**数据库课程设计**

**实验报告**

**选 题: 题目17 宾馆管理信息系统**

**姓 名: 周欣宇**

**学 号: 171310530**

**班 级: 卓越软件1702**

**日 期: 2020.4.23**

目录

[一、题目要求 3](#_Toc38589788)

[二、工具环境 3](#_Toc38589789)

[（1）实验环境： 3](#_Toc38589790)

[（2）开发软件： 3](#_Toc38589791)

[三、承担任务 4](#_Toc38589792)

[（1）需求分析 4](#_Toc38589793)

[（2）参与数据库设计 4](#_Toc38589794)

[（3）学习前端Layui和其他知识 4](#_Toc38589795)

[四、数据库设计 5](#_Toc38589796)

[（1）房型对应的押金表deposit 5](#_Toc38589797)

[（2）房间管理表room 6](#_Toc38589798)

[（3）入住事件表checkin 6](#_Toc38589799)

[（4）当前入住表now 6](#_Toc38589800)

[（5）预约信息表reserve 7](#_Toc38589801)

[（6）贵宾管理表vip 7](#_Toc38589802)

[（7）欠款管理表permanent 7](#_Toc38589803)

[（8）统计报表表overall 8](#_Toc38589804)

[五、个人部分 8](#_Toc38589805)

[（1）界面展示 8](#_Toc38589806)

[1、客户入住界面 8](#_Toc38589807)

[2、房间预约界面 9](#_Toc38589808)

[3、贵宾注册界面 9](#_Toc38589809)

[（2）入住业务及代码 9](#_Toc38589810)

[1、前端布局 10](#_Toc38589811)

[2、前端监听 13](#_Toc38589812)

[3、输入格式验证 18](#_Toc38589813)

[4、数据交换及处理 20](#_Toc38589814)

[（3）预约业务及代码 23](#_Toc38589815)

[1、前端布局 23](#_Toc38589816)

[2、前端监听 23](#_Toc38589817)

[3、输入格式验证 23](#_Toc38589818)

[4、数据交换及处理 25](#_Toc38589819)

[（2）贵宾注册业务及代码 27](#_Toc38589820)

[1、前端布局 27](#_Toc38589821)

[2、前端监听 28](#_Toc38589822)

[3、输入格式验证 29](#_Toc38589823)

[4、数据交换及处理 30](#_Toc38589824)

[六、遇到的问题及心得 31](#_Toc38589825)

[（1）关于debug 31](#_Toc38589826)

[（2）java和MySql时间类型匹配 31](#_Toc38589827)

[（3）后期整合 32](#_Toc38589828)

# 一、题目要求

题目17 宾馆管理信息系统

一套功能强大而又使用简便实用的客房管理软件，适用于宾馆管理、酒店管理、招待所管理。主要功能：团体,散客入住客人的信息登记，换房，正常退房，挂帐退房，挂单补款结帐，预定管理，贵宾卡管理，押金管理，各种强大的宾馆统计报表，异常处理。

# 二、工具环境

## （1）实验环境：

Windows Version: Windows10；

Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz；

操作系统: 64位；

## （2）开发软件：

开发平台：IntelliJ IDEA 2019.3.3 x64；

数据库：MySQL 8.0；

JDK：12.0；

Tomcat：9.0.33；

数据库可视化工具：Navicat

# 三、承担任务

## （1）需求分析

分析酒店管理的业务逻辑，询问酒店前台人员具体的实际操作流程。扩展和具体化项目所需要实现的功能需求。根据分析结果，大致确定前端界面包括入住、预约、退房、换房、查看报表，并且确定各个前端界面中需要实现的功能。

## （2）参与数据库设计

提供数据库设计的建议，包括建议新增预约信息Reserve表，建议Checkin的时间属性类型为datetime，但统计报表中时间属性类型为date等。并在数据库设计前期指出需要明确表中外键指向的问题。

## （3）学习前端Layui和其他知识

在队友确定前端开发框架使用Layui后，大量学习Layui的组件使用，学习Layui的官方文档以及参考Layui前端框架官方社区Fly上的问答。具体组件包括栅格/后台布局、form表单、导航栏、Tabs选项卡、折叠面板、table数据表格、timeline时间线、弹出层layer等。同时学习了Layui控件的重要的属性使用方法，其中包括属性事件过滤器lay-filter作用、表单中lay-verify用法等。在队友使用Layui出现问题时提供帮助。同时在开发过程中深入学习了jquery，js，ajax，servlet知识。

**（5）互帮互助解决问题**

帮忙解决了一些需要正则表达式匹配的问题；帮忙完成从Checkin表中找到某房间入住人的问题（主要是使用复杂sql语句）；解决了数据库中date和datetime与java中时间类型不匹配问题；提供能够复用的jsp、servlet、数据库操作的代码实现房间筛选和显示。

**（6）个人开发**

入住：Checkin.jsp入住界面；可入住房间的servlet：AvailableRoomServlet； 执行入住的servlet：CheckInServlet；查看预约信息的servlet：ResInfServlet；入住界面操作数据库的Checkin.java(db文件夹下)；

预约：Reserve.jasp预约界面；可以预约房间的servlet：ReserveRoomServlet；执行预约的servlet：ReserveServlet；预约界面操作数据库的Reserve.java(db文件夹下)；

贵宾注册：guest.jsp贵宾注册界面；贵宾注册的servlet：GuestServlet；贵宾注册界面操作数据库的GuestRegister.java(db文件夹下)

# 四、数据库设计

## （1）房型对应的押金表deposit

设计了四种房型，单人间、双人间、三人间、豪华套房，不同房型对应不同的押金。房型RoomType主键，需要交的押金depositprice。前端不对该表进行操作。

+------------------- +-------------- +-------------- +---------+--------------+-------- +

| Field | Type | Null | Key | Default | Extra |

+------------------- +-------------- +-------------- +---------+--------------+-------- +

| RoomType | varchar(20) | NO | PRI | NULL | |

| depositprice | int | YES | | NULL | |

+------------------- +-------------- +-------------- +-------- +-------------- +---------+

## （2）房间管理表room

房间管理表管理房间，含有房间的信息：房号RoomNumber（主键），房型RoomType（deposit外键），RoomProice价格，Telephone电话，Status状态（空闲、预约、有人）。

+------------------- +-------------- +-------- +-------- +-------------- +------------------- +

| Field | Type | Null | Key | Default | Extra |

+------------------- +-------------- +-------- +-------- +-------------- +------------------- +

| RoomNumber | int | NO | PRI | NULL | auto\_increment |

| RoomType | varchar(20) | YES | MUL | NULL | |

| Telephone | varchar(20) | YES | | NULL | |

| RoomPrice | int | YES | | NULL | |

| Status | varchar(20) | YES | | NULL | |

+------------------- +-------------- +-------- +-------- +-------------- +------------------- +

## （3）入住事件表checkin

每当有客户入住，加一条入住记录。记录客户入住时间入住房间

+------------------- +-------------- +--------+--------- +-------------- +------------------- +

| Field | Type | Null | Key | Default | Extra |

+------------------- +-------------- +-------- +-------- +-------------- +------------------- +

| EventNumber | int | NO | PRI | NULL | auto\_increment |

| IdentityNumber | varchar(100)| YES | MUL | NULL | |

| RoomLiving | int | YES | MUL | NULL | |

| YearMonthDay | datetime | YES | | NULL | |

| Gorl | varchar(10) | YES | | NULL | |

+------------------- +-------------- -+------- +-------- +-------------- +------------------- +

## （4）当前入住表now

当前入住表管理当前入住的客户信息。每当有客户入住，加一条入住记录。退房时，删除对应记房间的录

+------------------- +-------------- +-------- +-------- +---------+-------- +

| Field | Type | Null | Key | Default | Extra |

+------------------- +-------------- +-------- +-------- +---------+--------+

| IdentityNumber | varchar(100)| NO | PRI | NULL | |

| RoomLiving | int | YES | MUL | NULL | |

+------------------- +--------------+---------+-------- +---------+-------- +

## （5）预约信息表reserve

有客户预约时，加入一条记录。记录客户信息和预约的房间号，入住时删除对应记录。

+------------------- +---------------+--------+--------+----------+------------------+

| Field | Type | Null | Key | Default | Extra |

+------------------- +---------------+--------+--------+---------+-------------------+

| RoomNumber | int | NO | PRI | NULL | auto\_increment |

| IdentityNumber | varchar(100) | YES | | NULL | |

| customername | varchar(100) | YES | | NULL | |

+------------------- +-------------- -+-------+---------+---------+-------------------+

## （6）贵宾管理表vip

有客注册贵宾时，生成贵宾号，增加一条记录，保存贵宾信息和贵宾号

+------------------- +--------------- +------- +--------+--------- +--------+

| Field | Type | Null | Key | Default | Extra |

+------------------- +-------------- -+------- +-------- +---------+-------- +

| VipNumber | varchar(100) | NO | PRI | NULL | |

| IdentityNumber | varchar(100) | YES | MUL | NULL | |

+------------------- +--------------+-------- +-------- +---------+-------- +

## （7）欠款管理表permanent

结账时赊账退房、还款时修改记录

+------------------- +--------------+-------- +--------+---------+--------- +

| Field | Type | Null | Key | Default | Extra |

+------------------- +-------------- -+------- +--------+---------+--------- +

| customername | varchar(100) | YES | | NULL | |

| IdentityNumber | varchar(100) | NO | PRI | NULL | |

| VipStatus | varchar(11) | YES | | NULL | |

| OwnMoney | int | YES | | NULL | |

+------------------- +-------------- -+------- +--------+---------+--------- +

## （8）统计报表表overall

记录每天房间预约数、入住人数、盈利、赊账及还款数、贵宾增加数

+------------------- +-------------- -+--------+--------+---------+--------- +

| Field | Type | Null | Key | Default | Extra |

+------------------- +-------------- -+--------+--------+---------+---------+

| customername | varchar(100) | YES | | NULL | |

| IdentityNumber | varchar(100) | NO | PRI | NULL | |

| VipStatus | varchar(11) | YES | | NULL | |

| OwnMoney | int | YES | | NULL | |

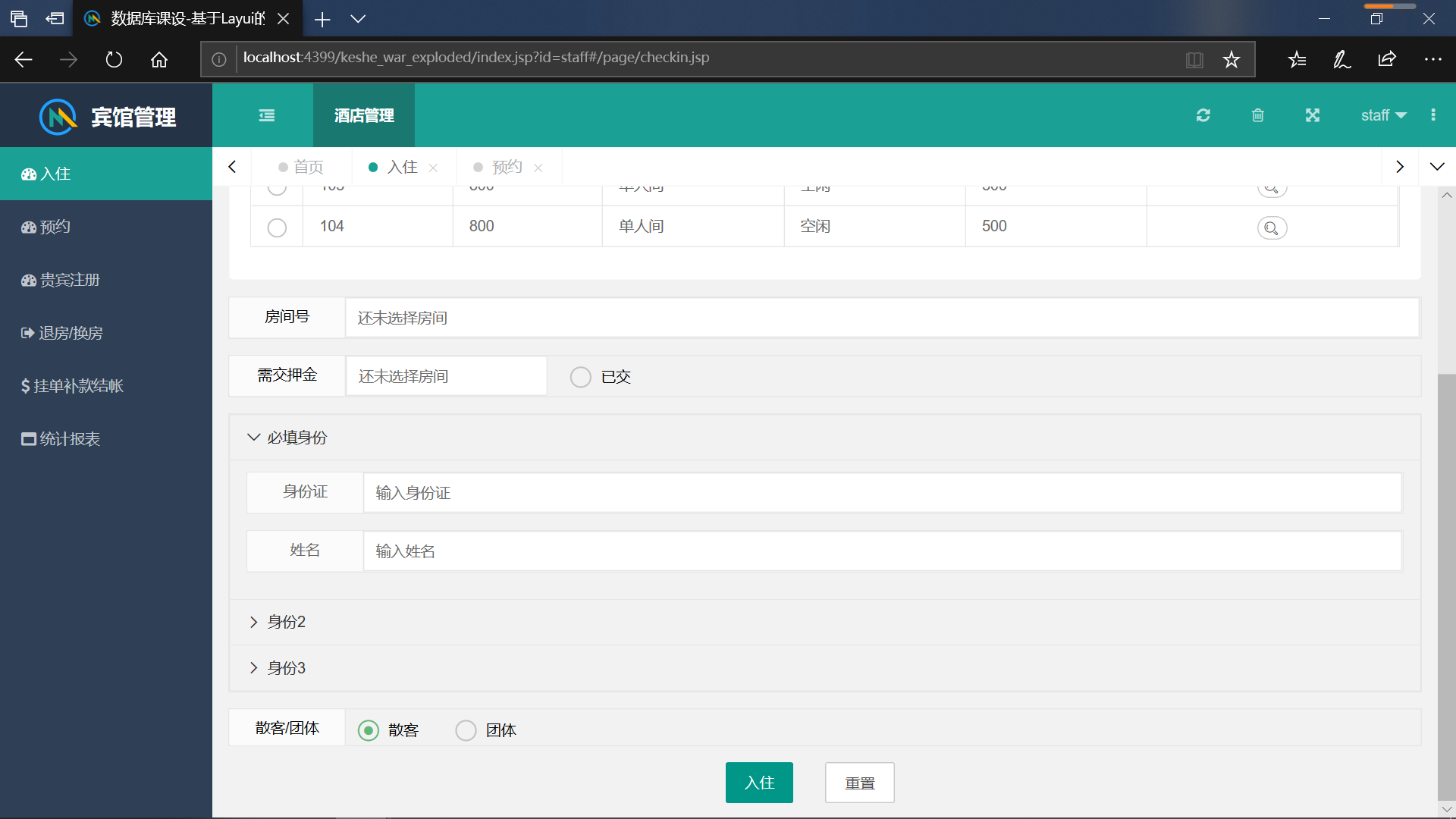
+------------------- +---------------+-------+---------+---------+--------- +

# 五、个人部分

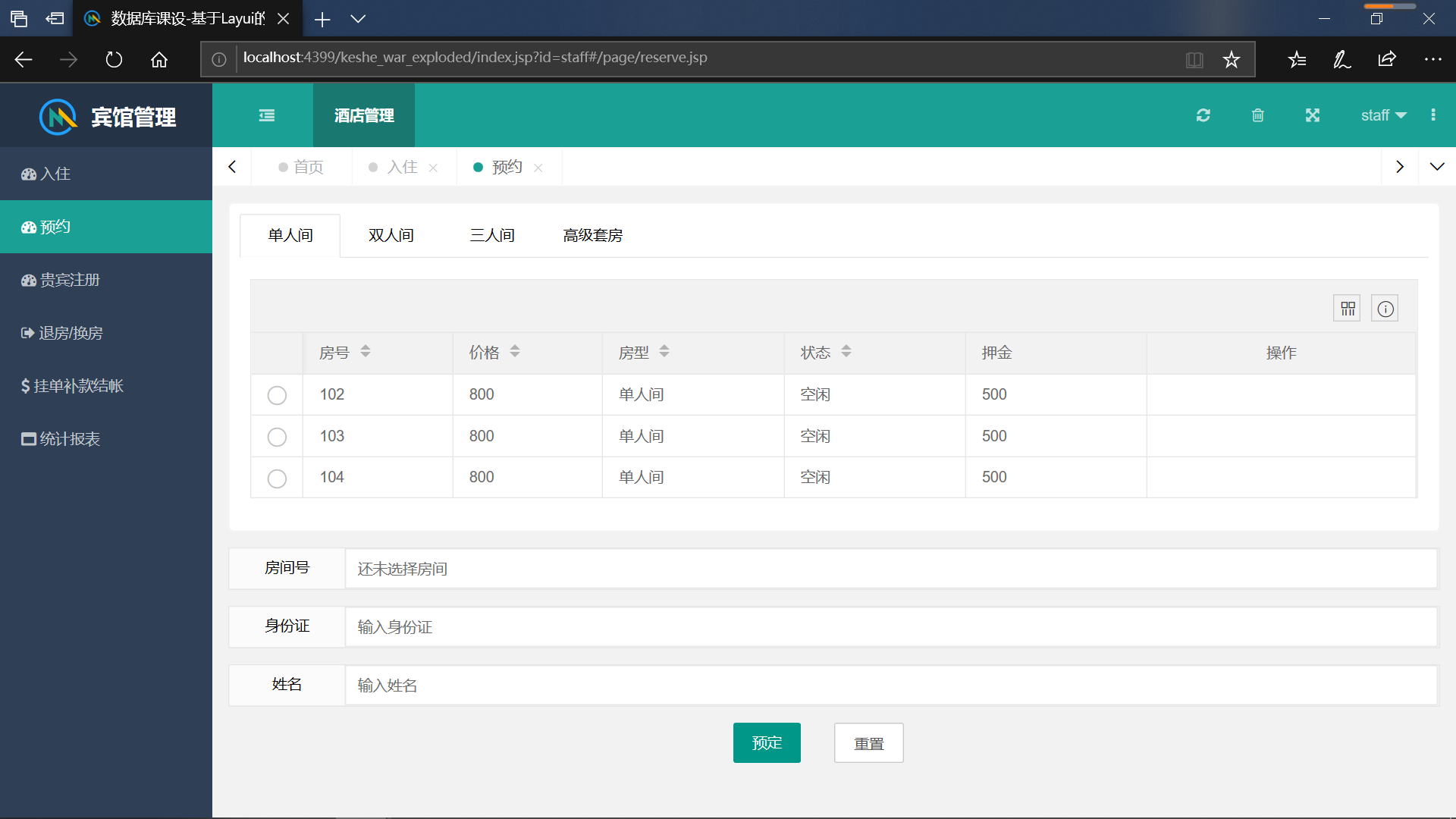
个人负责房间预约、房间入住、贵宾注册业务开发。本次课程设计前端是jsp文件。前端界面主要按照Layui框架，结合使用javascript、jquery等完成开发个人负责房间预约、房间入住、贵宾注册业务开发。

## （1）界面展示

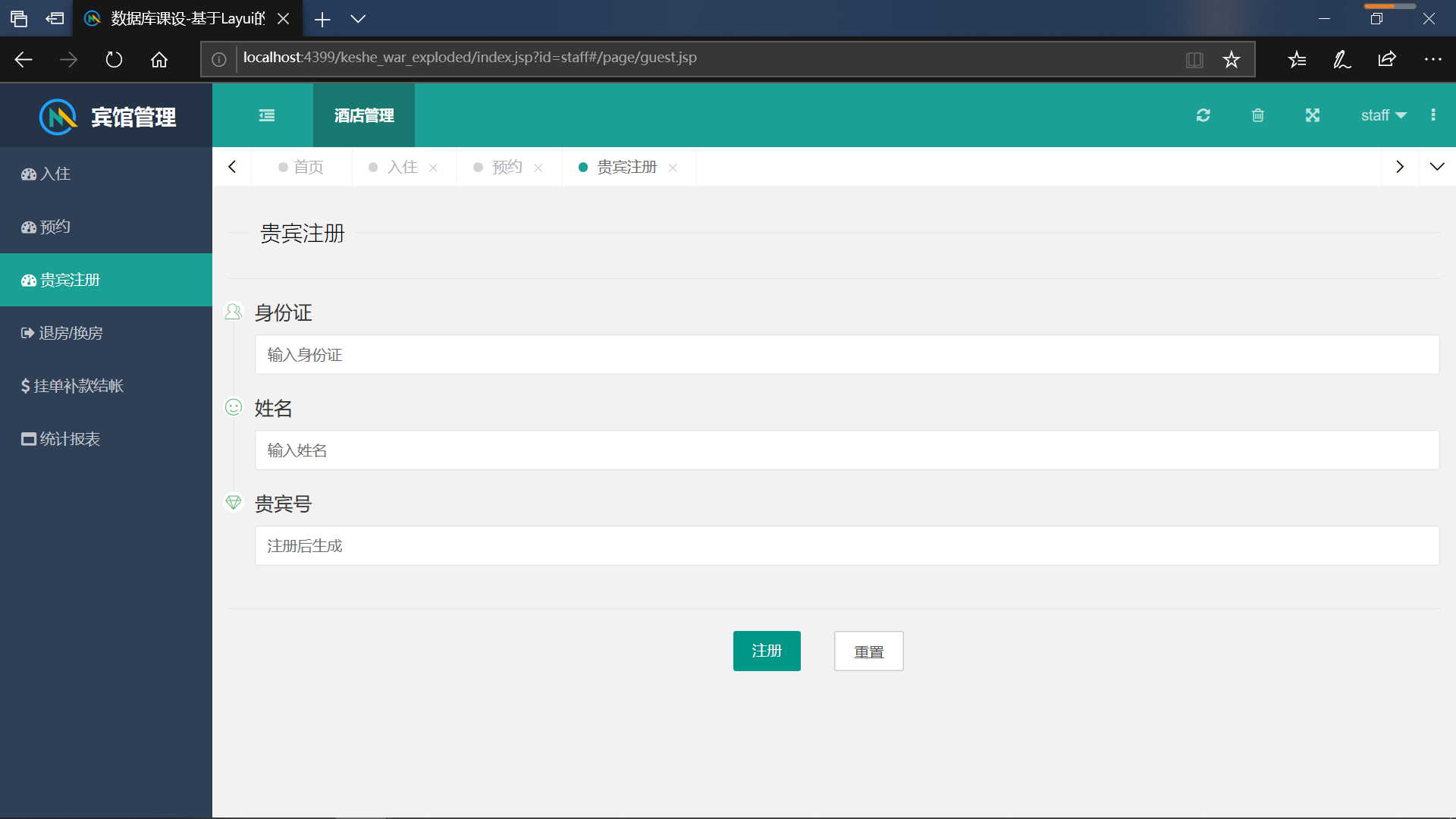
### 1、客户入住界面



### 2、房间预约界面



### 3、贵宾注册界面



## （2）入住业务及代码

在入住界面，需要能够选择房型，筛选出该房型空闲或被预约的房间。并且能够查看预约人的信息。选中房间后，填写入住人的身份信息，填写身份证，姓名。因为有单人房、双人房等房型，所以设置了三组身份证和姓名填写框。

并且在提交入入住时，需要检查信息是否漏填、错填并提示。入住时一定要选择房间，一定要确认收取押金后选择押金已交按钮。身份信息第一组是必填的，后面两个是选填的。同时身份证的输入要检验是否符合身份证要求。

### 1、前端布局

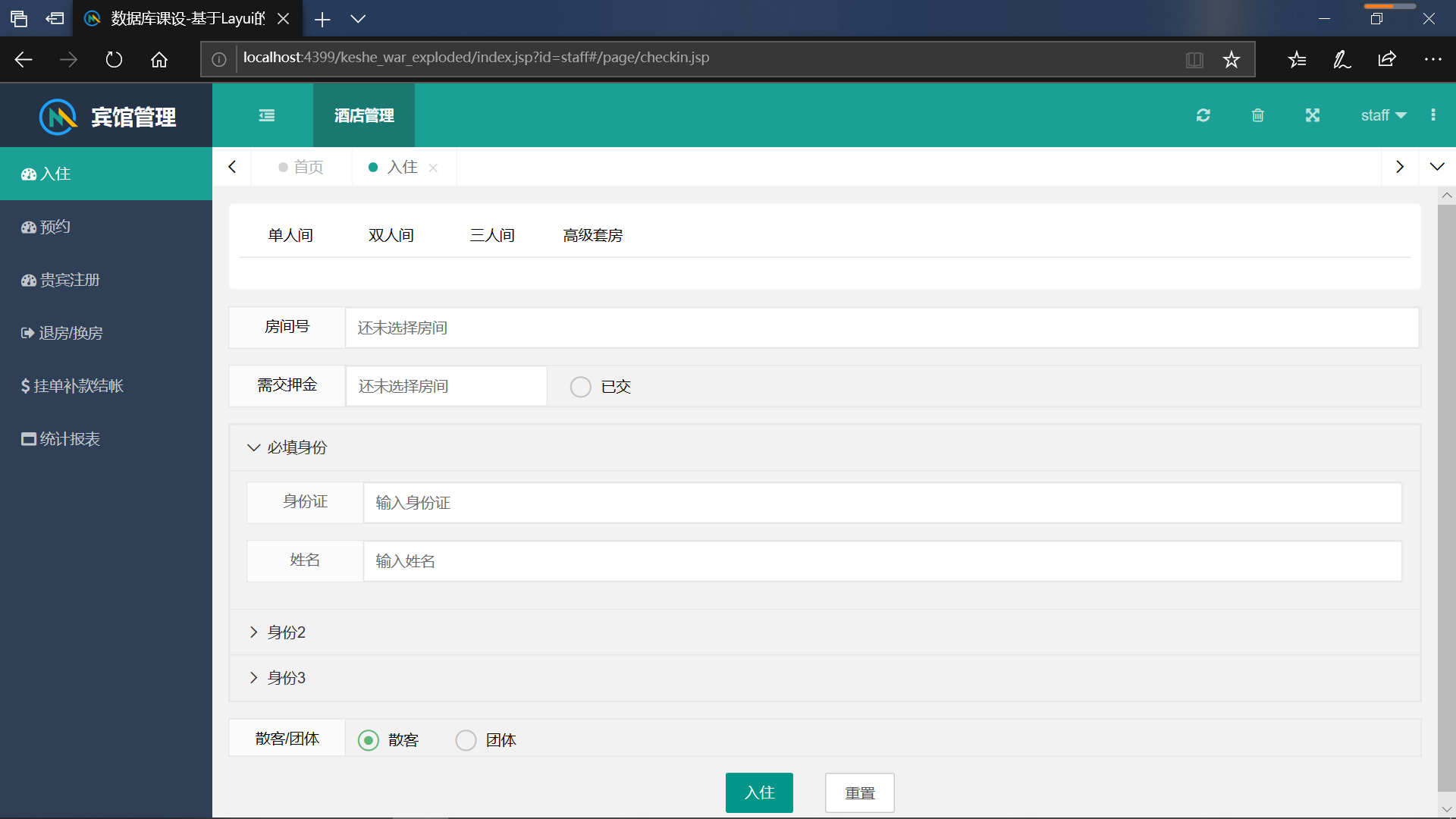
使用layui控件，整体是一个form表单。表单中放入文本框等。

<form class="layui-form layui-form-pane" action="">

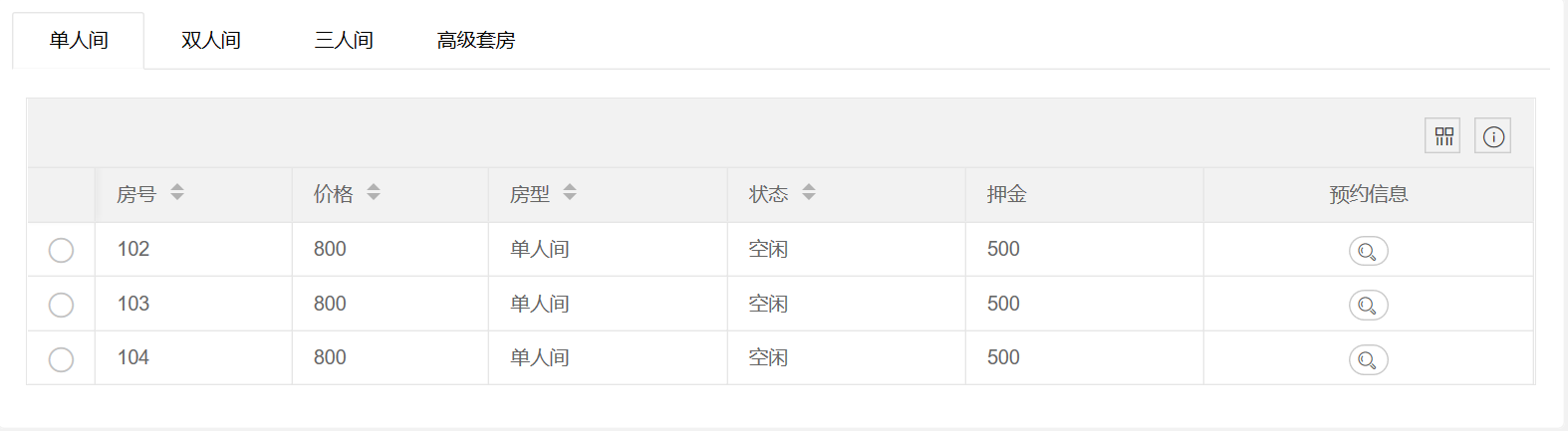
……

…….

<\form>



form表单中第一个表单元素是一个卡片，卡片用于选择想要入住的房间类型。刚进入到界面中时，卡片没有选中项，当卡片选中某项时，就能展示对应卡片的内容，卡片对应的内容是table。例如下图选中单人间



<div class="layui-form-item">

<div class="layuimini-container"><div class="layuimini-main">

<div class="layui-tab" lay-filter="filtertab">

<ul class="layui-tab-title">

<li>单人间</li>

<li>双人间</li>

<li>三人间</li>

<li>高级套房</li>

</ul>

<div class="layui-tab-content">

<div class="layui-tab-item">

<table class="layui-hide" id="SingleRoom" lay-filter="SingleRoom"></table>

</div>

<div class="layui-tab-item">

<table class="layui-hide" id="DoubleRoom" lay-filter="DoubleRoom"></table>

</div>

<div class="layui-tab-item">

<table class="layui-hide" id="TripleRoom" lay-filter="TripleRoom"></table>

</div>

<div class="layui-tab-item">

<table class="layui-hide" id="SuiteRoom" lay-filter="SuitRoom"></table>

</div>

</div>

</div>

</div></div>

</div>

点击卡片，然后切换到对应内容是script中的elem.on中实现，因为卡片切换同时需要影响到之后所讲的文本框和按钮，所以之后再讲该部分script代码。

第二个表单元素是一个不可编辑的房间号文本框，第三个表单元素是一个需交押金数短文本框和一个单选按钮。选中按钮表示客户已交押金。

<div class="layui-form-item" pane>

<label class="layui-form-label">房间号</label>

<div class="layui-input-block">

<input id="roomliving" …> //房间号文本框

</div>

</div>

<div class="layui-form-item" pane>

<label class="layui-form-label">需交押金</label>

<div class="layui-input-inline">

<input id="deposit"…..> //押金文本框

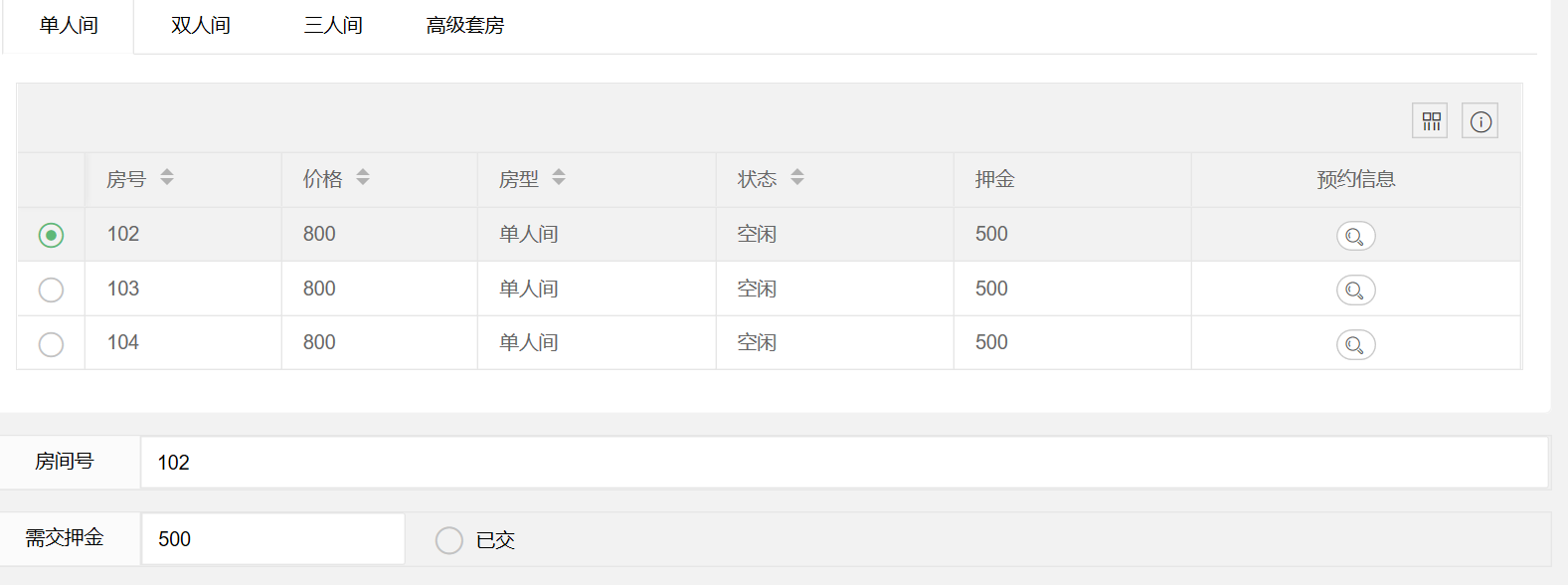
</div>

<input type="radio" …> //是否加押金的单选按钮

</div>

</div>

这两个文本框会在表格中选中房间后自动填写，但不可人工修改。



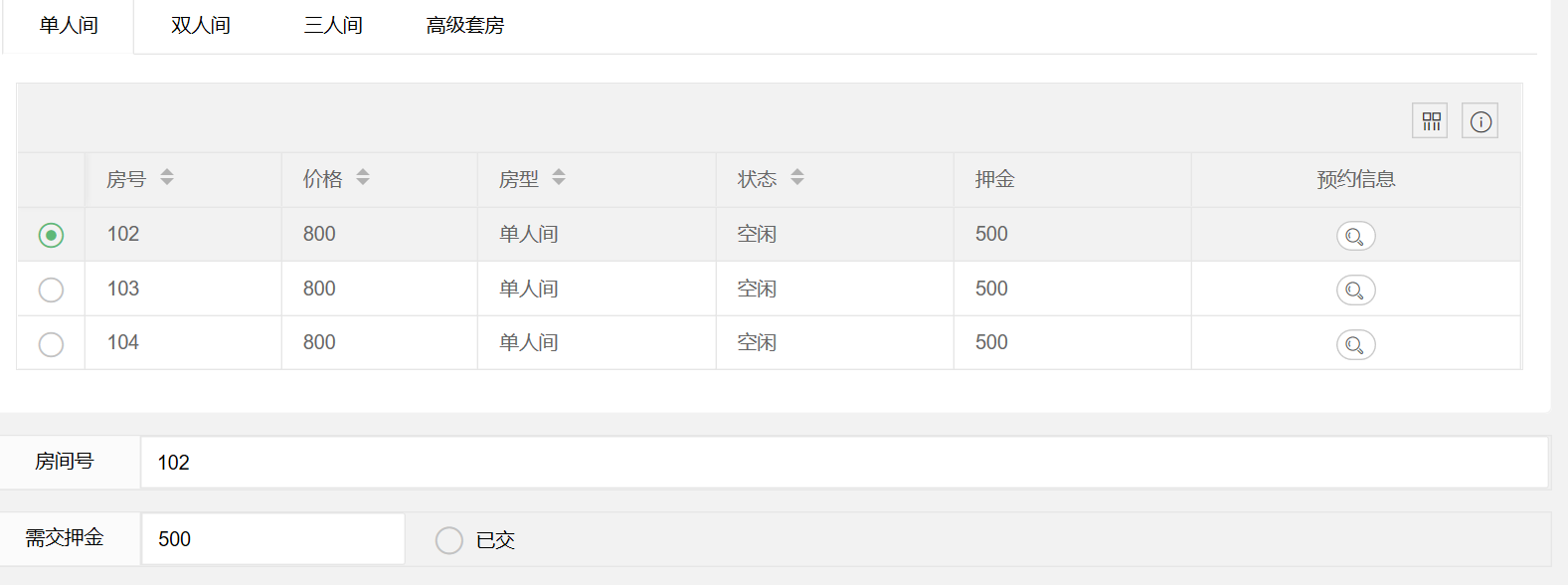
之后的表单元素是三个折叠框，第一个必填身份折叠框默认打开，后面两个选填身份的折叠框默认关闭。折叠框中又含有输入身份证和输入姓名的两个文本框。



再接下来的表单元素是一个复选按钮，选择是散客还是团体，默认选中散客。最后的表单元素是两个按钮，一个入住按钮，一个重置按钮。

### 2、前端监听

#### 2.1选中房间单选框监听



选中房间后，将选中行的房间号和需要交押金号自动填入房价号文本框和需

交押金文本框内。以单人间表格中的按钮为例：

table.on('radio(SingleRoom)', function(obj){

$("#roomliving").val(obj.data.roomNumber);

$("#deposit").val(obj.data.depositprice);

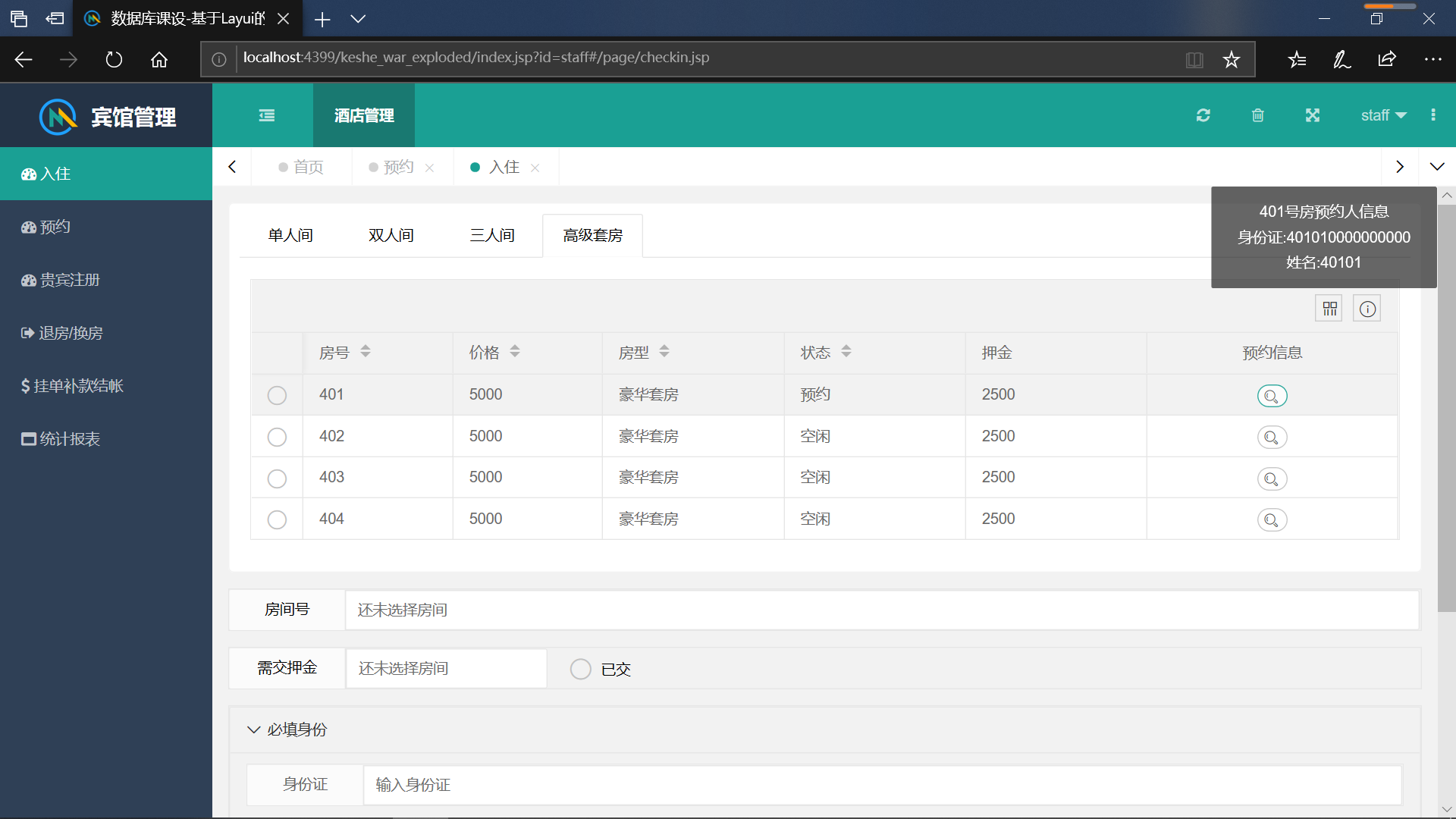
});

在script中，通过Layui的table接口，使用jquery，使用文本框的id修改指定值。该layui提供的接口需要给监听的table设定一个事件过滤器lay-filter，我将单人间的表格的事件过滤器设置为SingleRoom，所以通过('radio(SingleRoom)'就能监听到该表格的单选按钮的事件。

#### 2.2预约信息查看按钮监听

在表格中，最后一栏处又一个查看预约信息按钮的功能。

点击预约状态房间那一行的按钮，能够显示预约人的信息，点击空闲状态房间的会提示该房间未被预约，例如点击已被预约的高级套房401，和点击未被预约的高级套房402。



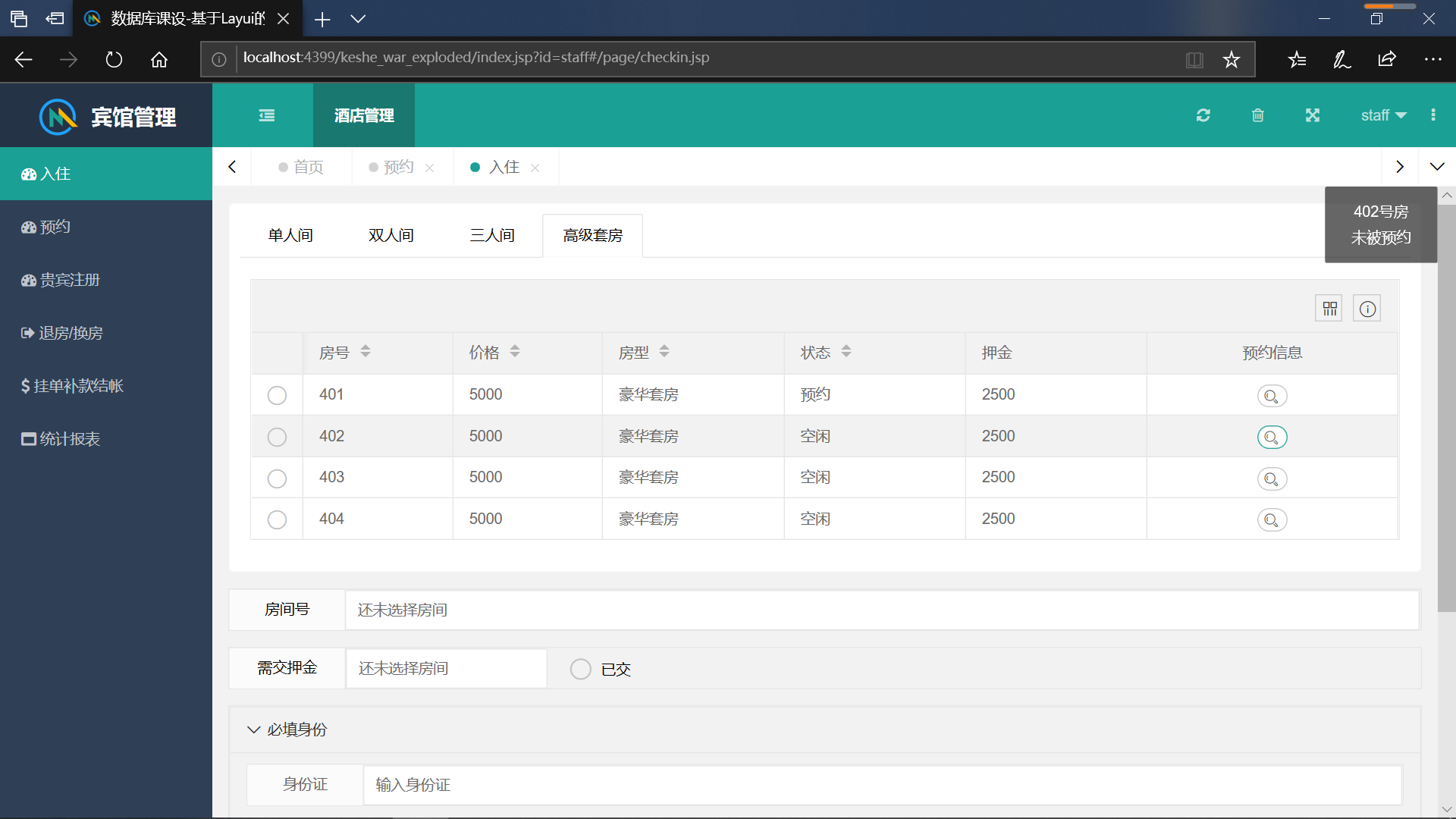


table.on('tool(SuitRoom)', function(obj){

$.ajax({

………

………

});

});

通过Layui的接口，给高级套房表设置事件过滤器SuitRoom，'tool(SuitRoom)'就能监听到该工具栏的事件。其中通过ajax向后端传递数据并返回数据，成功时，根据返回返回信息，使用Layui的消息框，输出预约信息。

当查看的房间是被预约状态时，输出预约人的身份证和姓名：

layer.msg(

obj.data.roomNumber

+'号房预约人信息<br>身份证:'

+result.id

+'<br>姓名:'

+result.name, {

time: 20000,

offset: 'rt'

});

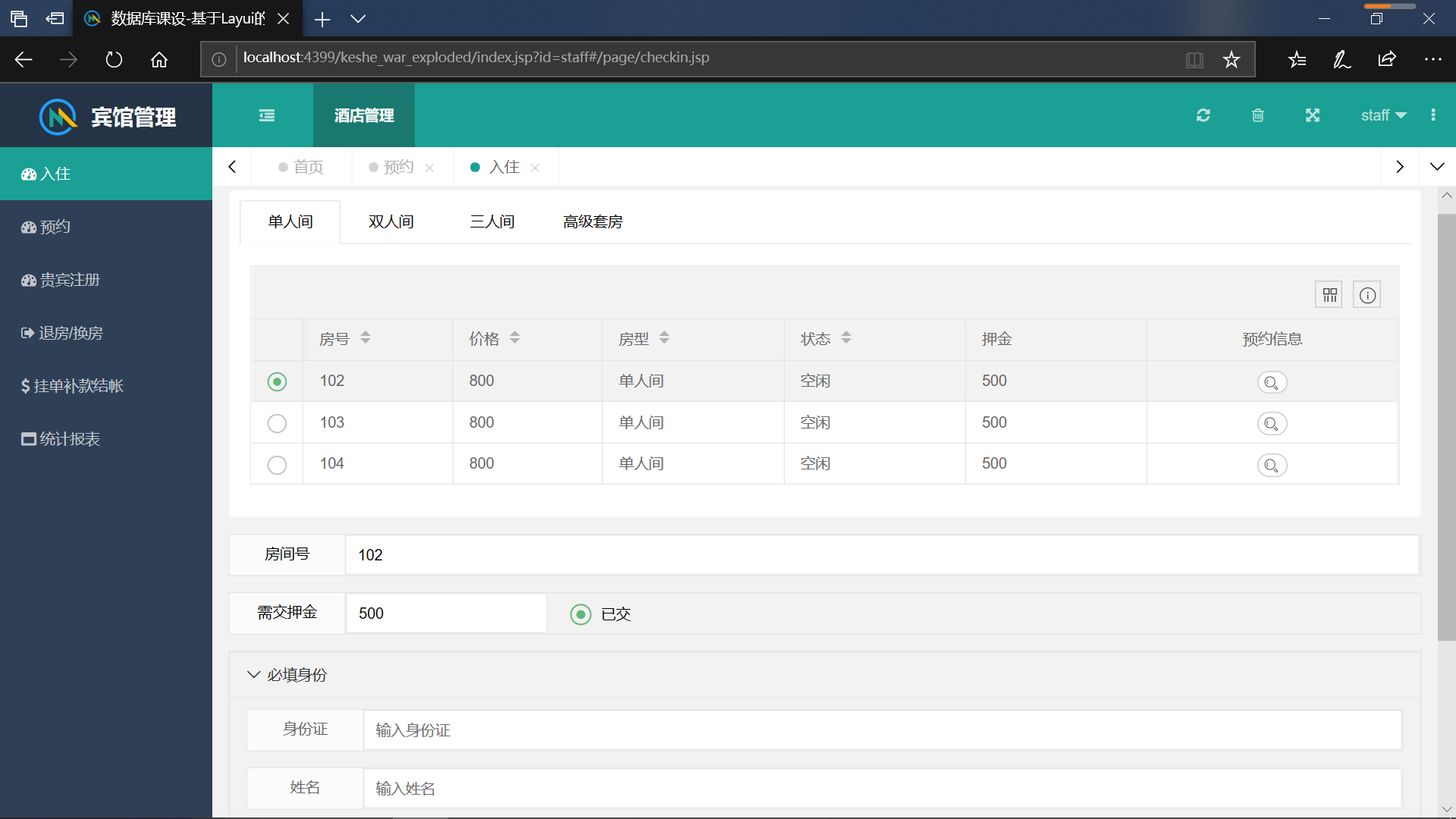
或当查看的房间是未被预约时，提示该房间没有被预约：

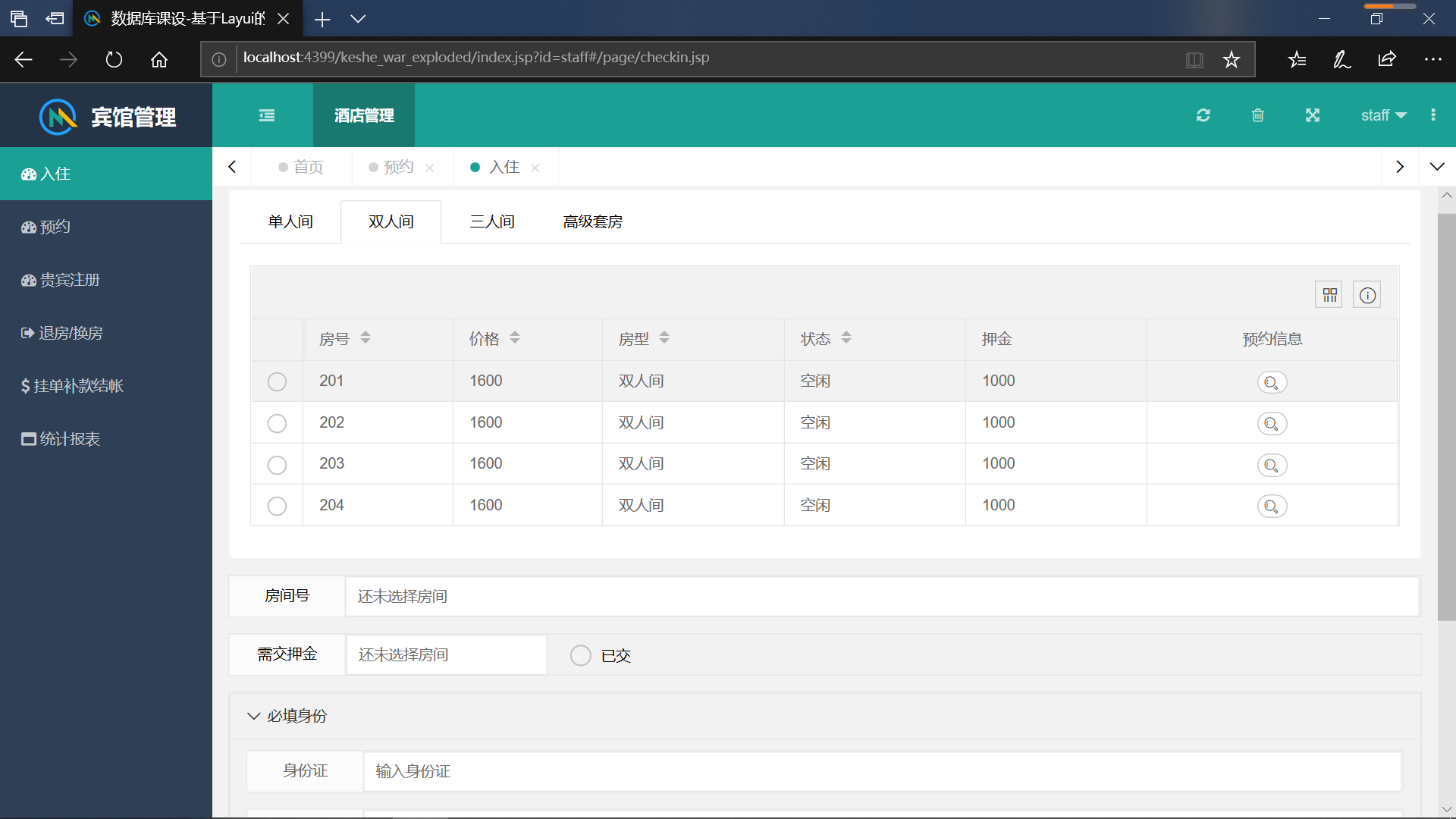
layer.msg(obj.data.roomNumber+'号房<br>未被预约', {  
 time: 20000,  
 offset: 'rt'  
});

#### 2.3卡片栏切换监听

在切换房型时，需要将房间号、需交押金文本框内容清空，并且清空押金已交按钮和表格上已选中房间的单选按钮。

例如一开始选中单人间的102室，然后需要切换房型选择双人间的，点击双人间的卡片时，纪要切换到双人间对应的房间表格，也要清空房间押金文本框和所选单选按钮。





Layui提供了卡片切换的监听接口，用data.index表示点击卡片的位置。其中也使用到了jquery修改文本框的值（清空）和修改单选按钮的值（取消选中）

element.on('tab(filtertab)', function(data){

if(data.index===0 || data.index===1 ||data.index===2 || data.index===3 ) {

//取消选中table的radio

$("input[type=radio]:checked").each(function() {

if( $(this).val()==="on" ) {

$(this).attr("checked", false);

form.render('radio');

}

})

//还原房间号押金文本和"已交"radio

$("#jiao").attr("checked", false);

form.render('radio');

$("#roomliving").val("");

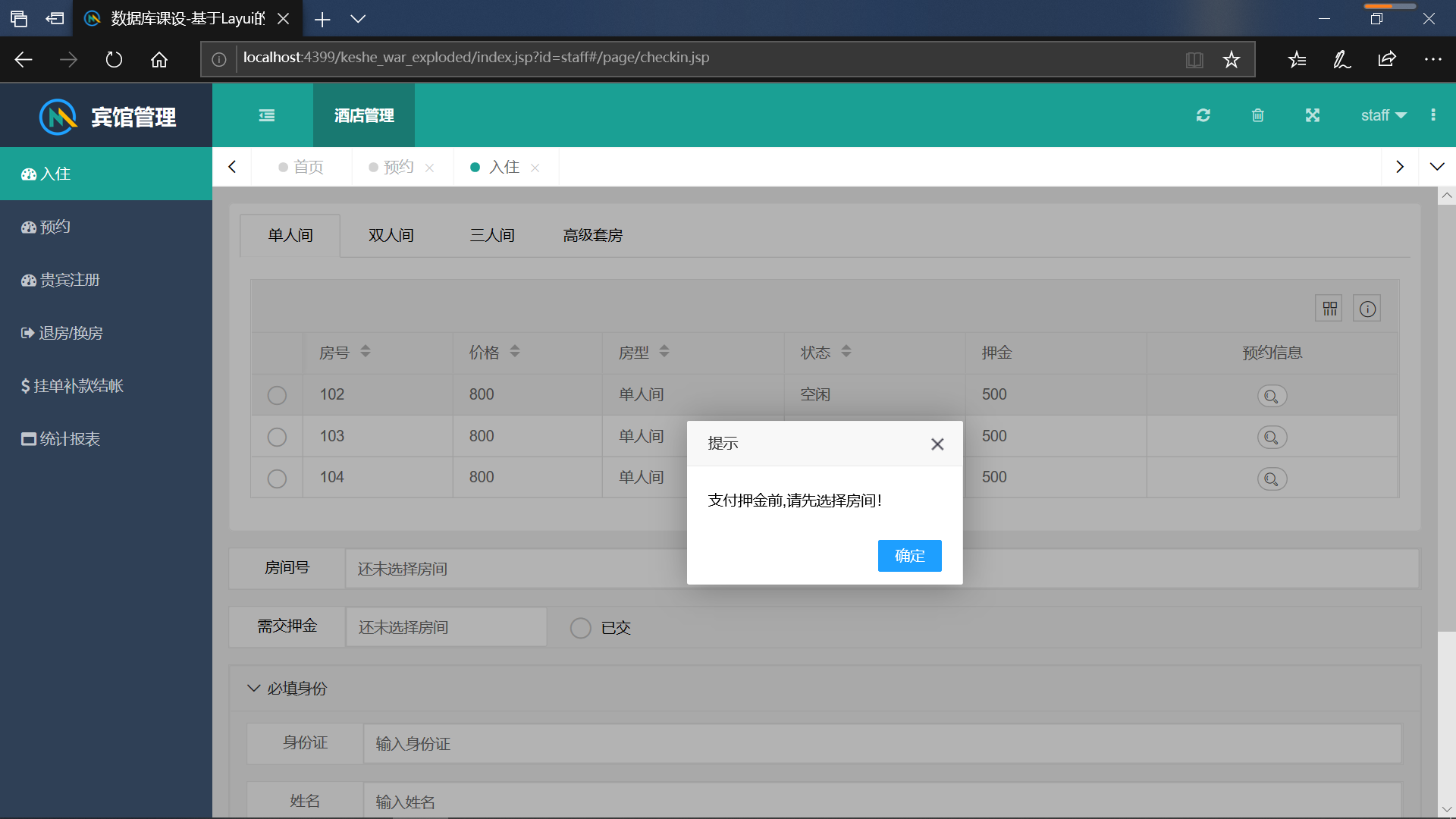
$("#deposit").val("");

}

});

#### 2.4已交押金监听

根据业务逻辑，交押金前必须先选中房间。所以需要给已交押金添加监听，未选房间时就点击已交，则提示。



form.on('radio(jiao)', function(){

if($("#roomliving").val()==""){

layer.open({

title: '提示',

content: '支付押金前,请先选择房间！',

});

$("#jiao").attr("checked", false);

form.render('radio');

}

});

#### 2.5提交入住按钮监听



点击入住需要将该界面的数据传向后端，对数据进行操作，使用Layui接口加jquery形式的ajax向后端传递数据，启用servlet。

form.on('submit(checkinbtn)',function () {

$.ajax({

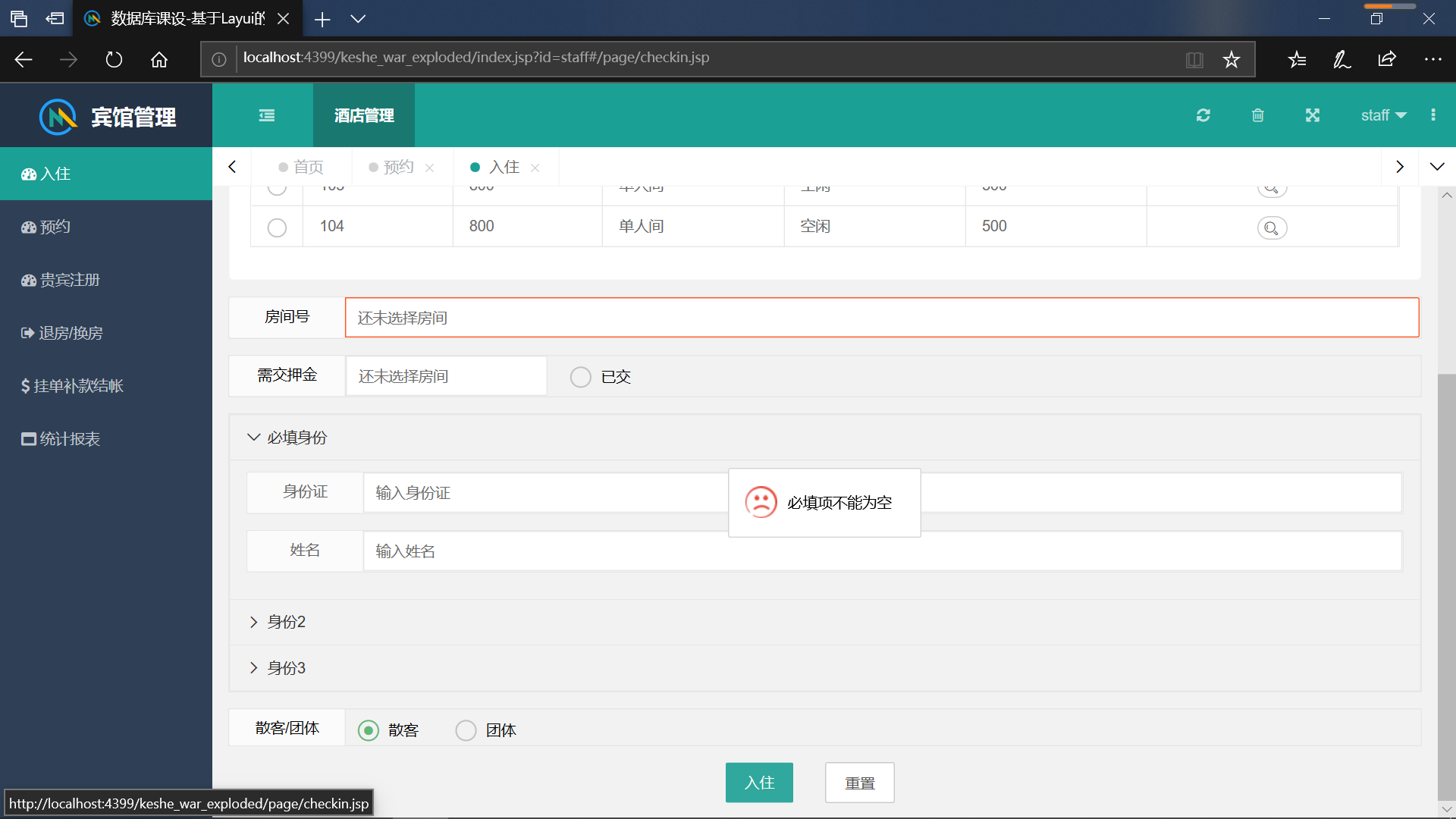
……

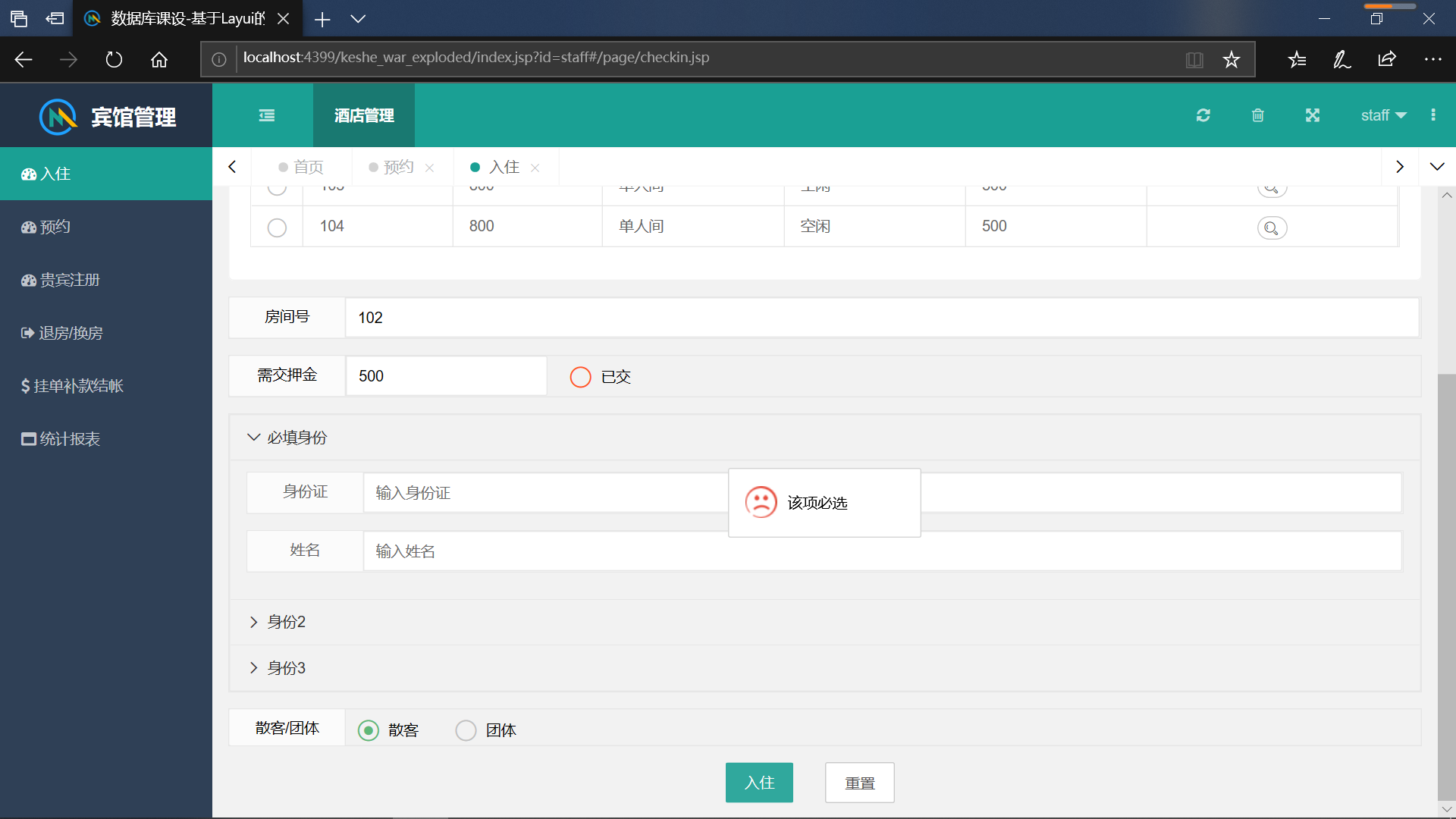
……

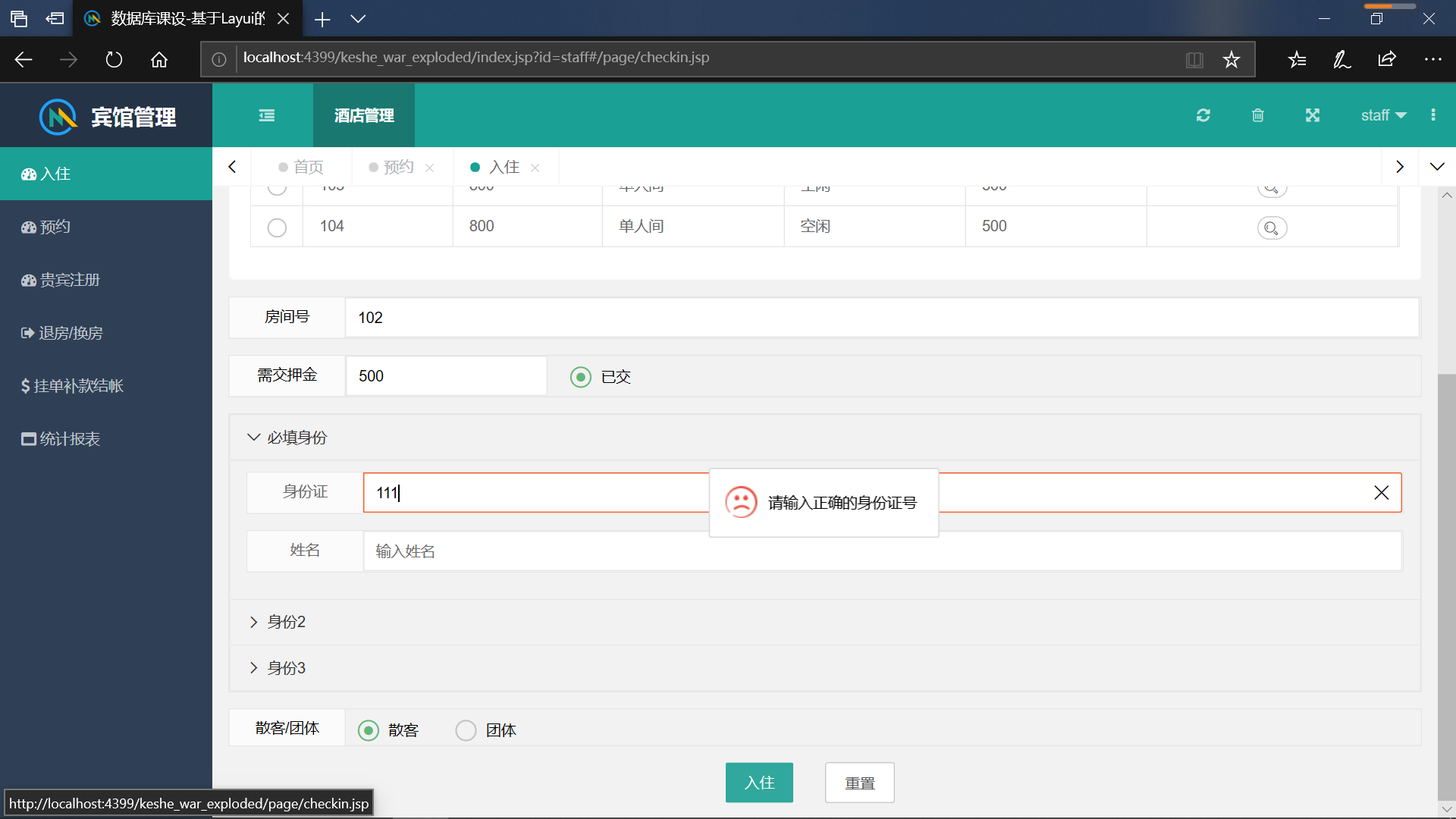
})

});

### 3、输入格式验证







在提交入住时，需要对一些输入格式验证在Layui中提供了一些已经定义好的lay-verify，例如必填，lay-verify="required"、身份验证验证的lay-verify="identity"。

但是有些验证在layui中没有给出，比如单选按钮必填的验证无法使用"required"、"identity"包含了必填。而入住业务逻辑中，已交押金按钮必填、选填身份可以选填（填时验证格式）等验证需要自己定义。

Layui提供了自定义的表单lay-verify机制：form.verify{}。在其中定义了单选必选和选填身份证验证后，就可以给lay-verify赋值对应自定义值，即可实现自定义验证。以单选必选为例，可选身份证类似：

其中单选必选代码部分：

form.verify{

otherReq: function(value,item){

……

……

}

}

押金已交必选单选按钮定义：

<input type="radio" title="已交" …… lay-verify="otherReq">

### 4、数据交换及处理

#### 4.1 layui自带url调用servlet

入住界面的table需要从后端得到空闲或预约的房间信息。layui的table自带渲染方法table.render，其中有一个url参数，若url参数设置成Servlet的地址，就能从servelt获得从后台传入的数据。以单人间为例：

在前端的单人间table.render中

table.render({

elem: '#SingleRoom',

url: '../../keshe\_war\_exploded/AvailableSingleRoomServlet',

toolbar: '#toolbarDemo',

defaultToolbar: ['filter', {

title: '提示',

layEvent: 'LAYTABLE\_TIPS',

icon: 'layui-icon-tips'

}],

cols: [[

……

……

]],

page: false

});

url: '../../keshe\_war\_exploded/AvailableSingleRoomServlet',调用到了可以得到可以入住的单人房间的servlet。在该servlet中，通过调用对数据库操作的Checkin类中方法获得数据。首先看servlet中代码：

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException {

resp.setContentType("text/json;charset=UTF-8");

PrintWriter out=resp.getWriter();

JSONObject jsonObject = new JSONObject();

jsonObject.put("code",0);

jsonObject.put("msg","");

jsonObject.put("count",100);

JSONArray result = null;

CheckIn ci = new CheckIn();

result = JSONArray.fromObject( ci.availableRoom("单人间"));

jsonObject.put("data",result);

out.println(jsonObject.toString());

out.flush();

out.close();

}

可以看出，使用了Checkin类的方法ci.availableRoom("单人间")获得信息，从而生成json对象传给前端显示。而availableRoom中的方法，就是访问Mysql然后查询房间类型是单人间，并且房间状态是预约或者空闲的房间，放入一个ArrayList中返回。

#### 4.2 使用ajax交换数据

入住界面中使用ajax传递数据的有两处，一处是查看预约信息，第二处是确定入住。ajax的基本结构是

$.ajax({

type: "post",

url: "...",

data: {... },

success: function (result) {

...

},

error : function(){

...

}

});

查看预约信息的ajax是放在监听按钮里的，所以能够从监听对象中获取需要的data值：obj.roomNumber获得需要查看预约信息的房间号，url指定调用的servlet路径，将data以键值对的形式传给指定servlet。若调用成功则将servlet中的返回值传到success的参数result中。

table.on('tool(TripleRoom)', function(obj){

$.ajax({

type: "post",

url: "../../keshe\_war\_exploded/ResInfServlet",

data: { par : obj.data.roomNumber },

success: function (result) {

if(result.id == "" && result.name == ""){

layer.msg(obj.data.roomNumber+'号房<br>未被预约', {

time: 20000,

offset: 'rt'

});

}

else{

layer.msg(obj.data.roomNumber

+'号房预约人信息<br>身份证:'

+result.id+'<br>姓名:'

+result.name, {

time: 20000,

offset: 'rt'

});

}

},

error : function(){

alert("数据传输失败!");

}

});

}

查看预约信息的servlet中通过request.getParameter("par");获得ajax中par建中的值。之后该servlet调用Checkin类的ci.ReserveInformation(roomnum);获取预约数据库中该房间号的预约信息。

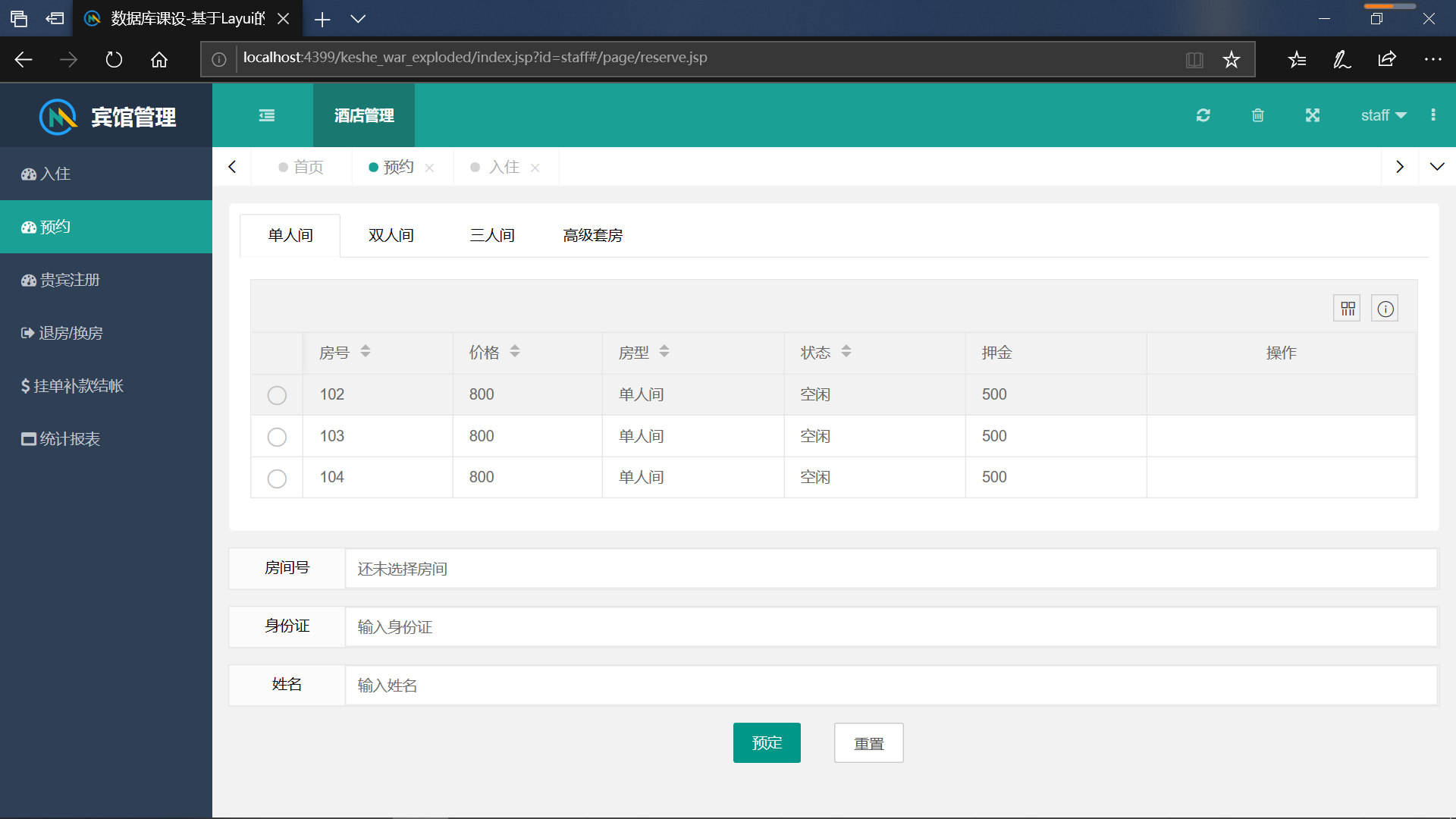
入住按钮的ajax也类似，通过data键值对的形式，从界面上获取入住房间号，入住人身份证，入住人姓名，然后传给入住的servlet，该servlet再去调用实现入住的类方法。

该方法首先修改房间的状态，将Room表对应房间号原来空闲或预约的状态改为入住状态。再在预约表Reserve中查找该房间，若找到删除预约信息。在入住表Checkin增加入住记录。在当前入住Now表中增加记录。在永久信息表中查找该客户，若没有则增加记录。最后在统计报表中找当天记录，若没有新增一条当天的记录，若找到，累加该记录的入住人数。

预约成功则在界面上提示入住成功，入住失败只可能是入住客户中存在已入住的情况，从而导致数据库表中无法插入该数据。提示客户已入住。

## （3）预约业务及代码

### 1、前端布局



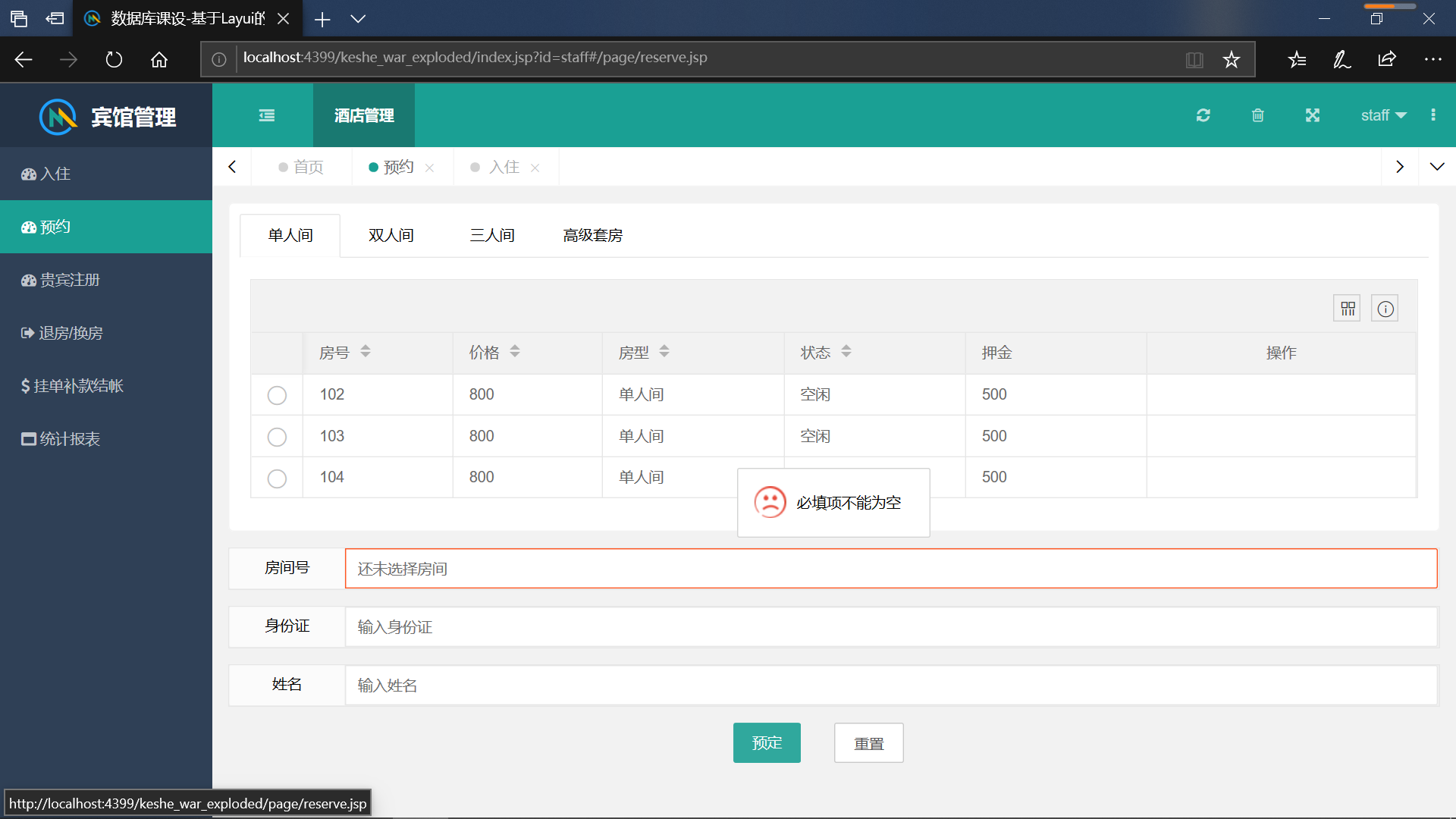
预约界面大致上和入住界面类似并且更简单。整体一个form表单，然后第一个表单元素是一个卡片，点击卡片，可以切换到对应的空闲房间表。第二个表单元素是不可编辑的房价号，通过选中房间自己填写。第三个表单元素是必填的身份证格式验证身份栏。第四个表单元素是一个文本框。最后一个表单元素包含提交按钮和重置按钮。

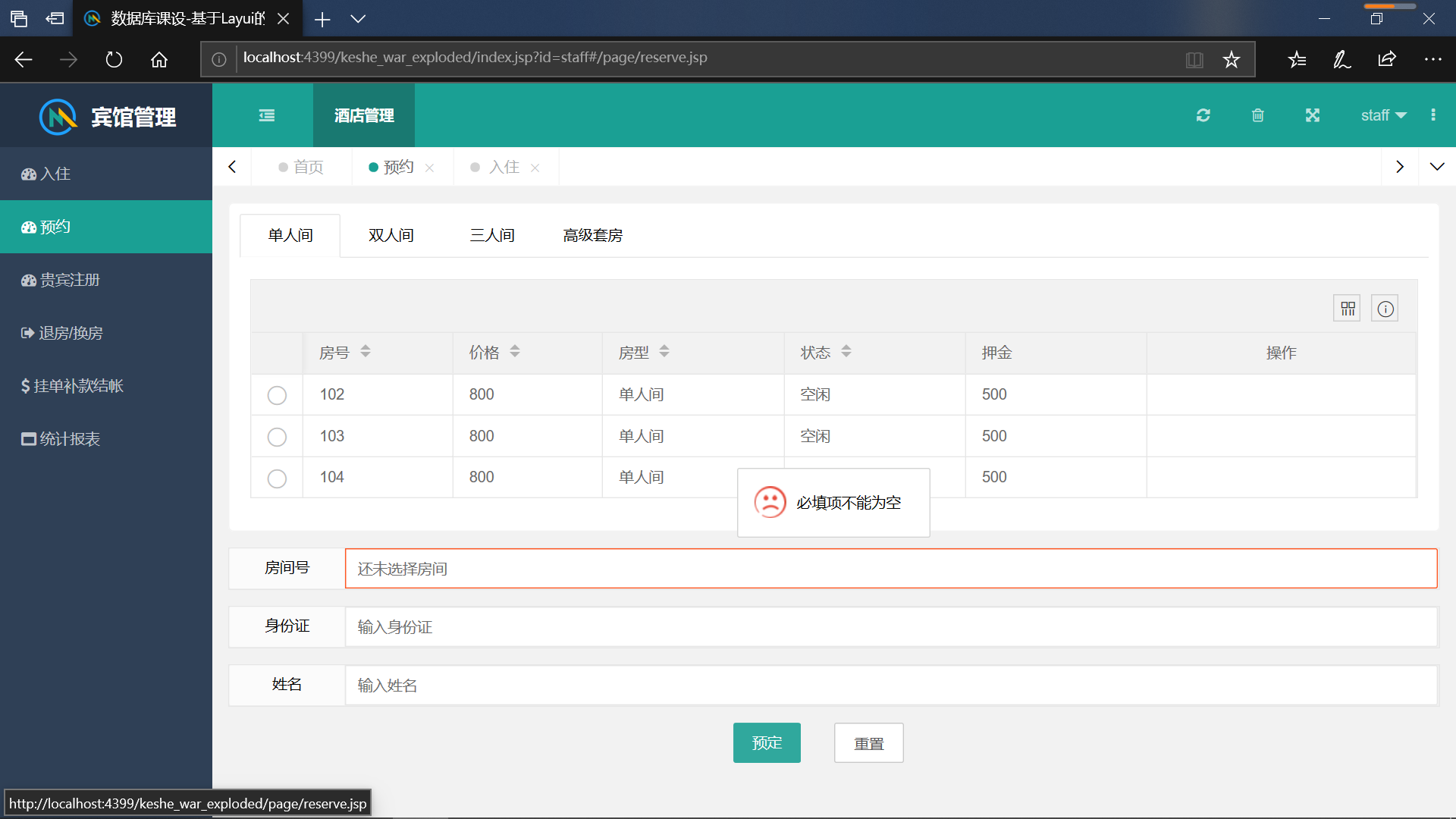
### 2、前端监听

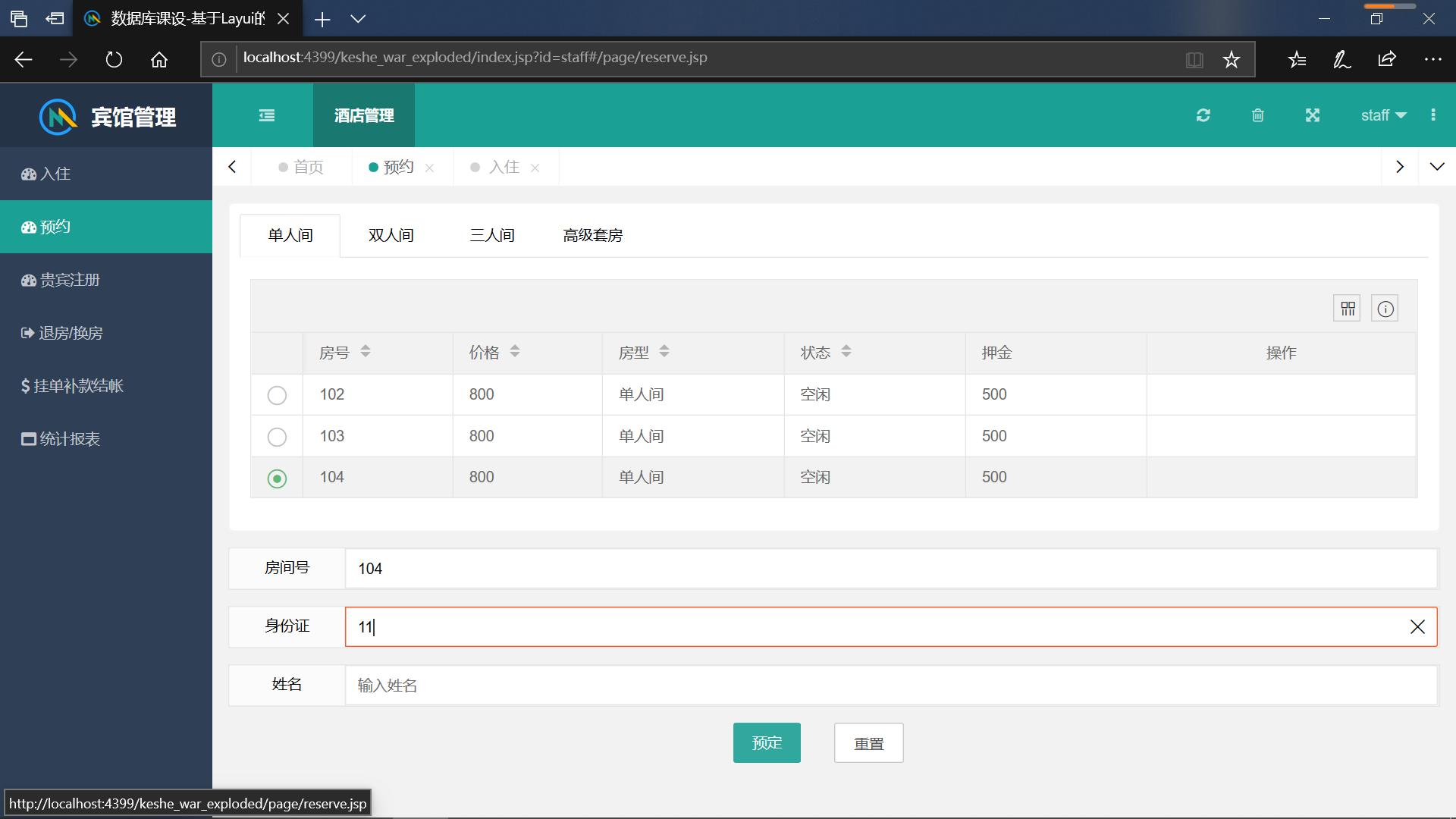
基本上也和入住类似，有卡片切换监听，当切换卡片时，显示对应房间类型的空闲房间的表格。监听预约按钮。

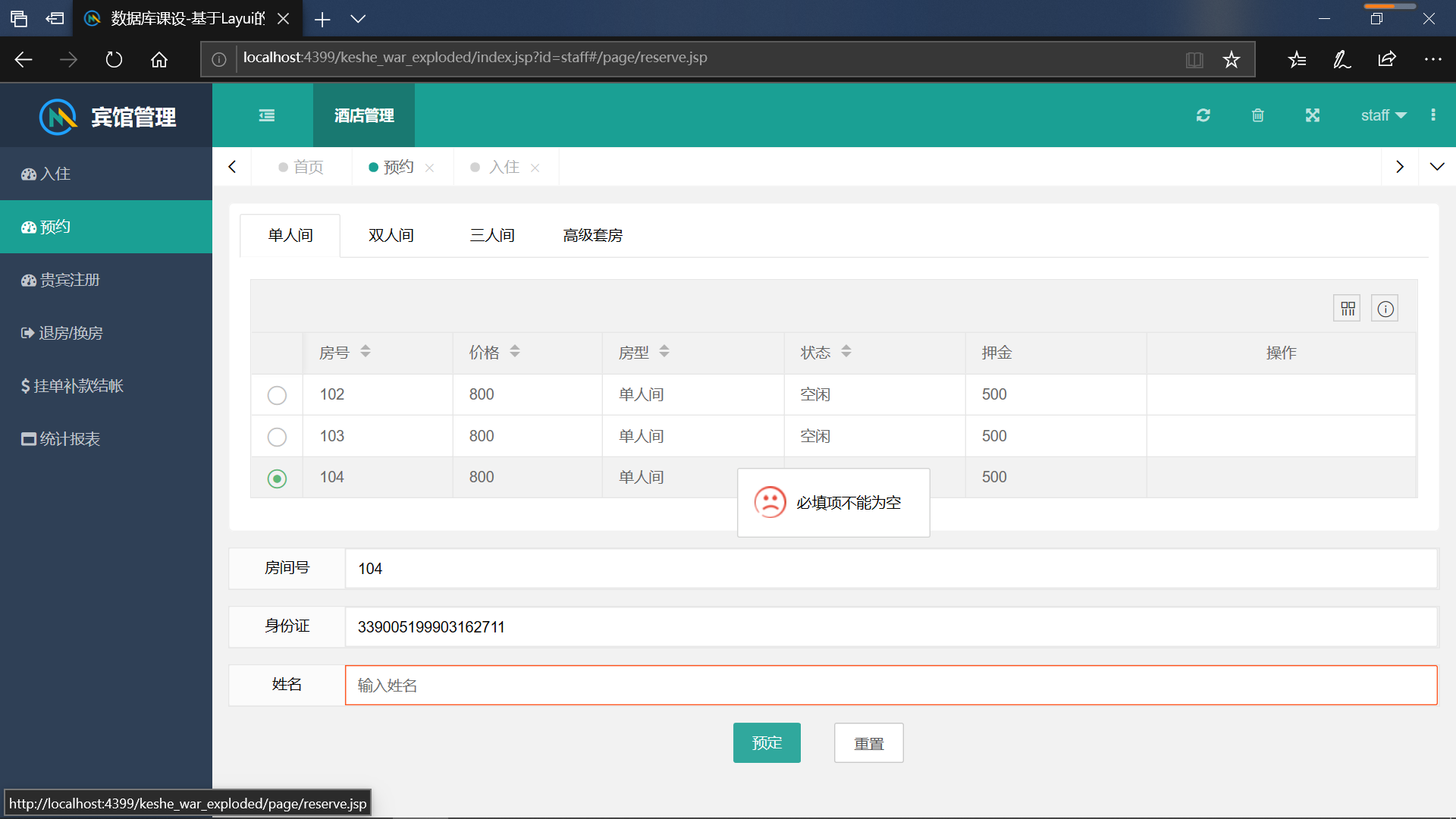
### 3、输入格式验证

无自定义的验证，身份验证使用“identity”，其他几项输入框：房间号、姓名使用“required”都为必填项









### 4、数据交换及处理

table和入住界面类似，通过layui的table.render中url指定servlet，在servlet中又调用查询房间函数。不同的是，预约界面的table需要显示的空闲状态的房间，入住界面显示的是空闲和预约的房间。

点击预约时，通过ajax获得房间、身份证、姓名框中值，以键值对形式，传入url指定ReserveServlet中，预定成功则显示预定成功。该项操作在数据库正确情况下不会出现错误。

form.on('submit(resbtn)',function () {

$.ajax({

type: "post",

url: "../../keshe\_war\_exploded/ReserveServlet",

data: {

roomnum: $('#roomnum').val(),

resid: $('#resid').val(),

resname: $('#resname').val(),

},

success: function (result) {

if(result.flag==1){

var index = layer.alert("预定成功！", {

skin: 'layui-layer-molv', //样式类名

closeBtn: 0,

title: '提示',

anim: 5

}, function () {

// 关闭弹出层

layer.close(index);

//关闭当前窗口

var iframeIndex = parent.layer.getFrameIndex(window.name);

parent.layer.close(iframeIndex);

parent.location.reload();

});

}

},

error: function () {

alert("数据传输失败!");

}

});

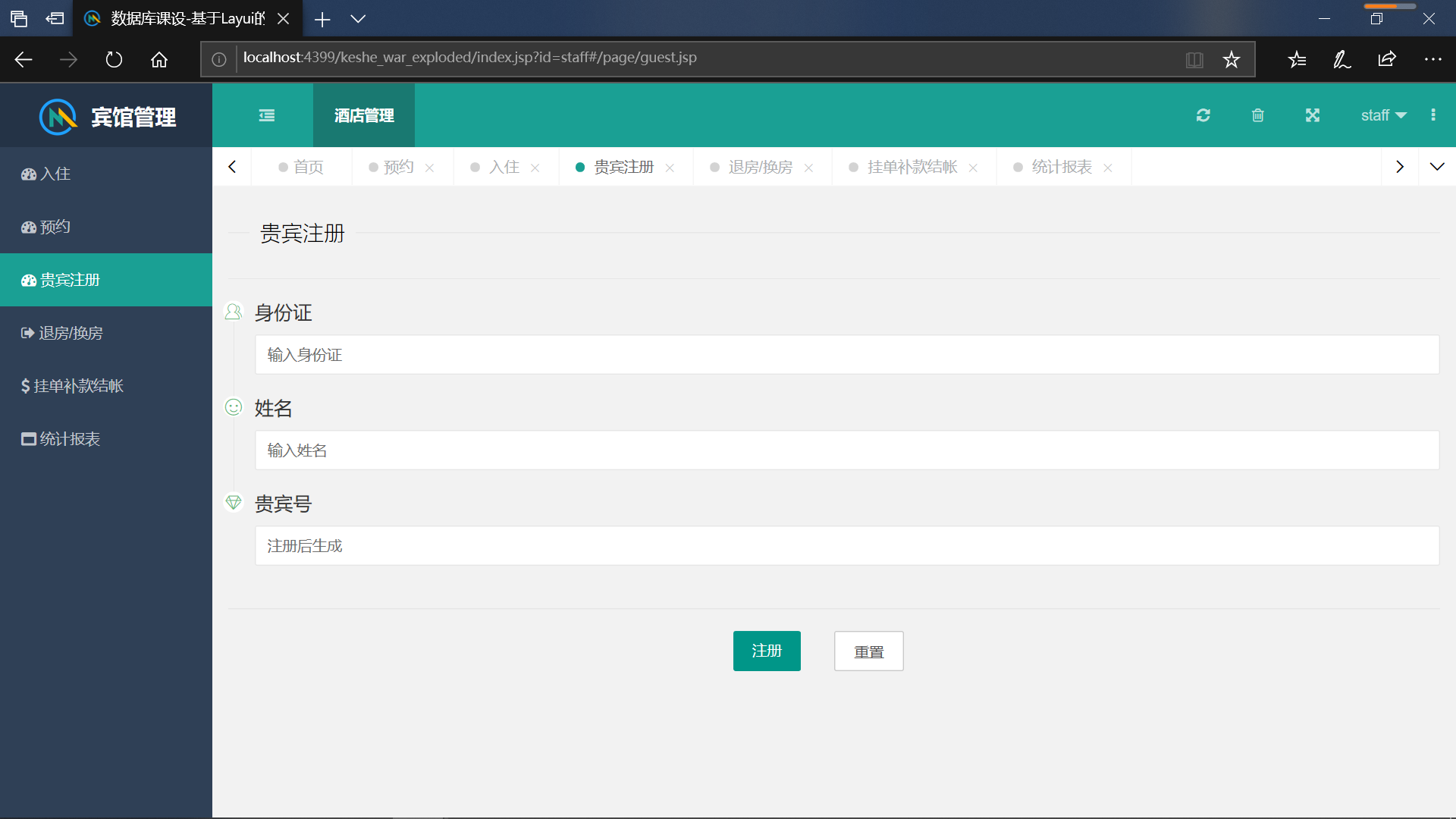
return false;

});

在ReserveServleti中，调用Reserve类的reserve函数，该函数操作数据库，首先在预约信息表Reserve中增加一条预约房间的记录。再在房间管理表中更新房间状态，将原来空闲改为预约。在永久信息表中查询预约的客户，若没找到，增加一条客户信息的记录。最后在统计报表中找当天记录，若没找到，增加一条当天记录，预约房间数设为1；若找到就把预约房间数加1。

## （2）贵宾注册业务及代码

### 1、前端布局



贵宾注册中，总的还是一个表单form，最不同的是其中一个表单元素使用到了时间先TimeLine。由于注册界面只需要三个文本框来填信息，使用了TimeLine后，界面行对比较充实。以其中一个时间线元素为例

<div class="layui-form-item" pane>

<li class="layui-timeline-item">

<i class="layui-icon layui-timeline-axis">&#xe770;</i>

<div class="layui-timeline-content layui-text">

<h3 class="layui-timeline-title">身份证</h3>

<input id="gueid" name="gueid" class="layui-input" type="text" placeholder="输入身份证" autocomplete="off" lay-verify="identity">

</div>

</li>

……

……

</div>

### 2、前端监听

监听只涉及到注册贵宾的按钮的监听

form.on('submit(guebtn)',function () {

$.ajax({

type: "post",

url: "../../keshe\_war\_exploded/GuestServlet",

data: {

id: $('#gueid').val(),

name: $('#guename').val()

},

success: function (result) {

if (result.vipnum != null && result.vipnum != "该用户已是贵宾,不需重复注册!") {

$('#guenum').val(result.vipnum);

layer.msg('注册成功!<br>该顾客贵宾号为' + result.vipnum, {

time: 10000

});

} else if (result.vipnum == "该用户已是贵宾,不需重复注册!") {

layer.msg('注册失败!<br>' + result.vipnum, {

time: 10000

});

} else if (result.vipnum == null) {

layer.msg('注册失败!<br>数据库错误', {

time: 10000

});

}

},

error: function () {

alert("数据传输失败!");

}

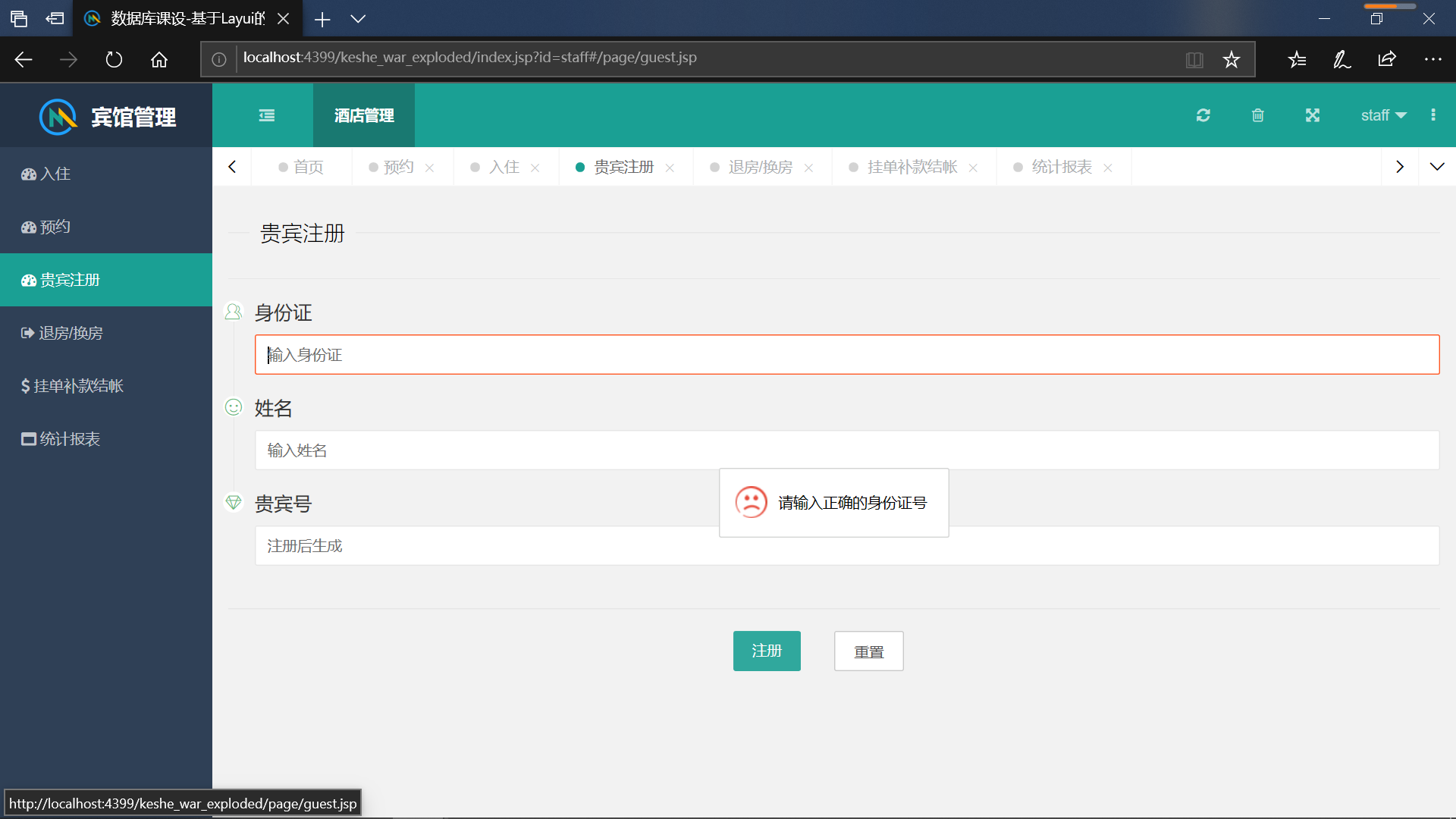
});

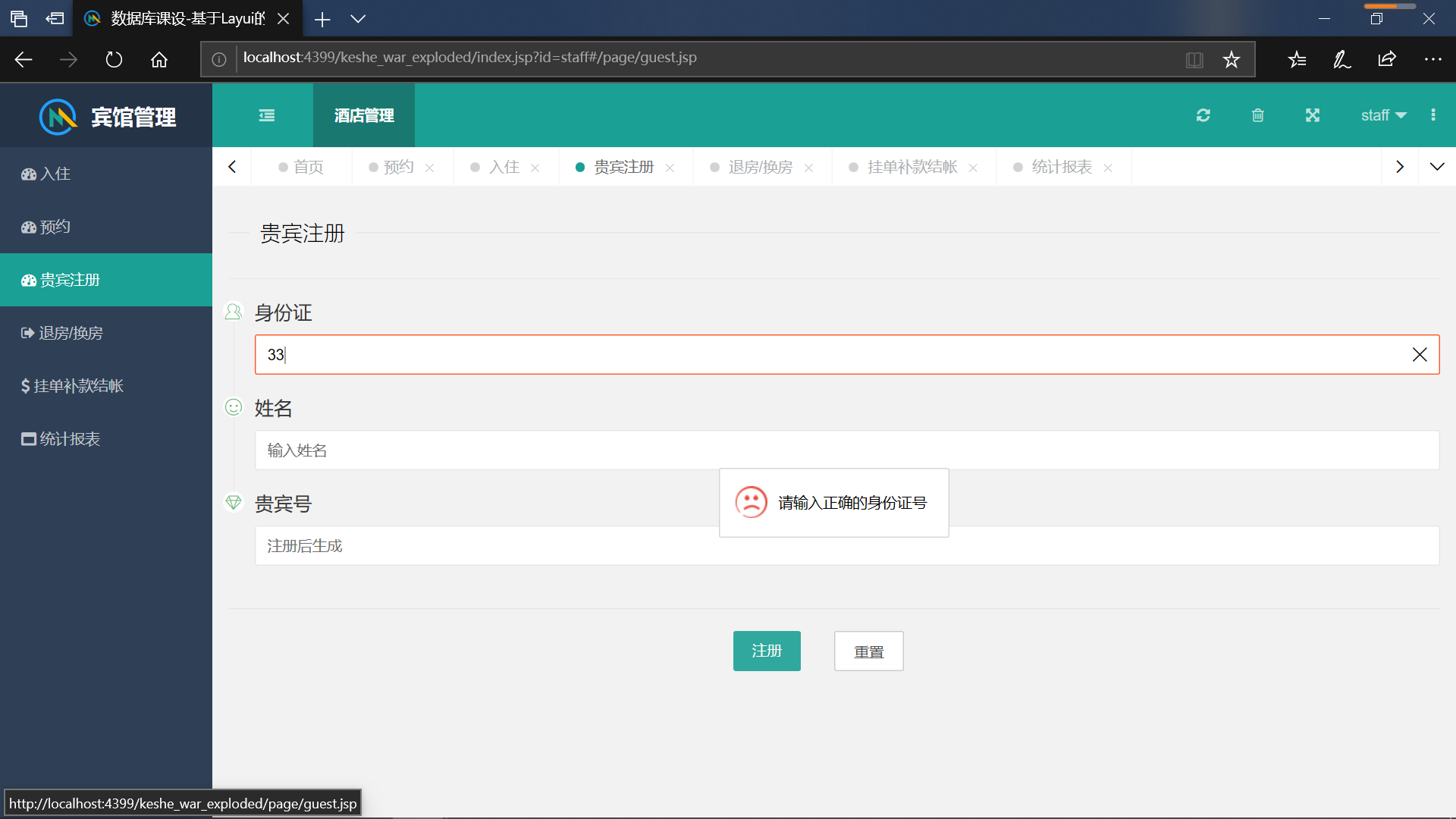
return false;

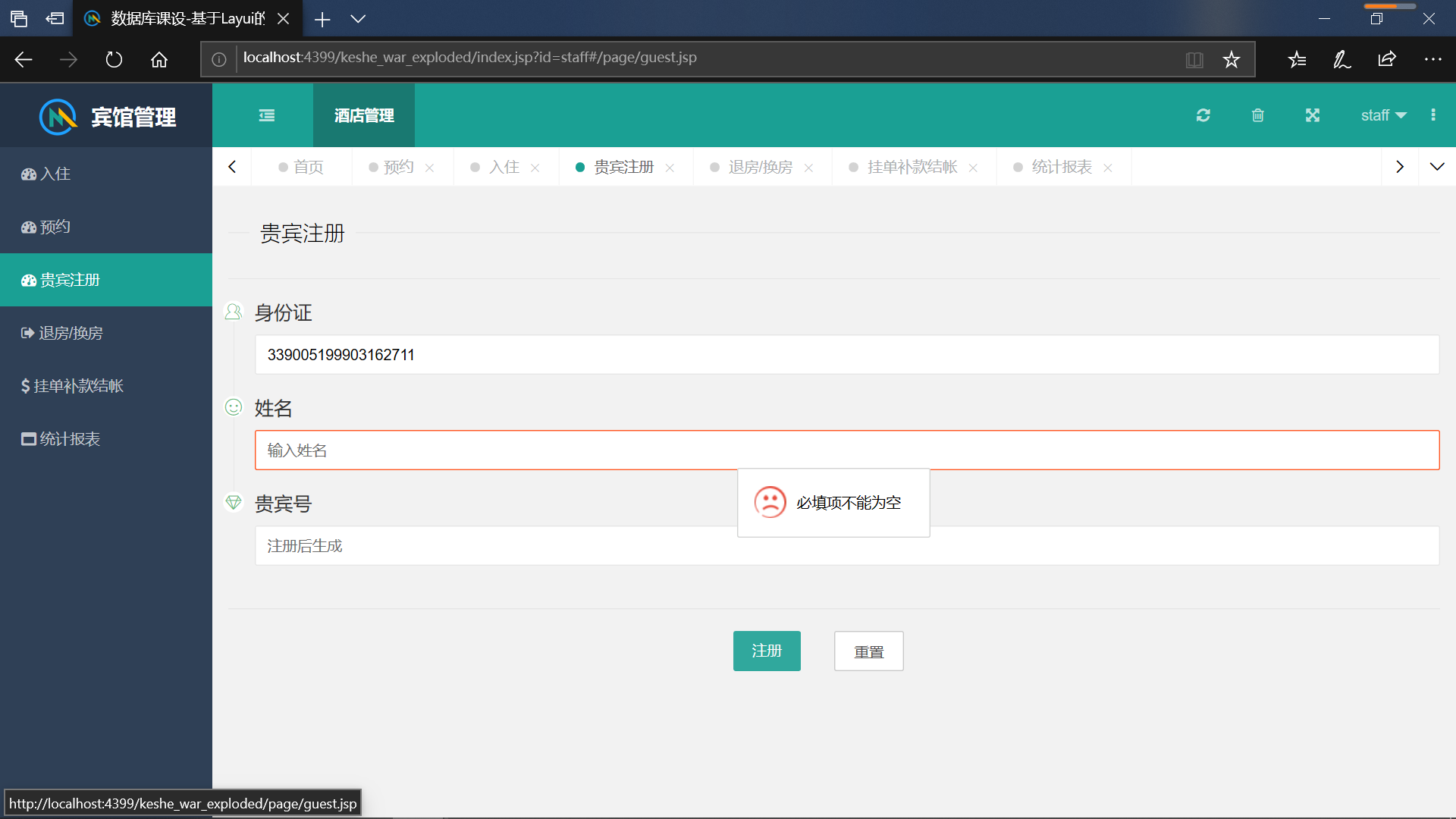
});

});

### 3、输入格式验证





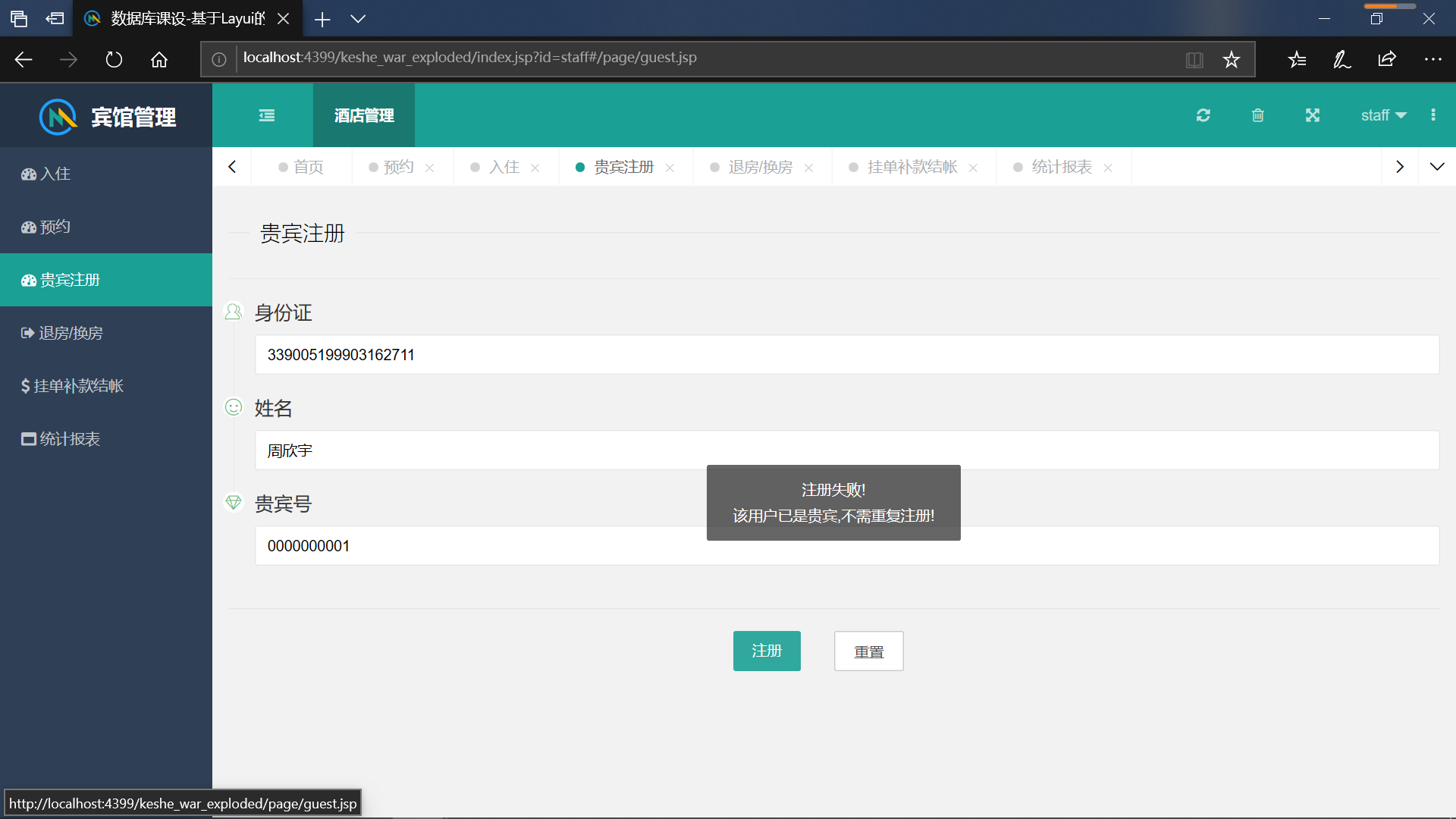


在贵宾注册界面中的格式验证，身份证必须要填，并且需要符合格式，姓名也必须要填。

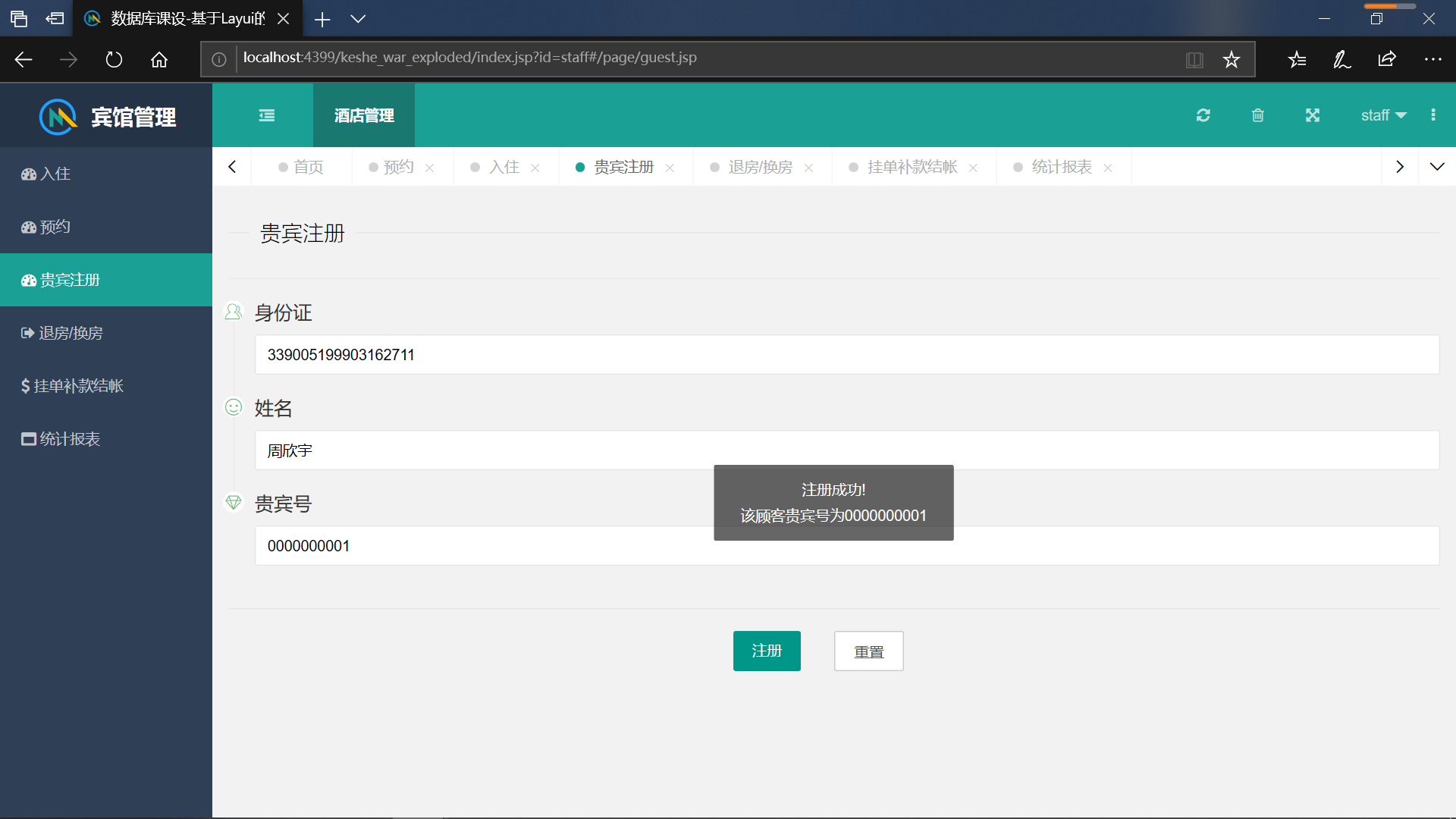
### 4、数据交换及处理

点击注册后，使用ajax将身份证和姓名传入servlet，在servlet中调用GuestRegister中函数，访问vip表。

首先搜索是否有该身份证，若有则返回消息不用重新注册。



若没有则执行下面的操作。倒序搜索取第一条记录的贵宾号，然后由于是Strng类型的，将其转成int型加1后再转成String型前面补0，然后以此为生成的贵宾号，和身份证以及姓名添加到vip表中。同时在永久信息表Permanent找身份证，若有则需要改变该客户的vipstaus，若没有增加一条客户记录。并将vip号返回到servlet，再传到前端显示在贵宾号栏中。



# 六、遇到的问题及心得

## （1）开发前期

开发前期比较迷茫，由于组内有一些分歧，组员有想用vue+springboot框架的，也有想用jsp+servletd。组内没有选出领队，所以大家努力的方向不一样，也对时间规划过于有信心。

## （2）关于debug

由于项目规模略微比较大，出现的错误很难定位到具体代码。在刚着手项目，经常花费很长时间去定位错误的地方。但在后来的开发过程中，jsp前端适当位置加入console.lof，java中适当模块加入System.out.println，一些日有效合适的志记录和输出语句，能大大提高定位错误的效率。

## （3）java和MySql时间类型匹配

从数据库中读取时间需要使用PreparedStatement.get…..()向数据库中插入时间需要使用PreparedStatement.set…..()。使用的类型不对就会出错，而且MySql中含有date（显示时分）和datetime（显示时分秒）等类型。java中的时间类型更是繁多比如java.sql.date、java.util.date、timstamp等，类型匹配就是个很大的问题。最后时间匹配类型总结如下：

date java.sql.Date

Datetime java.sql.Timestamp

Timestamp java.sql.Timestamp

Time java.sql.Time

Year java.sql.Date

## （4）后期整合

我们组采用的是分功能模块完成，而不是分前后端完成。由于我们考虑到学习阶段，想要前后端的知识都有所学到，才决定按功能模块分。

所以在最后的整合中，最大的问题就是数据库有略微的不统一。虽然在前期数据库设计时大家是一起参与的。但是后期实际操作过程中，一些变动没有及时讨论，导致整合时不是很顺利。这也给我们提了很重要的醒。前后端分离的设计方式可能确实比功能模块的方式要更加便于整合。