CSYE6220 final project
hotel management system
Yue liu

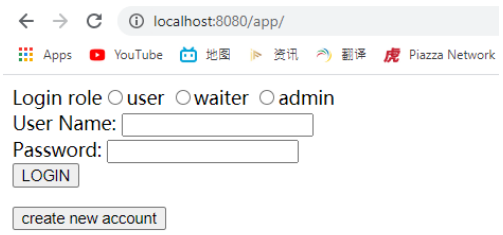
Summary:
This application can be used in a Hotel daily operation and maintenance scene. It has 3 roles: user, waiter and admin. Users can book room, check out, view order and view bill. Waiter can manage the user orders and hotel room, then send bill to customers. Admin can manage the waiter statement.

Functionality Performed:
In this application, it has 7 basic functions.

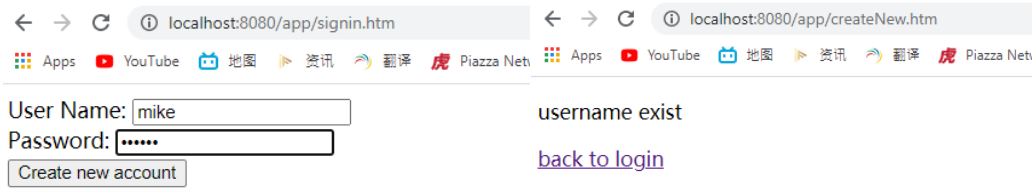
1. *book rooms*
   user can book 3 kinds of room and choose the start data and end date they want.
2. *view orders*
   user can view all their orders and change or remove them before check in. user also can see the live history and check out from the current living room.
3. *view bills*
   user can see all the bill that sending form waiter
4. *manage user order*
   waiter can manage all user orders and let them check in or check out
5. *manage room*
   waiter can see all the used room and reset its state
6. *manage bill*
   waiter can send the bill to user, the bill is calculate by the days between check in date and check out date
7. *manage waiter*
   admin can see all the waiter state, and add new waiter

Technologies used:
Spring MVC, Hibernate, Hibernate advance mapping, HQL, Annotation, JQUERY, AJAX, Validation, Interceptor

User Roles Performed:
1. **Register and login**
   The home page is the login page, user need to choose the login in role, then enter the username and password. If you are new, click “create new account” to create a user account.

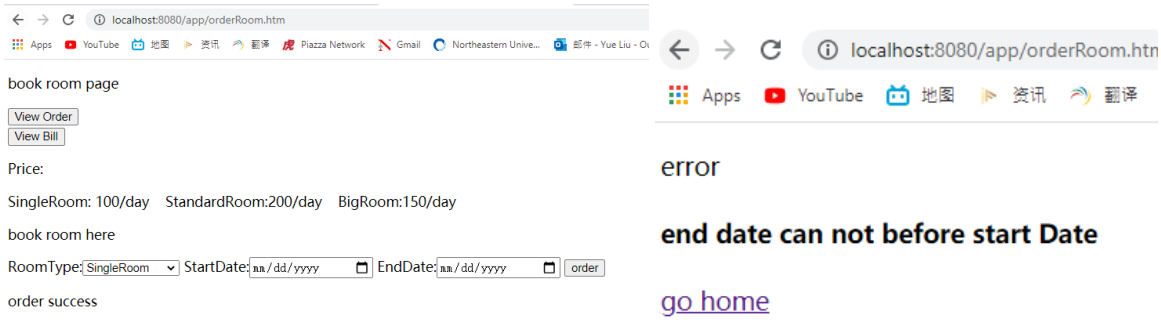


This is the register page, when create success, it will back to the login page. Once you enter the unsafe input or exist username, the register fails, it will go to the error page. This is the error page.

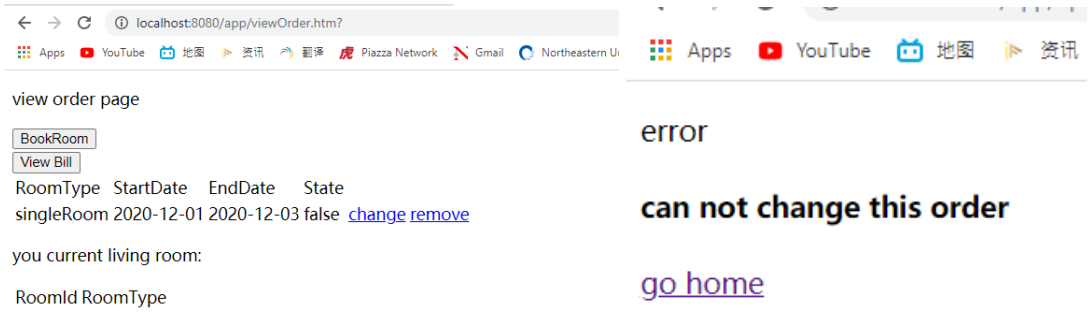


2. User

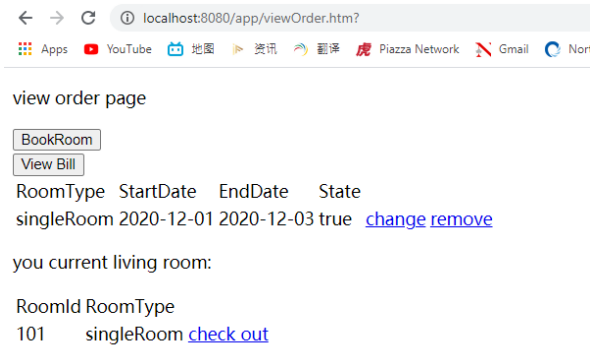
When you login in as a user role. It will come to the user book room page. You can see the room price and book room here. When you book room successfully, you can see the notification. Once you enter the end date before the start data, it will cause error



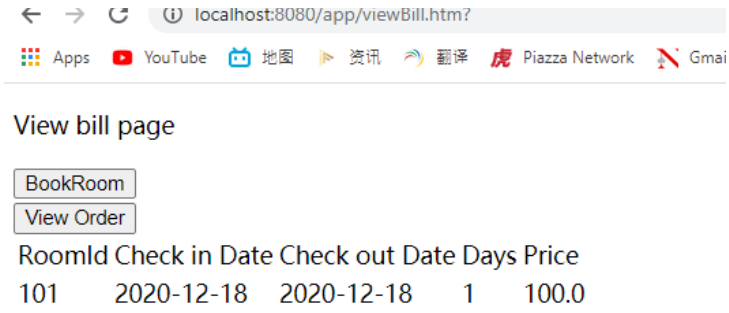
You can click the “view order” button to view you order and current room. If the order state is “false”, that means this order are not processed. So you can change it or remove it. Once waiter processed it, you can not change it.



Once the waiter let you check in, the room number will show below. You can check out by yourself or let waiter check out. If the room is already checked out, you can not check out again.

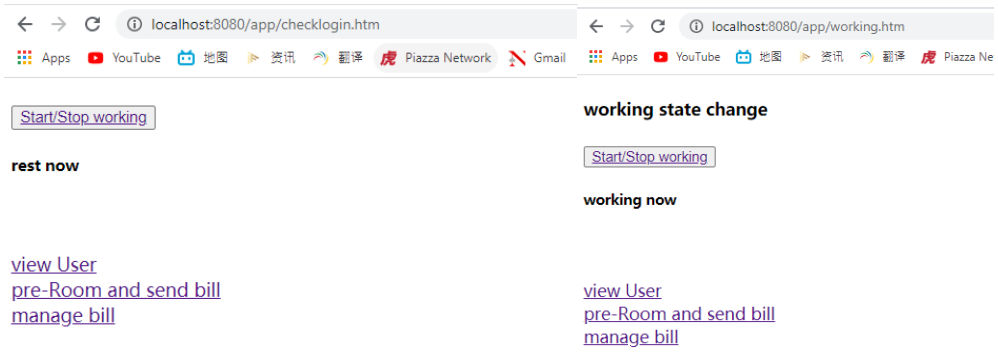


Once the waiter send the bill. You can see you bill in view bill page.

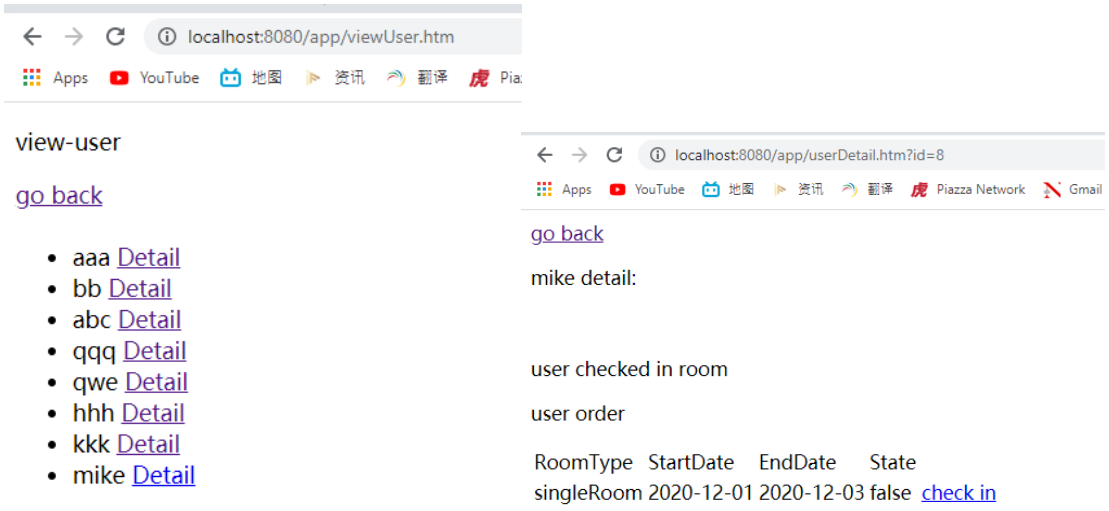


3. Waiter

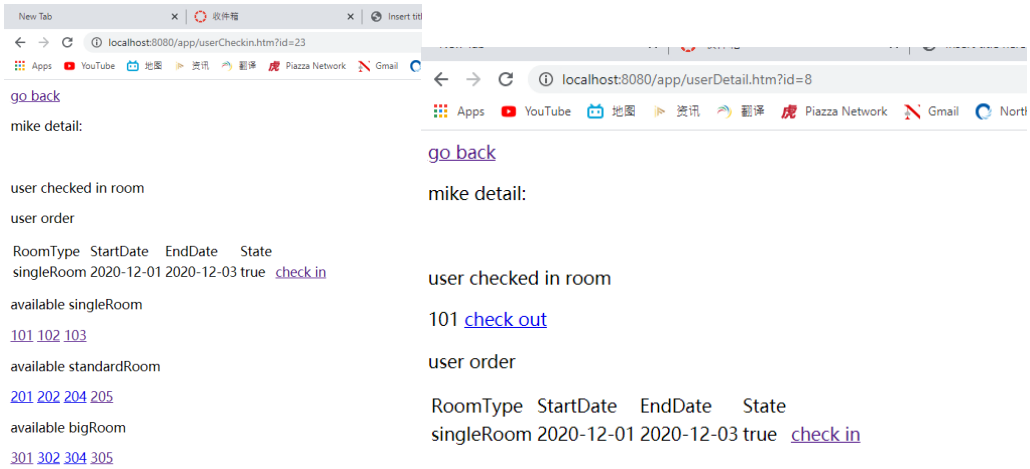
When you login as a waiter, you can go to the waiter manage page. When you click the “Start/stop working” button, it will change the waiter state.



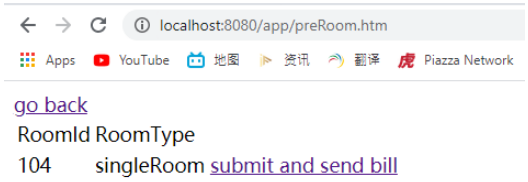
When you click view user you can see all the user names, and click “detail”, you can see the user order details.



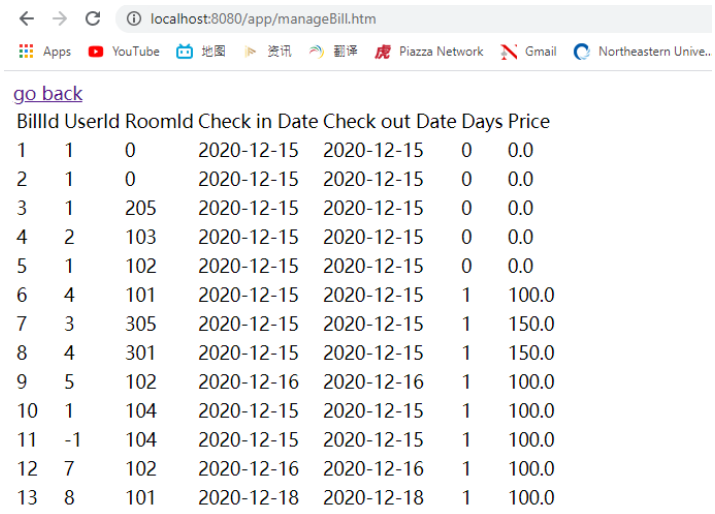
Click “check in”, you can see all available rooms, and click the room number to assign it to user. It will shows in user checked in room. Once a order is checked in, you can not check in again.



Once a room is check out, waiter can go to the prepare room page to reset the room state and send bill to user. The room will be available again.



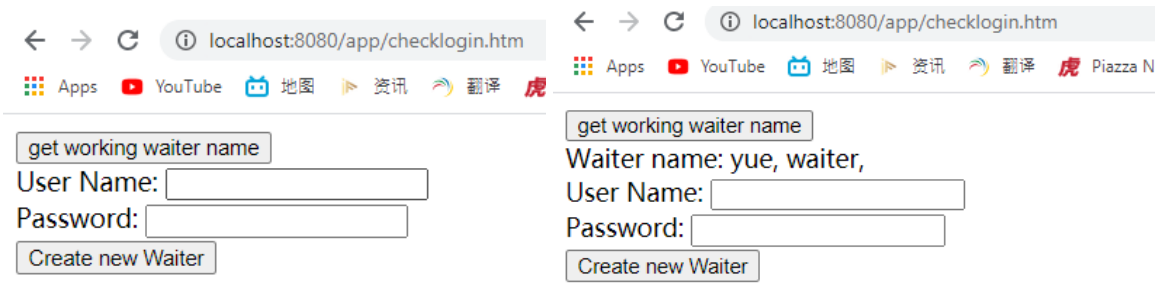
And waiter can see all bills in the manage bill page. When user check in and out, it will record the date, and the total price is calculate by the living days multiple daily price.



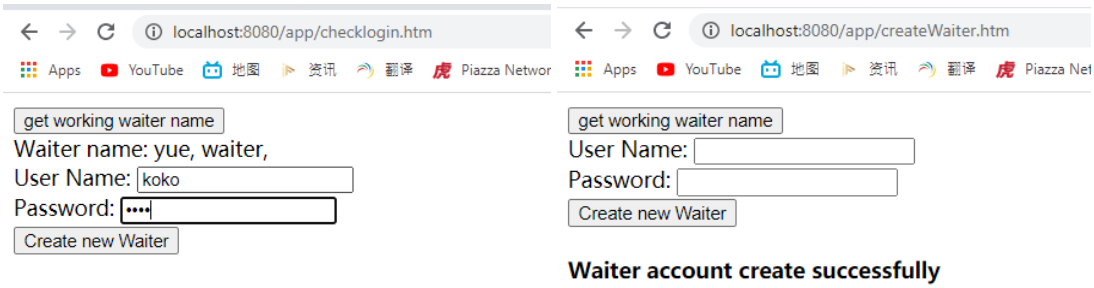
BillId	UserId	RoomId	Check in Date	Check out Date	Days	Price
1	1	0	2020-12-15	2020-12-15	0	0.0
2	1	0	2020-12-15	2020-12-15	0	0.0
3	1	205	2020-12-15	2020-12-15	0	0.0
4	2	103	2020-12-15	2020-12-15	0	0.0
5	1	102	2020-12-15	2020-12-15	0	0.0
6	4	101	2020-12-15	2020-12-15	1	100.0
7	3	305	2020-12-15	2020-12-15	1	150.0
8	4	301	2020-12-15	2020-12-15	1	150.0
9	5	102	2020-12-16	2020-12-16	1	100.0
10	1	104	2020-12-15	2020-12-15	1	100.0
11	-1	104	2020-12-15	2020-12-15	1	100.0
12	7	102	2020-12-16	2020-12-16	1	100.0
13	8	101	2020-12-18	2020-12-18	1	100.0

4. Admin

When you login as a admin, you see all the working state waiter.



You also can create new waiter account.



Controller:

```
@Controller
public class HomeController {

	private static final Logger logger = 
LoggerFactory.getLogger(HomeController.class);

	/**
	 * Simply selects the home view to render by returning its name.
	 */
	@RequestMapping(value = "/", method = RequestMethod.GET)
	public String home(Locale locale, Model model) {
		logger.info("Welcome home! The client locale is {}.", locale);

		Date date = new Date();
		DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, 
DateFormat.LONG, locale);

		String formattedDate = dateFormat.format(date);

		model.addAttribute("serverTime", formattedDate );

		return "home";
	}
	@RequestMapping(value = "/home.htm", method = RequestMethod.GET)
	public String gohome() {

		return "home";
	}
}


@Controller
public class AdminController {
	@Autowired
	LoginValidation lv;
	@RequestMapping(value = "/admin.htm", method = RequestMethod.GET)
	@ResponseBody
	public String handle(HttpServletRequest request, AdminDAO adminDao) {
		HttpSession session = request.getSession();
		List<Waiter> waiters = adminDao.getWaiters();
		if(waiters == null)return "no waiter is working now";
		String wname = "";
		for(Waiter w:waiters) {
```

```
                wname = wname+" "+w.getWname()+",";
            }

            return "Waiter name:"+wname;
      }

      @RequestMapping(value = "/createWaiter.htm", method = RequestMethod.POST)
      public ModelAndView createWaiter(HttpServletRequest request, AdminDAO
adminDao,WaiterDAO wDao, Waiter waiter) {
            if(request.getAttribute("unsafe-request") == "true" ){
           return new ModelAndView("login-error", "error", "unsafe input");
       }

            String wn = request.getParameter("wname");
            if(!lv.checkWaitername(wDao, wn))return new ModelAndView("admin-manage",
"success", "waitername exist");
            String wp = request.getParameter("wpassword");
            waiter.setWname(wn);
            waiter.setWpassword(wp);
            waiter.setState("false");
            adminDao.createWaiter(waiter);
            return new ModelAndView("admin-manage","success","Waiter account create
successfully");
      }
}

@Controller
public class LoginController {
      @Autowired
      LoginValidation lv;

      @RequestMapping(value = "/checklogin.htm", method = RequestMethod.POST)
      public ModelAndView handle(HttpServletRequest request, UserDAO userDao,
WaiterDAO wDao, AdminDAO adminDao) {
            if(request.getAttribute("unsafe-request") == "true" ){
           return new ModelAndView("login-error", "error", "unsafe input");
       }

            String un = request.getParameter("username");
            String up = request.getParameter("password");
            String role = request.getParameter("role");
            HttpSession session = request.getSession();


            if(role==null||un==null||up==null)return new ModelAndView("login-error",
"error", "role, username, password can not be empty");

            if(role.equals("user") ) {
                  User user = userDao.checkLogin(un, up);

                  if(user == null) {
                        System.out.print("user login error");

                        return new ModelAndView("login-error", "error", "username
or password is incorrect");
```

```
                        }
                        else {
                                session.setAttribute("user", user);
                        }
                        return new ModelAndView("book-room");
                }else if(role.equals("waiter")){
                        Waiter waiter = wDao.checkLogin(un, up);

                        if(waiter == null) {
                                System.out.print("waiter login error");
                                return new ModelAndView("login-error", "error", "username
or password is incorrect");
                        }
                        else {
                                session.setAttribute("waiter", waiter);
                                return new ModelAndView("manage-page");
                        }
                }else if(role.equals("admin")) {
                        Admin admin = adminDao.checkLogin(un, up);

                        if(admin == null) {
                                System.out.print("admin login error");
                                return new ModelAndView("login-error", "error", "username
or password is incorrect");
                        }
                        else {
                                session.setAttribute("admin", admin);
                                return new ModelAndView("admin-manage");
                        }
                }
                return new ModelAndView("login-error", "error", "other error happens");
        }

        @RequestMapping(value = "/signin.htm", method = RequestMethod.POST)
        public String signIn() {
                return "sign-in";
        }

        @RequestMapping(value = "/createNew.htm", method = RequestMethod.POST)
        public ModelAndView createUser(HttpServletRequest request, UserDAO userDao,
User user) {
                if(request.getAttribute("unsafe-request") == "true" ){
               return new ModelAndView("login-error", "error", "unsafe input");
          }

                String un = request.getParameter("username");
                if(!lv.checkusername(userDao, un))return new ModelAndView("login-error",
"error", "username exist");
                String up = request.getParameter("password");
                user.setUsername(un);
                user.setUserpassword(up);
                userDao.createUser(user);
                return new ModelAndView("home");
        }
}
```

```
@Controller
public class UserController {

	@Autowired
	DateValidation dv;
	@RequestMapping(value = "/bookRoom.htm")
	public String bookRoom() {
		return "book-room";
	}
	@RequestMapping(value = "/viewOrder.htm", method = RequestMethod.GET)
	public ModelAndView viewOrder(HttpServletRequest request, UserDAO userDao) {
		HttpSession session = request.getSession();
		User user = (User) session.getAttribute("user");
		int userid = user.getUserid();
		ModelAndView mv = new ModelAndView("view-order", "orders",
userDao.getOrders(userid));
		mv.addObject("rooms", userDao.getRooms(userid));

		return mv;
	}
	@RequestMapping(value = "/viewBill.htm")
	public ModelAndView viewBill(HttpServletRequest request, UserDAO userDao) {
		HttpSession session = request.getSession();
		User user = (User) session.getAttribute("user");
		List<Bill> bills = userDao.getBills(user.getUserid());
		return new ModelAndView("view-bill", "bills", bills);
	}

	@RequestMapping(value = "/orderRoom.htm", method = RequestMethod.POST)
	public ModelAndView orderRoom(UserOrder order,HttpServletRequest request,
UserDAO userDao) {
		if(request.getAttribute("unsafe-request") == "true" ){
		return new ModelAndView("error-page", "error", "unsafe input");
	  }
		HttpSession session = request.getSession();
		User user = (User) session.getAttribute("user");
		String roomType = request.getParameter("roomType");

		String startDate =request.getParameter("startDate");
		String endDate = request.getParameter("endDate");
		if(!dv.isBefore(startDate, endDate)) return new ModelAndView("error-
page", "error", "end date can not before start Date");

		order.setRoomType(roomType);
		order.setStartDate(startDate);
		order.setEndDate(endDate);
		order.setState("false");
		userDao.bookRoom(order,user.getUserid());


		session.setAttribute("user", user);
		return new ModelAndView("book-room", "success", "order success");
	}
	@RequestMapping(value = "/updateOrder.htm", method = RequestMethod.GET)
```

```
    public ModelAndView updateTodoGet(@RequestParam("id") Integer userId, UserDAO
userDao) {
        ModelAndView mv;

        UserOrder order = userDao.getOrder(userId);

        if(order != null) {
            mv = new ModelAndView("change-order", "existingorder", order);
        } else mv =  new ModelAndView("error-page","error", "can not change this
order");

        return mv;
    }

      @RequestMapping(value = "/updateOrder.htm", method = RequestMethod.POST)
    public ModelAndView updateTodoPost(@ModelAttribute("existingorder") UserOrder
order, UserDAO userDao, HttpServletRequest request) {
             if(request.getAttribute("unsafe-request") == "true" ){
            return new ModelAndView("error-page", "error", "unsafe input");
        }

             if(!dv.isBefore(order.getStartDate(), order.getEndDate())) return new
ModelAndView("error-page", "error", "end date can not before start Date");
        ModelAndView mv;
        HttpSession session = request.getSession();
             User user = (User) session.getAttribute("user");
             user.deleteOrder(order.getId());
             user.addOrder(order);
             session.setAttribute("user", user);
        int result = userDao.updateOrder(order);

        if(result == 1) {
            mv = new ModelAndView("redirect:/viewOrder.htm");

        } else mv =  new ModelAndView("error-page","error", "can not change this
order");

        return mv;
    }
      @RequestMapping(value = "/removeOrder.htm", method = RequestMethod.GET)
    public ModelAndView removeTodo(@RequestParam("id") int orderId, UserDAO userDao,
HttpServletRequest request) {
        ModelAndView mv;
        HttpSession session = request.getSession();
             User user = (User) session.getAttribute("user");
             user.deleteOrder(orderId);
        int result = userDao.deleteOrder(orderId,user.getUserid());

        if(result == 1) {
            mv = new ModelAndView("redirect:/viewOrder.htm");

        } else mv =  new ModelAndView("error-page", "error", "can not change this
order");

        return mv;
```

```
    }

	@RequestMapping(value = "/userCheckout.htm", method = RequestMethod.GET)
	public ModelAndView checkoutRoom(@RequestParam("id") Integer roomid,
HttpServletRequest request, UserDAO userDao) {
		int res = userDao.checkout(roomid);
		ModelAndView mv;

		if(res == 1) {
		mv = new ModelAndView("redirect:/viewOrder.htm", "success", "check out
success");
		 } else mv =  new ModelAndView("error-page", "error", "can not change this
order");

		return mv;
	}
}

@Controller
public class WaiterController {
	@RequestMapping(value = "/goManage.htm", method = RequestMethod.GET)
	public String goToManagePage() {

		return "manage-page";
	}
	@RequestMapping(value = "/viewUser.htm", method = RequestMethod.GET)
	public ModelAndView viewUsers(WaiterDAO wDao) {
		return new ModelAndView("view-user", "users", wDao.getUsers());

	}
	@RequestMapping(value = "/working.htm", method = RequestMethod.GET)
	public ModelAndView working(HttpServletRequest request, WaiterDAO wDao) {
		HttpSession session = request.getSession();
		Waiter waiter = (Waiter) session.getAttribute("waiter");
		wDao.changeState(waiter);
		if(waiter.getState()!=null&&waiter.getState().equals("true")) {
			waiter.setState("false");
		}else {
			waiter.setState("true");
		}
		session.setAttribute("waiter", waiter);
		return new ModelAndView("manage-page","success","working state change");

	}
	@RequestMapping(value = "/userDetail.htm", method = RequestMethod.GET)
	public ModelAndView userDtail(@RequestParam("id") int userId, WaiterDAO wDao,
HttpServletRequest request) {
		HttpSession session = request.getSession();
		session.setAttribute("userid", userId);
		session.setAttribute("username", wDao.getUsername(userId));
		ModelAndView mv = new ModelAndView("user-detail", "orders",
wDao.getUserorders(userId));
		mv.addObject("userRooms", wDao.getRoom(userId));

		return mv;
```

```
	}

	@RequestMapping(value = "/userCheckin.htm", method = RequestMethod.GET)
	public ModelAndView userCheckin(@RequestParam("id") Integer orderid, WaiterDAO
wDao, HttpServletRequest request) {
		HttpSession session = request.getSession();
		int userId = (Integer) session.getAttribute("userid");
		if(wDao.checkorder(orderid)==0)return new ModelAndView("view-user",
"success","this order has been checked");
		ModelAndView mv = new ModelAndView("user-detail", "orders",
wDao.getUserorders(userId));
		mv.addObject("userRooms", wDao.getRoom(userId));
		mv.addObject("singleRooms",wDao.getRooms("singleRoom"));
		mv.addObject("standardRooms",wDao.getRooms("standardRoom"));
		mv.addObject("bigRooms",wDao.getRooms("bigRoom"));

		return mv;
	}
	@RequestMapping(value = "/setRoom.htm", method = RequestMethod.GET)
	public ModelAndView setRoom(@RequestParam("id") Integer roomid,
HttpServletRequest request, WaiterDAO wDao) {
		HttpSession session = request.getSession();
		int userid = (Integer) session.getAttribute("userid");
		int res = wDao.checkin(roomid, userid);
		ModelAndView mv = new ModelAndView("view-user","success", "room check in
success");
		mv.addObject("users", wDao.getUsers());
		return mv;
	}
	@RequestMapping(value = "/checkoutRoom.htm", method = RequestMethod.GET)
	public ModelAndView checkoutRoom(@RequestParam("id") Integer roomid,
HttpServletRequest request, WaiterDAO wDao) {
		HttpSession session = request.getSession();
		int userid = (Integer) session.getAttribute("userid");
		int res = wDao.checkout(roomid);
		ModelAndView mv = new ModelAndView("view-user","success", "room check
out success");
		mv.addObject("users", wDao.getUsers());
		return mv;
	}
	@RequestMapping(value = "/preRoom.htm", method = RequestMethod.GET)
	public ModelAndView preRoom(HttpServletRequest request, WaiterDAO wDao) {


		ModelAndView mv = new ModelAndView("pre-room","finishRooms",
wDao.getFinish());

		return mv;
	}
	@RequestMapping(value = "/sendBill.htm", method = RequestMethod.GET)
	public ModelAndView sendBill(@RequestParam("id") Integer roomid,
HttpServletRequest request, WaiterDAO wDao, Bill bill) {

		int res = wDao.preRoom(roomid,bill);
```

```
            ModelAndView mv = new ModelAndView("manage-page", "success","bill send
success");
            return mv;
      }
      @RequestMapping(value = "/manageBill.htm", method = RequestMethod.GET)
      public ModelAndView viewBill(HttpServletRequest request, WaiterDAO wDao) {


            ModelAndView mv = new ModelAndView("manage-bill",
"bills",wDao.getBills());
            return mv;
      }
}
```