**武汉轻工大学**

**《Web程序设计》大作业报告**

**题目： 酒店管理系统的设计与实现**

**姓 名 陈意飞**

**学 号 1905110056**

**院 （系）数学与计算机学院**

**专 业 软件工程**

**指导教师 董正言**

**2021年12月05日**

1. **任务描述**

设计一个酒店管理系统，该系统的功能描述如下:

（1）客房管理，能够以图形方式直观地显示所有包间的空闲、预订、在用等状态，房间状态的实时显示，查询及统计分析

（2）人员入住登记，包括电话或网上预定的申请。填写必要信息的订单，入住或退房后，改变住房状态，如何合理的根据订单信息自动分配房间；

（3）人员退房，根据房间消费情况进行房费结算

（4）入住人员查询、管理，预定包间的管理分配。

（5）系统用户的管理

对此系统进行简单的需求分析，根据分析结果进行系统设计和数据库设计，完成编码实现，编写作业报告。

**二、需求分析**

**2.1可行性分析**

**2.1.1技术可行性**

本课程设计运用面向对象的开发语言Java在软件Intellij IDEA 2019进行本系统的开发，使用主流的tomcat开源免费的服务器、开源免费的MYSQL数据库。本系统主要完成一些简单业务，如系统用户管理（密码修改）、客户信息管理、客房信息管理、订单信息管理四大功能模块。前端界面主要采用jsp+css+ajax技术进行页面设计，后台采用java按照MVC架构模式来编码实现操作数据库进行增删改查等操作。

**2.1.2操作可行性**

酒店管理系统操作简单，使用人员通过阅读使用说明书可快速上手使用。

**2.2系统用户用例图**

系统用户是酒店管理系统的最高权限用户，主要功能包括：登录系统、客户信息管理、房间信息管理、订单信息管理。

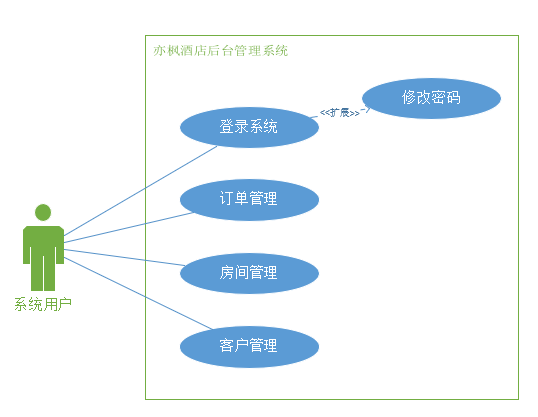


图1.系统用户UML用例图

**2.3功能性需求分析**

1）基础功能：系统用户进行正常登录和密码的修改等。例如前台工作人员对账号的密码修改。

2）房间管理：

A.房间信息的查看、修改和删除

B.添加房间

C.根据房间号查询房间

3)客户管理

A.客户信息的查看、修改和删除

B.添加客户

C.根据客户名和客户电话查询客户

4）订单管理

A.订单信息的查看、修改和删除

B.添加订单

C.根据客户姓名查询订单

**三、系统设计**

**3.1系统模块设计**

**3.1.1总体设计结构**

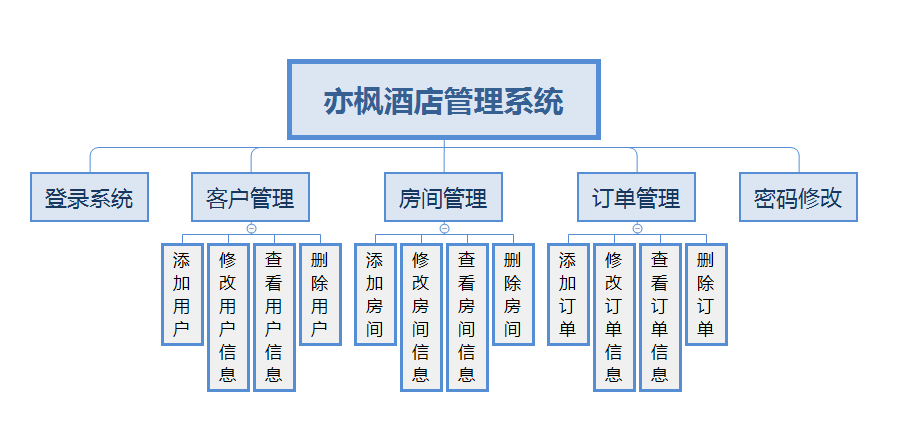


图2.总体设计功能结构图

**3.1.2具体模块数据项**

用户 t\_admin

员工号id 用户名username 密码userPw

客户 t\_customer

客户编号 id 姓名name

地址address 电话tel

房间 t\_room

房间号id 房间号r\_num 房间面积area 房间描述r\_desc 房间类型代码r\_typr\_num

客房类型 r\_room\_typr

类型编号id 名称name 价格price

订单 t\_preserve

订单号id 客房类型代码room\_id 预定时间p\_time

客户姓名c\_name 预定时长days 房间号r\_num

**3.2数据库设计**

**3.2.1数据库E-R图设计**

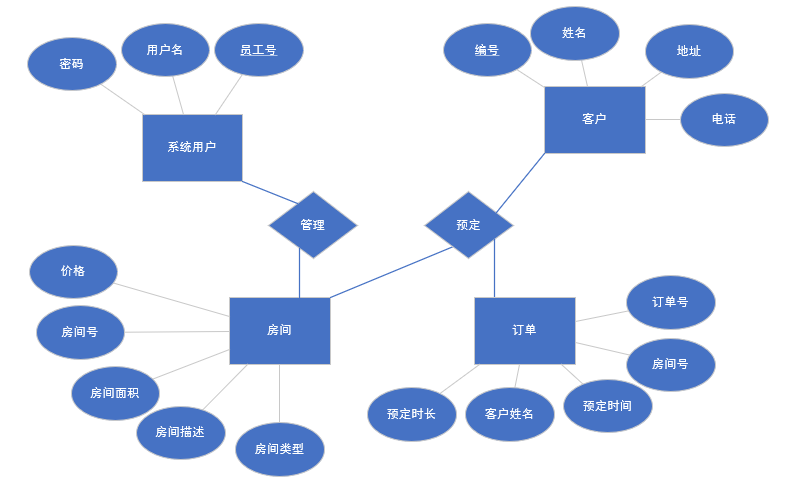


图3.酒店管理系统的数据库E-R图

**3.2.2数据库连接原理**

JDBC（Java Data Base Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。本系统的设计就是采用JDBC技术，将相同的数据库操作封装到BaseDao公共类中，让Dao层去调用。具体原理图如下：

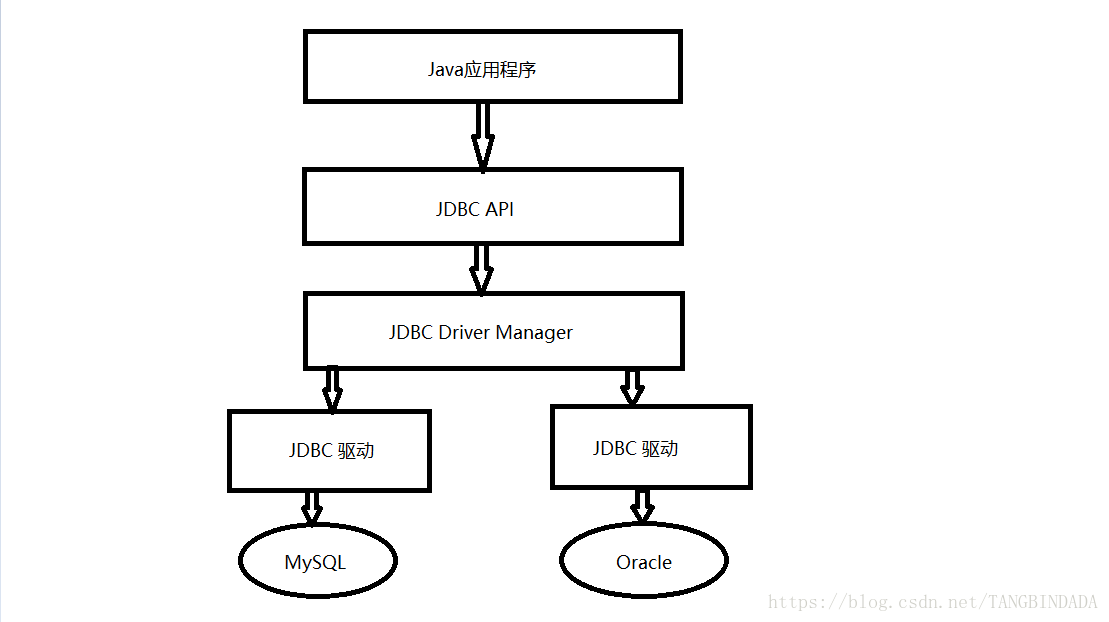


图4.JDBC数据库原理

**3.3设计思想**

1、采用B/S架构。

2、采用面向对象分析方法，按照不同功能进行模块划分。

3、采用MVC设计模式。Model-View-Controller（模型-视图-控制器） 模式。

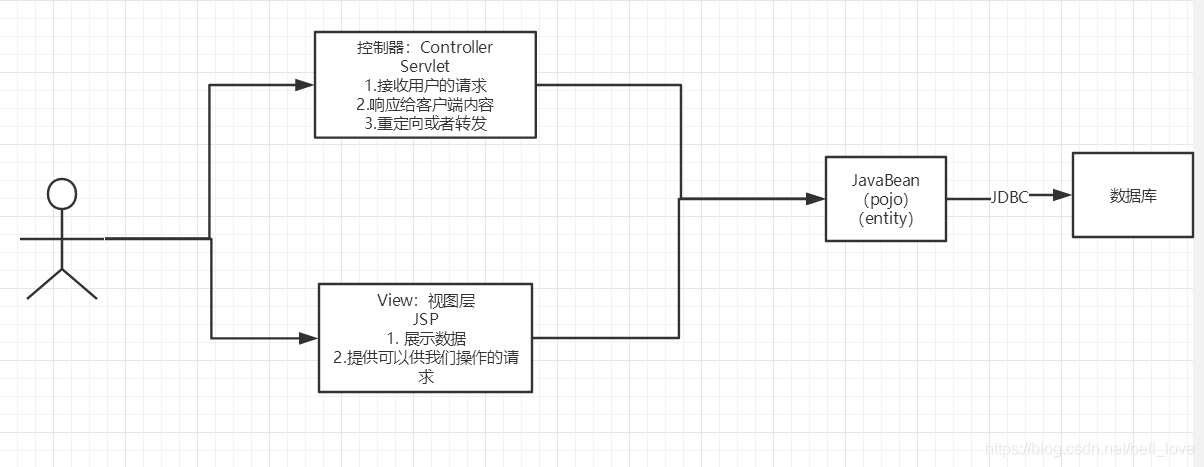
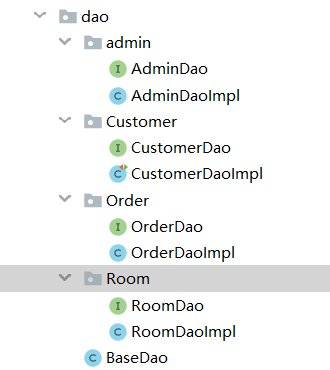
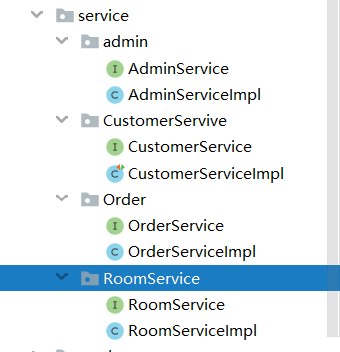


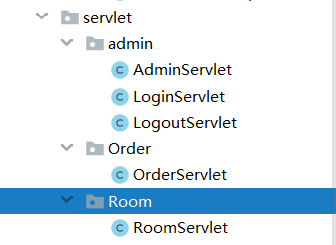
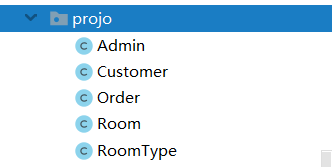
图5.本系统所用MVC架构模式示意图

**四、系统功能实现**

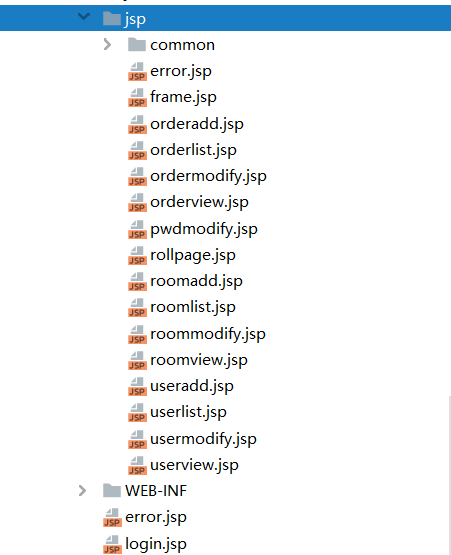
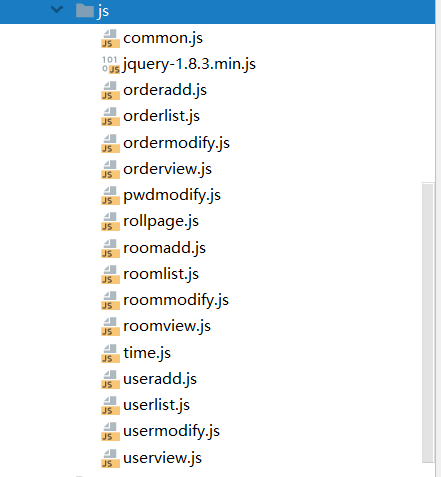
代码部分目录展示：

Dao层 Service层

Servlet层 基类

JSP代码 JS代码

**4.1系统登录界面实现**

注：关键代码大多servlet部分代码，省略异常处理部分，涉及其他部分如Dao层会将整体代码全部展示

通过正确的用户名和密码登录系统。进行登录时，输入用户名和密码进行登录。然后点击登录按钮进行信息的提交。然后我们的后台会寻找相应的service来进行用户名和密码的匹配操作。根据用户名和密码的具体情况，返回登陆结果。登录成功则跳转到主界面。否则会提示用户名或者密码错误

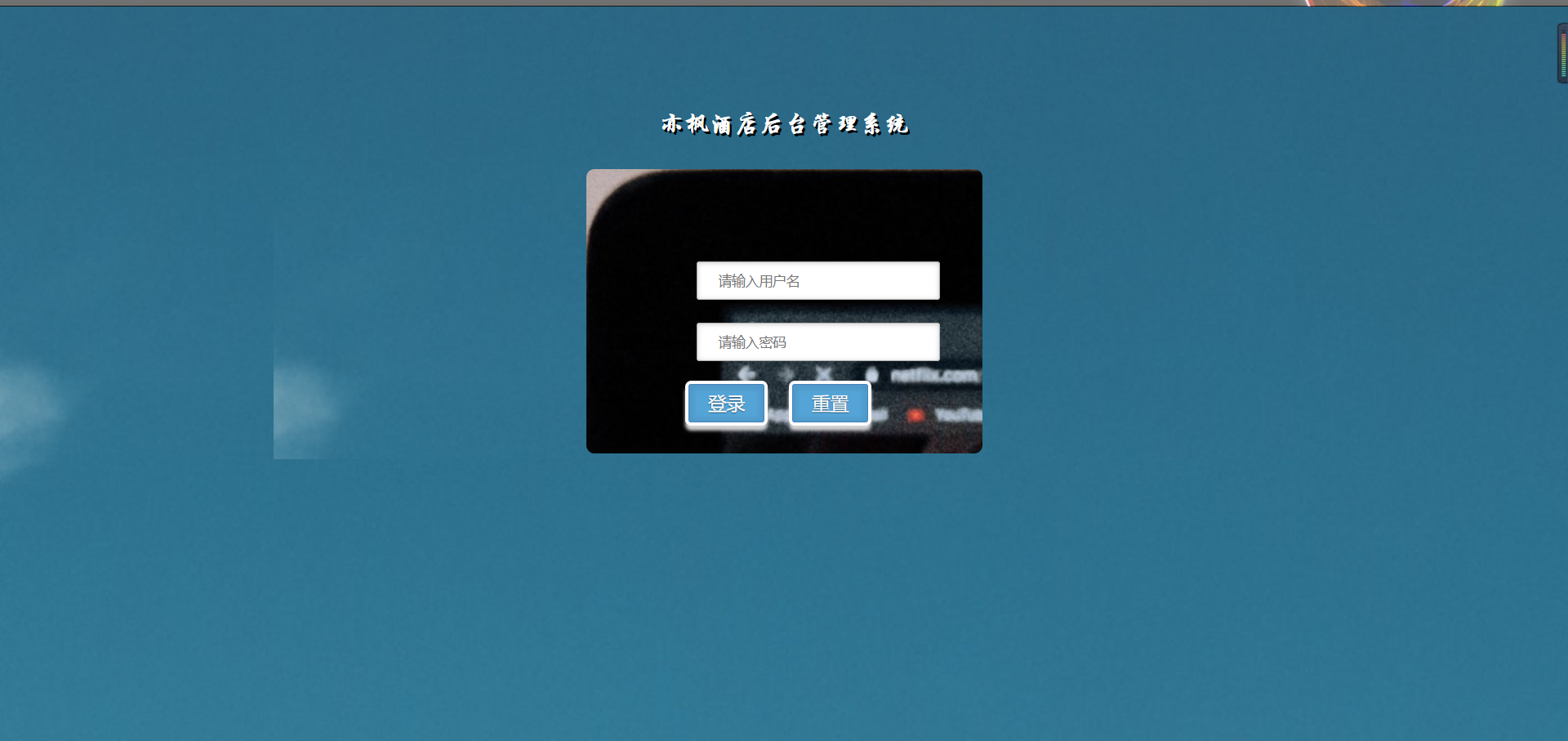


图6.登录界面截图

为确保系统的安全性，要保证只有用户登录后才可以正确访问主界面，我们需要进行登陆拦截（具体实现和4.2部分代码一致）。在用户未完成登录之前，不能访问主界面。如果访问了则跳转到提示界面：



图7.错误界面截图

关键代码：

String userName = req.getParameter("userName");  
 String userPw = req.getParameter("userPw");  
 //和数据库中的密码对比  
 AdminServiceImpl adminService = new AdminServiceImpl();  
 Admin admin= adminService.login(userName, userPw);

//已经把登录的人查出来了  
 if (admin != null) {//茶油此人，可以登录  
 //将用户的信息存在session;  
 req.getSession().setAttribute(Constants.ADMIN\_SESSION,admin);  
 //跳转到内部主页  
 resp.sendRedirect("jsp/frame.jsp");  
 }else {  
 //查无此人。无法登录  
 //转发回登录页面，提示用户名或密码错误  
 req.setAttribute("error","用户名或者密码不正确");  
 req.getRequestDispatcher("login.jsp").forward(req,resp);  
 }

**4.2主界面实现：**

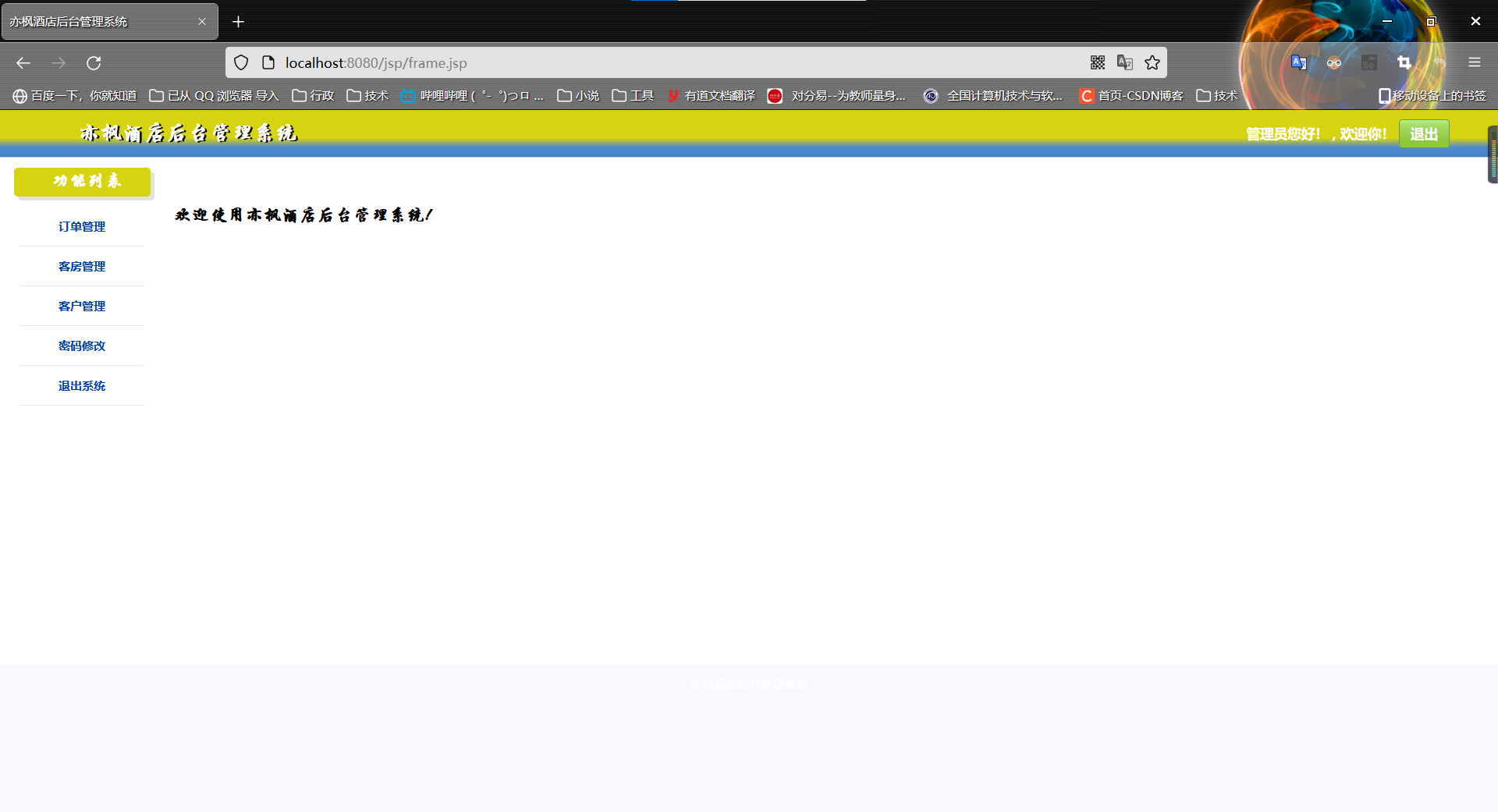


图8.主界面截图

为了保证安全，本系统对每个功能页面都进行了拦截，编写了过滤器代码，判断session中是否有该用户主要代码如下：

**关键代码：**

HttpServletRequest request = (HttpServletRequest) req;  
 HttpServletResponse response = (HttpServletResponse) resp;  
 //过滤器，从Session中获取用户  
Admin admin=

(Admin)request.getSession().getAttribute(Constants.ADMIN\_SESSION);  
 if(admin==null){  
 //已经被移除或注销  
 response.sendRedirect("/error.jsp");  
 }else {  
 filterChain.doFilter(req,resp);  
 }

**4.3订单管理模块实现**



图9.订单管理界面

**4.3.1添加订单：**

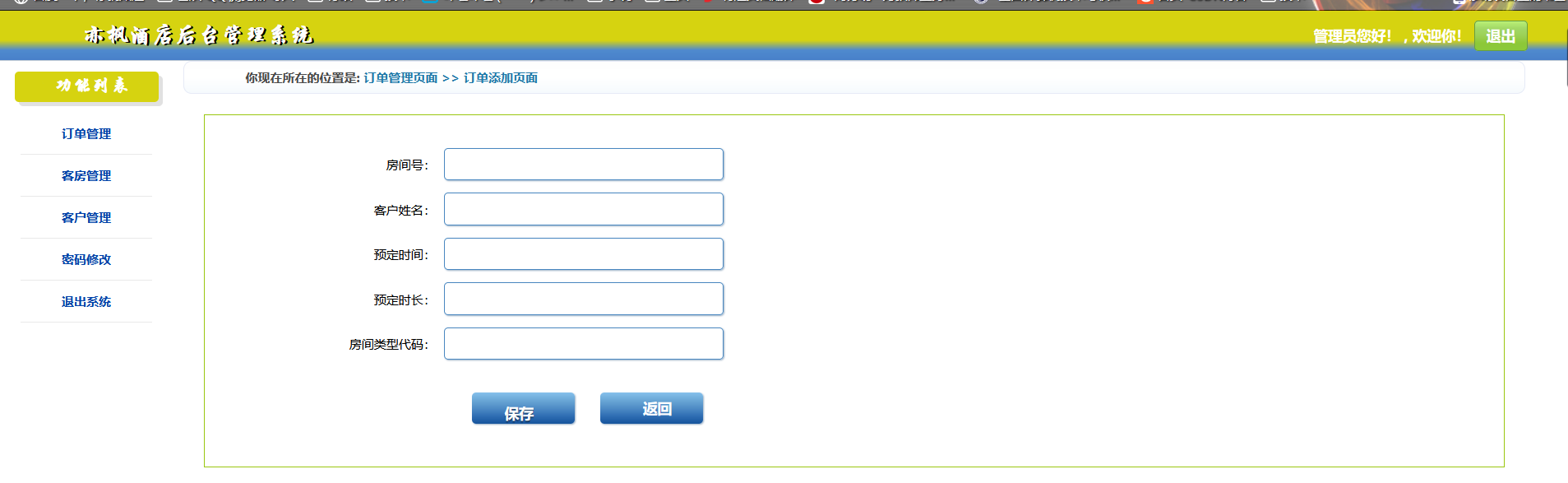


图10.添加订单界面

**关键代码：**

//从前端获取数据

String roomNum = request.getParameter("roomNum");  
 String cName = request.getParameter("cName");  
 String pTime = request.getParameter("pTime");  
 String days = request.getParameter("days") ;  
 String num = request.getParameter("typeNum");  
 int typeNum =Integer.parseInt(num);  
 Order order = new Order();  
 order.setRoomNum(roomNum);  
 order.setC\_name(cName);  
 order.setP\_time(pTime);  
 order.setDays(days);  
 order.setRoom\_id(typeNum);  
 boolean flag = false;  
 OrderService orderService = new OrderServiceImpl();//从数据库获取数据  
 flag = orderService.add(order);//进行添加  
 if(flag){  
response.sendRedirect(request.getContextPath()+"/jsp/order.do?method=query");  
 }else{  
request.getRequestDispatcher("orderadd.jsp").forward(request, response);  
 }  
}

**4.3.2查看订单信息**



图11.订单信息详情页

关键代码：

String id = request.getParameter("orderid");  
 if(!StringUtils.isNullOrEmpty(id)){  
 OrderService orderService = new OrderServiceImpl();  
 Order order = null;  
 order = orderService.getOrderById(id);  
 request.setAttribute("order", order);  
 request.getRequestDispatcher(url).forward(request, response);  
 }

**4.3.3修改订单信息：**



图12.订单修改页

**关键代码：**  
 String id = request.getParameter("id");  
 String rNum = request.getParameter("rNum");  
 String cName = request.getParameter("cName");  
 String pTime = request.getParameter("pTime");  
 String days = request.getParameter("days");  
  
 Order order = new Order();  
 order.setId(Integer.valueOf(id));  
 order.setRoomNum(rNum);  
 order.setC\_name(cName);  
 order.setP\_time(pTime);  
 order.setDays(days);  
  
 boolean flag = false;  
 OrderService orderService = new OrderServiceImpl();  
 flag = orderService.modify(order);  
 if(flag){

response.sendRedirect(request.getContextPath()+"/jsp/order.do?method=query");  
 }else{

request.getRequestDispatcher("ordermodify.jsp").forward(request, response);  
 }

**4.3.4删除订单：**

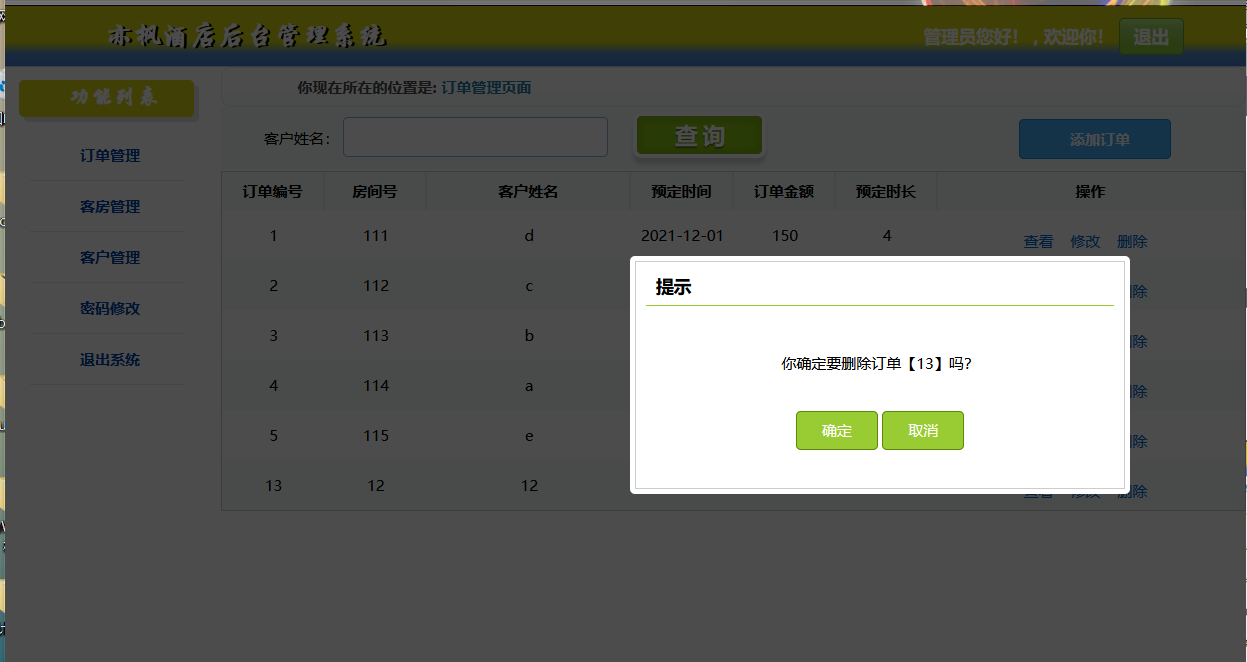


图13.删除订单

**关键代码：**

String id = request.getParameter("orderid");  
 HashMap<String, String> resultMap = new HashMap<String, String>();  
 if(!StringUtils.isNullOrEmpty(id)){  
 OrderService orderService = new OrderServiceImpl();  
 boolean flag = orderService.deleteOrderById(id);  
 if(flag){//删除成功  
 resultMap.put("delResult", "true");  
 }else{//删除失败  
 resultMap.put("delResult", "false");  
 }  
 }else{  
 resultMap.put("delResult", "notexit");  
 }  
 //把resultMap转换成json对象输出  
 response.setContentType("application/json");  
 PrintWriter outPrintWriter = response.getWriter();  
 outPrintWriter.write(JSONArray.toJSONString(resultMap));  
 outPrintWriter.flush();  
 outPrintWriter.close();

注：本系统的删除功能均应用了ajax。使用了ajax技术进行时实时更新：

以下为orderlist.js文件所包含的内容：

//订单管理页面上点击删除按钮弹出删除框(orderlist.jsp)  
function deleteBill(obj){  
 $.ajax({  
 type:"GET",  
 url:path+"/jsp/order.do",  
 data:{method:"delorder",orderid:obj.attr("billid")},  
 dataType:"json",  
 success:function(data){  
 if(data.delResult == "true"){//删除成功：移除删除行  
 cancleBtn();  
 obj.parents("tr").remove();  
 }else if(data.delResult == "false"){//删除失败 changeDLGContent("对不起，删除订单【"+obj.attr("billid")+"】失败");  
 }else if(data.delResult == "notexist”; changeDLGContent("对不起，订单【"+obj.attr("billid")+"】不存在");  
 }  
 },  
 error:function(data){  
 alert("对不起，删除失败");  
 }  
 });

**4.4．客房管理模块**



图14.客房管理界面

**4.4.1添加房间：**



图15.添加房间

**关键代码：**

String roomNum=req.getParameter("roomNum");  
 String roomArea=req.getParameter("roomArea");  
 String roomDesc=req.getParameter("roomDesc");  
 String roomType=req.getParameter("roomType");  
  
 Room room = new Room();  
 room.setNum(roomNum);  
 room.setArea(roomArea);  
 room.setDesc(roomDesc);  
 room.setType(roomType);  
 boolean flag = false;  
 RoomService roomService = new RoomServiceImpl();  
 flag = roomService.add(room);  
 if(flag){  
 resp.sendRedirect(req.getContextPath()+"/jsp/room.do?method=query");  
 }else{  
 req.getRequestDispatcher("roomadd.jsp").forward(req, resp);  
 }

**4.4.2查看房间信息：**



图16.房间信息详情页

**关键代码：**

String id = req.getParameter("roomid");  
 if (!StringUtils.isNullOrEmpty(id)) {  
 RoomService roomService = new RoomServiceImpl();  
 Room room = null;  
 room = roomService.getRoomById(Integer.valueOf(id));  
 req.setAttribute("room", room);  
 req.getRequestDispatcher(url).forward(req, resp);  
 }

**4.4.3修改房间信息**

说明：修改信息页需要展示本已经存储的信息，在此基础上进行修改



图17.修改房间信息页

**关键代码：**

String rNum = request.getParameter("rNum");  
 String rArea = request.getParameter("rArea");  
 String rDesc = request.getParameter("rDesc");  
 String rTypeNum = request.getParameter("rTypeNum");  
  
 String idt = request.getParameter("id");  
 Integer id=null;  
 Room room = new Room();  
 if(!StringUtils.isNullOrEmpty(idt)){  
 id=Integer.parseInt(idt);  
 }  
 room.setId(id);  
 room.setNum(rNum);  
 room.setArea(rArea);  
 room.setDesc(rDesc);  
 room.setType(rTypeNum);  
 boolean flag = false;  
 RoomService roomService = new RoomServiceImpl();  
 flag = roomService.modify(room);  
 if(flag){

response.sendRedirect(request.getContextPath()+"/jsp/room.do?method=query");  
 }else{  
 request.getRequestDispatcher("roommodify.jsp").forward(request, response); }}

**4.4.4根据房间号查询房间信息：**



图18.条件查询页面

**关键代码：**

String num = request.getParameter("num");  
 if(StringUtils.isNullOrEmpty(num)){  
 num= "";  
 }  
  
 List<Room> roomList = new ArrayList<Room>();  
  
 RoomService roomService = new RoomServiceImpl();  
 roomList = roomService.getRoomList(num);  
 request.setAttribute("roomList", roomList);  
 request.setAttribute("num", num);  
 request.getRequestDispatcher("roomlist.jsp").forward(request, response);  
}

这个部分的Dao层涉及多表查询操作，具体语句为：

select r.\*,t.price,t.name from t\_room r,t\_room\_type t where r.r\_type\_num=t.id

目的在于通过房间类型id获取房间的价格和类型名称

关键代码：

if(connection != null){  
 StringBuffer sql = new StringBuffer();  
 sql.append("select r.\*,t.price,t.name from t\_room r,t\_room\_type t where r.r\_type\_num=t.id");  
 List<Object> list = new ArrayList<>();  
 if(!StringUtils.isNullOrEmpty(num)){  
 sql.append(" and r.r\_num like ?");  
 list.add("%"+num+"%");  
 }  
 Object[] params = list.toArray();

**4.5客户管理模块：**

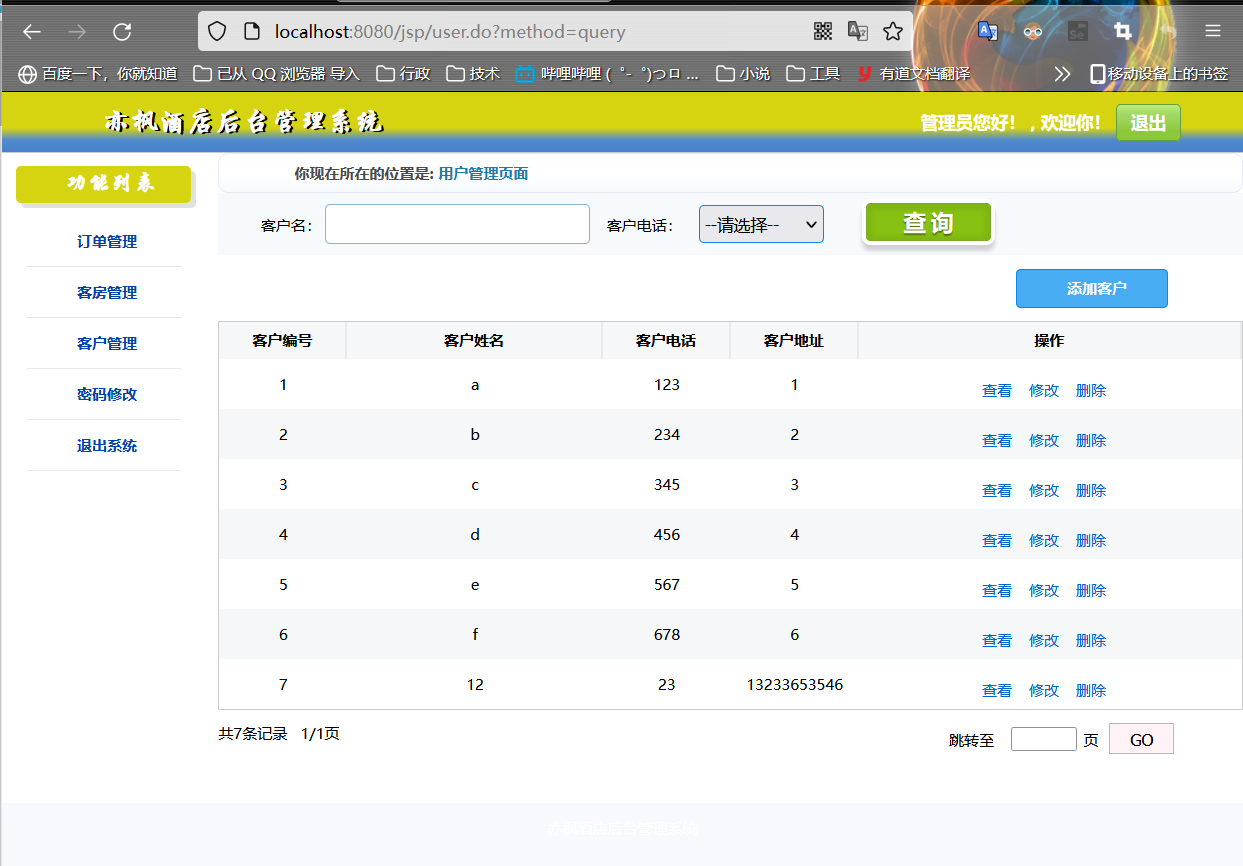


图19.客户管理界面

**4.5.1添加客户**



图20.添加客户界面

**关键代码：**

System.out.println("add()================");  
 String customerName = req.getParameter("userName");  
 String phone = req.getParameter("phone");  
 String address = req.getParameter("address");  
  
 Customer customer = new Customer();  
 customer.setName(customerName);  
 customer.setTel(phone);  
 customer.setAaddress(address);  
  
// UserService userService = new UserServiceImpl();  
 CustomerService customerService = new CustomerServiceImpl();  
 if (customerService.add(customer)) {  
 resp.sendRedirect(req.getContextPath() + "/jsp/user.do?method=query");  
 } else {  
 req.getRequestDispatcher("/useradd.jsp").forward(req, resp);  
 }

**4.5.2查看客户信息：**  


图21.客户信息详情页

**关键代码：**

String id = req.getParameter("uid");  
 if(id!=null){  
 //调用后台方法得到user对象  
 CustomerService customerService = new CustomerServiceImpl();  
 Customer customer = customerService.getUserById(id);  
 req.setAttribute("customer", customer);  
 req.getRequestDispatcher(url).forward(req, resp);  
 }  
}

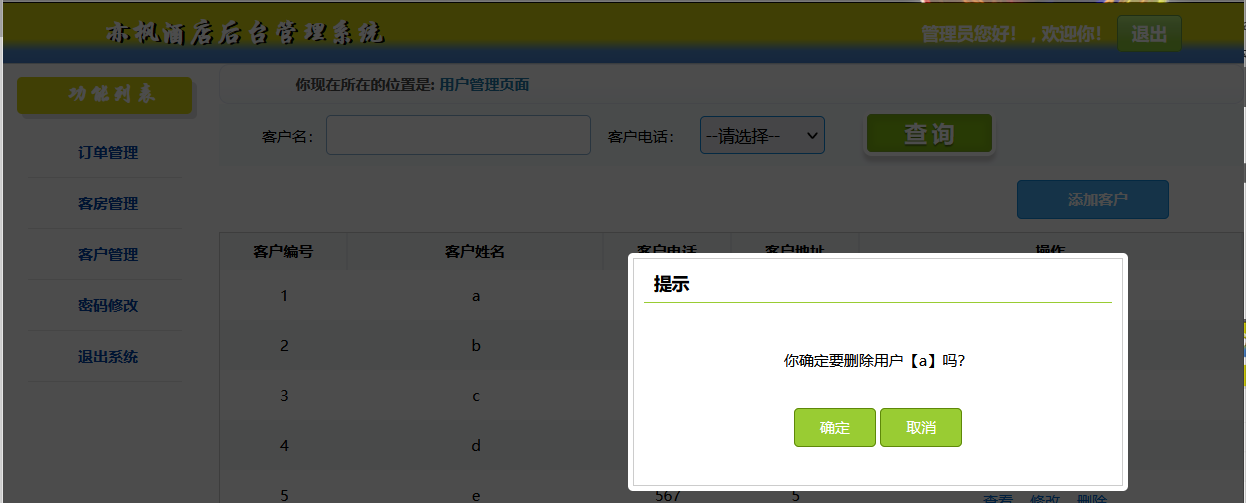
**4.5.3删除客户：**  


图22.删除客户

**关键代码：**  
 String id = request.getParameter("id");  
 Integer delId = 0;  
 try{  
 delId = Integer.parseInt(id);  
 }catch (Exception e) {  
 delId = 0;  
 }  
 HashMap<String, String> resultMap = new HashMap<String, String>();  
 if(delId <= 0){  
 resultMap.put("delResult", "notexist");  
 }else{  
 CustomerService customerService = new CustomerServiceImpl();  
 if(customerService.deleteCustomerById(delId)){  
 resultMap.put("delResult", "true");  
 }else{  
 resultMap.put("delResult", "false");  
 }  
  
 //把resultMap转换成json对象输出  
 response.setContentType("application/json");  
 PrintWriter outPrintWriter = response.getWriter();  
 outPrintWriter.write(JSONArray.toJSONString(resultMap));  
 outPrintWriter.flush();  
 outPrintWriter.close();  
}

**4.5.4根据客户名和电话查询客户信息：**  


图23.条件查询客户信息

这个部分使用了分页技术，使用了工具类PageSupport

**关键代码：**

public void query(HttpServletRequest req,HttpServletResponse resp){  
 //查询客户列表  
 //从前端获取数据  
 String queryName = req.getParameter("queryname");  
 String temp = req.getParameter("querytel");  
 String pageIndex = req.getParameter("pageIndex");  
 String querytel=null;  
  
 CustomerServiceImpl customerService = new CustomerServiceImpl();  
 int pageSize=8;  
 int currentPageNo=1;  
 List<Customer> customerList=null;  
 if(queryName==null){  
 queryName="";  
 }  
 if(temp!=null && !temp.equals("")){  
 querytel= temp;  
 }  
 if(pageIndex!=null){  
  
 currentPageNo = Integer.parseInt(pageIndex);  
 }  
 //获取客户的总数(分页：）  
 int totalCount = customerService.getCustomerCount(queryName, querytel);  
 PageSupport pageSupport=new PageSupport();  
 pageSupport.setCurrentPageNo(currentPageNo);  
 pageSupport.setPageSize(pageSize);  
 pageSupport.setTotalCount(totalCount);  
// int totalPageCount = pageSupport.getTotalCount();  
 int totalPageCount=(int)(totalCount/pageSize)+1;  
 //控制首页和尾页  
 if(totalPageCount<1){  
 currentPageNo=1;  
 }else if(currentPageNo>totalPageCount){  
 currentPageNo=totalPageCount;  
 }//避免500  
 //获取客户列表展示  
 customerList = customerService.getCustomerList(queryName, querytel, currentPageNo, pageSize);  
 req.setAttribute("userList",customerList);  
 req.setAttribute("roleList",customerList);  
  
 req.setAttribute("totalCount",totalCount);  
 req.setAttribute("currentPageNo",currentPageNo);  
 req.setAttribute("totalPageCount",totalPageCount);  
 req.setAttribute("queryName", queryName);  
 req.setAttribute("querytel", querytel);  
 //fanhui  
 try {  
 req.getRequestDispatcher("/jsp/userlist.jsp").forward(req,resp);  
 } catch (ServletException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 } }

**4.6密码修改模块**

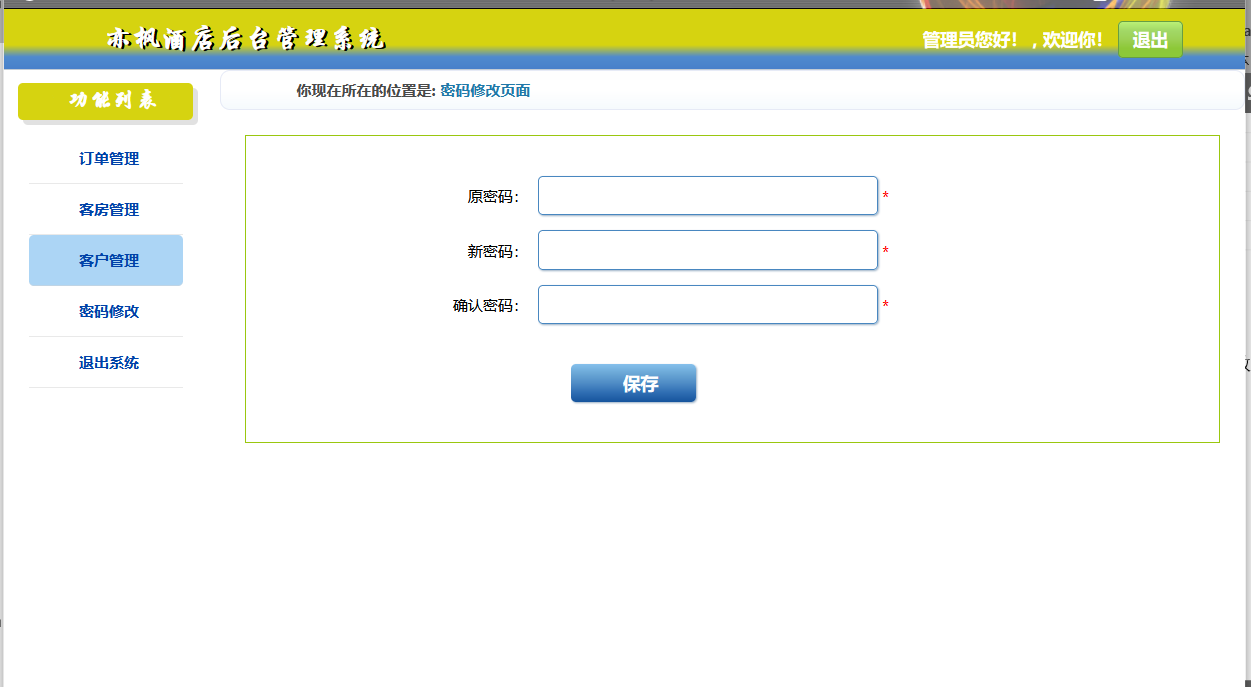


图24.密码修改界面

**关键代码：**

//修改密码  
public void updatePwd(HttpServletRequest req, HttpServletResponse resp){  
 // 从session获取id  
 Object o = req.getSession().getAttribute(Constants.ADMIN\_SESSION);  
 String newpassword = req.getParameter("newpassword");  
 boolean flag=false;  
 if(o!=null&& newpassword!=null){  
 AdminService adminService = new AdminServiceImpl();  
 //强转成Admin对象后获取id

flag = adminService.updatePw(((Admin) o).getUserId(), newpassword);  
 if(flag){  
 req.setAttribute("message","修改密码成功，请退出，使用新密码登录");  
 //密码修改成功，移除session  
 req.getSession().removeAttribute(Constants.ADMIN\_SESSION);  
 }else {  
 req.setAttribute("message","密码修改失败");  
 }  
 }else {  
 req.setAttribute("message","新密码有问题");  
 }  
 req.getRequestDispatcher("/jsp/pwdmodify.jsp").forward(req,resp);  
   
//验证旧密码,Session中有用户的密码  
public void pwdModify(HttpServletRequest req, HttpServletResponse resp){  
 Object o = req.getSession().getAttribute(Constants.ADMIN\_SESSION);  
 String oldpassword = req.getParameter("oldpassword");  
 System.out.println(oldpassword);  
 //Map封装数据  
 Map<String, String> resultMap = new HashMap<String,String>();  
 if(o==null){//Session失效了或者过期了  
 resultMap.put("result","sessionerror");  
 }else if(StringUtils.isNullOrEmpty(oldpassword)){//输入的密码为空  
 resultMap.put("result","error");  
 }else {  
 String uerPw = ((Admin) o).getUerPw();  
 System.out.println(uerPw);  
 if(oldpassword.equals(uerPw)){  
 resultMap.put("result","true");  
 }else {  
 resultMap.put("result","false");  
 }  
 }  
 resp.setContentType("application/json");  
 PrintWriter writer = resp.getWriter();  
 //alibabba的JASON工具类,转换格式  
 writer.write(JSONArray.toJSONString(resultMap));  
 writer.flush();  
 writer.close();

**4.7注销系统模块**

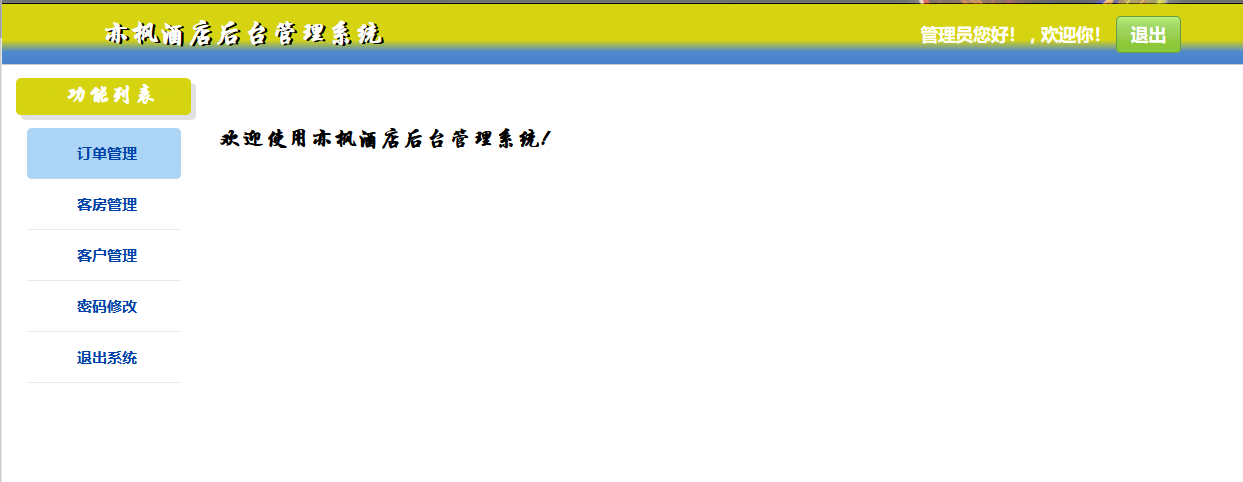


图25.系统注销

注销系统可以通过两种方式：左侧功能列表的“退出系统”和右上角的“退出”。用户退出时应移除用户的session，防止退出后还可以进入系统主界面，造成安全问题。

**关键代码：**

protected void doGet(HttpServletRequest req, HttpServletResponse resp) {  
req.getSession().removeAttribute(Constants.ADMIN\_SESSION);//移除用户的session  
 resp.sendRedirect(req.getContextPath()+"/login.jsp");//返回登录界面  
}

**五、系统调试**

**5.1解决中文乱码和字符问题**

为解决字符乱码和字符集编码不统一的问题，编写字符集编码过滤器，并在web.xml中注册

**public void** doFilter(ServletRequest request, ServletResponse response, FilterChain chain) **throws** IOException, ServletException {  
 request.setCharacterEncoding(**"utf-8"**);  
 response.setCharacterEncoding(**"utf-8"**);  
 chain.doFilter(request,response);  
}

**5.2前后端接口不一致**

在编写servlet部分编码时应注意，getParameter（）的参数名要和前端页面的name保持一致，否则会出错。并且在传入为字符时，应视情况考虑使用StringUtils.isNullorEmpty（）对其进行判空操作。如果需要的目标格式为Integer类型，那么还要考虑将传入的String类型判空后利用Integer.valueOf()转成Integer类型后，才可以传入service层。

**5.3前端文件路径找不到**

在初期，我没有使用pageContext.request.contextPath寻找文件，导致经常出现文件路径找不到的情况。不论是寻找js文件，还是images文件都应尽可能的使用pageContext.request.contextPath对文件进行绝对路径的查找，以达到准确地寻找到目标文件。

**六、课设总结**

我本次课程设计的题目为酒店管理系统，我的系统名称为“亦枫酒店后台管理系统”，本次课设让我收获颇多。首先是MVC模型，MVC模型可以使图形界面和后台逻辑以及数据库的操作分离开，就如在设计思想部分的示意图所示，当我们修改前端页面的时候，后端的逻辑没有变化的就不需要修改后台代码。在本系统中就体现的很突出，JSP和CSS只用进行页面交互和呈现页面即可，需要数据的时候就向后台service层请求数据，service再调用Dao层在数据库中进行增删改查，最后将结果返回到前端显示。这样的设计模式方便后续对代码的修改。本系统在很多地方还存在不足，仍需进一步优化，例如，还应该增加房间类型管理模块等。MVC模型对于后续的深入开发和优化有着先天的便利性。

本系统主要应用的开发技术有：JS、JSP、HTML、JDBC、ajax、Servlet、MYSQL、MVC等基础Java开发技术以及Maven项目管理工具。本次课程让我对这些技术有了新的学习和回顾，一次综合运用让我对这些零碎的知识有了一次整合的机会，让我掌握的更加深刻，收获颇多。

通过本次酒店管理系统的设计，也让我完成了一次完整的系统开发的过程，从需求分析、系统设计、数据库设计，再到前后端编码实现和单元测试、集成测试。让我对本学期所学的多门课程都有了一次综合的应用和实践，锻炼了我的自学能力和综合能力。也增强了自己继续学习的东西和信心。由于时间和技术的关系，本系统仍存在很多不足，因此我将此项目上传到了自己的Github仓库进行维护，以期待在学习完高级框架之后对其进行改造和完善。

**评分表**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **学生姓名** | **陈意飞** | | | **班级** | | 软工1901 | | | **学号** | 1905110056 |
| **评分项目** | **评分档次及分值** | | | | | | | | | **评分** |
| **分值** | **A** | **B** | | **C** | | **D** | **E** | |
| 学习态度 | 10 | 10 | 9 | | 8 | | 7 | <6 | |  |
| 分析与设计 | 30 | 30-27 | 27-24 | | 24-21 | | 21-18 | <18 | |  |
| 代码质量 | 30 | 30-27 | 27-24 | | 24-21 | | 21-18 | <18 | |  |
| 报告撰写质量和格式 | 30 | 30-27 | 27-24 | | 24-21 | | 21-18 | <18 | |  |
| **总 分** | | | | | | | | | |  |