****

**软 件 学 院**

**软件工程实践二报告**

**课题名称： 软件工程实践二**

**专业： 软件工程**

**班级： 卓越161**

**学号： 201619140220**

**学生姓名： 崔烁豪**

**指导老师： 韩玉民**

2019年3月17日

**中原工学院软件学院**

**软件工程实践二课题任务书**

|  |  |  |  |
| --- | --- | --- | --- |
| **姓 名** | 崔烁豪 | **班级** | 卓越161班 |
| **题 目** | 中工软考研帮-网页版 | | |
| **设**  **计**  **任**  **务** | 为了帮助我校打算考研的同学能对学长学姐的考研院校专业选择结果和初试成绩有一个直观的了解，在老师的指导下，我研发了一个基于Web的历届考研成绩等信息展示平台−中工软考研帮。主要功能包括：  1.报考信息展示  主要包括报考院校、报考专业、研究方向、考研年份等。  2.学生基本信息展示  展示报考学生的学号、姓名、性别、所在院校、所在学院、所在班级等。  3.考研成绩展示  主要包括初试总分、我院考生排名、单科成绩等。。  4.学生详细信息展示  主要包括学生的电话、QQ、微信等联系方式和博客、GitHub等技术博客网址。  开发工具：MySQL5.6，Eclipse4.6.3，Hbuilder、Chrome | | |
| **时**  **间**  **进**  **度** | 第1周（2019-02-25~2019-03-03）：对需求进行深入分析，并设计出架构与数据库，仔细理解需求，规划功能。  第2周（2019-03-04~2019-03-10）：学习并完成每一步的开发，在过程中发现问题并改进问题，学习使用前端框架Bootstrap来完成界面的设计，并进行下一步的功能改进，并优化界面。  第3周（2019-03-11~2019-03-17）：搜集并添加真实数据，调试测试，通过实际用户的操作，提出建议并修改，撰写报告。 | | |
| **原 主始 要资 参料 考与 文 献** | [01] 石志国，薛为民，董洁.JSP应用教程.北京：清华大学出版社.北京交通大学出版社，2004  [02] 李曙光.JSP开发实例完全剖析[M].北京：中国电力出版社，2006  [03] 杨易.JSP高级程序设计[M].人民邮电出版社，2006  [04] 姜承尧.MySQL技术内幕.机械工业出版社.2011  [05] 申吉红，廖学峰，余健.JSP课程设计案例精编[M].北京：清华大学出版社，2007  [06] 卢潇.软件工程[M].北京：清华大学出版社；北京交通大学出版社，2005  [07] 黑马程序员.Java EE企业级应用开发教程.北京：人民邮电出版社，2017 | | |

# 摘要

为了方便中工软考研的同学，中工软考研帮把中原工学院软件学院历年学长学姐的报考、分数、详细信息整合到一个网站，帮助有考研意愿的学生在选择考研目标和考研专业的时候，能参考一下历年学长学姐的报考信息。同学们可以根据学长学姐的报考情况，更好的明确自己的择校、报考等事宜。另一方面，当考研同学确定了院校专业后，也可以从这个网站上查找历年考上这个院校专业的学长学姐，通过网站提供的联系方式咨询学长学姐。

本系统采用JAVA、Servlet、JSP开发后台，用MySQL存储数据库，用JavaScript，HTML，CSS，BootStrap框架开发前端。并且录有2019年考研学子报考院校专业的真实信息。

本文共5章，第1章主要介绍了中工软考研帮开发的背景，研究意义及系统功能，第2章是对中工软考研帮的系统分析，第3章是中工软考研帮的数据库详细设计，第4章是对中工软考研帮的具体实现进行阐述，第5章介绍在实践成果完成过程中的一些心得体会，以及对此次实践成果的评价，进行总结。

关键字：中工软；考研帮；报考；信息；网站

至少3个关键字，要能反映题目

# 目录

[摘要 II](#_Toc8330371)

[目录 1](#_Toc8330372)

[第1章 概述 1](#_Toc8330373)

[1.1 课题背景 1](#_Toc8330374)

[1.2 中工软考研帮的功能 1](#_Toc8330375)

[1.3 中工软考研帮的开发意义 2](#_Toc8330376)

[第2章 系统分析 3](#_Toc8330377)

[2.1技术分析 3](#_Toc8330378)

[2.2需求分析 3](#_Toc8330379)

[第3章 数据库分析与设计 4](#_Toc8330380)

[3.1 实体分析 4](#_Toc8330381)

[3.2关系设计 6](#_Toc8330382)

[3.3数据表设计 7](#_Toc8330383)

[第4章 系统实现 9](#_Toc8330384)

[4.1修改学生联系方式流程 9](#_Toc8330385)

[4.2功能实现 11](#_Toc8330386)

[4.2.1学生基本信息的添加、修改和展示 11](#_Toc8330387)

[4.2.2学生联系方式的添加、修改和查看 12](#_Toc8330388)

[4.2.3学生成绩信息的添加、修改和查看 14](#_Toc8330389)

[4.3系统的发布 16](#_Toc8330390)

[第5章 结束语 17](#_Toc8330391)

[指导老师评语 18](#_Toc8330392)

[附录 19](#_Toc8330393)

[1.管理员数据操作的主要代码： 19](#_Toc8330394)

[2.学生数据库操作主要代码： 29](#_Toc8330395)

[3.数据库连接的主要代码： 35](#_Toc8330396)

[4.登录检查过滤器主要代码： 37](#_Toc8330397)

[4.1登录检查过滤器的servlet部分主要代码： 37](#_Toc8330398)

[4.2登录过滤器的XML部分主要代码 39](#_Toc8330399)

[5.添加学生基本信息主要代码： 40](#_Toc8330400)

[5.1添加学生基本信息jsp部分主要代码： 40](#_Toc8330401)

[5.2添加学生基本信息servlet部分的主要代码： 41](#_Toc8330402)

[5.3添加学生基本信息数据库添加部分的主要代码： 43](#_Toc8330403)

[6.主页导航栏主要代码： 44](#_Toc8330404)

# 第1章 概述

本章主要阐述中工软考研帮研发的开发背景、网站功能及开发意义。以下中工软考研帮简称为此网站。

## 1.1 课题背景

有研究调查表明，有七成以上的在校大学生有明确的考研意愿。调查结果显示，75.3%的同学有较为明确的考研意向，仍在考虑的占20.2%。而根据参与调查的学生的各种维度进行分类发现，这种近乎盲目的考研热几乎是不分年级、不分性别、不分学校层次的。而考研是为了什么呢？提高就业竞争力，给自己的学历加码是主要原因。大多数考研人害怕就业受歧视，非全日制研究生接受率不高；考研择校更关注学校整体知名度而非学术水平。

近年来，我校越来越鼓励中工学子考研，而同学们在选择考研后面临的第一个问题就是择校择专业。大多数同学在确定院校专业的时候对所选择的院校专业缺乏准确的、整体的认识。我们知道，学长学姐也曾经经历过这样的抉择。为了帮助中工学子考研选择院校和选择专业，在韩老师的引导下，我做出了这个尚不成熟的网站，即中工软考研帮。

此网站的把历年考生的考研报考信息、初试成绩、考生的联系方式等详细信息整合到一起，供有考研意向的在校学生在选择院校专业的时候，能参考一下历年学长学姐的报考信息，从而能有一个一个更好的选择。同时，在校学生也能通过此网站上提供的详细信息联系到学长学姐，进一步咨询更为详细的考研信息。

## 1.2 中工软考研帮的功能

中工软考研帮的功能为：历年考研学子的院校专业报考信息、初试成绩信息的展示，历年学子联系方式等详细信息的展示。中工软考研帮的使用群体为中工软有考研意愿的在校学生。

## 1.3 中工软考研帮的开发意义

近年来，我校越来越鼓励中工学子考研，而同学们在选择考研后面临的第一个问题就是择校择专业。大多数同学在确定院校专业的时候对所选择的院校专业缺乏准确的、整体的认识。我们知道，学长学姐曾经也经历过这样的抉择。而很久以来，很少有专门的平台是用来展示同学校同专业已经考过研的学长学姐在考研时所选择的院校专业和考试的结果。而此网站的功能，就是展示这些信息。帮助选择考研的在校学生在参考历年学长学姐的报考信息、咨询了学长关于考研的准备过程后，能对考研这件事有一个更清楚的认识，从而更好的考上研究生。

# 第2章 系统分析

本章主要是对中工软考研帮进行合理的分析，通过技术分析、需求分析这两个方面描述这个项目。

## 2.1技术分析

此网站的开发设计采用的是MVC+三层架构。

视图层用到的技术有：JavaScript、HTML、JSP、CSS、BootStrap框架等。

控制器用到的技术有：Servlet。

业务逻辑层用到的技术有：JAVA、Interfaces、Impl等。

数据访问层用到的技术有：JAVA、MySQL、JDBC等。

用到的开发工具有：Eclipse、Tomcat、MySQL、Hbuilder、Chrome等。

## 2.2需求分析

中工软考研帮所需实现的功能模块图如图2.1所示.

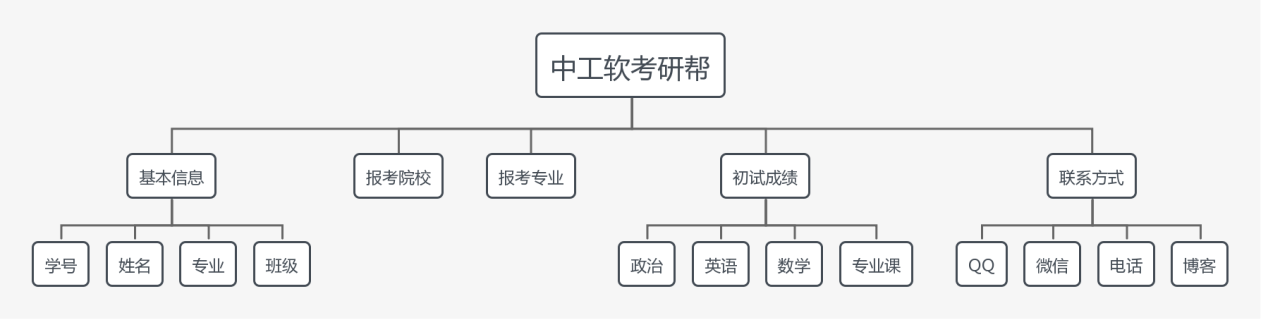


图2.1 系统功能模块图

用户需求如下：

信息展示。首页主要展示历年考研学子的院校专业报考信息、初试成绩信息的展示，详细成绩分布的展示，历年学子联系方式等详细信息的展示。

# 第3章 数据库分析与设计

数据库拥有很强的功能，可以建立一个数据库把我们所需要的数据给存储起来，并可以进行增删改查的操作，还可以对数据库进行导入导出等操作。我们采用MySQL数据库来保存中工软考研帮的各项信息。

## 3.1 实体分析

中工软考研帮涉及的实体包括学生、联系方式、成绩、管理员等。经过分析，学生实体E-R图如图3.1所示，联系方式实体E-R图如图3.2所示，成绩实体E-R图如图3.3所示，管理员实体E-R图如图3.4所示。



图3.1 学生实体E-R图



图3.2 联系方式实体E-R图



图3.3 成绩实体E-R图



图3.4 管理员实体E-R图

## 3.2关系设计

这里有三个实体，分别是：学生、联系方式、成绩、管理员。它们之间的关系是：成绩属于学生，一对一。联系方式属于学生，一对一。管理员管理学生，一对多。管理员管理联系方式，一对多。管理员管理成绩，一对多。中工软考研帮的系统E-R图如图3.5所示。



图3.5 系统总体E-R图

## 3.3数据表设计

中工软考研帮的数据库设计共有四个表：学生表、联系方式表、成绩表、管理员表。学生表如表3.1所示，联系方式表如表3.2所示、成绩表如表3.3所示、管理员表如表3.4所示。

1.学生表。记录学生的基本信息，包括学生的学号、姓名、学校、学院、专业、班级、报考院校、报考专业、初试成绩等。

表3.1 student数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| id | bigint | 12 | 否 | 是 |
| name | varchar | 5 | 是 | 否 |
| sch | varchar | 20 | 是 | 否 |
| college | varchar | 20 | 是 | 否 |
| major | varchar | 20 | 是 | 否 |
| class | varchar | 20 | 是 | 否 |
| register\_sch | varchar | 20 | 是 | 否 |
| register\_major | varchar | 20 | 是 | 否 |
| register\_num | varchar | 15 | 是 | 否 |
| score\_total | int | 3 | 是 | 否 |

2.联系方式表。记录学生的联系方式，包括学号、姓名、性别、电话、QQ、微信、博客地址、GitHub地址等。

表3.2 info数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| id | bigint | 12 | 否 | 是 |
| name | varchar | 5 | 是 | 否 |
| sex | varchar | 2 | 是 | 否 |
| tel | varchar | 11 | 是 | 否 |
| qq | varchar | 11 | 是 | 否 |
| wechat | varchar | 50 | 是 | 否 |
| blog | varchar | 50 | 是 | 否 |
| github | varchar | 50 | 是 | 否 |

3.成绩表，记录了学生的考研初试成绩信息。包括学号、姓名、各科单科分、考研年份等。

表3.3 score数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| id | bigint | 12 | 否 | 是 |
| name | varchar | 5 | 是 | 否 |
| politics\_score | int | 3 | 是 | 否 |
| english\_subject | varchar | 3 | 是 | 否 |
| english\_score | int | 3 | 是 | 否 |
| math\_subject | varchar | 5 | 是 | 否 |
| math\_score | int | 3 | 是 | 否 |
| major\_subject | varchar | 20 | 是 | 否 |
| major\_score | int | 3 | 是 | 否 |
| Score\_year | int | 4 | 是 | 否 |

4.管理员表。记录管理员信息。包括管理员的编号、姓名、密码、邮箱等。

表3.4 manager数据表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **列名** | **数据类型** | **长度** | **允许空** | **是否为主键** |
| id | bigint | 12 | 否 | 是 |
| name | varchar | 5 | 是 | 否 |
| pwd | varchar | 30 | 是 | 否 |
| email | varchar | 30 | 是 | 否 |

# 第4章 系统实现

软件设计是一个把软件需求变换成软件表示的过程。最初这种表示知识描绘出软件的总的框架,然后进一步细化，把它加工成在程序细节上非常接近于源程序的软件表示。从工程管理的角度来看，软件设计分两步完成。首先做概要设计，将软件需求转化成数据结构和软件的系统结构。然后是详细设计，即过程设计。通过对结构表示进行细化，得到软件的详细的数据结构和算法。

中工软考研帮设计阶段主要目的是将系统分析阶段所提出的反映了用户信息需求的系统逻辑方案转换成可以实施的基于计算机与通信系统的技术方案。

## 4.1修改学生联系方式流程

修改学生联系方式流程如图4.1所示。



图4.1 修改学生联系方式流程

## 4.2功能实现

系统的用户界面作为人机接口起着越来越重要的作用。它的好坏直接影响到系统的使用性。具有友好界面的系统，对于用户来说，无疑是一种享受。

本系统的各模块功能实现过程如下所示。

### 4.2.1学生基本信息的添加、修改和展示

管理员添加的学生信息存储到数据库的学生表。管理员修改学生信息时，鼠标右键学生基本信息即进入学生信息修改页面，修改成功后，数据库中学生表对于的信息也会修改。学生基本信息的添加修改只有管理员能操作。

由于学生数据较多，学生基本信息的展示采用分页和不同年份分别展示的方法展示数据。每页展示最多15条学生基本信息数据，剩余的数据在下一页展示。用户可以根据考研年份查看历年的考研报考数据。普通用户和管理员都可以查看学生基本信息。

学生基本信息添加页面如图4.2所示，学生基本信息修改页面如图4.3所示，学生基本信息展示页面如图4.4所示。



图4.2 学生基本信息添加页面



图4.3 学生基本信息修改页面



图4.4 学生基本信息展示页面

### 4.2.2学生联系方式的添加、修改和查看

在学生基本信息已经存在的基础上，管理员可以添加和修改学生的联系方式信息。如果学生没有联系方式数据，管理员鼠标右键学生基本信息展示页面中的学生姓名即进入学生联系方式添加页面。如果学生有联系方式数据，管理员鼠标右键学生基本信息展示页面中的学生姓名即进入学生联系方式修改页面。学生联系方式的添加修改只有管理员能操作。

点击学生基本信息展示页面中的学生姓名可以查看学生的联系方式信息。普通用户和管理员都可以查看学生的联系方式信息。

学生联系方式的添加页面如图4.5所示。学生联系方式的修改页面如图4.6所示。学生联系方式的查看页面如图4.7所示。



图4.5 学生联系方式添加页面



图4.6 学生联系方式修改页面



图4.7 学生联系方式查看页面

### 4.2.3学生成绩信息的添加、修改和查看

在学生基本信息已经存在的基础上，管理员可以添加和修改学生的成绩信息。如果学生没有成绩信息数据，管理员鼠标右键学生基本信息展示页面中的初试分数即进入学生成绩信息添加页面。如果学生有成绩信息数据，管理员鼠标右键学生基本信息展示页面中的初试成绩即进入学生成绩信息修改页面。学生成绩信息的添加修改只有管理员能操作。

点击学生基本信息展示页面中的初试成绩可以查看学生的成绩信息。普通用户和管理员都可以查看学生的成绩信息。

学生成绩信息的添加页面如图4.8所示。学生成绩信息的修改页面如图4.9所示。学生成绩信息的查看页面如图4.10所示。



图4.8 学生成绩信息添加页面



图4.9 学生成绩信息修改页面



图4.10 学生成绩信息查看页面

## 4.3系统的发布

中工软考研帮的发布，就是将整个项目部署到服务器上的过程。在JavaWeb项目已经写好的前提下，把项目导出到WAR包，把WAR包放到Tomcat的Webapps目录下，再打开Tomcat服务，系统就发布了。访问这个已经部署好的项目，只需要在浏览器中输入URL即可。

# 第5章 结束语

20世纪最重要的资源是石油，而21世纪这一状况已经改变了。Google 、脸书、亚马逊掌握着每一个人的信息、行踪、喜好，海量的数据带来巨大的财富和权利。信息时代的竞争已经完全改变，反垄断需要新的标准和紧跟时代的行动。数据经济中谁掌握了数据谁就掌握了市场。

在中工软考研帮的开发设计中，不仅要有易操作、友好的操作界面，还要有真实可靠的数据。在友好的操作界面设计上，我采用了前端框架BootStrap，增强了页面的交互性。我还大量使用JS设计悬浮事件、鼠标右键事件，并且把JS完成的鼠标右键事件与数据库操作、页面跳转等操作联系起来，完成了许多不错的功能。我把JS完成的悬浮事件与管理员操作时一些温馨的提醒结合起来，能帮助管理员更快的学会如何进行学生的添加修改操作，很大程度上提高了用户友好性。我大量使用CSS，把文字、表格的大小和布局更加可视化、符合人们的美观。我还用CSS完成了鼠标悬停事件，在鼠标悬停到表格的某一行时，改变这一行表格的颜色，鼠标从这一行移走时，再恢复它的颜色。在数据获取上，学长学姐的初试成绩从学院公布的2019年考研光荣榜上采集，学长学姐的报考院校专业从2019年考研报考确认缴费名单上采集，学长学姐的学号信息从学院公布的学期学业进展表上采集。虽然数据并不全面，但是绝对真实可靠。

这次的实践也让我认识到，开发设计需要充足的需求分析、网站细节设计、数据库设计等。虽然不是团队开发，但是仍然要随时记录下每一个开发过程中遇到的问题和解决的问题。还要实时跟踪项目进程，对接下来的项目规划有一个持续的清晰认识。遇到逻辑上、设计上的问题时，要善于利用画图工具，画下系统功能逻辑图、技术实现的过程图。整个项目的逻辑要梳理清楚，多做日志记录。然后在后期实际开发中，根据前期设计按部就班的推进开发进度。这样，不仅开发更高效，也能避免在后期走一些不必要的弯路。

非常感谢韩老师的指导帮助。

# 指导老师评语

成绩：

指导老师签字：

年 月 日

# 附录

## 1.管理员数据操作的主要代码：

/\*\*

\* Types

\* **@author** dandelion

\* **@time** 2019年3月5日下午2:47:12

\* **@作用** 实现管理员登录、添加学生、修改学生等功能

\* **@调用**

\* **@返回值类型**

\*/

**public** **class** ManagerDaoImpl **implements** ManagerDao{

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月5日下午2:48:28

\* **@方法名** login

\* **@作用** 实现管理员登录功能

\* **@调用**

\* **@返回值类型** String

\*/

@SuppressWarnings("finally")

**public** String login(String id,String pwd){

Connection conn = **null**;

PreparedStatement ps =**null**;

ResultSet rs=**null**;

**boolean** flagCheckId=**false**;

**boolean** flagCheckPwd=**false**;

String str="";

**try**{

conn=DBConn.*getConnection*();

String sql="select pwd from manager where id=?";

ps=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

ps.setString(1, id);

rs = ps.executeQuery();

**if**(rs.next()){

flagCheckId=**true**;

flagCheckPwd=rs.getString(1).equals(pwd);

}

}**catch**(Exception e){

System.***out***.println("查询数据时发生异常【ManagerDaoImpl.java\\login】");

e.printStackTrace();

}**finally**{

**if**(flagCheckId == **false**){

str = "无此账号";

}**else** **if**(flagCheckPwd == **false**){

str = "密码错误";

}**else**{

str="登录成功";

}

**return** str;

//注意，此处不能close,否则返回值无法传递

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月7日下午9:06:36

\* **@方法名** getManagerById

\* **@作用** 根据编号返回数据表manager内容

\* **@调用**

\* **@返回值类型** ResultSet

\*/

**public** ResultSet getManagerById(String id){

Connection conn = **null**;

PreparedStatement ps =**null**;

ResultSet rs=**null**;

**try**{

conn=DBConn.*getConnection*();

String sql="select \* from manager where id=?";

ps=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

ps.setString(1, id);

rs = ps.executeQuery();

**return** rs;

}**catch**(Exception e){

System.***out***.println("查询数据时发生异常【ManagerDaoImpl.java\\getManagerById】");

e.printStackTrace();

**return** **null**;

}**finally**{

//注意，此处不能close,否则返回值无法传递

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月12日下午4:56:49

\* **@方法名** addStudent

\* **@作用** 添加学生基本信息

\* **@调用** AddStudentService.java

\* **@返回值类型** boolean，true表示插入成功，false表示插入失败

\*/

**public** **boolean** addStudent(Student student){

Connection conn = **null**; //声明Connection对象

PreparedStatement perstat =**null**; //声明PreparedStatement对象

**int** rows=0;

**try**{

conn=DBConn.*getConnection*();

String sql="insert into student(id,name,sch,college,major,class,register\_sch,register\_major,register\_num,score\_total,score\_year) values(?,?,?,?,?,?,?,?,?,?,?)";

perstat=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

perstat.setLong(1,student.getId());

perstat.setString(2,student.getName());

perstat.setString(3,student.getSch());

perstat.setString(4,student.getCollege());

perstat.setString(5,student.getMajor());

perstat.setString(6,student.getClazz());

perstat.setString(7,student.getRegister\_sch());

perstat.setString(8,student.getRegister\_major());

perstat.setString(9,student.getRegister\_num());

perstat.setInt(10,student.getScore\_total());

perstat.setString(11,student.getScore\_year());

rows=perstat.executeUpdate();//rows接收更新结果,rows大于0表示插入成功

}**catch**(Exception e){

System.***out***.println("【ManagerDaoImpl.java\\addStudent】插入基本信息数据时发生异常");

e.printStackTrace();

}**finally**{

DBConn.*close*(perstat);

DBConn.*close*(conn);

}

**if**(rows>0){

**return** **true**;

}**else**{

**return** **false**;

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月12日下午4:59:00

\* **@方法名** addInfo

\* **@作用** 添加学生详细信息

\* **@调用** AddStudentService.java

\* **@返回值类型** boolean，true表示插入成功，false表示插入失败

\*/

**public** **boolean** addInfo(Info info){

Connection conn = **null**; //声明Connection对象

PreparedStatement perstat =**null**; //声明PreparedStatement对象

//用i代替123456，当更换次序时，修改更方法，解耦

**int** rows=0,i=1;

**try**{

conn=DBConn.*getConnection*();

String sql="insert into info(id,name,sex,tel,qq,wechat,blog,github) values(?,?,?,?,?,?,?,?)";

perstat=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

perstat.setLong(i,info.getId());

i+=1;

perstat.setString(i,info.getName());

i+=1;

perstat.setString(i,info.getSex());

i+=1;

perstat.setString(i,info.getTel());

i+=1;

perstat.setString(i,info.getQq());

i+=1;

perstat.setString(i,info.getWechat());

i+=1;

perstat.setString(i,info.getBlog());

i+=1;

perstat.setString(i,info.getGithub());

rows=perstat.executeUpdate();//rows接收更新结果,rows大于0表示插入成功

}**catch**(Exception e){

System.***out***.println("【ManagerDaoImpl.java\\addInfo】插入详细信息数据时发生异常");

e.printStackTrace();

}**finally**{

DBConn.*close*(perstat);

DBConn.*close*(conn);

}

**if**(rows>0){

**return** **true**;

}**else**{

**return** **false**;

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月12日下午4:59:20

\* **@方法名** addScore

\* **@作用** 添加学生成绩信息

\* **@调用** AddStudentService.java

\* **@返回值类型** boolean，true表示插入成功，false表示插入失败

\*/

**public** **boolean** addScore(Score score){

Connection conn = **null**; //声明Connection对象

PreparedStatement perstat =**null**; //声明PreparedStatement对象

**int** rows=0;

**try**{

conn=DBConn.*getConnection*();

String sql="insert into score(id,name,politics\_score,english\_subject,english\_score,math\_subject,math\_score,major\_subject,major\_score,score\_year) values(?,?,?,?,?,?,?,?,?,?)";

perstat=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

perstat.setLong(1,score.getId());

perstat.setString(2,score.getName());

perstat.setString(3,score.getPolitics\_score());

perstat.setString(4,score.getEnglish\_subject());

perstat.setString(5,score.getEnglish\_score());

perstat.setString(6,score.getMath\_subject());

perstat.setString(7,score.getMajor\_score());

perstat.setString(8,score.getMajor\_subject());

perstat.setString(9,score.getMajor\_score());

perstat.setString(10,score.getScore\_year());

rows=perstat.executeUpdate();//rows接收更新结果,rows大于0表示插入成功

}**catch**(Exception e){

System.***out***.println("【ManagerDaoImpl.java\\addScore】插入成绩信息数据时发生异常");

e.printStackTrace();

}**finally**{

DBConn.*close*(perstat);

DBConn.*close*(conn);

}

**if**(rows>0){

**return** **true**;

}**else**{

**return** **false**;

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月15日上午10:02:16

\* **@方法名** modifyStudent

\* **@作用** 修改学生基本信息

\* **@调用** ModifyStudentService

\* **@返回值类型** boolean 返回true表示修改成功，返回false表示返回失败

\*/

**public** **boolean** modifyStudent(Student student){

Connection conn = **null**; //声明Connection对象

PreparedStatement perstat =**null**; //声明PreparedStatement对象

//用i代替123456，当更换次序时，修改更方法，解耦

**int** rows=0,i=1;

**try**{

conn=DBConn.*getConnection*();

String sql="update student set name=?,sch=?,college=?,major=?,class=?,register\_sch=?,register\_major=?,register\_num=?,score\_total=?,score\_year=? where id=?";

perstat=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

perstat.setString(i,student.getName());

i+=1;

perstat.setString(i,student.getSch());

i+=1;

perstat.setString(i,student.getCollege());

i+=1;

perstat.setString(i,student.getMajor());

i+=1;

perstat.setString(i,student.getClazz());

i+=1;

perstat.setString(i,student.getRegister\_sch());

i+=1;

perstat.setString(i,student.getRegister\_major());

i+=1;

perstat.setString(i,student.getRegister\_num());

i+=1;

perstat.setInt(i,student.getScore\_total());

i+=1;

perstat.setString(i,student.getScore\_year());

i+=1;

perstat.setLong(i,student.getId());

rows=perstat.executeUpdate();//rows接收更新结果,rows大于0表示插入成功

}**catch**(Exception e){

System.***out***.println("【ManagerDaoImpl.java\\modifyStudent】修改基本信息数据时发生异常");

e.printStackTrace();

}**finally**{

DBConn.*close*(perstat);

DBConn.*close*(conn);

}

**if**(rows>0){

**return** **true**;

}**else**{

**return** **false**;

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月15日下午2:05:02

\* **@方法名** modifyInfo

\* **@作用** 修改学生详细信息

\* **@调用** ModifyStudentService

\* **@返回值类型** boolean 返回true表示修改成功，返回false表示返回失败

\*/

**public** **boolean** modifyInfo(Info info){

Connection conn = **null**; //声明Connection对象

PreparedStatement perstat =**null**; //声明PreparedStatement对象

//用i代替123456，当更换次序时，修改更方法，解耦

**int** rows=0,i=1;

**try**{

conn=DBConn.*getConnection*();

String sql="update Info set name=?,sex=?,tel=?,qq=?,wechat=?,blog=?,github=? where id=?";

perstat=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

perstat.setString(i,info.getName());

i+=1;

perstat.setString(i,info.getSex());

i+=1;

perstat.setString(i,info.getTel());

i+=1;

perstat.setString(i,info.getQq());

i+=1;

perstat.setString(i,info.getWechat());

i+=1;

perstat.setString(i,info.getBlog());

i+=1;

perstat.setString(i,info.getGithub());

i+=1;

perstat.setLong(i,info.getId());

rows=perstat.executeUpdate();//rows接收更新结果,rows大于0表示插入成功

}**catch**(Exception e){

System.***out***.println("【ManagerDaoImpl.java\\modifyInfo】修改详细信息数据时发生异常");

e.printStackTrace();

}**finally**{

DBConn.*close*(perstat);

DBConn.*close*(conn);

}

**if**(rows>0){

**return** **true**;

}**else**{

**return** **false**;

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月15日下午2:25:35

\* **@方法名** modifyScore

\* **@作用** 修改学生成绩信息

\* **@调用** ModifyStudentService

\* **@返回值类型** boolean 返回true表示修改成功，返回false表示返回失败

\*/

**public** **boolean** modifyScore(Score score){

Connection conn = **null**; //声明Connection对象

PreparedStatement perstat =**null**; //声明PreparedStatement对象

//用i代替123456，当更换次序时，修改更方法，解耦

**int** rows=0,i=1;

**try**{

conn=DBConn.*getConnection*();

String sql="update score set name=?,politics\_score=?,english\_subject=?,english\_score=?,math\_subject=?,math\_score=?,major\_subject=?,major\_score=?,score\_year=? where id=?";

perstat=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

perstat.setString(i,score.getName());

i+=1;

perstat.setString(i,score.getPolitics\_score());

i+=1;

perstat.setString(i,score.getEnglish\_subject());

i+=1;

perstat.setString(i,score.getEnglish\_score());

i+=1;

perstat.setString(i,score.getMath\_subject());

i+=1;

perstat.setString(i,score.getMajor\_score());

i+=1;

perstat.setString(i,score.getMajor\_subject());

i+=1;

perstat.setString(i,score.getMajor\_score());

i+=1;

perstat.setString(i,score.getScore\_year());

i+=1;

perstat.setLong(i,score.getId());

rows=perstat.executeUpdate();//rows接收更新结果,rows大于0表示插入成功

}**catch**(Exception e){

System.***out***.println("【ManagerDaoImpl.java\\modifyScore】修改成绩信息数据时发生异常");

e.printStackTrace();

}**finally**{

DBConn.*close*(perstat);

DBConn.*close*(conn);

}

**if**(rows>0){

**return** **true**;

}**else**{

**return** **false**;

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@方法名** updateNamebyId

\* **@作用** 根据id更新三个表里的name数据

\* **@调用** 修改学生基本信息的servlet：ModifyStudentService.java

\* **@返回值类型**

\*/

**public** **boolean** updateNamebyId(Long id,String name){

Connection conn = **null**; //声明Connection对象

PreparedStatement perstat =**null**; //声明PreparedStatement对象

//用i代替123456，当更换次序时，修改更方法，解耦

**int** rows=0,i=1;

**try**{

conn=DBConn.*getConnection*();

String sql="update student, info ,score "

+ "set student.`name`=?,info.`name`=?,score.`name`=?"

+ "where student.id=? and info.id=? and score.id=?";

perstat=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

perstat.setString(i,name);

i+=1;

perstat.setString(i,name);

i+=1;

perstat.setString(i,name);

i+=1;

perstat.setLong(i,id);

i+=1;

perstat.setLong(i,id);

i+=1;

perstat.setLong(i,id);

rows=perstat.executeUpdate();//rows接收更新结果,rows大于0表示插入成功

}**catch**(Exception e){

System.***out***.println("【ManagerDaoImpl.java\\updateNamebyId】根据id修改三表姓名时发生异常");

e.printStackTrace();

}**finally**{

DBConn.*close*(perstat);

DBConn.*close*(conn);

}

**if**(rows>0){

**return** **true**;

}**else**{

**return** **false**;

}

}

}

## 2.学生数据库操作主要代码：

/\*\*

\* Types

\* **@author** dandelion

\* **@time** 2019年3月12日下午9:27:38

\* **@方法名** StudentDao

\* **@作用** 不需要管理员权限就可以进行的操作

\* 1.mainPage的数据获取，以及学生详细数据、成绩数据的获取

\* **@调用**

\* **@返回值类型**

\*/

**public** **class** StudentDaoImpl **implements** StudentDao{

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月2日下午2:09:52

\* **@方法名** getAllStudent

\* **@作用** 根据考研年份返回数据表student的所有内容

\* **@调用** mainPage.jsp

\* **@返回值类型** ResultSet

\* rs.getString(1)---学号

\* rs.getString(2)---姓名

\* rs.getString(3)---学校

\* rs.getString(4)---学院

\* rs.getString(5)---专业

\* rs.getString(6)---班级

\* rs.getString(7)---报考院校

\* rs.getString(8)---报考专业

\* rs.getString(9)---准考证号

\* rs.getString(10)---总分

\* rs.getSting(11)---考研年份

\*/

**public** ResultSet getAllStudent(){

Connection conn = **null**;

Statement stat =**null**;

ResultSet rs=**null**;

**try**{

conn=DBConn.*getConnection*();

String sql="select \* from student order by score\_total desc";

stat=conn.createStatement();//执行SQL语句

rs = stat.executeQuery(sql);

**return** rs;

}**catch**(Exception e){

System.***out***.println("查询数据时发生异常【StudentDao.java\\getAllStudent】");

e.printStackTrace();

**return** **null**;

}**finally**{

//注意，此处不能close,否则返回值无法传递

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月4日下午9:44:15

\* **@方法名** getStudentById

\* **@作用** 根据学号返回数据表student内容

\* **@调用** infoPage.jsp

\* **@返回值类型** ResultSet

\* rs.getString(1)---学号

\* rs.getString(2)---姓名

\* rs.getString(3)---学校

\* rs.getString(4)---学院

\* rs.getString(5)---专业

\* rs.getString(6)---班级

\* rs.getString(7)---报考院校

\* rs.getString(8)---报考专业

\* rs.getString(9)---准考证号

\* rs.getInt(10)---总分

\* rs.getSting(11)---考研年份

\*/

**public** ResultSet getStudentById(String id){

Connection conn = **null**;

PreparedStatement ps =**null**;

ResultSet rs=**null**;

**try**{

conn=DBConn.*getConnection*();

String sql="select \* from student where id=?";

ps=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

ps.setString(1, id);

rs = ps.executeQuery();

**return** rs;

}**catch**(Exception e){

System.***out***.println("查询数据时发生异常【StudentDao.java\\getStudentById】");

e.printStackTrace();

**return** **null**;

}**finally**{

//注意，此处不能close,否则返回值无法传递

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月4日下午1:05:05

\* **@方法名** getInfoById

\* **@作用** 根据学号返回数据表info内容

\* **@调用** infoPage.jsp

\* **@返回值类型** ResultSet

\* rs.getString(1)---学号

\* rs.getString(2)---姓名

\* rs.getString(3)---性别

\* rs.getString(4)---电话

\* rs.getString(5)---QQ

\* rs.getString(6)---微信

\* rs.getString(7)---博客

\* rs.getString(8)---GitHub

\*/

**public** ResultSet getInfoById(String id){

Connection conn = **null**;

PreparedStatement ps =**null**;

ResultSet rs=**null**;

**try**{

conn=DBConn.*getConnection*();

String sql="select \* from info where id=?";

ps=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

ps.setString(1, id);

rs = ps.executeQuery();

**return** rs;

}**catch**(Exception e){

System.***out***.println("查询数据时发生异常【StudentDao.java\\getInfoById】");

e.printStackTrace();

**return** **null**;

}**finally**{

//注意，此处不能close,否则返回值无法传递

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月4日下午8:48:44

\* **@方法名** getScoreById

\* **@作用** 根据学号返回数据表score内容

\* **@调用** scorePage.jsp

\* **@返回值类型** ResultSet

\* rs.getString(1)---学号

\* rs.getString(2)---姓名

\* rs.getString(3)---政治成绩

\* rs.getString(4)---英语备注

\* rs.getString(5)---英语成绩

\* rs.getString(6)---数学备注

\* rs.getString(7)---数学成绩

\* rs.getString(8)---专业课备注

\* rs.getString(9)---专业课成绩

\* rs.getString(10)---参加考研年份

\*/

**public** ResultSet getScoreById(String id){

Connection conn = **null**;

PreparedStatement ps =**null**;

ResultSet rs=**null**;

**try**{

conn=DBConn.*getConnection*();

String sql="select \* from score where id=?";

ps=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

ps.setString(1, id);

rs = ps.executeQuery();

**return** rs;

}**catch**(Exception e){

System.***out***.println("查询数据时发生异常【StudentDao.java\\getScoreById】");

e.printStackTrace();

**return** **null**;

}**finally**{

//注意，此处不能close,否则返回值无法传递

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@方法名** getStudentByYear

\* **@作用** 根据考研年份获取学生信息

\* **@返回值类型** ResultSet

\*/

@Override

**public** ResultSet getStudentByYear(String score\_year) {

Connection conn = **null**;

PreparedStatement ps =**null**;

ResultSet rs=**null**;

**try**{

conn=DBConn.*getConnection*();

String sql="select \* from student where score\_year=? order by score\_total desc";

ps=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

ps.setString(1, score\_year);

rs = ps.executeQuery();

**return** rs;

}**catch**(Exception e){

System.***out***.println("查询数据时发生异常【StudentDao.java\\getStudentByYear】");

e.printStackTrace();

**return** **null**;

}**finally**{

//注意，此处不能close,否则返回值无法传递

}

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@方法名** getStudentLengthByYear

\* **@作用** 获取某个考研年份的学生的数据量

\*/

@Override

**public** **int** getStudentCountByYear(String score\_year){

**int** length = 0;

ResultSet rs = getStudentByYear(score\_year);

**try** {

//移动到最后

rs.last();

//getRow:得到光标当前所指定的行号

//获取结果集长度

length = rs.getRow();

} **catch** (SQLException e) {

System.***out***.println("查询数据时发生异常【StudentDao.java\\getStudentLengthByYear】");

e.printStackTrace();

}

**return** length;

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@方法名** getStudentByYearPage

\* **@作用** 根据考研年份获取学生信息，但是数据分页，根据page和page的长度分成好几个页

\* pageNow：希望显示第几页

\* pageSize：每页显示多少条记录，暂时准备设定为15

\* pageCount:一共有多少页

\* **@返回值类型** List<Student>

\*/

@Override

**public** List<Student> getStudentByYearPage(String score\_year, **int** pageNow, **int** pageSize) {

ResultSet rs = getStudentByYear(score\_year);

//定义listStudent为要返回的List，内容是第pageNow页的学生报考基本数据

List<Student> listStudent = **new** ArrayList<Student>();

Student student = **new** Student();

//i记录的是第i条数据

//第pageNow页

**int** i=0;

**try** {

**while**(rs.next()){

**if**(i/pageSize+1!=pageNow){

i++;

**continue**;

}

student.setId(Long.*parseLong*(rs.getString(1)));

student.setName(rs.getString(2));

student.setClazz(rs.getString(6));

student.setRegister\_sch(rs.getString(7));

student.setRegister\_major(rs.getString(8));

student.setRegister\_num(rs.getString(9));

student.setScore\_total(Integer.*parseInt*(rs.getString(10)));

listStudent.add(student);

student = **new** Student();

i++;

}

} **catch** (SQLException e){

e.printStackTrace();

}

**return** listStudent;

}

/\*\*

\* Methods

\* **@author** dandelion

\* **@方法名** getPageCountByYear

\* **@作用** 根据考研年份获取一共有多少页数据

\* **@返回值类型** int

\*/

@Override

**public** **int** getPageCountByYear(String score\_year, **int** pageSize) {

**int** pageCount =0;

**int** studentCount = getStudentCountByYear(score\_year);

pageCount = studentCount/pageSize;

**if**(studentCount%pageSize>0){

pageCount++;

}

**return** pageCount;

}

}

## 3.数据库连接的主要代码：

/\*\*

\* Types

\* **@author** dandelion

\* **@time** 2019年3月2日下午2:37:28

\* **@作用** 连接数据库的工具类

\* **@调用**

\* StudentDao

\* **@返回值类型**： Connection

\*/

**public** **class** DBConn {

**private** **static** String *dbUrl*="jdbc:mysql://localhost:3306/rbshengxue";

**private** **static** String *dbUser*="root";

**private** **static** String *dbPassword*="root";

**private** **static** String *jdbcName*="com.mysql.jdbc.Driver";

//连接数据库

**public** **static** Connection getConnection(){

Connection conn = **null**;

**try**{

Class.*forName*(*jdbcName*);

}

**catch**(Exception e){}

**try**{

conn=DriverManager.*getConnection*(*dbUrl*,*dbUser*,*dbPassword*);

}

**catch**(SQLException ex){

System.***out***.println("DBConn.java\\getConnection");

ex.printStackTrace();

}

System.***out***.println("连接成功【db/DBConn.java】");

**return** conn;

}

// 关闭连接数据库资源

**public** **static** **void** close(Connection conn) {

**if** (conn != **null**) { //判断Connection对象是否为空

**try** {

conn.close(); // 关闭连接数据库资源

} **catch** (SQLException e){ //判断关闭Connection对象时是否发生异常

System.***out***.println("关闭数据库连接发生异常");

}

}

}

//关闭操作数据库资源

**public** **static** **void** close(Statement stmt) {

**if** (stmt != **null**){ //判断Statement对象是否为空

**try** {

stmt.close(); //关闭操作数据库资源

} **catch** (SQLException e){ //判断关闭Statement对象时是否发生异常

JOptionPane.*showMessageDialog*(**null**,"关闭数据库操作资源发生异常！！！","☆★提示信息☆★",

JOptionPane.***INFORMATION\_MESSAGE***); //提示关闭数据库操作资源发生异常

}

}

}

//关闭结果集

**public** **static** **void** close(ResultSet rs) {

**if** (rs != **null**) { //判断结果集是否为空

**try** {

rs.close(); //关闭结果集

} **catch** (SQLException e){ //判断结果集是否发生异常

JOptionPane.*showMessageDialog*(**null**,"关闭结果集发生异常！！！","☆★提示信息☆★",

JOptionPane.***INFORMATION\_MESSAGE***); //提示关闭结果集发生异常

}

}

}

}

## 4.登录检查过滤器主要代码：

### 4.1登录检查过滤器的servlet部分主要代码：

/\*\*

\* Types

\* **@author** dandelion

\* **@time** 2019年3月16日下午8:18:29

\* **@类名** FilterService

\* **@作用** 登录检查过滤器，

\* 所有需要管理员权限的页面，访问时都要过一下这个过滤器

\* 管理员已登录，正常访问

\* 管理员未登录，跳转到登录页面

\* **@调用**

\* **@返回值类型**

\*/

**public** **class** FilterService **implements** Filter {

**public** FilterService() {

}

//服务器销毁时运行

**public** **void** destroy() {

System.***out***.println("过滤器销毁了【service/FilterService.java】...");

}

//范围内的HTML或jsp运行时运行【jsp运行一次，这个运行一次】

**public** **void** doFilter(ServletRequest srequest,

ServletResponse sresponse, FilterChain chain)

**throws** IOException, ServletException {

HttpServletRequest request = (HttpServletRequest)srequest;

HttpServletResponse response = (HttpServletResponse)sresponse;

//chain的作用是给下一个过滤器放行

chain.doFilter(srequest, sresponse);

System.***out***.println("我是过滤器【service/FilterService.java】...");

//设置编码、防治乱码

request.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charset=UTF-8");

//用URL接收需要进行cookieCheck登录验证的jsp的url

//URL表示最终会跳转的页面，也可能是loginPage.jsp

//-------------cookie与数据库比对的开始--------------------

Cookie[] cs = request.getCookies();

ManagerDao md = **new** ManagerDaoImpl();

//checkRes接收登录验证的结果，并返回给原网页

String checkRes = "登录测试";

**boolean** havaLogin = **false**;

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

/\* 如果找到管理员的cookie，且账号密码正确，则跳转到管理员页面 \*/

**for**(Cookie c:cs){

String id = c.getName();

String pwd = c.getValue();

//checkRes可能接收的值："无此账号"、"密码错误"、"登录成功"

checkRes = md.login(id, pwd);

/\*

\* 如果管理员已登录:

\* 1.havaLogin标记为已登录true，

\* 2.重定向传参【登录验证的结果checkRes】

\*/

**if**(checkRes.equals("登录成功")){

havaLogin = **true**;

**break**;

}

}

/\* 如果管理员未登录，跳转到登录页面 \*/

**if**(!havaLogin){

response.sendRedirect("../JSP/mainPage.jsp");

}

}

}

//服务器开启时运行

**public** **void** init(FilterConfig fConfig)

**throws** ServletException {

System.***out***.println("过滤器创建了【service/FilterService.java】...");

}

}

### 4.2登录过滤器的XML部分主要代码

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:web=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>

<filter>

<filter-name>FilterService</filter-name>

<filter-class>service.FilterService</filter-class>

</filter>

<filter-mapping>

<filter-name>FilterService</filter-name>

<url-pattern>/JSP/modifyPage.jsp</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>FilterService</filter-name>

<url-pattern>/JSP/addStudentPage.jsp</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>FilterService</filter-name>

<url-pattern>/JSP/addInfoPage.jsp</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>FilterService</filter-name>

<url-pattern>/JSP/addScorePage.jsp</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>FilterService</filter-name>

<url-pattern>/JSP/modifyStudentPage.jsp</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>FilterService</filter-name>

<url-pattern>/JSP/modifyInfoPage.jsp</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>FilterService</filter-name>

<url-pattern>/JSP/modifyScorePage.jsp</url-pattern>

</filter-mapping>

</web-app>

## 5.添加学生基本信息主要代码：

### 5.1添加学生基本信息jsp部分主要代码：

<%@page import=*"dao.impl.ManagerDaoImpl"*%>

<%@ page language=*"java"*

import=*"dao.ManagerDao,java.sql.ResultSet"*

contentType=*"text/html; charset=UTF8"*

pageEncoding=*"UTF8"*%>

<html>

<!-- 作用：管理员添加学生的三个信息的页面 -->

<head>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<script src=*"../JS/my2.js"*></script>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF8"*>

<title>addStudentPage</title>

</head>

<body>

<jsp:include page=*"topBar.jsp"*></jsp:include>

<div class=*"container pagediv"*>

<form action=*"../addStudentService"* method=*"post"* align=*"center"*>

<h3 align=*"center"*>添加学生基本信息（1）</h3>

<!-- type设为number，只能输入数字。

oninput设置最大长度 -->

<input id=*"idinput"* class=*"form-control input"* type=*"number"* placeholder=*"学号"*

value=*""* name=*"id"* oninput=*"if(value.length>12)value=value.slice(0,12)"*

onchange=*"btnAbled()"*><br>

<input class=*"form-control input"* type=*"text"* placeholder=*"姓名"* value=*""* name=*"name"* maxlength=*"5"*><br>

<input class=*"form-control input"* type=*"text"* placeholder=*"学校"* value=*"中原工学院"* name=*"sch"*><br>

<input class=*"form-control input"* type=*"text"* placeholder=*"学院"* value=*"软件学院"* name=*"college"*><br>

<input class=*"form-control input"* type=*"text"* placeholder=*"专业"* value=*"软件工程"* name=*"major"*><br>

<input class=*"form-control input"* type=*"text"* placeholder=*"班级"* value=*""* name=*"clazz"*><br>

<input class=*"form-control input"* type=*"text"* placeholder=*"报考院校"* value=*""* name=*"register\_sch"*><br>

<input class=*"form-control input"* type=*"text"* placeholder=*"报考专业"* value=*""* name=*"register\_major"*><br>

<input class=*"form-control input"* type=*"text"* placeholder=*"研究方向"* name=*"register\_num"*><br>

<input class=*"form-control input"* type=*"number"* placeholder=*"总分"* value=*""* name=*"score\_total"* oninput=*"if(value.length>3)value=value.slice(0,3)"*><br>

<input class=*"form-control input"* type=*"number"* placeholder=*"考研年份"* value=*""* name=*"score\_year"* oninput=*"if(value.length>4)value=value.slice(0,4)"*><br>

<!-- 默认按钮不可点击，当id文本框输入学号后，

文本框的onchange事件调用js方法 ，

在js中去除disabled，按钮可用 -->

<input id=*"btn"* type=*"submit"* value=*"提交"* class=*"btn btn-success btn-lg input"* disabled=*"disabled"*>

<span id=*"btnflag"* style="color:*red*;">输入学号，解锁按钮</span>

</form>

</div>

</body>

</html>

### 5.2添加学生基本信息servlet部分的主要代码：

/\*\*

\* Types

\* **@author** dandelion

\* **@time** 2019年3月13日下午8:56:30

\* **@类名** AddStudentService

\* **@作用** 从管理员添加页面接收学生的三个数据，

\* 调用DAO，添加学生数据到三个数据库。

\* **@调用**

\*/

@WebServlet("/addStudentService")

**public** **class** AddStudentService **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest request,

HttpServletResponse response)

**throws** ServletException, IOException {

doPost(request, response);

}

**protected** **void** doPost(HttpServletRequest request,

HttpServletResponse response)

**throws** ServletException, IOException {

request.setCharacterEncoding("UTF-8");

response.setContentType("text/html;charset=UTF-8");

StudentDao sd = **new** StudentDaoImpl();

//-----------------添加学生基本信息（1）--------------------

**long** id = Long.*parseLong*(request.getParameter("id"));

//如果插入的学号已存在，提示学号重复并返回addStudentPage.jsp

**try** {

**if**(sd.getStudentById(String.*valueOf*(id)).next()){

response.getWriter().println(

"<script type='text/javascript'>"

+ " alert('学号已存在');"

+ " window.location.href='JSP/addStudentPage.jsp';"

+ "</script>");

}**else**{

String name = request.getParameter("name");

String sch = request.getParameter("sch");

String college = request.getParameter("college");

String major = request.getParameter("major");

String clazz = request.getParameter("clazz");

String register\_sch = request.getParameter("register\_sch");

String register\_major = request.getParameter("register\_major");

String register\_num = request.getParameter("register\_num");

String score\_total\_str = request.getParameter("score\_total");

//score\_total总分的默认值是0

**int** score\_total = 0;

**if**(!score\_total\_str.equals("")){

score\_total = Integer.*parseInt*(score\_total\_str);

}

System.***out***.println("score\_total:"+score\_total);

String score\_year = request.getParameter("score\_year");

//实例化学生基本信息类student

Student student = **new** Student();

student.setId(id);

student.setName(name);

student.setSch(sch);

student.setCollege(college);

student.setMajor(major);

student.setClazz(clazz);

student.setRegister\_sch(register\_sch);

student.setRegister\_major(register\_major);

student.setRegister\_num(register\_num);

student.setScore\_total(score\_total);

student.setScore\_year(score\_year);

student.print();

//将学生的基本信息数据，添加到student表

ManagerDao md = **new** ManagerDaoImpl();

**boolean** flagStu = md.addStudent(student);

**if**(flagStu){

response.getWriter().println(

"<script type='text/javascript'>"

+ " alert('成功添加学生基本数据');"

+ " window.location.href='JSP/mainPage.jsp';"

+ "</script>");

}**else**{

response.getWriter().println(

"<script type='text/javascript'>"

+ " alert('添加学生基本数据似乎出了一些问题');"

+ " window.location.href='JSP/mainPage.jsp';"

+ "</script>");

}

}

} **catch** (SQLException e) {

System.***out***.println("【service/AddStudentService.java出现了错误】");

e.printStackTrace();

}

}

}

### 5.3添加学生基本信息数据库添加部分的主要代码：

/\*\*

\* Methods

\* **@author** dandelion

\* **@time** 2019年3月12日下午4:56:49

\* **@方法名** addStudent

\* **@作用** 添加学生基本信息

\* **@调用** AddStudentService.java

\* **@返回值类型** boolean，true表示插入成功，false表示插入失败

\*/

**public** **boolean** addStudent(Student student){

Connection conn = **null**; //声明Connection对象

PreparedStatement perstat =**null**; //声明PreparedStatement对象

**int** rows=0;

**try**{

conn=DBConn.*getConnection*();

String sql="insert into student(id,name,sch,college,major,class,register\_sch,register\_major,register\_num,score\_total,score\_year) values(?,?,?,?,?,?,?,?,?,?,?)";

perstat=(PreparedStatement) conn.prepareStatement(sql);//执行SQL语句

perstat.setLong(1,student.getId());

perstat.setString(2,student.getName());

perstat.setString(3,student.getSch());

perstat.setString(4,student.getCollege());

perstat.setString(5,student.getMajor());

perstat.setString(6,student.getClazz());

perstat.setString(7,student.getRegister\_sch());

perstat.setString(8,student.getRegister\_major());

perstat.setString(9,student.getRegister\_num());

perstat.setInt(10,student.getScore\_total());

perstat.setString(11,student.getScore\_year());

rows=perstat.executeUpdate();//rows接收更新结果,rows大于0表示插入成功

}**catch**(Exception e){

System.***out***.println("【ManagerDaoImpl.java\\addStudent】插入基本信息数据时发生异常");

e.printStackTrace();

}**finally**{

DBConn.*close*(perstat);

DBConn.*close*(conn);

}

**if**(rows>0){

**return** **true**;

}**else**{

**return** **false**;

}

}

## 6.主页导航栏主要代码：

<%@page import=*"dao.impl.ManagerDaoImpl"*%>

<%@ page language=*"java"*

import=*"dao.ManagerDao,java.sql.ResultSet"*

contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!-- 这是页面的顶部导航栏，可以说是非常重要了 -->

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<meta http-equiv=*"X-UA-Compatible"* content=*"IE=edge"*>

<meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>

<!-- Bootstrap -->

<link href=*"https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css"* rel=*"stylesheet"*>

<!-- jQuery (Bootstrap 的所有 JavaScript 插件都依赖 jQuery，所以必须放在前边) -->

<script src=*"https://cdn.jsdelivr.net/npm/jquery@1.12.4/dist/jquery.min.js"*></script>

<!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加载单个插件。 -->

<script src=*"https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/js/bootstrap.min.js"*></script>

<link href=*"../CSS/myCss.css"* rel=*"stylesheet"*/>

<script src=*"../JS/my2.js"*></script>

<title>topBar</title>

</head>

<body>

<div class=*"container"* style="border='10px'">

<div class=*"row"*><img src=*"../IMG/Head.png"* width=*"100%"* ></div>

<div class=*"row bggray"* align=*"center"*>

<div class=*"col-md-1 bargreen divhover"* onclick="mainPage(2019)">主页</div>

<div id=*"bardiv"* class=*"col-md-1 dropdown bargreen divhover"* onmouseover="barOver()" onmouseout="barOut()">

年份

<a id=*"bardropdown"* class=*"dropdown-toggle"* data-toggle=*"dropdown"* role=*"button"* aria-expanded=*"false"*></a>

<ul class=*"dropdown-menu"* role=*"menu"*>

<li><a onclick="mainPage(2020)">2020考研信息</a></li>

<li><a onclick="mainPage(2019)">2019考研信息</a></li>

<li><a onclick="mainPage(2018)">2018考研信息</a></li>

</ul>

</div>

<div class=*"col-md-9"* style="background-color: *gray*;">

<span class=*"spanTips"* id=*"say"*></span>

</div>

<div id=*"manager"* class=*"col-md-1 bargreen divhover"* onclick="managerPage()">登录</div>

</div><br>

</div>

<%

response.setContentType("text/html;charset=utf-8");

Cookie[] cs = request.getCookies();

ManagerDao md = **new** ManagerDaoImpl();

String str = "";

**boolean** havaLogin = **false**;

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

**for**(Cookie c:cs){

String id = c.getName();

String pwd = c.getValue();

String name = "用户";

str = md.login(id, pwd);

**if**(str.equals("登录成功")){

havaLogin = **true**;

//如果管理员已登录，根据id从数据库中查找管理员姓名并赋值给name

ResultSet rs = md.getManagerById(id);

**if**(rs.next()){

name = rs.getString(2);

}

%>

<!-- 我不知道如何把java中的变量直接赋值给js中的var，所以通过这种方法 -->

<input type=*"hidden"* id=*"a"* value=*"*<%=name%>*"*>

<script>

**var** jsname = document.getElementById('a').value;

setManagerName(jsname);

</script>

<%

**break**;

}

}

}

%>

</body>

</html>