**Project Start Report**

**项目立项报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prepared by**  **制定** | ***周凌羽*** | **Date**  **日期** | ***2020.5.15*** |
| **Reviewed by**  **评审** | ***黎卓武*** | **Date**  **日期** | ***2020.5.17*** |
| **Approved by**  **批准** | ***罗明岱*** | **Date**  **日期** | ***2020.5.18*** |

# 导言

## 目的

该文档对学生作业管理系统概要设计进行相信描述，对系统的模块进行了详细设计，包括静态页面、系统登录、作业管理、记录查询、通知栏、作业反馈、基本信息等功能模块的详细设计

本文档的预期读者为：

* 用户
* 项目管理人员
* 系统开发人员
* 软件测试人员。

## 范围

该文档涵盖了系统的功能模块，主要模型进行了详细描述。

## 引用标准

[1] 《软件工程案例教程 第 3 版》 机械工业出版社

[2] 《软件工程——实践者的研究方法》Roger S.Pressman[美], 郑人杰, 马素霞, 白晓颖译, 机械工业出版社

[3] 《需求分析与系统设计》麦斯阿塞克, 机械工业出版社

## 参考资料

[1] 《项目描述 160930》软件需求工程&软件工程管理课程组

## 版本更新信息

本文档的更新记录如下表 1-1 所示。

表 1-1 更新记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改日期 | 修改后版本 | 修改位置 | 修改内容概述 | 修改日期 |
|  |  |  |  |  |

# 详细设计简述

## 设计简介

概要设计中将项目进行了详细设计，包括静态页面、 系统登录、选课管理，成绩查询、 网上测试、公告栏、 进度管理、 基本信息等功能模块的详细设计。

项目在设计思路上将客户端划分为了五个模块，分别是教师模块、学生模块、管理员模块、助教模块、公共功能模块。具体划分如图1-1所示。

## 模块简介

**客户端 IPO 图：**

IPO 图：该系统客户端是用户与系统进行交互的窗口，涉及诸多的数据输入与输出。因此，做出以下IPO 图以表示用户数据输入与系统数据输出情况。

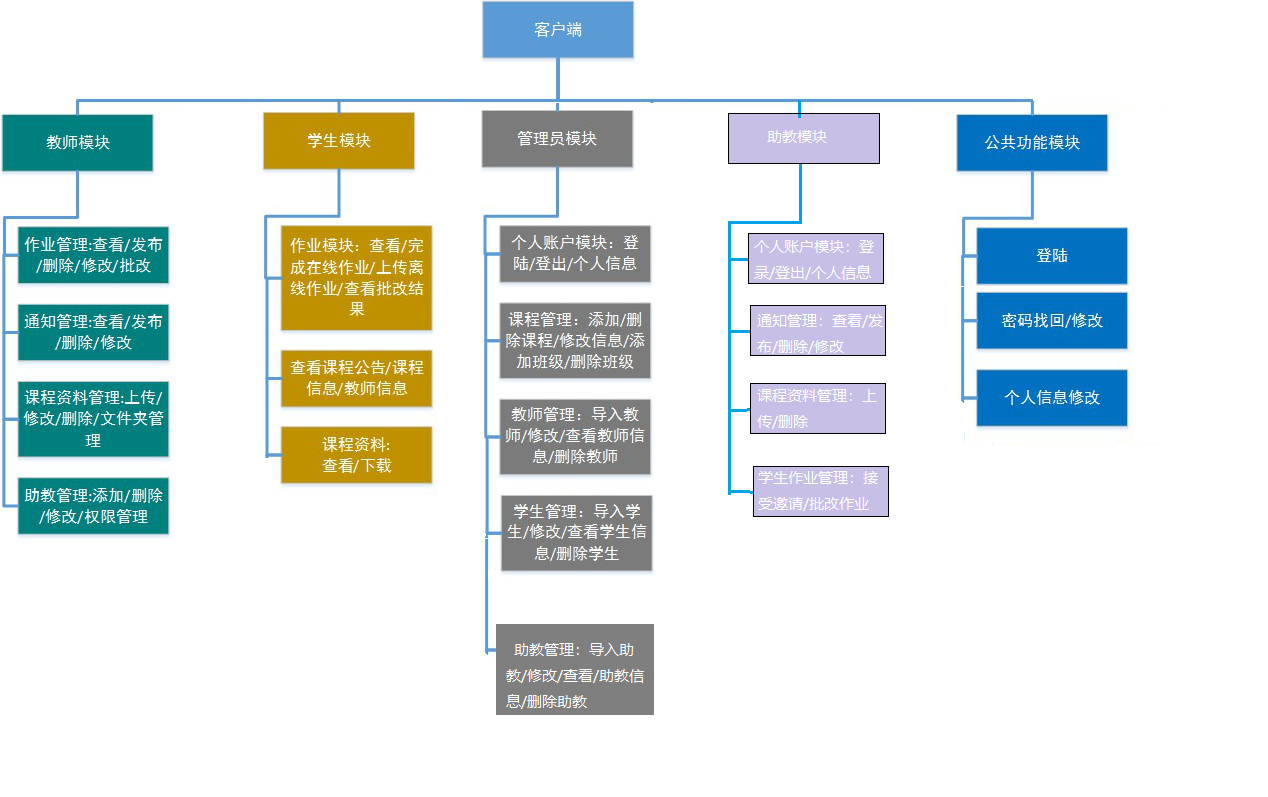


图 1-1 客户端模块划分-客户端IPO图

1. 用户进入系统首页后，系统即显示登录界面，用户登录后系统可判断

用户类型进而提供不同的服务操作。

2、 用户第一次进入系统之前， 若没有从其他平台导入账号，需要通过管理员添加新用户。

3、 教师模块为教师提供了作业查看、发布、删除、修改、批阅等功能。

4、 学生模块为学生提供了教师发布作业的通知，可以查看、完成在线作业，在完成作业后，学生应将作业提交上传，待教师批阅完成，可查看批阅结果。

5、 管理员模块提供了一系列对用户个人账户的添加、删除、修改等功能。也提供了班级设置相关操作。对于教师、学生以及助教的账号管理，可以导入、修改、删除。

6、 除了账户的登录登出等功能外、还有课程资料的管理，包括上传、删除等；也可以接受作业批改的邀请。

7、 对于所有的用户都提供了登录登出、信息修改、密码找回等公共模块。

8、 线上作业的发布，主要是是以老师老布，通告，学生接收并完成的形式，作业完成的文档支持本地上传提交。批阅完成后，学生能看见自己的得分。老师能够录入试题并且对试题进行修改。

9、 公告栏里主要是用来显示各种通知消息。 仅管理员能修改公告栏里  
面的内容。学生和老师都能查看公告栏里的内容，老师和管理员能够查看，  
修改，增加和删除公告栏内容。

# 管理员端详细设计

## 表现层

管理员端的模块表现层，主要完成个人账户的功能、用户管理（包含教师、学生、助教）以及课程管理。

表 3-1 用户管理模块表现层 JSP 列表

|  |  |  |
| --- | --- | --- |
| 界面 | jsp | 功能描述 |
| 管理教师账户 | m\_teachermanager.jsp、  m\_teacherupdate.jsp | 导入教师/修改查看/教师信息/删除教师 |
| 管理学生账户 | m\_studentmanager.jsp、  m\_studentupdate.jsp | 导入学生/修改查看/学生信息/删除学生 |
| 管理助教账户 | m\_assistantmanager.jsp、  m\_assistantupdate.jsp | 导入助教/修改查看/助教信息/删除助教 |
| 管理学生课程 | m\_studentcoursemanager.jsp、  m\_studentcourseupdate.jsp | 添加删除学生课程/修改查看/课程信息/删除班级 |
| 管理个人账户 | m\_personalsetting.jsp | 登入/登出/个人信息 |

## 控制层

管理员端的控制层，主要是控制页面的跳转与将从业务逻辑层获取到的数据传送到表现层。

表 3-2 用户管理模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 登录 | loginservlet.java | SUCCESS | 当查询、添加等操作成功时，显示操作成功。 |
| ERROR | 当查询、添加等操作失败时，显示操作失败。 |
| 登出 | logoutservlet.java | SUCCESS | 当添加操作成功时，显示操作成功 |
| EORROR | 当添加操作成失败时，显示操作失败 |
| 管理用户 | managerservlet.java | SUCCESS | 当查询、添加、修改等操作成功时，显示操作成功。 |
| EORROR | 当查询、添加、修改等操作失败时，显示操作失败。 |

在控制层中 loginservlet.java 的主要属性与方法如下所示：

package hwms.servlet;

/\*\*

 \* Servlet implementation class loginServlet

 \*/

public class loginServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    /\*\*

     \* @see HttpServlet#HttpServlet()

     \*/

    public loginServlet();

    protected void doGet(HttpServletRequest req, HttpServletResponse resp) ;

    /\*\*

     \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse

     \*      response)

     \*/

    protected void doPost(HttpServletRequest request, HttpServletResponse response);

}

在控制层中 logoutservlet.java 的主要属性与方法如下所示：

package hwms.servlet;

/\*\*

 \* Servlet implementation class logoutServlet

 \*/

@WebServlet("/logoutServlet")

public class logoutServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    /\*\*

     \* @see HttpServlet#HttpServlet()

     \*/

    public logoutServlet();

    protected void doGet(HttpServletRequest request, HttpServletResponse response);

    protected void doPost(HttpServletRequest request, HttpServletResponse response);

}

在控制层中 managerservlet.java的主要属性与方法如下所示：

package hwms.servlet;

public class manageServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response);

    //获取全部教师

    protected void allT(HttpServletRequest request, HttpServletResponse response);

    //获取全部助教

    protected void allA(HttpServletRequest request, HttpServletResponse response);

    //获取全部学生

    protected void allS(HttpServletRequest request, HttpServletResponse response);

    //获取全部课程

    protected void allC(HttpServletRequest request, HttpServletResponse response);

    //获取全部学生选课列表

    protected void allSC(HttpServletRequest request, HttpServletResponse response);

    // 获取全部邀请列表

    protected void allI(HttpServletRequest request, HttpServletResponse response);

    //添加教师

    protected void addT(HttpServletRequest request, HttpServletResponse response);

    //添加助教

    protected void addA(HttpServletRequest request, HttpServletResponse response);

    //添加学生

    protected void addS(HttpServletRequest request, HttpServletResponse response);

    //添加课程

    protected void addC(HttpServletRequest request, HttpServletResponse response);

    // 添加学生选课

    protected void addSC(HttpServletRequest request, HttpServletResponse response);

    // 添加邀请

    protected void addI(HttpServletRequest request, HttpServletResponse response);

    // 导入教师列表

    protected void addTeacherExcel(HttpServletRequest request, HttpServletResponse response);

    // 导入助教列表

    protected void addAssistantExcel(HttpServletRequest request, HttpServletResponse response);

    // 导入学生列表

    protected void addStudentExcel(HttpServletRequest request, HttpServletResponse response);

    // 导入课程列表

    protected void addCourseExcel(HttpServletRequest request, HttpServletResponse response);

    // 导入课程列表

    protected void addStudentCourseExcel(HttpServletRequest request, HttpServletResponse response);

    //删除教师

    protected void delT(HttpServletRequest request, HttpServletResponse response);

    // 删除助教

    protected void delA(HttpServletRequest request, HttpServletResponse response);

    // 删除学生

    protected void delS(HttpServletRequest request, HttpServletResponse response);

    // 删除课程

    protected void delC(HttpServletRequest request, HttpServletResponse response) ;

    // 删除学生选课

    protected void delSC(HttpServletRequest request, HttpServletResponse response);

    // 删除邀请

    protected void delI(HttpServletRequest request, HttpServletResponse response);

    // 获取修改教师信息

    protected void getUpdateT(HttpServletRequest request, HttpServletResponse response);

    // 获取修改助教信息

    protected void getUpdateA(HttpServletRequest request, HttpServletResponse response);

    // 获取修改学生信息

    protected void getUpdateS(HttpServletRequest request, HttpServletResponse response);

    // 获取修改课程信息

    protected void getUpdateC(HttpServletRequest request, HttpServletResponse response);

    // 修改教师信息

    protected void updateT(HttpServletRequest request, HttpServletResponse response);

    // 修改助教信息

    protected void updateA(HttpServletRequest request, HttpServletResponse response);

    // 修改学生信息

    protected void updateS(HttpServletRequest request, HttpServletResponse response);

    // 修改课程信息

    protected void updateC(HttpServletRequest request, HttpServletResponse response);

    //导入教师表

    protected void addTeacherList(HttpServletRequest request, HttpServletResponse response);

    //导入助教表

    protected void addAssistantList(HttpServletRequest request, HttpServletResponse response);

    //导入学生表

    protected void addStudentList(HttpServletRequest request, HttpServletResponse response);

    //导入课程表

    protected void addCourseList(HttpServletRequest request, HttpServletResponse response);

    //更改用户名

    protected void updateAccount(HttpServletRequest request, HttpServletResponse response);

    //更改姓名

    protected void updateName(HttpServletRequest request, HttpServletResponse response);

    //更改密码

    protected void updatePassword(HttpServletRequest request, HttpServletResponse response);

}

## 业务逻辑层

管理员端的业务逻辑层，模块的业务逻辑层将调用底层的数据持久层将相应的信息获取，并传递给控制层该层实现了各种逻辑判断， 即业务逻辑的封装，实现各种需求功能， 并将处理后的数据传输给控制层，再由控制层进行简单处理。

表 3-3 用户管理模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | service | 调用说明 | 出口 |
| 登录、修改个人信息、管理用户 | AdminManager.java | 调用ManagerDao | 返回给Action |
| 学生账号管理 | StudentManage.java | 调用StudentDao | 返回给Action |
| 教师账号管理 | TeacherManage.java | 调用TeacherDao | 返回给Action |
| 助教账号管理 | AssistantManager.java | 调用AssistantDao | 返回给Action |

AdminManager.java主要方法：

package hwms.service;

public class AdminManage {

    private ManageDao manageDao = new ManageDao();  //引入管理员dao层接口

    public void setManageDao(ManageDao manageDao);

    public List<Manage> getAllManage();

    public boolean getManageLogin(Manage manage);

    public Manage getManageByAccount(String m\_account);

    public boolean updateManageAccount(String newAccount, String oldAccount);

    public boolean updateManageName(Manage manage) ;

    public boolean updateManagePassword(Manage manage);

}

StudentManage.java 接口主要方法：

package hwms.service;

public class StudentManage {

    private StudentDao studentDao = new StudentDao();   //引入学生dao层接口

    public void setStudentDao(StudentDao studentDao);

    public Student getStudentByAccount(String account);

    public boolean addStudent(Student student);

    public boolean deleteStudent(String account);

    public boolean updateStudentName(Student student);

    public boolean updateStudentPassword(Student student);

    public List<Student> getAllStudent();

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

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

    public boolean getStudentLogin(Student student);

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

    public boolean getCheckWorkStatus(int w\_num);

    public boolean updateStudent(Student student);

}

TeacherManage.java接口主要方法：

package hwms.service;

public class TeacherManage {

    private TeacherDao teacherDao = new TeacherDao();   //引入教师dao层接口

    public void setStudentDao(TeacherDao teacherDao);

    public Teacher getTeacherByAccount(String account);

    public boolean addTeacher(Teacher teacher);

    public boolean deleteTeacher(String account);

    public boolean updateTeacherName(Teacher teacher);

    public boolean updateTeacherPassword(Teacher teacher);

    public boolean updateTeacher(Teacher teacher);

    public List<Teacher> getAllTeacher();

    public boolean getTeacherLogin(Teacher teacher);

}

AssistantManage.java接口主要方法：

package hwms.service;

public class AssistantManage {

    private AssistantDao assistantDao = new AssistantDao(); //引入学生dao层接口

    public void setAssistantDao(AssistantDao assistantDao);

    public boolean addAssistant(Assistant assistant);

    public boolean deleteAssistant(String account);

    public boolean updateAssistant(Assistant assistant);

    public List<Assistant> getAllAssistant();

    public boolean getAssistantLogin(Assistant assistant);

    public Assistant getAssistantByAccount(String account);

    public boolean updateAssistantName(Assistant assistant);

    public boolean updateAssistantPassword(Assistant Assistant);

}

## 数据持久层

用户管理模块的数据持久层能对基本数据进行增删改查。

表 3-4 用户管理模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用模型 | 说明 |
| 登录、修改信息、获取用户列表 | ManagerDao.java | 调用Manager.java | 获取管理员列表、个人信息的修改 |
| 课程操作列表 | CourseDao.java | Course.java | 获取课程列表、课程的增删查改 |

ManagerDao.java定义对用户列表获取以及个人信息的修改的接口：

package hwms.dao;

public class ManageDao extends DBOper {

    private String server;

    private String dbname;

    private String dbuser;

    private String dbpwd;

    // 获取管理员列表

    public List<Manage> getManage();

    // 根据用户名获取用户

    public Manage getManageByAccount(String account);

    // 管理员登录

    public boolean loginManage(Manage manage);

    // 修改管理员账号

    public boolean updateManageAccount(String newAccount,String oldAccount);

    // 修改管理员姓名

    public boolean updateManageName(Manage manage);

    // 修改管理员密码

    public boolean updateManagePassword(Manage manage);

}

CourseDao.java 接口定义对课程信息进行增、 删、 改、 查的接口：

package hwms.dao;

public class CourseDao extends DBOper {

    private String server;

    private String dbname;

    private String dbuser;

    private String dbpwd;

    // 添加课程

    public boolean addCourse(Course course);

    // 删除课程

    public boolean delCourse(String cour\_num);

    public List<Course> getCourseByTeacher(String cour\_t\_id);

    // 查询课程号

    public Course getCourseNumByCourName(String cour\_name);

    public Course getCourseNameByCourseID(String cour\_id);

    public List<Course> getAllCourse();

    public boolean updateCourse(Course course);

    public Course getCourseByNum(String cour\_num);

    public Course getCourseByCourseID(String cour\_id);

}

## 数据模型层

管理员端的数据模型层，是数据库中表与数据实现一种数据载体，是数据库中表中数据到Java对象的映射

|  |  |
| --- | --- |
| 模型 | 描述 |
| Assistant.java | 对助教信息持久化 |
| Course.java | 对课程信息持久化 |
| Student.java | 对学生信息持久化 |
| Manage.java | 对管理员信息持久化 |
| Teacher.java | 对教师信息持久化 |

Assistant.java 主要属性与方法：

package hwms.entity;

public class Assistant {

    private int a\_Num;

    private String a\_Account;       //ID号

    private String a\_Name;          //姓名

    private String a\_Password;      //密码

    //以及相关get、set方法

}

Course.java 主要属性与方法：

package hwms.entity;

public class Course {

    int cour\_Num;

    String cour\_Name;

    int cour\_Credit;

    String cour\_ID;

    int cour\_Count;

    String cour\_t\_ID;

    //以及相关get、set方法

}

Student.java 主要属性与方法：

package hwms.entity;

public class Student {

    private int s\_Num;

    private String s\_Account;       //学号

    private String s\_Name;          //姓名

    private String s\_Class;         //班级

    private String s\_Password;      //密码

    //以及相关get、set方法

}

Manage.java 主要属性与方法：

package hwms.entity;

public class Manage {

    private int m\_Num;

    private String m\_ID;            //ID号

    private String m\_Account;       //用户名

    private String m\_Name;          //姓名

    private String m\_Password;      //密码

    //以及相关get、set方法

}

Teacher.java 主要属性与方法：

package hwms.entity;

public class Teacher {

    private int t\_Num;

    private String t\_Account;       //ID号

    private String t\_Name;          //姓名

    private String t\_Password;      //密码

    //以及相关get、set方法

}

# 学生端详细设计

## 表现层

学生端的模块表现层，主要完成作业的查看、上传和公告查看，课程查询，教师信息查询，以及课程资源的查看与下载。

表 4-1 学生端模块表现层 JSP 列表

|  |  |  |
| --- | --- | --- |
| 界面 | jsp | 功能描述 |
| 作业批阅查看界面 | s\_showcheck.jsp | 查看教师批阅结果 |
| 作业的上传、下载界面 | s\_downloadwork.jsp、  s\_uploadwork.jsp | 作业的上传、下载 |
| 公告查看等相关信息 | s\_home.jsp | 公告显示 |
| 学生主页 | student.jsp | 包含其他页面的入口 |
| 管理个人账户界面 | s\_personalsetting.jsp | 登入/登出/个人信息 |

## 控制层

学生端的控制层，主要是控制页面的跳转与将从业务逻辑层获取到的数据传送到表现层。

表 4-2用户管理模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 登录 | loginservlet.java | SUCCESS | 当查询、添加等操作成功时，显示操作成功。 |
| ERROR | 当查询、添加等操作失败时，显示操作失败。 |
| 登出 | logoutservlet.java | SUCCESS | 当添加操作成功时，显示操作成功 |
| EORROR | 当添加操作成失败时，显示操作失败 |
| 学生相关信息（包含公告、课程、作业列表等） | studentservlet.java | SUCCESS | 当查询、添加、修改等操作成功时，显示操作成功。 |
| EORROR | 当查询、添加、修改等操作失败时，显示操作失败。 |

在控制层中 studentservlet.java的主要属性与方法如下所示：

package hwms.servlet;

/\*\*

 \* Servlet implementation class studentServlet

 \*/

@WebServlet("/studentServlet")

public class studentServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    /\*\*

     \* @see HttpServlet#HttpServlet()

     \*/

    public studentServlet() ;

    /\*\*

     \* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse

     \*      response)

     \*/

    protected void doGet(HttpServletRequest request, HttpServletResponse response);

    /\*\*

     \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse

     \*      response)

     \*/

    protected void doPost(HttpServletRequest request, HttpServletResponse response);

    private void s\_home(HttpServletRequest request, HttpServletResponse response);

    protected void s\_personalseting(HttpServletRequest request, HttpServletResponse response);

    protected void s\_showcheck(HttpServletRequest request, HttpServletResponse response);

    protected void s\_uploadwork\_jsp(HttpServletRequest request, HttpServletResponse response);

    protected void updateName(HttpServletRequest request, HttpServletResponse response);

    protected void updatePassword(HttpServletRequest request, HttpServletResponse response);

}

## 业务逻辑层

学生端的业务逻辑层，模块的业务逻辑层将调用底层的数据持久层将相应的信息获取，并传递给控制层该层实现了各种逻辑判断， 即业务逻辑的封装，实现各种需求功能， 并将处理后的数据传输给控制层，再由控制层进行简单处理。

表 4-3 用户管理模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | service | 调用说明 | 出口 |
| 获取作业列表 | CheckWorkManage.java | 调用CheckWorkDao | 返回给Action |

CheckWorkManage.java主要方法：

package hwms.service;

public class CheckWorkManage {

    private CheckWorkDao checkworkDao = new CheckWorkDao();

    public void setCheckWorkDao(CheckWorkDao checkworkDao);

    public boolean addCheckWork(String ch\_s\_account,String ch\_cour\_id, int ch\_w\_num, int ch\_score);

    public CheckWork getCheckWorkBySnoWno(String ch\_s\_account, int ch\_w\_num);

    public List<CheckWork> getCheckWorkByCourIDWNum(String ch\_cour\_id, String ch\_w\_num);

    public boolean updateCheckWork(String ch\_upTime, String ch\_path,String ch\_s\_account,String ch\_cour\_id, int ch\_w\_num );

    public boolean updateCheckWorkByTeacher(String ch\_num,String ch\_score,String ch\_mark,String ch\_checkTime);

    public boolean deleteCheckWork(String w\_num);

    public CheckWork getCheckWorkByNum(String ch\_num);

}

## 数据持久层

学生模块的数据持久层能对基本数据进行修改。

表 4-4 学生模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用模型 | 说明 |
| 登录、修改信息、获取用户列表 | StudentDao.java | 调用Student.java | 获取学生列表、个人信息的修改 |

StudentDao.java定义对学生列表获取以及个人信息的修改的接口：

package hwms.dao;

public class StudentDao extends DBOper {

    private String server;

    private String dbname;

    private String dbuser;

    private String dbpwd;

    // 获取学生列表

    public List<Student> getStudent();

    // 根据用户名获取用户

    public Student getStudentByAccount(String Account);

    // 修改学生姓名

    public boolean editStudentName(Student student);

    // 修改学生密码

    public boolean editStudentPassword(Student student);

    // 添加学生

    public boolean addStudent(Student student);

    // 删除指定学生

    public boolean delStudent(String Account);

    // 学生登录

    public boolean loginStudent(Student student);

    //

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

    public boolean getCheckWorkStatus(int w\_num);

    // 获取作业列表

    public List<Work> getStudentWork(String w\_cour\_id);

    // 获取作业审核列表

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

    public boolean updateStudent(Student student);

}

## 数据模型层

学生端的数据模型层，是数据库中表与数据实现一种数据载体，是数据库中表中数据到Java对象的映射

|  |  |
| --- | --- |
| 模型 | 描述 |
| Student.java | 对学生信息持久化 |

# 助教端详细设计

## 表现层

助教端的模块表现层，主要完成作业的批阅与受邀、公告的发布、查看和修改，以及课程资源的上传。

表 5-1 助教端模块表现层 JSP 列表

|  |  |  |
| --- | --- | --- |
| 界面 | jsp | 功能描述 |
| 作业批阅 | a\_checkwork.jsp  a\_invitation.jsp | 接收批阅邀请，完成作业的批阅 |
| 提交统计 | a\_statistics.jsp | 作业的提交 |
| 课程相关信息 | a\_course.jsp | 公告显示 |
| 助教主页 | assistant.jsp | 包含其他页面的入口 |
| 管理个人账户界面 | a\_personalsetting.jsp | 登入/登出/个人信息 |

## 控制层

助教端的控制层，主要是控制页面的跳转与将从业务逻辑层获取到的数据传送到表现层。

表 5-2用户管理模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 登录 | loginservlet.java | SUCCESS | 当查询、添加等操作成功时，显示操作成功。 |
| ERROR | 当查询、添加等操作失败时，显示操作失败。 |
| 登出 | logoutservlet.java | SUCCESS | 当添加操作成功时，显示操作成功 |
| EORROR | 当添加操作成失败时，显示操作失败 |
| 助教相关信息（包含公告、作业列表等） | assistantServlet.java | SUCCESS | 当查询、添加、修改等操作成功时，显示操作成功。 |
| EORROR | 当查询、添加、修改等操作失败时，显示操作失败。 |

在控制层中 assistantServlet.java的主要属性与方法如下所示：

package hwms.servlet;

/\*\*

 \* Servlet implementation class assistantServlet

 \*/

@WebServlet("/assistantServlet")

public class assistantServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    /\*\*

     \* @see HttpServlet#HttpServlet()

     \*/

    public assistantServlet();

    /\*\*

     \* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse

     \*      response)

     \*/

    protected void doPost(HttpServletRequest request, HttpServletResponse response);

    protected void a\_course(HttpServletRequest request, HttpServletResponse response);

    protected void a\_invitation(HttpServletRequest request, HttpServletResponse response);

    protected void a\_statistics(HttpServletRequest request, HttpServletResponse response);

    protected void a\_checkwork(HttpServletRequest request, HttpServletResponse response);

    protected void correctWork(HttpServletRequest request, HttpServletResponse response);

    protected void accept(HttpServletRequest request, HttpServletResponse response);

    protected void refuse(HttpServletRequest request, HttpServletResponse response);

    protected void updateName(HttpServletRequest request, HttpServletResponse response);

    protected void updatePassword(HttpServletRequest request, HttpServletResponse response);

}

## 业务逻辑层

助教端的业务逻辑层，模块的业务逻辑层将调用底层的数据持久层将相应的信息获取，并传递给控制层该层实现了各种逻辑判断， 即业务逻辑的封装，实现各种需求功能， 并将处理后的数据传输给控制层，再由控制层进行简单处理。

表 5-3 用户管理模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | service | 调用说明 | 出口 |
| 公告管理 | NoticeManage.java | 调用NoticeDao | 返回给Action |

NoticeManage.java主要方法：

package hwms.service;

public class NoticeManage {

    private NoticeDao noticeDao = new NoticeDao();  //引入公告dao层接口

    public void setNoticeDao(NoticeDao noticeDao);

    public List<Notice> getStudentNotice(String nt\_c\_id);

    public List<Notice> getTeacherNotice(String nt\_t\_id);

    public boolean addNotice(Notice notice);

    public boolean deleteNotice(int id);

    public boolean updateNotice(Notice notice);

}

## 数据持久层

助教模块的数据持久层能对基本数据进行增删查改。

表 5-4 助教模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用模型 | 说明 |
| 获取用户列表 | StudentDao.java | 调用Student.java | 获取学生列表、个人信息的修改 |
| 设置公告 | NoticeDao.java | 调用Notice.java | 获取公告的列表信息 |

NoticeDao.java定义对公告获取：

package hwms.entity;

public class Notice {

    private int nt\_Num;         //公告号

    private String nt\_Title;    //主题

    private String nt\_Content;  //内容

    private String nt\_Time;     //发布时间

    private String nt\_t\_ID;     //教师号

    private String nt\_t\_Name;   //教师姓名

    private String nt\_cour\_ID;      //课程号

    //以及get、set方法

}

## 数据模型层

助教端的数据模型层，是数据库中表与数据实现一种数据载体，是数据库中表中数据到Java对象的映射

|  |  |
| --- | --- |
| 模型 | 描述 |
| Notice.java | 对公告信息持久化 |

# 教师端详细设计

## 表现层

教师端的模块表现层，主要完成作业的批阅、邀请助教、查看和修改，以及课程资源的上传。

表 5-1 教师端模块表现层 JSP 列表

|  |  |  |
| --- | --- | --- |
| 界面 | jsp | 功能描述 |
| 作业批阅 | **t\_statistics.jsp** | 作业的批阅 |
| 邀请助教 | **t\_allinvitation.jsp**  **t\_allnotice.jsp**  **t\_allworks.jsp** | 公告发布 |
| 教师主页 | **t\_publishwork.jsp** | 查看和修改 |
| 管理个人账户界面 | **t\_personalseting.jsp** | 设置个人信息 |

## 控制层

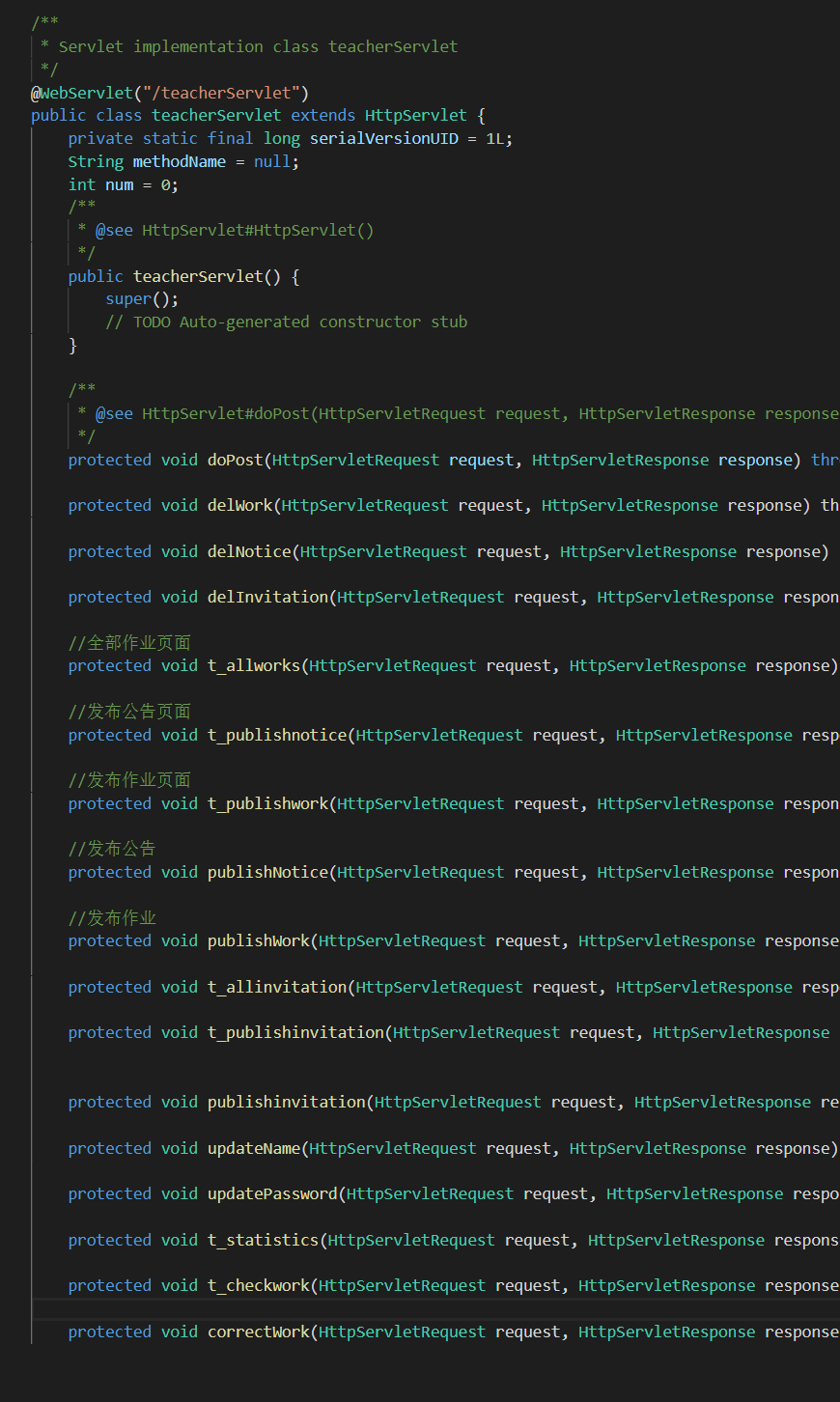
教师端的控制层，主要是控制页面的跳转与将从业务逻辑层获取到的数据传送到表现层。

表 5-2用户管理模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 登录 | loginservlet.java | SUCCESS | 当查询、添加等操作成功时，显示操作成功。 |
| ERROR | 当查询、添加等操作失败时，显示操作失败。 |
| 登出 | logoutservlet.java | SUCCESS | 当添加操作成功时，显示操作成功 |
| EORROR | 当添加操作成失败时，显示操作失败 |
| 教师相关信息（包含公告、作业列表等） | **teacherServlet.java** | SUCCESS | 当查询、添加、修改等操作成功时，显示操作成功。 |
| EORROR | 当查询、添加、修改等操作失败时，显示操作失败。 |

在控制层中 **teacherServlet.java**的主要属性与方法如下所示：





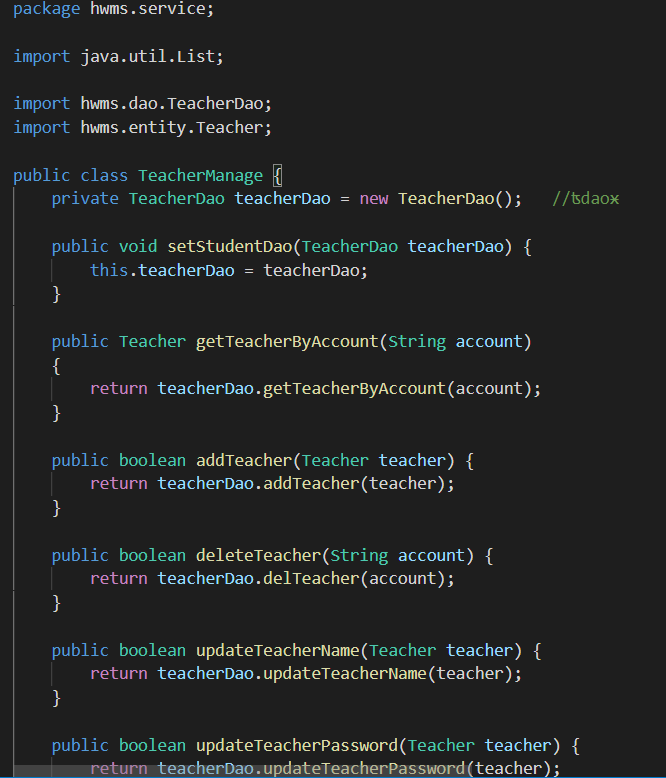
## 业务逻辑层

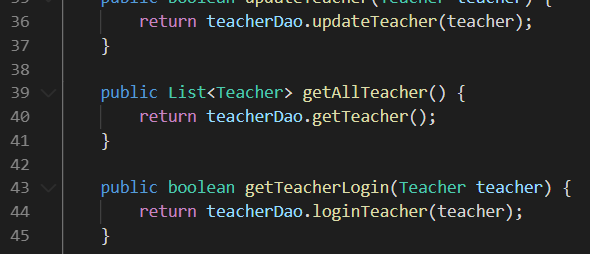
教师端的业务逻辑层，模块的业务逻辑层将调用底层的数据持久层将相应的信息获取，并传递给控制层该层实现了各种逻辑判断， 即业务逻辑的封装，实现各种需求功能， 并将处理后的数据传输给控制层，再由控制层进行简单处理。

表 5-3 教师管理模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | service | 调用说明 | 出口 |
| 公告管理 | **TeacherManage.java** | 调用**TeacherDao** | 返回给Action |

**TeacherManage.java**主要方法：





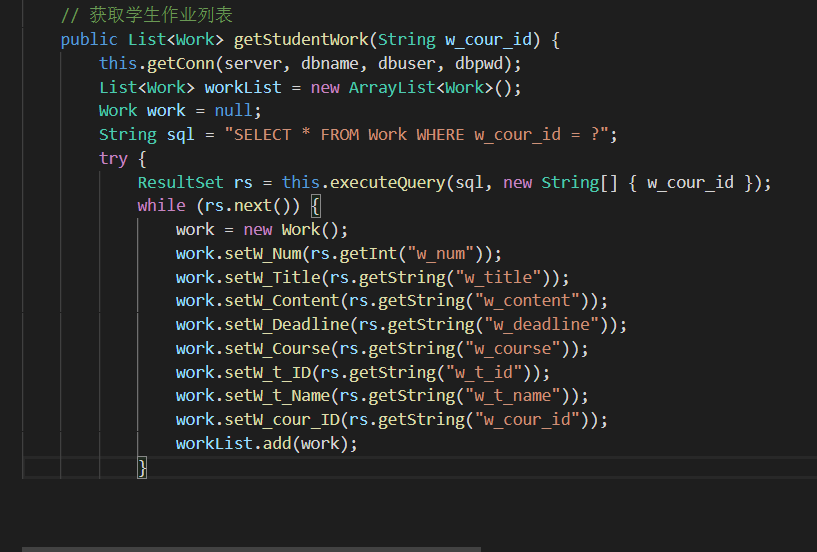
## 数据持久层

教师模块的数据持久层能对基本数据进行增删查改。

表 5-4 教师模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用模型 | 说明 |
| 获取用户列表 | **TeacherDao.java** | 调用**Teacher.java** | 获取教师列表、个人信息的修改 |
| 设置作业 | **WorkDao.java** | 调用**Work.java** | 发布删除作业 |

**WorkDao.java**定义对作业安排：



## 数据模型层

助教端的数据模型层，是数据库中表与数据实现一种数据载体，是数据库中表中数据到Java对象的映射

|  |  |
| --- | --- |
| 模型 | 描述 |
| **Work.java** | 对作业信息的持久化 |

# 用户通用模块详细设计

## 表现层

通用端的模块表现层，主要完成密码找回/登录、个人信息修改。

表 5-1 教师端模块表现层 JSP 列表

|  |  |  |
| --- | --- | --- |
| 界面 | jsp | 功能描述 |
| 密码找回/登录 | **manager.jsp** | 找回/登录 |
| 管理个人账户界面 | **t\_personalseting.jsp** | 修改个人信息 |

## 控制层

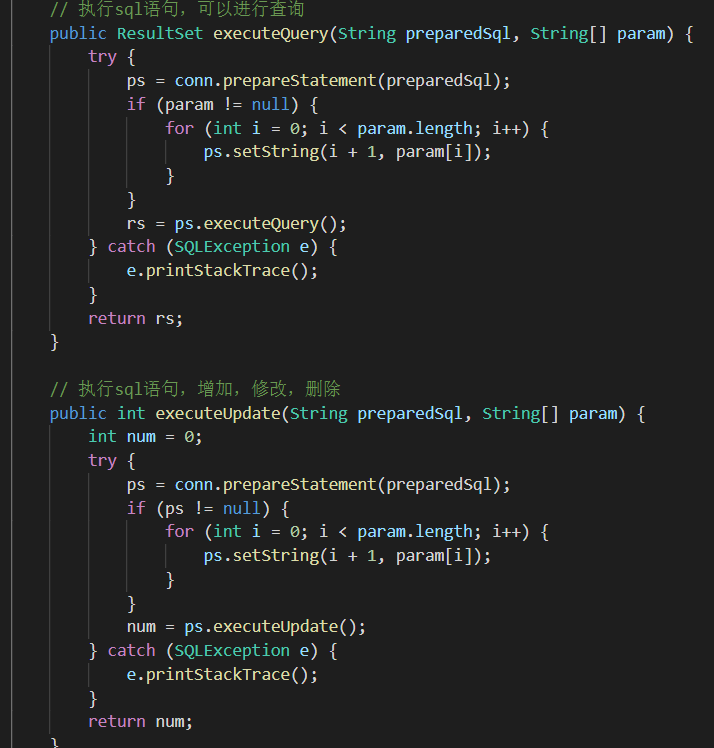
教师端的控制层，主要是控制页面的跳转与将从业务逻辑层获取到的数据传送到表现层。

表 5-2用户管理模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 登录 | loginservlet.java | SUCCESS | 当查询、添加等操作成功时，显示操作成功。 |
| ERROR | 当查询、添加等操作失败时，显示操作失败。 |
| 登出 | logoutservlet.java | SUCCESS | 当添加操作成功时，显示操作成功 |
| EORROR | 当添加操作成失败时，显示操作失败 |
| 登录相关信息 | **DBOper.java** | SUCCESS | 当查询、添加、修改等操作成功时，显示操作成功。 |
| EORROR | 当查询、添加、修改等操作失败时，显示操作失败。 |

**DBOper.java的相关代码：**





## 业务逻辑层

通用端的业务逻辑层，模块的业务逻辑层将调用底层的数据持久层将相应的信息获取，并传递给控制层该层实现了各种逻辑判断， 即业务逻辑的封装，实现各种需求功能， 并将处理后的数据传输给控制层，再由控制层进行简单处理。

表 5-3 通用管理模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | service | 调用说明 | 出口 |
| 公告管理 | **Manage.java** | 调用**Dao** | 返回给Action |

**Manage.java**主要方法：



## 数据持久层

通用模块的数据持久层能对基本数据进行增删查改。

表 5-4 通用模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用模型 | 说明 |
| 获取教师信息 | **AddteacherDao.jsp** | 调用**Addteacher.java** | 获取教师列表、个人信息的修改 |

**AddteacherDao.Jsp**定义对作业安排：



## 数据模型层

通用端的数据模型层，是数据库中表与数据实现一种数据载体，是数据库中表中数据到Java对象的映射

|  |  |
| --- | --- |
| 模型 | 描述 |
| **Work.java** | 对作业信息的持久化 |