**上海电力学院**

**JavaEE实验报告**



实验题目: 计算机应用能力大赛报名系统

院　　系： 　　 计算机科学与技术学院

专 业： 　 计算机科学与技术专业（创新创业）

班 级： 2014055

学 号： 20140685

学生姓名： 郭浩明

2017年　6 月　7日

目录

[1、前言 1](#_Toc1131)

[2、需求分析 1](#_Toc28929)

[2.1系统功能性需求 1](#_Toc18721)

[2.2参与者 1](#_Toc29129)

[2.3用例 2](#_Toc1195)

[2.4用例图 2](#_Toc1445)

[2.5用例规约 3](#_Toc30020)

[2.5.1 注册用例规约 3](#_Toc12402)

[2.5.2 登录用例规约 3](#_Toc31674)

[2.5.3发布新闻用例规约 3](#_Toc4764)

[2.5.4学生报名用例规约 4](#_Toc22071)

[2.5.5学校报名用例规约 4](#_Toc23375)

[2.5.6编辑报名信息用例规约 5](#_Toc25150)

[2.5.7查看报名信息用例规约 5](#_Toc14591)

[2.5.8参赛作品上传用例规约 5](#_Toc10111)

[2.5.9修改个人信息用例规约 6](#_Toc9061)

[3、 概要设计 6](#_Toc14968)

[3.1总体设计 6](#_Toc29213)

[3.2运行环境 6](#_Toc15617)

[3.3 系统体系结构 7](#_Toc3471)

[3.4技术结构 7](#_Toc13351)

[3.5功能架构 8](#_Toc13258)

[3.6界面设计 9](#_Toc26124)

[3.7数据库设计 10](#_Toc22898)

[3.7.1概念数据模型 10](#_Toc18345)

[3.7.2数据逻辑结构 10](#_Toc12026)

[3.7.3数据库表结构 11](#_Toc410)

[3.7.4数据库视图 12](#_Toc21684)

[3.7.5数据库创建脚本 13](#_Toc3172)

[4、详细设计 19](#_Toc23694)

[4.1MVC模式实现 19](#_Toc10195)

[4.2模块设计（以登录为例） 20](#_Toc28265)

[5、系统创新 21](#_Toc20068)

[5.1后端部分创新 21](#_Toc19565)

[5.2前端的创新 23](#_Toc3283)

[6、课程总结 26](#_Toc7886)

[6.1遇到的问题 26](#_Toc10053)

[6.2个人总结 27](#_Toc27351)

# 1、前言

上海市计算机应用能力大赛旨在培养大学生的创新精神和创新实践能力，调动学生学习计算机技术的积极性，提高运用信息技术解决实际问题的综合水平。本系统是为了方便上海市计算机应用能力而设计的，通过该系统，实现参赛队伍报名，参赛队伍管理，新闻发布等功能，更加方便地管理各种相关信息，为上海市计算机应用能力更好地服务。

# 2、需求分析

在该系统中，访客可以浏览到计算机应用能力大赛的相关新闻信息以及参赛报名的信息。学生注册成为普通用户，从而提交报名信息和作品；学校负责人能将本学校的报名信息进行汇总，分类，统计；管理员负责管理所有参赛队伍的作品和报名情况，除此之外还负责新闻的发布工作。

## 2.1系统功能性需求

本系统主要分为三种用户：

1、普通用户，可以查看浏览报名信息

2、学校负责人：负责学校参赛作品的报名，统计，汇总等功能

3、系统管理员：负责所有学校参赛作品的报名，统计，汇总等功能。

主要功能模块描述如下：

1. 新闻发布管理。即发布、删除、编辑新闻公告等。

2、学校负责人基本信息管理

（1）主要是为了以学校为单位收集报名资料，收集后由学校负责人统一报名。

（2）还应该指定一位学校负责人，负责汇总统计报名信息。

（3）能够将网站报名信息下载导入到Excel 文件中。

3、报名信息管理、统计、汇总等。

## 2.2参与者

根据分析，确定该系统参与者如下：

|  |  |
| --- | --- |
| 参与者名称 | 参与者描述 |
| 普通用户 | 要是学生，能进行参赛报名，作品提交 |
| 学校负责人 | 负责统计和编辑本校参赛作品以及为本校参赛队伍报名 |
| 系统管理员 | 负责统计和编辑所有报名作品以及负责新闻发布 |

表2-1 参与者

## 2.3用例

通过系统功能分析，确定系统主要用例如下：

|  |  |  |
| --- | --- | --- |
| 用例名称 | 用例描述 | 预期参与者 |
| 注册 | 该用例描述访客注册成为普通用户 | 访客 |
| 登录 | 该用例描述访客登录 | 访客 |
| 发布新闻 | 该用例描述系统管理员发布新闻信息 | 系统管理员 |
| 学生报名 | 该用例描述普通用户报名参加比赛 | 普通用户 |
| 学校报名 | 该用例描述学校负责人为学校报名参加比赛 | 学校负责人 |
| 编辑报名信息 | 该用例描述系统管理员或学校负责人编辑信息 | 系统管理员，学校负责人 |
| 查看报名信息 | 该用例描述系统管理员或学校负责人查看信息 | 系统管理员，学校负责人 |
| 参赛作品上传 | 该用例描述普通用户上传参赛作品 | 普通用户 |
| 修改个人信息 | 该用例描述普通用户修改个人信息 | 普通用户 |

表2-2 用例

## 2.4用例图

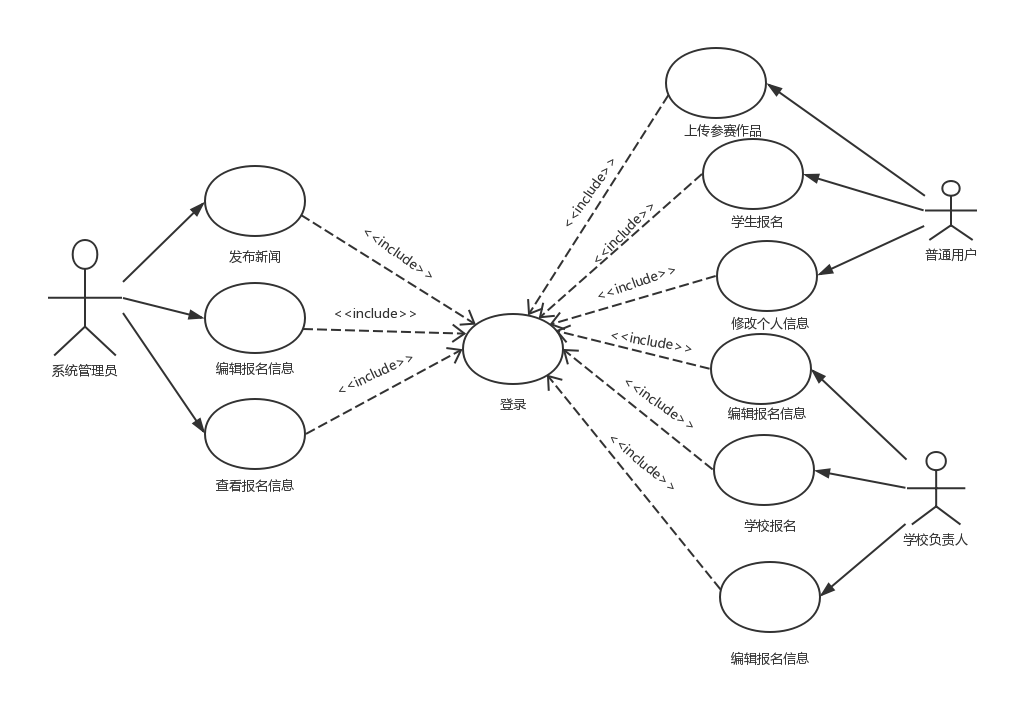


图2-1 系统用例图

## 2.5用例规约

### 2.5.1 注册用例规约

|  |  |
| --- | --- |
| 用例名称 | 注册 |
| 简要说明 | 允许访客注册成为用户 |
| 基本事件流 | 用户先请求注册，在注册界面中，用户输入密码以及注册信息，系统分配登陆号 |
| 备选事件流 | 1. 两次输入密码不一致：   弹出警告窗，要求用户重新输入   1. 输入信息不完整：   提示警告   1. 用户取消注册：   返回主界面 |
| 前置条件 | 无 |
| 后置条件 | 登陆 |

表2-3 注册用例规约

### 2.5.2 登录用例规约

|  |  |
| --- | --- |
| 用例名称 | 登录 |
| 简要说明 | 允许用户登录并进行后续操作 |
| 基本事件流 | 用户先请求登录，在注册界面中，用户输入用户名密码，系统进行验证，验证通过后登陆成功跳转到其他页面 |
| 备选事件流 | 1.用户名密码错误：  弹出错误窗口信息，要求用户重新输入   1. 用户名不存在：   弹出错误窗口信息，提示用户名不存在 |
| 前置条件 | 注册 |
| 后置条件 | 报名，作品上传等 |

表2-4 登录用例规约

### 2.5.3发布新闻用例规约

|  |  |
| --- | --- |
| 用例名称 | 发布新闻 |
| 简要说明 | 允许管理员用户在网页上发布新闻信息 |
| 基本事件流 | 管理员用户登录后，进入发布新闻模块，编写新闻的主要信息并发布到网上 |
| 备选事件流 | 1. 新闻信息不完整：   弹出提示框警告，要求重新输入 |
| 前置条件 | 登录 |
| 后置条件 | 无 |

表2-5 发布新闻用例规约

### 2.5.4学生报名用例规约

|  |  |
| --- | --- |
| 用例名称 | 学生报名 |
| 简要说明 | 允许学生报名参赛 |
| 基本事件流 | 学生登录后，进入报名请求页面，填写个人信息确认完成报名，系统自动为其生成一个参赛号 |
| 备选事件流 | 1.个人信息不完整：  弹出提示框警告，要求重新输入   1. 个人信息已注册：   弹出窗口提示，要求重新输入   1. 个人信息格式错误：   弹出提示框警告，要求重新输入   1. 有参赛队员没有注册   弹出提示框警告，提示注册 |
| 前置条件 | 登录 |
| 后置条件 | 修改个人信息，上传参赛作品等 |

表2-6 学生报名用例规约

### 2.5.5学校报名用例规约

|  |  |
| --- | --- |
| 用例名称 | 学校报名 |
| 简要说明 | 允许学校负责人为本校参赛作品报名 |
| 基本事件流 | 学校负责人登录后，进入学校报名模块，填写该校报名队伍信息和报名人员信息，确认提交 |
| 备选事件流 | 1. 参赛队员不存在：   弹出窗口提示错误   1. 参赛队伍不存在：   弹出窗口提示错误   1. 信息不完整：   弹出窗口提示错误，要求重新输入 |
| 前置条件 | 登录 |
| 后置条件 | 无 |

表2-7 学校报名用例规约

### 2.5.6编辑报名信息用例规约

|  |  |
| --- | --- |
| 用例名称 | 编辑报名信息 |
| 简要说明 | 允许学校负责人以及管理员为参赛作品编辑报名信息 |
| 基本事件流 | 学校负责人或管理员登录后，进入编辑报名信息模块，选择需要编辑的报名信息项目，确认后转入报名信息编辑。若选择删除报名信息则要经过一次确认，弹出对话框确认是否要删除信息。 |
| 备选事件流 | 1. 输入编辑信息格式错误：   弹出警告框，要求重新输入 |
| 前置条件 | 登录 |
| 后置条件 | 无 |

表2-8 编辑报名信息用例规约

### 2.5.7查看报名信息用例规约

|  |  |
| --- | --- |
| 用例名称 | 查看报名信息 |
| 简要说明 | 允许学校负责人以及管理员查看已报名的信息 |
| 基本事件流 | 学校负责人或管理员登录后，进入查看报名信息模块，选择需要查看的报名信息项目，确认后转入报名信息查看页面。 |
| 备选事件流 | 无 |
| 前置条件 | 登录 |
| 后置条件 | 无 |

表2-9 查看报名信息用例规约

### 2.5.8参赛作品上传用例规约

|  |  |
| --- | --- |
| 用例名称 | 参赛作品上传 |
| 简要说明 | 允许参赛队员上传个人参赛作品 |
| 基本事件流 | 普通用户登录后，进入参赛作品上传模块，选择需要上传的参赛作品文件，点击确认后上传参赛作品 |
| 备选事件流 | 1. 上传参赛作品过大：   弹出提示窗口提示错误   1. 本参赛队伍已有参赛作品上传：   弹出提示窗口询问是否覆盖已有作品 |
| 前置条件 | 登录 |
| 后置条件 | 无 |

表2-10 参赛作品上传用例规约

### 2.5.9修改个人信息用例规约

|  |  |
| --- | --- |
| 用例名称 | 修改个人信息 |
| 简要说明 | 允许参赛队员修改已经注册的个人信息 |
| 基本事件流 | 普通用户登录后，进入修改个人信息模块，修改已注册的个人信息 |
| 备选事件流 | 1.个人信息填写格式错误：  弹出提示窗口提示错误 |
| 前置条件 | 登录 |
| 后置条件 | 无 |

表2-11 修改个人信息用例规约

# 概要设计

## 3.1总体设计

计算机应用能力大赛管理系统主要供比赛管理方，参赛方使用，方便对比赛的各种信息进行浏览以及对比赛各种新闻的发布以及浏览。

组织结构主要分为三个管理层次：

1. 普通用户
2. 学校负责人
3. 系统管理员

（1）普通用户

普通用户主要是学生，能够报名参加比赛，普通用户可以注册、修改自己的相关信息。

（2）学校负责人

学校负责人以学校作为整体，可以浏览、修改本学校的参赛队伍以及参赛作品，并且负责将该学校的参赛作品上报到系统。

（3）系统管理员

系统管理员可以增加、修改、删除所有学校的参赛作品信息，并且对能够发布新闻

## 3.2运行环境

根据系统的使用范围来看，本系统有3个特点：

1. 系统覆盖地理范围广，没有对应的城域网做支持
2. 采用集中数据库管理
3. 业务覆盖范围大，系统用户多

根据这三个特点，对系统的总体结构（包括技术结构和功能结构）进行了设计

## 3.3 系统体系结构

该系统使用MVC模式，将逻辑功能分割为Model层，View层和Controller层三层。

（1）Model层

该层负责业务流程的处理，接受View层的请求数据并返回最终处理结果。

（2）View层

该层负责用户交互界面，用户在该层上将数据提交给服务器端。

（3）Controller层

该层负责从用户处接受请求，将Model与View匹配，共同完成用户请求。

系统体系结构如图3-1:

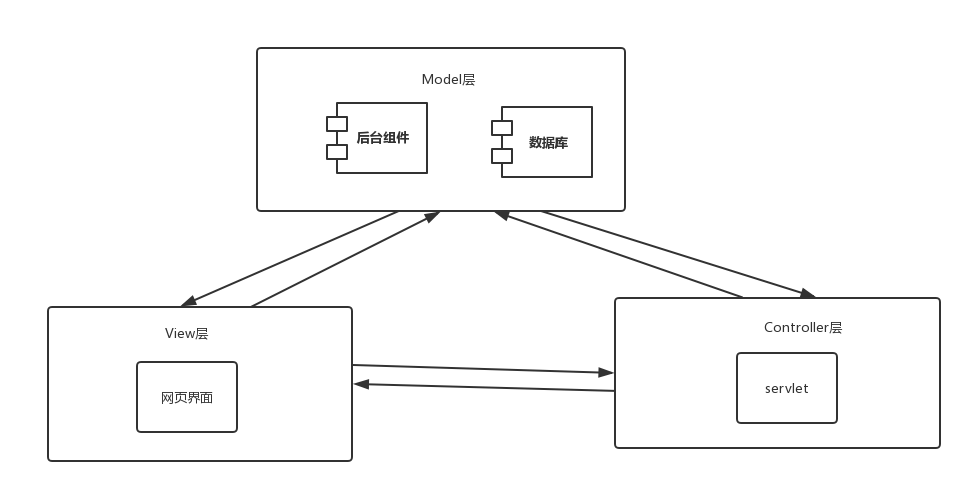


图3-1系统体系结构图

## 3.4技术结构

为了满足计算机应用能力大赛的应用需求，使得没有专线的情况下使用统一的数据库进行业务处理，并进行灵活的查询统计分析，本系统采用B/S结构作为系统的解决方案。系统结构如图3-1所示：

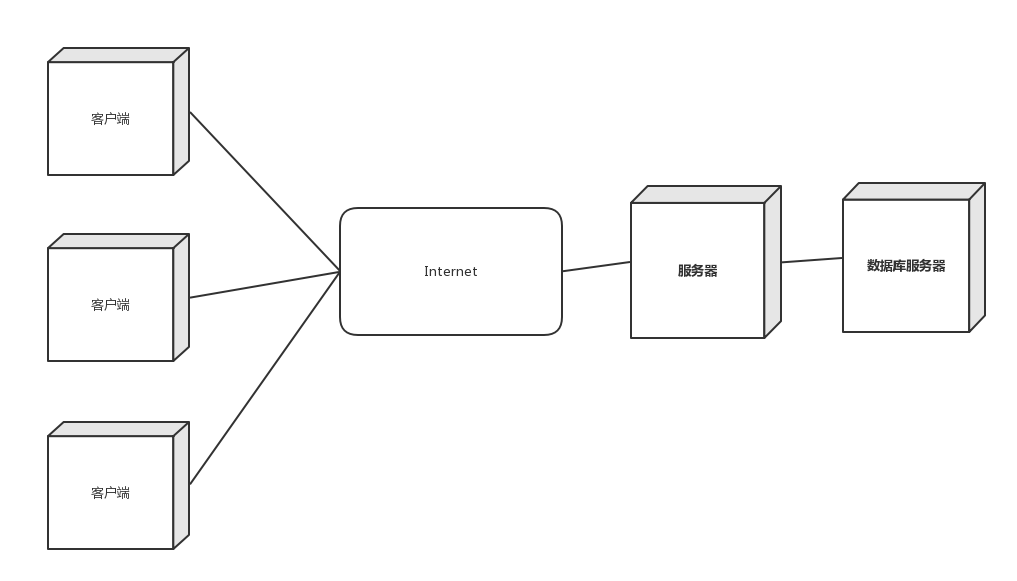


图3-2 系统架构图

如图3-1，系统分为3个部分：数据库、服务器、浏览器客户端。所有用户通过浏览器访问应用服务器，服务器与数据库交互获得需要的数据。

1、采用MySQL2008作为数据库存贮数据。

2、采用JSP/Servelet/JavaBean技术作为系统的解决方案，使得系统具有更好的扩展性。

3、客户端采用浏览器形式，满足客户端的易用性需求。

## 3.5功能架构

系统的功能结构图如图3-2。

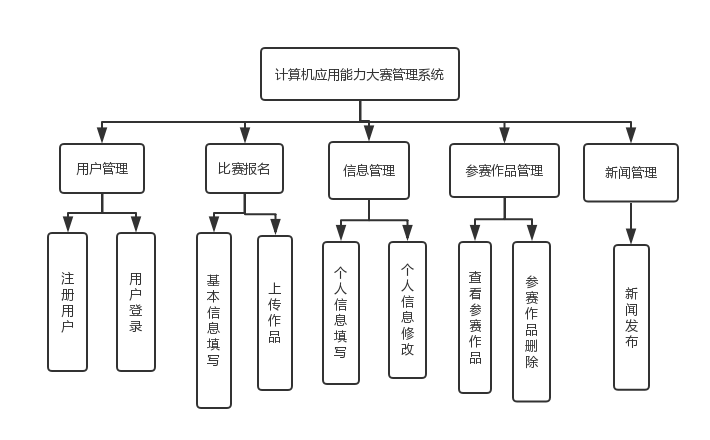


图3-3系统功能结构图

|  |  |
| --- | --- |
| 模块名 | 描述 |
| 注册用户 | 该模块处理访客注册成为普通用户 |
| 用户登录 | 该模块处理访客登录成为各种各样不同权限用户 |
| 基本信息填写 | 该模块处理普通用户完善比赛信息 |
| 上传作品 | 该模块处理报名后上传参赛作品 |
| 个人信息填写 | 该模块处理普通用户完善个人基本信息 |
| 个人信息修改 | 该模块处理普通用户修改个人基本信息 |
| 查看参赛作品 | 该模块处理学校负责人和系统管理员查看参赛作品 |
| 参赛作品删除 | 该模块处理学校负责人和系统管理员删除参赛作品 |
| 新闻发布 | 该模块处理系统管理员发布新闻 |

表3-1模块功能描述

## 3.6界面设计

根据系统功能要求，界面的概要设计如图3-3所示

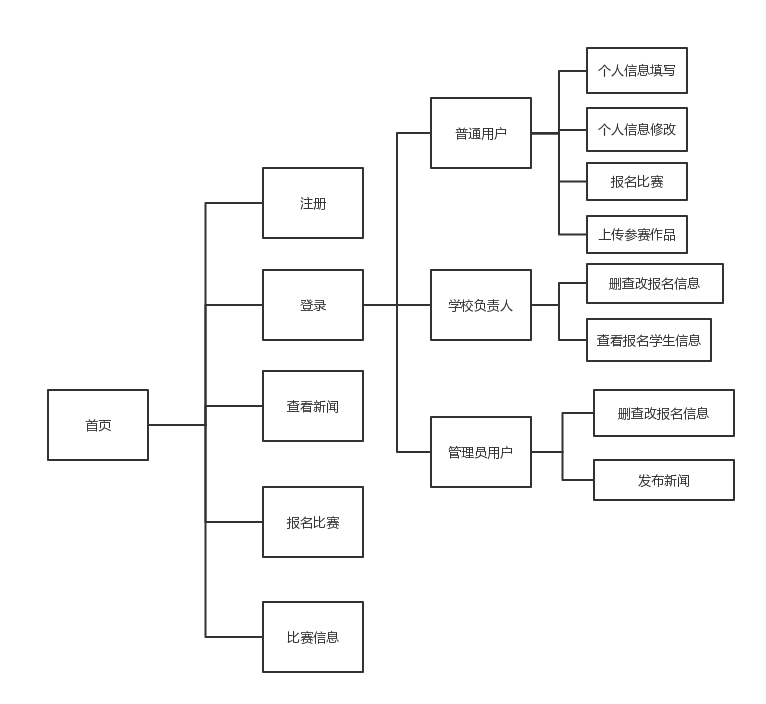


图3-4 界面设计图

## 3.7数据库设计

### 3.7.1概念数据模型

根据系统需求分析，确定系统ER图如图3-5

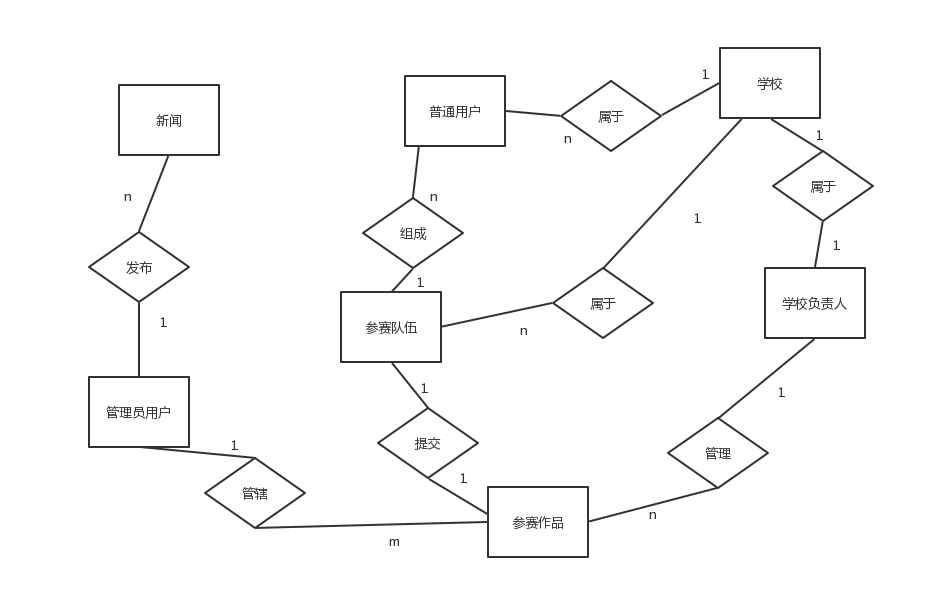


图3-5 ER图

### 3.7.2数据逻辑结构

普通用户（登录名，身份证，密码，姓名，性别，专业，学院，年级，学号，联系电话，email，参赛队伍编号，学校编号）

参赛队伍（队伍编号，队名，指导老师姓名，指导老师电话，学校编号，作品编号）

学校（学校编号，学校名，学校类型）

学校负责人（学校负责人编号，密码，姓名，电话）

参赛作品（作品编号，学校负责人编号，作品描述，作品类别，管理员用户编号）

管理员用户（管理员用户编号，密码，姓名，电话）

新闻（新闻编号，管理员用户编号，新闻内容）

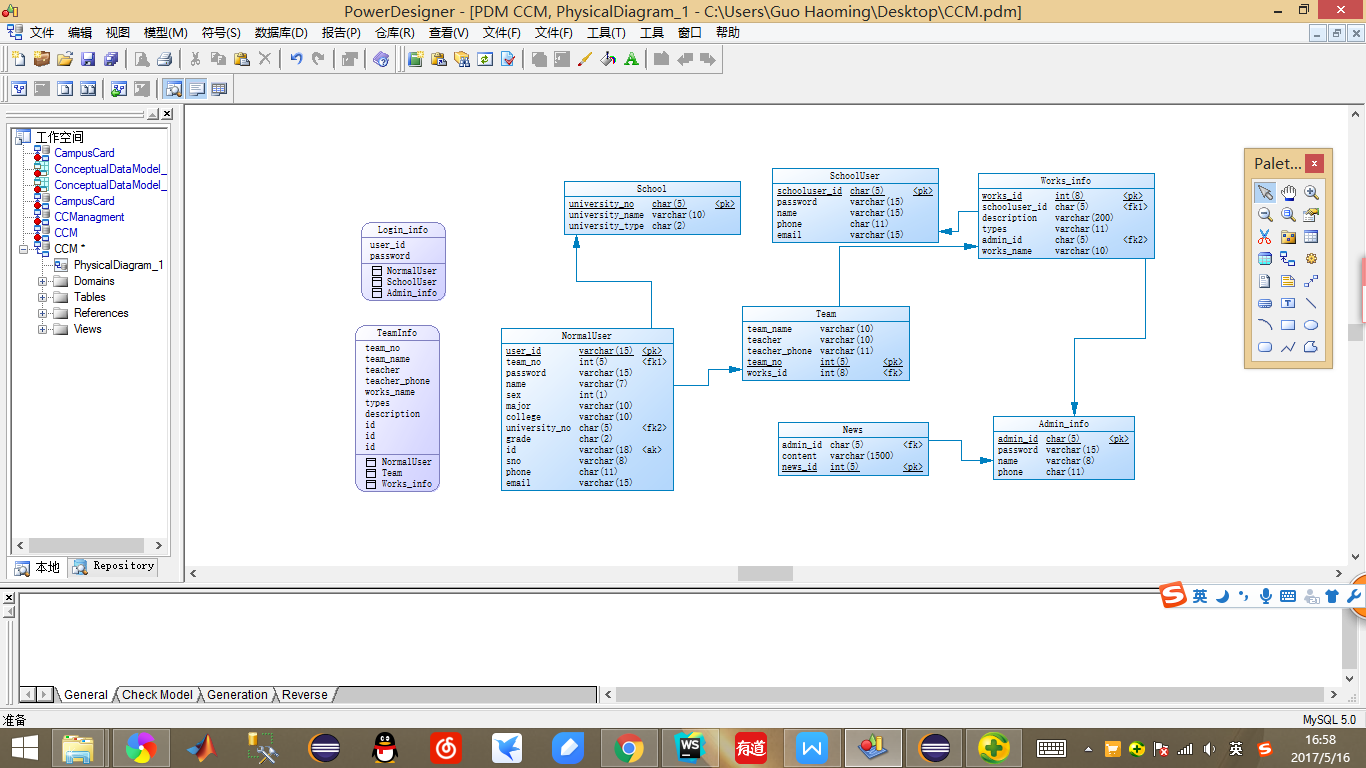


图3-6数据库结构图

### 3.7.3数据库表结构

普通用户表——NormalUser

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 类型 | KEY | 允许空 | 备注 |
| user | 登录名 | varchar(15) | p |  |  |
| password | 密码 | varchar(15) |  |  |  |
| name | 姓名 | varchar(7) |  | 是 |  |
| sex | 性别 | int(1) |  | 是 | 男—1，女—0 |
| major | 专业 | varchar(10) |  | 是 |  |
| college | 学院 | varchar(10) |  | 是 |  |
| university\_no | 学校编号 | char(5) | F | 是 |  |
| grade | 年级 | char(2) |  | 是 |  |
| id | 身份证 | varchar(18) |  | 是 | 唯一性约束 |
| sno | 学号 | varchar(8) |  | 是 |  |
| phone | 电话 | char(11) |  | 是 |  |
| email | email | varchar(15) |  | 是 |  |
| team\_no | 队伍号 | char(5) | F | 是 | 未报名为空 |

表3-2普通用户表

参赛队伍表——Team

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 类型 | KEY | 允许空 | 备注 |
| team\_no | 队伍号 | char(5) | p |  | 自动分配 |
| team\_name | 队名 | varchar(10) |  |  |  |
| teacher | 指导老师 | varchar(10) |  |  |  |
| teacher\_phone | 指导老师电话 | varchar(11) |  |  |  |
| university\_no | 学校编号 | char(5) | F |  |  |
| works\_id | 作品编号 | char(5) | F | 是 |  |

表3-3普通用户表

学校表——School

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 类型 | KEY | 允许空 | 备注 |
| university\_no | 学校编号 | char(5) | p |  | 与该学校负责人编号相同 |
| university\_name | 学校名 | varchar(10) |  |  |  |
| university\_type | 学历 | char(1) |  |  | 本科—0，专科—1 |

表3-4学校表

学校负责人表——SchoolUser

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 类型 | KEY | 允许空 | 备注 |
| schoooluser\_id | 负责人编号 | char(5) | p |  | 与该学校编号相同 |
| password | 密码 | varchar(15) |  |  | 初始与编号相同 |
| name | 姓名 | varcahr(8) |  |  |  |
| phone | 电话 | char(11) |  |  |  |
| email | email | varchar(15) |  |  |  |

表3-5学校负责人表

参赛作品表——Works\_info

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 类型 | KEY | 允许空 | 备注 |
| works\_id | 作品号 | char(8) | p |  | 自动分配 |
| schoooluser\_id | 负责人编号 | char(5) | F |  | 与学校编号同 |
| description | 作品描述 | varcahr(200) |  |  |  |
| types | 作品类别 | varchar(11) |  |  |  |
| admin\_id | 管理员id | char(5) | F |  |  |
| works\_name | 作品名 | varchar(10) |  |  |  |
| path | 路径 | varchar(30) |  | 是 | 未上传为空 |

表3-6参赛作品表

管理员表——Adminuser

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 类型 | KEY | 允许空 | 备注 |
| admin\_id | 管理员id | char(5) | p |  |  |
| password | 密码 | varchar(15) |  |  |  |
| name | 姓名 | varcahr(8) |  |  |  |
| phone | 电话 | char(11) |  |  |  |

表3-7管理员表

新闻表——News

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 字段描述 | 类型 | KEY | 允许空 | 备注 |
| news\_id | 新闻id | char(5) | p |  | 自动生成 |
| admin\_id | 管理员id | char(5) | F |  |  |
| content | 新闻内容 | varcahr(1500) |  |  |  |

表3-8管理员表

### 3.7.4数据库视图

参赛队伍作品视图——TeamInfo

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段描述 | 备注 |
| team\_no | 队伍id |  |
| team\_name | 队伍名 |  |
| teacher | 指导老师 |  |
| teacher\_phone | 指导老师联系方式 |  |
| works\_name | 作品名 |  |
| works\_type | 作品类型 |  |
| description | 作品描述 |  |
| id1 | 队员1身份证 |  |
| id2 | 队员2身份证 |  |
| id3 | 队员3身份证 |  |
| school | 所属学校 |  |

表3-5用户名密码视图

描述：该视图用于描述作品，队伍和用户之间的联系，便于管理员用户和学校负责人浏览。

### 3.7.5数据库创建脚本

drop table if exists Login\_info;

drop table if exists TeamInfo;

drop table if exists Admin\_info;

drop table if exists News;

drop table if exists NormalUser;

drop table if exists School;

drop table if exists SchoolUser;

drop table if exists Team;

drop table if exists Works\_info;

/\*==============================================================\*/

/\* Table: Admin\_info \*/

/\*==============================================================\*/

create table Admin\_info

(

admin\_id char(5) not null,

password varchar(15) not null,

name varchar(8) not null,

phone char(11) not null,

primary key (admin\_id)

);

/\*==============================================================\*/

/\* Table: News \*/

/\*==============================================================\*/

create table News

(

admin\_id char(5) not null,

content varchar(5000) not null,

news\_id int(5) not null auto\_increment,

primary key (news\_id)

)

engine=INNODB auto\_increment=10000;

;

/\*==============================================================\*/

/\* Table: NormalUser \*/

/\*==============================================================\*/

create table NormalUser

(

user\_id varchar(15) not null,

team\_no int(5),

password varchar(15) not null,

name varchar(7),

sex int(1),

major varchar(10),

college varchar(10),

university\_no char(5),

grade char(2),

id varchar(18),

sno varchar(8),

phone char(11),

email varchar(25),

primary key (user\_id),

unique key AK\_Key\_2 (id)

);

/\*==============================================================\*/

/\* Table: School \*/

/\*==============================================================\*/

create table School

(

university\_no char(6) not null,

university\_name varchar(14) not null,

university\_type char(1) not null,

primary key (university\_no)

);

/\*==============================================================\*/

/\* Table: SchoolUser \*/

/\*==============================================================\*/

create table SchoolUser

(

schooluser\_id char(5) not null,

password varchar(15) not null,

name varchar(15) not null,

phone char(11) not null,

email varchar(25) not null,

primary key (schooluser\_id)

);

/\*==============================================================\*/

/\* Table: Team \*/

/\*==============================================================\*/

create table Team

(

team\_name varchar(10) not null,

teacher varchar(10) not null,

teacher\_phone varchar(11) not null,

team\_no int(5) not null auto\_increment,

works\_id int(8),

primary key (team\_no)

)

engine=INNODB auto\_increment=10000;

;

/\*==============================================================\*/

/\* Table: Works\_info \*/

/\*==============================================================\*/

create table Works\_info

(

works\_id int(8) not null auto\_increment,

schooluser\_id char(5) not null,

description varchar(200) not null,

types varchar(11) not null,

admin\_id char(5) not null,

works\_name varchar(10) not null,

primary key (works\_id)

)

engine=INNODB auto\_increment=1000000;

/\*==============================================================\*/

/\* View: TeamInfo \*/

/\*==============================================================\*/

create VIEW TeamInfo

AS

SELECT SQL\_NO\_CACHE

`k`.`team\_no` AS `team\_no`,

MAX((CASE

WHEN (`k`.`rowno` = 2) THEN `k`.`id`

ELSE 0

END)) AS `id1，

MAX((CASE

WHEN (`k`.`rowno` = 0) THEN `k`.`id`

ELSE 0

END)) AS `id2`,

MAX((CASE

WHEN (`k`.`rowno` = 1) THEN `k`.`id`

ELSE 0

END)) AS `id3`,

`k`.`university\_no` AS `school`,

`k`.`team\_name` AS `team\_name`,

`k`.`teacher` AS `teacher`,

`k`.`teacher\_phone` AS `teacher\_phone`,

`k`.`works\_name` AS `works\_name`,

`k`.`types` AS `types`,

`k`.`description` AS `description`

FROM

`view2` `k`

GROUP BY

`k`.`team\_no`

/\*==============================================================\*/

/\* View: View2 \*/

/\*==============================================================\*/

VIEW `view2` AS

SELECT SQL\_NO\_CACHE

`sub`.`team\_no` AS `team\_no`,

`sub`.`id` AS `id`,

`cac`.`rownum`() AS `rowno`,

`sub`.`university\_no` AS `university\_no`,

`sub`.`team\_name` AS `team\_name`,

`sub`.`teacher` AS `teacher`,

`sub`.`teacher\_phone` AS `teacher\_phone`,

`sub`.`works\_name` AS `works\_name`,

`sub`.`types` AS `types`,

`sub`.`description` AS `description`

FROM

`view1` `sub`

GROUP BY

`sub`.`team\_no` , `sub`.`id`

/\*==============================================================\*/

/\* View: View1 \*/

/\*==============================================================\*/

VIEW `view1` AS

SELECT

`a`.`id` AS `id`,

`a`.`team\_no` AS `team\_no`,

`a`.`university\_no` AS `university\_no`,

`t`.`team\_name` AS `team\_name`,

`t`.`teacher` AS `teacher`,

`t`.`teacher\_phone` AS `teacher\_phone`,

`w`.`works\_name` AS `works\_name`,

`w`.`types` AS `types`,

`w`.`description` AS `description`

FROM

((`normaluser` `a`

JOIN `team` `t`)

JOIN `works` `w`)

WHERE

((`a`.`team\_no` = `t`.`team\_no`)

AND (`t`.`works\_id` = `w`.`works\_id`))

ORDER BY `a`.`team\_no` DESC

alter table News add constraint FK\_Reference\_5 foreign key (admin\_id)

references Admin\_info (admin\_id) on delete restrict on update restrict;

alter table NormalUser add constraint FK\_Reference\_1 foreign key (team\_no)

references Team (team\_no) on delete restrict on update restrict;

alter table NormalUser add constraint FK\_Reference\_6 foreign key (university\_no)

references School (university\_no) on delete restrict on update restrict;

alter table Team add constraint FK\_Reference\_7 foreign key (works\_id)

references Works\_info (works\_id) on delete restrict on update restrict;

alter table Works\_info add constraint FK\_Reference\_3 foreign key (schooluser\_id)

references SchoolUser (schooluser\_id) on delete restrict on update restrict;

alter table Works\_info add constraint FK\_Reference\_4 foreign key (admin\_id)

references Admin\_info (admin\_id) on delete restrict on update restrict;

# 4、详细设计

## 4.1MVC模式实现

Model层：Hibernate

Controller层：struts2+spring

View层：jsp

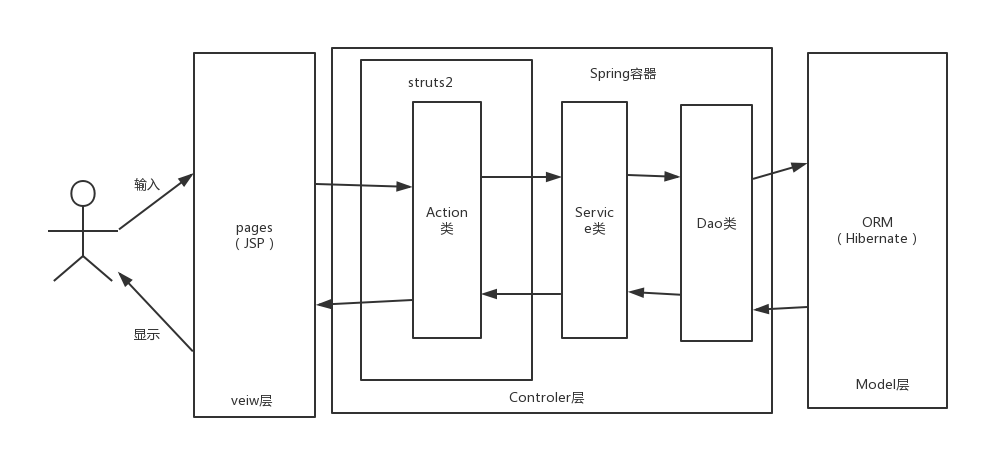


图4-1系统MVC架构图

说明：

1. view层jsp接收用户输入，通过action方式执行action类中的方法
2. action类的声明、跳转、调用方法由struts来管理
3. service类封装了需要实现的功能的操作，减少了action类方法的代码量，操作类dao作为service类的成员变量被调用
4. dao类主要负责数据库的具体操作方法
5. action，service，dao的具体实例化由spring容器管理

## 4.2模块设计（以登录为例）

1.模块名称：登录

2.功能：实现普通用户登录

3.模块时序图：

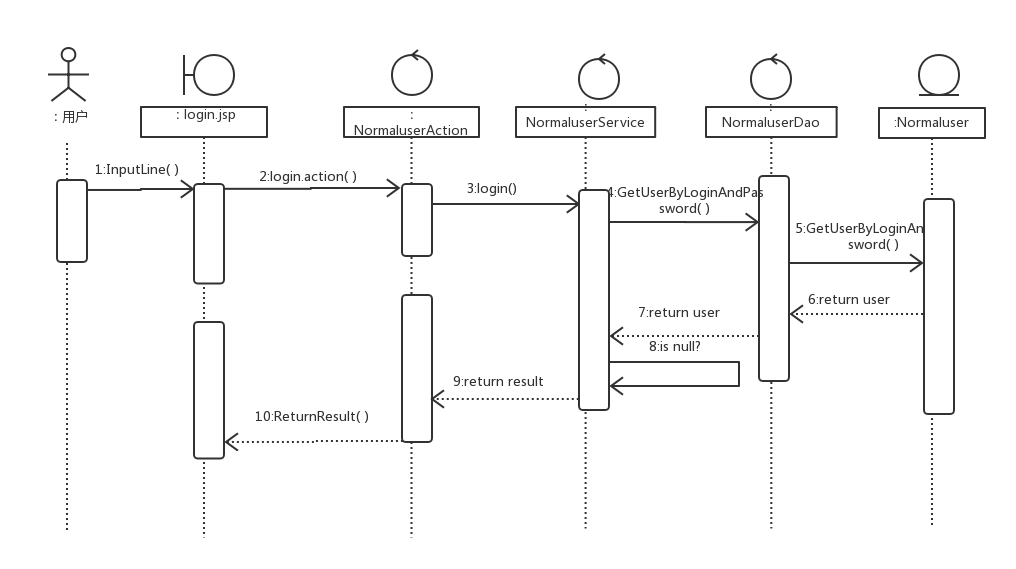


图4-2 登录模块时序图

4.相关方法说明：

1. login():登录方法，根据登录是否成功返回SUCCESS和INPUT
2. getUserByLoginAndPassword()：根据登录名和密码在Hibernate中获取Normaluser实体类对象，若获取对象为空则返回登录界面，提示

5.模块界面



图4-3登录模块界面

6.模块操作

1. 输入用户名和密码
2. 选择用户类别
3. 点击登录按钮

7.相关错误及提示

1. 用户名或密码错误



图4-4错误提示

# 5、系统创新

## 5.1后端部分创新

**1.运用了Struts2+Hibernate3.2+Spring4.0的ssh框架完成该网页**

**SSH框架**作为当今Java最流行的开源框架，作为一个学生，学习并掌握**SSH框架**是很有必要的，**SSH框架整合起来十分复杂和繁琐**，导入的数据包很多时候有可能会用冲突，要自己去排查，浪费了我很多时间

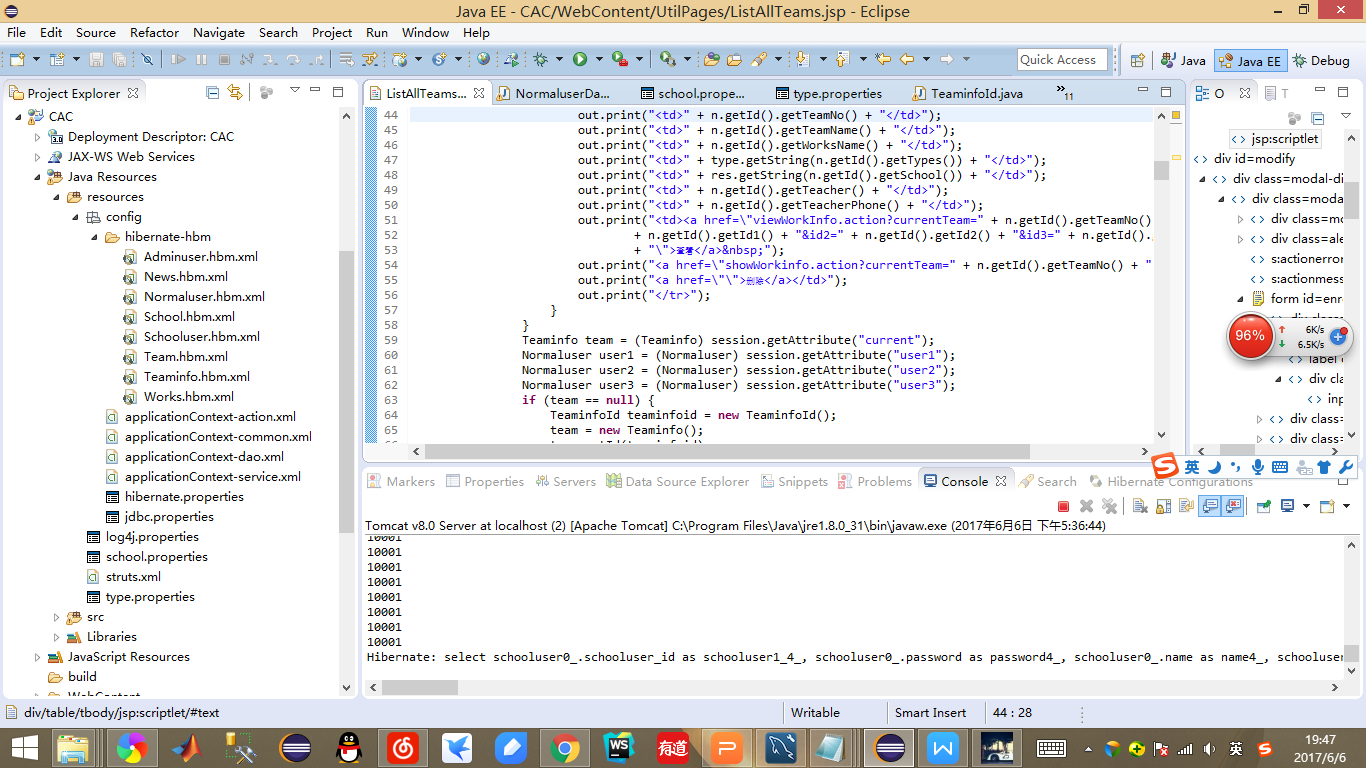


图5-1SSH框架结构图

**在本项目中，ORM模型用Hibernate来实现，所有的action，dao，service对象由spring负责去生成，struts则负责所有的action的管理和跳转，比较规范地实现了SSH框架的代码结构。**

1. **Spring实现对象的反向注入的一些小技巧**

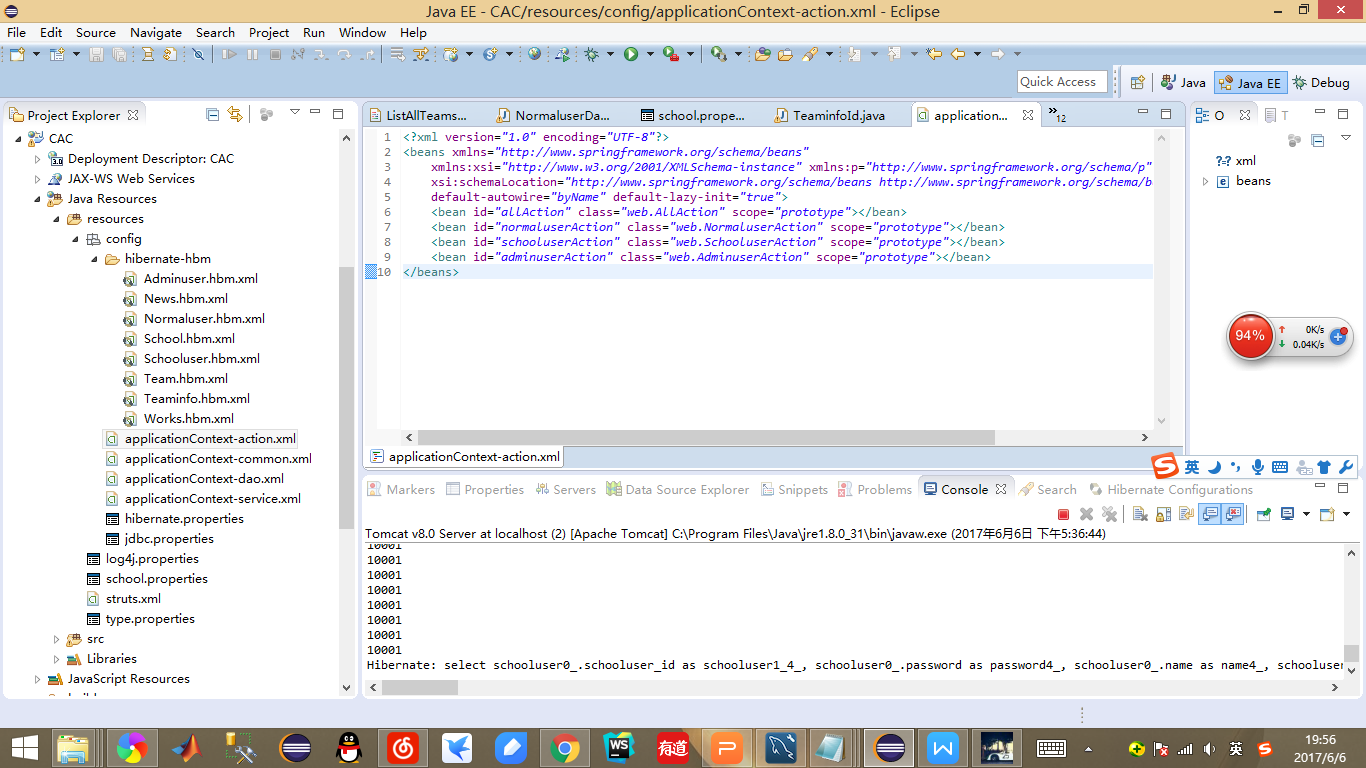


图5-2spring配置文件截图（其一）

在beans标签配置中，设置default-autowire=”byName”，default-lazy-init=”true”设置这两个参数的作用是

**default-autowire=”byName”**：在对象注入时候，根据生成对象的名字来注入对象，便于对象的管理，使得同名的对象只用声明一次就好，减少了代码量

**default-lazy-init=”true”**：延迟生成对象，简单点来说就是只要在用到的时候再生成，不用到就不生成该对象，这样做有利于提高容器的效率

1. **struts2**

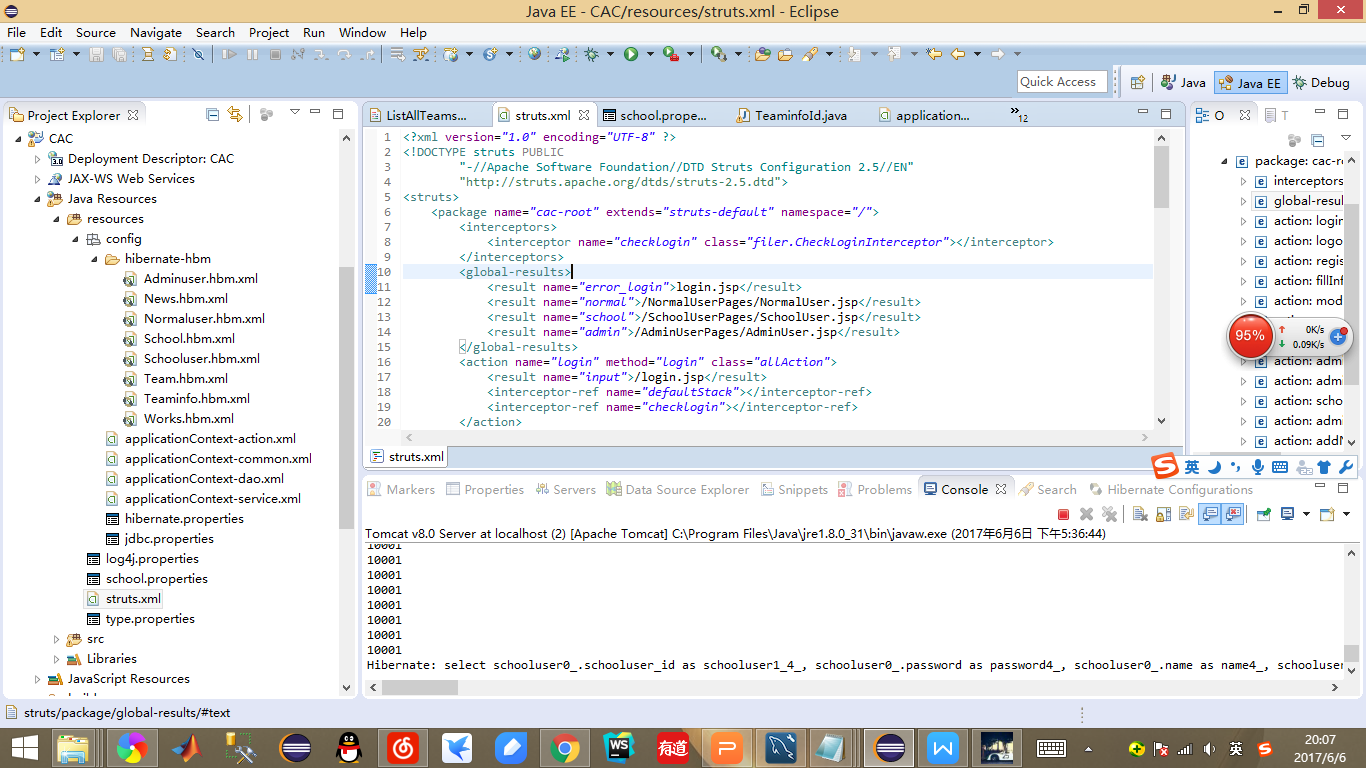


图5-3 struts配置文件截图

所有的struts2用到的action全部由spring统一生成和管理，例如上图中的allaction，执行action后跳转到相应的页面，并配置了自定义的拦截器checklogin

1. **面向接口编程的构想与实现**

**面向接口编程**是面向对象编程的一种更加规范的实现，使用面向接口编程的好处是，**实现了定义与实现的分离**，增强了程序的可扩展性，再次修改代码时候只需要修改接口的实现类而不用改接口定义。对于面向接口，我是这样实现的：

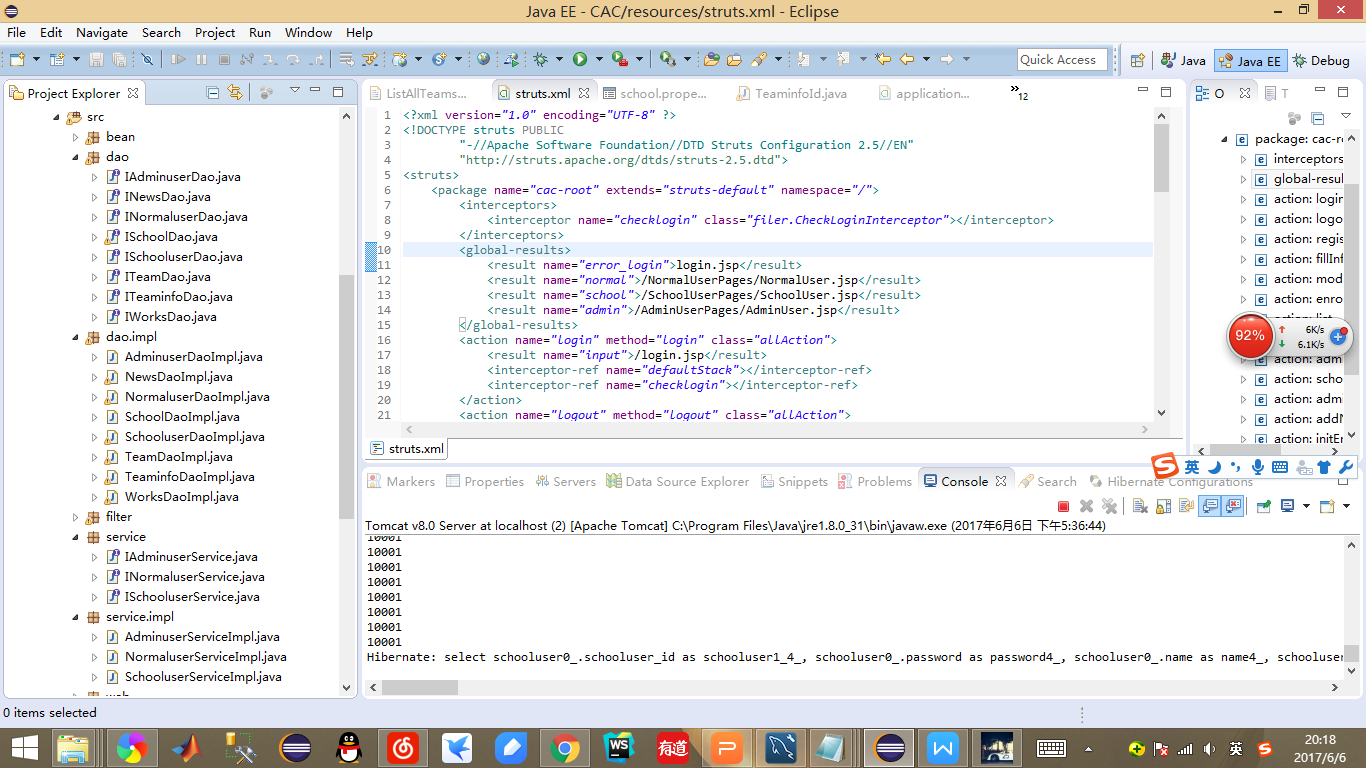


图5-4工程结构目录图

如图，dao包和service包里面放的是定义好的接口，命名为\*Dao和\*Service，然后再dao.impl和service.impl包里面放的是接口的实现类，命名为\*DaoImpl和\*ServiceImpl接口的定义和实现类相互分离，整个程序项目结构十分清晰

在ServiceImpl类中，**用接口作为成员变量，该接口的实例化由Spirng负责**

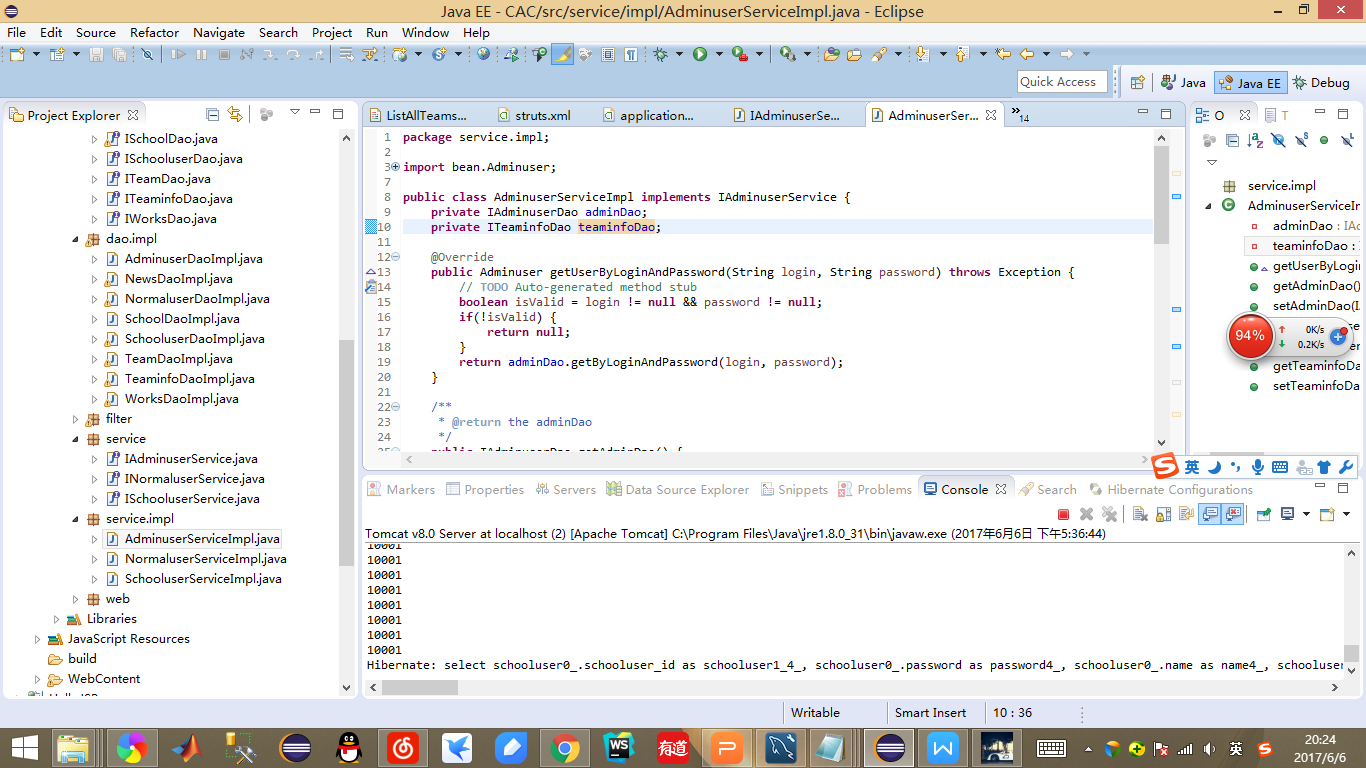


图5-5接口作为类属性

1. **数据库视图的实现**

为了使得数据库满足BC范式要求，消除插入和删除的异常，数据库的表要分开很多个小的表，在获取完整报名信息时候，就需要同时去访问用户表，队伍表和作品表，而且**由于获取报名信息这个操作很频繁，并且实现起来的sql语句比较复杂，用mysql的视图机制是最方便的选择，查询视图避免了每次都需要执行大量复杂繁琐的sql语句，减轻了服务器压力**。

在创建我需要的视图的过程中，主要遇到2个问题：

1. mysql的视图系统不支持子查询，只能从现有的表中获取列数据
2. 在我设计的视图中，需要用到Oracle数据库的ROW\_NUMBER() OVER ( PARTITION BY teamno ORDER BY id )函数，但mysql数据库没有这个函数。

**针对问题1，我将子查询的结果建立为新的视图（见上方视图view1和view2），并利用3层嵌套sql语句的方式实现了这个mysql没有的函数，虽然这个sql语句十分冗长且复杂（详细代码见3.7.5小节）**，但视图teaminfo的成功实现为我后面的编程带来了极大的便利，可以说是整个数据库设计的画龙点睛之笔。

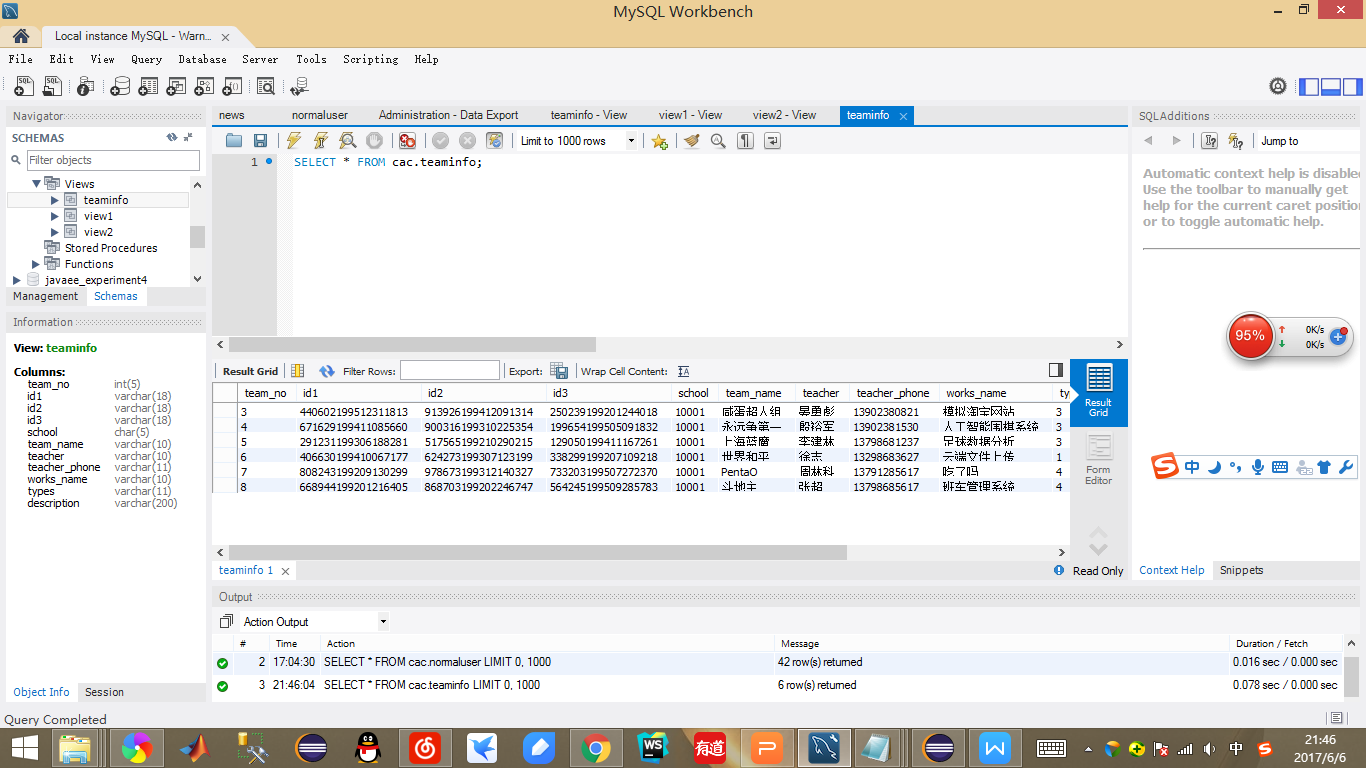


图5-6视图teaminfo的效果

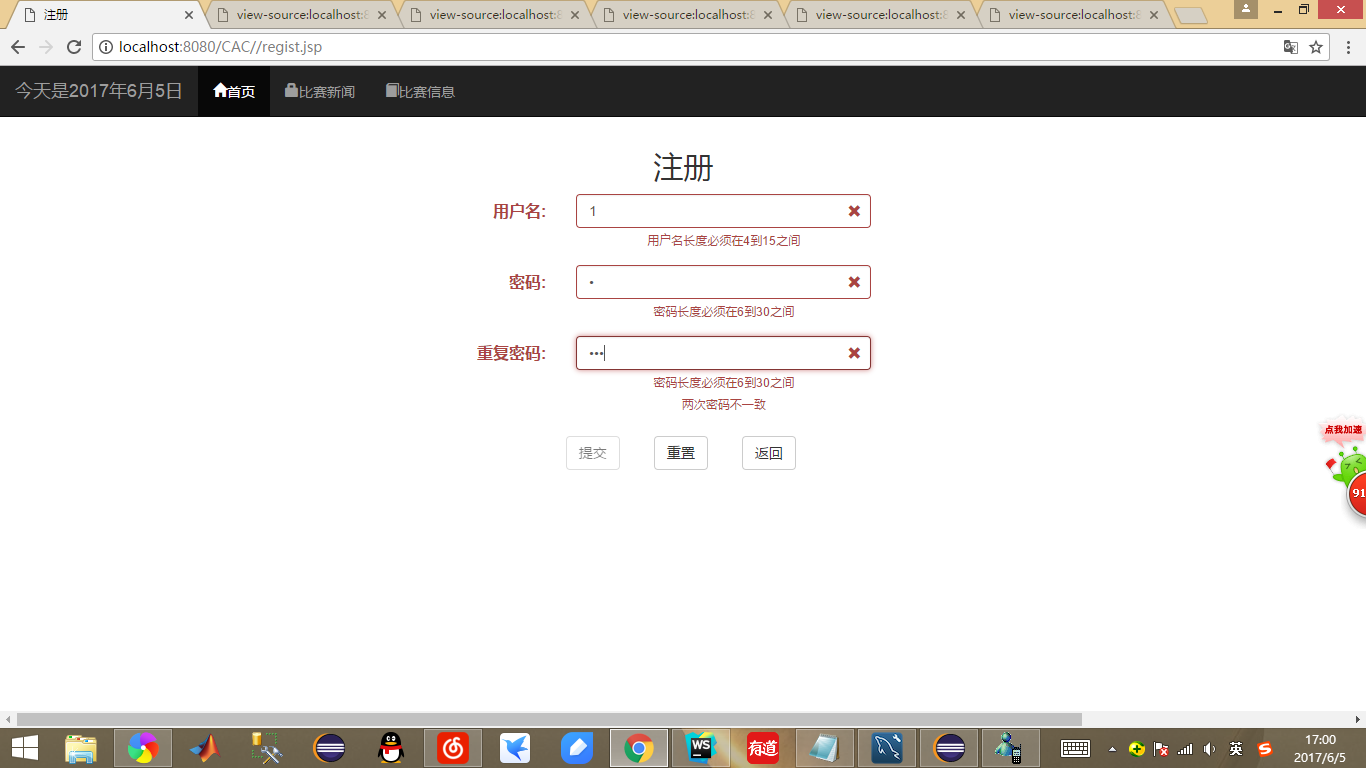
## 5.2前端的创新

1. **使用bootstrap框架完成前端设计**

bootstrap是当前最火的前端框架之一，只有当真正去学习和使用过bootstrap框架之后才会理解bootstrap的便利性，**bootstrap实现了很多页面组件的封装，使得前端设计者不用花费大量的时间去写底层的javascript和css代码**，都能实现出很好的页面效果，省去了很多的麻烦。

1. **使用bootstrap validator完成客户端校验**

传统的客户端校验，一般需要手写js代码实现。在struts2框架中，提供validator校验方式，一定程度上减轻了代码量，但struts2的校验实际上也是由struts2框架自动生成相应的js代码到页面编译，而且校验的css效果不能由前端设计者来实现。在本项目中，我使用的是bootstrap validator完成客户端输入校验



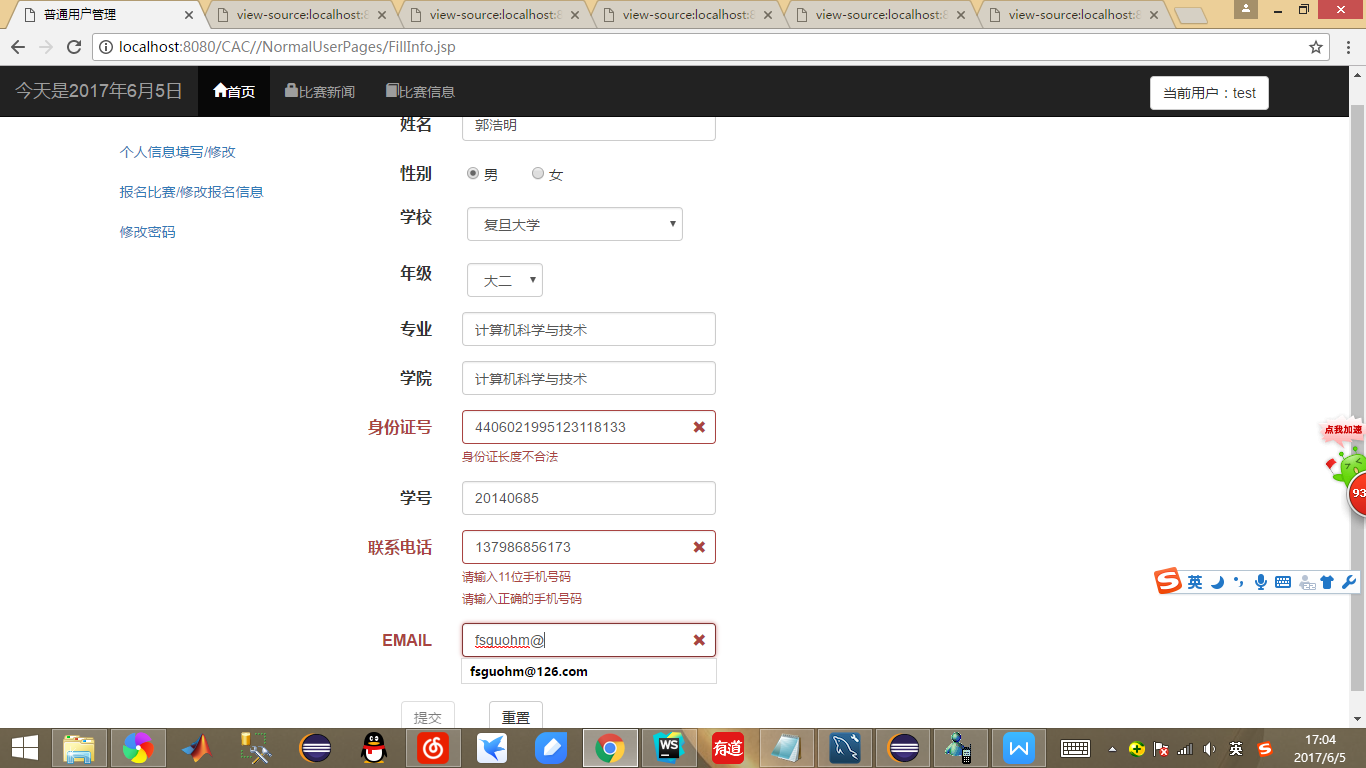
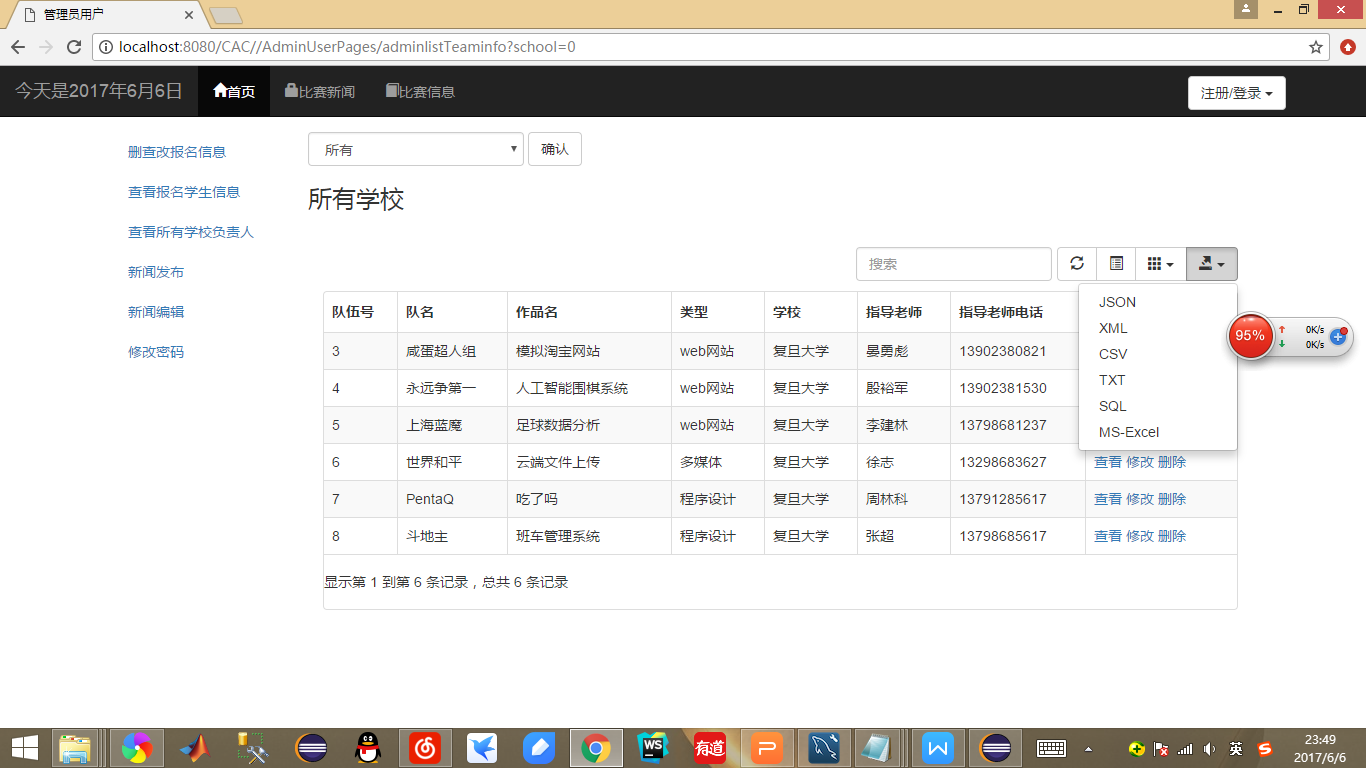


图5-6输入校验

1. 使用bootstrap table代替传统表格

bootstrap table作为一个开源的表格插件，能够很方便地实现排序，分页，导出等功能，只要写在bootstrap table配置的初始化js中就可以方便地实现功能。



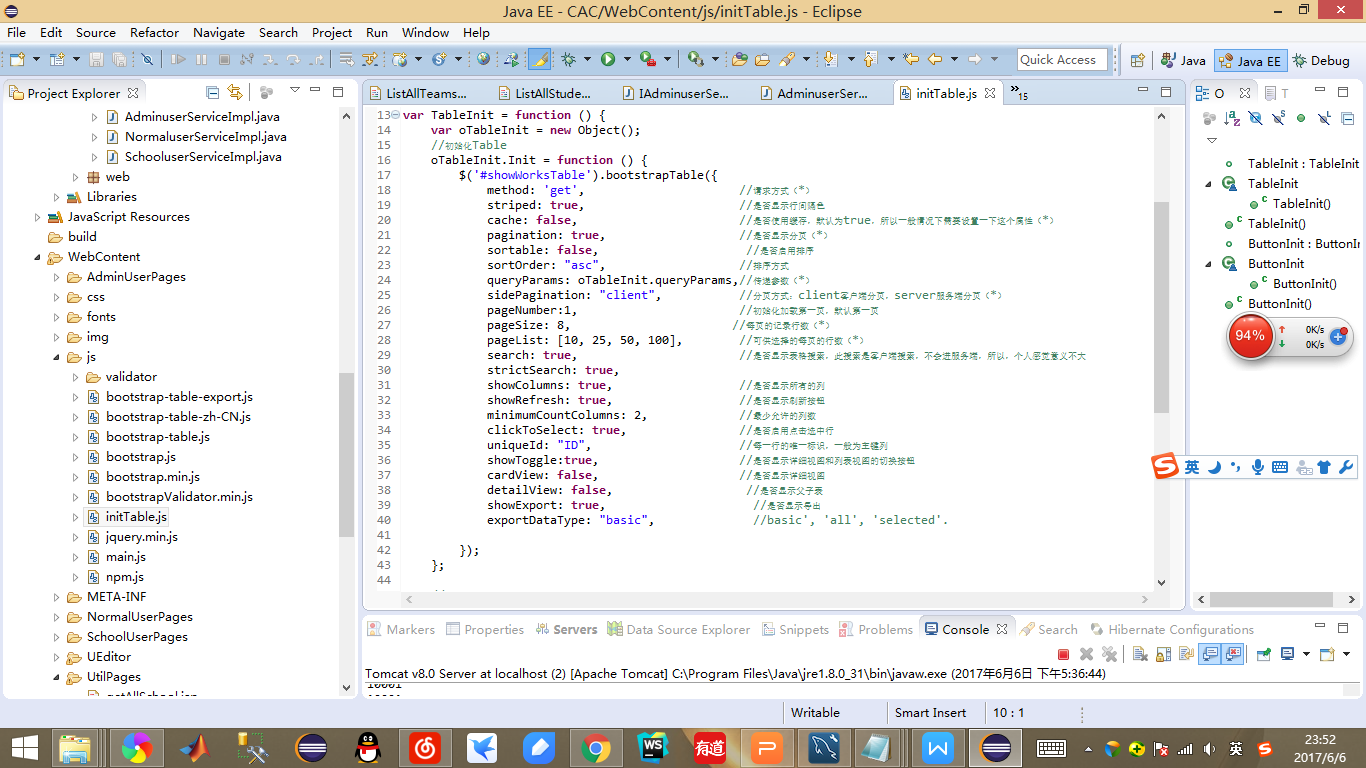


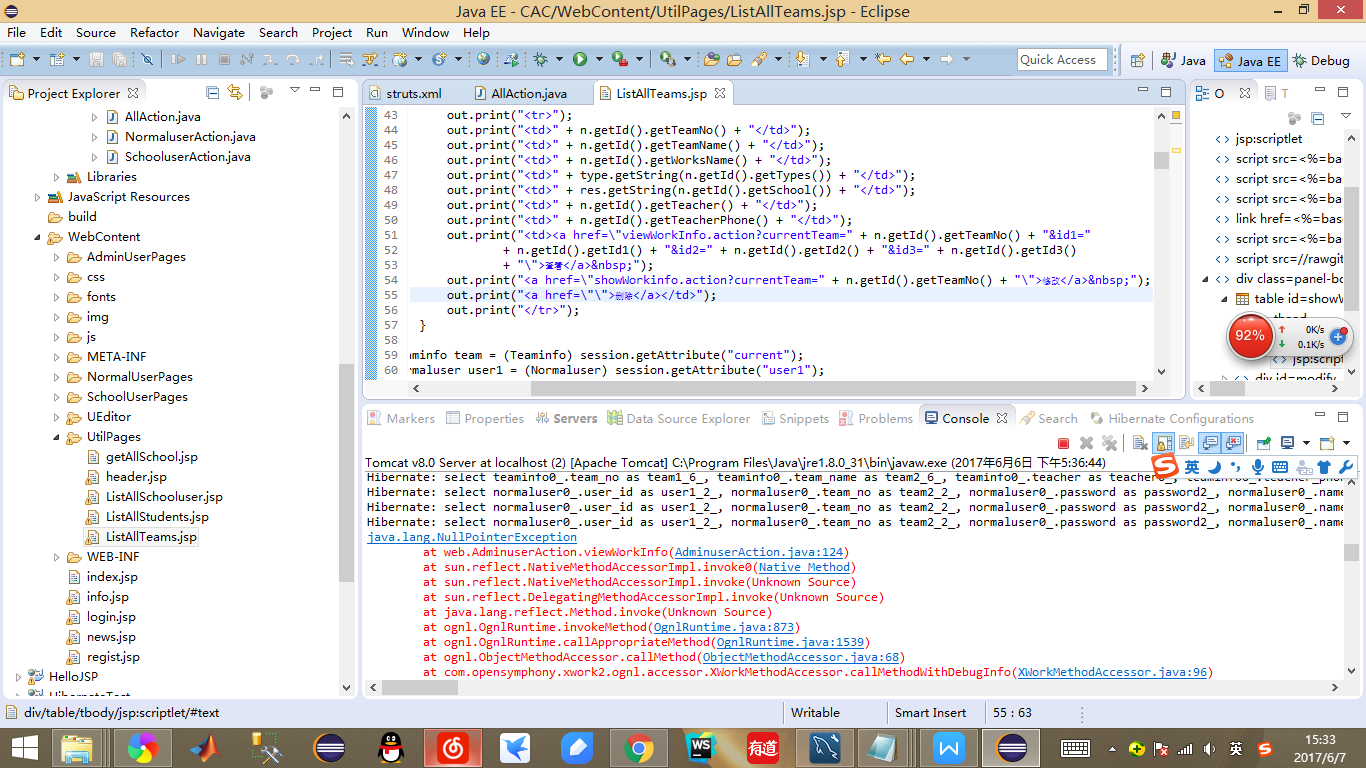
图5-7bootstrap table实现及其配置文件

1. 使用modal弹出框显示信息和修改信息

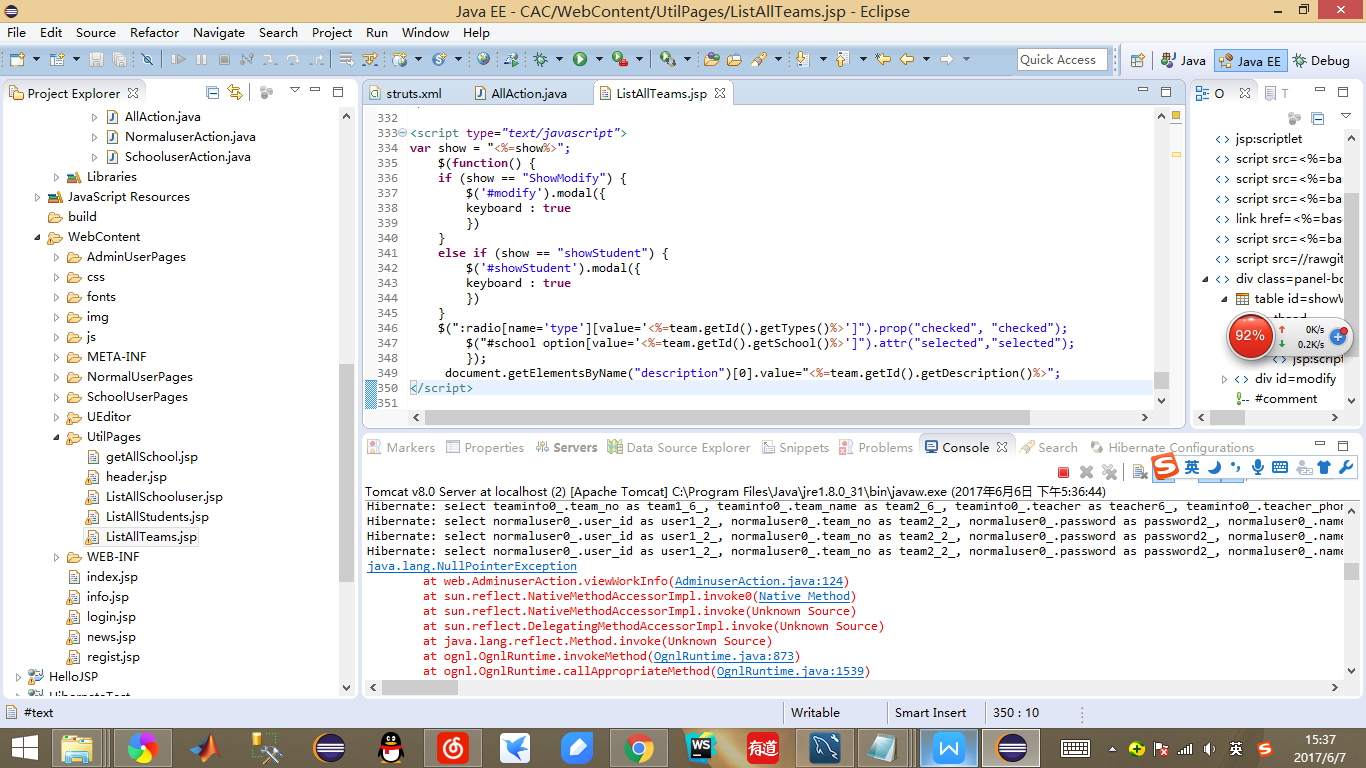




使用modal弹出框显示信息，避免了页面的频繁跳转，提供了更良好的用户体验。实现方法是在a标签用url调用action的同时传递参数



在该action中调用方法获取相关信息并存入session，同时增加一个session参数show，该参数用来判断是否用js显示modal，action执行完毕后跳转本页面，在页面初始化过程中调用js函数显示modal，js函数如下

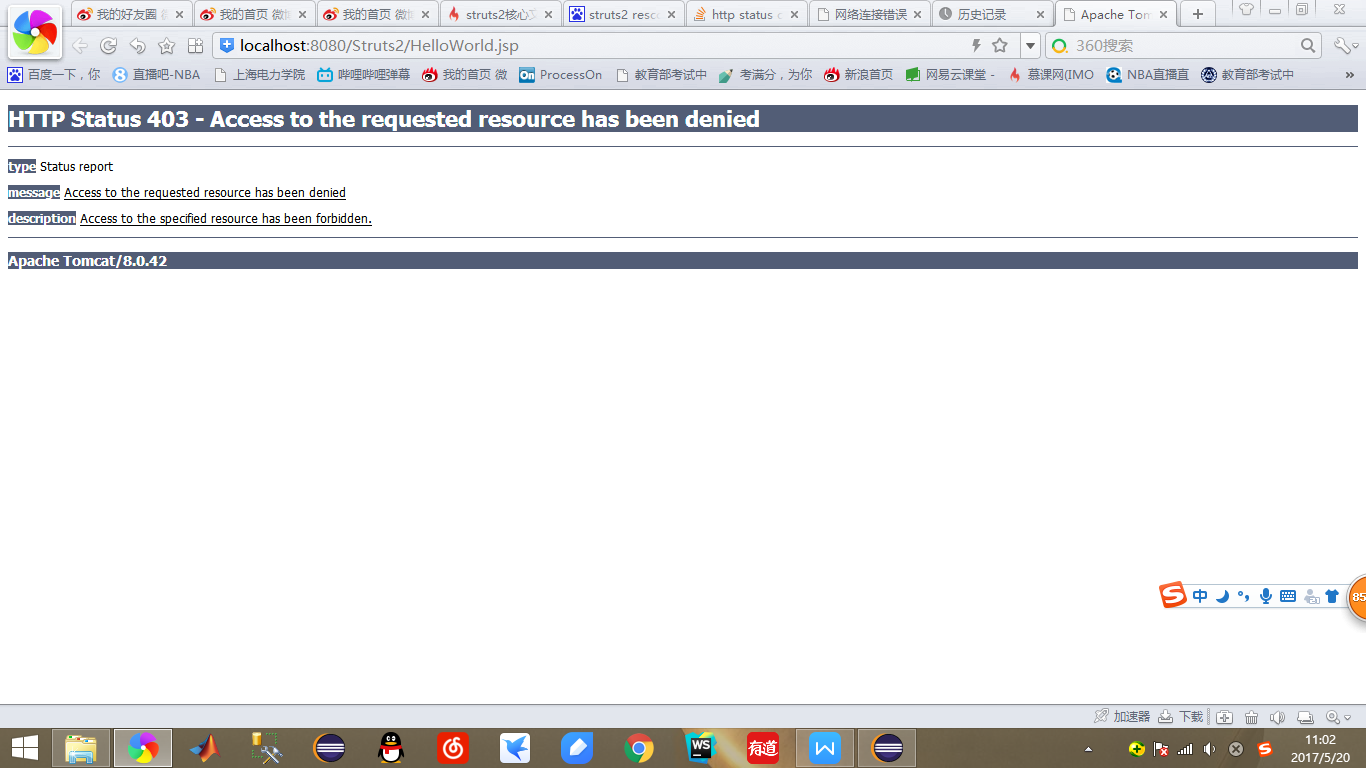


通过这种方法造成没有页面跳转的假象

# 6、课程总结

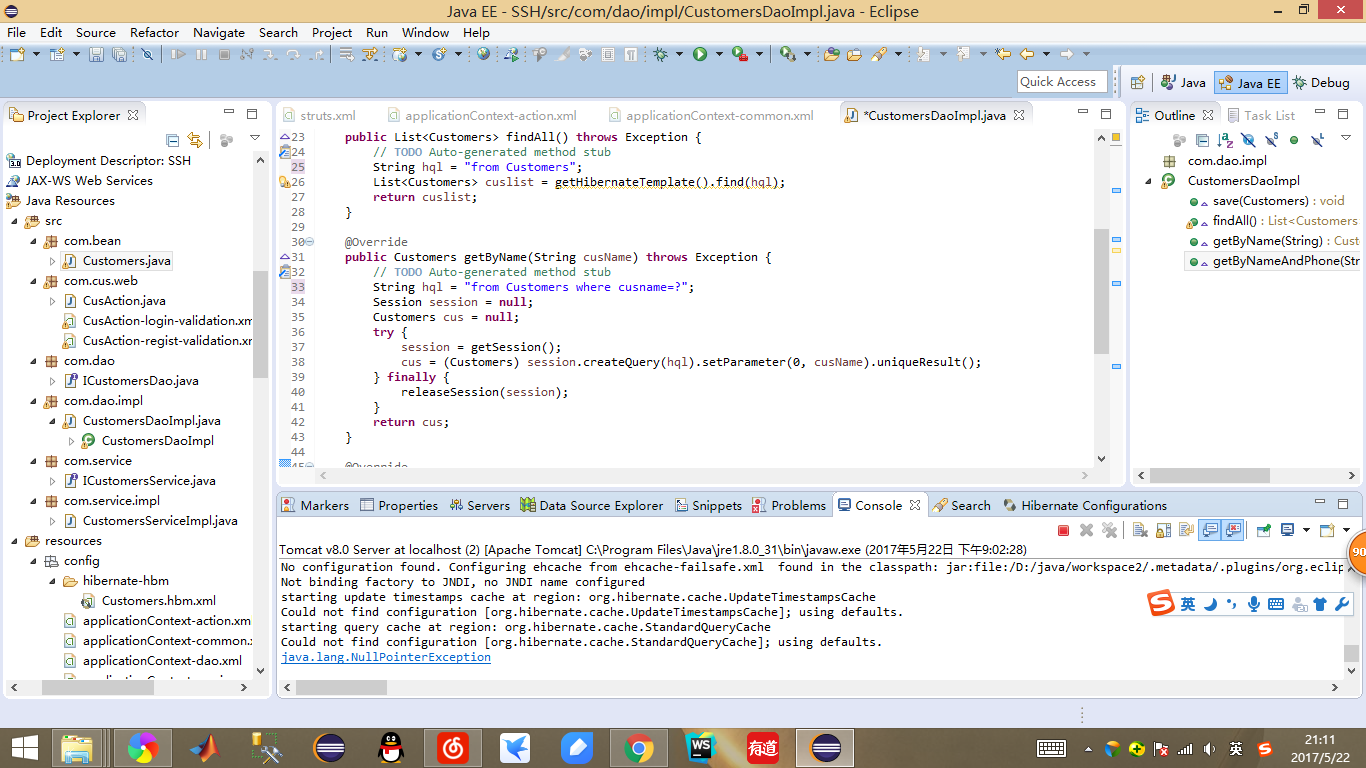
## 6.1遇到的问题

1. 在Struts2使用过程中，测试实例不能显示403错误



解决办法：访问的URL写错了，应该是写struts.xml文件中action标签的名称，而不是写java action类的类名

1. 运用Hibernate中出现了nullpointerexception



解决办法：hql语句中写的是生成实体类名，而不是数据库表名

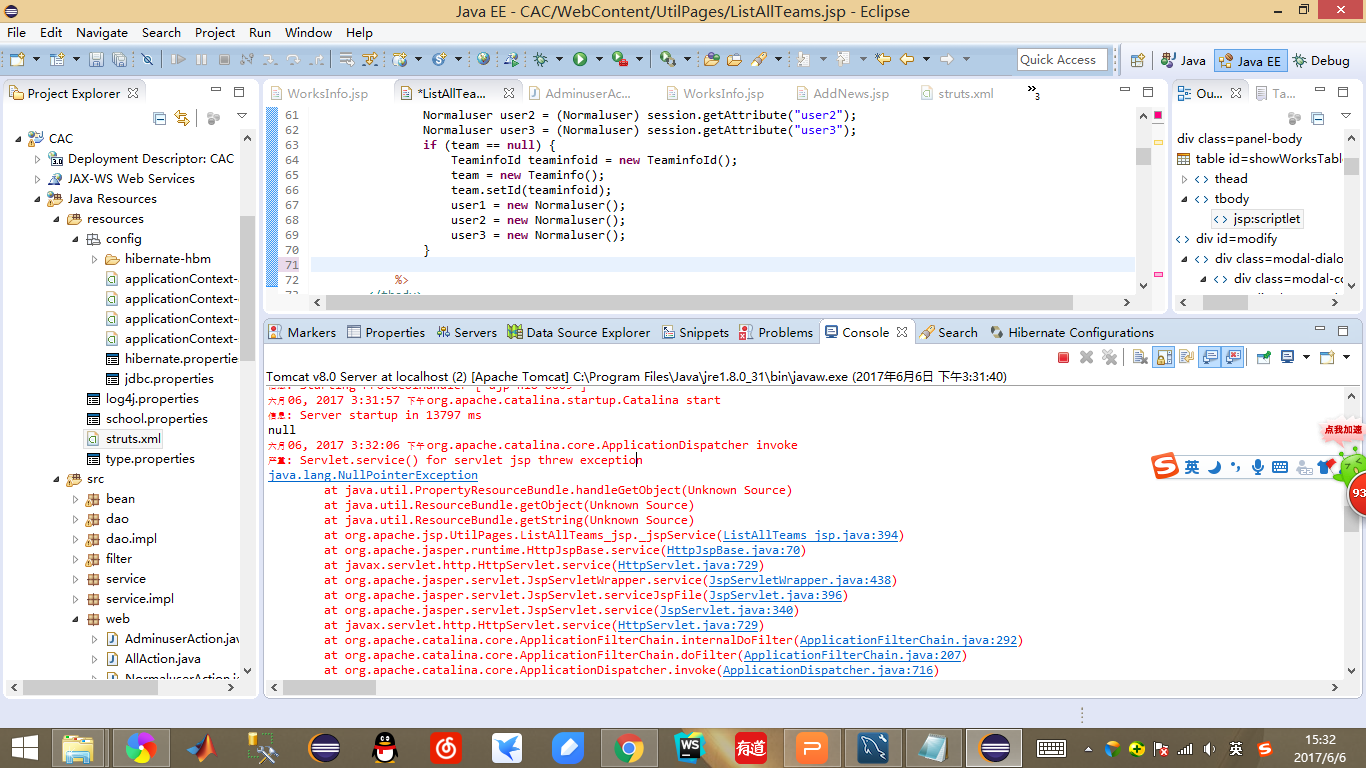
1. Hibernate与Spring整合后Spring注入对象值为null

解决办法：当hbm文件与实体类不在同一个包时，写类要写上包名，否则Spring会找不到这个类报错

1. 引用外部js文件，但引用还是原来未修改的文件

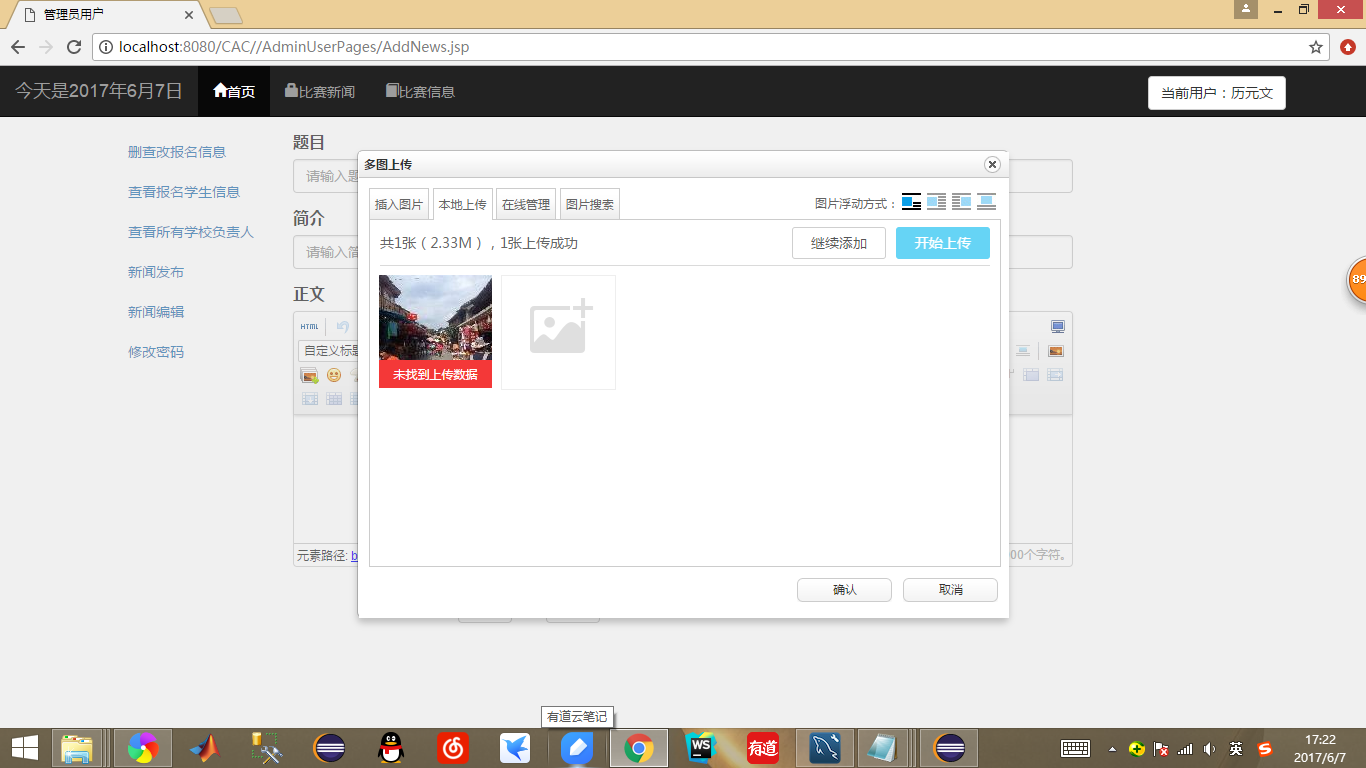
解决办法：浏览器的缓存，清除浏览器缓存

1. 页面加载时候报nullpointer错误



解决办法：错误原因是因为网页在加载过程中需要读到我自定义的资源文件school.properties,在初始化时候传入的参数为null，该资源文件中没有参数为null对应的字段，所以会报错。在该资源文件中加入一行null=””，在取值时改成res.getString("" + user2.getUniversityNo())即可

1. UEditor上传图片未找到数据



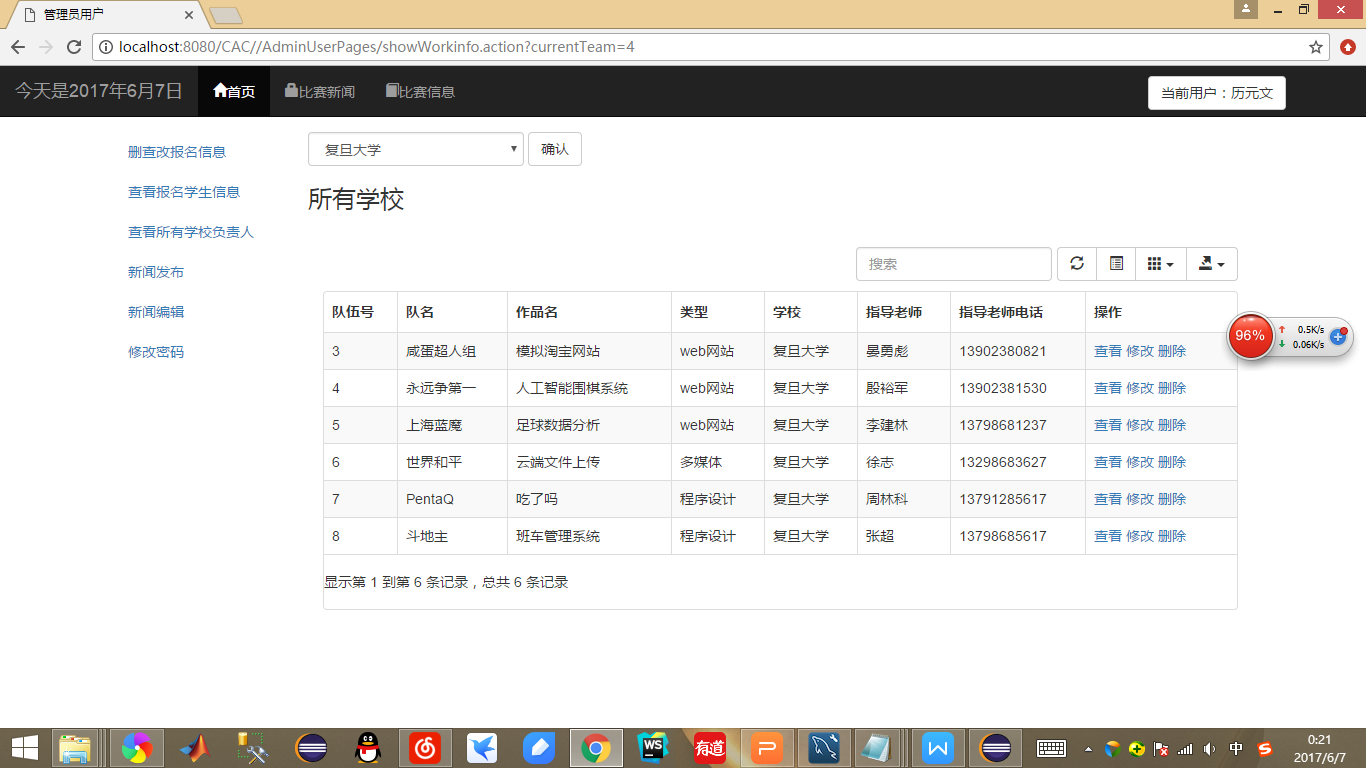
出先该问题的原因是：在配置struts过滤器，过滤路径设置/\*方式时，由于struts2框架默认使用apache的Commons FileUpload组件和内建的FileUploadInterceptor拦截器实现上传，会将request文件域封装到action中一个File类型的属性中，并删除request中的文件域，因此会上传文件失败。

以上只是节选了部分错误，实际上在SSH框架整合过程中出现了更多的错误，由于篇幅原因没有全部列出

## 6.3个人总结

对于这次课程设计，感觉对自己的能力有一个比较大的挑战，本来老师的要求只是用servelet实现MVC架构，但因为我希望借这次课程设计更多地接触一下之前从来没有用过的框架，所有决定用SSH来实现MVC。坦白说作为J2EE的初学者，一上来就接触SSH这3大框架的整合，是很有难度的，也正是因为这个原因，在项目初期我的进度一直很慢，从来没用过框架，第一次用就是SSH3个一起的整合，对于新手来说会比较痛苦。

因为框架的知识在课堂上没怎么讲过，在项目中一度也觉得自己力不从心，一度有过想放弃框架转回去写servlet的想法，到后面自己查阅了大量资料，经过自己的摸索，对SSH的整合有了初步熟悉，也渐渐上手，后面项目进度加快了，但即使是这样，也只是基本实现了要求的功能，还有一些自己有兴趣想写的东西也没来的及写，例如下图中表格右上角的小功能，把按钮加上去了本来想着有时间去完善，后面发现时间来不及了，类似的还有很多。



坦白说这个网站做的并不完美，只是实现了基本功能，细节处有待完善，主要是因为我在SSH入门整合花了大量时间。但我觉得时间花在这上面是值得的，开源是程序设计未来的趋势，接触优秀的开源框架，这次接触框架的初体验，不是说仅仅为了完成这次作业，对自己未来工作都打下了基础，有了这次经验，感觉以后再SSH框架的处理上会显得更加得心应手点。

当然，毕竟是初次接触SSH对于这个框架自己还有理解不到位的地方，对于AOP的概念自己也不是很了解，有待于在日后的学习中继续完善。