《面向对象技术》

学生管理系统

院 系： 信息工程学院

年 级： 2016 级

班 级： 计科一班

姓 名： 李志伟

学 号： 201677I0215

日 期： 2018年12月

目录

[1 系统总体设计 2](#_Toc532306945)

[1.1 需求概述 2](#_Toc532306946)

[1.2 基本设计概念和处理流程 3](#_Toc532306947)

[1.3 结构 4](#_Toc532306948)

[1.3 功能分配 6](#_Toc532306949)

[1.4 运行设计 7](#_Toc532306950)

[2 系统详细设计 8](#_Toc532306951)

[2.1 编写目的 8](#_Toc532306952)

[2.2 程序系统结构 8](#_Toc532306953)

[2.3 模块详细设计 11](#_Toc532306954)

[2.4 接口设计 15](#_Toc532306955)

[2.5 数据库设计 21](#_Toc532306956)

[2.6 系统出错处理设计 24](#_Toc532306957)

[2.7 补救措施 24](#_Toc532306958)

[附录 25](#_Toc532306959)

## 1 系统总体设计

### 1.1 需求概述

本系统旨在开发一个统一的学生信息管理系统，利用信息化手段辅助学生信息的管理，达到提升学校学生管理信息化的水平的目的。教务管理人员能够利用管理员账户方便地进行教务管理工作，包括学生信息管理、教师信息管理、公告管理、成绩与课程管理等；教师用户能够利用该系统进行学生成绩录入及修改、查看和发布公告等；学生用户能够利用该系统进行成绩查询及导出、查看教务通知等。

SIMS学生信息管理系统主要需求列表如下：

1. 学生信息管理。包括学生信息增加、删除、修改、查询。
   1. 教师用户和管理员用户具有对所有学生信息的增加、删除、修改、查询权限；
   2. 学生用户只能查看和修改自身的信息；
   3. 学生信息能够批量下载。
2. 教师信息管理。包括学生信息增加、删除、修改、查询。
   1. 管理员用户具有对所有学生信息的增加、删除、修改、查询权限；
   2. 教师用户只能查看和修改自身的信息；
   3. 教师信息能够批量下载。
3. 注册管理功能。包括用户注册、密码修改等。
   1. 当新增教师或学生信息时，数据库中会自动在用户表中插入新的用户信息，用户ID为其职工号或学号，密码默认为000000；
   2. 在新增用户信息时，允许修改证件照。一旦信息录入结束，则只能通过管理员进行修改。
4. 课程及成绩管理。包括课程信息的增加、删除、修改，以及成绩的录入和修改。
   1. 课程信息只能通过管理员从数据库进行修改。一旦一门课程被删除，其关联的所有成绩也将被删除。
   2. 教师及管理员用户能够录入或修改所有学生的成绩；
   3. 学生可以查看或导出自己的成绩
5. 用户登录功能。包括用户名、密码检查，验证码校验，密码找回等。

### 1.2 基本设计概念和处理流程

针对本系统的教务管理人员、教师、学生三种不同的角色，系统的处理流程如下图所示：

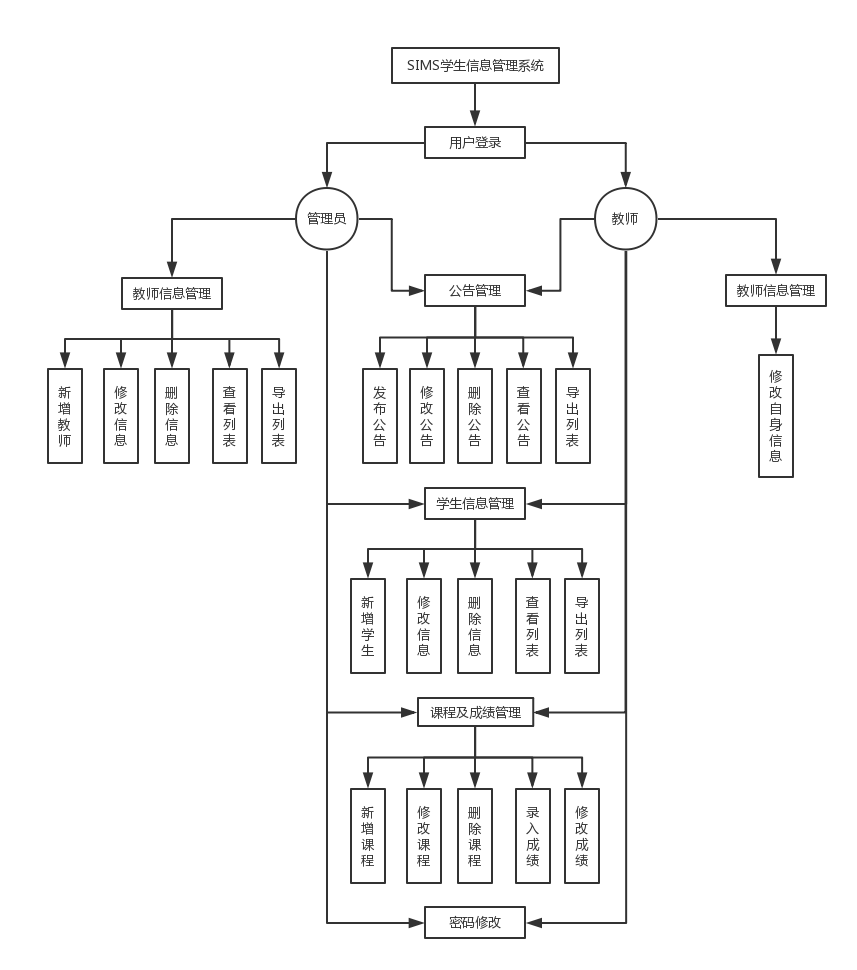


图 1 管理员及教师登陆时的系统流程图

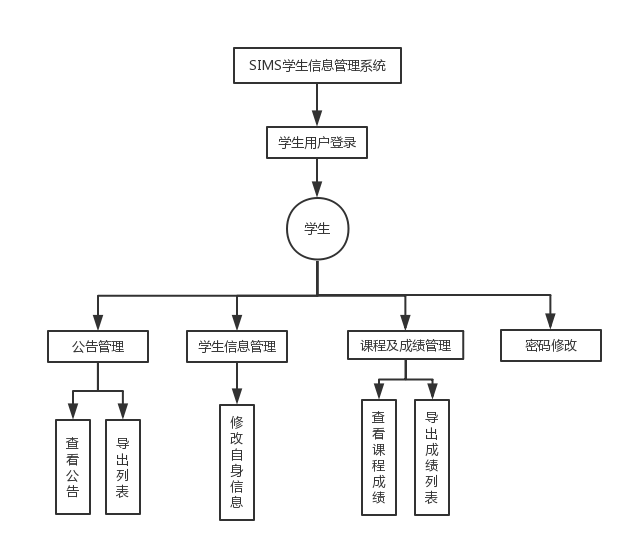


图 2 学生登录时的系统流程图

### 1.3 结构

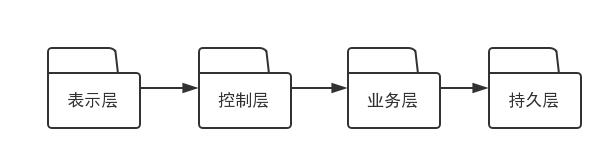
本系统的体系结构如图3所示：

图 3 系统架构图

本系统体系结构大致可以定义为：

1. 用户浏览器中的表示层主要是通过Servlet + JSP技术实现的。由显示视图产生一个请求，请求被在web.xml中寻求对应的URL，找到对应的Servlet类后， Servlet类执行相应的业务逻辑。一旦Servlet类处理完相应的业务逻辑，就会提供对应的参数返回给前端JSP，它指明了处理的结果。当Servlet类将结果返回给JSP时，请求的过程也就完成了。
2. 中间业务层是通过Spring Boot框架实现的，首先建立一个BackEndApplication启动类，该类需要导入org.springframework.boot.SpringApplication、org.springframework.boot.autoconfigure.SpringBootApplication、org.springframework.boot.web.servlet.ServletComponentScan，之后通过注解@SpringBootApplication，这样就可以保证业务层能持续运转。之后业务层在部署到服务器上时为了能够自动运行，需要再建立BackEndServletIntializer类，通过继承SpringBootServletIntializer以正常运行。
3. 持久层(PO)由hibernate架构及Spring Boot框架实现，对每个表的JavaBean类添加注解以及实现CrudRepository，就可以实现通过简单几行代码实现对该表的增删改查。如果需要更多的复杂操作，则需要继承JpaSpecificationExecutor类来实现相关功能。

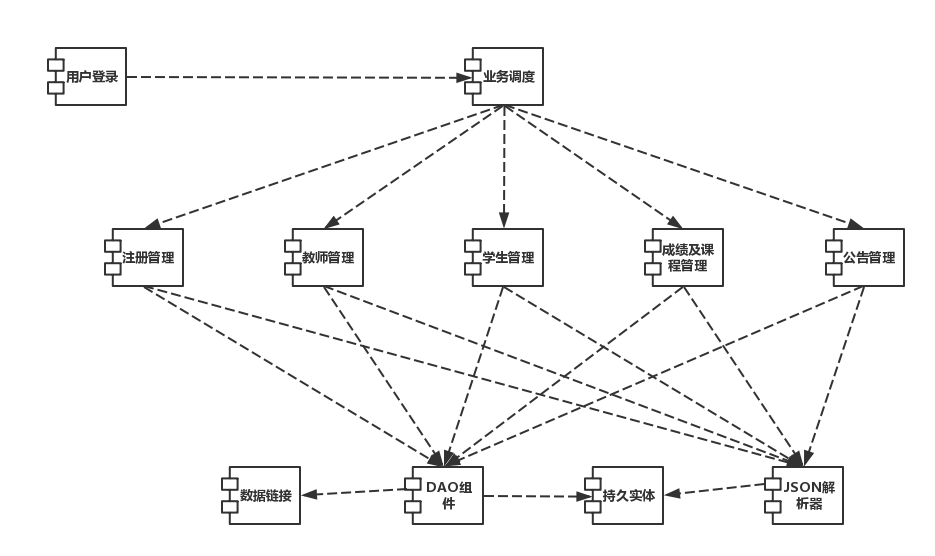
本系统组件如图4所示：

图 4 系统组件图

### 1.3 功能分配

本系统中各功能需求与程序模块（组件）之间的关系如下表所示：

表 1 功能分配表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **功能需求** | **注册管理** | **教师管理** | **学生管理** | **成绩及课程管理** | **公告管理** |
| **用户注册** | **√** |  |  |  |  |
| **修改密码** | **√** |  |  |  |  |
| **添加教师** |  | **√** |  |  |  |
| **修改教师** |  | **√** |  |  |  |
| **删除教师** |  | **√** |  |  |  |
| **查询教师** |  | **√** |  |  |  |
| **导出教师** |  | **√** |  |  |  |
| **添加学生** |  |  | **√** |  |  |
| **修改学生** |  |  | **√** |  |  |
| **删除学生** |  |  | **√** |  |  |
| **查询学生** |  |  | **√** |  |  |
| **导出学生** |  |  | **√** |  |  |
| **添加课程** |  |  |  | **√** |  |
| **删除课程** |  |  |  | **√** |  |
| **录入成绩** |  |  |  | **√** |  |
| **修改成绩** |  |  |  | **√** |  |
| **查询成绩** |  |  |  | **√** |  |
| **导出成绩** |  |  |  | **√** |  |
| **发布公告** |  |  |  |  | **√** |
| **修改公告** |  |  |  |  | **√** |
| **删除公告** |  |  |  |  | **√** |
| **查询公告** |  |  |  |  | **√** |
| **导出公告** |  |  |  |  | **√** |

### 1.4 运行设计

#### 1.4.1 运行模块组合

前端网页在有输入时启动接收数据模块，通过各模块之间的调用，读入并对输入进行格式化。在接收数据模块得到充分的数据时，将调用网络传输模块，将数据通过网络送到服务器，并等待接收服务器返回的信息。接收到返回信息后随即调用数据传输模块，对信息进行处理，产生相应的输出。

服务器程序的接收网络数据模块必须始终处于活动状态。接收到数据后，调用数据处理/查询模块对数据库进行访问，完成调用网络发送模块，将信息返回前端。

#### 1.4.2 运行控制

运行控制将严格按照各模块间函数调用关系来实现。在各事务中心模块中，需对运行控制进行正确的判断，选择正确的运行控制路径。

在网络传输方面，前端网页在发送数据后，将等待服务器确认收到信号。前端收到后，再次等待服务器发送响应数据，然后对数据进行确认。服务器在接到数据后发送确认信号，在对数据处理、访问数据库后，将返回信息送回前端网页，并做相应处理。

## 2 系统详细设计

### 2.1 编写目的

在前一阶段（概要设计说明书）中，已解决了实现该系统需求的程序模块设计问题。包括如何把该系统划分成若干个模块、决定各个模块之间的接口、模块之间传递的信息，以及数据结构、模块结构的设计等。在以下的详细设计报告中将对在本阶段中对系统所做的所有详细设计进行说明。

### 2.2 程序系统结构

#### 2.2.1 主要类图

图 5 SIMS实体-联系图

#### 2.2.2 主要界面设计

图6 SIMS主要界面设计

#### 2.2.3 主要模块程序名

表 2 主要模块程序名

|  |  |
| --- | --- |
| 模块名 | Servlet名 |
| 登录管理 | LoginServlet //登录校验  ImageServlet //验证码生成  VerificationServlet //验证码校验 |
| 公告信息操作 | BulletinDeleteServlet //公告删除  BulletinInfoServlet //公告修改  BulletinInsertServlet //公告插入  BulletinListServlet //公告列表 |
| 教师信息操作 | TeacherDeleteServlet //教师删除  TeacherInfoServlet //教师修改  TeacherListServlet //教师列表 |
| 学生信息操作 | StudentDeleteServlet //学生删除  StudentInfoServlet //学生修改  StudentListServlet //学生列表 |
| 课程及成绩操作 | ScoreSubmitServlet //成绩提交  SubjectListGetServlet //课程列表获取  SubjectListQueryServlet //课程列表查询 |
| 用户信息操作 | UserInsertServlet //新增用户  UserPasswdFindServlet //密码找回  UserPasswdUpdateServlet //密码修改 |

### 2.3 模块详细设计

#### 2.3.1 登录模块设计

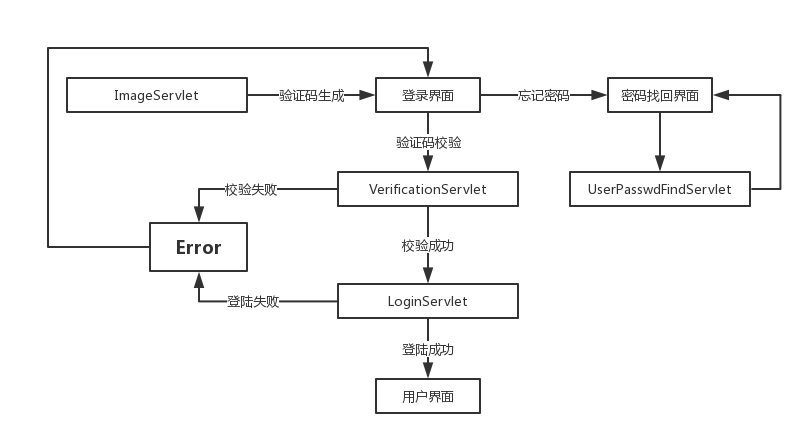
登录身份：Student (level=1), Teacher (level=2), Admin (level=3)

图 7 登陆模块设计

#### 2.3.2 公告模块设计

图 8 公告模块设计

#### 2.3.3教师模块设计

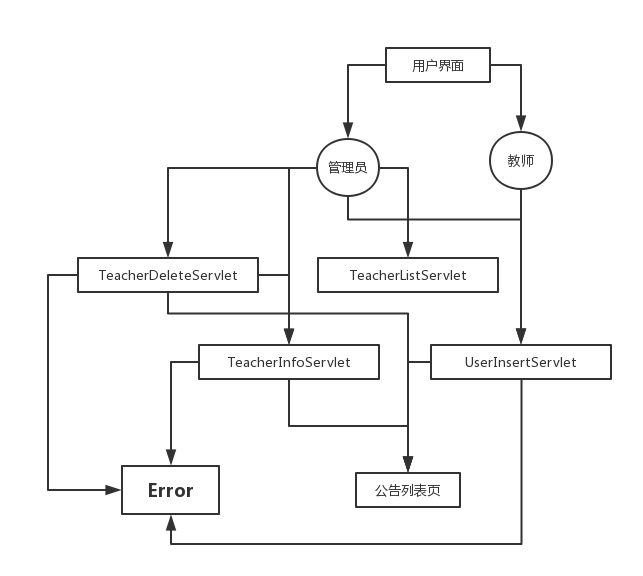


图 9 教师模块设计

#### 2.3.4 学生模块设计

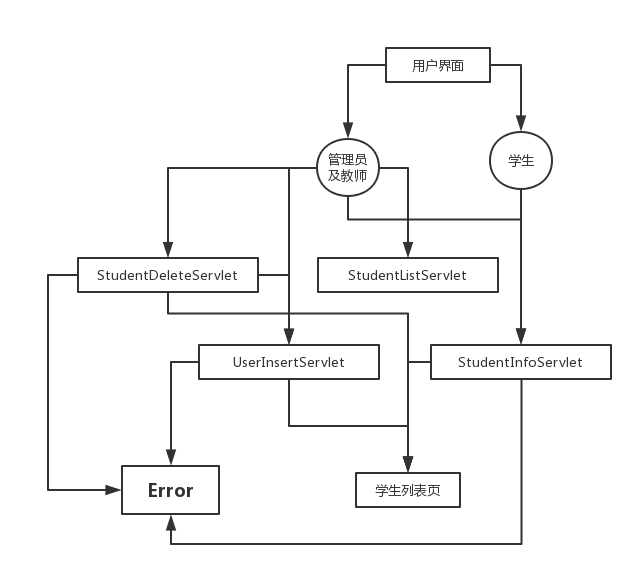


图 10 学生模块设计

#### 2.3.5 课程及成绩模块设计

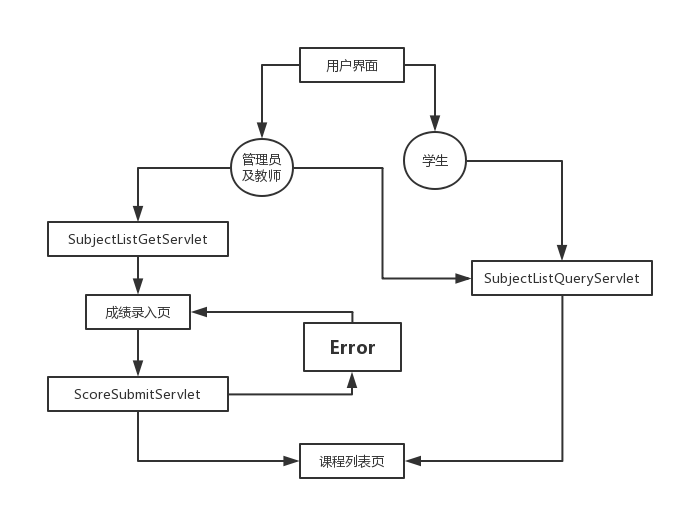


图 11 课程及成绩模块设计

### 2.4 接口设计

#### 2.4.1 用户接口

在用户界面即网页端部分，用户需要一个交互友善的界面，且尽可能的简单明了、易于操作。系统首页如图所示：

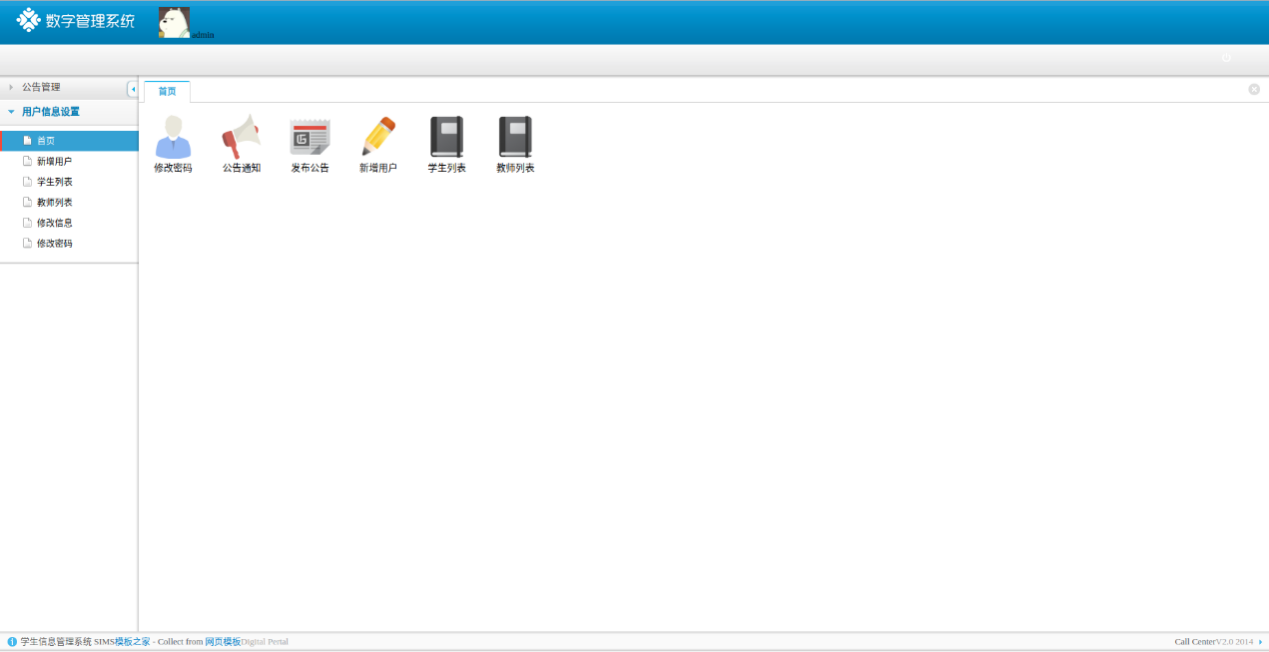


图 12 系统首页

#### 2.4.2 外部接口

服务器端通过使用高度集成的Spring Boot完成了对MySql数据库所有的访问，接口详情如下：

该项目下所有外部接口的访问前缀都是<http://server.aspi.tech:8080/backend>。

1. User类：

增加访问前缀"/user"。

* save(User user): 通过调用该接口传入要存入的用户信息。若用户ID已经存在，则会更新该用户ID对应的用户信息。
  + 参数：userId, passwd, userLevel
  + 访问方式: POST
  + 访问路径: "/save"
* deleteByid(String userId): 通过调用该接口删除指定ID的用户。若成功删除，则会返回"SUCCESS。
  + 参数：userId
  + 访问方式: POST
  + 访问路径: "/deletebyid"
* findAll( ): 通过调用该接口以JSON数组的形式返回所有的用户信息。
  + 访问方式: GET
  + 访问路径: "/findall"
* findById(String userId): 通过调用该接口以JSON形式返回指定ID的用户信息。
  + 访问方式: GET
  + 访问路径: "/findbyid"
* findByLevel(int userLevel): 通过调用该接口以JSON数组的形式返回指定等级的用户信息。
  + 访问方式: GET
  + 访问路径: "/findbylevel"

1. Teacher类：

增加访问前缀"/teacher"。

* save(Teacher teacher): 通过调用该接口传入要存入的教师信息。若教师ID已经存在，则会更新该教师ID对应的教师信息。
  + 参数：teacherId, teacherName, collegeId, gender, birthday(格式:"yyyy-MM-dd"), email, portrait
  + 访问方式: POST
  + 访问路径: "/save"
* deleteByid(String teacherId): 通过调用该接口删除指定ID的教师。若成功删除，则会返回"SUCCESS。
  + 参数：teacherId
  + 访问方式: POST
  + 访问路径: "/deletebyid"
* findAll(Teacher teacher): 通过调用该接口以JSON数组的形式返回教师信息。若该请求有参数，则可以按条件查找指定的教师信息；若无参数，则返回所有的教师信息
  + 参数（可选）：teacherId, teacherName, collegeId, gender, birthday(格式:"yyyy-MM-dd")
  + 访问方式: GET
  + 访问路径: "/findall"

1. Student类：

增加访问前缀"/student"。

* save(Student student): 通过调用该接口传入要存入的学生信息。若学生ID已经存在，则会更新该学生ID对应的学生信息。
  + 参数：studentId, studentName, studentId, gender, birthday(格式:"yyyy-MM-dd"), email, portrait
  + 访问方式: POST
  + 访问路径: "/save"
* deleteByid(String studentId): 通过调用该接口删除指定ID的学生。若成功删除，则会返回"SUCCESS。
  + 参数：studentId
  + 访问方式: POST
  + 访问路径: "/deletebyid"
* findAll(Student student): 通过调用该接口以JSON数组的形式返回学生信息。若该请求有参数，则可以按条件查找指定的学生信息；若无参数，则返回所有的学生信息
  + 参数（可选）：studentId, studentName, majorId, gender, birthday(格式:"yyyy-MM-dd")
  + 访问方式: GET
  + 访问路径: "/findall"

1. Subject类：

增加访问前缀"/subject"。

* save(Subject subject): 通过调用该接口传入要存入的课程信息。若课程ID已经存在，则会更新该课程ID对应的课程信息。
  + 参数：subjectId, subjectName, majorId
  + 访问方式: POST
  + 访问路径: "/save"
* deleteByid(int subjectId): 通过调用该接口删除指定ID的课程。若成功删除，则会返回"SUCCESS。
  + 参数：subjectId
  + 访问方式: POST
  + 访问路径: "/deletebyid"
* findAll(Subject subject): 通过调用该接口以JSON数组的形式返回课程信息。若该请求有参数，则可以按条件查找指定的课程信息；若无参数，则返回所有的课程信息
  + 参数（可选）：subjectId, subjectName, majorId
  + 访问方式: GET
  + 访问路径: "/findall"

1. Score类：

增加访问前缀"/score"。

* save(Score score): 通过调用该接口传入要存入的成绩信息。若成绩ID已经存在，则会更新该成绩ID对应的成绩信息。
  + 参数：scoreId, subjectId, studentId, scoreValue
  + 访问方式: POST
  + 访问路径: "/save"
* deleteByid(int scoreId): 通过调用该接口删除指定ID的成绩。若成功删除，则会返回"SUCCESS。
  + 参数：scoreId
  + 访问方式: POST
  + 访问路径: "/deletebyid"
* findAll(Score score): 通过调用该接口以JSON数组的形式返回成绩信息。若该请求有参数，则可以按条件查找指定的成绩信息；若无参数，则返回所有的成绩信息
  + 参数（可选）：scoreId, subjectId, studentId
  + 访问方式: GET
  + 访问路径: "/findall"

1. Bulletin

增加访问前缀"/bulletin"。

* save(Bulletin bulletin): 通过调用该接口传入要存入的成绩信息。若成绩ID已经存在，则会更新该成绩ID对应的成绩信息。
  + 参数：bulletinId, userId, bulletinTitle, publishedDate(格式:"yyyy-MM-dd HH:mm:ss"), bulletinContext
  + 访问方式: POST
  + 访问路径: "/save"
* deleteByid(int bullId): 通过调用该接口删除指定ID的成绩。若成功删除，则会返回"SUCCESS。
  + 参数：scoreId
  + 访问方式: POST
  + 访问路径: "/deletebyid"
* findAll(Bulletin bulletin): 通过调用该接口以JSON数组的形式返回成绩信息。若该请求有参数，则可以按条件查找指定的成绩信息；若无参数，则返回所有的成绩信息
  + 参数（可选）：bulletinId, userId, bulletinTitle, bulletinContext
  + 访问方式: GET
  + 访问路径: "/findall"

1. 其他：

因其他接口在本次项目中未使用，故不再列出。

#### 2.4.2 内部接口

为了提升代码的可重用率、增强代码的封装性，以及代码的可读性和可维护性，本项目创建了大量的内部接口，具体说明如下：

1. 客户端工具模块ClientUtil。该模块用于发送Post请求到指定的URL，若成功会返回true。
2. Cookie实例化模块CookieUtil。该模块用于创建一个编码为UTF-8，有效期为24小时的Cookie实例。
3. 文件导出模块ExpertUtil。该模块用于导出一个CSV文件或EXCEL文件。
4. 文件操作模块FileUtil。该模块用于将输入流写入文件中。
5. JSON处理模块JsonUtil。该模块能够处理调用外部接口返回的JSON格式数据。
6. 腾讯云对象存储操作模块CosUtil。该模块能创建一个Cos客户端，并将指定文件上传到腾讯云对象存储指定存储桶中，并返回一个外链用于之后的操作。
7. 电子邮件发送模块Email。该模块能够通过调用将指定内容通过指定邮箱发送到用户邮箱中。
8. 分页模块PageDaoImpl。该模块将数据列表一次读入，由用户设定每页显示的数据条数，按页读取一定数据，从而实现分页功能。
9. 验证码生成及校验模块。该模块用于生成验证码图片，用户在登录时需要准确无误地输入图片中的字符，以防止网站被恶意攻击。
10. 公告信息操作模块。该模块用于对公告的操作，包括：新增、删除、修改、删除等。
11. 学生信息操作模块。该模块用于对学生信息的操作，包括：新增、删除、修改、删除等。
12. 教师信息操作模块。该模块用于对教师信息的操作，包括：新增、删除、修改、删除等。
13. 课程及成绩操作模块。该模块用于成绩录入及查询等。
14. 用户信息操作模块。该模块包括用户登录功能、密码找回功能以及密码修改功能等。

### 2.5 数据库设计

#### 2.5.1 数据库表结构

(1) 公告信息表bulletin

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** | ****约束**** | ****备注**** |
| **bull\_id** | int(11) | 否 |  | 主键 | 公告ID，自增 |
| user\_id | char(13) | 否 |  | 外键 | 发布者ID |
| bull\_title | varchar(66) | 否 |  |  | 标题 |
| bull\_published\_date | datetime | 否 | CURRENT\_TIMESTAMP |  | 发布日期 |
| bull\_context | varchar(1023) | 否 |  |  | 公告内容 |

(2) 院系信息表college

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** | **约束** | **备注** |
| **college\_id** | int(11) | 否 |  | 主键 | 院系ID，自增 |
| college\_name | varchar(150) | 否 |  |  | 院系名 |

(3) 专业信息表major

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** | **约束** | **备注** |
| **major\_id** | int(11) | 否 |  | 主键 | 专业ID，自增 |
| college\_id | int(11) | 否 |  | 外键 | 院系ID |
| major\_name | varchar(150) | 否 |  |  | 专业名 |

(4) 成绩表score

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** | **约束** | **备注** |
| **score\_id** | int(11) | 否 |  | 主键 | 成绩ID，自增 |
| subject\_id | int(11) | 否 |  | 外键 | 课程ID |
| student\_id | char(13) | 否 |  | 外键 | 学号 |
| score\_value | decimal(5,2) | 否 |  |  | 成绩 |

(5) 用户信息表sims\_user

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** | **约束** | **备注** |
| **user\_id** | char(13) | 否 |  | 主键 | 用户ID |
| passwd | char(20) | 否 |  |  | 密码 |
| user\_level | smallint(6) | 否 |  |  | 用户等级（1为学生，2为教师，3为管理员） |

(6) 学生信息表student

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** | **约束** | **备注** |
| **student\_id** | char(13) | 否 |  | 主键 | 学号 |
| student\_name | varchar(20) | 否 |  |  | 姓名 |
| gender | char(5) | 是 | NULL |  | 性别 |
| birthday | date | 是 | NULL |  | 生日 |
| email | char(30) | 是 | NULL |  | 电子邮件 |
| portrait | varchar(1023) | 是 | NULL |  | 证件照（存的是外链） |
| major\_id | int(11) | 是 | NULL | 外键 | 专业ID |

(7) 课程信息表subject

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** | **约束** | **备注** |
| **subject\_id** | int(11) | 否 |  | 主键 | 课程ID |
| subject\_name | varchar(100) | 否 |  |  | 课程名 |
| major\_id | int(11) | 否 |  | 外键 | 专业ID |

(8) 教师-课程关联表teach

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** | **约束** | **备注** |
| teacher\_id | char(13) | 否 |  | 外键 | 教师号 |
| sub\_id | int(11) | 否 |  | 外键 | 课程ID |

(9) 教师信息表teacher

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段 | **类型** | **空** | **默认** | **约束** | **备注** |
| **teacher\_id** | char(13) | 否 |  | 主键 | 教师号 |
| teacher\_name | varchar(30) | 否 |  |  | 教师姓名 |
| college\_id | int(11) | 否 |  | 外键 | 院系ID |
| gender | char(5) | 是 | NULL |  | 性别 |
| birthday | date | 是 | NULL |  | 生日 |
| email | char(30) | 是 | NULL |  | 电子邮件 |
| portrait | varchar(1023) | 是 | NULL |  | 证件照（存的是图片外链） |

#### 2.5.2 数据结构与程序的关系

表 3 功能模块与相应的数据表之间的关系表

|  |  |
| --- | --- |
| 程序模块 | 主要涉及的表结构 |
| 公告信息操作模块 | bulletin |
| 学生信息操作模块 | student |
| 教师信息操作模块 | teacher |
| 用户信息操作模块 | user、teacher、student |
| 课程及成绩操作模块 | subject、score |

### 2.6 系统出错处理设计

表 4 系统出错及处理

|  |  |  |
| --- | --- | --- |
| **出错名称** | **系统输出信息** | **处理方法** |
| **用户名与密码不匹配** | “密码错误，请重新输入！” | 返回登录页 |
| **用户名或密码没有填** | “用户名或密码不得为空！” | 返回登陆页 |
| **验证码错误** | “验证码错误！” | 返回登录页 |
| **公告标题或内容未填写** | “标题或内容未填写！” | 返回公告新增页 |
| **学号/教师号、姓名未填写** | “信息不得为空!” | 返回用户新增页 |
| **找回密码时填写用户名不存在** | “该用户不存在” | 返回密码找回页 |
| **修改密码两次输入不匹配** | “两次密码不相同，请重新输入！” | 返回密码修改页 |
| **录入成绩大于100或小于0** | “成绩录入有误！” | 该课成绩设置为0 |

### 2.7 补救措施

由于数据在数据库中已经有备份， 且数据库明天都会自动备份一次，故在系统出错后可以依靠数据库的恢复功能，并且依靠日志文件使系统再启动，就算系统崩溃用户数据也不会丢失或遭到破坏，但有可能占用更多的数据存储空间。

## 附录

1 公告查询条件拼接功能

// 通过部分内容模糊查询数据

@Transactional

public List<Bulletin> findAllByParams(final Bulletin bulletinParams) {

List<Bulletin> bulletins = bulletinRepository.findAll(new Specification<Bulletin>(){

@Override

public Predicate toPredicate(Root<Bulletin> root, CriteriaQuery<?> criteriaQuery, CriteriaBuilder criteriaBuilder) {

List<Predicate> predicates = new ArrayList<Predicate>();

// 如果有bulletinId，那就把它算上

if(bulletinParams.getBulletinId()!=0){

predicates.add(criteriaBuilder.equal(root.get("bulletinId").as(String.class), bulletinParams.getBulletinId()));

}

// 如果有bulletinTitle，那就把它算上

if(bulletinParams.getBulletinTitle()!=null&&!StringUtils.isEmptyOrWhitespaceOnly(bulletinParams.getBulletinTitle())){

predicates.add(criteriaBuilder.like(root.get("bulletinTitle").as(String.class), "%"+(String)bulletinParams.getBulletinTitle()+"%"));

}

// 如果有bulletinContext，那就把它算上

if(bulletinParams.getBulletinContext()!=null&&!StringUtils.isEmptyOrWhitespaceOnly(bulletinParams.getBulletinContext())){

predicates.add(criteriaBuilder.like(root.get("bulletinContext").as(String.class), "%"+(String)bulletinParams.getBulletinContext()+"%"));

}

// 如果有userId，那就把它算上

if(bulletinParams.getUserId()!=null&&!StringUtils.isEmptyOrWhitespaceOnly(bulletinParams.getUserId())){

predicates.add(criteriaBuilder.equal(root.get("userId").as(String.class), bulletinParams.getUserId()));

}

Predicate[] predicateArray = new Predicate[predicates.size()];

//criteriaQuery.where(predicates.toArray(new Predicate[]));

return criteriaBuilder.and(predicates.toArray(predicateArray));

}

});

return bulletins;

}

2 验证码随机生成程序

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

BufferedImage bfi = new BufferedImage(80, 25, BufferedImage.TYPE\_INT\_RGB);

Graphics g = bfi.getGraphics();

g.fillRect(0, 0, 80, 25);

//验证码字符范围

char[] ch = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789".toCharArray();

Random r = new Random();

int index;

StringBuffer sb = new StringBuffer(); //保存字符串

for (int i = 0; i < 4; i++) {

index = r.nextInt(ch.length);

g.setColor(new Color(r.nextInt(255), r.nextInt(255), r.nextInt(255)));

Font font = new Font("宋体", 30, 20);

g.setFont(font);

g.drawString(ch[index] + "", (i \* 20) + 2, 23);

sb.append(ch[index]);

}

// 添加噪点

int area = (int) (0.02 \* 80 \* 25);

for (int i = 0; i < area; ++i) {

int x = (int) (Math.random() \* 80);

int y = (int) (Math.random() \* 25);

bfi.setRGB(x, y, (int) (Math.random() \* 255));

}

//设置验证码中的干扰线

for (int i = 0; i < 6; i++) {

//随机获取干扰线的起点和终点

int xstart = (int) (Math.random() \* 80);

int ystart = (int) (Math.random() \* 25);

int xend = (int) (Math.random() \* 80);

int yend = (int) (Math.random() \* 25);

g.setColor(interLine(1, 255));

g.drawLine(xstart, ystart, xend, yend);

}

HttpSession session = request.getSession(); //保存到session

session.setAttribute("verificationCode", sb.toString());

ImageIO.write(bfi, "JPG", response.getOutputStream()); //写到输出流

}

private Color interLine(int Low, int High){

if(Low > 255)

Low = 255;

if(High > 255)

High = 255;

if(Low < 0)

Low = 0;

if(High < 0)

High = 0;

int interval = High - Low;

int r = Low + (int)(Math.random() \* interval);

int g = Low + (int)(Math.random() \* interval);

int b = Low + (int)(Math.random() \* interval);

return new Color(r, g, b);

}

3 生成csv格式文件

public File getCsvFile(String allKeyNames, String allValues, String fileName){

File file = new File(fileName);

FileOutputStream fileOutputStream = null;

OutputStreamWriter outputStreamWriter = null;

BufferedWriter bufferedWriter = null;

try {

fileOutputStream = new FileOutputStream(file);

outputStreamWriter = new OutputStreamWriter(fileOutputStream);

bufferedWriter = new BufferedWriter(outputStreamWriter);

// 先输出列名

bufferedWriter.write(allKeyNames+"\r"+allValues);

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}finally {

try {

if (bufferedWriter != null){

bufferedWriter.close();

}

if (outputStreamWriter != null){

outputStreamWriter.close();

}

if (fileOutputStream != null){

fileOutputStream.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

return file;

}

4 JSON转List程序

public List<Bulletin> jsonToBulletinList(String jsonStr) throws ParseException {

JSONArray jsonArray = JSONArray.fromObject(jsonStr);

List<Bulletin> list = new ArrayList<Bulletin>();

Bulletin bulletin = null;

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

JSONObject jsonObject = jsonArray.getJSONObject(i);

bulletin = new Bulletin(jsonObject.getInt("bulletinId"), jsonObject.getString("userId"),

jsonObject.getString("bulletinTitle"), sdf.parse(jsonObject.getString("publishedDate")),

jsonObject.getString("bulletinContext"));

list.add(bulletin);

}

return list;

}

5文件上传腾讯云COS功能

// 线程池大小，建议在客户端与COS网络充足(如使用腾讯云的CVM，同园区上传COS)的情况下，设置成16或32即可, 可较充分的利用网络资源

// 对于使用公网传输且网络带宽质量不高的情况，建议减小该值，避免因网速过慢，造成请求超时。

ExecutorService threadPool = Executors.newFixedThreadPool(32);

// 传入一个 threadpool, 若不传入线程池, 默认 TransferManager 中会生成一个单线程的线程池。

TransferManager transferManager = new TransferManager(CosUtil.getCosClient(), threadPool);

/\*\*

\* @Description: 生成客户端COSClient对象

\* @return

\*/

public static COSClient getCosClient(){

// 1 初始化用户身份信息(secretId, secretKey)

COSCredentials cred = new BasicCOSCredentials(CosConstant.getAPPID(),CosConstant.getSecretId(), CosConstant.getSecretKey());

// 2 设置bucket的区域, COS地域的简称请参照 https://cloud.tencent.com/document/product/436/6224

// clientConfig中包含了设置region, https(默认http), 超时, 代理等set方法, 使用可参见源码或者接口文档FAQ中说明

ClientConfig clientConfig = new ClientConfig(new Region("ap-guangzhou"));

// 3 生成cos客户端

COSClient cosClient = new COSClient(cred, clientConfig);

return cosClient;

}

/\*\*

\* 上传路径中的文件到COS

\* 返回的是外链。

\* 如果上传的文件名重复，会覆盖原文件

\* @param file

\*/

public String uploadFile(final File file) throws InterruptedException {

// 创建新线程

UploadThread uploadThread = new UploadThread();

// 设置要上传的文件路径

uploadThread.setFile(file);

// 新线程上传文件

uploadThread.start();

//将异步执行变成同步执行

uploadThread.join();

// 外链=前缀+文件名

String url = CosConstant.getPreUrl()+ uploadThread.getFileName();

// 文件上传后将临时文件删除

file.delete();

// 返回接收上传文件后获得的外链

return url;

}

6分页功能

/\*\*

\* 将所需页面和page信息传入

\* 目前分页方法是将所有的信息都从数据库中读出，

\* 然后存入page中的list。

\* @param currentPage

\* @param page

\* @return

\*/

@Override

public List<T> getDataListWithPage(int currentPage, Page<T> page) {

List<T> list = null;

// 当前页码的索引

int currentPageIndex = currentPage-1;

// 设置当前页数据开始的索引(相对于每一页的"0")

int startIndex = page.getPageSize()\*(currentPageIndex);

// 设置当前页数据结束的索引(相对于每一页的pageSize-1)

int endIndex = page.getPageSize()\*(currentPageIndex+1);

//获取page中的list

List<T> tempList = page.getList();

// 若当前页是最后一页

// 因为页码索引是从0开始的，

// 所以(总页数-1)才是最后一页的索引

if(currentPageIndex == page.getTotalPages()-1){

// 数据结束索引应该是总条数

endIndex = page.getTotalCount();

}

// 从List中截取出当前页面的数据

list = tempList.subList(startIndex==0?0:startIndex-1,endIndex);

return list;

}