# 5. 系统详细设计与实现

## 5.1 主要界面设计

在线考试系统中主要设计了登录界面、后台管理页面、学生考试页面。 其中，根据页面设计的具体要求，主要使用了java模板引擎freemaker 技术，并配合相应的代码，来实现 预期的功能。而且为了界面更美观，还特地用到了PS技术在背景和相关部位插入了图片。





图5-1 在线考试系统主界面图

|  |
| --- |
| <html>  <head>  <#include "forecomment/head.ftl">  </head>  <body>  <#include "navbar.ftl">  <div class="content">  <img src="${request.contextPath}/static/images/index02.png">  <img src="${request.contextPath}/static/images/index03.png">  <button id="" type="button" class="btn btn-success" style="width: 980px;margin: 65px auto 65px auto;display: block"  onclick="exam()">立即考试  </button>  </div>  <#include "./forecomment/footer.ftl"></body><script> function exam(){  location.href="${request.contextPath}/web/kskm"; }</script></html> |

## 5.2 登录页面的实现

在地址栏输入http://111.230.27.166:8080/onlineTest/，进入网站登入界面，输入用 户名与密码，并选择相应的身份进入网站。例如用户名：admin，密码：123456，系统就会进入正确的页面。简单验证从本质上说就是在服务端来验证客户端提交的 form 中的数据。这种验 证只是对 form 中的数据规则进行检查，如必须输入用户名，密码必须正确。在这个 验证过程中，并不需要访问数据库。因此，简单验证需要在用户提交 form 后，并且 在服务器处理 form 中的数据之前进行。在进行完简单验证后，如果 form 中的数据不 合法，程序就会 forward 到指定的页面（一般是包含 form 的页面），并显示相应的 错误信息。如果 form 中的数据完全正确，程序就会继续执行。在这里如果不成功， 就会跳到登录页面，重新登录。登录页面如图 4.1 所示。

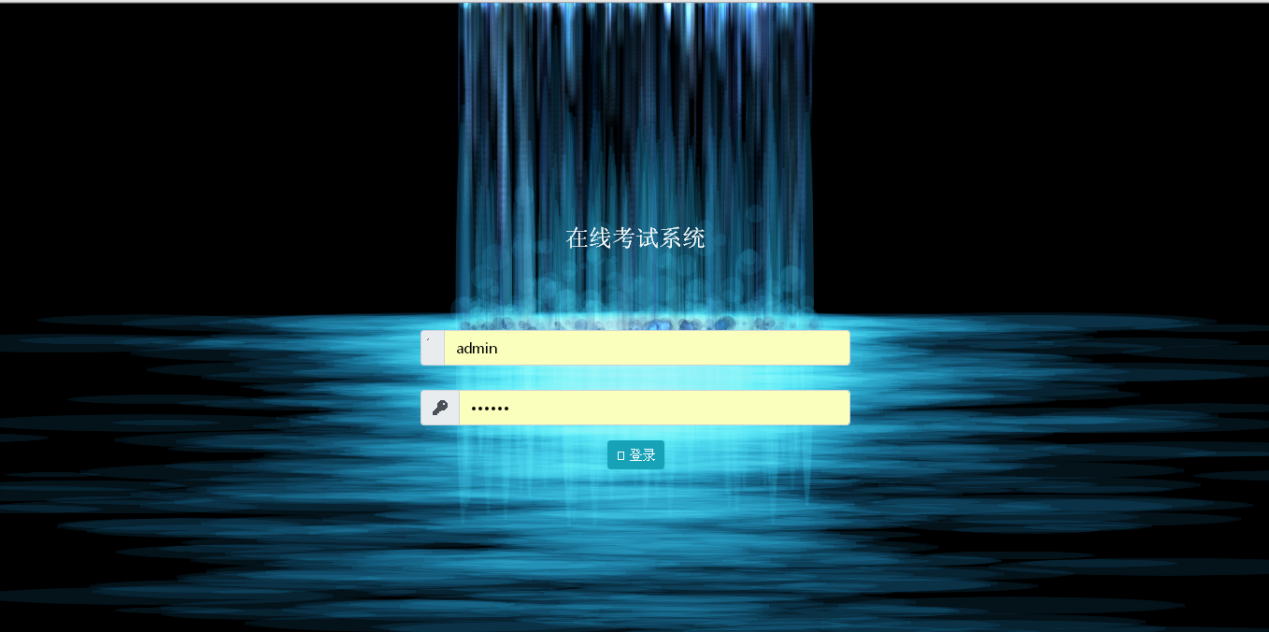


图5-2 登陆界面

|  |
| --- |
| <div id="container">  <canvas id="waterfall"></canvas> <div class="emma flex"> <div>  <div class="login-title">  在线考试系统  </div>  <div class="login-body">  <form method="post" id="login-form">  <div class="input-group mb-3" style="padding: 50px 20px 0 20px">  <div class="input-group-prepend">  <span class="input-group-text" id="basic-addon1"><i class="fas fa-user"></i></span>  </div>  <input type="text" class="form-control" placeholder="用户名" id="username">  </div>  <div class="input-group mb-3" style="padding: 10px 20px 0 20px">  <div class="input-group-prepend">  <span class="input-group-text" id="basic-addon2"><i class="fas fa-key"></i></span>  </div>  <input type="password" class="form-control" placeholder="密码" name="password" id="password">  </div>  <div class="input-group mb-3">  <button type="button" class="btn btn-sm btn-info loginBtn" id="loginBtn">  <i class="fab fa-angellist"></i>  登录  </button>  </div>  <div class="input-group mb-3" style="width: fit-content;margin: auto">  <#--<c:when test="${not empty errMsg}">-->  <#--<div style="color: red">登录失败：</div>-->  <#--</c:when>-->  </div>  </form>  </div>  </div></div>  </div> |
| @RequestMapping(value = "web/login/",method = RequestMethod.POST)  @ResponseBody  **public** MsgBean **userlogin**(@RequestBody User user, HttpSession session, Model model){  **return** **comLogin**(user,session,**true**);  } |
| **public** MsgBean **comLogin**(User user, HttpSession session,Boolean isweb){  **if**(user.getUname()==**null**){  **return** **new** **MsgBean**(**false**,"账号不为空 ",**false**);  }  Subject subject = SecurityUtils.getSubject();  User tempuser;  **try** {  UsernamePasswordToken token=**new** UsernamePasswordToken(user.getUname(),user.getPwd());  logger.info("========================================");  logger.info("1、进入认证方法");  subject.login(token);  tempuser = (User)subject.getPrincipal();  **if**(session.getAttribute("UserData")==**null**&&isweb){  session.setAttribute("UserData",subject);  }  logger.info("登录完成");  **return** **new** **MsgBean**(**true**,"登录成功",**true**);  } **catch** (AuthenticationException e) {  **return** **new** **MsgBean**(**false**,"账号密码不正确 ",**false**);  }  } |

## 5.3 管理员页面的实现

在地址栏输入http://111.230.27.166:8080/onlineTest/Admin/quit/登录页面后，可以进行考试管理，试卷管理，用户管理，试题管理，和退出。jsp 发出请求提交给 ActionServlet， ActionServlet 把 用 户 请 求 交 给 模 块 控 制 器 RequestProcessor ， 模 块 控 制 器 RequestProcessor 把 request 请求的数据封装成 ActionForm，调用对应的 Action，执行 它 的 execute 方 法 ， execute 方 法 调 用 业 务 逻 辑 进 行 处 理 ， 处 理 完 返 回 一 个 ActionForward 对象，最后把响应结果通过 jsp 展现给客户。jsp 页面中用户通过填写 数据库，再提交到不同 actionForm 中，actionForm 提交到总的控制器 ActionServlet， 然后再提交到不同的 Action，通过配置文件 struts-config.xml 以及 Hibernate 的配置文 件与数据库中的数据进行实时的更新。如下管理员页面如图 5-2

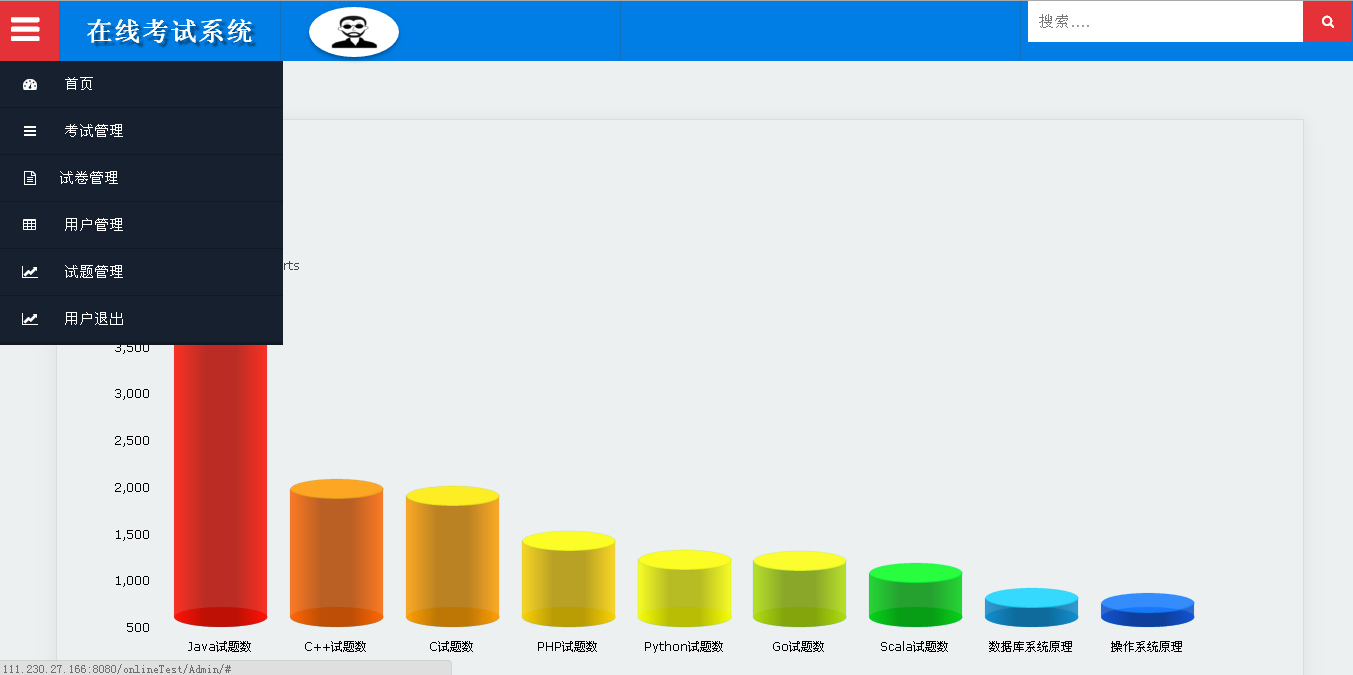


图 5-3 管理员页面

|  |
| --- |
| <div class="inner\_content" id="inner\_content">  <!-- /inner\_content\_w3\_agile\_info-->  <div class="inner\_content\_w3\_agile\_info">  <div class="w3ls\_agile\_cylinder chart agile\_info\_shadow">  <h3 class="w3\_inner\_tittle two">题库图表</h3>  <div id="chartdiv"></div>  </div>  <!-- /agile\_top\_w3\_grids-->  <div class="row">  <ul class="ca-menu">  <li>  <a href="#">  <i class="fa fa-building-o" aria-hidden="true"></i>  <div class="ca-content">  <h4 class="ca-main">16,808</h4>  <h3 class="ca-sub">用户数</h3>  </div> </a> </li><li>  <a href="#">  <i class="fa fa-user" aria-hidden="true"></i>  <div class="ca-content">  <h4 class="ca-main one">26,808</h4>  <h3 class="ca-sub one">题目数</h3>  </div> </a> </li>  <li>  <a href="#">  <i class="fa fa-database" aria-hidden="true"></i>  <div class="ca-content">  <h4 class="ca-main two">29,008</h4>  <h3 class="ca-sub two">访问量</h3>  </div>  </a>  </li>  <li>  <a href="#">  <i class="fa fa-tasks" aria-hidden="true"></i>  <div class="ca-content">  <h4 class="ca-main three">49,436</h4>  <h3 class="ca-sub three">注册用户数</h3>  </div>  </a>  </li>  <li>  <a href="#">  <i class="fa fa-clone" aria-hidden="true"></i>  <div class="ca-content">  <h4 class="ca-main four">30,808</h4>  <h3 class="ca-sub four">今日开考次数</h3>  </div>  </a>  </li>  <div class="clearfix"></div>  </ul>  </div> |

## 5.4 学生考试页面的实现

学生登录后，根据相应的角色登录进来以后，、修改资料、在线考试、也可以进行相应的成绩查询。最后可以进行注销。在这个 项目中，我们经常需要进行一些页面跳转，常用方法是<jsp:forward>跳转或者直接超链接，但是如果我们在 Struts 项目中使用这样的做法则在一定程度上非常笨重，我们采用的是新一点的技术springMVC+Freemaker模板技术，因为这种轻量级的框架使，提出了 ActionForward 类，客户端发送跳转请求通过控制器跳转。学生考试页面如图 5-4



图 5-4 学生考试页面图

|  |
| --- |
| @RequestMapping(value = "web/main",method = RequestMethod.GET)  **public** ModelAndView **main**(HttpSession session, Model model){  ModelAndView modelAndView=**new** ModelAndView();  modelAndView.setViewName("main");  **return** modelAndView;  } |
| <html>  <head>  <#include "forecomment/head.ftl">  </head>  <body>  <#include "navbar.ftl">  <img src="${request.contextPath}/static/images/fenlei01.png" style="display: block;width: fit-content;margin: 66px auto 0">  <div class="content-grid">  <img src="${request.contextPath}/static/images/fenlei02.png" style="margin-left: -29px">  <table class="content-table"><tr>  <#if testlist??>  <#list testlist as testItem> <td>  <a class="table-kskm" style="color: #a94442"  href="${request.contextPath}/web/paper/${testItem.id}">${testItem.testname}</a>  <div class="table-jiankao">考试方式：  <#switch testItem.type>  <#case 0>实时考试<#break>  <#case 1>模拟考试<#break>  <#default >类别错误  </#switch></div>  </td>  </#list>  </#if>  </tr>  </table>  </div>  <#include "./forecomment/footer.ftl">  </body>  </html> |

## 5.5成绩查询页面的实现

考生在考试结束后就可以随时查询刚才考试的基本成绩，如下图5-5 成绩查询图。



图5-5 成绩查询图

|  |
| --- |
| @RequestMapping(value = "web/grade",method = RequestMethod.GET)  **public** String **grade**(HttpSession session, Model model){  User login = (User) SecurityUtils.getSubject().getPrincipal();  String userid=login.getId();  List<Score> scoreList=scoreService.selectByAllByUserId(userid);  model.addAttribute("scoreList",scoreList); **return** "grade";}  @RequestMapping(value = "web/grade/{score}/{subject}",method = RequestMethod.GET)  **public** String **grade**(@PathVariable("score")String score,@PathVariable("subject")String subject,HttpSession session, Model model){  Score scoreObject=**new** Score();  scoreObject.setId(**new** UUIDgenarater().getUUID());  scoreObject.setSubject(subject);  User login = (User) SecurityUtils.getSubject().getPrincipal();  **if**(login==**null**){  logger.error("用户未登录");  **return** "login";  }  String userid=login.getId();  scoreObject.setStuid(userid);  scoreObject.setTestscore(Integer.valueOf(score));  **try**{  Date date=**new** Date();  scoreObject.setTime(date.getTime());  }**catch** (Exception e){  logger.error(e.getMessage());  }  scoreService.setBaseDao(scoreMapper);  MsgBean scoremsgBean=scoreService.instert(scoreObject);  **if**(scoremsgBean.isStatus()){  logger.info("添加成绩成功！");  }  List<Score> scoreList=scoreService.selectByAllByUserId(userid);  model.addAttribute("scoreList",scoreList);  **return** "grade"; } |

## 5.6 用户管理页面的实现

当管理员选择用户管理页面时，会进入到管理页面进行添加用户。添加用户内容有用户名、用户密码、电话号码、用户类型。当用户输入相应的信息之后，用户会根据用户的填写进行 对表单的验证，验证成功之后证明没有 ActionError 之后，数据是 validate 的之后，这 些数据会通过 ActionForm 处理，通过 Action，dao 等与数据库进行连接。从而实现对 数据的持久化操作。如果验证不成功，就会跳到本页面进行重新填写。用户管理页面如图 4.4 所示。

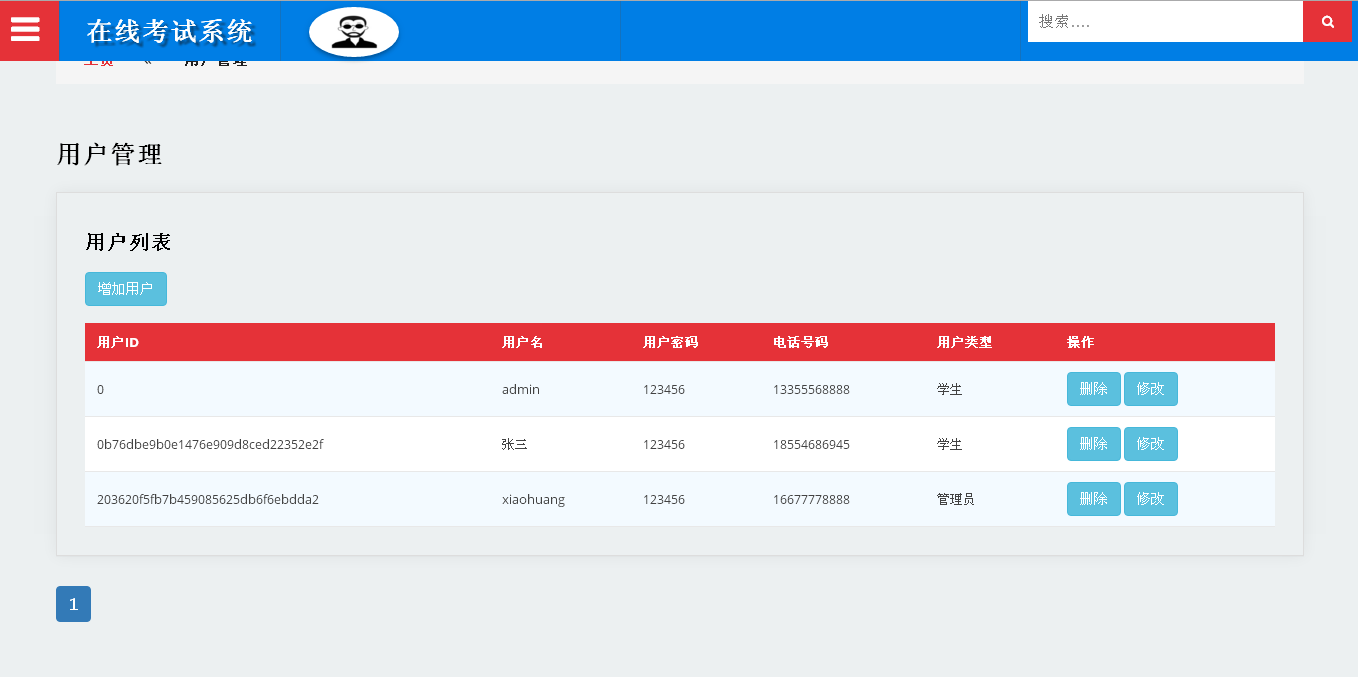


图5-6 用户管理页面

|  |
| --- |
| **public** MsgBean **page**(HttpServletRequest request,**int** page,**int** count) {  String reslut="";  Map<String,Object> map=**new** HashMap<>();  request.getContextPath();  map.put("request",request);  userService.setBaseDao(userMapper);  MsgBean userList=userService.selectAllByPage(page,count);  List<Map> tempList=(List<Map>) userList.getData();  Map tempManp=tempList.get(0);  List list=(List)tempManp.get("data");  map.put("userlist",list);  map.put("pagesize",list.size());  **try** {  reslut=FreemarkerUtils.getTemplate("admin/UserManager.ftl",map);  }**catch** (Exception e){  logger.error(e.toString());  }  Map mappage=**new** HashMap();  mappage.put("page",reslut);  mappage.put("pageindex",page);  mappage.put("pagesize",tempManp.get("totalPage"));  **return** **new** **MsgBean**(**true**,"返回页面成功！",mappage);  } |

## 5.7试题管理页面的实现

管理员可以查看当前已有的试题，可以增加试题，可以删除试题，当然这一切都是 在相应的科目里完成的。首先后台根据页面操作，接收到的默认参数，选择相应的 action 类（SubjectAction），调用相应的操作方法，得到该用户正确的信息。 然后后 台会根据前台页面传过来的 subjectid，调用相应的方法，(ActionForward 后面紧跟的 方法)，对数据库中相应的用户进行修改状态参数，修改成功后调用 query 方法，跳转 前台页面。页面利用 Struts1 标签读取后台的结果集，显示在页面上。下面是图5-7试卷管理页面图。

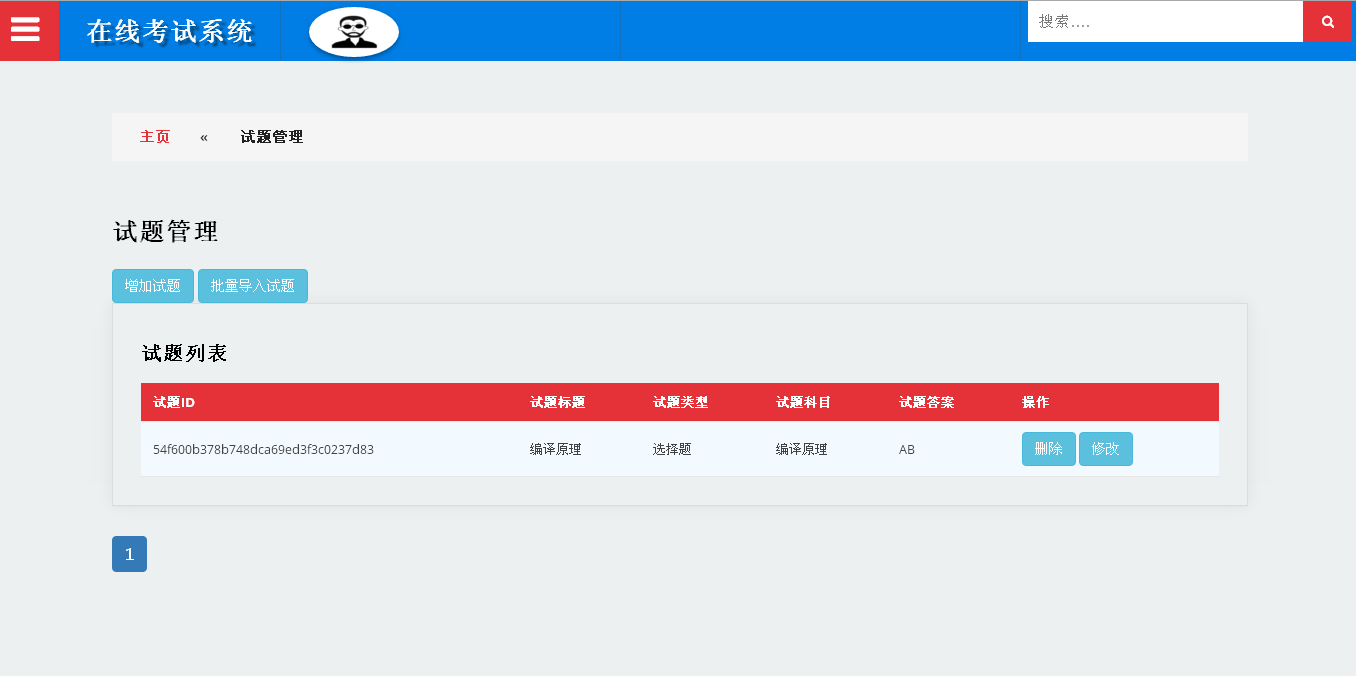


图5-7试卷管理页面图

|  |
| --- |
| **public** MsgBean **page**(HttpServletRequest request,**int** page,**int** count) {  String reslut="";  Map<String,Object> map=**new** HashMap<>();  request.getContextPath();  map.put("request",request);  userService.setBaseDao(userMapper);  MsgBean userList=userService.selectAllByPage(page,count);  List<Map> tempList=(List<Map>) userList.getData();  Map tempManp=tempList.get(0);  List list=(List)tempManp.get("data");  map.put("userlist",list);  map.put("pagesize",list.size());  **try** {  reslut=FreemarkerUtils.getTemplate("admin/UserManager.ftl",map);  }**catch** (Exception e){  logger.error(e.toString());  }  Map mappage=**new** HashMap();  mappage.put("page",reslut);  mappage.put("pageindex",page);  mappage.put("pagesize",tempManp.get("totalPage"));  **return** **new** **MsgBean**(**true**,"返回页面成功！",mappage);} |

## 5.8 试卷管理页面的实现

本页面可以按照考题科目中给出的选项，然后添加题目，各个选择项，正确答案选项，由于本操作提供了单选与多选两种类型，因此在选择正确答案的时候，可以选择多个，刪除时，先选中某一个选项，由于本项与考试科目有关联关系，因此删除时要注意，别轻易的操作，否则会造成系统的异常，修改时，由于多个选项会存在不同的问题。 如下图5-7试卷管理图

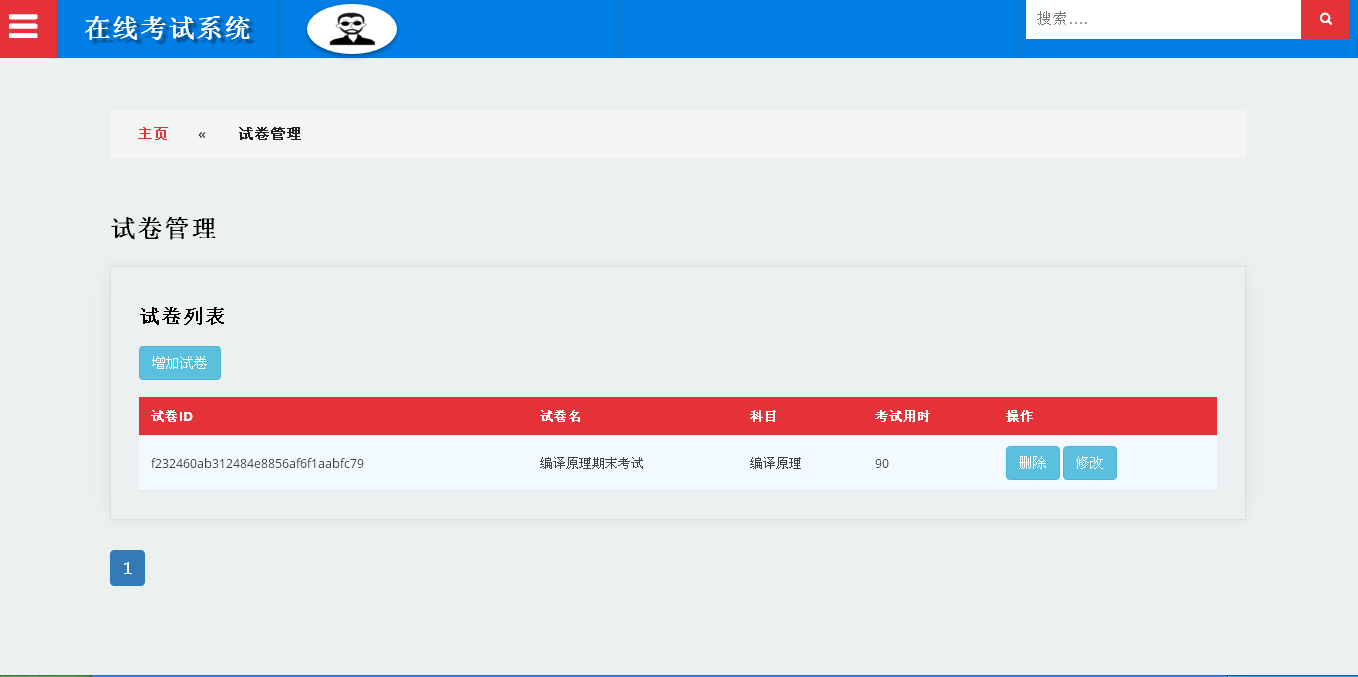


图5-8试卷管理图

|  |
| --- |
| **public** MsgBean **page**(HttpServletRequest request,**int** page,**int** count) {  String reslut="";  Map<String,Object> map=**new** HashMap<>();  request.getContextPath();  map.put("request",request);  testingService.setBaseDao(testingMapper);  MsgBean paperList=testingService.selectAllByPage(page,count);  List<Map> tempList=(List<Map>) paperList.getData();  Map tempManp=tempList.get(0);  List<String> paperids=testpaperMapper.selectAllpaperId();  List list=(List)tempManp.get("data");  map.put("testlist",list);  map.put("paperlist",paperids);  map.put("pagesize",tempManp.get("totalPage"));  **try** {  reslut= FreemarkerUtils.getTemplate("admin/testingManager.ftl",map);  }**catch** (Exception e){  logger.error(e.toString());  }  Map mappage=**new** HashMap();  mappage.put("page",reslut);  mappage.put("pageindex",page);  mappage.put("pagesize",tempManp.get("totalPage"));  **return** **new** **MsgBean**(**true**,"返回页面成功！",mappage);} |