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

**课程作业管理详细设计**

**V2.0**

**广西民族大学**

评审日期： 2018年7月14日

目录

[1．导言 4](#_Toc519352015)

[1.1 目的 4](#_Toc519352016)

[1.2 范围 4](#_Toc519352017)

[1.3 引用标准 4](#_Toc519352018)

[1.4 参考资料 4](#_Toc519352019)

[1.5 版本更新信息 4](#_Toc519352020)

[2．详细设计简述 5](#_Toc519352021)

[3、界面详细设计 15](#_Toc519352022)

[3.1、页面设计说明 16](#_Toc519352023)

[3.2、静态页面详细设计 16](#_Toc519352024)

[4、客户端模块详细设计 17](#_Toc519352025)

[4.1、公用模块的详细设计 17](#_Toc519352026)

[4.1.1 表现层 17](#_Toc519352027)

[4.1.2 控制层 18](#_Toc519352028)

[4.1.3 数据持久层 42](#_Toc519352029)

[4.2、登录模块 145](#_Toc519352030)

[4.2.1 表现层 145](#_Toc519352031)

[4.2.2 控制层 147](#_Toc519352032)

[4.2.3 业务逻辑层 148](#_Toc519352033)

[4.2.4 数据持久层 150](#_Toc519352034)

[4.2.5 域模型层 153](#_Toc519352035)

[4.3、发布作业模块 155](#_Toc519352036)

[4.3.1 表现层 156](#_Toc519352037)

[4.3.2 控制层 157](#_Toc519352038)

[4.3.3 业务逻辑层 158](#_Toc519352039)

[4.3.4 数据持久层 170](#_Toc519352040)

[4.3.5 域模型层 170](#_Toc519352041)

[4.4、学生提交作业 188](#_Toc519352042)

[4.4.1 表现层 189](#_Toc519352043)

[4.4.2 控制层 192](#_Toc519352044)

[4.4.3 业务逻辑层 192](#_Toc519352045)

[4.4.4 数据持久层 200](#_Toc519352046)

[4.4.5 域模型层 200](#_Toc519352047)

[4.5老师修改和查看作业情况 210](#_Toc519352048)

[4.5.1 表现层 210](#_Toc519352049)

[4.5.2 控制层 213](#_Toc519352050)

[4.5.3 业务逻辑层 213](#_Toc519352051)

[4.5.4 数据持久层 221](#_Toc519352052)

[4.5.5 域模型层 221](#_Toc519352053)

[5、管理端模块详细设计 223](#_Toc519352054)

[5.1、后台新增数据模块详细设计 223](#_Toc519352055)

[5.1.1 表现层 223](#_Toc519352056)

[5.1.2 控制层 224](#_Toc519352057)

[5.1.3 业务逻辑层 225](#_Toc519352058)

[5.1.4 数据持久层 236](#_Toc519352059)

[5.1.5 域模型层 236](#_Toc519352060)

[6、公共部分模块详细设计 243](#_Toc519352061)

[6.1 公共页面 243](#_Toc519352062)

[6.2 安全模块的详细设计 243](#_Toc519352063)

# 1．导言

## 1.1 目的

该文档是对课程作业管理系统概要设计进行的详细说明，是为了指导和规范课程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

该文档内容涵盖课程作业管理系统系统的所有功能模块的详细设计描述。

## 1.3 引用标准

[1] 《软件工程案例教程 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

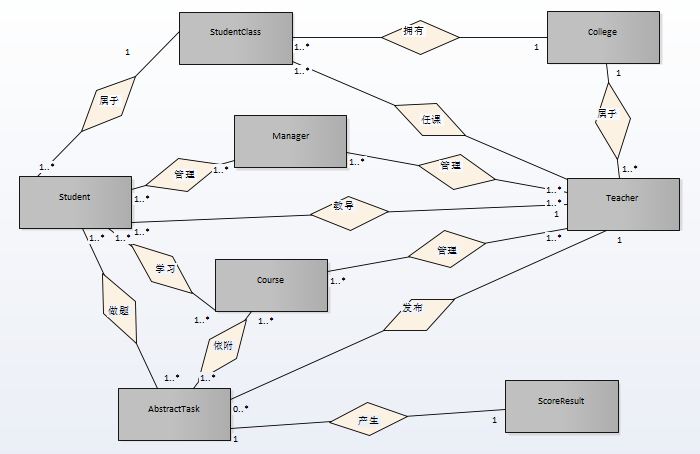
## 1.5 版本更新信息

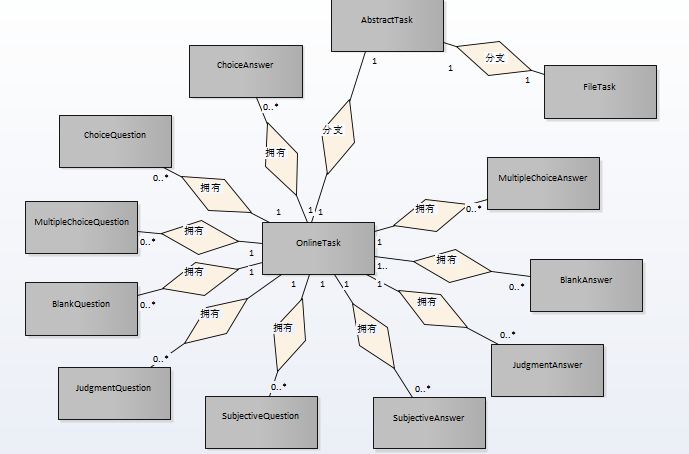
本文档的更新记录如下表1-1所示。

表1-1 文档更新记录

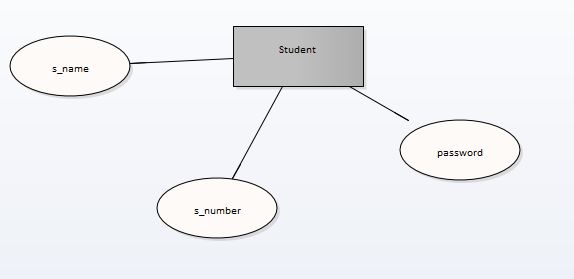
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改编号 | 修改日期 | 修改后版本 | 修改位置 | 修改内容概述 |
| 001 | 2018.6.26 | 1.0 | 全部 | 初始发布版本 |
| 002 | 2018.7.14 | 2.0 | 全部 | 全部改版 |

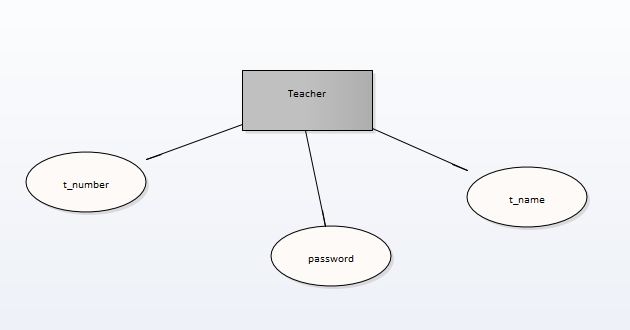
# 2．详细设计简述

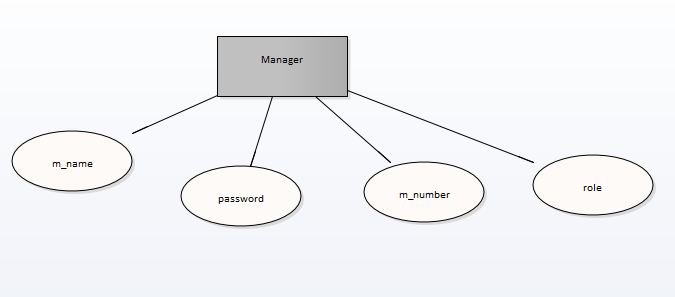
图2-1 总体实体关系图（一）

图2-2总体实体关系图（二）：

抽象作业和具体类型作业实体关系图

图2-3学生实体图

图2-4 教师实体图

图2-5 管理员实体图

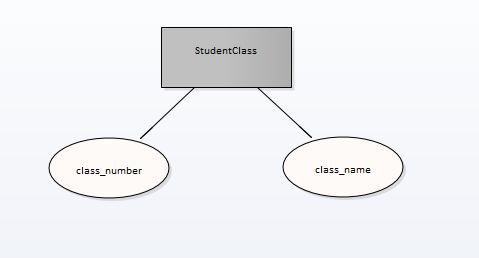


图2-6 班级实体图

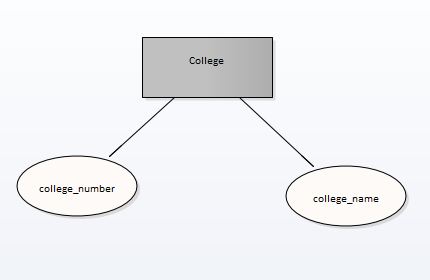
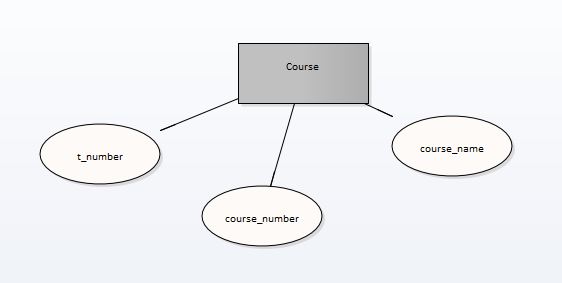
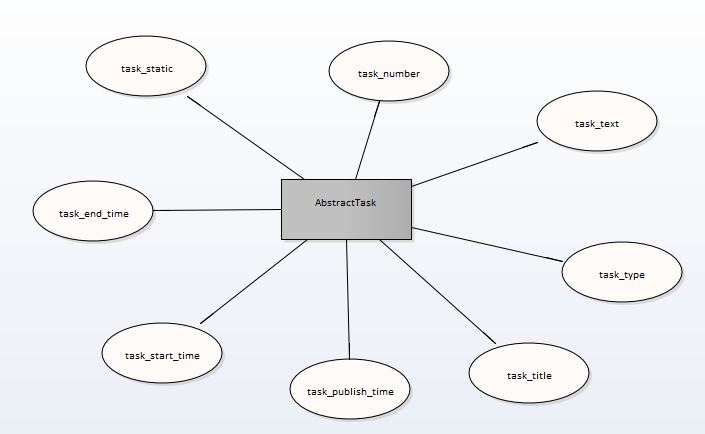
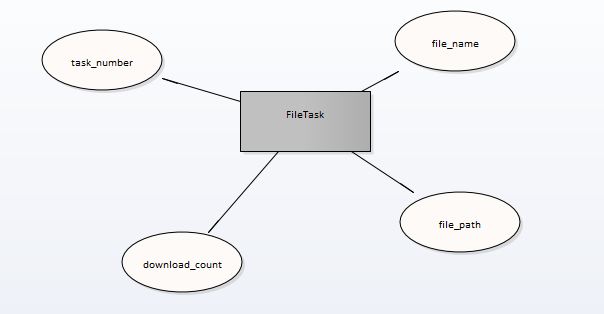


图2-7 学院实体图

图2-8 课程实体图

图2-9 抽象作业实体图

图2-10线下型作业（文件式作业）实体图

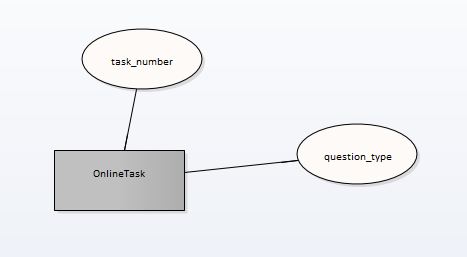
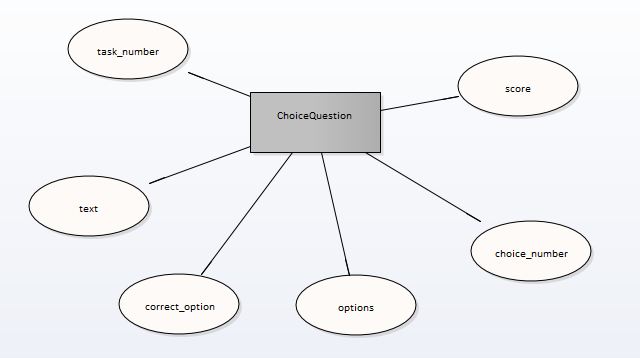
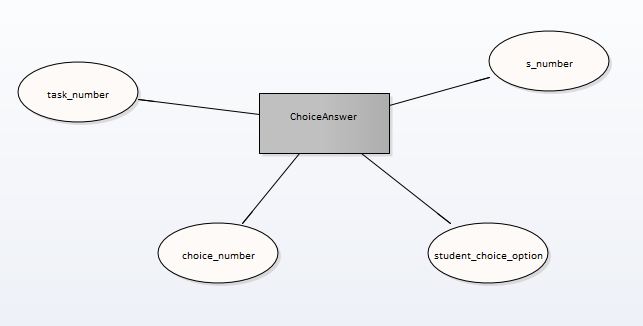
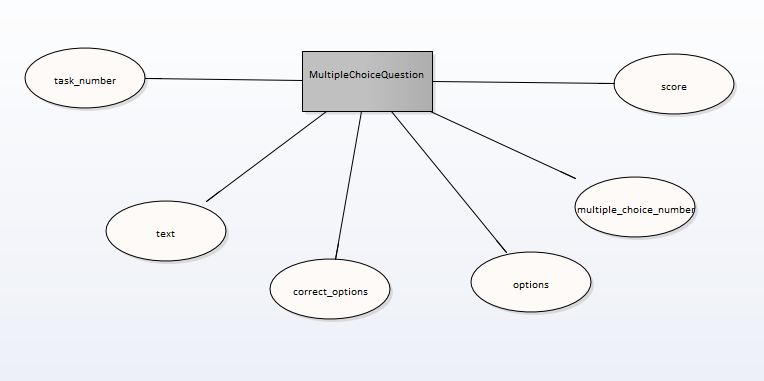
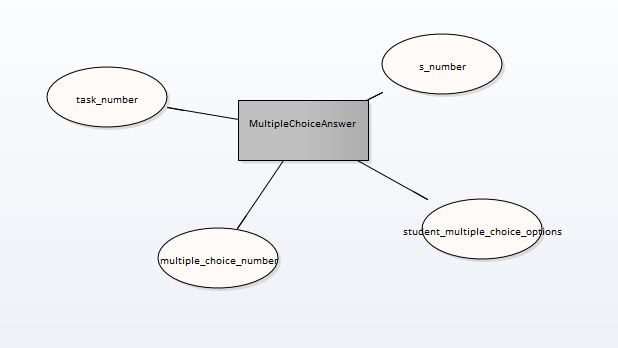


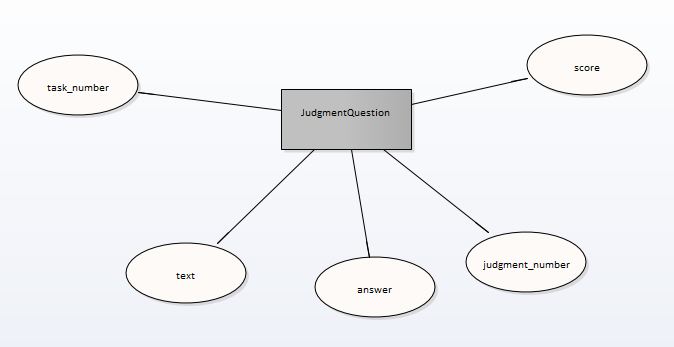
图2-11在线型作业实体图

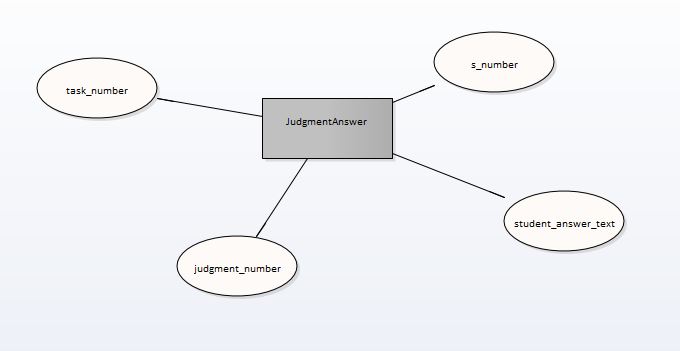
图2-12在线作业单选题题目实体图

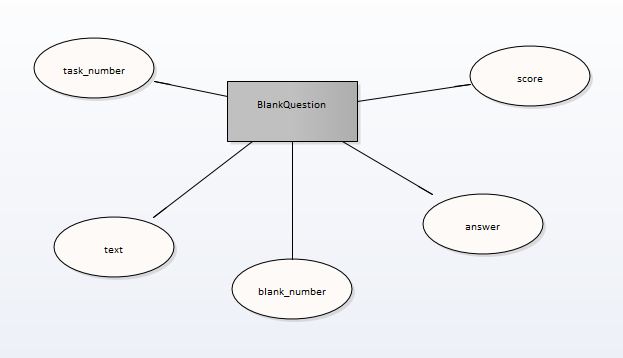
图2-13在线作业单选题答案实体图

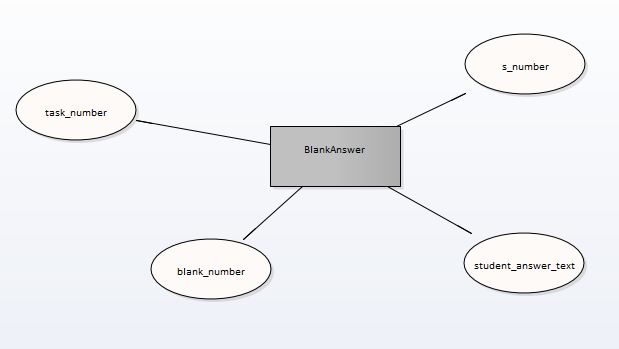
图2-14在线作业多选题题目实体图

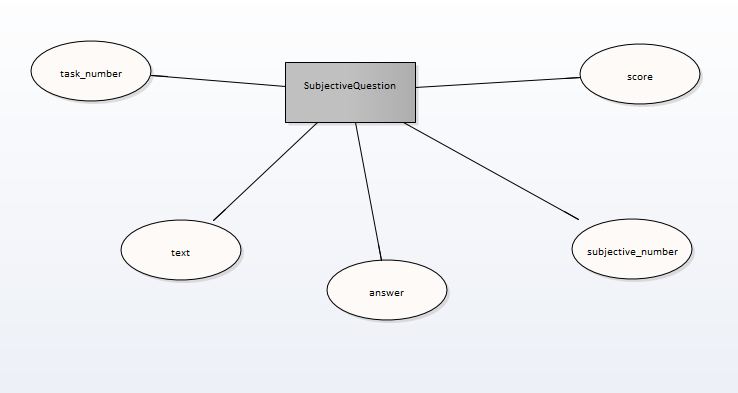
图2-15在线作业多选题答案实体图

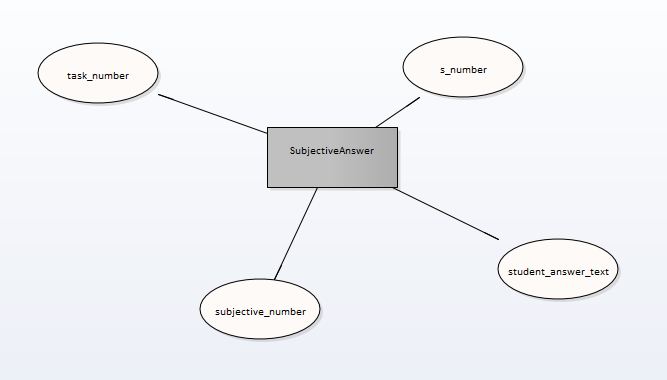
图2-16在线作业判断选题题目实体图

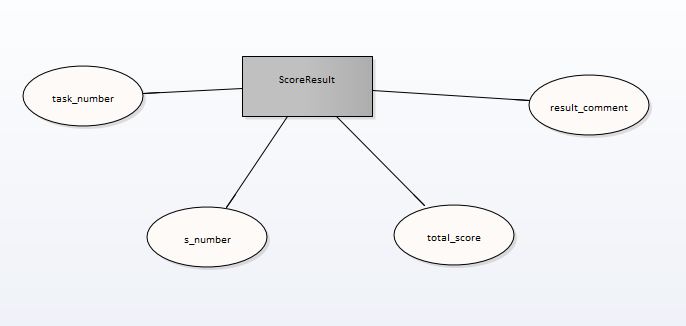
图2-17在线作业判断题题答案实体图

图2-18在线作业填空题题目实体图

图2-19在线作业填空题题答案实体图

图2-20在线作业客观题题目实体图

图2-21在线作业客观题答案实体图

图2-22作业评分和评语实体图

# 3、界面详细设计

## 3.1、页面设计说明

前端界面采用 bootstrap、jQuery、layui。

## 3.2、静态页面详细设计

课程作业管理系统静态页面见表3-1

表3-1 页面表现层设计

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 登录界面 | login.jsp | 进入系统前需先登录 |
| 教师主页 | index.jsp | 教师主页，可在此页选择布置作业、管理作业或查看作业 |
| 教师发布作业 | publish.jsp | 教师可在此发布在线型和文档型作业 |
| 教师管理作业 | manageHomework.jsp | 教师可在此删除、查看、编辑自己发过的作业，可根据班级或科目检索作业 |
| 教师查看作业 | checkHomework.jsp | 教师可在此查看发布的作业信息 |
| 教师查阅学生作业 | homeworkList.jsp | 教师可在此通过作业号或作业名以及提交状态来检索学生提交作业信息，以表格的形式显示 |
| 教师批改在线作业 | correct.jsp | 教师可在此批改在线型的作业，并输入评分及评语 |
| 教师批改文档型作业 | fileCorrect.jsp | 教师可在此批改文档型的作业，并输入评分及评语 |
| 教师查看某次作业具体情况前的列表 | gradeManage.jsp | 教师可在此通过选择班级、科目或作业号检索某次作业 |
| 教师查看分数统计信息 | homeworkDetails.jsp | 教师可在此查看合格人数及不合格人数的详细信息 |
| 学生首页 | studentMain.jsp | 学生可在此通过老师姓名或作业号搜索作业 |
| 学生完成线下作业 | stuDoHomework.jsp | 学生可在此下载作业附件，并提交学生线下作业的文件 |
| 作业列表 | stuHWList.jsp | 学生搜索出来的作业会在此以表格的形式展示 |
| 作业评分详情 | stuScore.jsp | 学生可在此查看某次作业的分数、班级排名和教师评语 |
| 学生在线做作业 | stuDoHomework.jsp | 学生可在此在线完成作业 |
| 后台管理页面 | manage.jsp | 管理员再次进行教师、学生、课程、学院、班级的增删查改 |

# 4、客户端模块详细设计

## 4.1、公用模块的详细设计

### 4.1.1 表现层

公用模块的表现层主要是一些页面出错时的跳转页面，显示当前错误的结果处理报告。具体见表4-1

表 4-1

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 登录出错页面 | loginError.jsp | 登入时出现严重错误时跳转到该页面。 |
|  |  |  |

### 4.1.2 控制层

##### 4.1.2.1新增数据模块的控制层

（1）继承的接口

|  |
| --- |
| package biz;  public interface AddFunctionBiz {  public String addStudent(Student student,Student\_class student\_class) throws Exception;  public String addTeacher(Teacher teacher,College college) throws Exception;  public String addCollege(College college) throws Exception;  public String addClass(Student\_class studentClass,College college) throws Exception;  public String addCourse(Course course) throws Exception;  public String addFileTask(HttpServletRequest request,Abstract\_task task,File\_task fileTask) throws Exception;  public String addOnlineTask(HttpServletRequest request,Abstract\_task task,Online\_task onlineTask) throws Exception;  public String submitOnlineTaskAnswer(HttpServletRequest request,Online\_task onlineTask,Student student) throws Exception;  public String submitDelineTaskAnswer(HttpServletRequest request,File\_task fileTask,Student student) throws Exception;  public String submitOnlineTaskFinalScore(Score\_result scoreResult);  public String connect\_course\_and\_student(Course course,Student student);  } |

（2）代码实现

|  |
| --- |
| package controller;  @Component//表示该类为组件类  @Controller//表示其为控制器  public class AddFunctionController implements AddFunctionBiz{  @Autowired  private HttpServletRequest request;  @Autowired  private HttpServletResponse response;    //@Resource  @Autowired  //@Qualifier("ADDFACTIONSERVICE\_ID")  private AddFunctionService addService;  @Autowired  private UploadFileService uploadFileService;      @Override  @RequestMapping(value="/add\_student",method=RequestMethod.POST)  @ResponseBody  public String addStudent(Student student,Student\_class student\_class) throws Exception {  return addService.addStudent(student, student\_class);  }  @Override  @RequestMapping(value="/add\_teacher",method=RequestMethod.POST)  @ResponseBody  public String addTeacher(Teacher teacher,College college) throws Exception{  return addService.addTeacher(teacher, college);  }  @Override  @RequestMapping(value="/add\_college",method=RequestMethod.POST)  @ResponseBody  public String addCollege(College college) throws Exception{  return addService.addCollege(college);  }  @Override  @RequestMapping(value="/add\_class",method=RequestMethod.POST)  @ResponseBody  public String addClass(Student\_class studentClass,College college) throws Exception{  return addService.addClass(studentClass, college);  }  @Override  @RequestMapping(value="/add\_course",method=RequestMethod.POST)  @ResponseBody    public String addCourse(Course course) throws Exception{  return addService.addCourse(course);  }    @Override  @RequestMapping(value="/publish\_file\_task",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String addFileTask(HttpServletRequest request, Abstract\_task task,File\_task fileTask) throws Exception {    Map<String,String> fileMessage=new HashMap<String,String>();  fileMessage=uploadFileService.uploadMultiFile(request);    fileTask.setDownload\_count(0);  fileTask.setFile\_name(fileMessage.get("fileName"));  fileTask.setFile\_path(fileMessage.get("relativePath"));  String result=addService.addFileTask(request, task,fileTask);    return result;  }  @Override  @RequestMapping(value="/publish\_online\_task",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String addOnlineTask(HttpServletRequest request, Abstract\_task task, Online\_task onlineTask)  throws Exception {    System.out.println("task\_number:"+task.getTask\_number());  String result=addService.addOnlineTask(request, task, onlineTask);  System.out.println(result);  return result;  }  @Override  @RequestMapping(value="/submit\_student\_online\_task\_answer",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String submitOnlineTaskAnswer(HttpServletRequest request, Online\_task onlineTask, Student student)  throws Exception {  return addService.submitOnlineTaskAnswer(request, onlineTask, student);  }  @Override  @RequestMapping(value="/submit\_student\_deline\_task\_answer",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String submitDelineTaskAnswer(HttpServletRequest request, File\_task fileTask, Student student)  throws Exception {  // TODO Auto-generated method stub  return null;  }  @Override  @RequestMapping(value="/submit\_online\_task\_score",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String submitOnlineTaskFinalScore(Score\_result scoreResult) {  return addService.submitOnlineTaskFinalScore(scoreResult);  }  @Override  @RequestMapping(value="/connect\_course\_and\_student",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String connect\_course\_and\_student(Course course, Student student) {  return addService.connect\_course\_and\_student(course, student);  }  } |

##### 4.1.2.2查询数据控制层

（1）继承的接口

|  |
| --- |
| package biz;  public interface SearchFunctionBiz {  public String getAllCollege();  public String getTeachersByCollege(College college);  public String getClassesByCollege(College college);  public String existTask(Abstract\_task task);  public String getAbstractTasksByTask(Abstract\_task task,String sortOrder,Set<String> sortParams);  public String getDelineTasksByTask\_number(String taskNumber);  public String getOnlineTasksByTask\_number(String taskNumber);  public String getOneStudentCompletion\_DelineTask\_ByTask\_numberAndS\_number(String task\_number,String s\_number);  /\*包括学生的上传的文件路径、文件名、上传时间\*/  public String getOneStudentCompletion\_OnlineTask\_ByTask\_numberAndS\_number(String task\_number,String s\_number);  /\*包括学生的每一道题目的信息，以及客观题的对错情况\*/  public String getStudentsByClass(Student\_class studentClass);  /\*根据班级的班级编号或班级名称获取本班学生\*/  public String getStudentsByCourse(Course course);  /\*根据课程的编号、课程名获取本课程的学生\*/  public String getStydentsByTask\_number(String taskNumber);//单个学生的信息需要包含：学号、姓名、是否完成该作业的状态，完成则"static":"true"  //否则"static":"false"  public String getOtherMessageByTask\_number(String taskNumber);  /\*根据作业好获取改作业的所属课程的编号、名字、任课老师的编号、老师的名字、课程所属的班级编号、班级名字\*/  public String getStudentScoreAndCommentByS\_number(String studentNumber);  /\*根据学生学号获取作业完成后，教师的给的分数和评语\*/  public String getStudentByS\_number(String s\_number);  public String getCourseByCourse\_number(String course\_number);    } |

（2）代码实现

|  |
| --- |
| package controller;  @Component//表示该类为组件类  @Controller//表示其为控制器  public class SearchFunctionController implements SearchFunctionBiz{  @Autowired  private HttpServletRequest request;  @Autowired  private HttpServletResponse response;  @Autowired  private SearchFunctionService searchService;    @Override  @RequestMapping(value="/get\_colleges",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getAllCollege() {  //System.out.println();  return searchService.getAllCollege();  }  @Override  @RequestMapping(value="/get\_teachers",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getTeachersByCollege(College college) {  return searchService.getTeachersByCollege(college);  }  @Override  @RequestMapping(value="/get\_classes",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getClassesByCollege(College college) {  return searchService.getClassesByCollege(college);  }  @Override  @RequestMapping(value="/judge\_exist\_task",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String existTask(Abstract\_task task) {  return searchService.existTask(task);  }  @Override    public String getAbstractTasksByTask(Abstract\_task task, String sortOrder, Set<String> sortParams) {  System.out.println("task\_number: "+task.getTask\_number());  //return searchService.getAbstractTasksByTask(task, sortOrder, sortParams);  return null;  }  @RequestMapping(value="/get\_abstract\_task",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getOneAbstractTasksByTask\_number(String task\_number){  Abstract\_task task = new Abstract\_task();  task.setTask\_number(task\_number);  Set<String> sortParams = new HashSet<>();  sortParams.add("task\_end\_time");  SearchFunctionService service=new SearchFunctionService();  return service.getAbstractTasksByTask(task,null,null);  }  @Override  @RequestMapping(value="/get\_deline\_task\_by\_taskNumber",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getDelineTasksByTask\_number(String taskNumber) {  // TODO Auto-generated method stub  return null;  }  @Override  @RequestMapping(value="/get\_online\_task\_by\_taskNumber",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getOnlineTasksByTask\_number(String taskNumber) {  return searchService.getOnlineTasksByTask\_number(taskNumber);  }  @Override  @RequestMapping(value="/get\_s\_completion\_deline\_task\_by\_taskNumber",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getOneStudentCompletion\_DelineTask\_ByTask\_numberAndS\_number(String task\_number, String s\_number) {  // TODO Auto-generated method stub  return null;  }  @Override  @RequestMapping(value="/get\_s\_completion\_online\_task\_by\_taskNumber",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getOneStudentCompletion\_OnlineTask\_ByTask\_numberAndS\_number(String task\_number, String s\_number) {  return searchService.getOneStudentCompletion\_OnlineTask\_ByTask\_numberAndS\_number(task\_number, s\_number);  }  @Override  @RequestMapping(value="/get\_students\_by\_class",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getStudentsByClass(Student\_class studentClass) {  // TODO Auto-generated method stub  return null;  }  @Override  @RequestMapping(value="/get\_students\_by\_course",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getStudentsByCourse(Course course) {  // TODO Auto-generated method stub  return null;  }  @Override  @RequestMapping(value="/get\_students\_by\_taskNumber",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getStydentsByTask\_number(String taskNumber) {  // TODO Auto-generated method stub  return null;  }  @Override  @RequestMapping(value="/get\_otherMessage\_by\_taskNumber",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getOtherMessageByTask\_number(String taskNumber) {  return searchService.getOtherMessageByTask\_number(taskNumber);  }  @Override  @RequestMapping(value="/get\_student\_result\_by\_task\_s\_number",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getStudentScoreAndCommentByS\_number(String studentNumber) {  // TODO Auto-generated method stub  return null;  }  @Override  @RequestMapping(value="/search\_student\_by\_studentNumber",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getStudentByS\_number(String s\_number) {  System.out.println("当前学号："+s\_number);  return searchService.getStudentByS\_number(s\_number);  }  @Override  @RequestMapping(value="/search\_course\_by\_courseNumber",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String getCourseByCourse\_number(String course\_number) {  System.out.println("当前课程号："+course\_number);  return searchService.getCourseByCourse\_number(course\_number);  }  } |

##### 4.1.2.3更新控制模块的控制层

（1）继承的接口

|  |
| --- |
| package biz;  public interface UpdateFunctionBiz {  public String updateStudent(Student student);  public String updateTeacher(Teacher teacher);  public String updateCourse(Course course);  public String updateCollege(College college);  public String updateClass(Student\_class studentClass);  public String updateAbstractTask(Abstract\_task task);  public String updateDelineTask(File\_task task);  public String updateOnlineTask(Online\_task task);  public String updateScoreResult(Score\_result scoreResult);    } |

（2）代码实现

|  |
| --- |
| package controller;  @Component//表示该类为组件类  @Controller//表示其为控制器  public class UpdateFunctionController implements UpdateFunctionBiz {  @Autowired  private UpdateFunctionService updateService;    @Override  @RequestMapping(value="/update\_student",method=RequestMethod.POST)  @ResponseBody  public String updateStudent(Student student) {  return updateService.updateStudent(student);  }  @Override  @RequestMapping(value="/update\_teacher",method=RequestMethod.POST)  @ResponseBody  public String updateTeacher(Teacher teacher) {  return updateService.updateTeacher(teacher);  }  @Override  @RequestMapping(value="/update\_course",method=RequestMethod.POST)  @ResponseBody  public String updateCourse(Course course) {  return updateService.updateCourse(course);  }  @Override  @RequestMapping(value="/update\_college",method=RequestMethod.POST)  @ResponseBody  public String updateCollege(College college) {  return updateService.updateCollege(college);  }  @Override  @RequestMapping(value="/update\_class",method=RequestMethod.POST)  @ResponseBody  public String updateClass(Student\_class studentClass) {  return updateService.updateClass(studentClass);  }  @Override  @RequestMapping(value="/update\_abstract\_task",method=RequestMethod.POST)  @ResponseBody  public String updateAbstractTask(Abstract\_task task) {  return updateService.updateAbstractTask(task);  }  @Override  @RequestMapping(value="/update\_deline\_task",method=RequestMethod.POST)  @ResponseBody  public String updateDelineTask(File\_task task) {  return updateService.updateDelineTask(task);  }  @Override  @RequestMapping(value="/update\_online\_task",method=RequestMethod.POST)  @ResponseBody  public String updateOnlineTask(Online\_task task) {  return updateService.updateOnlineTask(task);  }  @Override  @RequestMapping(value="/update\_score\_result",method=RequestMethod.POST)  @ResponseBody  public String updateScoreResult(Score\_result scoreResult) {  return updateService.updateScoreResult(scoreResult);  }  } |

##### 4.1.2.4删除模块的控制层

（1）继承的接口

|  |
| --- |
| package biz;  public interface DeleteFunctionBiz {  public String deleteStudent(Student student);  public String deleteTeacher(Teacher teacher);  public String deleteCourse(Course course);  public String deleteCollege(College college);  public String deleteClass(Student\_class studentClass);  public String deleteAbstractTask(Abstract\_task task);  public String deleteDelineTask(File\_task task);  public String deleteOnlineTask(Online\_task task);  } |

（2）代码实现

|  |
| --- |
| package controller;  @Component//表示该类为组件类  @Controller//表示其为控制器  public class DeleteFunctionController implements DeleteFunctionBiz {  @Autowired  private DeleteFunctionService deleteService;    @Override  @RequestMapping(value="/delete\_student",method=RequestMethod.POST)  @ResponseBody  public String deleteStudent(Student student) {  return deleteService.deleteStudent(student);  }  @Override  @RequestMapping(value="/delete\_teacher",method=RequestMethod.POST)  @ResponseBody  public String deleteTeacher(Teacher teacher) {  return deleteService.deleteTeacher(teacher);  }  @Override  @RequestMapping(value="/delete\_course",method=RequestMethod.POST)  @ResponseBody  public String deleteCourse(Course course) {  return deleteService.deleteCourse(course);  }  @Override  @RequestMapping(value="/delete\_college",method=RequestMethod.POST)  @ResponseBody  public String deleteCollege(College college) {  return deleteService.deleteCollege(college);  }  @Override  @RequestMapping(value="/delete\_class",method=RequestMethod.POST)  @ResponseBody  public String deleteClass(Student\_class studentClass) {  return deleteService.deleteClass(studentClass);  }  @Override  @RequestMapping(value="/delete\_abstract\_task",method=RequestMethod.POST)  @ResponseBody  public String deleteAbstractTask(Abstract\_task task) {  return deleteService.deleteAbstractTask(task);  }  @Override  @RequestMapping(value="/delete\_deline\_task",method=RequestMethod.POST)  @ResponseBody  public String deleteDelineTask(File\_task task) {  return deleteService.deleteDelineTask(task);  }  @Override  @RequestMapping(value="/delete\_online\_task",method=RequestMethod.POST)  @ResponseBody  public String deleteOnlineTask(Online\_task task) {  return deleteService.deleteOnlineTask(task);  }  } |

##### 4.1.2.5上传文件模块的控制层

（1）继承的接口

|  |
| --- |
| package biz;  public interface UploadFileBiz {  public Map<String,String> uploadMultiFile(HttpServletRequest request);  } |

（2）代码实现

|  |
| --- |
| package controller;  @Component  @Controller  public class UploadFileController implements UploadFileBiz {  @Autowired  private UploadFileService uploadFileService;  @Override  @RequestMapping(value="/upload\_file",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  //@ResponseBody  public Map<String,String> uploadMultiFile(HttpServletRequest request) {  uploadFileService.uploadMultiFile(request);  return null;  }  } |

##### 4.1.2.6登录模块的控制层

（1）继承的接口

|  |
| --- |
| package biz;  public interface LoginBiz {  public String login(User user);  } |

（2）代码实现

|  |
| --- |
| package controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Component;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import bean.User;  import biz.LoginBiz;  import service.logicservice.LoginService;  import service.logicservice.SearchFunctionService;  //@Component  @Controller  public class LoginController implements LoginBiz {  @Autowired  private LoginService loginService;  @Override  @RequestMapping(value="/login",method=RequestMethod.POST)  public String login(User user) {  String result=loginService.login(user);  if(!result.equals("login")){  if(result.equals("teacher")){  result="index";  }  else if(result.equals("student")){  result="studentMain";  }  else if(result.equals("notDefound")){  result="loginError";  }  }      return result;  }  } |

### 4.1.3 数据持久层

##### 4.1.3.1新增数据模块的持久层

（1）继承的接口

|  |
| --- |
| package biz;  public interface AddFunctionBiz {  public String addStudent(Student student,Student\_class student\_class) throws Exception;  public String addTeacher(Teacher teacher,College college) throws Exception;  public String addCollege(College college) throws Exception;  public String addClass(Student\_class studentClass,College college) throws Exception;  public String addCourse(Course course) throws Exception;  public String addFileTask(HttpServletRequest request,Abstract\_task task,File\_task fileTask) throws Exception;  public String addOnlineTask(HttpServletRequest request,Abstract\_task task,Online\_task onlineTask) throws Exception;  public String submitOnlineTaskAnswer(HttpServletRequest request,Online\_task onlineTask,Student student) throws Exception;  public String submitDelineTaskAnswer(HttpServletRequest request,File\_task fileTask,Student student) throws Exception;  public String submitOnlineTaskFinalScore(Score\_result scoreResult);  public String connect\_course\_and\_student(Course course,Student student);  } |

（2）代码实现

|  |
| --- |
| package service.logicservice;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.text.SimpleDateFormat;  import java.util.ArrayList;  import java.util.Date;  import java.util.HashMap;  import java.util.Map;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.context.annotation.EnableAspectJAutoProxy;  import org.springframework.stereotype.Component;  import org.springframework.stereotype.Service;  import org.springframework.transaction.annotation.Transactional;  import org.springframework.transaction.interceptor.TransactionAspectSupport;  import bean.Abstract\_task;  import bean.Blank\_answer;  import bean.Blank\_question;  import bean.Choice\_answer;  import bean.Choice\_question;  import bean.College;  import bean.ConstantString;  import bean.Course;  import bean.File\_task;  import bean.Judgment\_answer;  import bean.Judgment\_question;  import bean.Multiple\_choice\_answer;  import bean.Multiple\_choice\_question;  import bean.Online\_task;  import bean.Score\_result;  import bean.Student;  import bean.Student\_class;  import bean.Subjective\_answer;  import bean.Subjective\_question;  import bean.Teacher;  import biz.AddFunctionBiz;  import dao.JsonUtilDao;  import net.sf.json.JSONArray;  import net.sf.json.JSONObject;  import net.sf.json.util.JSONStringer;  import service.mysqlservice.MysqlUtil;  @EnableAspectJAutoProxy(proxyTargetClass=true)//避免把proxy对象（即声明为Service的我类）当做bean来生成，不然会报错  //@Service("ADDFACTIONSERVICE\_ID")  @Service  public class AddFunctionService implements AddFunctionBiz {  @Autowired  private HttpServletResponse response;  /\*@Autowired  private MysqlUtil util;//自定义封装的jdbc操作工具类  \*/  /\*本项目的事务回滚并没有使用框架的配置回滚，也就是说并没有使用@Transactional和@Service，  \* 因为采用的jdbc的操作是自定义封装的，所以手动抛出异常后数据库事务并没有进行回滚（已经有事务被拦截了）  \* 如果以后换成使用Mytabits，项目的结构层次像该类的addStudent()配置即可  \* 本项目的jdbc回滚均为手动添加\*/  @Transactional(rollbackFor=Exception.class)//将事务响应回滚的异常的类型声明为Exception，因为spring默认为RuntimeException  //@Transactional  @Override  public String addStudent(Student student, Student\_class student\_class) throws Exception {    System.out.println(student.getS\_number()+" "+student.getS\_name()+" "+student.getPassword());  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="student";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("s\_number", student.getS\_number());  insertParams.put("s\_name", student.getS\_name());  insertParams.put("password", student.getPassword());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  tableName="class\_relationship\_student";  insertParams.clear();  insertParams.put("s\_number", student.getS\_number());  insertParams.put("class\_number", student\_class.getClass\_number());  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();  //依旧返回数据，也抛出异常  stringer.object().  key("result").value("false").  endObject();  /\*Exception e=new Exception();  throw e;\*/  util.getConn().rollback();    }  }  else{  //TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();  //依旧返回数据，也抛出异常  stringer.object().  key("result").value("false").  endObject();    }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  System.out.println(result);  return result;    }  @Override  @Transactional(rollbackFor=Exception.class)  public String addTeacher(Teacher teacher, College college) throws Exception {  System.out.println(teacher.getT\_number()+" "+teacher.getT\_name()+" "+teacher.getPassword());  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="teacher";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("t\_number", teacher.getT\_number());  insertParams.put("t\_name", teacher.getT\_name());  insertParams.put("password", teacher.getPassword());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  insertParams.clear();  insertParams.put("t\_number", teacher.getT\_number());  insertParams.put("college\_number", college.getCollege\_number());  tableName="college\_relationship\_teacher";  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  response.setContentType("text/json; charset=UTF-8");  System.out.println(result);  return result;  }  @Override  @Transactional(rollbackFor=Exception.class)  public String addCollege(College college) throws Exception{  System.out.println(college.getCollege\_name()+" "+college.getCollege\_number());  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="college";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("college\_number", college.getCollege\_number());  insertParams.put("college\_name", college.getCollege\_name());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  response.setContentType("text/json; charset=UTF-8");  System.out.println(result);  return result;  }  @Override  @Transactional(rollbackFor=Exception.class)  public String addClass(Student\_class studentClass, College college) throws Exception{  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="student\_class";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("class\_number", studentClass.getClass\_number());  insertParams.put("class\_name", studentClass.getClass\_name());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  tableName="college\_relationship\_class";  insertParams.clear();  insertParams.put("college\_number", college.getCollege\_number());  insertParams.put("class\_number", studentClass.getClass\_number());  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  response.setContentType("text/json; charset=UTF-8");  System.out.println(result);  return result;  }  @Override  @Transactional(rollbackFor=Exception.class)  public String addCourse(Course course) throws Exception{  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="course";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("course\_number", course.getCourse\_number());  insertParams.put("course\_name", course.getCourse\_name());  insertParams.put("t\_number", course.getT\_number());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  response.setContentType("text/json; charset=UTF-8");  System.out.println(result);  return result;  }  @Override  @Transactional(rollbackFor=Exception.class)  public String addFileTask(HttpServletRequest request, Abstract\_task task,File\_task fileTask) throws Exception {    Date date= new Date();//创建一个时间对象，获取到当前的时间  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");//设置时间显示格式  String str = sdf.format(date);//将当前时间格式化为需要的类型    System.out.println("t\_number:"+task.getT\_number());  System.out.println("task\_number:"+task.getTask\_number());  System.out.println("task\_title:"+task.getTask\_title());  System.out.println("task\_end\_timer:"+task.getTask\_end\_time());  System.out.println("task\_type:"+task.getTask\_type());  System.out.println("task\_text:"+task.getTask\_text());  System.out.println("course\_number:"+task.getCourse\_number());  System.out.println("class\_number:"+task.getClass\_number());    System.out.println("file\_name:"+fileTask.getFile\_name());  System.out.println("file\_path:"+fileTask.getFile\_path());    JSONStringer stringer = new JSONStringer();//  stringer.array();  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="abstract\_task";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("t\_number", task.getT\_number());  insertParams.put("course\_number", task.getClass\_number());  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("task\_text", task.getTask\_text());  insertParams.put("task\_title", task.getTask\_title());  insertParams.put("task\_type", task.getTask\_type());  insertParams.put("class\_number", task.getClass\_number());  insertParams.put("task\_end\_time", task.getTask\_end\_time());    insertParams.put("task\_publish\_time", str);  insertParams.put("task\_start\_time", str);  /\*调用自定义软件接口新增数据E\*/  if(util.insert(tableName, insertParams)==1){  tableName="file\_task";  insertParams.clear();  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("file\_name", fileTask.getFile\_name());  insertParams.put("file\_path", fileTask.getFile\_path());  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  util.getConn().rollback();  }  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  String result=stringer.toString();  return result;  }  @Override  public String addOnlineTask(HttpServletRequest request,Abstract\_task task, Online\_task onlineTask) throws Exception {  boolean allRight=true;  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="abstract\_task";  Map<String,Object> insertParams=new HashMap<String,Object>();  /\*添加进抽象在线作业表，该表可以存在线作业和文件作业\*/  Date date= new Date();//创建一个时间对象，获取到当前的时间  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");//设置时间显示格式  String str = sdf.format(date);//将当前时间格式化为需要的类型  insertParams.put("t\_number", task.getT\_number());  insertParams.put("course\_number", task.getClass\_number());  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("task\_text", task.getTask\_text());  insertParams.put("task\_title", task.getTask\_title());  insertParams.put("task\_type", task.getTask\_type());  insertParams.put("class\_number", task.getClass\_number());  insertParams.put("task\_end\_time", task.getTask\_end\_time());  insertParams.put("task\_publish\_time", str);  insertParams.put("task\_start\_time", str);  if(util.insert(tableName, insertParams)!=1){  allRight=false;  }      /\*依次添加进不同类型的题目表\*/  insertParams.clear();  String simpleJsonArray=request.getParameter("simpleJsonArray");  String multipleJsonArray=request.getParameter("multipleJsonArray");  String gapfillingJsonArray=request.getParameter("gapfillingJsonArray");  String judgeJsonArray=request.getParameter("judgeJsonArray");  String shortJsonArray=request.getParameter("shortJsonArray");  System.out.println("单选题："+simpleJsonArray);  System.out.println("多选题："+multipleJsonArray);  System.out.println("填空题："+gapfillingJsonArray);  System.out.println("判断题："+judgeJsonArray);  System.out.println("客观题："+shortJsonArray);  JsonUtilDao jsonUtil=new JsonUtilDao();  if(simpleJsonArray!=null&&!simpleJsonArray.equals("")&&allRight){  Choice\_question pojo=new Choice\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(simpleJsonArray, pojo);  tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "choice\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="choice\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Choice\_question) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("correct\_option", pojo.getCorrect\_option());  insertParams.put("choice\_options", pojo.getChoice\_options());  insertParams.put("choice\_number", pojo.getChoice\_number());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  if(multipleJsonArray!=null&&!multipleJsonArray.equals("")&&allRight){  insertParams.clear();  Multiple\_choice\_question pojo=new Multiple\_choice\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(multipleJsonArray, pojo);    tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "multiple\_choice\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="multiple\_choice\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Multiple\_choice\_question) arrayList.get(i);  System.out.println("task\_number:"+task.getTask\_number());  System.out.println("task\_number:"+pojo.getTask\_number());  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("correct\_options", pojo.getCorrect\_options());  insertParams.put("options", pojo.getOptions());  insertParams.put("multiple\_choice\_number", pojo.getMultiple\_choice\_number());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  if(gapfillingJsonArray!=null&&!gapfillingJsonArray.equals("")&&allRight){  insertParams.clear();  Blank\_question pojo=new Blank\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(gapfillingJsonArray, pojo);    tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "blank\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="blank\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Blank\_question) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("blank\_number", pojo.getBlank\_number());  insertParams.put("answer", pojo.getAnswer());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  if(judgeJsonArray!=null&&!judgeJsonArray.equals("")&&allRight){  insertParams.clear();  Judgment\_question pojo=new Judgment\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(judgeJsonArray, pojo);    tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "judgment\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="judgment\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Judgment\_question) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("judgment\_number", pojo.getJudgment\_number());  insertParams.put("answer", pojo.getAnswer());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  if(shortJsonArray!=null&&!shortJsonArray.equals("")&&allRight){  insertParams.clear();  Subjective\_question pojo=new Subjective\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(shortJsonArray, pojo);  pojo=(Subjective\_question) arrayList.get(0);    tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "subjective\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="subjective\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Subjective\_question) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("subjective\_number", pojo.getSubjective\_number());  insertParams.put("answer", pojo.getAnswer());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  String result;  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(allRight){  util.getConn().commit();  stringer.object().  key("result").value("true").  endObject();  }  else{  util.getConn().rollback();  stringer.object().  key("result").value("false").  endObject();  }  stringer.endArray();  result=stringer.toString();  return result;//为[{"result":"true"}]前端解析不需要newd=JSON.parse(data)  //如果是[{result:"true"}]前端解析需要newd=JSON.parse(data)  }  @Override  public String submitOnlineTaskAnswer(HttpServletRequest request, Online\_task onlineTask, Student student)  throws Exception {  boolean allRight=true;  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="";  Map<String,Object> insertParams=new HashMap<String,Object>();      /\*依次添加进不同类型的题目表\*/  String simpleJsonArray=request.getParameter("simpleJsonArray");  String multipleJsonArray=request.getParameter("multipleJsonArray");  String gapfillingJsonArray=request.getParameter("gapfillingJsonArray");  String judgeJsonArray=request.getParameter("judgeJsonArray");  String shortJsonArray=request.getParameter("shortJsonArray");  System.out.println("单选题答题："+simpleJsonArray);  System.out.println("多选题答题："+multipleJsonArray);  System.out.println("填空题答题："+gapfillingJsonArray);  System.out.println("判断题答题："+judgeJsonArray);  System.out.println("客观题答题："+shortJsonArray);  JsonUtilDao jsonUtil=new JsonUtilDao();  if(simpleJsonArray!=null&&!simpleJsonArray.equals("")&&allRight){  Choice\_answer pojo=new Choice\_answer();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(simpleJsonArray, pojo);    tableName="choice\_answer";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Choice\_answer) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("s\_number", pojo.getS\_number());  insertParams.put("choice\_number", pojo.getChoice\_number());  insertParams.put("student\_choice\_option", pojo.getStudent\_choice\_option());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }    }  if(multipleJsonArray!=null&&!multipleJsonArray.equals("")&&allRight){  Multiple\_choice\_answer pojo=new Multiple\_choice\_answer();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(multipleJsonArray, pojo);    tableName="multiple\_choice\_answer";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Multiple\_choice\_answer) arrayList.get(i);  //System.out.println("task\_number:"+onlineTask.getTask\_number());  //System.out.println("task\_number:"+pojo.getTask\_number());  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("multiple\_choice\_number", pojo.getMultiple\_choice\_number());  insertParams.put("student\_multiple\_choice\_options", pojo.getStudent\_multiple\_choice\_options());  insertParams.put("s\_number", pojo.getS\_number());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  if(gapfillingJsonArray!=null&&!gapfillingJsonArray.equals("")&&allRight){  Blank\_answer pojo=new Blank\_answer();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(gapfillingJsonArray, pojo);    tableName="blank\_answer";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Blank\_answer) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("blank\_number", pojo.getBlank\_number());  insertParams.put("s\_number", pojo.getS\_number());  insertParams.put("student\_answer\_text", pojo.getStudent\_answer\_text());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }    }  if(judgeJsonArray!=null&&!judgeJsonArray.equals("")&&allRight){  Judgment\_answer pojo=new Judgment\_answer();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(judgeJsonArray, pojo);  tableName="judgment\_answer";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Judgment\_answer) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("judgment\_number", pojo.getJudgment\_number());  insertParams.put("s\_number", pojo.getS\_number());  insertParams.put("student\_answer\_text", pojo.getStudent\_answer\_text());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }    }  if(shortJsonArray!=null&&!shortJsonArray.equals("")&&allRight){  Subjective\_answer pojo=new Subjective\_answer();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(shortJsonArray, pojo);    tableName="subjective\_answer";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Subjective\_answer) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("subjective\_number", pojo.getSubjective\_number());  insertParams.put("s\_number", pojo.getS\_number());  insertParams.put("student\_answer\_text", pojo.getStudent\_answer\_text());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }    }  String result;  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(allRight){  util.getConn().commit();  stringer.object().  key("result").value("true").  endObject();  }  else{  util.getConn().rollback();  stringer.object().  key("result").value("false").  endObject();  }  stringer.endArray();  result=stringer.toString();  return result;//为[{"result":"true"}]前端解析不需要newd=JSON.parse(data)  //如果是[{result:"true"}]前端解析需要newd=JSON.parse(data)  }  @Override  public String submitDelineTaskAnswer(HttpServletRequest request, File\_task fileTask, Student student)  throws Exception {  // TODO Auto-generated method stub  return null;  }  @Override  public String submitOnlineTaskFinalScore(Score\_result scoreResult) {  JSONStringer stringer = new JSONStringer();//  stringer.array();  MysqlUtil util=new MysqlUtil();  String tableName="";  String result="";  Map<String,Object> insertParams=new HashMap<String,Object>();  try {  util.getConn().setAutoCommit(false);    int totalObjectiveScore=0;//记录所有客观题的分数和  Map<String,Object> conditionalParams=new HashMap<String,Object>();  ResultSet rs1=null;//rs1记录该在线作业的题型的表名  ResultSet rs2=null;//rs2记录某个题型的所有题目的答题  ResultSet rs3=null;//rs3记录某个题型的所有题目  if(scoreResult.getTask\_number()!=null&&!scoreResult.getTask\_number().equals("")){  if(scoreResult.getS\_number()!=null&&!scoreResult.getS\_number().equals("")){  if(scoreResult.getTotal\_score()!=null&&!scoreResult.getTotal\_score().equals("")){  /\*------------------------获取客观题总分S------------------\*/  System.out.println("主观题得分："+scoreResult.getTotal\_score());  tableName="online\_task";  conditionalParams.put("task\_number", scoreResult.getTask\_number());  rs1=util.select(tableName, null, conditionalParams);  while(rs1.next()){    /\*------------------找到正确答案，用于匹配客观题S---------------------\*/  tableName=rs1.getString("question\_type");  System.out.println("当前查找的题型："+tableName);  System.out.println("当前查询条件："+conditionalParams);  conditionalParams.clear();  conditionalParams.put("task\_number", scoreResult.getTask\_number());  rs3=util.select(tableName, null, conditionalParams);  /\*------------------找到正确答案，用于匹配客观题E---------------------\*/  /\*------------------找到学生客观题答题情况S---------------------\*/  int length=rs1.getString("question\_type").split("\_").length;  if(length==2)  tableName=rs1.getString("question\_type").split("\_")[0]+"\_answer";  if(length>2){  int originalLength=rs1.getString("question\_type").length();  int lastStringLength=rs1.getString("question\_type").split("\_")[length-1].length();  int needCutLength=lastStringLength;  tableName=rs1.getString("question\_type").substring(0,originalLength-needCutLength);  tableName=tableName+"answer";  }  conditionalParams.put("s\_number", scoreResult.getS\_number());  System.out.println("当前查找的题型："+tableName);  rs2=util.select(tableName, null, conditionalParams);  while(rs2.next()&&rs3.next()){  switch(rs1.getString("question\_type")){  case "choice\_question":  if(rs2.getString("student\_choice\_option").equals(rs3.getString("correct\_option"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("单选题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;  case "multiple\_choice\_question":  if(rs2.getString("student\_multiple\_choice\_options").equals(rs3.getString("correct\_options"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("多选题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;  /\*case "blank\_question":  if(rs2.getString("student\_answer\_text").equals(rs3.getString("answer"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("填空题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;\*/  case "judgment\_question":  if(rs2.getString("student\_answer\_text").equals(rs3.getString("answer"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("判断题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;  /\*case "subjective\_question":  if(rs2.getString("student\_answer\_text").equals(rs3.getString("answer"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("简答题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;\*/  }    }  /\*------------------找到学生客观题答题情况E---------------------\*/  System.out.println("客观题总得分： "+totalObjectiveScore);  }  /\*------------------------获取客观题总分E------------------\*/  tableName="score\_result";  if(scoreResult.getResult\_comment()!=null&&!scoreResult.getResult\_comment().equals("")){  insertParams.put("task\_number", scoreResult.getTask\_number());  insertParams.put("s\_number", scoreResult.getS\_number());  int totalScore=Integer.parseInt(scoreResult.getTotal\_score())+totalObjectiveScore;  insertParams.put("total\_score", totalScore+"");  insertParams.put("result\_comment", scoreResult.getResult\_comment());  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  else{  insertParams.put("task\_number", scoreResult.getTask\_number());  insertParams.put("s\_number", scoreResult.getS\_number());  int totalScore=Integer.parseInt(scoreResult.getTotal\_score())+totalObjectiveScore;  insertParams.put("total\_score", totalScore+"");  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  }  }  }    } catch (SQLException e) {  e.printStackTrace();  }  stringer.endArray();  result=stringer.toString();  return result;//为[{"result":"true"}]前端解析不需要newd=JSON.parse(data)  //如果是[{result:"true"}]前端解析需要newd=JSON.parse(data)  }  @Override  public String connect\_course\_and\_student(Course course, Student student) {  String result="";  MysqlUtil util=new MysqlUtil();  JSONStringer stringer = new JSONStringer();//  Map<String,Object> insertParams=new HashMap<String,Object>();  stringer.array();  try {  util.getConn().setAutoCommit(false);  if(course.getCourse\_number()!=null&&course.getCourse\_number().equals("")){  if(student.getS\_number()!=null&&student.getS\_number().equals("")){  insertParams.put("course\_number", course.getCourse\_number());  insertParams.put("s\_number", student.getS\_number());  String tableName="course\_relationship\_student";  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  }  } catch (SQLException e) {  e.printStackTrace();  }  stringer.endArray();  result=stringer.toString();  return result;  }  } |

##### 4.1.3.2查询模块的持久层

（1）继承的接口

|  |
| --- |
| package biz;  public interface SearchFunctionBiz {  public String getAllCollege();  public String getTeachersByCollege(College college);  public String getClassesByCollege(College college);  public String existTask(Abstract\_task task);  public String getAbstractTasksByTask(Abstract\_task task,String sortOrder,Set<String> sortParams);  public String getDelineTasksByTask\_number(String taskNumber);  public String getOnlineTasksByTask\_number(String taskNumber);  public String getOneStudentCompletion\_DelineTask\_ByTask\_numberAndS\_number(String task\_number,String s\_number);  /\*包括学生的上传的文件路径、文件名、上传时间\*/  public String getOneStudentCompletion\_OnlineTask\_ByTask\_numberAndS\_number(String task\_number,String s\_number);  /\*包括学生的每一道题目的信息，以及客观题的对错情况\*/  public String getStudentsByClass(Student\_class studentClass);  /\*根据班级的班级编号或班级名称获取本班学生\*/  public String getStudentsByCourse(Course course);  /\*根据课程的编号、课程名获取本课程的学生\*/  public String getStydentsByTask\_number(String taskNumber);//单个学生的信息需要包含：学号、姓名、是否完成该作业的状态，完成则"static":"true"  //否则"static":"false"  public String getOtherMessageByTask\_number(String taskNumber);  /\*根据作业好获取改作业的所属课程的编号、名字、任课老师的编号、老师的名字、课程所属的班级编号、班级名字\*/  public String getStudentScoreAndCommentByS\_number(String studentNumber);  /\*根据学生学号获取作业完成后，教师的给的分数和评语\*/  public String getStudentByS\_number(String s\_number);  public String getCourseByCourse\_number(String course\_number);    } |

（2）代码实现

|  |
| --- |
| package service.logicservice;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.HashSet;  import java.util.List;  import java.util.Map;  import java.util.Set;  import java.util.TreeMap;  import org.springframework.stereotype.Service;  import org.springframework.transaction.annotation.Transactional;  import bean.Abstract\_task;  import bean.College;  import bean.ConstantString;  import bean.Course;  import bean.Student\_class;  import biz.SearchFunctionBiz;  import net.sf.json.JSONArray;  import net.sf.json.JSONObject;  import net.sf.json.util.JSONStringer;  import service.mysqlservice.MysqlUtil;  @Service  public class SearchFunctionService implements SearchFunctionBiz {  public String getAllCollege() {  /\* 调用自定义软件接口新增数据S \*/  MysqlUtil util = new MysqlUtil();  String tableName = "college";  /\* 调用自定义软件接口新增数据E \*/  String result = "";  JSONStringer stringer = new JSONStringer();// 相当于开头的[号  stringer.array();  ResultSet rs = util.select(tableName, null, null);  try {  while (rs.next()) {// 每循环一次生成一个{"college\_name":"学院名i","college\_number":"学院编号i"}  stringer.object().key("college\_name")  .value((String) rs.getString("college\_name"))  .key("college\_number")  .value((String) rs.getString("college\_number"))  .endObject();  }  } catch (SQLException e) {  stringer.object().key("college\_name").value("数据丢失")  .key("college\_name").value("数据丢失").endObject();  e.printStackTrace();  }  stringer.endArray();// 相当于结尾的]号  result = stringer.toString();  System.out.println(result);  return result;  // return  // "[{\"college\_name\":\"软件与信息安全学院\",\"college\_number\":\"001\"}]";  }  public String getTeachersByCollege(College college) {  /\* 调用自定义软件接口新增数据S \*/  MysqlUtil util = new MysqlUtil();  String tableName = "college\_relationship\_teacher";  Map<String, Object> conditionalParams = new HashMap<String, Object>();  conditionalParams.put("college\_number", college.getCollege\_number());  /\* 调用自定义软件接口新增数据E \*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  ResultSet rs = util.select(tableName, null, conditionalParams);  try {  ResultSet rs2 = null;  while (rs.next()) {  tableName = "teacher";  conditionalParams.clear();  conditionalParams.put("t\_number", rs.getString("t\_number"));  rs2 = util.select(tableName, null, conditionalParams);  if (rs2.next()) {  stringer.object().key("t\_name")  .value((String) rs2.getString("t\_name"))  .key("t\_number")  .value((String) rs2.getString("t\_number"))  .endObject();  }  }  } catch (SQLException e) {  stringer.object().key("t\_name").value("数据丢失").key("t\_name")  .value("数据丢失").endObject();  e.printStackTrace();  }  stringer.endArray();  result = stringer.toString();  // response.setContentType("text/json; charset=UTF-8");//不生效  System.out.println(result);  return result;  }  public String getClassesByCollege(College college) {  /\* 调用自定义软件接口新增数据S \*/  MysqlUtil util = new MysqlUtil();  String tableName = "college\_relationship\_class";  Map<String, Object> conditionalParams = new HashMap<String, Object>();  conditionalParams.put("college\_number", college.getCollege\_number());  /\* 调用自定义软件接口新增数据E \*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  ResultSet rs = util.select(tableName, null, conditionalParams);  try {  ResultSet rs2 = null;  while (rs.next()) {  tableName = "student\_class";  conditionalParams.clear();  conditionalParams.put("class\_number",  rs.getString("class\_number"));  rs2 = util.select(tableName, null, conditionalParams);  if (rs2.next()) {  stringer.object().key("class\_name")  .value((String) rs2.getString("class\_name"))  .key("class\_number")  .value((String) rs2.getString("class\_number"))  .endObject();  }  }  } catch (SQLException e) {  stringer.object().key("t\_name").value("数据丢失").key("t\_name")  .value("数据丢失").endObject();  e.printStackTrace();  }  stringer.endArray();  result = stringer.toString();  // response.setContentType("text/json; charset=UTF-8");//不生效  System.out.println(result);  return result;  }  public String existTask(Abstract\_task task) {  JSONStringer stringer = new JSONStringer();//  stringer.array();  MysqlUtil util = new MysqlUtil();  String tableName = "abstract\_task";  Map<String, Object> conditionalParams = new HashMap<String, Object>();  conditionalParams.put("task\_number", task.getTask\_number());  System.out.println("task\_number:" + task.getTask\_number());  ResultSet rs = util.select(tableName, null, conditionalParams);  try {  if (rs.next()) {  stringer.object().key("result").value("true").endObject();  System.out  .println("task\_number:" + rs.getString("task\_number"));  } else {  stringer.object().key("result").value("false").endObject();  }  } catch (SQLException e) {  e.printStackTrace();  }  stringer.endArray();  String result = stringer.toString();  return result;  }  public String getAbstractTasksByTask(Abstract\_task task, String sortOrder,  Set<String> sortParams) {  MysqlUtil util = new MysqlUtil();  String tableName = "abstract\_task";  Map<String, Object> conditionalParams = new HashMap<String, Object>();  if (task.getTask\_number() != null) {  conditionalParams.put("task\_number", task.getTask\_number());  }  if (task.getTask\_title() != null) {  conditionalParams.put("task\_title", task.getTask\_title());  }  if (task.getCourse\_number() != null) {  conditionalParams.put("course\_number", task.getCourse\_number());  }  if (task.getT\_number() != null) {  conditionalParams.put("t\_number", task.getT\_number());  }  if (task.getClass\_number() != null) {  conditionalParams.put("class\_number", task.getClass\_number());  }  ResultSet rs = util.selectOrder(tableName, null, conditionalParams,  sortOrder, sortParams);  /\* 调用自定义软件接口新增数据E \*/  String result = "";  JSONStringer stringer = new JSONStringer();/\* 相当于开头的[号 \*/  stringer.array();  try {  while (rs.next()) {/\*  \* 每循环一次生成一个{"college\_name":"学院名i","college\_number"  \* :"学院编号i"}  \*/  stringer.object().key("autoId")  .value((String) rs.getString("autoId"))  .key("task\_number")  .value((String) rs.getString("task\_number"))  .key("task\_text")  .value((String) rs.getString("task\_text"))  .key("task\_type")  .value((String) rs.getString("task\_type"))  .key("task\_title")  .value((String) rs.getString("task\_title"))  .key("task\_publish\_time")  .value((String) rs.getString("task\_publish\_time"))  .key("task\_start\_time")  .value((String) rs.getString("task\_start\_time"))  .key("task\_end\_time")  .value((String) rs.getString("task\_end\_time"))  .key("t\_number")  .value((String) rs.getString("t\_number"))  .key("course\_number")  .value((String) rs.getString("course\_number"))  .key("class\_number")  .value((String) rs.getString("class\_number"))  .key("result\_static")  .value((String) rs.getString("result\_static"))  .endObject();  }  } catch (SQLException e) {  stringer.object().key("autoId").value("数据丢失").key("task\_number")  .value("数据丢失").key("task\_text").value("数据丢失")  .key("task\_type").value("数据丢失").key("task\_title")  .value("数据丢失").key("task\_publish\_time").value("数据丢失")  .key("task\_start\_time").value("数据丢失").key("task\_end\_time")  .value("数据丢失").key("t\_number").value("数据丢失")  .key("course\_number").value("数据丢失").key("class\_number")  .value("数据丢失").key("result\_static").value("数据丢失")  .endObject();  e.printStackTrace();  }  stringer.endArray();/\* 相当于结尾的]号 \*/  result = stringer.toString();  System.out.println(result);  return result;  }  public String getDelineTasksByTask\_number(String taskNumber) {  MysqlUtil util = new MysqlUtil();    String tableName="file\_task";  Set<String> LookupParams = new HashSet<>();  Map<String,Object> conditionalParams=new HashMap<String,Object>();  LookupParams.add("task\_number");  conditionalParams.put("task\_number", "1");  conditionalParams.put("task\_number", taskNumber);  String result = "";  JSONStringer stringer = new JSONStringer();//相当于开头的[号  stringer.array();  ResultSet rs=util.select(tableName, null, conditionalParams);  try {  while(rs.next()){//每循环一次生成一个{"college\_name":"学院名i","college\_number":"学院编号i"}  stringer.object().  key("task\_number").value((String)rs.getString("task\_number")).  key("file\_name").value((String)rs.getString("file\_name")).  key("file\_path").value((String)rs.getString("file\_path")).  key("download\_count").value((Integer)rs.getInt("download\_count")).  endObject();  }  } catch (SQLException e) {  stringer.object().  key("task\_number").value("数据丢失").  key("file\_name").value("数据丢失").  key("file\_path").value("数据丢失").  key("download\_count").value("数据丢失").  endObject();  e.printStackTrace();  }    stringer.endArray();//相当于结尾的]号  result=stringer.toString();  System.out.println(result);  // TODO Auto-generated method stub  return null;  }  public String getOnlineTasksByTask\_number(String taskNumber) {  System.out.println("=============检索在线作业题目S===============");  JSONArray allMessageArray=new JSONArray();  Map<String,String> allMessageMap=new HashMap<String,String>();    String result="";  MysqlUtil util=new MysqlUtil();  Map<String,String> typeMap=new HashMap<String,String>();//用于判断re集合有什么字段  String tableName="abstract\_task";  Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("task\_number", taskNumber);  ResultSet rs;//rs记录抽象作业信息  rs=util.select(tableName, null, conditionalParams);  ResultSet rs2;//rs2记录该在线作业的题型的表名  ResultSet rs3;//rs3记录某个题型的所有题目  try {  if(rs.next()){  result=rs.getString("task\_type");  //System.out.println("task\_type:"+rs.getString("task\_type"));  if(result.equals(ConstantString.ONLINE)){//判断是否为在线课程  tableName="online\_task";  rs2=util.select(tableName, null, conditionalParams);  while(rs2.next()){  tableName=rs2.getString("question\_type");  System.out.println("当前查找的题型："+tableName);  rs3=util.select(tableName, null, conditionalParams);  typeMap=util.detectType(tableName);//获取当前查询的表有什么字段，key为字段名，value为字段类型    ArrayList<String> allQuestionTypeList=new ArrayList<String>();  JSONArray curQuestionTypeArray=new JSONArray();  Map<String,String> curQuestionTypeMap=new HashMap<String,String>();  String curQuestionTypeString="";    while(rs3.next()){  for(String key : typeMap.keySet()){//获取当前查询的表有什么字段  //if(typeMap.get(key).equals("VARCHAR"))  curQuestionTypeMap.put(key, rs3.getString(key));  }  System.out.println(JSONObject.fromObject(curQuestionTypeMap).toString());  allQuestionTypeList.add(JSONObject.fromObject(curQuestionTypeMap).toString());  }  curQuestionTypeArray=JSONArray.fromObject(allQuestionTypeList);  curQuestionTypeString=curQuestionTypeArray.toString();  allMessageMap.put(tableName, curQuestionTypeString);  //System.out.println("allMessageMap: key: "+tableName+" value: "+curQuestionTypeString);    }  }  else{    }  }  else{  System.out.println("没有该作业："+taskNumber);  }  } catch (SQLException e) {  e.printStackTrace();  }  allMessageArray=JSONArray.fromObject(allMessageMap);  System.out.println("在线作业"+taskNumber+": "+allMessageArray);  System.out.println("=============检索在线作业题目E===============");  return allMessageArray.toString();  }  public String getOneStudentCompletion\_DelineTask\_ByTask\_numberAndS\_number(String task\_number, String s\_number) {  MysqlUtil util = new MysqlUtil();    String tableName = "file\_task\_answer";  Set<String> LookupParams = new HashSet<>();  Map<String, Object> conditionalParams = new HashMap<String, Object>();  LookupParams.add("task\_number");  conditionalParams.put("task\_number", task\_number);  LookupParams.add("s\_number");  conditionalParams.put("s\_number", s\_number);  String result = "";  JSONStringer stringer = new JSONStringer();  stringer.array();  ResultSet rs = util.select(tableName, null, conditionalParams);  try {  while (rs.next()) {// 每循环一次生成一个{"college\_name":"学院名i","college\_number":"学院编号i"}  stringer.object().key("task\_number")  .value((String) rs.getString("task\_number"))  .key("s\_number")  .value((String) rs.getString("s\_number"))  .key("file\_name")  .value((String) rs.getString("file\_name"))  .key("file\_path")  .value((String) rs.getString("file\_path"))  .key("file\_upload\_time")  .value((String) rs.getString("file\_path")).endObject();  }  } catch (SQLException e) {  stringer.object().key("task\_number").value("数据丢失").key("s\_number")  .value("数据丢失").key("file\_name").value("数据丢失")  .key("file\_path").value("数据丢失").key("file\_upload\_time")  .value("数据丢失").endObject();  e.printStackTrace();  }  stringer.endArray();// 相当于结尾的]号  result = stringer.toString();  System.out.println(result);  return null;  }  public String getOneStudentCompletion\_OnlineTask\_ByTask\_numberAndS\_number(  String task\_number, String s\_number) {  System.out.println("=============检索学生"+s\_number+"在线作业答题S===============");  JSONArray allMessageArray=new JSONArray();  Map<String,String> allMessageMap=new HashMap<String,String>();    String result="";  MysqlUtil util=new MysqlUtil();  Map<String,String> typeMap=new HashMap<String,String>();//用于判断re集合有什么字段  String tableName="abstract\_task";  Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("task\_number", task\_number);  ResultSet rs;//rs记录抽象作业信息  rs=util.select(tableName, null, conditionalParams);  ResultSet rs2;//rs2记录该在线作业的题型的表名  ResultSet rs3;//rs3记录某个题型的所有题目  try {  if(rs.next()){  result=rs.getString("task\_type");  //System.out.println("task\_type:"+rs.getString("task\_type"));  if(result.equals(ConstantString.ONLINE)){//判断是否为在线课程  tableName="online\_task";  rs2=util.select(tableName, null, conditionalParams);  while(rs2.next()){  int length=rs2.getString("question\_type").split("\_").length;  if(length==2)  tableName=rs2.getString("question\_type").split("\_")[0]+"\_answer";  if(length>2){  int originalLength=rs2.getString("question\_type").length();  int lastStringLength=rs2.getString("question\_type").split("\_")[length-1].length();  int needCutLength=lastStringLength;  tableName=rs2.getString("question\_type").substring(0,originalLength-needCutLength);  //System.out.println(rs2.getString("question\_type"));  //System.out.println(tableName);  tableName=tableName+"answer";  //System.out.println(tableName);  //System.out.println(originalLength);  //System.out.println(lastStringLength);  //System.out.println(needCutLength);    }  conditionalParams.put("s\_number", s\_number);  System.out.println("当前查找的题型："+tableName);  rs3=util.select(tableName, null, conditionalParams);  typeMap=util.detectType(tableName);//获取当前查询的表有什么字段，key为字段名，value为字段类型    ArrayList<String> allQuestionTypeList=new ArrayList<String>();  JSONArray curQuestionTypeArray=new JSONArray();  Map<String,String> curQuestionTypeMap=new HashMap<String,String>();  String curQuestionTypeString="";    while(rs3.next()){  for(String key : typeMap.keySet()){//获取当前查询的表有什么字段  //if(typeMap.get(key).equals("VARCHAR"))  curQuestionTypeMap.put(key, rs3.getString(key));  }  System.out.println(JSONObject.fromObject(curQuestionTypeMap).toString());  allQuestionTypeList.add(JSONObject.fromObject(curQuestionTypeMap).toString());  }  curQuestionTypeArray=JSONArray.fromObject(allQuestionTypeList);  curQuestionTypeString=curQuestionTypeArray.toString();  allMessageMap.put(tableName, curQuestionTypeString);  //System.out.println("allMessageMap: key: "+tableName+" value: "+curQuestionTypeString);    }  }  else{    }  }  else{  System.out.println("没有该作业："+task\_number);  }  } catch (SQLException e) {  e.printStackTrace();  }  allMessageArray=JSONArray.fromObject(allMessageMap);  System.out.println("在线作业"+task\_number+": "+allMessageArray);  System.out.println("=============检索学生"+s\_number+"在线作业答题E===============");  return allMessageArray.toString();  }  public String getStudentsByClass(Student\_class studentClass) {  // TODO Auto-generated method stub  return null;  }  public String getStudentsByCourse(Course course) {  // TODO Auto-generated method stub  return null;  }  public String getStydentsByTask\_number(String taskNumber) {  MysqlUtil util=new MysqlUtil();  String task\_type = null;  String question\_type = null;  String class\_number = null;  String s\_number = null;  Map<String,String> student\_sum=new TreeMap<String,String>();  String tableName = "abstract\_task";    Set<String> LookupParams = new HashSet<>();  LookupParams.add("task\_type");  LookupParams.add("class\_number");  Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("task\_number",taskNumber);  ResultSet rs=util.select(tableName,LookupParams,conditionalParams);  try {  while(rs.next()){  task\_type = rs.getString("task\_type");  class\_number = rs.getString("class\_number");  System.out.println("task\_type:"+task\_type+"------"+"class\_number:"+class\_number);  }  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  if(task\_type.equals("online")) {  /\*---------------获取question\_type-------------------\*/  String tableName1 = "online\_task";    Set<String> LookupParams1 = new HashSet<>();  LookupParams1.add("question\_type");    Map<String,Object> conditionalParams1=new HashMap<String,Object>();  conditionalParams1.put("task\_number",taskNumber);    ResultSet rs1 = util.select(tableName1,LookupParams1,conditionalParams1);    try {  while(rs1.next()){  question\_type = rs1.getString("question\_type").replace("question","answer");  System.out.println("question\_type:"+question\_type);  }  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }          /\*--------------------获取所有学生----------------------\*/    String class\_relationship\_student = "class\_relationship\_student";    Set<String> LookupParams3 = new HashSet<>();  LookupParams3.add("s\_number");    Map<String,Object> conditionalParams3=new HashMap<String,Object>();  conditionalParams3.put("class\_number",class\_number);    Set<String> sortParams3 = new HashSet<>();  sortParams3.add("s\_number");    ResultSet rs3 = util.selectOrder(class\_relationship\_student,LookupParams3,conditionalParams3,"ASC",sortParams3);    try {  while(rs3.next()){  s\_number = rs3.getString("s\_number");    student\_sum.put(s\_number,"false");  System.out.println("s\_number:"+s\_number);  }  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    for (String in : student\_sum.keySet()) {  //map.keySet()返回的是所有key的值  String str = student\_sum.get(in);//得到每个key多对用value的值  System.out.println(in + " " + str);  }    /\*--------------整合-----------------------\*/  Set<String> LookupParams2 = new HashSet<>();  LookupParams2.add("s\_number");    Map<String,Object> conditionalParams2=new HashMap<String,Object>();  conditionalParams2.put("task\_number",taskNumber);    Set<String> sortParams = new HashSet<>();  sortParams.add("s\_number");  ResultSet rs2 = util.selectOrder(question\_type,LookupParams2,conditionalParams2,"ASC",sortParams);  try {  while(rs2.next()){  s\_number = rs2.getString("s\_number");    student\_sum.put(s\_number,"true");  System.out.println("s\_number:"+s\_number);  }  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  for (String in : student\_sum.keySet()) {  //map.keySet()返回的是所有key的值  String str = student\_sum.get(in);//得到每个key多对用value的值  System.out.println(in + " " + str);  }  /\*--------------获取s\_name-----------------------\*/  Set<String> LookupParams4 = new HashSet<>();  LookupParams4.add("s\_name");    Map<String,Object> conditionalParams4=new HashMap<String,Object>();    /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();/\*相当于开头的[号\*/  stringer.array();  ResultSet rs4;  String s\_name = null;  try {    for (String in : student\_sum.keySet()) {  //map.keySet()返回的是所有key的值  conditionalParams4.put("s\_number", in);  rs4 = util.select("student",LookupParams4,conditionalParams4);  rs4.next();  s\_name = rs4.getString("s\_name");  stringer.object().  key("s\_number").value(in).  key("s\_name").value(s\_name).  key("static").value(student\_sum.get(in)).  endObject();  }    } catch (SQLException e) {  stringer.object().  key("s\_number").value("数据丢失").  key("s\_name").value("数据丢失").  key("static").value("数据丢失").  endObject();  e.printStackTrace();  }    stringer.endArray();/\*相当于结尾的]号\*/  result=stringer.toString();  System.out.println(result);    return result;  }else if(task\_type.equals("deline")) {  /\*--------------------获取所有学生----------------------\*/    String class\_relationship\_student = "class\_relationship\_student";    Set<String> LookupParams3 = new HashSet<>();  LookupParams3.add("s\_number");    Map<String,Object> conditionalParams3=new HashMap<String,Object>();  conditionalParams3.put("class\_number",class\_number);    Set<String> sortParams3 = new HashSet<>();  sortParams3.add("s\_number");    ResultSet rs3 = util.selectOrder(class\_relationship\_student,LookupParams3,conditionalParams3,"ASC",sortParams3);    try {  while(rs3.next()){  s\_number = rs3.getString("s\_number");    student\_sum.put(s\_number,"false");  System.out.println("s\_number:"+s\_number);  }  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    for (String in : student\_sum.keySet()) {  //map.keySet()返回的是所有key的值  String str = student\_sum.get(in);//得到每个key多对用value的值  System.out.println(in + " " + str);  }    /\*--------------整合-----------------------\*/  Set<String> LookupParams2 = new HashSet<>();  LookupParams2.add("s\_number");    Map<String,Object> conditionalParams2=new HashMap<String,Object>();  conditionalParams2.put("task\_number",taskNumber);    Set<String> sortParams = new HashSet<>();  sortParams.add("s\_number");  ResultSet rs2 = util.selectOrder("file\_task\_answer",LookupParams2,conditionalParams2,"ASC",sortParams);  try {  while(rs2.next()){  s\_number = rs2.getString("s\_number");    student\_sum.put(s\_number,"true");  System.out.println("s\_number:"+s\_number);  }  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  for (String in : student\_sum.keySet()) {  //map.keySet()返回的是所有key的值  String str = student\_sum.get(in);//得到每个key多对用value的值  System.out.println(in + " " + str);  }  /\*--------------获取s\_name-----------------------\*/  Set<String> LookupParams4 = new HashSet<>();  LookupParams4.add("s\_name");    Map<String,Object> conditionalParams4=new HashMap<String,Object>();    /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();/\*相当于开头的[号\*/  stringer.array();  ResultSet rs4;  String s\_name = null;    try {    for (String in : student\_sum.keySet()) {  //map.keySet()返回的是所有key的值  conditionalParams4.put("s\_number", in);  rs4 = util.select("student",LookupParams4,conditionalParams4);  rs4.next();  s\_name = rs4.getString("s\_name");  stringer.object().  key("s\_number").value(in).  key("s\_name").value(s\_name).  key("static").value(student\_sum.get(in)).  endObject();  }    } catch (SQLException e) {  stringer.object().  key("s\_number").value("数据丢失").  key("s\_name").value("数据丢失").  key("static").value("数据丢失").  endObject();  e.printStackTrace();  }    stringer.endArray();/\*相当于结尾的]号\*/  result=stringer.toString();  System.out.println(result);    return result;    }  return "";  }  public String getOtherMessageByTask\_number(String taskNumber) {  MysqlUtil util=new MysqlUtil();  Map<String,Object> map\_sum=new HashMap<String,Object>();    String course\_number = null;  String t\_number = null;  String class\_number = null;  String course\_name = null;  String t\_name = null;  String class\_name = null;  String tableName="abstract\_task";    Set<String> LookupCname\_Cnumber = new HashSet<>();  LookupCname\_Cnumber.add("course\_number");  LookupCname\_Cnumber.add("t\_number");  LookupCname\_Cnumber.add("class\_number");    Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("task\_number",taskNumber);  ResultSet rs=util.select(tableName,LookupCname\_Cnumber,conditionalParams);    try {  while(rs.next()){  course\_number = rs.getString("course\_number");  t\_number = rs.getString("t\_number");  class\_number = rs.getString("class\_number");  System.out.println(rs.getString("course\_number"));  System.out.println(rs.getString("t\_number"));  System.out.println(rs.getString("class\_number"));  }  } catch (SQLException e) {    e.printStackTrace();  }  /\*-------------------------寻找course\_name----------------------------\*/  String tableName1 = "course";    Set<String> LookupCoursename = new HashSet<>();  LookupCoursename.add("course\_name");    Map<String,Object> conditionalParams1=new HashMap<String,Object>();  conditionalParams1.put("course\_number",course\_number);    ResultSet rs1=util.select(tableName1,LookupCoursename,conditionalParams1);  try {  while(rs1.next()){  course\_name = rs1.getString("course\_name");  System.out.println(course\_name);  }  } catch (SQLException e) {    e.printStackTrace();  }  /\*-------------------------寻找t\_name----------------------------\*/  String tableName2 = "teacher";    Set<String> LookupTname = new HashSet<>();  LookupTname.add("t\_name");    Map<String,Object> conditionalParams2=new HashMap<String,Object>();  conditionalParams2.put("t\_number",t\_number);    ResultSet rs2=util.select(tableName2,LookupTname,conditionalParams2);  try {  while(rs2.next()){  t\_name = rs2.getString("t\_name");  System.out.println(t\_name);  }  } catch (SQLException e) {    e.printStackTrace();  }  /\*-------------------------寻找class\_name----------------------------\*/  String tableName3 = "student\_class";    Set<String> LookupClassname = new HashSet<>();  LookupClassname.add("class\_name");    Map<String,Object> conditionalParams3=new HashMap<String,Object>();  conditionalParams3.put("class\_number",class\_number);    ResultSet rs3=util.select(tableName3,LookupClassname,conditionalParams3);  try {  while(rs3.next()){  class\_name = rs3.getString("class\_name");  System.out.println(class\_name);  }  } catch (SQLException e) {    e.printStackTrace();  }    /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();/\*相当于开头的[号\*/  stringer.array();    stringer.object().  key("course\_number").value(course\_number).  key("course\_name").value(course\_name).  key("t\_number").value(t\_number).  key("t\_name").value(t\_name).  key("class\_number").value(class\_number).  key("class\_name").value(class\_name).  endObject();  stringer.endArray();/\*相当于结尾的]号\*/  result=stringer.toString();  System.out.println(result);  return result;  }  public String getStudentScoreAndCommentByS\_number(String studentNumber) {  // TODO Auto-generated method stub  String s\_number = studentNumber;  MysqlUtil util=new MysqlUtil();  String tableName="score\_result";  String result = "";  JSONStringer stringer = new JSONStringer();  stringer.array();  Set<String> LookupParams = new HashSet<>();  LookupParams.add("s\_number");    Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("s\_number", s\_number);  ResultSet rs = util.select(tableName, null, conditionalParams);  try {  while(rs.next()){  stringer.object().key("task\_number").value((String)rs.getString("task\_number")).key("s\_number").value((String)rs.getString("s\_number")).key("total\_score").value((String)rs.getString("total\_score")).key("result\_comment").value((String)rs.getString("result\_comment")).endObject();  }  }catch(SQLException e){  stringer.object().key("task\_number").value("数据丢失").key("s\_number").value("数据丢失").key("total\_score").value("数据丢失").key("result\_comment").value("数据丢失").endObject();  e.printStackTrace();  }  stringer.endArray();  result = stringer.toString();  System.out.println(result);  return result;  }  @Override  public String getStudentByS\_number(String s\_number) {  MysqlUtil util=new MysqlUtil();  String tableName="student";  String result = "";  JSONStringer stringer = new JSONStringer();  stringer.array();  Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("s\_number", s\_number);  ResultSet rs = util.select(tableName, null, conditionalParams);  try {  if(rs.next()){  stringer.object().  key("s\_number").value(rs.getString("s\_number")).  key("s\_name").value(rs.getString("s\_name")).  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  }  } catch (SQLException e) {  stringer.object().  key("result").value("false").  endObject();  e.printStackTrace();  }  stringer.endArray();  result = stringer.toString();  return result;  }  @Override  public String getCourseByCourse\_number(String course\_number) {  MysqlUtil util=new MysqlUtil();  String tableName="course";  String result = "";  JSONStringer stringer = new JSONStringer();  stringer.array();  Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("course\_number", course\_number);  ResultSet rs = util.select(tableName, null, conditionalParams);  try {  if(rs.next()){  stringer.object().  key("course\_number").value(rs.getString("course\_number")).  key("course\_name").value(rs.getString("course\_name")).  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  }  } catch (SQLException e) {  stringer.object().  key("result").value("false").  endObject();  e.printStackTrace();  }  stringer.endArray();  result = stringer.toString();  return result;  }  } |

##### 4.1.3.3更新数据模块的持久层

（1）继承的接口

|  |
| --- |
| package biz;  public interface UpdateFunctionBiz {  public String updateStudent(Student student);  public String updateTeacher(Teacher teacher);  public String updateCourse(Course course);  public String updateCollege(College college);  public String updateClass(Student\_class studentClass);  public String updateAbstractTask(Abstract\_task task);  public String updateDelineTask(File\_task task);  public String updateOnlineTask(Online\_task task);  public String updateScoreResult(Score\_result scoreResult);    } |

（2）代码实现

|  |
| --- |
| package service.logicservice;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.HashMap;  import java.util.HashSet;  import java.util.Map;  import java.util.Set;  import net.sf.json.util.JSONStringer;  import org.springframework.stereotype.Service;  import org.springframework.transaction.interceptor.TransactionAspectSupport;  import service.mysqlservice.MysqlUtil;  import bean.Abstract\_task;  import bean.College;  import bean.Course;  import bean.File\_task;  import bean.Online\_task;  import bean.Score\_result;  import bean.Student;  import bean.Student\_class;  import bean.Teacher;  import biz.UpdateFunctionBiz;  @Service  public class UpdateFunctionService implements UpdateFunctionBiz {  @Override  public String updateStudent(Student student) {  // TODO Auto-generated method stub  JSONStringer stringer = new JSONStringer();//  stringer.array();  MysqlUtil util = new MysqlUtil();  try {  util.getConn().setAutoCommit(false);  String tableName = "student";  Map<String, Object> conditionalParams = new HashMap<String, Object>();  Map<String,Object> updateParams=new HashMap<String,Object>();  Set<String> LookupParams = new HashSet<>();  LookupParams.add("s\_number");  LookupParams.add("s\_name");  if((student.getS\_number())!=null){  conditionalParams.put("s\_number", student.getS\_number());      } else{  conditionalParams.put("s\_number",null);  }  updateParams.put("password", student.getPassword());  updateParams.put("s\_name", student.getS\_name());    int rs = util.update(tableName,updateParams, conditionalParams);  if(rs==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{    //依旧返回数据，也抛出异常  stringer.object().  key("result").value("false").  endObject();  /\*Exception e=new Exception();  throw e;\*/  try {  util.getConn().rollback();  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    }  } catch (SQLException e1) {  // TODO Auto-generated catch block  e1.printStackTrace();  }      stringer.endArray();  String result = stringer.toString();  System.out.println(result);  return result;    }    @Override  public String updateTeacher(Teacher teacher) {  // TODO Auto-generated method stub  // TODO Auto-generated method stub  JSONStringer stringer = new JSONStringer();//  stringer.array();  MysqlUtil util = new MysqlUtil();  try {  util.getConn().setAutoCommit(false);  String tableName = "teacher";  Map<String, Object> conditionalParams = new HashMap<String, Object>();  Map<String,Object> updateParams=new HashMap<String,Object>();  if((teacher.getT\_number())!=null){  conditionalParams.put("t\_number", teacher.getT\_number());      } else{  conditionalParams.put("t\_number",null);  }  updateParams.put("password", teacher.getPassword());  updateParams.put("t\_name", teacher.getT\_name());    int rs = util.update(tableName,updateParams, conditionalParams);  if(rs==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{    //依旧返回数据，也抛出异常  stringer.object().  key("result").value("false").  endObject();  /\*Exception e=new Exception();  throw e;\*/  try {  util.getConn().rollback();  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    }  } catch (SQLException e1) {  // TODO Auto-generated catch block  e1.printStackTrace();  }      stringer.endArray();  String result = stringer.toString();  System.out.println(result);  return result;  }  @Override  public String updateCourse(Course course) {  // TODO Auto-generated method stub  JSONStringer stringer = new JSONStringer();//  stringer.array();  MysqlUtil util = new MysqlUtil();  try {  util.getConn().setAutoCommit(false);  String tableName = "course";  Map<String, Object> conditionalParams = new HashMap<String, Object>();  Map<String,Object> updateParams=new HashMap<String,Object>();  if((course.getCourse\_number())!=null){  conditionalParams.put("course\_number", course.getCourse\_number());      } else{  conditionalParams.put("course\_number",null);  }  updateParams.put("course\_name", course.getCourse\_name());  updateParams.put("t\_number", course.getT\_number());    int rs = util.update(tableName,updateParams, conditionalParams);  if(rs==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{    //依旧返回数据，也抛出异常  stringer.object().  key("result").value("false").  endObject();  /\*Exception e=new Exception();  throw e;\*/  try {  util.getConn().rollback();  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    }  } catch (SQLException e1) {  // TODO Auto-generated catch block  e1.printStackTrace();  }      stringer.endArray();  String result = stringer.toString();  System.out.println(result);  return result;  }  @Override  public String updateCollege(College college) {  // TODO Auto-generated method stub  JSONStringer stringer = new JSONStringer();//  stringer.array();  MysqlUtil util = new MysqlUtil();  try {  util.getConn().setAutoCommit(false);  String tableName = "college";  Map<String, Object> conditionalParams = new HashMap<String, Object>();  Map<String,Object> updateParams=new HashMap<String,Object>();  if((college.getCollege\_number())!=null){  conditionalParams.put("college\_number", college.getCollege\_number());    } else{  conditionalParams.put("college\_number",null);  }  updateParams.put("college\_name", college.getCollege\_name());      int rs = util.update(tableName,updateParams, conditionalParams);  if(rs==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{    //依旧返回数据，也抛出异常  stringer.object().  key("result").value("false").  endObject();  /\*Exception e=new Exception();  throw e;\*/  try {  util.getConn().rollback();  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    }  } catch (SQLException e1) {  // TODO Auto-generated catch block  e1.printStackTrace();  }      stringer.endArray();  String result = stringer.toString();  System.out.println(result);  return result;  }  @Override  public String updateClass(Student\_class studentClass) {  // TODO Auto-generated method stub  JSONStringer stringer = new JSONStringer();//  stringer.array();  MysqlUtil util = new MysqlUtil();  try {  util.getConn().setAutoCommit(false);  String tableName = "student\_class";  Map<String, Object> conditionalParams = new HashMap<String, Object>();  Map<String,Object> updateParams=new HashMap<String,Object>();  if((studentClass.getClass\_number())!=null){  conditionalParams.put("class\_number", studentClass.getClass\_number());    } else{  conditionalParams.put("class\_number",null);  }  updateParams.put("class\_name", studentClass.getClass\_name());      int rs = util.update(tableName,updateParams, conditionalParams);  if(rs==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{    //依旧返回数据，也抛出异常  stringer.object().  key("result").value("false").  endObject();  /\*Exception e=new Exception();  throw e;\*/  try {  util.getConn().rollback();  } catch (SQLException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }    }  } catch (SQLException e1) {  // TODO Auto-generated catch block  e1.printStackTrace();  }      stringer.endArray();  String result = stringer.toString();  System.out.println(result);  return result;  }  @Override  public String updateAbstractTask(Abstract\_task task) {  // TODO Auto-generated method stub  return null;  }  @Override  public String updateDelineTask(File\_task task) {  // TODO Auto-generated method stub  return null;  }  @Override  public String updateOnlineTask(Online\_task task) {  // TODO Auto-generated method stub  return null;  }  @Override  public String updateScoreResult(Score\_result scoreResult) {  // TODO Auto-generated method stub  return null;  }  } |

##### 4.1.3.4删除数据模块的持久层

（1）继承的接口

|  |
| --- |
| package biz;  public interface DeleteFunctionBiz {  public String deleteStudent(Student student);  public String deleteTeacher(Teacher teacher);  public String deleteCourse(Course course);  public String deleteCollege(College college);  public String deleteClass(Student\_class studentClass);  public String deleteAbstractTask(Abstract\_task task);  public String deleteDelineTask(File\_task task);  public String deleteOnlineTask(Online\_task task);  } |

（2）代码实现

|  |
| --- |
| package service.logicservice;  import java.util.HashMap;  import java.util.Map;  import net.sf.json.util.JSONStringer;  import org.springframework.stereotype.Service;  import service.mysqlservice.MysqlUtil;  import bean.Abstract\_task;  import bean.College;  import bean.Course;  import bean.File\_task;  import bean.Online\_task;  import bean.Student;  import bean.Student\_class;  import bean.Teacher;  import biz.DeleteFunctionBiz;  @Service  public class DeleteFunctionService implements DeleteFunctionBiz {  @Override  public String deleteStudent(Student student) {  // TODO Auto-generated method stub  String s\_number = student.getS\_number();  MysqlUtil util = new MysqlUtil();  String[] tableName = {"student","score\_result","blank\_answer","choice\_answer","class\_relationship\_student","judgment\_answer","multiple\_choice\_answer","subjective\_answer"};  Map<String,Object> conditionalParams = new HashMap<String,Object>();  conditionalParams.put("s\_number", s\_number);  for(int i = 0;i<=7;i++){  util.delete(tableName[i], conditionalParams);  System.out.println("成功删除");  }  JSONStringer stringer = new JSONStringer();  String result = "";  stringer.object().  key("result").value("true").  endObject();  result = stringer.toString();  System.out.println(result);  return result;  }  @Override  public String deleteTeacher(Teacher teacher) {  // TODO Auto-generated method stub  String t\_number = teacher.getT\_number();  MysqlUtil util = new MysqlUtil();  String[] tableName = {"teacher","abstract\_task","course","college\_relationship\_teacher"};  Map<String,Object> conditionalParams = new HashMap<String,Object>();  conditionalParams.put("t\_number", t\_number);  for(int i = 0;i<=3;i++){  util.delete(tableName[i], conditionalParams);  System.out.println("成功删除");  }  JSONStringer stringer = new JSONStringer();  String result = "";  stringer.object().  key("result").value("true").  endObject();  result = stringer.toString();  System.out.println(result);  return result;  }  @Override  public String deleteCourse(Course course) {  MysqlUtil util=new MysqlUtil();    String tableName[]={"course","course\_relationship\_student","abstract\_task"};  Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("course\_number",course.getCourse\_number());  for(int i=0;i<4;i++){  util.delete(tableName[i], conditionalParams);  System.out.println("删除成功");  }    JSONStringer stringer = new JSONStringer();  String result = "";  stringer.object().  key("result").value("true").  endObject();  result = stringer.toString();  System.out.println(result);    return result;  }  @Override  public String deleteCollege(College college) {  MysqlUtil util = new MysqlUtil();    String tableName[]={"college","abstract\_task","college\_relationship\_class","college\_relationship\_teacher"};  Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("college\_number", college.getCollege\_number());  for(int i=0;i<4;i++){  util.delete(tableName[i], conditionalParams);  System.out.println("删除成功");  }    return null;  }  @Override  public String deleteClass(Student\_class studentClass) {  MysqlUtil util = new MysqlUtil();    String tableName[]={"student\_class","abstract\_task","college\_relationship\_class","class\_relationship\_student"};  Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("class\_number", studentClass.getClass\_number());  for(int i=0;i<4;i++){  util.delete(tableName[i], conditionalParams);  System.out.println("删除成功");  }  return null;  }  @Override  public String deleteAbstractTask(Abstract\_task task) {    // TODO Auto-generated method stub  return null;  }  @Override  public String deleteDelineTask(File\_task task) {  // TODO Auto-generated method stub  return null;  }  @Override  public String deleteOnlineTask(Online\_task task) {  // TODO Auto-generated method stub  return null;  }  } |

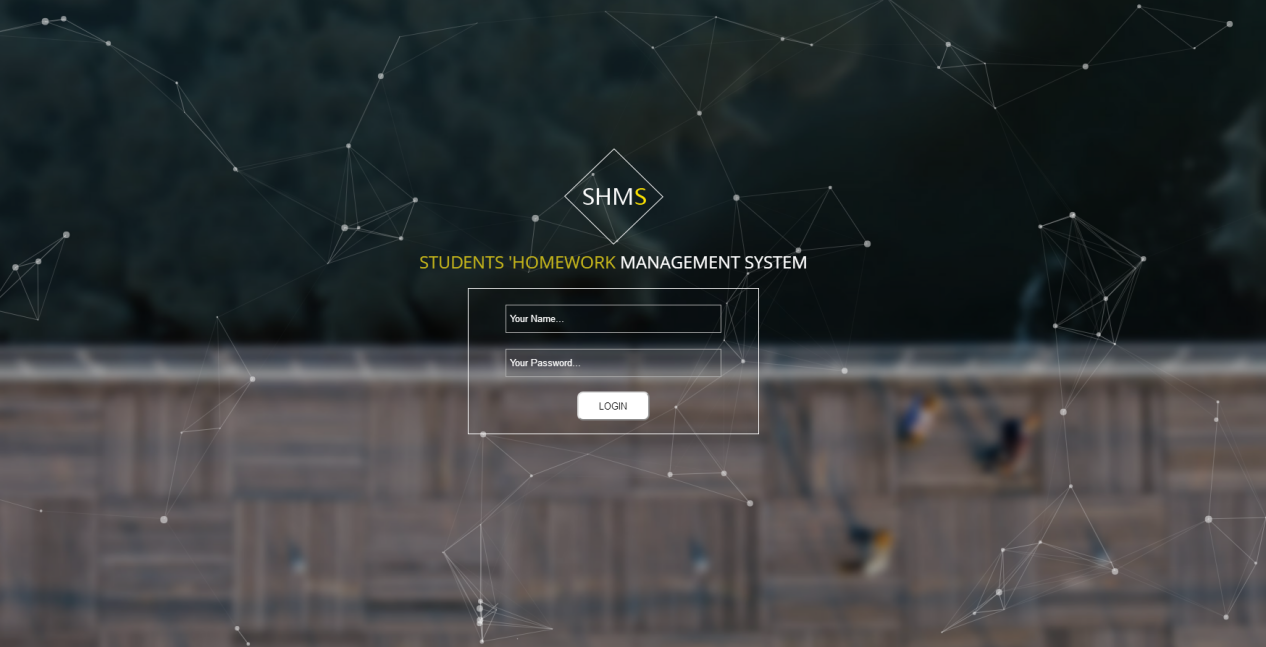
## 4.2、登录模块

在登录模块时，系统内部的相应响应操作示意图如图4-2所示。

图 4-2 系统响应示意图

### 4.2.1 表现层

（1）登录界面



输入用户名和密码，点击登录即可进入主界面。若用户名、密码错误，输入框清空，重新再输入。直到正确为止。

表 4-3 登入模块表现层JSP列表

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

（2）登录流程

login.jsp的流程图如图4-3所示。

进入游首页

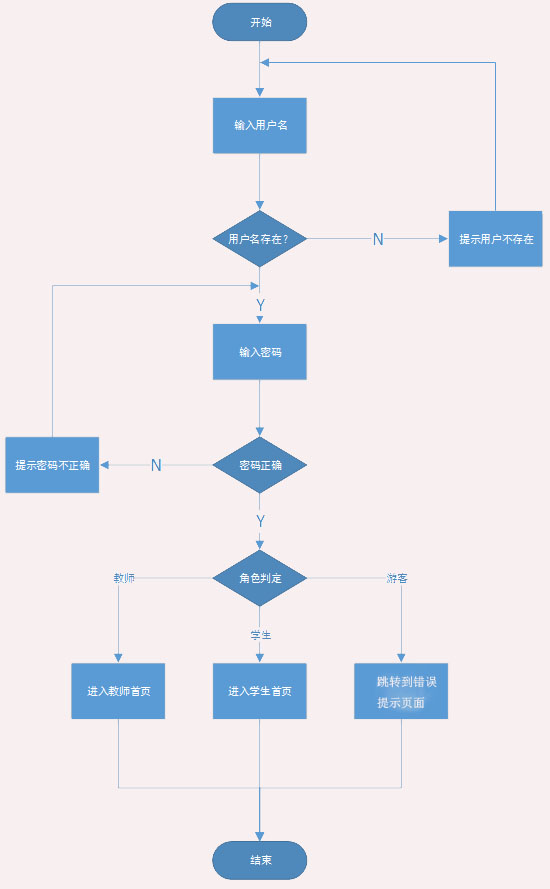


图 4-3 登录模块详细设计流程图

### 4.2.2 控制层

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

表 4-4 登入模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | LoginController | 转移说明 | 出口 |
| 登录 | LoginController.java | index | 教师身份登录成功，跳转到index.jsp页面 |
| studentMain | 学生身份登陆成功，跳转到studentMain.jsp页面 |
| loginError | 登录失败跳转到loginError.jsp页面 |

在控制层中LoginController.java的描述如下所示：

### 4.2.3 业务逻辑层

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

表 4-5登入模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 登录 | IUserService.java  UserServiceImpl.java | 调用IUserDao | 返回给Action |

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

（1）继承的接口

|  |
| --- |
| package biz;  public interface LoginBiz {  public String login(User user);  } |

（2）代码实现

|  |
| --- |
| package controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Component;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import bean.User;  import biz.LoginBiz;  import service.logicservice.LoginService;  import service.logicservice.SearchFunctionService;  //@Component  @Controller  public class LoginController implements LoginBiz {  @Autowired  private LoginService loginService;  @Override  @RequestMapping(value="/login",method=RequestMethod.POST)  public String login(User user) {  String result=loginService.login(user);  if(!result.equals("login")){  if(result.equals("teacher")){  result="index";  }  else if(result.equals("student")){  result="studentMain";  }  else if(result.equals("notDefound")){  result="loginError";  }  }      return result;  }  } |

### 4.2.4 数据持久层

登录模块的是数据持久层能对用户数据进行增删改查。登入模块数据持久层列表见表4-6所示。

表 4-6 登入模块数据持久层列表

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

|  |
| --- |
| package service.logicservice;  @Service  public class LoginService implements LoginBiz {  @Autowired  private HttpServletRequest request;  @Override  public String login(User user) {  /\*调用自定义软件接口查询数据S\*/  MysqlUtil util=new MysqlUtil();  String tableName="student";  Map<String,Object> conditionalParams=new HashMap<String,Object>();  conditionalParams.put("s\_number", user.getNumber());  conditionalParams.put("password", user.getPassword());  /\*调用自定义软件接口查询数据E\*/  String result = "login";  JSONStringer stringer = new JSONStringer();//  stringer.array();  ResultSet rs=util.select(tableName, null, conditionalParams);  try {  if(rs.next()){  result=tableName;  request.getSession().setAttribute("userName", rs.getString("s\_name"));  request.getSession().setAttribute("userNumber", rs.getString("s\_number"));  request.getSession().setAttribute("role", "student");  System.out.println("学生登录成功");  }  else{  tableName="teacher";  conditionalParams.clear();  conditionalParams.put("t\_number", user.getNumber());  conditionalParams.put("password", user.getPassword());  rs=util.select(tableName, null, conditionalParams);  if(rs.next()){    result=tableName;  request.getSession().setAttribute("userName", rs.getString("t\_name"));  request.getSession().setAttribute("userNumber", rs.getString("t\_number"));  request.getSession().setAttribute("role", "teacher");  System.out.println("教师登录成功");  }  else{  System.out.println("============================");    result="notDefound";  System.out.println("没有该用户:"+user.getNumber()+" 密码:"+user.getPassword());  }  }  } catch (SQLException e) {  e.printStackTrace();  System.out.println("登录查询异常");  }    return result;  }  } |

### 4.2.5 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 4-7登入模块域模型层列表

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

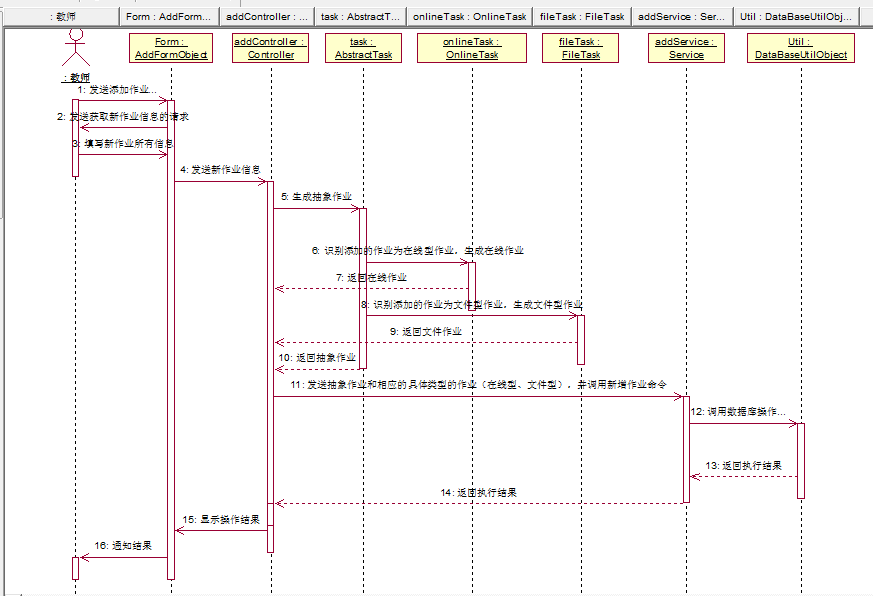
User.java主要属性与方法：

|  |
| --- |
| package bean;  public class User {  private String number;  private String name;  private String password;  public String getNumber() {  return number;  }  public void setNumber(String number) {  this.number = number;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }      } |

## 4.3、发布作业模块

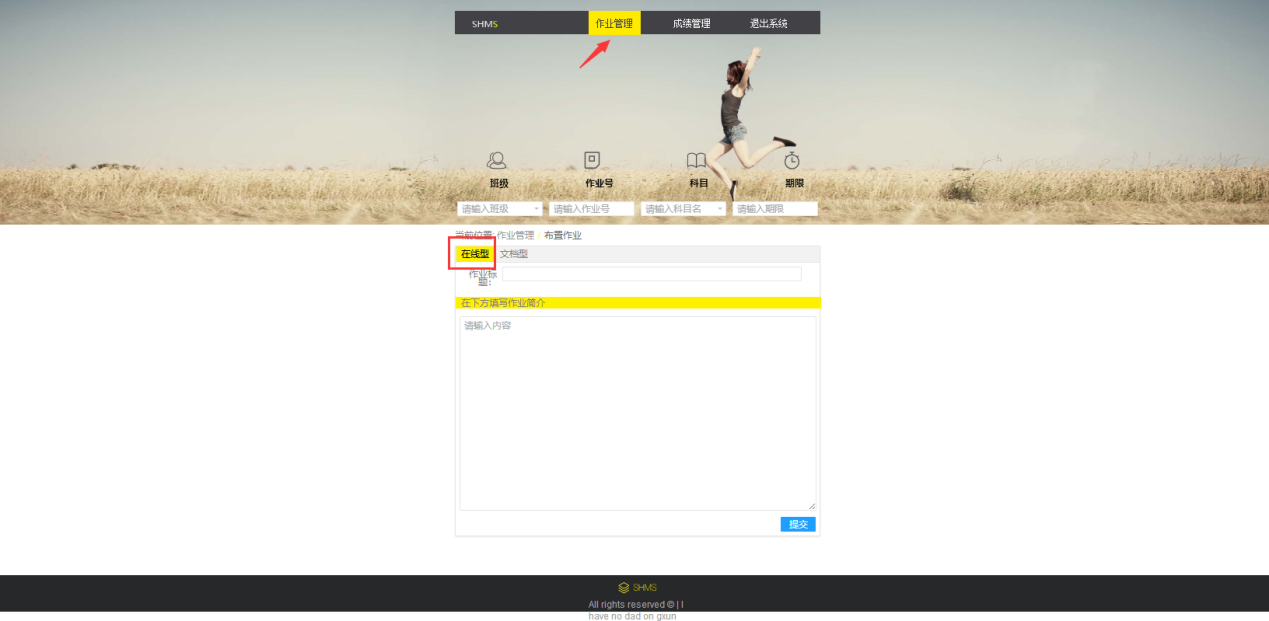
|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 发布作业页面 | publish.jsp  editOnlineTask.jsp | publish.jsp页面用于发布作业的简要信息，如果是选择发布在线型作业，那么就跳转到editOnlineTask.jsp页面进行在线作业的编辑和发布；如果是发布线下型作业，那么久在本页面选择作业文件上传并发布作业 |

图 4-3-1 发表作业序列图



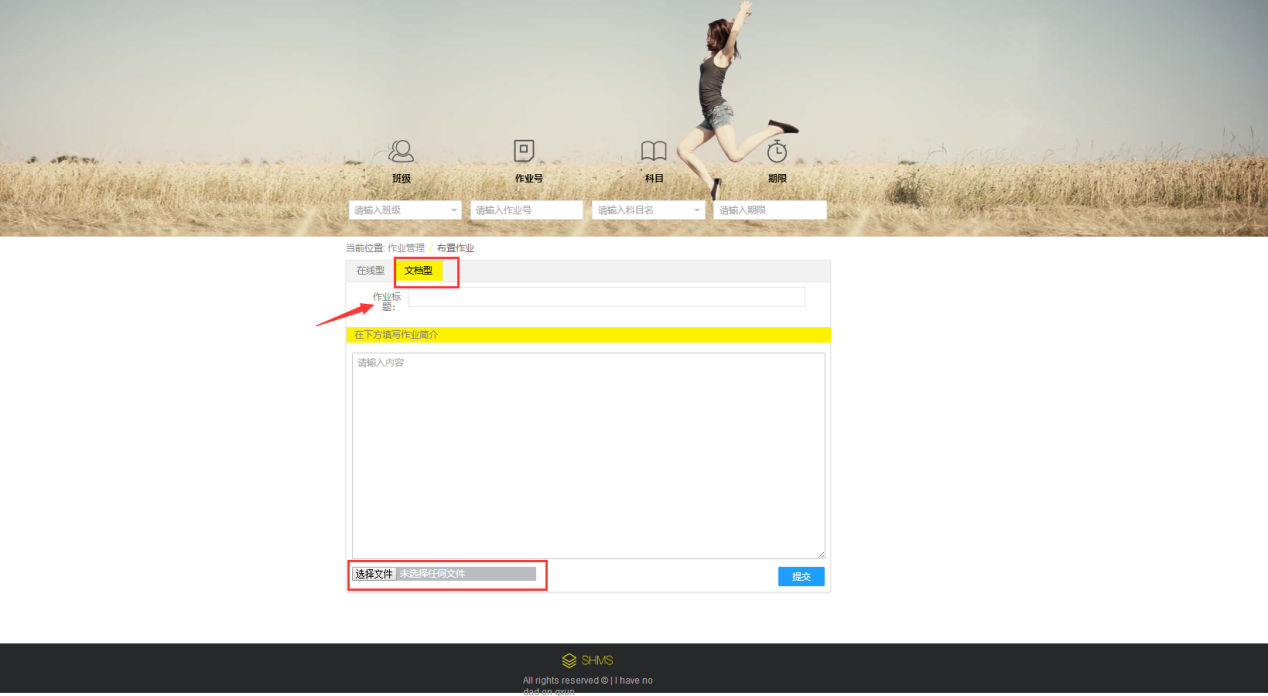
### 4.3.1 表现层

（1）布置作业-在线型

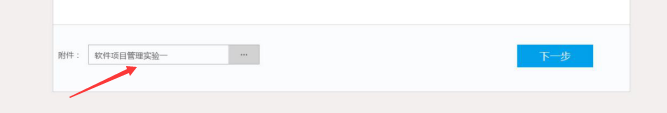


首页作业管理点击布置作业，进入布置作业页面，选择在线性，布置在线作业。在页面顶部录入作业信息，然后写作业标题，在空白区布置作业。完成后点击下一步。

（2）布置作业-下线型



点击文档型，可以编写文档作业，添加标题和文本内容。也可以添加附件，并发布。



添加文档型作业后，附件栏会显示文档名，以便想改附件时还可以改

### 4.3.2 控制层

AddFunctionController.java

|  |
| --- |
| @Override  @RequestMapping(value="/publish\_file\_task",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String addFileTask(HttpServletRequest request, Abstract\_task task,File\_task fileTask) throws Exception {  fileTask.setDownload\_count(0);  fileTask.setFile\_name(fileMessage.get("fileName"));  fileTask.setFile\_path(fileMessage.get("relativePath"));  String result=addService.addFileTask(request, task,fileTask);  //return "{\"result\":\"true\"}";  return result;  }  @Override  @RequestMapping(value="/publish\_online\_task",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String addOnlineTask(HttpServletRequest request, Abstract\_task task, Online\_task onlineTask)  throws Exception {  System.out.println("task\_number:"+task.getTask\_number());  String result=addService.addOnlineTask(request, task, onlineTask);  System.out.println(result);  return result;  } |

### 4.3.3 业务逻辑层

AddFunctionService.java

|  |
| --- |
| @Override  @Transactional(rollbackFor=Exception.class)  public String addFileTask(HttpServletRequest request, Abstract\_task task,File\_task fileTask) throws Exception {    Date date= new Date();//创建一个时间对象，获取到当前的时间  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");//设置时间显示格式  String str = sdf.format(date);//将当前时间格式化为需要的类型    System.out.println("t\_number:"+task.getT\_number());  System.out.println("task\_number:"+task.getTask\_number());  System.out.println("task\_title:"+task.getTask\_title());  System.out.println("task\_end\_timer:"+task.getTask\_end\_time());  System.out.println("task\_type:"+task.getTask\_type());  System.out.println("task\_text:"+task.getTask\_text());  System.out.println("course\_number:"+task.getCourse\_number());  System.out.println("class\_number:"+task.getClass\_number());    System.out.println("file\_name:"+fileTask.getFile\_name());  System.out.println("file\_path:"+fileTask.getFile\_path());    JSONStringer stringer = new JSONStringer();//  stringer.array();  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="abstract\_task";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("t\_number", task.getT\_number());  insertParams.put("course\_number", task.getClass\_number());  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("task\_text", task.getTask\_text());  insertParams.put("task\_title", task.getTask\_title());  insertParams.put("task\_type", task.getTask\_type());  insertParams.put("class\_number", task.getClass\_number());  insertParams.put("task\_end\_time", task.getTask\_end\_time());    insertParams.put("task\_publish\_time", str);  insertParams.put("task\_start\_time", str);  /\*调用自定义软件接口新增数据E\*/  if(util.insert(tableName, insertParams)==1){  tableName="file\_task";  insertParams.clear();  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("file\_name", fileTask.getFile\_name());  insertParams.put("file\_path", fileTask.getFile\_path());  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  util.getConn().rollback();  }  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  String result=stringer.toString();  return result;  }  @Override  public String addOnlineTask(HttpServletRequest request,Abstract\_task task, Online\_task onlineTask) throws Exception {  boolean allRight=true;  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="abstract\_task";  Map<String,Object> insertParams=new HashMap<String,Object>();  /\*添加进抽象在线作业表，该表可以存在线作业和文件作业\*/  Date date= new Date();//创建一个时间对象，获取到当前的时间  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");//设置时间显示格式  String str = sdf.format(date);//将当前时间格式化为需要的类型  insertParams.put("t\_number", task.getT\_number());  insertParams.put("course\_number", task.getClass\_number());  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("task\_text", task.getTask\_text());  insertParams.put("task\_title", task.getTask\_title());  insertParams.put("task\_type", task.getTask\_type());  insertParams.put("class\_number", task.getClass\_number());  insertParams.put("task\_end\_time", task.getTask\_end\_time());  insertParams.put("task\_publish\_time", str);  insertParams.put("task\_start\_time", str);  if(util.insert(tableName, insertParams)!=1){  allRight=false;  }      /\*依次添加进不同类型的题目表\*/  insertParams.clear();  String simpleJsonArray=request.getParameter("simpleJsonArray");  String multipleJsonArray=request.getParameter("multipleJsonArray");  String gapfillingJsonArray=request.getParameter("gapfillingJsonArray");  String judgeJsonArray=request.getParameter("judgeJsonArray");  String shortJsonArray=request.getParameter("shortJsonArray");  System.out.println("单选题："+simpleJsonArray);  System.out.println("多选题："+multipleJsonArray);  System.out.println("填空题："+gapfillingJsonArray);  System.out.println("判断题："+judgeJsonArray);  System.out.println("客观题："+shortJsonArray);  JsonUtilDao jsonUtil=new JsonUtilDao();  if(simpleJsonArray!=null&&!simpleJsonArray.equals("")&&allRight){  Choice\_question pojo=new Choice\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(simpleJsonArray, pojo);  tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "choice\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="choice\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Choice\_question) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("correct\_option", pojo.getCorrect\_option());  insertParams.put("choice\_options", pojo.getChoice\_options());  insertParams.put("choice\_number", pojo.getChoice\_number());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  if(multipleJsonArray!=null&&!multipleJsonArray.equals("")&&allRight){  insertParams.clear();  Multiple\_choice\_question pojo=new Multiple\_choice\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(multipleJsonArray, pojo);    tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "multiple\_choice\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="multiple\_choice\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Multiple\_choice\_question) arrayList.get(i);  System.out.println("task\_number:"+task.getTask\_number());  System.out.println("task\_number:"+pojo.getTask\_number());  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("correct\_options", pojo.getCorrect\_options());  insertParams.put("options", pojo.getOptions());  insertParams.put("multiple\_choice\_number", pojo.getMultiple\_choice\_number());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  if(gapfillingJsonArray!=null&&!gapfillingJsonArray.equals("")&&allRight){  insertParams.clear();  Blank\_question pojo=new Blank\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(gapfillingJsonArray, pojo);    tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "blank\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="blank\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Blank\_question) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("blank\_number", pojo.getBlank\_number());  insertParams.put("answer", pojo.getAnswer());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  if(judgeJsonArray!=null&&!judgeJsonArray.equals("")&&allRight){  insertParams.clear();  Judgment\_question pojo=new Judgment\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(judgeJsonArray, pojo);    tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "judgment\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="judgment\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Judgment\_question) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("judgment\_number", pojo.getJudgment\_number());  insertParams.put("answer", pojo.getAnswer());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  if(shortJsonArray!=null&&!shortJsonArray.equals("")&&allRight){  insertParams.clear();  Subjective\_question pojo=new Subjective\_question();  ArrayList<Object> arrayList=new ArrayList<Object>();  arrayList=jsonUtil.fromJsonToJavaArray(shortJsonArray, pojo);  pojo=(Subjective\_question) arrayList.get(0);    tableName="online\_task";  insertParams.put("task\_number", task.getTask\_number());  insertParams.put("question\_type", "subjective\_question");//类型名和表名一样  if(util.insert(tableName, insertParams)==1){  tableName="subjective\_question";  for(int i=0;i<arrayList.size();i++){  insertParams.clear();  pojo=(Subjective\_question) arrayList.get(i);  insertParams.put("task\_number", pojo.getTask\_number());  insertParams.put("text", pojo.getText());  insertParams.put("subjective\_number", pojo.getSubjective\_number());  insertParams.put("answer", pojo.getAnswer());  insertParams.put("score", pojo.getScore());  if(util.insert(tableName, insertParams)!=1){  allRight=false;  break;  }    }  }  else{  allRight=false;  }  }  String result;  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(allRight){  util.getConn().commit();  stringer.object().  key("result").value("true").  endObject();  }  else{  util.getConn().rollback();  stringer.object().  key("result").value("false").  endObject();  }  stringer.endArray();  result=stringer.toString();  return result;//为[{"result":"true"}]前端解析不需要newd=JSON.parse(data)  //如果是[{result:"true"}]前端解析需要newd=JSON.parse(data)  } |

### 4.3.4 数据持久层

MysqlUtil.java

### 4.3.5 域模型层

（1）Abstract\_task.java

|  |
| --- |
| package bean;  public class Abstract\_task {  private String task\_number;  private String task\_text;  private String task\_type;  private String task\_title;  private String task\_publish\_time;  private String task\_start\_time;  private String task\_end\_time;  private String t\_number;  private String course\_number;  private String class\_number;    public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public String getTask\_text() {  return task\_text;  }  public void setTask\_text(String task\_text) {  this.task\_text = task\_text;  }  public String getTask\_type() {  return task\_type;  }  public void setTask\_type(String task\_type) {  this.task\_type = task\_type;  }  public String getTask\_title() {  return task\_title;  }  public void setTask\_title(String task\_title) {  this.task\_title = task\_title;  }  public String getTask\_publish\_time() {  return task\_publish\_time;  }  public void setTask\_publish\_time(String task\_publish\_time) {  this.task\_publish\_time = task\_publish\_time;  }  public String getTask\_start\_time() {  return task\_start\_time;  }  public void setTask\_start\_time(String task\_start\_time) {  this.task\_start\_time = task\_start\_time;  }  public String getTask\_end\_time() {  return task\_end\_time;  }  public void setTask\_end\_time(String task\_end\_time) {  this.task\_end\_time = task\_end\_time;  }  public String getT\_number() {  return t\_number;  }  public void setT\_number(String t\_number) {  this.t\_number = t\_number;  }  public String getCourse\_number() {  return course\_number;  }  public void setCourse\_number(String course\_number) {  this.course\_number = course\_number;  }  public String getClass\_number() {  return class\_number;  }  public void setClass\_number(String class\_number) {  this.class\_number = class\_number;  }      } |

（2）File\_task.java

|  |
| --- |
| package bean;  public class File\_task {  private String task\_number;  private String file\_name;  private String file\_path;  private int download\_count;    public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public String getFile\_name() {  return file\_name;  }  public void setFile\_name(String file\_name) {  this.file\_name = file\_name;  }  public String getFile\_path() {  return file\_path;  }  public void setFile\_path(String file\_path) {  this.file\_path = file\_path;  }  public int getDownload\_count() {  return download\_count;  }  public void setDownload\_count(int download\_count) {  this.download\_count = download\_count;  }      } |

（3）Online\_task.java

|  |
| --- |
| package bean;  public class Online\_task {  private String task\_number;  private String question\_type;    public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public String getQuestion\_type() {  return question\_type;  }  public void setQuestion\_type(String question\_type) {  this.question\_type = question\_type;  }  } |

（4）Choice\_question.java

|  |
| --- |
| package bean;  public class Choice\_question {  private String task\_number;  private String text;  private String correct\_option;  private String choice\_options;  private int choice\_number;  private int score;  public Choice\_question() {    }  public Choice\_question(String task\_number, String text, String correct\_option, String choice\_options,  int choice\_number, int score) {  super();  this.task\_number = task\_number;  this.text = text;  this.correct\_option = correct\_option;  this.choice\_options = choice\_options;  this.choice\_number = choice\_number;  this.score = score;  }  public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public String getCorrect\_option() {  return correct\_option;  }  public void setCorrect\_option(String correct\_option) {  this.correct\_option = correct\_option;  }  public String getChoice\_options() {  return choice\_options;  }  public void setChoice\_options(String choice\_options) {  this.choice\_options = choice\_options;  }  public int getChoice\_number() {  return choice\_number;  }  public void setChoice\_number(int choice\_number) {  this.choice\_number = choice\_number;  }  public int getScore() {  return score;  }  public void setScore(int score) {  this.score = score;  }  public String getText() {  return text;  }  public void setText(String text) {  this.text = text;  }  @Override  public String toString() {  return "Choice\_question [task\_number=" + task\_number + ", text=" + text + ", correct\_option=" + correct\_option  + ", choice\_options=" + choice\_options + ", choice\_number=" + choice\_number + ", score=" + score + "]";  }      } |

（5）Multiple\_choice\_question.java

|  |
| --- |
| package bean;  public class Multiple\_choice\_question {  private String task\_number;  private String text;  private String correct\_options;  private String options;  private int multiple\_choice\_number;  private int score;  public Multiple\_choice\_question() {    }  public Multiple\_choice\_question(String task\_number, String text, String correct\_options, String options,  int multiple\_choice\_number, int score) {  super();  this.task\_number = task\_number;  this.text = text;  this.correct\_options = correct\_options;  this.options = options;  this.multiple\_choice\_number = multiple\_choice\_number;  this.score = score;  }  public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public String getCorrect\_options() {  return correct\_options;  }  public void setCorrect\_options(String correct\_options) {  this.correct\_options = correct\_options;  }  public String getOptions() {  return options;  }  public void setOptions(String options) {  this.options = options;  }  public int getMultiple\_choice\_number() {  return multiple\_choice\_number;  }  public void setMultiple\_choice\_number(int multiple\_choice\_number) {  this.multiple\_choice\_number = multiple\_choice\_number;  }  public int getScore() {  return score;  }  public void setScore(int score) {  this.score = score;  }  public String getText() {  return text;  }  public void setText(String text) {  this.text = text;  }  @Override  public String toString() {  return "Multiple\_choice\_question [task\_number=" + task\_number + ", text=" + text + ", correct\_options="  + correct\_options + ", options=" + options + ", multiple\_choice\_number=" + multiple\_choice\_number  + ", score=" + score + "]";  }      } |

（6）Blank\_question.java

|  |
| --- |
| package bean;  public class Blank\_question {  private String task\_number;  private String text;  private int blank\_number;  private String answer;  private int score;    public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public String getText() {  return text;  }  public void setText(String text) {  this.text = text;  }  public int getBlank\_number() {  return blank\_number;  }  public void setBlank\_number(int blank\_number) {  this.blank\_number = blank\_number;  }  public String getAnswer() {  return answer;  }  public void setAnswer(String answer) {  this.answer = answer;  }  public int getScore() {  return score;  }  public void setScore(int score) {  this.score = score;  }      } |

（7）Judgment\_question.java

|  |
| --- |
| package bean;  public class Judgment\_question {  private String task\_number;  private String text;  private int judgment\_number;  private String answer;  private int score;    public Judgment\_question() {    }    @Override  public String toString() {  return "Judgment\_question [task\_number=" + task\_number + ", text=" + text + ", judgment\_number="  + judgment\_number + ", answer=" + answer + ", score=" + score + "]";  }  public Judgment\_question(String task\_number, String text, int judgment\_number, String answer, int score) {  super();  this.task\_number = task\_number;  this.text = text;  this.judgment\_number = judgment\_number;  this.answer = answer;  this.score = score;  }  public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public String getText() {  return text;  }  public void setText(String text) {  this.text = text;  }  public int getJudgment\_number() {  return judgment\_number;  }  public void setJudgment\_number(int judgment\_number) {  this.judgment\_number = judgment\_number;  }  public String getAnswer() {  return answer;  }  public void setAnswer(String answer) {  this.answer = answer;  }  public int getScore() {  return score;  }  public void setScore(int score) {  this.score = score;  }      } |

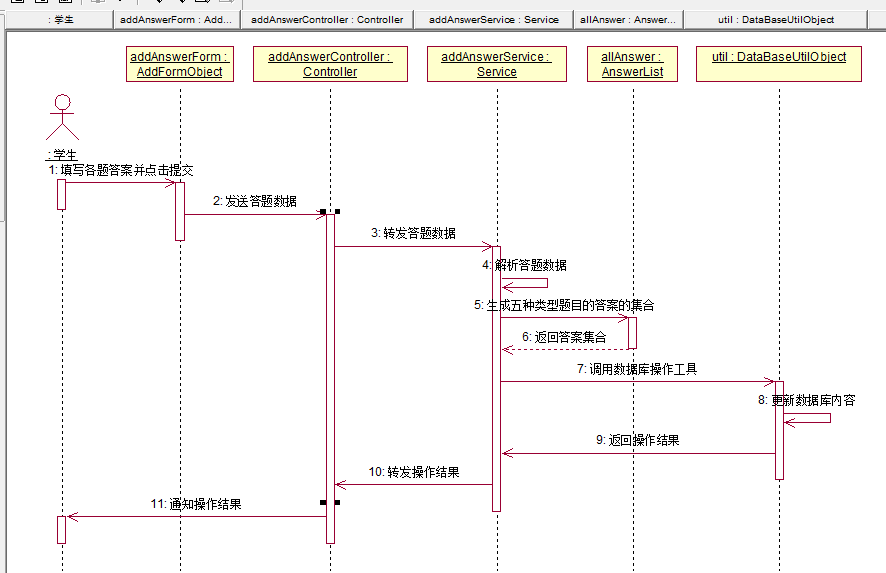
（8）Subjective\_question.java

|  |
| --- |
| package bean;  public class Subjective\_question {  private String task\_number;  private String text;  private int subjective\_number;  private String answer;  private int score;    public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public String getText() {  return text;  }  public void setText(String text) {  this.text = text;  }  public int getSubjective\_number() {  return subjective\_number;  }  public void setSubjective\_number(int subjective\_number) {  this.subjective\_number = subjective\_number;  }  public String getAnswer() {  return answer;  }  public void setAnswer(String answer) {  this.answer = answer;  }  public int getScore() {  return score;  }  public void setScore(int score) {  this.score = score;  }      } |

## 4.4、学生提交作业

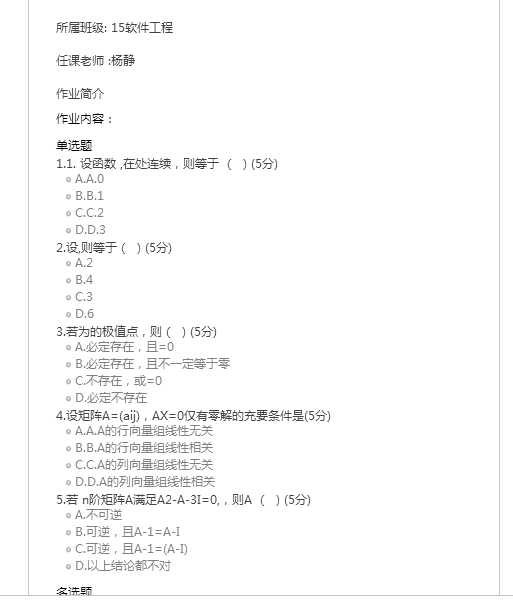
|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 发布作业页面 | studentDoOnlineTask.jsp  stuDoHomework.jsp | 在studentDoOnlineTask.jsp  提交在线作业；  在stuDoHomework.jsp提交线下型作业 |

图 4-4-1 学生提交作业序列图



### 4.4.1 表现层

（1）studentDoOnlineTask.jsp



（2）stuDoHomework.jsp



### 4.4.2 控制层

（1）提交在线型作业

|  |
| --- |
| @Override  @RequestMapping(value="/submit\_student\_online\_task\_answer",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String submitOnlineTaskAnswer(HttpServletRequest request, Online\_task onlineTask, Student student)  throws Exception {  return addService.submitOnlineTaskAnswer(request, onlineTask, student);  } |

（2）提交线下型作业

|  |
| --- |
| @Override  @RequestMapping(value="/submit\_student\_deline\_task\_answer",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String submitDelineTaskAnswer(HttpServletRequest request, File\_task fileTask, Student student)  throws Exception {  return null;  } |

### 4.4.3 业务逻辑层

（1）提交在线型作业AddFunctionService.java

|  |
| --- |
| @Override  public String submitOnlineTaskFinalScore(Score\_result scoreResult) {  JSONStringer stringer = new JSONStringer();//  stringer.array();  MysqlUtil util=new MysqlUtil();  String tableName="";  String result="";  Map<String,Object> insertParams=new HashMap<String,Object>();  try {  util.getConn().setAutoCommit(false);    int totalObjectiveScore=0;//记录所有客观题的分数和  Map<String,Object> conditionalParams=new HashMap<String,Object>();  ResultSet rs1=null;//rs1记录该在线作业的题型的表名  ResultSet rs2=null;//rs2记录某个题型的所有题目的答题  ResultSet rs3=null;//rs3记录某个题型的所有题目  if(scoreResult.getTask\_number()!=null&&!scoreResult.getTask\_number().equals("")){  if(scoreResult.getS\_number()!=null&&!scoreResult.getS\_number().equals("")){  if(scoreResult.getTotal\_score()!=null&&!scoreResult.getTotal\_score().equals("")){  /\*------------------------获取客观题总分S------------------\*/  System.out.println("主观题得分："+scoreResult.getTotal\_score());  tableName="online\_task";  conditionalParams.put("task\_number", scoreResult.getTask\_number());  rs1=util.select(tableName, null, conditionalParams);  while(rs1.next()){    /\*------------------找到正确答案，用于匹配客观题S---------------------\*/  tableName=rs1.getString("question\_type");  System.out.println("当前查找的题型："+tableName);  System.out.println("当前查询条件："+conditionalParams);  conditionalParams.clear();  conditionalParams.put("task\_number", scoreResult.getTask\_number());  rs3=util.select(tableName, null, conditionalParams);  /\*------------------找到正确答案，用于匹配客观题E---------------------\*/  /\*------------------找到学生客观题答题情况S---------------------\*/  int length=rs1.getString("question\_type").split("\_").length;  if(length==2)  tableName=rs1.getString("question\_type").split("\_")[0]+"\_answer";  if(length>2){  int originalLength=rs1.getString("question\_type").length();  int lastStringLength=rs1.getString("question\_type").split("\_")[length-1].length();  int needCutLength=lastStringLength;  tableName=rs1.getString("question\_type").substring(0,originalLength-needCutLength);  tableName=tableName+"answer";  }  conditionalParams.put("s\_number", scoreResult.getS\_number());  System.out.println("当前查找的题型："+tableName);  rs2=util.select(tableName, null, conditionalParams);  while(rs2.next()&&rs3.next()){  switch(rs1.getString("question\_type")){  case "choice\_question":  if(rs2.getString("student\_choice\_option").equals(rs3.getString("correct\_option"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("单选题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;  case "multiple\_choice\_question":  if(rs2.getString("student\_multiple\_choice\_options").equals(rs3.getString("correct\_options"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("多选题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;  /\*case "blank\_question":  if(rs2.getString("student\_answer\_text").equals(rs3.getString("answer"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("填空题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;\*/  case "judgment\_question":  if(rs2.getString("student\_answer\_text").equals(rs3.getString("answer"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("判断题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;  /\*case "subjective\_question":  if(rs2.getString("student\_answer\_text").equals(rs3.getString("answer"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("简答题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;\*/  }      }  /\*------------------找到学生客观题答题情况E---------------------\*/  System.out.println("客观题总得分： "+totalObjectiveScore);  }  /\*------------------------获取客观题总分E------------------\*/  tableName="score\_result";  if(scoreResult.getResult\_comment()!=null&&!scoreResult.getResult\_comment().equals("")){  insertParams.put("task\_number", scoreResult.getTask\_number());  insertParams.put("s\_number", scoreResult.getS\_number());  int totalScore=Integer.parseInt(scoreResult.getTotal\_score())+totalObjectiveScore;  insertParams.put("total\_score", totalScore+"");  insertParams.put("result\_comment", scoreResult.getResult\_comment());  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  else{  insertParams.put("task\_number", scoreResult.getTask\_number());  insertParams.put("s\_number", scoreResult.getS\_number());  int totalScore=Integer.parseInt(scoreResult.getTotal\_score())+totalObjectiveScore;  insertParams.put("total\_score", totalScore+"");  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  }  }  }    } catch (SQLException e) {  e.printStackTrace();  }  stringer.endArray();  result=stringer.toString();  return result;//为[{"result":"true"}]前端解析不需要newd=JSON.parse(data)  //如果是[{result:"true"}]前端解析需要newd=JSON.parse(data)  } |

### 4.4.4 数据持久层

MysqlUtil.java

### 4.4.5 域模型层

（1）Choice\_answer.java

|  |
| --- |
| package bean;  public class Choice\_answer {  private String task\_number;  private int choice\_number;  private String s\_number;  private String student\_choice\_option;    public Choice\_answer() {    }    public Choice\_answer(String task\_number, int choice\_number, String s\_number, String student\_choice\_option) {  super();  this.task\_number = task\_number;  this.choice\_number = choice\_number;  this.s\_number = s\_number;  this.student\_choice\_option = student\_choice\_option;  }  public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public int getChoice\_number() {  return choice\_number;  }  public void setChoice\_number(int choice\_number) {  this.choice\_number = choice\_number;  }  public String getS\_number() {  return s\_number;  }  public void setS\_number(String s\_number) {  this.s\_number = s\_number;  }  public String getStudent\_choice\_option() {  return student\_choice\_option;  }  public void setStudent\_choice\_option(String student\_choice\_option) {  this.student\_choice\_option = student\_choice\_option;  }  @Override  public String toString() {  return "Choice\_answer [task\_number=" + task\_number + ", choice\_number=" + choice\_number + ", s\_number="  + s\_number + ", student\_choice\_option=" + student\_choice\_option + "]";  }      } |

（2）Multiple\_choice\_answer.java

|  |
| --- |
| package bean;  public class Multiple\_choice\_answer {  private String task\_number;  private int multiple\_choice\_number;  private String s\_number;  private String student\_multiple\_choice\_options;  public Multiple\_choice\_answer() {}  public Multiple\_choice\_answer(String task\_number, int multiple\_choice\_number, String s\_number,  String student\_multiple\_choice\_options) {  super();  this.task\_number = task\_number;  this.multiple\_choice\_number = multiple\_choice\_number;  this.s\_number = s\_number;  this.student\_multiple\_choice\_options = student\_multiple\_choice\_options;  }  public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public int getMultiple\_choice\_number() {  return multiple\_choice\_number;  }  public void setMultiple\_choice\_number(int multiple\_choice\_number) {  this.multiple\_choice\_number = multiple\_choice\_number;  }  public String getS\_number() {  return s\_number;  }  public void setS\_number(String s\_number) {  this.s\_number = s\_number;  }  public String getStudent\_multiple\_choice\_options() {  return student\_multiple\_choice\_options;  }  public void setStudent\_multiple\_choice\_options(  String student\_multiple\_choice\_options) {  this.student\_multiple\_choice\_options = student\_multiple\_choice\_options;  }  @Override  public String toString() {  return "Multiple\_choice\_answer [task\_number=" + task\_number + ", multiple\_choice\_number="  + multiple\_choice\_number + ", s\_number=" + s\_number + ", student\_multiple\_choice\_options="  + student\_multiple\_choice\_options + "]";  }      } |

（3）Judgment\_answer.java

|  |
| --- |
| package bean;  public class Judgment\_answer {  private String task\_number;  private int judgment\_number;  private String s\_number;  private String student\_answer\_text;  public Judgment\_answer() {    }  public Judgment\_answer(String task\_number, int judgment\_number, String s\_number, String student\_answer\_text) {  super();  this.task\_number = task\_number;  this.judgment\_number = judgment\_number;  this.s\_number = s\_number;  this.student\_answer\_text = student\_answer\_text;  }  public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public int getJudgment\_number() {  return judgment\_number;  }  public void setJudgment\_number(int judgment\_number) {  this.judgment\_number = judgment\_number;  }  public String getS\_number() {  return s\_number;  }  public void setS\_number(String s\_number) {  this.s\_number = s\_number;  }  public String getStudent\_answer\_text() {  return student\_answer\_text;  }  public void setStudent\_answer\_text(String student\_answer\_text) {  this.student\_answer\_text = student\_answer\_text;  }  @Override  public String toString() {  return "Judgment\_answer [task\_number=" + task\_number + ", judgment\_number=" + judgment\_number + ", s\_number="  + s\_number + ", student\_answer\_text=" + student\_answer\_text + "]";  }      } |

（4）Blank\_answer.java

|  |
| --- |
| package bean;  public class Blank\_answer {  private String task\_number;  private int blank\_number;  private String s\_number;  private String student\_answer\_text;    public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public int getBlank\_number() {  return blank\_number;  }  public void setBlank\_number(int blank\_number) {  this.blank\_number = blank\_number;  }  public String getS\_number() {  return s\_number;  }  public void setS\_number(String s\_number) {  this.s\_number = s\_number;  }  public String getStudent\_answer\_text() {  return student\_answer\_text;  }  public void setStudent\_answer\_text(String student\_answer\_text) {  this.student\_answer\_text = student\_answer\_text;  }        } |

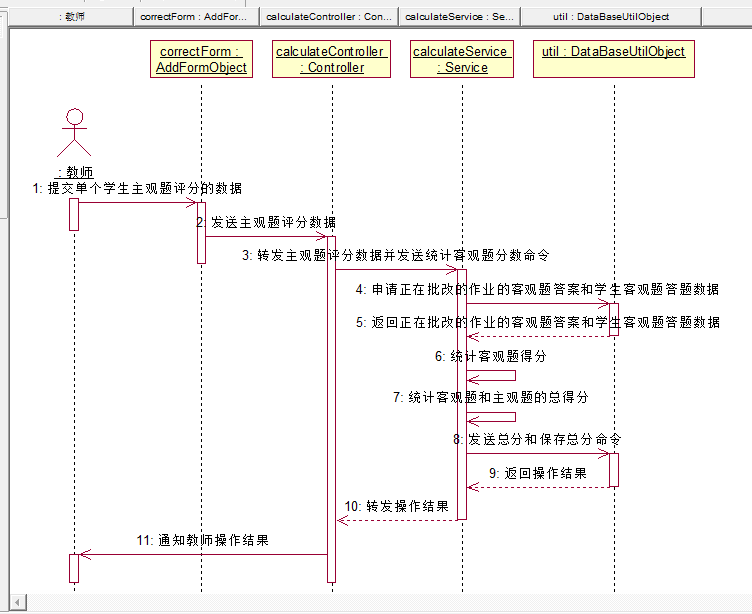
（5）Subjective\_answer.java

|  |
| --- |
| package bean;  public class Subjective\_answer {  private String task\_number;  private int subjective\_number;  private String s\_number;  private String student\_answer\_text;    public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public int getSubjective\_number() {  return subjective\_number;  }  public void setSubjective\_number(int subjective\_number) {  this.subjective\_number = subjective\_number;  }  public String getS\_number() {  return s\_number;  }  public void setS\_number(String s\_number) {  this.s\_number = s\_number;  }  public String getStudent\_answer\_text() {  return student\_answer\_text;  }  public void setStudent\_answer\_text(String student\_answer\_text) {  this.student\_answer\_text = student\_answer\_text;  }      } |

## 4.5老师修改和查看作业情况

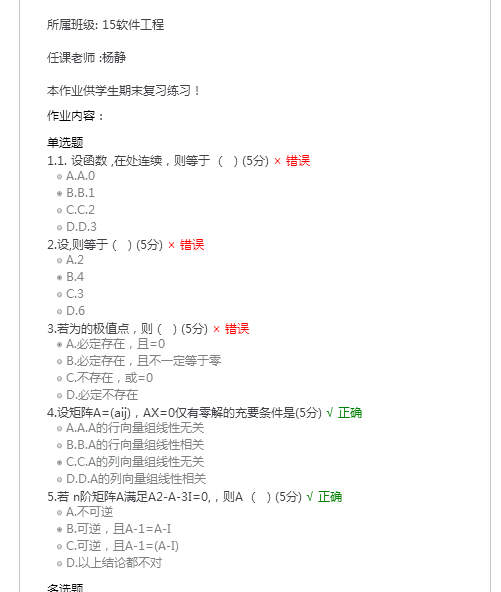
|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 在线批改作业 | onlineCorrect.jsp | 教师在此批改学生在线作业，客观题由系统自动批改，主观题由教师输入每题得分和最后的评语，最终点击“完成批改”，系统将自动计算并评分 |

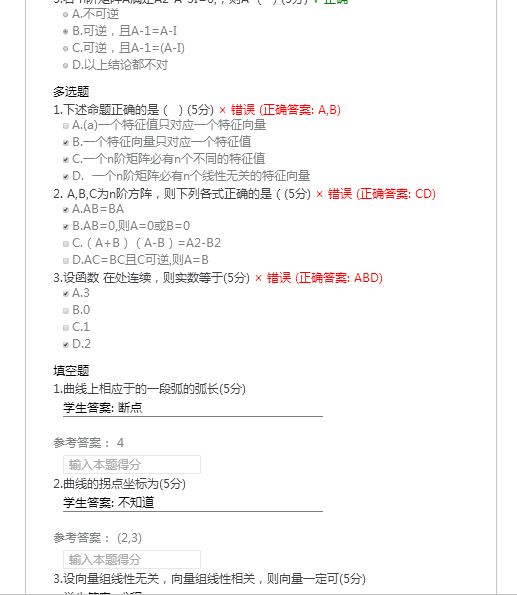
图 4-5-1 教师在线批改在线作业序列图

****

### 4.5.1 表现层

（1）在线型作业的批改







### 4.5.2 控制层

AddFunctionController.java

|  |
| --- |
| @Override  @RequestMapping(value="/submit\_online\_task\_score",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String submitOnlineTaskFinalScore(Score\_result scoreResult) {  return addService.submitOnlineTaskFinalScore(scoreResult);  } |

### 4.5.3 业务逻辑层

AddFunctionService.java

|  |
| --- |
| @Override  public String submitOnlineTaskFinalScore(Score\_result scoreResult) {  JSONStringer stringer = new JSONStringer();//  stringer.array();  MysqlUtil util=new MysqlUtil();  String tableName="";  String result="";  Map<String,Object> insertParams=new HashMap<String,Object>();  try {  util.getConn().setAutoCommit(false);    int totalObjectiveScore=0;//记录所有客观题的分数和  Map<String,Object> conditionalParams=new HashMap<String,Object>();  ResultSet rs1=null;//rs1记录该在线作业的题型的表名  ResultSet rs2=null;//rs2记录某个题型的所有题目的答题  ResultSet rs3=null;//rs3记录某个题型的所有题目  if(scoreResult.getTask\_number()!=null&&!scoreResult.getTask\_number().equals("")){  if(scoreResult.getS\_number()!=null&&!scoreResult.getS\_number().equals("")){  if(scoreResult.getTotal\_score()!=null&&!scoreResult.getTotal\_score().equals("")){  /\*------------------------获取客观题总分S------------------\*/  System.out.println("主观题得分："+scoreResult.getTotal\_score());  tableName="online\_task";  conditionalParams.put("task\_number", scoreResult.getTask\_number());  rs1=util.select(tableName, null, conditionalParams);  while(rs1.next()){    /\*------------------找到正确答案，用于匹配客观题S---------------------\*/  tableName=rs1.getString("question\_type");  System.out.println("当前查找的题型："+tableName);  System.out.println("当前查询条件："+conditionalParams);  conditionalParams.clear();  conditionalParams.put("task\_number", scoreResult.getTask\_number());  rs3=util.select(tableName, null, conditionalParams);  /\*------------------找到正确答案，用于匹配客观题E---------------------\*/  /\*------------------找到学生客观题答题情况S---------------------\*/  int length=rs1.getString("question\_type").split("\_").length;  if(length==2)  tableName=rs1.getString("question\_type").split("\_")[0]+"\_answer";  if(length>2){  int originalLength=rs1.getString("question\_type").length();  int lastStringLength=rs1.getString("question\_type").split("\_")[length-1].length();  int needCutLength=lastStringLength;  tableName=rs1.getString("question\_type").substring(0,originalLength-needCutLength);  tableName=tableName+"answer";  }  conditionalParams.put("s\_number", scoreResult.getS\_number());  System.out.println("当前查找的题型："+tableName);  rs2=util.select(tableName, null, conditionalParams);  while(rs2.next()&&rs3.next()){  switch(rs1.getString("question\_type")){  case "choice\_question":  if(rs2.getString("student\_choice\_option").equals(rs3.getString("correct\_option"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("单选题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;  case "multiple\_choice\_question":  if(rs2.getString("student\_multiple\_choice\_options").equals(rs3.getString("correct\_options"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("多选题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;  /\*case "blank\_question":  if(rs2.getString("student\_answer\_text").equals(rs3.getString("answer"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("填空题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;\*/  case "judgment\_question":  if(rs2.getString("student\_answer\_text").equals(rs3.getString("answer"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("判断题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;  /\*case "subjective\_question":  if(rs2.getString("student\_answer\_text").equals(rs3.getString("answer"))){  int score=Integer.parseInt(rs3.getString("score"));  System.out.println("简答题"+rs3.getString("text")+" 本题得分："+score);  totalObjectiveScore=totalObjectiveScore+score;  }  break;\*/  }      }  /\*------------------找到学生客观题答题情况E---------------------\*/  System.out.println("客观题总得分： "+totalObjectiveScore);  }  /\*------------------------获取客观题总分E------------------\*/  tableName="score\_result";  if(scoreResult.getResult\_comment()!=null&&!scoreResult.getResult\_comment().equals("")){  insertParams.put("task\_number", scoreResult.getTask\_number());  insertParams.put("s\_number", scoreResult.getS\_number());  int totalScore=Integer.parseInt(scoreResult.getTotal\_score())+totalObjectiveScore;  insertParams.put("total\_score", totalScore+"");  insertParams.put("result\_comment", scoreResult.getResult\_comment());  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  else{  insertParams.put("task\_number", scoreResult.getTask\_number());  insertParams.put("s\_number", scoreResult.getS\_number());  int totalScore=Integer.parseInt(scoreResult.getTotal\_score())+totalObjectiveScore;  insertParams.put("total\_score", totalScore+"");  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  }  }  }    } catch (SQLException e) {  e.printStackTrace();  }  stringer.endArray();  result=stringer.toString();  return result;//为[{"result":"true"}]前端解析不需要newd=JSON.parse(data)  //如果是[{result:"true"}]前端解析需要newd=JSON.parse(data)  } |

### 4.5.4 数据持久层

MysqlUtil.java

### 4.5.5 域模型层

Score\_result.java

|  |
| --- |
| package bean;  public class Score\_result {    private String task\_number;  private String s\_number;  private String total\_score;  private String result\_comment;    public String getTask\_number() {  return task\_number;  }  public void setTask\_number(String task\_number) {  this.task\_number = task\_number;  }  public String getS\_number() {  return s\_number;  }  public void setS\_number(String s\_number) {  this.s\_number = s\_number;  }  public String getTotal\_score() {  return total\_score;  }  public void setTotal\_score(String total\_score) {  this.total\_score = total\_score;  }  public String getResult\_comment() {  return result\_comment;  }  public void setResult\_comment(String result\_comment) {  this.result\_comment = result\_comment;  }        } |

# 5、管理端模块详细设计

## 5.1、后台新增数据模块详细设计

### 5.1.1 表现层



### 5.1.2 控制层

|  |
| --- |
| @Override  @RequestMapping(value="/add\_student",method=RequestMethod.POST)  @ResponseBody  public String addStudent(Student student,Student\_class student\_class) throws Exception {  return addService.addStudent(student, student\_class);  }  @Override  @RequestMapping(value="/add\_teacher",method=RequestMethod.POST)  @ResponseBody  public String addTeacher(Teacher teacher,College college) throws Exception{  return addService.addTeacher(teacher, college);  }  @Override  @RequestMapping(value="/add\_college",method=RequestMethod.POST)  @ResponseBody  public String addCollege(College college) throws Exception{  return addService.addCollege(college);  }  @Override  @RequestMapping(value="/add\_class",method=RequestMethod.POST)  @ResponseBody  public String addClass(Student\_class studentClass,College college) throws Exception{  return addService.addClass(studentClass, college);  }  @Override  @RequestMapping(value="/add\_course",method=RequestMethod.POST)  @ResponseBody    public String addCourse(Course course) throws Exception{  return addService.addCourse(course);  }  @Override  @RequestMapping(value="/connect\_course\_and\_student",method=RequestMethod.POST,produces="text/json;charset=UTF-8")  @ResponseBody  public String connect\_course\_and\_student(Course course, Student student) {  return addService.connect\_course\_and\_student(course, student);  } |

### 5.1.3 业务逻辑层

（1）新增学生

|  |
| --- |
| @Transactional(rollbackFor=Exception.class)//将事务响应回滚的异常的类型声明为Exception，因为spring默认为RuntimeException  //@Transactional  @Override  public String addStudent(Student student, Student\_class student\_class) throws Exception {    System.out.println(student.getS\_number()+" "+student.getS\_name()+" "+student.getPassword());  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="student";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("s\_number", student.getS\_number());  insertParams.put("s\_name", student.getS\_name());  insertParams.put("password", student.getPassword());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  tableName="class\_relationship\_student";  insertParams.clear();  insertParams.put("s\_number", student.getS\_number());  insertParams.put("class\_number", student\_class.getClass\_number());  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();  //依旧返回数据，也抛出异常  stringer.object().  key("result").value("false").  endObject();  /\*Exception e=new Exception();  throw e;\*/  util.getConn().rollback();    }  }  else{  //TransactionAspectSupport.currentTransactionStatus().setRollbackOnly();  //依旧返回数据，也抛出异常  stringer.object().  key("result").value("false").  endObject();    }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  System.out.println(result);  return result;    } |

（2）新增教师

|  |
| --- |
| @Override  @Transactional(rollbackFor=Exception.class)  public String addTeacher(Teacher teacher, College college) throws Exception {  System.out.println(teacher.getT\_number()+" "+teacher.getT\_name()+" "+teacher.getPassword());  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="teacher";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("t\_number", teacher.getT\_number());  insertParams.put("t\_name", teacher.getT\_name());  insertParams.put("password", teacher.getPassword());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  insertParams.clear();  insertParams.put("t\_number", teacher.getT\_number());  insertParams.put("college\_number", college.getCollege\_number());  tableName="college\_relationship\_teacher";  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  response.setContentType("text/json; charset=UTF-8");  System.out.println(result);  return result;  } |

（3）新增学院

|  |
| --- |
| @Override  @Transactional(rollbackFor=Exception.class)  public String addCollege(College college) throws Exception{  System.out.println(college.getCollege\_name()+" "+college.getCollege\_number());  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="college";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("college\_number", college.getCollege\_number());  insertParams.put("college\_name", college.getCollege\_name());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  response.setContentType("text/json; charset=UTF-8");  System.out.println(result);  return result;  } |

（4）新增班级

|  |
| --- |
| @Override  @Transactional(rollbackFor=Exception.class)  public String addClass(Student\_class studentClass, College college) throws Exception{  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="student\_class";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("class\_number", studentClass.getClass\_number());  insertParams.put("class\_name", studentClass.getClass\_name());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  tableName="college\_relationship\_class";  insertParams.clear();  insertParams.put("college\_number", college.getCollege\_number());  insertParams.put("class\_number", studentClass.getClass\_number());  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  response.setContentType("text/json; charset=UTF-8");  System.out.println(result);  return result;  } |

（5）新增课程

|  |
| --- |
| @Override  @Transactional(rollbackFor=Exception.class)  public String addCourse(Course course) throws Exception{  /\*调用自定义软件接口新增数据S\*/  MysqlUtil util=new MysqlUtil();  util.getConn().setAutoCommit(false);  String tableName="course";  Map<String,Object> insertParams=new HashMap<String,Object>();  insertParams.put("course\_number", course.getCourse\_number());  insertParams.put("course\_name", course.getCourse\_name());  insertParams.put("t\_number", course.getT\_number());  /\*调用自定义软件接口新增数据E\*/  String result = "";  JSONStringer stringer = new JSONStringer();//  stringer.array();  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  }  else{  stringer.object().  key("result").value("false").  endObject();  }  util.getConn().commit();  stringer.endArray();  result=stringer.toString();  response.setContentType("text/json; charset=UTF-8");  System.out.println(result);  return result;  } |

（6）绑定学生与课程关系

|  |
| --- |
| @Override  public String connect\_course\_and\_student(Course course, Student student) {  String result="";  MysqlUtil util=new MysqlUtil();  JSONStringer stringer = new JSONStringer();//  Map<String,Object> insertParams=new HashMap<String,Object>();  stringer.array();  try {  util.getConn().setAutoCommit(false);  if(course.getCourse\_number()!=null&&course.getCourse\_number().equals("")){  if(student.getS\_number()!=null&&student.getS\_number().equals("")){  insertParams.put("course\_number", course.getCourse\_number());  insertParams.put("s\_number", student.getS\_number());  String tableName="course\_relationship\_student";  if(util.insert(tableName, insertParams)==1){  stringer.object().  key("result").value("true").  endObject();  util.getConn().commit();  }  else{  stringer.object().  key("result").value("false").  endObject();  util.getConn().rollback();  }  }  }  } catch (SQLException e) {  e.printStackTrace();  }        stringer.endArray();  result=stringer.toString();  return result;  }  } |

### 5.1.4 数据持久层

MysqlUtil.java

### 5.1.5 域模型层

（1）Student.java

|  |
| --- |
| package bean;  public class Student {  private String s\_number;  private String password;  private String s\_name;  public Student() {    }  public Student(String s\_number, String password, String s\_name) {  super();  this.s\_number = s\_number;  this.password = password;  this.s\_name = s\_name;  }  public String getS\_number() {  return s\_number;  }  public void setS\_number(String s\_number) {  this.s\_number = s\_number;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getS\_name() {  return s\_name;  }  public void setS\_name(String s\_name) {  this.s\_name = s\_name;  }  @Override  public String toString() {  return "Student [s\_number=" + s\_number + ", password=" + password + ", s\_name=" + s\_name + "]";  }    } |

（2）Teacher.java

|  |
| --- |
| package bean;  public class Teacher {  private String t\_number;  private String password;  private String t\_name;  public String getT\_number() {  return t\_number;  }  public void setT\_number(String t\_number) {  this.t\_number = t\_number;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public String getT\_name() {  return t\_name;  }  public void setT\_name(String t\_name) {  this.t\_name = t\_name;  }  } |

（3）College.java

|  |
| --- |
| package bean;  public class College {  public String college\_number ;  public String college\_name ;    public String getCollege\_number() {  return college\_number;  }  public void setCollege\_number(String college\_number) {  this.college\_number = college\_number;  }  public String getCollege\_name() {  return college\_name;  }  public void setCollege\_name(String college\_name) {  this.college\_name = college\_name;  }      } |

（4）Student\_class.java

|  |
| --- |
| package bean;  public class Student\_class {  public String class\_number;  public String class\_name;  public String getClass\_number() {  return class\_number;  }  public void setClass\_number(String class\_number) {  this.class\_number = class\_number;  }  public String getClass\_name() {  return class\_name;  }  public void setClass\_name(String class\_name) {  this.class\_name = class\_name;  }  } |

（5）Course.java

|  |
| --- |
| package bean;  public class Course {  public String course\_number;  public String course\_name;  public String t\_number;  public String getCourse\_number() {  return course\_number;  }  public void setCourse\_number(String course\_number) {  this.course\_number = course\_number;  }  public String getCourse\_name() {  return course\_name;  }  public void setCourse\_name(String course\_name) {  this.course\_name = course\_name;  }  public String getT\_number() {  return t\_number;  }  public void setT\_number(String t\_number) {  this.t\_number = t\_number;  }      } |

# 6、公共部分模块详细设计

## 6.1 公共页面

登录界面

## 6.2 安全模块的详细设计

采用MD5加密