录*  ***@param*** *stuId*  ***@return***  *\*/*  List<Stu\_work> findStuWorkByStuId(Integer stuId);  */\*\**  *\* 获取全部答题记录*  ***@return***  *\*/*  List<Stu\_work> getAllStu\_works();  } |

Stu\_workServiceImp.java实现了Stu\_workService.java接口， 同时需要调用数据持久层的Stu\_workMapper.Java，利用@Service来实现对数据持久层接口的调用。

Stu\_workServiceImp.java主要实现属性与方法:

|  |
| --- |
| package gxun.soft.homework\_system.service.impl;  import gxun.soft.homework\_system.domain.Stu\_work;  import gxun.soft.homework\_system.mapper.Stu\_workMapper;  import gxun.soft.homework\_system.service.Stu\_workService;  import org.springframework.beans.factory.annotation.Autowired;  import java.util.List;  public class Stu\_workServiceImp implements Stu\_workService {  @Autowired  Stu\_workMapper stu\_workMapper;  @Override  public int addStu\_Work(Stu\_work stu\_work) {  return stu\_workMapper.addStu\_Work(stu\_work);  }  @Override  public int updateStu\_work(Stu\_work stu\_work) {  return stu\_workMapper.updateStu\_work(stu\_work);  }  @Override  public List<Stu\_work> findStu\_workByWorkId(Integer workId) {  return stu\_workMapper.findStu\_workByWorkId(workId);  }  @Override  public List<Stu\_work> findStuWorkByStuId(Integer stuId) {  return stu\_workMapper.findStuWorkByStuId(stuId);  }  @Override  public List<Stu\_work> getAllStu\_works() {  return stu\_workMapper.getAllStu\_works();  }  } |

#### 4.2.3.4 数据持久层

数据是一个应用系统的核心，应用系统的所有功能的运行都以来与数据的支持。持久化是指应用系统中运行数据和数据库中的数据之间的交互，主要包括数据的查询、添加、修改、删除等操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 批改作业 | Stu\_workMapper.java  Stu\_workMapper.xml | 调用 Stu\_work.Java | 教师批改作业 |

Stu\_workMapper.java定义了教师批改作业的接口：

|  |
| --- |
| package gxun.soft.homework\_system.mapper;  import gxun.soft.homework\_system.domain.Stu\_work;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Param;  import java.util.List;  @Mapper  public interface Stu\_workMapper {  //学生答题情况操作  */\*\**  *\* 新增答题记录*  ***@param*** *stu\_work*  ***@return***  *\*/*  int addStu\_Work(@Param("stu\_work") Stu\_work stu\_work);  */\*\**  *\* 更新答题记录,可更新 分数、答题状态、答题答案*  ***@param*** *stu\_work*  ***@return***  *\*/*  int updateStu\_work(@Param("Stu\_work") Stu\_work stu\_work);  */\*\**  *\*以workId查找学生答题记录*  ***@param*** *workId*  ***@return***  *\*/*  List<Stu\_work> findStu\_workByWorkId(@Param("workId") Integer workId);  */\*\**  *\* 按stuId查找答题记录*  ***@param*** *stuId*  ***@return***  *\*/*  List<Stu\_work> findStuWorkByStuId(@Param("stuId") Integer stuId);  */\*\**  *\* 获取全部答题记录*  ***@return***  *\*/*  List<Stu\_work> getAllStu\_works();  } |

Stu\_workMapper.xml是Stu\_workMapper.java接口的实现：

|  |
| --- |
| <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "<http://mybatis.org/dtd/mybatis-3-mapper.dtd>">  <mapper namespace="gxun.soft.homework\_system.mapper.Stu\_workMapper">  <insert id="addStu\_Work" parameterType="gxun.soft.homework\_system.domain.Stu\_work">  insert into stu\_work(workId, stuId, score, answerList, status) values (#{stu\_work.workId}, #{stu\_work.stuId},#{stu\_work.score},#{stu\_work.answerList},#{stu\_work.status})  </insert>  <update id="updateStu\_work" parameterType="gxun.soft.homework\_system.domain.Stu\_work">  update stu\_work set score = #{stu\_work.score}, answerList = #{stu\_work.answerList}, status = #{stu\_work.status} where workId = #{stu\_work.workId} and stuId = #{stu\_work.stuId}  </update>  <select id="findStu\_workByWorkId" parameterType="int" resultType="gxun.soft.homework\_system.domain.Stu\_work">  select from stu\_work where workId = #{workId}  </select>  <select id="findStuWorkByStuId" parameterType="int" resultType="gxun.soft.homework\_system.domain.Stu\_work">  select from stu\_work where stuId = #{stuId}  </select>  <select id="getAllStu\_works" resultType="gxun.soft.homework\_system.domain.Stu\_work">  select from stu\_work  </select>  </mapper> |

#### 4.2.3.5 域模型层

批改作业的域模型层是由Stu\_work.java组成，Stu\_work.java拥有了学生部分的实体been类:

表 4-5学生模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Stu\_work.java | 批改作业的实体been类 |

Stu\_work.java的主要属性及方法：

|  |
| --- |
| package gxun.soft.homework\_system.domain;  import springfox.documentation.spring.web.json.Json;  //@TableName("stu\_work" )  public class Stu\_work {  private Integer workId;  private Integer stuId;  private Integer score;  private Json answerList;  private Integer status;  public Integer getWorkId() {  return workId;  }  public void setWorkId(Integer workId) {  this.workId = workId;  }  public Integer getStuId() {  return stuId;  }  public void setStuId(Integer stuId) {  this.stuId = stuId;  }  public Integer getScore() {  return score;  }  public void setScore(Integer score) {  this.score = score;  }  public Json getAnswerList() {  return answerList;  }  public void setAnswerList(Json answerList) {  this.answerList = answerList;  }  public Integer getStatus() {  return status;  }  public void setStatus(Integer status) {  this.status = status;  }  } |

## 4.3、学生端

### 4.3.1 写作业、查看分数

学生登录后，系统内部的相应响应操作示意图如图4-4所示。



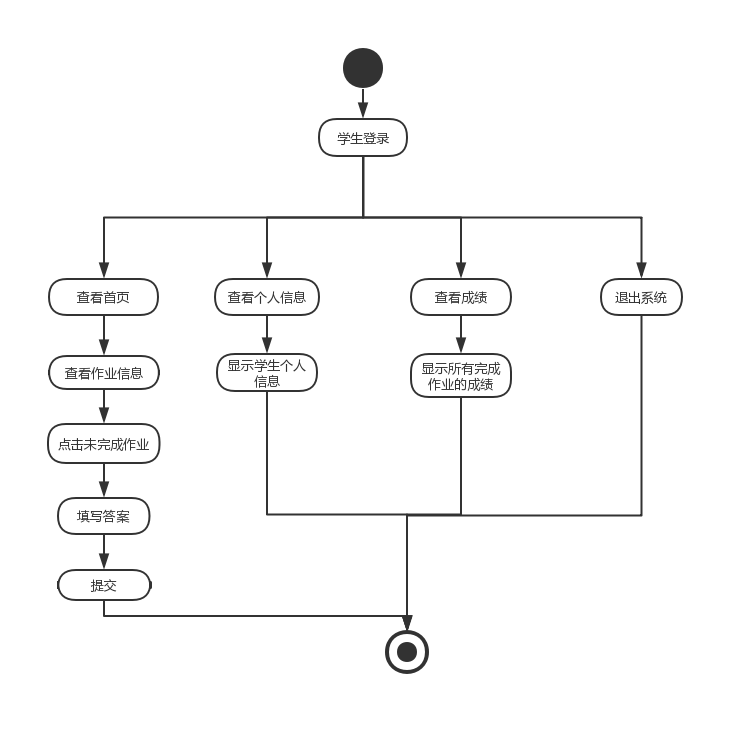
图4-4学生模块示意图

#### 4.3.1.1 表现层

学生模块的表现层主要完成查看首页、成绩和个人信息管理等功能。表现层对应的html页面列表见表4-4所示。

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 学生页面 | gradeList.html | 学生登录后，可以查看个人信息、作业成绩 |

gradeList.html的流程图如图4-4所示。



4-4学生模块流程图

#### 4.3.1.2 控制层

控制层是系统提供给外界的功能接口，对外部的请求，首先过滤并校验请求参数，然后才调用业务逻辑接口，执行相应操作，对外部请求做出响应。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 学生端 |  | SUCCESS | 跳转到。。界面。 |
| ERROR | 跳转页面失败 |

在控制层中StudentManageController.java的描述如下所示：

#### 4.3.1.3 业务逻辑层

业务逻辑层是为控制层提供完成系统功能的接口，主要处理一个业务的逻辑。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 学生端 | Stu\_workServiceImp.java  Stu\_workService.java | 调用Stu\_workMapper.java | 返回给Stu\_workController |

Stu\_workService.java接口主要方法：

|  |
| --- |
| package gxun.soft.homework\_system.service;  import gxun.soft.homework\_system.domain.Stu\_work;  import java.util.List;  public interface Stu\_workService {  */\*\**  *\* 新增答题记录*  *\** ***@param*** *stu\_work*  *\** ***@return***  *\*/*  int addStu\_Work(Stu\_work stu\_work);  */\*\**  *\* 更新答题记录,可更新 分数、答题状态、答题答案*  *\** ***@param*** *stu\_work*  *\** ***@return***  *\*/*  int updateStu\_work(Stu\_work stu\_work);  */\*\**  *\*以workId查找学生答题记录*  *\** ***@param*** *workId*  *\** ***@return***  *\*/*  List<Stu\_work> findStu\_workByWorkId(Integer workId);  */\*\**  *\* 按stuId查找答题记录*  *\** ***@param*** *stuId*  *\** ***@return***  *\*/*  List<Stu\_work> findStuWorkByStuId(Integer stuId);  */\*\**  *\* 获取全部答题记录*  *\** ***@return***  *\*/*  List<Stu\_work> getAllStu\_works();  } |

Stu\_workServiceImp.java实现了Stu\_workService.java接口， 同时需要调用数据持久层的Stu\_workMapper.Java，利用@Service来实现对数据持久层接口的调用。

Stu\_workServiceImp.java主要实现属性与方法:

|  |
| --- |
| package gxun.soft.homework\_system.service.impl;  import gxun.soft.homework\_system.domain.Stu\_work;  import gxun.soft.homework\_system.mapper.Stu\_workMapper;  import gxun.soft.homework\_system.service.Stu\_workService;  import org.springframework.beans.factory.annotation.Autowired;  import java.util.List;  public class Stu\_workServiceImp implements Stu\_workService {  @Autowired  Stu\_workMapper stu\_workMapper;  @Override  public int addStu\_Work(Stu\_work stu\_work) {  return stu\_workMapper.addStu\_Work(stu\_work);  }  @Override  public int updateStu\_work(Stu\_work stu\_work) {  return stu\_workMapper.updateStu\_work(stu\_work);  }  @Override  public List<Stu\_work> findStu\_workByWorkId(Integer workId) {  return stu\_workMapper.findStu\_workByWorkId(workId);  }  @Override  public List<Stu\_work> findStuWorkByStuId(Integer stuId) {  return stu\_workMapper.findStuWorkByStuId(stuId);  }  @Override  public List<Stu\_work> getAllStu\_works() {  return stu\_workMapper.getAllStu\_works();  }  } |

#### 4.3.1.4 数据持久层

数据是一个应用系统的核心，应用系统的所有功能的运行都以来与数据的支持。持久化是指应用系统中运行数据和数据库中的数据之间的交互，主要包括数据的查询、添加、修改、删除等操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 学生端 | Stu\_workMapper.java  Stu\_workMapper.xml | 调用 Stu\_work.Java | 对作业进行解答和提交 |

Stu\_workMapper.java定义了学生写作业和查看分数的接口：

|  |
| --- |
| package gxun.soft.homework\_system.mapper;  import gxun.soft.homework\_system.domain.Stu\_work;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Param;  import java.util.List;  @Mapper  public interface Stu\_workMapper {  //学生答题情况操作  */\*\**  *\* 新增答题记录*  *\** ***@param*** *stu\_work*  *\** ***@return***  *\*/*  int addStu\_Work(@Param("stu\_work") Stu\_work stu\_work);  */\*\**  *\* 更新答题记录,可更新 分数、答题状态、答题答案*  *\** ***@param*** *stu\_work*  *\** ***@return***  *\*/*  int updateStu\_work(@Param("Stu\_work") Stu\_work stu\_work);  */\*\**  *\*以workId查找学生答题记录*  *\** ***@param*** *workId*  *\** ***@return***  *\*/*  List<Stu\_work> findStu\_workByWorkId(@Param("workId") Integer workId);  */\*\**  *\* 按stuId查找答题记录*  *\** ***@param*** *stuId*  *\** ***@return***  *\*/*  List<Stu\_work> findStuWorkByStuId(@Param("stuId") Integer stuId);  */\*\**  *\* 获取全部答题记录*  *\** ***@return***  *\*/*  List<Stu\_work> getAllStu\_works();  } |

Stu\_workMapper.xml是Stu\_workMapper.java接口的实现：

|  |
| --- |
| <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "<http://mybatis.org/dtd/mybatis-3-mapper.dtd>">  <mapper namespace="gxun.soft.homework\_system.mapper.Stu\_workMapper">  <insert id="addStu\_Work" parameterType="gxun.soft.homework\_system.domain.Stu\_work">  insert into stu\_work(workId, stuId, score, answerList, status) values (#{stu\_work.workId}, #{stu\_work.stuId},#{stu\_work.score},#{stu\_work.answerList},#{stu\_work.status})  </insert>  <update id="updateStu\_work" parameterType="gxun.soft.homework\_system.domain.Stu\_work">  update stu\_work set score = #{stu\_work.score}, answerList = #{stu\_work.answerList}, status = #{stu\_work.status} where workId = #{stu\_work.workId} and stuId = #{stu\_work.stuId}  </update>  <select id="findStu\_workByWorkId" parameterType="int" resultType="gxun.soft.homework\_system.domain.Stu\_work">  select \* from stu\_work where workId = #{workId}  </select>  <select id="findStuWorkByStuId" parameterType="int" resultType="gxun.soft.homework\_system.domain.Stu\_work">  select \* from stu\_work where stuId = #{stuId}  </select>  <select id="getAllStu\_works" resultType="gxun.soft.homework\_system.domain.Stu\_work">  select \* from stu\_work  </select>  </mapper> |

#### 4.3.1.5 域模型层

答题的域模型层是由Stu\_work.java组成，Stu\_work.java拥有了学生部分的实体been类:

表 4-5学生模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Stu\_work.java | 作业解答的实体been类 |

Stu\_work.java的主要属性及方法：

|  |
| --- |
| package gxun.soft.homework\_system.domain;  import springfox.documentation.spring.web.json.Json;  //@TableName("stu\_work" )  public class Stu\_work {  private Integer workId;  private Integer stuId;  private Integer score;  private Json answerList;  private Integer status;  public Integer getWorkId() {  return workId;  }  public void setWorkId(Integer workId) {  this.workId = workId;  }  public Integer getStuId() {  return stuId;  }  public void setStuId(Integer stuId) {  this.stuId = stuId;  }  public Integer getScore() {  return score;  }  public void setScore(Integer score) {  this.score = score;  }  public Json getAnswerList() {  return answerList;  }  public void setAnswerList(Json answerList) {  this.answerList = answerList;  }  public Integer getStatus() {  return status;  }  public void setStatus(Integer status) {  this.status = status;  }  } |

# 5、管理端模块详细设计

## 5.1学生管理

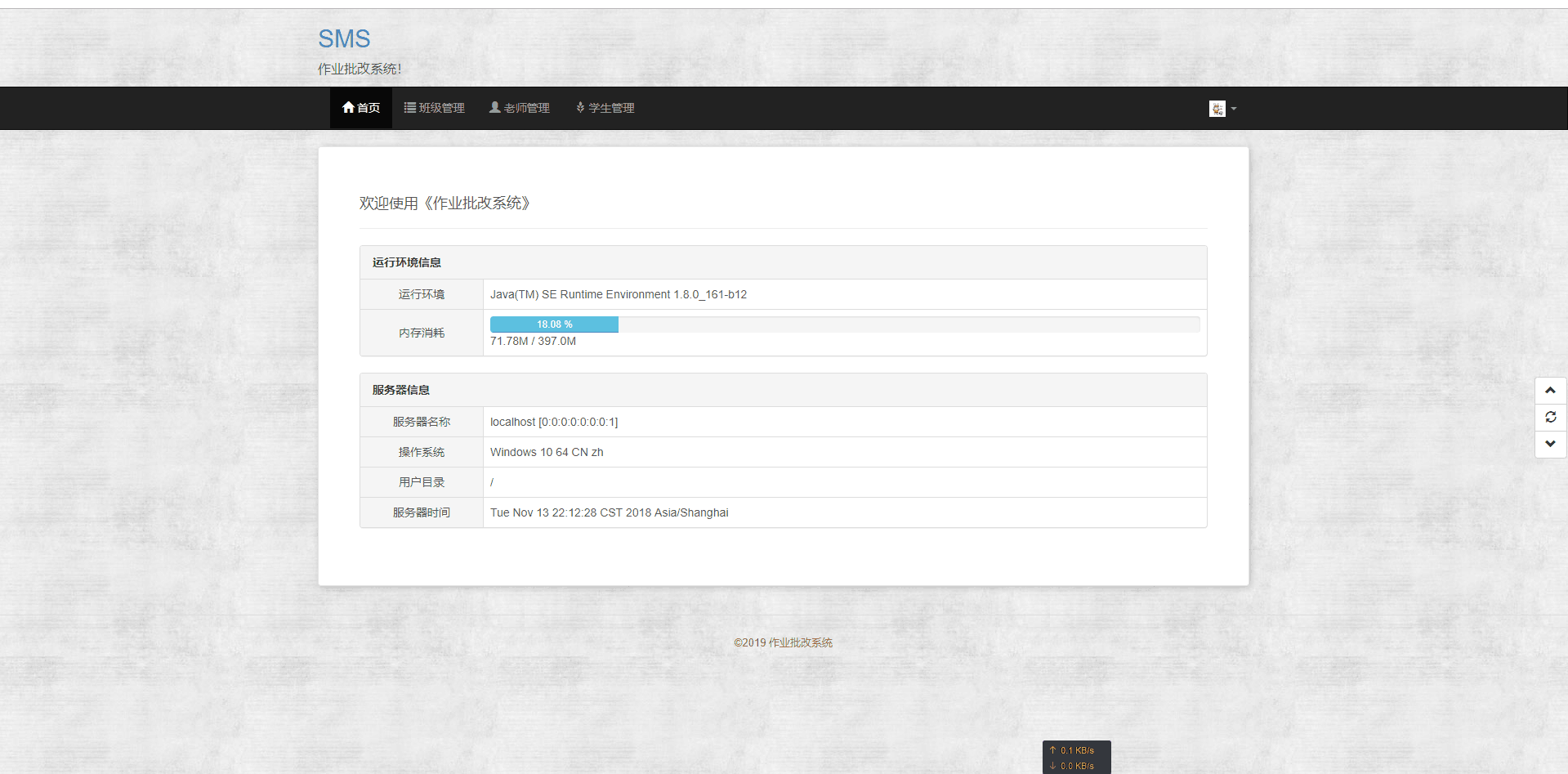
在管理员登录系统时，系统内部的相应响应操作示意图如图5-1所示。 

图 5-1 系统响应示意图

### 5.1.1 表现层

用户登录成功后去到学生管理页面。表现层对应的html页面列表见表5-1所示。

表 5-1 登入模块表现层JSP列表

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 学生管理页面 | studentAdd.html  studentList.html  studentUpdate.html | 管理员对用户信息管理 |

进入游首页

### 5.1.2 控制层

控制层是系统提供给外界的功能接口，对外部的请求，首先过滤并校验请求参数，然后才调用业务逻辑接口，执行相应操作，对外部请求做出响应。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 学生管理 | StudentManageController.java | SUCCESS | 跳转到studentList.html界面。 |
| ERROR | 跳转页面失败 |

在控制层中StudentManageController.java的描述如下所示：

|  |
| --- |
| package gxun.soft.homework\_system.controller.admin;  import com.alibaba.fastjson.JSONArray;  import gxun.soft.homework\_system.component.ToJsonArray;  import gxun.soft.homework\_system.domain.Student;  import gxun.soft.homework\_system.service.StudentService;  import io.swagger.annotations.ApiOperation;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.ResponseBody;  import java.util.List;  @Controller  public class StudentManageController {  @Autowired  private StudentService studentService;  @ApiOperation(value = "学生管理")  @GetMapping("/studentList")  public String classManage(Model model){  List<Student> studentList = studentService.getAllStudent();  model.addAttribute("studentList",studentList);  return "admin/studentList";  }  } |

### 5.1.3 业务逻辑层

业务逻辑层是为控制层提供完成系统功能的接口，主要处理一个业务的逻辑。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 学生管理 | StudentServiceImp.java  StudentService.java | 调用StudentMapper.java  AccountMapper.java | 返回给StudentController |

StudentService.java接口主要方法：

|  |
| --- |
| package gxun.soft.homework\_system.service;  import gxun.soft.homework\_system.domain.Account;  import gxun.soft.homework\_system.domain.Student;  import java.util.List;  import java.util.Map;  public interface StudentService {  int addStudent(Map studentMap);  int updateStudentPassword(Account account);  int deleteStudentByStudentId(Integer stuId);  Student findStudentByStudentId(Integer stuId);  List<Student> findStudentByStudentName(String stuName);  List<Student> findStudentByClassId(Integer classId);  List<Student> findStudentByClassName(String className);  List<Student> getAllStudent();  } |

StudentServiceImp.java实现了StudentService.java接口， 同时需要调用数据持久层的StudentMapper.Java，AccountMapper.java,利用@Service来实现对数据持久层接口的调用。

StudentServiceImp.java主要实现属性与方法:

|  |
| --- |
| package gxun.soft.homework\_system.service.impl;  import gxun.soft.homework\_system.domain.Account;  import gxun.soft.homework\_system.domain.Stu\_work;  import gxun.soft.homework\_system.domain.Student;  import gxun.soft.homework\_system.mapper.AccountMapper;  import gxun.soft.homework\_system.mapper.StudentMapper;  import gxun.soft.homework\_system.service.StudentService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.transaction.annotation.Transactional;  import java.util.List;  import java.util.Map;  public class StudentServiceImp implements StudentService {  @Autowired  StudentMapper studentMapper;  AccountMapper accountMapper;  @Override  @Transactional  public int addStudent(Map studenMap) {  Student student = (Student) studenMap.get("student");  Account account = (Account) studenMap.get("account");  accountMapper.addAccount(account);  studentMapper.addStudent(student);  return 0;  }  @Override  public int updateStudentPassword(Account account) {  return accountMapper.updatePassword(account);  }  @Override  public int deleteStudentByStudentId(Integer stuId) {  return studentMapper.deleteStudentByStudentId(stuId);  }  @Override  public Student findStudentByStudentId(Integer stuId) {  return studentMapper.findStudentByStudentId(stuId);  }  @Override  public List<Student> findStudentByStudentName(String stuName) {  return studentMapper.findStudentByStudentName(stuName);  }  @Override  public List<Student> findStudentByClassId(Integer classId) {  return studentMapper.findStudentByClassId(classId);  }  @Override  public List<Student> findStudentByClassName(String className) {  return studentMapper.findStudentByClassName(className);  }  @Override  public List<Student> getAllStudent() {  return studentMapper.getAllStudent();  }  } |

### 5.1.4 数据持久层

数据是一个应用系统的核心，应用系统的所有功能的运行都以来与数据的支持。持久化是指应用系统中运行数据和数据库中的数据之间的交互，主要包括数据的查询、添加、修改、删除等操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 学生管理 | StudentMapper.java  StudentMapper.xml | 调用 Student.Java | 对学生信息进行增删查改 |

StudentMapper.java定义了对学生信息进行增、删、改、查、作业提交的接口：

|  |
| --- |
| package gxun.soft.homework\_system.mapper;  import gxun.soft.homework\_system.domain.Student;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Param;  import java.util.List;  @Mapper  public interface StudentMapper {  //学生操作  */\*\**  *\* 添加学生*  ***@param*** *student*  ***@return***  *\*/*  int addStudent(@Param("student") Student student);  int deleteStudentByStudentId(@Param("stuId") Integer stuId);  Student findStudentByStudentId(@Param("stuId") Integer stuId);  List<Student> findStudentByStudentName(@Param("stuName") String stuName);  List<Student> findStudentByClassId(@Param("classId") Integer classId);  List<Student> findStudentByClassName(@Param("className") String className);  List<Student> getAllStudent();  } |

StudentMapper.xml是StudentMapper.java接口的实现：

|  |
| --- |
| <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "<http://mybatis.org/dtd/mybatis-3-mapper.dtd>">  <mapper namespace="gxun.soft.homework\_system.mapper.StudentMapper">  <insert id="addStudent" parameterType="gxun.soft.homework\_system.domain.Student">  insert into student(stuId, classId, stuName) values (#{student.stuId},#{student.classId},#{student.stuName})  </insert>  <delete id="deleteStudentByStudentId" parameterType="int">  delete from account where userId = #{stuId}  </delete>  <select id="findStudentByStudentId" parameterType="int" resultType="gxun.soft.homework\_system.domain.Student">  select from student where stuId = #{stuId}  </select>  <select id="findStudentByStudentName" parameterType="string" resultType="gxun.soft.homework\_system.domain.Student">  select from student where stuName = #{stuName}  </select>  <select id="findStudentByClassId" parameterType="int" resultType="gxun.soft.homework\_system.domain.Student">  select from student where classId = #{classId}  </select>  <select id="findStudentByClassName" parameterType="string" resultType="gxun.soft.homework\_system.domain.Student">  select from student where classId in {  select classId from class where className = #{className}  }  </select>  <select id="getAllStudent" resultType="gxun.soft.homework\_system.domain.Student">  select from student  </select>  </mapper> |

### 5.1.5 域模型层

管理端的域模型层由Student.java组成，里面封装了管理端的实体been类:

表5-7登入模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Student.java | 学生的实体been类 |

Student.java的属性如下所示：

|  |
| --- |
| package gxun.soft.homework\_system.domain;  //import com.baomidou.mybatisplus.annotations.TableName;  //  //@TableName("student")  public class Student {  private Integer stuId;  private Integer classId;  private String stuName;  public Integer getStuId() {  return stuId;  }  public void setStuId(Integer stuId) {  this.stuId = stuId;  }  public String getStuName() {  return stuName;  }  public void setStuName(String stuName) {  this.stuName = stuName;  }  public Integer getClassId() {  return classId;  }  public void setClassId(Integer classId) {  this.classId = classId;  }  } |

## 5.2教师管理

### 5.2.1 表现层

用户登录成功后去到教师管理页面。表现层对应的html页面列表见表5-1所示。

表 5-1 登入模块表现层JSP列表

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 教师管理页面 | teacherAdd.html  teacherList.html  teacherUpdate.html | 管理员对教师信息管理 |

游

### 5.2.2 控制层

控制层是系统提供给外界的功能接口，对外部的请求，首先过滤并校验请求参数，然后才调用业务逻辑接口，执行相应操作，对外部请求做出响应。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 教师管理 | TeacherManageController.java | SUCCESS | 跳转到teacherList.html界面。 |
| ERROR | 跳转页面失败 |

在控制层中TeacherManageController.java的描述如下所示：

|  |
| --- |
| package gxun.soft.homework\_system.controller.admin;  import gxun.soft.homework\_system.domain.Teacher;  import gxun.soft.homework\_system.service.TeacherService;  import io.swagger.annotations.ApiOperation;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.GetMapping;  import java.util.List;  @Controller  public class TeacherManageController {  @Autowired  private TeacherService teacherService;  @ApiOperation(value = "教师管理")  @GetMapping("/teacherList")  public String getClassList(Model model){  List<Teacher> teacherList = teacherService.getAllTeacher();  model.addAttribute("teacherList",teacherList);  return "admin/teacherList";  }  } |

### 5.2.3 业务逻辑层

业务逻辑层是为控制层提供完成系统功能的接口，主要处理一个业务的逻辑。（批改作业）

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 教师管理 | TeacherServiceImp.java  TeacherService.java | 调用TeacherMapper.java  AccountMapper.java | 返回给TeacherController |

TeacherService.java接口主要方法：

|  |
| --- |
| package gxun.soft.homework\_system.service;  import gxun.soft.homework\_system.domain.Account;  import gxun.soft.homework\_system.domain.Teacher;  import java.util.List;  import java.util.Map;  public interface TeacherService {  int addTeacher(Map teacherMap);  int updateTeacherPassword(Account account);  int deleteTeacherByTeacherId(Integer teaId);  Teacher findTeacherByTeacherId(Integer teaId);  List<Teacher> findTeacherByTeacherName(String teaName);  List<Teacher> findTeacherByClassId(Integer classId);  List<Teacher> getAllTeacher();  } |

TeacherServiceImp.java实现了TeacherService.java接口， 同时需要调用数据持久层的TeacherMapper.Java，AccountMapper.java,利用@Service来实现对数据持久层接口的调用。

TeacherServiceImp.java主要实现属性与方法:

|  |
| --- |
| package gxun.soft.homework\_system.service.impl;  import gxun.soft.homework\_system.domain.Account;  import gxun.soft.homework\_system.domain.Teacher;  import gxun.soft.homework\_system.mapper.AccountMapper;  import gxun.soft.homework\_system.mapper.TeacherMapper;  import gxun.soft.homework\_system.service.TeacherService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.transaction.annotation.Transactional;  import java.util.List;  import java.util.Map;  public class TeacherServiceImp implements TeacherService {  @Autowired  TeacherMapper teacherMapper;  AccountMapper accountMapper;  @Override  @Transactional  public int addTeacher(Map teacherMap) {  Account account = (Account) teacherMap.get("account");  Teacher teacher = (Teacher) teacherMap.get("teacher");  accountMapper.addAccount(account);  teacherMapper.addTeacher(teacher);  return 0;  }  @Override  public int updateTeacherPassword(Account account) {  return accountMapper.updatePassword(account);  }  @Override  public int deleteTeacherByTeacherId(Integer teaId) {  return teacherMapper.deleteTeacherByTeacherId(teaId);  }  @Override  public Teacher findTeacherByTeacherId(Integer teaId) {  return teacherMapper.findTeacherByTeacherId(teaId);  }  @Override  public List<Teacher> findTeacherByTeacherName(String teaName) {  return teacherMapper.findTeacherByTeacherName(teaName);  }  @Override  public List<Teacher> findTeacherByClassId(Integer classId) {  return teacherMapper.findTeacherByClassId(classId);  }  @Override  public List<Teacher> getAllTeacher() {  return teacherMapper.getAllTeacher();  }  } |

### 5.2.4 数据持久层

数据是一个应用系统的核心，应用系统的所有功能的运行都以来与数据的支持。持久化是指应用系统中运行数据和数据库中的数据之间的交互，主要包括数据的查询、添加、修改、删除等操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 教师管理 | TeacherMapper.java  TeacherMapper.xml | 调用 Teacher.Java | 对教师信息进行增删查改 |

TeacherMapper.java定义了对教师信息进行增、删、改、查、作业发布和批改的接口：

|  |
| --- |
| package gxun.soft.homework\_system.mapper;  import gxun.soft.homework\_system.domain.Teacher;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Param;  import java.util.List;  @Mapper  public interface TeacherMapper {  //教师操作  int addTeacher(@Param("teacher") Teacher teacher);  int deleteTeacherByTeacherId(@Param("teaId") Integer teaId);  Teacher findTeacherByTeacherId(@Param("teaId") Integer teaId);  List<Teacher> findTeacherByTeacherName(@Param("teaName") String teaName);  List<Teacher> findTeacherByClassId(@Param("classId") Integer classId);  List<Teacher> getAllTeacher();  } |

TeacherMapper.xml是TeacherMapper.java接口的实现：

|  |
| --- |
| <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "<http://mybatis.org/dtd/mybatis-3-mapper.dtd>">  <mapper namespace="gxun.soft.homework\_system.mapper.LoginMapper">  <!-- 匹配账号密码登录-->  <select id="accountLogin" parameterType="gxun.soft.homework\_system.domain.Account" resultType="gxun.soft.homework\_system.domain.Account">  select from account where userId = #{account.userId} and password = #{account.password}  </select>  </mapper> |

### 5.2.5 域模型层

管理端的域模型层由Teacher.java组成，里面封装了管理端的实体been类:

表5-7登入模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| Teacher.java | 管理端的实体been类 |

Teacher.java的属性如下所示：

|  |
| --- |
| package gxun.soft.homework\_system.domain;  //import com.baomidou.mybatisplus.annotations.TableName;  //  //@TableName("teacher")  public class Teacher {  private Integer teaId;  private Integer classId;  private String teaName;  public Integer getTeaId() {  return teaId;  }  public void setTeaId(Integer teaId) {  this.teaId = teaId;  }  public String getTeaName() {  return teaName;  }  public void setTeaName(String teaName) {  this.teaName = teaName;  }  public Integer getClassId() {  return classId;  }  public void setClassId(Integer classId) {  this.classId = classId;  }  } |

## 5.3 班级管理

### 5.3.1 表现层

用户登录成功后去到班级管理页面。表现层对应的html页面列表见表5-1所示。

表 5-1 登入模块表现层JSP列表

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 班级管理页面 | classAdd.html  classList.html  classUpdate.html | 管理员对班级管理 |

### 5.3.2 控制层

控制层是系统提供给外界的功能接口，对外部的请求，首先过滤并校验请求参数，然后才调用业务逻辑接口，执行相应操作，对外部请求做出响应。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 班级管理 | ClassManageController.java | SUCCESS | 跳转到classList.html界面。 |
| ERROR | 跳转页面失败 |

在控制层中ClassManageController.java的描述如下所示：

|  |
| --- |
| package gxun.soft.homework\_system.controller.admin;  import com.alibaba.fastjson.JSON;  import com.alibaba.fastjson.JSONArray;  import gxun.soft.homework\_system.component.ToJsonArray;  import gxun.soft.homework\_system.domain.MyClass;  import gxun.soft.homework\_system.service.MyClassService;  import io.swagger.annotations.ApiOperation;  import org.apache.ibatis.annotations.Param;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.RequestParam;  import org.springframework.web.bind.annotation.ResponseBody;  import java.util.List;  @Controller  public class ClassManageController {  @Autowired  private MyClassService myClassService;  @ApiOperation(value = "班级列表")  @GetMapping("/classList")  public String getClassList(Model model){  List<MyClass> myClassList = myClassService.getAllClasses();  model.addAttribute("myClassList", myClassList);  return "admin/classList";  }  @ApiOperation(value = "班级查询")  @GetMapping("/findByClassName")  public String findByClassName(@Param("className")String className, Model model){  List<MyClass> myClassList = myClassService.findClassByClassName(className);  model.addAttribute("myClassList", myClassList);  return "admin/classList";  }  @ApiOperation(value = "班级添加")  @GetMapping("/addClass")  public String findByClassName(@Param("className")String className,@Param("classId") Integer classId,  Model model){  MyClass myClass = new MyClass();  myClass.setClassId(classId);  myClass.setClassName(className);  myClassService.addClass(myClass);  List<MyClass> myClassList = myClassService.getAllClasses();  model.addAttribute("myClassList", myClassList);  return "admin/classList";  }  @ApiOperation(value = "班级删除")  @GetMapping("/deleteClass")  public String findByClassName(@Param("classId") Integer classId,  Model model){  myClassService.deleteClassByClassId(classId);  List<MyClass> myClassList = myClassService.getAllClasses();  model.addAttribute("myClassList", myClassList);  return "admin/classList";  }  } |

### 5.3.3 业务逻辑层

业务逻辑层是为控制层提供完成系统功能的接口，主要处理一个业务的逻辑。（批改作业）

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 教师端 | MyClassServiceImp.java  MyClassService.java | 调用  ClassMapper.java | 返回给ClassManageController |

MyClassService.java接口主要方法：

|  |
| --- |
| package gxun.soft.homework\_system.service;  import gxun.soft.homework\_system.domain.MyClass;  import java.util.List;  public interface MyClassService {  */\*\**  *\* 新增班级*  *\** ***@param*** *myClass*  *\** ***@return***  *\*/*  int addClass(MyClass myClass);  */\*\**  *\* 按classId删除班级*  *\** ***@param*** *classId*  *\** ***@return***  *\*/*  int deleteClassByClassId(Integer classId);  */\*\**  *\* 按classId查找班级*  *\** ***@param*** *classId*  *\** ***@return***  *\*/*  MyClass findClassByClassId(Integer classId);  */\*\**  *\* 按班级名查找班级*  *\** ***@param*** *className*  *\** ***@return***  *\*/*  List<MyClass> findClassByClassName(String className);  */\*\**  *\* 获取所有班级*  *\** ***@return***  *\*/*  List<MyClass> getAllClasses();  } |

MyClassServiceImp.java实现了MyClassService.java接口， 同时需要调用数据持久层的ClassMapper.Java，利用@Service来实现对数据持久层接口的调用。

MyClassServiceImp.java主要实现属性与方法:

|  |
| --- |
| package gxun.soft.homework\_system.service.impl;  import gxun.soft.homework\_system.domain.MyClass;  import gxun.soft.homework\_system.mapper.ClassMapper;  import gxun.soft.homework\_system.service.MyClassService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import java.util.List;  @Service  public class MyClassServiceImp implements MyClassService {  @Autowired  private ClassMapper classMapper;  @Override  public int addClass(MyClass myClass) {  return classMapper.addClass(myClass);  }  @Override  public int deleteClassByClassId(Integer classId) {  return classMapper.deleteClassByClassId(classId);  }  @Override  public MyClass findClassByClassId(Integer classId) {  return classMapper.findClassByClassId(classId);  }  @Override  public List<MyClass> findClassByClassName(String className) {  return classMapper.findClassByClassName(className);  }  @Override  public List<MyClass> getAllClasses() {  return classMapper.getAllClasses();  }  } |

### 5.3.4 数据持久层

数据是一个应用系统的核心，应用系统的所有功能的运行都以来与数据的支持。持久化是指应用系统中运行数据和数据库中的数据之间的交互，主要包括数据的查询、添加、修改、删除等操作。

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 班级管理 | ClassMapper.java  ClassMapper.xml | 调用 Myclass.Java | 对班级信息进行增删查改 |

ClassMapper.java定义了对教师信息进行增、删、改、查、作业发布和批改的接口：

|  |
| --- |
| package gxun.soft.homework\_system.mapper;  import gxun.soft.homework\_system.domain.MyClass;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Param;  import java.util.List;  @Mapper  public interface ClassMapper {  //班级操作  */\*\**  *\* 新增班级*  *\** ***@param*** *myClass*  *\** ***@return***  *\*/*  int addClass(@Param("myClass") MyClass myClass);  */\*\**  *\* 按classId删除班级*  *\** ***@param*** *classId*  *\** ***@return***  *\*/*  int deleteClassByClassId(@Param("classId") Integer classId);  */\*\**  *\* 按classId查找班级*  *\** ***@param*** *classId*  *\** ***@return***  *\*/*  MyClass findClassByClassId(@Param("classId") Integer classId);  */\*\**  *\* 按className模糊查找班级*  *\** ***@param*** *className*  *\** ***@return***  *\*/*  List<MyClass> findClassByClassName(@Param("className") String className);  */\*\**  *\* 获取所有班级*  *\** ***@return***  *\*/*  List<MyClass> getAllClasses();  } |

ClassMapper.xml是ClassMapper.java接口的实现：

|  |
| --- |
| <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "<http://mybatis.org/dtd/mybatis-3-mapper.dtd>">  <mapper namespace="gxun.soft.homework\_system.mapper.ClassMapper">  <!-- 新增班级-->  <insert id="addClass" parameterType="gxun.soft.homework\_system.domain.MyClass">  insert into class(classId, className) values (#myClass.classId, #myClass.className)  </insert>  <!-- 按班级id删除班级-->  <delete id="deleteClassByClassId" parameterType="int">  delete from class where classId = #{classId}  </delete>  <!-- 按班级id查找班级-->  <select id="findClassByClassId" parameterType="int" resultType="gxun.soft.homework\_system.domain.MyClass">  select \* from class where classId = classId = #{classId}  </select>  <!-- 按班级名称查找班级-->  <select id="findClassByClassName" parameterType="string" resultType="gxun.soft.homework\_system.domain.MyClass">  select \* from class where className like "%"#{className}"%"  </select>  <!-- 查看所有班级-->  <select id="getAllClasses" resultType="gxun.soft.homework\_system.domain.MyClass">  select \* from class  </select>  </mapper> |

### 5.3.5 域模型层

管理端的域模型层由Teacher.java组成，里面封装了管理端的实体been类:

表5-7登入模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| MyClass.java | 班级的实体been类 |

MyClass.java的属性如下所示：

|  |
| --- |
| package gxun.soft.homework\_system.domain;  //import com.baomidou.mybatisplus.annotations.TableName;  //  //@TableName("class")  public class MyClass {  private Integer classId;  private String className;  public Integer getClassId() {  return classId;  }  public void setClassId(Integer classId) {  this.classId = classId;  }  public String getClassName() {  return className;  }  public void setClassName(String className) {  this.className = className;  }  @Override  public String toString() {  return "Class{" +  "classId=" + classId +  ", className='" + className + '\'' +  '}';  }  } |

## 6、签字

本详细设计经过双方认可，特签字如下表 8-1 所示。

|  |  |  |  |
| --- | --- | --- | --- |
| **用户签署信息** | | **企业签署信息** | |
| **单位名称** |  | **单位名称** | **广西民族大学** |
| **签署人姓名** |  | **签署人姓名** |  |
| **签署日期** | **2019.7** | **签署日期** | **2019.7** |