

**课程设计报告**

**课程名称:** J2EE框架与项目开发

**设计题目：** 前后端安全的考试系统

**院 系：** 计算机科学与工程学院

**专 业：** 软件工程

**班　级：** 软件142

**学 号：** 073013108 09414233

**姓 名：** 黄有为 曹琳琳

**指导教师：** 沈 健

**设计地点：** 宿舍

**开课时间：** 2016 至 2017 学年第 2 学期

目录

1. **引言**

1.1题目描述

设计开发一网络考试系统，可以在线答题，在线评分，在考试途中一旦出现因网络原因和计算机故障而意外退出考试，再次进入时还是当前退出时的状态，可以继续考试。考试完成后在线提交，可当即知道自己的成绩。

1.2设计意义

大部分的考试系统难以做到前后端的安全，通过这次开发的考试系统在功能和UI上虽然做了精简，但是在安全控制这一块做了很多，保证了考试系统前后端的一致性，无论是计时还是计分都做了严格控制，对技术作弊做了有效屏蔽

**2. 开发环境**

2.1 硬件环境

|  |  |
| --- | --- |
| 设备 | 台式主机i7 7700k |
| 硬盘 | 256ssd |
| 内存 | 8G |

2.2 软件环境

|  |  |
| --- | --- |
| 操作系统 | Win10 |
| IDE | eclipse for j2ee |

**3. 相关技术及知识点**

3.1相关技术

java web开发，企业级java开发，mysql数据库，多线程，session安全控制，加密，MVC模式，多层后端业务分离，数据模型自动化生成

3.2所运用的知识点

AmazeUI，国产的UI框架，非常简单，响应式布局

Spring框架：Spring使用的是基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅仅限于服务器端的开发。从简单性、可测试性和松耦合性角度而言，绝大部分Java应用都可以从Spring中受益。Spring的一个最大的目的就是使J2EE开发更加容易。同时，Spring之所以与Struts、Hibernate等单层框架不同，是因为Spring致力于提供一个以统一的、高效的方式构造整个应用，并且可以将单层框架以最佳的组合揉和在一起建立一个连贯的体系。可以说Spring是一个提供了更完善开发环境的一个框架。

hibernate ORM：数据模型映射，由于java是静态语言，对数据类型有强要求，故而使用hibernate将数据映射为模型对象，实现数据库增删改查

多线程：考试期间，对用户的考试对象开启多线程计时

MD5加密算法：加密用户密码

**4. 需求分析**

4.1 登录需求分析

可以根据登录信息来识别是否是管理员还是普通考生。新考生需要注册自己的个人信息

4.2 系统性能需求分析

考试期间客户端中断，随机生成数，高并发查询，多线程稳定

考试系统在高并发的情况下，对系统性能要就较高，并非一般网站管理系统，需要支持较多线程同时运行，快速准时的切换，需要高并发的随机数生成和网页发送，软件上尽量使用linux系统，针对http服务器尽量使用可抗并发的反向代理服务器。

硬件上，尽量使用多台主机组合为基于反向代理服务器分布式http网络服务器

可以使用nginx作为反向代理，tomcat作为上游服务器，实现考试开始和结束时简点的负载均衡，防止出现宕机线程中断等问题

每台主机瓶颈为IO与cpu，cpu应选择低频多线，带有超线程技术的E3，E5处理器，对每台上游的tomcat要求服务器cpu大于32线程，可组合双路64线程或128线程，IO存储使用ddr4和ssd，session存储于机械硬盘即可

如果是单台主机服务器，应该使用双路多路cpu，实现多线程的稳定

4.3系统功能需求分析

管理员：

1.删除用户

2.新建试卷

3.删除试卷

4.添加考生

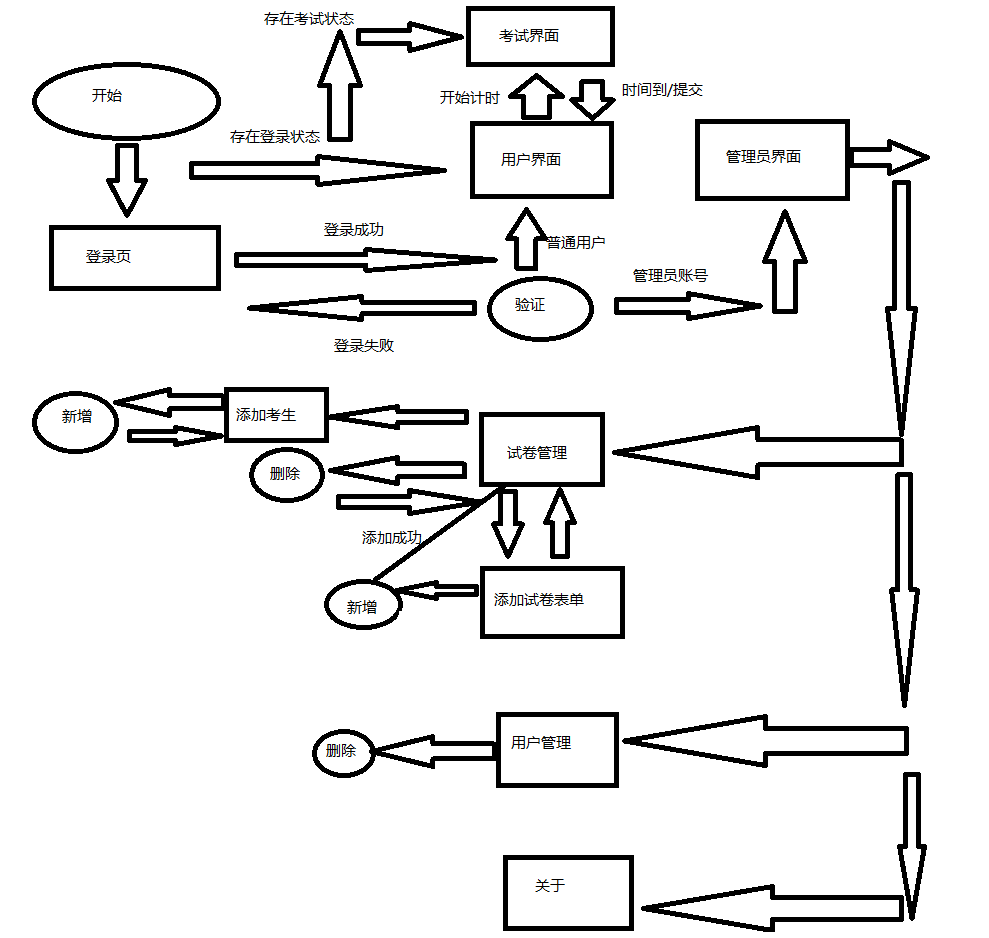
用户：

1.参与考试

2.考试提交计分

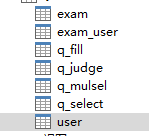
3.安全计时和反技术作弊

4.4 系统流程图

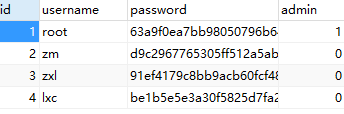


**5. 设计与实现**

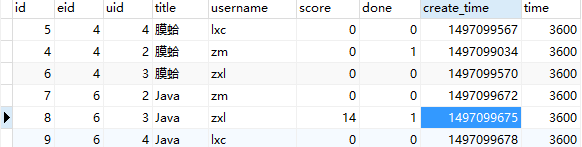
5.1数据库设计



User表：存放了普通用户与管理员的身份



Admin中是存放了辨别是否是管理员，1是管理员，而且密码是使用了MD5数字签名来保证密码的安全。

Exam\_user表：里面存放了相对应的试卷完成情况与成绩情况

5.2添加试卷

 可以以管理员身份来管理试卷

代码如下：

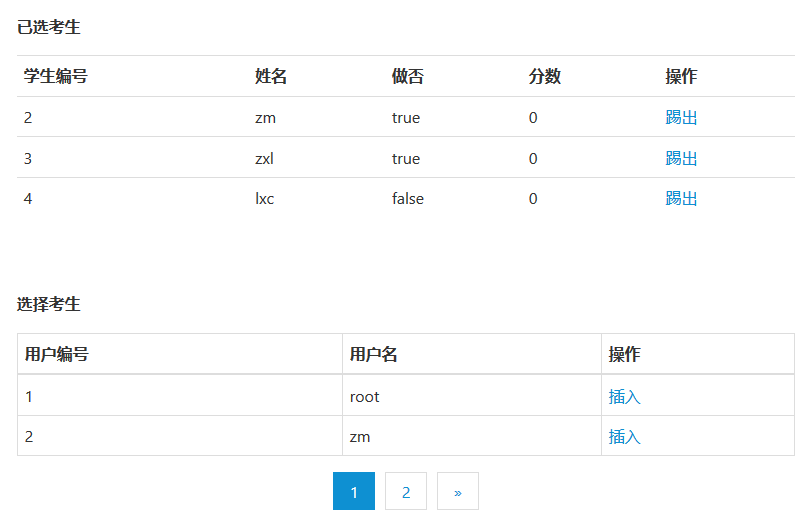
|  |
| --- |
| // 试卷管理  @RequestMapping("/")  public ModelAndView index(String cpage, HttpSession s) {  int rows = PAGE\_SIZE;  String cPage = cpage == null ? "0" : cpage;  ModelAndView mv;  if (s.getAttribute("admin") == null)  return new ModelAndView("redirect:/");  List<Exam> exams;  try {  exams = examService.findExamsByPage(Integer.parseUnsignedInt(cPage), rows);  } catch (NumberFormatException e) {  exams = examService.findExamsByPage(0, rows);  }  int pages = examService.getPages(rows);  mv = new ModelAndView("admin/index");  mv.addObject("EXAMS", exams); // 试卷总数  mv.addObject("PAGES", pages); // 总页数  mv.addObject("CPAGE", cPage); // 当前页  return mv;  }  // 删除试卷  @RequestMapping("/delExam")  public String delExam(String cpage, int id, HttpSession s) {  if (s.getAttribute("admin") == null)  return "redirect:/";  examService.DelExamById(id);  return "redirect:/Admin/?cpage=" + cpage;  }  @RequestMapping("/addExamUser")  public ModelAndView addExamUser(String cpage, int eid, HttpSession s) {  int rows = PAGE\_SIZE;  String cPage = cpage == null ? "0" : cpage;  if (s.getAttribute("admin") == null)  return new ModelAndView("redirect:/");  List<ExamUser> eus = examUserService.findByEid(eid);  Exam exam = examService.findExamById(eid);  List<User> us = userService.findUsersByPage(Integer.valueOf(cPage), rows);  int pages = userService.getPages(rows);  ModelAndView mv = new ModelAndView("admin/add\_exam\_user");  mv.addObject("EU", eus);  mv.addObject("USERS", us);  mv.addObject("EXAM", exam);  mv.addObject("PAGES", pages); // 总页数  mv.addObject("CPAGE", cPage); // 当前页  return mv;  } |

5.3 用户管理

管理员身份可以添加删除考生



也可以把已添加的考生踢除



代码如下：

|  |
| --- |
| // 向考试中添加一个学生  @RequestMapping("/addOneUser")  public String addOneUser(int uid, String cpage, int eid, HttpSession s) {  if (s.getAttribute("admin") == null)  return "redirect:/";  User user = userService.findUserById(uid);  Exam exam = examService.findExamById(eid);  if (user == null || exam == null)  return "redirect:/Admin/addExamUser?cpage=" + cpage + "&eid=" + eid;  ExamUser eu = new ExamUser();  eu.setEid(exam.getId());  eu.setUid(user.getId());  if (!examUserService.exists(eu)) {  eu.setCreateTime(getTimeStamp());  eu.setDone(false);  eu.setScore(0);  eu.setUsername(user.getUsername());  eu.setTitle(exam.getTitle());  eu.setTime(exam.getTime());  examUserService.save(eu);  }  return "redirect:/Admin/addExamUser?cpage=" + cpage + "&eid=" + eid;  }  @RequestMapping("/delExamUser")  public String delExamUser(int id, HttpSession s, String cpage, int eid) {  if (s.getAttribute("admin") == null)  return "redirect:/";  examUserService.delExamUserById(id);  return "redirect:/Admin/addExamUser?eid=" + eid + "&cpage=" + cpage;  } |

5.4试卷管理

考生可看到的已考试卷与未考试卷，不同考生看的也不同





关键代码如下：

|  |
| --- |
| // 检查有没有正在做的试卷，有就进去做！  if (s.getAttribute("randexam") != null)  return new ModelAndView("redirect:/User/exam");  User u = (User) s.getAttribute("user");  List<ExamUser> eus = examUserService.findByUid(u.getId());  ModelAndView mv = new ModelAndView("user/index");  mv.addObject("EUS", eus);  return mv;  @RequestMapping("/exam")  public ModelAndView exam(HttpSession s, Integer id) {  if (s.getAttribute("admin") != null)  return new ModelAndView("redirect:/Admin/");// 转发给管理员  if (s.getAttribute("user") == null)  return new ModelAndView("redirect:/");  // 当前存在未做完的试卷，有就先做当前试卷，并且忽略id请求  if (s.getAttribute("randexam") != null) {  RandExam r = (RandExam) s.getAttribute("randexam");  ModelAndView mv = new ModelAndView("user/exam");  mv.addObject("EXAM", r);  return mv;  }  // 不存在正在做的试卷的session，又没有传入试卷id，那么直接弹出去  if (id == null) {  return new ModelAndView("redirect:/User/");  }  // 初次进入本场考试，进行初始化，产生随机题目  ExamUser eu = examUserService.findById(id);  User u = (User) s.getAttribute("user");  // 防止做别人的试卷  if (eu.getUid() != u.getId()) {  return new ModelAndView("redirect:/User/");  }  // 做过的试卷不准做  if (eu.isDone()) {  return new ModelAndView("redirect:/User/");  } |

5.5意外保护

考生的考试途中如遇断电保护，再次登入可继续参加考试，时间从意外退出那一时间点继续计算，考试数据全数保存在session中，关键代码如下：

|  |
| --- |
| // 分别保存：考试基本信息，随机选择，随机段暄，随机填空。随机判断  RandExam re = new RandExam(eu, selects, mulsel, fill, judge, eu.getTime());  s.setAttribute("randexam", re); // session存储当前正在做的试卷  以上两行代码保存了以下的模型：  public class RandExam {  private ExamUser examUser; //考试基本信息  //考试产生的临时随机  private List<QSelect> qSelect;  private List<QMulsel> qMulsel;  private List<QFill> qFill;  private List<QJudge> qJudge;  private int time;//计时器 |

5.6考生与试卷

考生可在自己的用户界面看到已考的试卷与未考的试卷



关键代码如下：

|  |
| --- |
| ExamUser eu = examUserService.findById(id);  User u = (User) s.getAttribute("user");  // 防止做别人的试卷  if (eu.getUid() != u.getId()) {  return new ModelAndView("redirect:/User/");  }  // 做过的试卷不准做  if (eu.isDone()) {  return new ModelAndView("redirect:/User/");  }  // 取得试卷的模板  Exam exam = examService.findExamById(eu.getEid());  // 根据试卷模板生成随机题  List<QSelect> selects = qSelectService.findQSelectByRand(exam.getCSelect());  List<QMulsel> mulsel = qMulselService.findQMulselByLimitRand(exam.getCMultselect());  List<QFill> fill = qFillService.findQFillByRand(exam.getCFill());  List<QJudge> judge = qJudgeService.findQJudgeByRand(exam.getCTf());  // 分别保存：考试基本信息，随机选择，随机段暄，随机填空。随机判断  RandExam re = new RandExam(eu, selects, mulsel, fill, judge, eu.getTime());  s.setAttribute("randexam", re); // session存储当前正在做的试卷  // 开线程计时！后端安全  new Thread() {  @Override  public void run() {  // 根据考试时间设定线程沉睡时间  try {  while (re.getTime() > 0) {  // 用户提交  if (s.getAttribute("randexam") == null) {  break;  }  re.killTime();  s.setAttribute("randexam", re);  Thread.sleep(1000);  }  } catch (InterruptedException e) {  e.printStackTrace();  } finally {  s.setAttribute("randexam", null);  System.out.println("时间到，交卷！");  }  }  }.start();  eu.setDone(true); // 置为已做  examUserService.updateExamUser(eu); // 刷新数据  ModelAndView mv = new ModelAndView("user/exam");  mv.addObject("EXAM", re);  return mv;  }  @ResponseBody  @RequestMapping("/subExam")  // 用户唯一提交分数获得成绩的方法，其他情况全部0分处理  public String subForm(HttpSession s, @RequestParam(value = "select[]") String[] select,  @RequestParam(value = "fill[]") String[] fill, @RequestParam(value = "mulsel[]") String[] mulsel,  @RequestParam(value = "judge[]") String[] judge) {  if (s.getAttribute("user") == null)  return "nologin";  RandExam re = (RandExam) s.getAttribute("randexam");  if (re == null)  return "lost";  int score = checkAnswer((RandExam) s.getAttribute("randexam"), select, mulsel, fill, judge);  ExamUser eu = re.getExamUser();  eu.setScore(score); //更新成绩  examUserService.updateExamUser(eu);  s.setAttribute("randexam", null);  return "" + score;  } |

5.7处理试题

收到提交到的试卷后，与session randExam对象中的答案进行匹配，相同得分，不同不得分，最后返回一个总分，关键代码如下：

|  |
| --- |
| // 处理答案，计分器，返回分数  private int checkAnswer(RandExam r, String[] select, String[] mulsel, String[] fill, String[] judge) {  int c\_select = 0;  int c\_mulsel = 0;  int c\_fill = 0;  int c\_judge = 0;  List<QSelect> rselect = r.getqSelect();  List<QMulsel> rmulsel = r.getqMulsel();  List<QFill> rfill = r.getqFill();  List<QJudge> rjudge = r.getqJudge();  if (select.length > 0)  for (int i = 0; i < rselect.size(); i++) {  if (rselect.get(i).getAnswer().equals(select[i]))  c\_select++;  }  if (mulsel.length > 0)  for (int i = 0; i < rmulsel.size(); i++) {  if (rmulsel.get(i).getAnswer().equals(mulsel[i]))  c\_mulsel++;  }  if (fill.length > 0)  for (int i = 0; i < rfill.size(); i++) {  if (rfill.get(i).getAnswer().equals(fill[i]))  c\_fill++;  }  if (judge.length > 0)  for (int i = 0; i < rjudge.size(); i++) {  if (rjudge.get(i).getAnswer().equals(judge[i]))  c\_judge++;  }  return c\_select + c\_mulsel + c\_fill + c\_judge;  }  } |

5.8数据加密

我们对数据库的密码项采用了MD5加密，关键代码如下：

|  |
| --- |
| public class MD5 {  public static void main(String[] args) throws Exception {  String salt = "sgdjkajdhakld";  System.out.println(md5(""));  System.out.println(md5("a"));  System.out.println(md5("abc"));  System.out.println(md5("abcdefghijklmnopqrstuvwxyz"));  System.out.println(md5("abcdefghijklmnopqrstuvwxyy"));  System.out.println(md5("123456"));  System.out.println(md5(salt + md5("123456")));  }  public static String md5(byte[] input) throws NoSuchAlgorithmException {  MessageDigest md = MessageDigest.getInstance("MD5");  byte[] b = md.digest(input);  return ByteArrayUtil.ByteArray2HexString(b);  }  public static String md5(String s) throws UnsupportedEncodingException, NoSuchAlgorithmException {  return md5(s.getBytes("UTF8"));  }  public static String md5File(String filename) throws NoSuchAlgorithmException, FileNotFoundException, IOException {  MessageDigest md = MessageDigest.getInstance("MD5");  DigestInputStream in = new DigestInputStream(new FileInputStream(filename), md);  byte[] buffer = new byte[8192];  while (in.read(buffer) != -1) {  md.update(buffer);  }  byte[] raw = md.digest();  in.close();  return ByteArrayUtil.ByteArray2HexString(raw);  }  } |

5.9 MVC设计

有Controller,Model,View,Service和DAO五层，控制层有LoginController，UserController和AdminController。Model层，数据库有几张表就有几个类，多出来的RandExam放在了session中，存放的是当场考试所有的信息。View中放了两个目录，一个是admin目录，一个是user目录，管理员所有的请求都放在了admin目录下，用户都放在user用户下。Service是数据处理层，提供数据加工服务，数据库中每个表都有个service，主要用来加工和处理数据，是不与数据库直接交互的。对数据库直接的读取都是在DAO层里，所有类都继承于BaseDao这个父类，BaseDao用泛型传入一个模型作为<T>，由于模型和数据库中表一一对应，所以BaseDao的任意一个方法都可以对表进行不同操作。

**6.小结**

这是一次有挑战的大作业，端午节期间开发完成，大概3天左右的时间，由于之前并没有参与过j2ee项目和考试系统需求开发，所以从零开始，让我觉得有点吃力，很多需求上的设计需要反复斟酌，结合自身考试经验来设计，代码上使用java，包括java多线程，spring，hibernte技术事实上也是我第一次使用，事实上对于本次的大作业，并不需要orm这样的数据模型来驱动，这反而拉低了性能增加了复杂度，考试系统主要应该做到功能精简，后期维护少，稳定工作等特点。

**参考资料**

[1] 书籍无

[2] 博客 <http://www.cnblogs.com/fuck1/p/5373700.html> 多线程