重庆三峡学院毕业设计（论文）

学习成绩信息管理系统

**学 院：数学与统计学院**

**专 业：信息与计算科学**

**年 级：2013级**

**学 号：201306014101**

**作 者：汪龙飞**

**指导老师：鄢沛（高级实验师）**

**完成时间：2017年5月**

目 录

[摘要 I](#_Toc14512)

[Abstract II](#_Toc5947)

[1 引言 1](#_Toc7663)

[2 需求分析 1](#_Toc27400)

[2.1总体需求分析 1](#_Toc18322)

[2.2 用户需求分析 1](#_Toc28706)

[3 系统分析与设计 2](#_Toc18498)

[3.1 总体思想和目标 2](#_Toc1456)

[3.2 ER模型 2](#_Toc7622)

[3.3 系统结构、功能设计 4](#_Toc16662)

[3.3.1 系统结构设计 4](#_Toc82)

[3.3.2 功能模块设计 5](#_Toc17625)

[3.4 数据库设计 6](#_Toc5312)

[3.4.1初始关系 6](#_Toc8193)

[3.4.2关系模式规范化 6](#_Toc21262)

[3.4.3模式修正 6](#_Toc10857)

[3.4.4存储记录结构设计 6](#_Toc17919)

[3.4.5数据库关系图 7](#_Toc29882)

[3.5 数据字典 7](#_Toc28870)

[3.6 数据流图 9](#_Toc7473)

[4 系统平台与技术介绍 10](#_Toc13165)

[4.1系统平台 11](#_Toc21326)

[4.2 集中式管理——主机/终端模式 11](#_Toc13951)

[4.2 C/S和B/S模式的比较 11](#_Toc17952)

[4.3 SpringMVC处理流程 12](#_Toc9878)

[4.4 Hibernate持久层框架 13](#_Toc26196)

[5 系统实现 14](#_Toc5670)

[5.1 前端UI 14](#_Toc16961)

[5.2 后端数据控制 20](#_Toc10954)

[6 系统设计的关键技术 30](#_Toc15271)

[7 系统试运行结果与评价 31](#_Toc16331)

[结束语 32](#_Toc607)

[参考文献 33](#_Toc17064)

**学生成绩信息管理系统**

汪龙飞

（重庆三峡学院数学与统计学院信息与计算科学专业2013级 重庆万州 404000）

摘要**：**近年来，随着不断的扩招，运用常规的文本管理方法对学生成绩的管理已经变得越来越困难了，因此学校需要一种高效的，可行的成及管理系统来帮助管理学生成绩。

在选用技术和模式的时候着重考虑了系统的高效性和简易性。本系统是基于Java作为后端技术、EasyUI为前端技术并结合MySQL 数据库而设计的、能够便于管理人员操作的B/S模式成绩管理系统。

该系统结合权限管理的理念加入了权限控制机制，解决了不同身份做不同权限能力的问题。从细节方面，本系统着重考虑了用户在新增成绩和查询成绩两方面的简单操作，力求为用户带来方便。通过本系统学生能够直接的查看到科目的成绩，而任课老师可以查看或处理学生的成绩信息；在提高学生成绩管理工作效率的基础上，让同学们通过这个系统对各个科目进行分析和总结。通过试运行，本系统实现了学生成绩的基本操作和权限控制的功能。

关键词： Java，MySQL ，EasyUI，Hibernate，Spring

Management information system design

Wanglongfei

(Grade 2013, Information and Computing Science, School of Mathematics and Statistics, Chongqing Three Gorges University, Wanzhou, Chongqing 404000 )

Abstract:In recent years, with the continuous expansion of the use of text, conventional methods of management of student achievement management has become more and more difficult, so the school needs an efficient, feasible and management system to help the management of student achievement.

When selecting technology and model, the efficiency and simplicity of the system are considered. This system is based on Java as back-end technology, EasyUI as front-end technology, and MySQL database design, and can easily manage the operation of B/S mode performance management system.

With the concept of privilege management, the system has added the access control mechanism to solve the problem of different identities and different permissions. From the details, the system focuses on the user's new achievements and query results in two aspects of simple operations, and strive to bring convenience to users. Through this system, students can directly see the subjects, and teachers can view or deal with student achievement information; improve work efficiency based on the management of student achievement, so that students of all subjects were analyzed and summarized through the system. Through trial operation, the system has achieved the basic operation of student achievement and the function of access control.

**Key words:** Java,MySQL,EasyUI,Hibernate,spring

# **1 引言**

以前是通过学生档案和公告栏来提供各种信息和通告新的变化，对于科技飞速发展的今天，这无疑是一种费力费神的体力活，当一个优秀的成绩管理系统出现之后，这个繁杂的工作已被网站和校园局域网所取代，使得学生信息的管理更方便、安全。学生成绩管理系统，它不仅提高了工作效率，还减少了出错的可能，即使出现错误，也能在很短时间内修正。在进行系统设计时，我们对系统的现状进行了充分的分析，发现以往的学生成绩系统往往是只能通过进入系统的登录界面选择身份、随即通过你选择的身份进行显示你能控制的区域。本系统根据上面我们提出的提高工作效率，减少工作时间为目的，通过改变现在需在用户登录界面选择身份的需求并且选用Java+MySQL 的组合模式进行开发的一套B/S模式系统，通过逐步的完善以及老师的建议和给予的帮助，一个愈来愈成熟的系统油然而生。

学生成绩管理系统可以实现学生基本情况的新增、修改、删除等基本的操作；本系统增加了角色与资源的概念，通过不同的人物性质设置不同的角色，角色与资源连接，这样就可以做到通过不同的人物性质访问不同的资源，用于区分超级管理员、学生和老师；MySQL 数据库提供数据存储，同时可以便于访问数据，计算数据。

# **2 需求分析**

# **2.1总体需求分析**

本系统是打算做一个基于权限控制处理的学生成绩管理系统，所谓权限控制也就是拥有不同身份的用户只能进行权限范围内的操作，譬如一个学生是不能修改自己的成绩分数的，所以在本系统中这些基本的权限控制尤为重要。其次就是对用户自己的成绩系统需要一个清晰的比较和显示。

# **2.2 用户需求分析**

根据总体分析可以得出：由于本系统可以设置角色这个特殊的权限集合功能，所以对于用户我们可以大致的分为学生用户、教师用户和管理员用户。除此之外还可以用无权限用户，测试用户等特殊用户的存在。

# **2.1.1 学生用户**

学生用户只能对该系统中的学习成绩、个人信息等进行查看的功能。对自己成绩查询是本系统的基础功能，而能够查看其他同学的功能使得自己能够对其他同学的成绩进行参考，已达到明确自己成绩位置的用途。

# **2.1.2 教师用户**

教师用户设置了对学生个人信息的管理，成绩的理性操作外，本系统还设置了对自身信息及其他同事信息的审查修改功能。同时在查看学生成绩的时候能够查看到其他教师管属下的学生成绩信息，已达到对各个学生各科成绩的了解。界面上提供了快速搜索功能，以能够只查询单方面的需求。

# **2.1.3 超级管理员**

超级管理员作为系统的特殊用户，它具有本系统所有的权限，其中包括学生信息的管理、教师信息的管理、学生成绩的录入和修改功能以及对用户权限的分配、新增修改用户角色资源等的权限。

# **3 系统分析与设计**

# **3.1 总体思想和目标**

本系统是做一个让不同的角色能够做不一样的操作即有不一样的权限的学生成绩管理系统，能够简单的完成学生对成绩的查看，教师对成绩的管理查看。在界面上你可以通过对角色的分配进行授权，也就是做权限的控制功能，例如作为学生的用户只能够查看自己成绩、名称、班级、科目等信息进行查看，但不能对自己的学习成绩、个人信息等相关信息进行操作，而老师不仅能够查看自己的的学生信息和教师信息外还可以对学生的信息进行编辑修改。所以面对这两个不一样的角色我们就需要分配不一样的权限。通过权限的设置来判断用户能进行的相关操作。

# **3.2 ER模型**

# **3.2.1 学生实体**

学生

联系电话

学号

qq

性别

姓名

生日

学生ID

**图3.1 学生实体E-R图**

# **3.2.2 教师实体**

教师

工号

姓名

职称

入职日期

教师ID

**图3.2 教师实体E-R图**

# **3.2.3 用户实体**

用户

用户ID

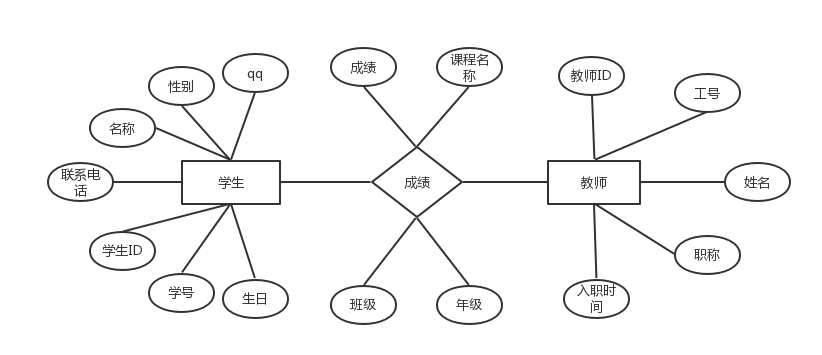
用户名

密码

用户权限

**图3.3 用户实体E-R图**

# **3.2.4 全局模型**

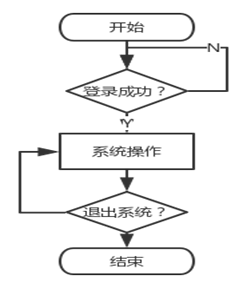


**图3.4 全局E-R图**

# **3.3 系统业务、功能设计**

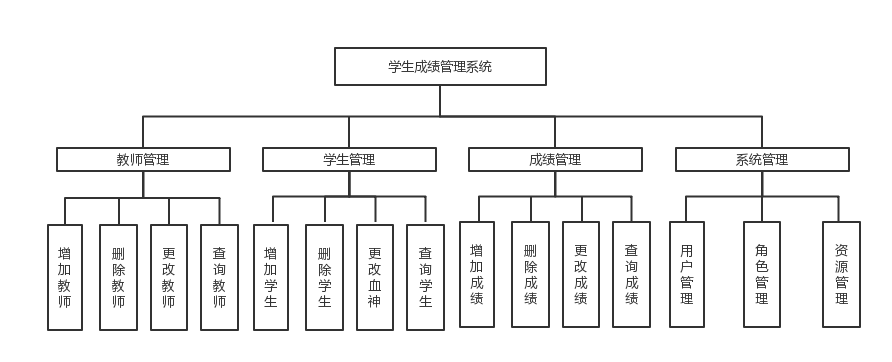
# **3.3.1 系统业务设计**

本系统采用当前比较流行的输入姓名密码的模式进行系统操作，只有在姓名存在且输入了与之对应的密码才能进入系统，具体成绩管理系统系统业务图如图3.5所示：



**图3.5 系统业务图**

# **3.3.2 功能模块设计**



**图3.6 系统功能模块**

# **3.4 数据库设计**

# **3.4.1初始关系**

从需求分析可以看出，系统存在以下几个实体：

学生实体：用于显示学生的相关属性，包括学生姓名、学生性别、学号、联系方式、所在班级。

教师实体：用于显示教师的相关信息，包括工号，姓名，联系方式，入职时间，职称。

用户实体：用于显示能登录系统的用户信息，包括用户名、密码和用户权限。

# **3.4.2关系模式规范化**

考虑到实体表之间的关系，老师管理成绩，学生查看成绩，中间的联系就可以创建一个成绩的实体，所以就需要以下实体：

学生实体：用于显示学生的相关属性，包括学生姓名、学生性别、学号、联系方式、所在班级。

教师实体：用于显示教师的相关信息，包括工号，姓名，联系方式，入职时间，职称。

成绩实体：用于学生和教师之间的联系，包括成绩，课程名称，学期，班级名称，学生ID，教师ID。

用户实体：用于显示能登录系统的用户信息，包括用户名、密码和用户权限。

# **3.4.3模式修正**

综合以上考虑，加上对权限的管理可以得出本系统需要以下几个实体：

学生实体：用于显示学生的相关属性，包括学生姓名、学生性别、学号、联系方式、所在班级。

教师实体：用于显示教师的相关信息，包括工号，姓名，联系方式，入职时间，职称。

成绩实体：用于学生和教师之间的联系，包括成绩，课程名称，学期，班级名称。

角色实体：资源的集合，用于管理资源，包括角色名，角色描述。

资源实体：系统资源，包括资源类型，资源地址，资源名称。

用户实体：用于显示能登录系统的用户信息，包括用户名、密码和用户权限。

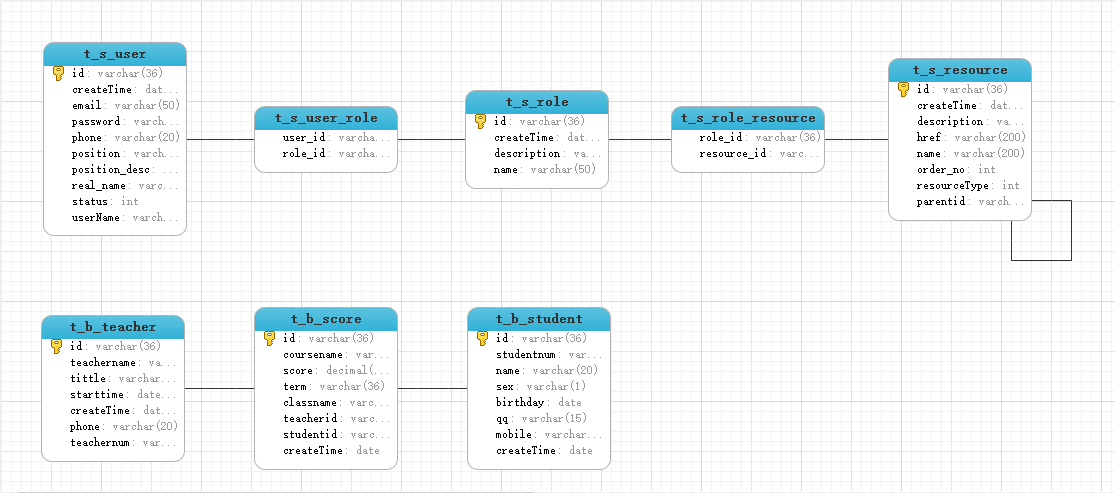
用户-角色：通过用户与角色的绑定进行权限控制；包括角色ID，用户ID。

角色-资源：通过资源与角色的绑定进行资源筛选；包括资源ID，角色ID。

# **3.4.4存储记录结构设计**

信息存储结构的设计在系统中尤为重要，其中需满足系统代码执行效率、数据冗余、系统维护以及信息控制等方面的要求。信息管理是离不开数据库的支撑。基于以上效率的考虑，本系统决定采用MySQL 数据库作为成绩管理系统的数据源。

# **3.4.5数据库关系图**

**图3.7 数据库关系图**

# **3.5 数据字典**

在数据库开发实现当中，数据库表的设计尤为重要，我们需要去恰当的字段和命名来避免冗余和写明注释。若数据冗余则会造成对内存的需求过大，数据库响应过慢等问题，而若命名没有含义且没有具体的注释，在后面数据库的维护会带来很大的麻烦，所以在系统分析和功能设计这两个过程中需要给用户提供一些数据相关的参考信息，即为我们下面给出的数据字典。

# **3.5.1 学生信息表**

在每一个数据表中都需要唯一的标识来表明这条数据，目前使用最多的字段为ID，当然本系统也是通过ID来确认的唯一性，并将其设置为主键。在学生信息表中我除了定义学生相关信息如学号，姓名，性别，qq，手机以外我还加入了创建事件，该字段是用于该条数据的生成时间，方便后期维护。在字段设计中学生ID是通过h框架自动生成的标准的32位字符串，所以我在数据库中为了避免后期字段长度的增加带来的麻烦我就设置我36位。还有其中的性别字段，由于该字段一般只有三种情况（男，女，未知），所以在数据库中只给了1位长度，同时通过1表示男，2表示女，3表示未知的转换。具体的信息表如表3.1所示：

**表3.1 学生信息表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 数据长度 | 是否为空 | 说明 | Key |
| 学生ID | Varchar | 36 | N |  | P\_K |
| 学号 | Varchar | 15 | N |  |  |
| 姓名 | Varchar | 20 |  |  |  |
| 性别 | Varchar | 1 |  | 男或女 |  |
| 生日 | Date |  |  |  |  |
| qq | Varchar | 15 |  |  |  |
| 手机 | Varchar | 11 |  |  |  |
| 创建时间 | Date |  |  |  |  |

# **3.5.2 教师信息表**

在教师信息的数据表中，除了确定唯一性的ID，教师的基本信息（名称，手机号，编号）外，还添加了教师职称，入职时间信息，而对于教师名称和职称而言，一般情况下不会多余10位字符所以在设计数据库的时候将数长度设置为10，而教师编号则是默认定义的20位的长度，该系统会在前端输入以及后端进入数据库插入数据操作的时候进行校验是否通过。具体的教师信息如表3.2所示：

**表3.2 教师信息表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 数据长度 | 是否为空 | 说明 | Key |
| 教师ID | Varchar | 36 | N |  | P\_K |
| 教师名称 | Varchar | 10 |  |  |  |
| 教师职称 | Varchar | 10 |  |  |  |
| 入职时间 | Varchar |  |  |  |  |
| 教师编号 | Varchar | 20 |  |  |  |
| 手机 | Varchar | 20 |  |  |  |

# **3.5.3 成绩信息表**

成绩信息表是用于来统计和分析学生成绩信息的数据表，同时也是学生和教师的关系表。这张表中包括了成绩ID，课程名称，学生成绩，学期，班级等基本信息和学生ID和教师ID两个外键。

而这两个外键我都是选择了用cascade作为外键约束（外键约束分为：cascade,set null,no action,restrict四种约束，其中cascade是当在父表也就是被选用的表上产生变化的时候同时将子表也就是当前表进行相应的操作，set null则是在父表上产生变化时将子表上对应的信息设置为null ，no action是和由子表反应到父表的一种操作，不管是否在子表中有更改的记录，父表中都不进行相关操作，restrict约束其实和no action有相同的地方，都会立即检查外键约束），这样就可以在当学生ID或者教师ID发生改变的时候进行相对应的更新或删除。具体的成绩信息表如表3.3所示：

**表3.3 成绩信息表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 数据长度 | 是否为空 | 说明 | Key |
| 成绩ID | Varchar | 36 | N |  | P\_K |
| 课程名称 | Varchar | 15 |  |  |  |
| 学生分数 | decimal | 10,0 |  |  |  |
| 学期 | Varchar | 36 |  |  |  |
| 班级 | Varchar | 20 |  |  |  |
| 学生ID | Varchar | 36 |  |  | F\_K |
| 教师ID | Varchar | 36 |  |  | F\_K |

# **3.5.4 用户信息表**

用户信息表是用来记录能够登录本系统的用户信息，该用户信息表定义了用户的基本信息（用户姓名，手机号，邮箱）、密码、职务、状态等字段，其中用户需要通过自己的登录名和密码来确定是否有进入本系统的权限，而用户又和学生教师不为同一个数据源，所以在用户表中我添加了一个职务的字段用于标注本用户的职务，字段状态为该用户是否启用，若为禁用则该用户就相当于没有进入本系统的权限，即不能进入该学生成绩信息管理系统，本表中有登录名和用户名的区分，登录名是用户在进入系统的时候输入的用户名称，而用户名只是相当于一个别名，用于管理人员对人员信息的区分。具体的用户表信息如表3.4所示：

**表3.4 用户信息表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 数据长度 | 是否为空 | 说明 | Key |
| 用户ID | Varchar | 36 | N |  | P\_K |
| 用户名 | Varchar | 10 |  |  |  |
| 密码 | Varchar | 32 | N |  |  |
| 手机 | Varchar | 20 |  |  |  |
| 邮箱 | Varchar | 50 |  |  |  |
| 职务 | Varchar | 20 |  |  |  |
| 职务描述 | Varchar | 36 |  |  |  |
| 状态 | Int | 11 |  | 正常或禁用 |  |
| 登录名 | Varchar | 50 |  |  |  |

# **3.6 数据流图**

数据流程的目的是介绍数据在储存或读取等操作的情况下的数据走向，能够方便读者在应用本系统的同时清晰的知道数据的流向。下面简要介绍部分需求的数据流图：

# **3.6.1 成绩查询**

学生成绩查询的功能中，首先需要根据界面上的操作来确定数据的查询方式，在数据库服务端接收到了需要查询的需求时对数据筛选成你所需要的数据并返回结果，具体的数据流图如图3.8所示：

学生成绩信息表

学生成绩信息

学生成绩信息

查询信息

查询方式

学生

教师

接受查询方式

信息处理

产生查询结果

学生

教师

**图3.8 学生成绩查询的数据流图**

# **3.6.2 成绩录入**

在学生成绩录入的过程中，首先你需要将需要录入的信息发送给服务端，服务端接收到信息后对已有信息进行查询，如果符合要求则录入数据库并将成功信息返回界面。具体的数据流图（DFD）过程表示如图3.9所示：

学生成绩信息表

学生成绩信息

学生成绩

查询信息

录入信息

教师

接收录入信息

信息处理

产生录入结果

教师

**图3.9 学生成绩录入的数据流图**

# **3.6.3 角色授权**

在角色授权的过程中，首先你需要将需要用户的信息发送给服务端，服务端接收到信息后对已有信息进行查询，如果符合要求则对该用户进行权限分配并将成功信息返回界面。具体的数据流图（DFD）过程表示如图3.10所示：

角色信息表

角色信息

角色信息

查询信息

分配权限

用户

接收用户信息

信息处理

产生分配结果

用户

**图3.10 角色授权的数据流图**

# **4 系统平台与技术介绍**

# **4.1系统平台**

⑴ 系统的硬件平台

主机：华硕K550D

CPU：AMD A8处理器

内存：4G

硬盘：500G

⑵ 系统的软件平台

操作系统：Microsoft Windows 10操作系统

数据库系统：MySQL 5.7数据库

浏览器：chrome浏览器

# **4.2 集中式管理——主机/终端模式**

集中式管理即集中式计算机网络是利用一个大型的中央系统（服务器），终端是客户机，全部的数据都储存在中央系统中，通过数据库管理系统（DBMS）进行统一的管理，所有的任务处理都在其大型的中央系统上完成，而终端客户机只做输入和输出的操作。客户机自己不用作任何任务的处理，所有的任务处理都放在主机中央系统上进行处理。所以集中式管理数据存储的显要特点就是可以把所有数据都保存在同一个地方。

# **4.2 C/S和B/S模式的比较**

C/S模式即是大家熟悉的客户/服务器模式，全称为Client/Server。客户端我们需要安装一个装用的特定开发的客户端软件，而服务器端则一般会用工作站、高性能的PC或者使用一些小型机，且需要采用大型得数据库管理系统（DBMS），比如MySQL ，SQL Server，Oracle，Sybase或Informix等。但是由于C/S模式需要安装特定得客户端软件，所以首先我们需要找到特定得客户端软件，然后进行安装，这就涉及到一个安装的工作量问题，若当软件需要升级，我们需要在每个客户端的客户机上面进行重新下载安装，这样就会出现升级成本比较高且维护比较麻烦。处于安全性能的考虑，倘若其中一台客户机出现了病毒攻击，磁盘损坏等问题，我们就需要对其进行重新安装或者更新升级，除了以上我们总结出的升级维护问题外，我们还需要对各个客户端进行排查和处理，若遇到一些公司的分部或者专卖店比较多的情况下，这已经不是一个工作量大小的问题了，而是你需要来回跑多少路程的问题，除此之外还有一个系统兼容性问题，这特定的软件需要特定的开发，也许软件只能使用与Win7但是不能用于Win Xp或其他系统。

B/S模式就是与C/S模式对应的浏览器/服务器模式，全称为Brower/Server。B/S模式中也有客户机的这一端，只是将特定的客户端代替为灵活已安装的浏览器，在客户机上面我们就只需要安装一个浏览器就可以进行系统的访问，比如一些常见的浏览器：Chrome浏览器，百度浏览器，搜狗浏览器，Microsoft Edge或者是Internet Explorer。服务器端可以采用与C/S相同的大型数据库系统，这样浏览器就可以通过Web Server与服务端的数据库进行数据交互，但是B/S模式也有一定的局限性第一个是安全性问题，由于所有的资源都是在互联网上进行交互，这样就带来了中病毒攻击或者被侵权等相关问题。还有一个就是浏览器的兼容性问题，很多开发人员肯定都知道在web开发中IE是浏览器的一个bug，由于IE在前几年独占螯头的优势，所以目前仍然有大部分用户还在坚持用着比较老版的IE8及其更低版本，所以在开发的工程中我们还需要解决IE，Chrome，FireFox等浏览器的兼容性问题。

鉴于以上C/S和B/S模式的认识，总结出B/S模式虽然有安全性和兼容性问题的存在，但是在当下技术的发展下，安全性问题无论是从互联网的管制来讲还是从研发过程中的安全性能测试角度而言都是严格把关的，而对于兼容性问题，随着互联网的快速发展，人们越来越希望看见精湛的界面、交互性更加完善、体验度更加高的界面，所以早在几年前都陆续出现了解决IE8甚至更低版本的兼容代码供我们参考引用。而对于C/S模式而言，B/S模式还有一个更大的优势就是其不需要特定的安装客户端软件就可以在任何有浏览器能够展示界面的地方进行运行操作，只要这在一台客户机上面安装一个浏览器就可以完成功能演示操作，这样就对客户端的维护成本就低了下来，同时倘若你需要升级的情况下，你可以直接对你的软件进行升级，然后同步在互联网的服务端这样客户机的救护做响应的更新，这样不仅可以降低维护成本还可以降低升级难度。对于系统的扩展性来说，操作是十分简单的，只需要由系统管理员在数据库中添加或者在界面上添加一个用户，并为其设置一个专属密码就可以对系统进行访问了，当然有些系统还提供自己申请的方式注册用户，只要得到了系统管理员或者系统原本设计的审核人同意就可以进入系统了。

# **4.3 SpringMVC处理流程**

SpringMVC是Java中典型的三层架构框架，其中包括了Controller层（前端控制层），Model层（模型层），View层（视图层）。在三个架构中他们有响应的请求进行处理，具体执行步骤如下：

1. 当用户发送请求的时候也就是在前端控制器接收到请求信息时，前端控制器对信息分析后决定选择信息带入的页面控制器来进行渲染请求，即完成当前控制器层所控制的逻辑处理部分。如图4.1中的1、2步；
2. 然后界面控制器接收到消息后，对功能进行处理分析，带着接收到的参数去收集和绑定一个对象并进行校验，若校验通过则会返回一个ModelAndView（模型数据和逻辑视图名）的对象；图4.1中的3、4、5步。
3. 前端控制器得到了对象后对相应的界面进行渲染，并把对象中的数据传入界面中。图6、7步。
4. 最后前端控制器将响应的结果返回给用户。图4.1中的8步。



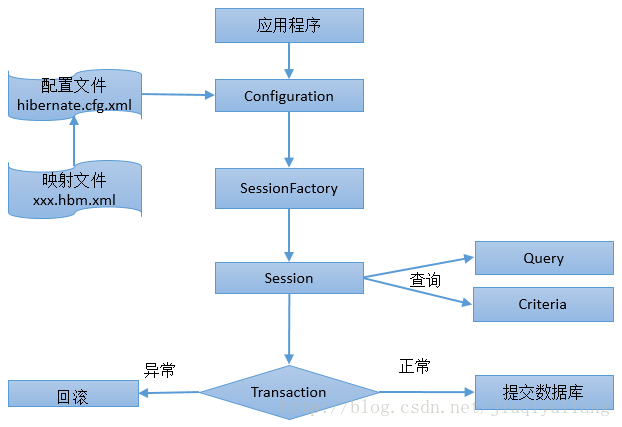
**图4.1 Spring Web MVC处理请求的流程图**

# **4.4 Hibernate持久层框架**

原来没有使用Hiberante做持久层开发时，存在很多冗余，如：各种JDBC语句，connection的管理，所以出现了Hibernate把JDBC封装了一下，我们不用操作数据，直接操作它就行了。

我们知道非常典型的三层[架构](http://lib.csdn.net/base/architecture" \o "大型网站架构知识库" \t "http://blog.csdn.net/jiuqiyuliang/article/details/39078749/_blank)：表示层（Controller），业务层（Service），还有持久层（Dao）。Hiberante也是持久层的框架，而且持久层的框架还有很多，比如：IBatis，Nhibernate，JDO，OJB，EJB等等。

**Hibernate是开源的一个ORM（对象关系映射）框架**。ORM，即Object-Relational Mapping，它的作用就是在关系型数据库和对象之间做了一个映射。从对象（Object）映射到关系（Relation），再从关系映射到对象。这样，我们在操作数据库的时候，不需要再去和复杂SQL打交道，只要像操作对象一样操作它就可以了（把关系数据库的字段在内存中映射成对象的属性）。Hibernate的核心如图4.2所示：



**图4.2 Hibernate核心图**

从上图中，我们可以看出Hibernate六大核心接口，两个主要配置文件，以及他们直接的关系。Hibernate的所有内容都在这了。那我们从上到下简单的认识一下，每个接口进行一句话总结。

（1）、Configuration接口:负责配置并启动Hibernate

（2）、SessionFactory接口:负责初始化Hibernate

（3）、Session接口:负责持久化对象的CRUD操作

（4）、Transaction接口:负责事务

（5）、Query接口和Criteria接口:负责执行各种数据库查询

**注意**：Configuration实例是一个启动期间的对象，一旦SessionFactory创建完成它就被丢弃了。

# **5 系统实现**

# **5.1 前端UI界面**

# **5.1.1 前端搭建**

前端部分，也可以叫UI部分，包括了一个系统中所有你能看见的界面样式和交互，每个系统都会又一个属于自己的UI部分。在2008年以前，前端界面其实是由后端工程师一起编写的，甚至有些界面的跳转都是通过后端校验后进行重定向的方式进行渲染，之前出来的界面就只能是以后端功能为主体，前端界面为功能实现。所以在这之前都是先确定了需求后就进行技术研究，功能需求分析等后台规划，对于前端都是在后端规划结束后对其进行显示展示规划。还有一个问题就是在前后端数据传输的过程中由于后端处理的信息过多就会造成延迟，这样就影响了系统的流畅性，所以在2008年之后出现了前端这个概念，从此就有一批前端工程师的出现，他们的工作内容就是用自己总结的前端界面经验来开发所有你能看见的部分也就是UI部分。当经验逐渐累积，UI部分各个框架油然而生，如EasyUI，requireJS，react，vue等一大批先詹的框架。运用这些框架可以根据业务需求来编写用户体验度高，交互性比较好的界面，除此之外，一些输入校验、友好型提醒都可以通过前端来实现，从表面上解决了用户看见延迟等尴尬的现象，基于此就出现了前端决定后端的格局。

在编写过程中，为了不反复调用公共资源，所以本系统定义一个公共应用的文件base.jsp。具体的代码如下：

<!--引用Css文件-->

<link rel=*"stylesheet"* type=*"text/css"* href=*"plug-in/EasyUI-1.4.1/themes/default/EasyUI.css"*>

<link rel=*"stylesheet"* type=*"text/css"* href=*"plug-in/EasyUI-1.4.1/themes/icon.css"*>

<!--引用jquery文件-->

<script type=*"text/Javascript"* src=*"plug-in/jquery/jquery-1.11.2.min.js"*></script>

<!--引用EasyUI文件-->

<script type=*"text/Javascript"* src=*"plug-in/EasyUI-1.4.1/jquery.EasyUI.min.js"*></script>

<script type=*"text/Javascript"* src=*"plug-in/EasyUI-1.4.1/locale/EasyUI-lang-zh\_CN.js"*></script>

<!--引用校验Js文件-->

<script type=*"text/Javascript"* src=*"js/system/validform.js"*></script>

使用MVC框架时，为了在后端渲染前端界面有个统一的标准我们一般会在Xml文件中对文件所在路径进行控制，具体的控制代码如下：

<bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<!--引用文件前缀-->

<property name=*"prefix"* value=*"/pages/"* />

<!--引用文件后缀-->

<property name=*"suffix"* value=*".jsp"* />

</bean>

配置成功后系统控制层在渲染界面的时候只需要写文件主要的部分文件名就可以了，如文件为位于pages下的index.jsp；我们在渲染文件时只需要写ModelAndView(“index”)就可以了。

在每个系统中都又一个首页界面，也是初始界面。这个是需要在web.xml里面对首页界面路径进行配置，当配置了welcome-file-list后，系统将会自动检索该节点下的<welcome-file>，若发现该地址时合法的就访问该路径。具体的代码如下：

<welcome-file-list>

<welcome-file>/pages/system/login.jsp</welcome-file>

</welcome-file-list>

配置了初始页之后，每个系统都会出现错误的界面，所以在xml文件中，我们仍需要对错误界面进行分类和重定向，本系统在web.xml文件中对错误文件进行分类并重定向。在这里我们用404错误作实例：

<error-page>

<error-code>404</error-code>

<location>/pages/common/404.jsp</location>

</error-page>

# **5.1.2 登录界面**

每一个信息管理系统都是需要对用户进行校验的，所以都需要一个登录的界面来区分你是否拥有进入本系统的权利。参照其他的学生成绩管理系统，一般的都会使用在街面上对用户类型进行筛选，如管理员，学生，教师这三个不同的职务，等我们选择了不同的职务，我们就将进入属于本职务的一个系统，这样当你在开发的时候你需要对界面进行显示控制，或者你需要写不同职务种类个数的界面来区分显示。倘若一位学生家长需要登录这个管理平台对自己学生成绩进行查看，当然家长进入该系统只能用查看的权限，由于系统在界面上只给出了学生，教师，管理员三种类型的用户，所以当家长需要登录的时候只有选择学生这个选项，且拥有了学生一样的控制权限了，出于对系统多因素和每个职位或身份可能有不同的系统需求，所以在登录界面我采用了不选择用户类型的方式，只要你有该系统的账号你就可以登录本系统。另将用户名：soaadmin 密码：admin作为管理员账号。具体界面效果如图5.1所示：；

**图5.1 成绩管理系统主界面**

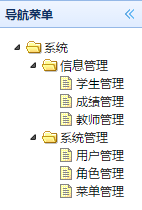
只有这样一个界面的情况下，可以让人觉得没有进行身份的筛选，就会出现无论你是教师还是学生你都拥有类似于管理员的全部访问权限，解决这个问题的方法就是在需求分析中已经提出的基本权限控制的功能，所以在系统实现的过程中，权限的控制成为了本系统的重点也是难点之一，但现在我们只讲对于前端界面而言的难度。

在图5.1中可以看出，界面登录只要你有账号和密码是不需要选择用户类型就可以登录成功的，但是登录成功后的界面应该如何显示成了当下的问题，考虑到之前的大部分系统需要对不同用户的界面进行隐藏或者重写的方式有点啰嗦，所以在考虑这个难点我从以下几个问题下手：

1. 权限控制是控制什么？
2. 如何在界面上只通过显示就能实现权限控制？
3. 在界面如何对权限的划分？

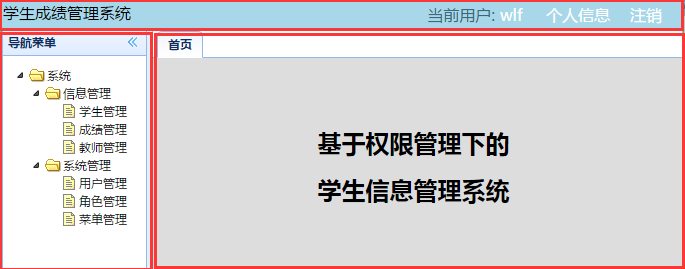
在结合网上教程的分析和导师的意见之后得出了答案

1. 信息系统权限的控制其实就是对信息的控制，对于信息的控制又分为对数据信息的控制和对能操作的系统资源信息控制（系统里面的按钮、菜单等归属为系统资源）。
2. 在界面上，能够对信息进行动态的加载且很够对数据进行筛选的控件为树，列表。如果我要进行相关操作，列表是不能满足我的需求的，所以在界面上我选择了用树来代替一级菜单对权限进行控制。
3. 权限的划分是必须要在界面上实时形成的，并且是能够修改的，基于第二条我们使用树形来展示能访问的权限，权限的划分其实就是对树形数据的筛选，所以在树形节点上加入菜单管理，可对所有用户进行分配。当然这只能是管理员才有的权限。具体的界面设计图如图5.2所示：

**图5.2 导航菜单图**

# **5.1.3 首页界面**

系统登录成功后，按照流程就进入了首页的界面。首页本系统采用了网页的基本结构即上左中的结构来展示，如图5.3所示：

 **图5.3 系统首页界面**

本系统选用EasyUI作为前端框架，所以也就用EasyUI自带的功能实现了界面结构的完成。EasyUI选择用东、南、西、北和中五个方向来区分的五个领域，其中必须包含的是中间这部分，也就是说只要你需要用EasyUI的布局，中间这块领域是不能缺少的，但是这一块是不需要设置任何宽度和高度的，因为这是由其他四个领域确定出来的。而其他领域就和正常的地理位置是一致的了。如以下实例：

<div class="EasyUI-layout" style="width:600px;height:400px;margin:0 auto;">

<div data-options="region:'north',title:'North Title',split:true" style="height:100px;"></div>

<div data-options="region:'south',title:'South Title',split:true" style="height:100px;"></div>

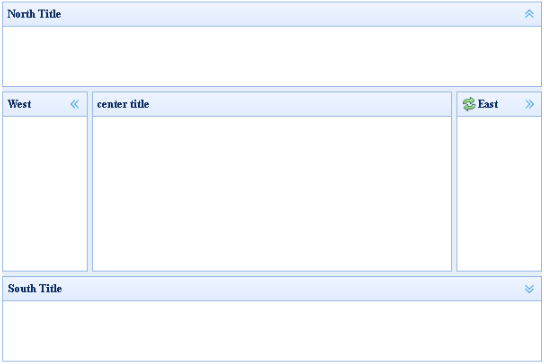
<div data-options="region:'east',iconCls:'icon-reload',title:'East',split:tr ue" style="width:100px;"></div>

<div data-options="region:'west',title:'West',split:true" style="width:100px;"></div>

<div data-options="region:'center',title:'center title'" style="padding:5px;"></div>

</div>

运行结果如图5.4所示：



**图5.4 布局运行结果图**

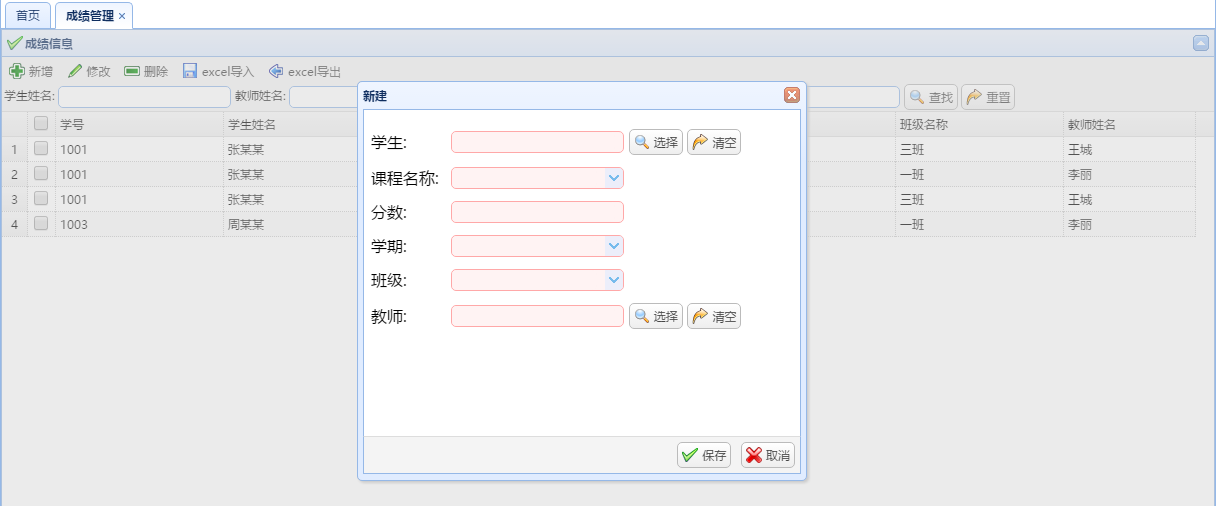
在首页北部也就是顶部，本系统包括系统主题个人信息、当前登录用户和注销功能或信息，左边就只是一个导航菜单，中间就是正文部分，当点击顶部个人信息的当点击左侧树形节点的时候就对中间的tab页进行一个添加并将内容填充到新tab标签内。当我们点击成绩管理信息间出现如图5.5所示的界面：



**图5.5 成绩管理界面**

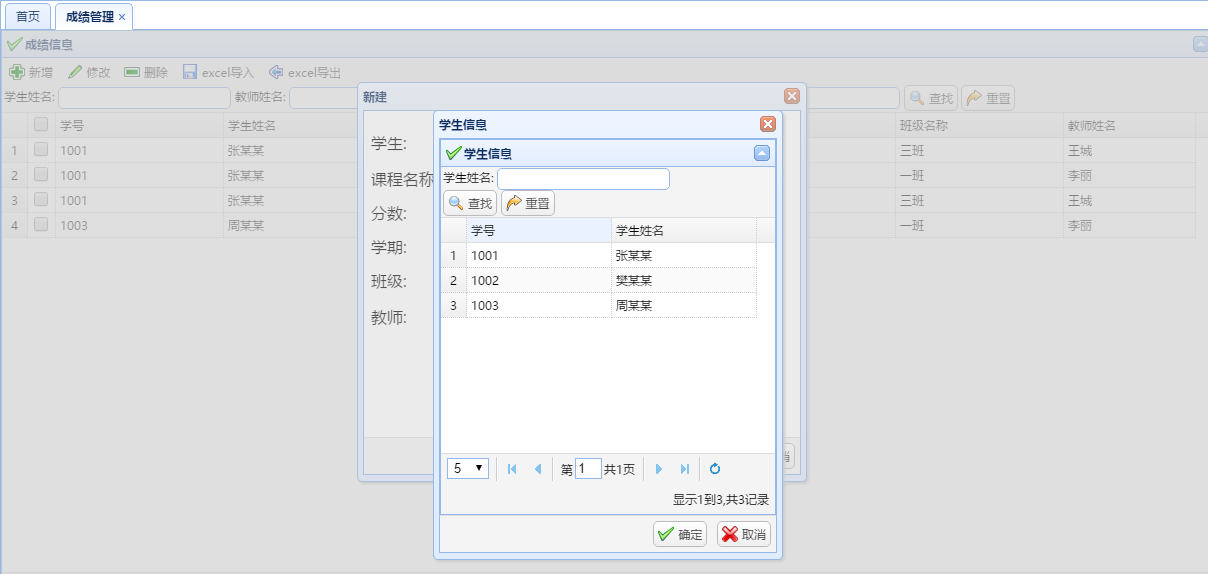
# **5.1.4 成绩录入**

对于学生成绩的录入，本系统采用了弹窗的方式进行数据的添加，在弹出的模态框中由文字字段的说明和需要输入数据的输入框已经筛选框组成。具体的界面如图5.6所示：



**图5.6 成绩录入图**

对于学生和教师这两个实体而言，本系统在这个输入框设置只读属性并在其后放置了选择和重置按钮，点击选择将出现学生信息框，教师同理。具体界面如图5.7所示：



**图5.7 学生选择界面**

# **5.2 后端数据控制**

# **5.2.1 后端配置**

与前端对应的后端部分，不像前端这样能够形象的展示效果，后端更多的是抽象的实现对数据库的操作。本系统采用了三层架构的方式来编写后端的代码，所以在调用上本系统只在Controller接受前端传来的数据，然后通过调用Service层进行逻辑处理后进入Dao层对数据库信息处理。这样的方式在后期维护是非常方便的，可以明确的定位你的问题是出现在了控制层，逻辑处理层还是数据处理层。本系统中对实体的增改都是用泛型的方法来实现，这样做就有了提高扩展性、提高复用性、代码简洁等优势。

首先本系统的服务端是用的本地数据库，所以在需要配置本地数据库的连接信息。如本系统是用hibernate作为后端框架来对数据库进行操作，所以对于数据库的连接，我们大致只需要配置数据库的地址，数据库登录名，数据库登录密码，具体配置如下：

connection.url=jdbc:MySQL ://localhost:3306/score?autoReconnect=true&autoReconnectForPools=true&useUnicode=true&characterEncoding=utf8；

connection.username=root；//数据库登录名

connection.password=root；//数据库登陆密码

由于登录界面都是可以访问查看的，所以本系统在配置Spring时对登录功能的控制层进行了拦截。具体的代码如下：

<mvc:interceptors>

<mvc:interceptor>

<mvc:mapping path="/\*\*" />

<bean class="com.wlf.system.interceptor.AuthInterceptor">

<property name="excludeUrls">

<list>

<value>loginController.do?login</value>

<value>loginController.do?doCheck</value>

</list>

</property>

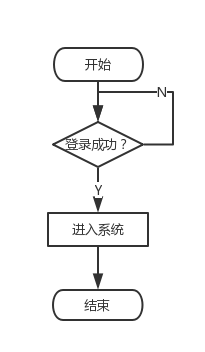
</bean>

</mvc:interceptor>

</mvc:interceptors>

# **5.2.2 登录功能**

一个完整的登录功能，能够实现用户正常的进入系统，并且当出现错误的时候能够友好的给出提醒，在登录的过程中，首先我们需要接收前端传来的用户名和密码。当接收到用户名和密码的时候我们将对用户名和密码进行校验，这部分将实现用户能否正常进入系统，所以登录功能的流程大致如图5.8所示：



**图5.8 登录流程图**

由于登录界面的特殊性，所以业务处理都在Controller层操作。根据流程，本系统设置在点击登录的时候进行对登录信息的校验，当成功后将成功信息返回给前端界面。

在登录的过程中，我们需要按照以下的步骤来进行校验：

（1）获取登录用户输入的登录名和密码，通过本控制层引用业务层中getUserByNameAndPassword方法通过用户名称和密码进行查询用户，并对返回数据格式进行封装（该对象已用一个Java类封装）

AjaxJson j = new AjaxJson();

UserEntity user = new UserEntity();

user.setUsername(username);

user.setPassword(password);

user = this.systemService.getUserByNameAndPassword(user);

（2）判断是否存在该用户

if(user == null){

j.setSuccess(false);

j.setMsg("用户名或密码错误！");

return j;

}

（3）获取该用户的资源，并将用户资源放入在一个list中

List<ResourceEntity> resourceList = new ArrayList<ResourceEntity>();

List<RoleEntity> roleList = user.getRoles();

for(RoleEntity re : roleList){

List<ResourceEntity> tempRes = re.getResource();

for(ResourceEntity res : tempRes){

if(!resourceList.contains(res)){

resourceList.add(res);

}

}

}

（4）对该用户信息进行封装

Client client = new Client();

client.setIp(ResourceUtil.getIpAddr(req));

client.setLogindatetime(new Date());

client.setUser(user);

client.setMenuList(resourceList);

ClientManager.getInstance().addClinet(session.getId(),client);

（5）返回结果

if(user != null && user.getId() != null){

j.setSuccess(true);

j.setMsg("登陆成功！");

}else{

j.setSuccess(false);

j.setMsg("用户名或密码错误!");

}

return j;

前端界面接收到doCheck方法返回来的数据后，对数据信息判断，根据EasyUI的用法如果返回来的信息是成功，则会跳转表单中action的地址，否则就将错误的信息打印到界面，具体的代码如下：

<form action="loginController.do?doLogin" method="post" check="loginController.do?doCheck" >

<input type="text" id="username" name="username" placeholder="用户名" >

<input type="password" id="password" name="password" placeholder="密码">

<br/>

<div id="errormsg" style="color: red"></div>

<input id="login" type="button" style="width:100%" value="登陆">

</form>

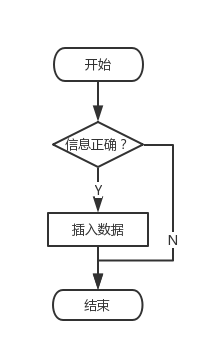
若后端返回来的是成功，就直接进入系统，否则就会在id为errormsg的控件内显示错误信息。具体的界面如图5.9所示：

****

**图5.9 登录失败图**

# **5.2.3 成绩录入**

在成绩录入的这个功能中，我们需要在Controller层接收前端界面传来的参数，由于成绩录入的信息中包含了学生ID和教师ID，所以我们需要在控制层判断该条数据是否在数据库中存在，具体的流程如图5.10所示：



**图5.10 成绩录入流程图**

如流程所示，当信息正确返回为真的时候则对数据进行插入，否则直接抛出异常结束，在录入的过程中，由于我们使用的是SpringMVC三层架构的模式，所以程序应该要进入以下几个步骤：

（1）首先对返回数据进行封装并初始化

AjaxJson j = new AjaxJson();

j.setMsg("保存成功！");

j.setSuccess(true);

（2）调用本控制层引用业务层的get方法获取教师信息和学生信息

TeacherEntity teacher = this.scoreService.get(TeacherEntity.class, teacherid);

StudentEntity student = this.scoreService.get(StudentEntity.class, studentid);

（3）业务层的get方法获取教师信息,而业务层中不存在get方法，但是该业务层是继承与SystemService这个类，而这个类中也没有该方法，只有在其父类继续查找，最后当找到BaseServiceImpl这个类的时候才发现这个方法，该方法继续调用了该层引用的持久层的get方法

public <T> T get(Class<T> entityClass,String id) {

return this.baseDao.get(entityClass,id);

}

（4）持久层get方法，由于本系统引用的hibernate框架，所以只需要用this.getSession().get(实体类,实体ID)方法就可以获取

public <T> T get(Class<T> entityClass, String id) {

return (T) this.getSession().get(entityClass, id);

}

（5）得到实体后，对成绩实体设置教师和学生信息

scoreEntity.setTeacherEntity(teacher);

scoreEntity.setStudentEntity(student);

（6）调用本控制层引用业务层的save方法

this.scoreService.save(scoreEntity)

（7）业务层save方法，同3步相同直到找到save方法为止

public <T> void save(T entity){

this.baseDao.save(entity);

}

（8）持久层save方法，同4步一致，同时加入保存判断，若成功则能够正常插入数据进数据库并运用flush()方法清空缓冲区数据，否则抛出异常

public <T> void save(T entity) {

try{

this.getSession().save(entity);

this.getSession().flush();

}catch(RuntimeException e){

logger.error("保存实体异常",e);

throw e;

}

}

（9）对控制层的代码进行try-catch包裹并返回结果

try {

TeacherEntity teacher = this.scoreService.get(TeacherEntity.class, teacherid);

StudentEntity student = this.scoreService.get(StudentEntity.class, studentid);

scoreEntity.setTeacherEntity(teacher);

scoreEntity.setStudentEntity(student);

this.scoreService.save(scoreEntity);

} catch (Exception e) {

j.setMsg("保存失败！");

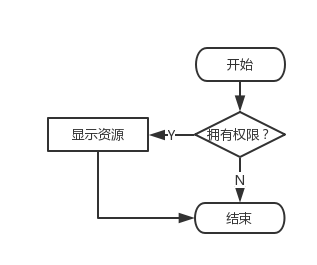
j.setSuccess(false);

}

return j;

# **5.2.4 权限控制**

权限控制实质上就是是否显示系统资源的一个功能，当用户登录系统成功后就需要查询一边该用户拥有的资源，然后对资源的展示过程，具体的流程图如图5.11所示：

****

**图5.11 获取权限图**

本系统的权限控制主要分为菜单权限和按钮权限，而菜单权限是只是通过左侧的树进行数据筛选的一个结果，然后将这个结果数据进行封装成前端树形需要的数据并返回给前端，前端接收到数据后对树形的一个展示。由于权限需要在登录的时候就进行控制，所以本系统选择在登录成功的时候就对权限进行筛选，所以该方法就放在了登录功能的控制层中。在获取菜单的过程中需要调用getTreeMenu方法并按照以下步骤进行：

（1）获取当前用户，并初始化一个资源容器

Client client = ResourceUtil.*getClient*();

List<ResourceEntity> resourceList = new ArrayList<ResourceEntity>();

（2）判断当前用户是否存在，若存在就获取用户拥有的资源，否则返回登录页。并初始化一个List对象用于存放初始值为‘1’的资源

if(client == null || client.getUser() == null){

return "system/login";

}else{

resourceList = client.getMenuList();

}

List<ResourceEntity> resource = new ArrayList<ResourceEntity>();

（3）对用户信息判断，若小于0则终止，否则就添加ID为‘1’的资源到resource这个容器里面并终止

for(ResourceEntity re:resourceList){

if(resourceList.size()<=0){

break;

}

if("1".equals(re.getId())){

resource.add(re);

break;

}

}

（4）调用resourceToTreeNode方法，将sysResource类型的数据集合转化为前端较好识别的TreeNode

resourceToTreeNode(resource,resourceList)

（5）在方法中先对数据进行判断非空且资源类型为菜单按钮（TYPE\_MENU已在资源实体表中设置为菜单类型，按钮类型为TYPE\_BTN），否则返回空对象

**if** (resource != **null** && !resource.isEmpty() && resource.get(0).getResourceType() == ResourceEntity.TYPE\_MENU)

（6）若以上方法合法则初始化一个list用于存放最后用户拥有的资源，对传入的第二个参数进行循环，若循环结束则返回参数ch

List<TreeNode> ch = new ArrayList<TreeNode>();

for (ResourceEntity rr : resource){}

return ch

（7）在循环中初始化一个TreeNode类型的对象（已用一个Java文件将该类型格式进行封装），并判断传入的第一个参数是否存在于第二个参数中

TreeNode node = new TreeNode();

if(userResource.contains(rr)){}

（8）如果存在则对node对象添加href字段进行数据填充封装，并将node存放在用户资源ch对象中

node.setId(rr.getId());

node.setState("open");

node.setText(rr.getName());

Map<String, Object> \_temp = new HashMap<String, Object>();

\_temp.put("href", rr.getHref());

node.setAttributes(\_temp);

ch.add(node);

（9）调用本方法获取node的子节点并设置为node的setChildren属性，直到循环结束返回ch

node.setChildren(resourceToTreeNode(rr.getResources(),userResource));

（10）循环结束后，getTreeMenu对获取到的ch转换为字符串类型并返回给后台

return JSONObject.valueToString(resourceToTreeNode(resource,resourceList));

对于按钮的权限，我们则需要特定的字段来区分按钮的功能，所以在数据库中我设计了一个编码（CODE）的字段且是唯一不能重复的，所以需要在调用每个显示界面的时候对权限进行一次查询。以成绩管理菜单为例，点击成绩管理节点，界面上新增一个tab标签页，然后加载系统资源，在加载系统资源的时候查询当前用户是否拥有成绩管理界面中按钮功能的权限。先由管理员进入系统会发现所有资源都由权限操作如图5.12所示，然后换只有部分权限的学生A进入时就会只出现该用户拥有的权限如图5.13所示：



**图5.12 管理员权限图**

****

**图5.13 学生权限图**

由于需要获取到当前用户的信息，所以本系统采用了直接获取在线用户信息的方式来进行后续的工作，具体实现获取在线用户信息的方法请查看源代码Client.Java文件。所以后端就只在Controller层进行处理，在处理按钮权限控制的过程中，需要调用goScore方法并按照以下步骤进行：

（1）获取当前用户并初始化一个名为resourceList的list对象用户存放用户资源

Client client = ResourceUtil.*getClient*();

List<ResourceEntity> resourceList = new ArrayList<ResourceEntity>();

（2）判断用户是否存在，若存在就获取资源，否则返回登录界面。再初始化一个名为ch的list对象用于存放按钮编码

if(client == null || client.getUser() == null){

return new ModelAndView("system/login");

}else{

resourceList = client.getMenuList();

}

List<String> ch=new ArrayList<String>();

（3）判断资源不为空且类型为按钮类型，若为true则将对资源进行遍历所取得的code编码存放在ch中

for(ResourceEntity rr:resourceList){

String code=rr.getCode();

ch.add(code);

}

（4）对返回值设置code属性并返回渲染类型

request.setAttribute("code", ch);

return new ModelAndView("buss/score");

（5）前端jsp中对返回的code进行遍历

<c:forEach var=*"i"* items=*"*${code}*"*></c:forEach>

（6）在遍历中对字段编码进行比较，若为true则显示控件，否则不显示

<c:forEach var=*"i"* items=*"*${code}*"*>

<c:if test=*"*${i=='scoreadd'}*"*>

<a href=*"#"* class=*"EasyUI-linkbutton add"* iconCls=*"icon-add"* onclick="create()" plain=*"true"*>新增</a>

</c:if>

<c:if test=*"*${i=='scoreedit'}*"*>

<a href=*"#"* class=*"EasyUI-linkbutton edit"* iconCls=*"icon-edit"* onclick="edit()" plain=*"true"*>修改</a>

</c:if>

<c:if test=*"*${i=='delescore'}*"*>

<a class=*"EasyUI-linkbutton remove"* iconCls=*"icon-remove"* onclick="del()" plain=*"true"*>删除</a>

</c:if>

<c:if test=*"*${i=='importscore'}*"*>

<a class=*"EasyUI-linkbutton save"* iconCls=*"icon-save"* onclick="upload()" plain=*"true"*>excel导入</a>

</c:if>

<c:if test=*"*${i=='exportscore'}*"*>

<a class=*"EasyUI-linkbutton back"* iconCls=*"icon-back"* onclick="expoExcel()" plain=*"true"*>excel导出</a>

</c:if>

</c:forEach>

# **6 系统设计的关键技术**

# **6.1 Java**

1. 简单，Java简单，虽然是有些东西让人反复的定义编写，但是语法不多，以简单的语法执行各种逻辑处理，目前市场上的逻辑问题都是能够解决的。只是最新出来的Java 8才有点难度。

1. J2EE/Spring 虽然初始配置繁琐，但是只要首次配置完成，后续使用还是很简单。加上Java是强类型静态语言，项目大了之后所带来的优越性是其他动态语言完全无法比较的。所以它带来的最大好处是，就算代码中没有注释或者注释了也看不懂的情况下，也能够保证基本的代码质量。这个优势静态语言都拥有，当然就包括了Java。

3. 库多而且质量普遍有保障，很多东西不需要自己实现

4. Java的开发工具Eclipse，虽然被很多人吐槽，但是他的开放性和免费是作大型商业开发的IDE中应该算最好的了。而且用起来也是简洁方便。其中对于刚刚入行的程序员而言Eclipse的代码自动提示，错误提示，自动格式化等是一个不错的选择。

# **6.2 MySQL**

1. 借鉴互联网产品，例如淘宝，底层的关系数据库都是使用MySQL 。

1. MySQL 是开源数据库，提供的接口支持多种语言连接操作
2. MySQL 支持不同的存储引擎，可以根据业务的需要选择不同的引擎。

# **6.3 EasyUI**

1.使用 EasyUI 你不需要写很多的 Javascript 代码，通常只需要写 HTML 标记来定义用户界面即可；

2.为一些当前用于交互的 js 应用提供必要的功能；

3.基于 jquery 用户界面插件的集合；

4.支持 HTML5；

5.开发产品时可节省时间和资源；

6.简单，但很强大；

7.支持扩展，可根据自己的需求扩展控件；

8.目前各项不足正已版本递增的方式不断完善；

# **6.4 Tomcat**

1. Tomcat是一款小型的且轻量级的应用服务器，能够支持负载平衡、邮箱服务等在开发应用系统过程中很常用的功能，同时在运行的时候占用系统空间资源少，扩展性比较好等优势，并且还一直在不断的改进和完善。
2. Tomcat能够在中小型系统和同时访问量不是很多也就是并发访问用户不太多的情况下能发挥显著作用。本系统使用Java+jsp的前后端分离模式进行开发的中小型系统，所以tomcat非常适合我们当前的模式。
3. Tomcat具有稳定的性能、免费获取、性能稳定和源码开发等机制，所以在Java开发的这一块中深受Java爱好者的欢迎并且很多软件的开发商都很认同这种服务器，也就成就了tomcat成为市场上相当流行的应用服务器。

# **7 系统试运行结果与评价**

经测试和试运行，其功能已经基本完成。总结出该系统有以下几个特色：

1. 做了动态权限控制；管理员可以根据具体的用户进行权限设置，其中包括了菜单和按钮两种资源，设置成功后就形成了相应的权限。
2. 操作简单，条目鲜明；系统在菜单一栏用树的形式展现，点击节点出现相对应的内容且节点说明一目了然。
3. 系统可移植性强；后端采用Java语言编写，可以将程序导出为以jar为后缀格式的文件在安装有Java的虚拟机上进行部署。

除以上三点特点外还是有些不足之处及其改进：

本系统只是在用户权限管里的这块上面着重开发，而对于学生成绩这块的处理只是呈现出基础功能的层面，加上时间和精力的限制，界面的效果呈现的不是很美观，只是用了EasyUI的原生样式，看上去有些差强人意。本系统也采用了非数据库作为数据源的数据，如班级和学科，当时考虑到单独建立两个表存放独立实体对数据库依赖太多，所以就采用了用json格式的数据代替了数据源的功能。 还有就是导入成绩对格式的要求没有精确的判断，导致导入过程过于严格。

# **结束语**

通过本次的论文设计开发，我从中学到了许多的知识，在一开始只是对数据库编程有一个抽象的概念，像这样从设计到自己实现都是很少接触到，所以在一开始的时候我只有跟着参考文献中的案例和网上的教程进行学习开发，使用eclipse时和大部分新入行的程序员一样，对其配置和框架的使用，路径的选择上面纠结了很久，以至于迟迟不能开始进入编程实现，但是后来完成这一系列的配置后就能轻松的应付各个插件的配置。

进入编程实现功能阶段，很多次碰见了瓶颈，比如用在ssh框架的时候何时应该自己写SQL语句而何时又该调用本身自己提供的方法这个问题上纠结了一天，当自己意识到其实这是自己对框架不熟悉造成的时候果断在对ssh框架进行了再一次的学习，学习中又收获了第一遍没有领会到的知识点。有了前车之鉴后每有类似的问题我都和再次学习以解决问题。

本系统虽然看起来还是比较简陋格格不入，但是这都是自己的想法自己独立完成，所以满满的成就感油然而生，毕竟这是通过自己的努力完成的一套较完整的系统，在开发实现中自己也学习了很多的知识，接触了很完整的数据库编程，能够独立的开发完简单的数据库系统，是我的变成能力得到了提高，增加了自己的编程经验。

# **参考文献**

[1]李书杰,李志刚.B/S三层体系结构模式[J].河北工学院院报,2002,24(1):2

[2]林海.浏览器/服务器应用开发[M].北京:科学出版社,2000

[3]康博,马亮.Web应用程序开发新技术[M].北京:人民邮电出版社,1999

[4]张海藩.软件工程导论[M].北京:清华大学出版社,1998

[5] 冯燕奎, 赵德奎. JSP实用案例教程[M]. 清华大学出版社,2004.5

[6] 林信良. spring2.0技术手册[M]. 电子工业出版社,2006.6

[7] 王海涛,贾宗璞. 基于Struts和Hibernate的Web应用开发[J].计算机工程,2011

[8] 李峰,刘彦隆. 基于SSH框架与jquery技术的Java Web开发应用[J].科技情报开发与经济,2010

[9] 刘京华. Java Web整合开发王者归来[M].北京：清华大学出版社，2010

[10] 孙卫琴，李洪成. Tomcat 与 Java Web 开发技术详解[M]. 电子工业出版社，2003.6