软件项目说明书（B/S部分）

负责人：张天浩独立完成

一、程序配置

1.1 软件使用及框架

编程平台：IntelliJ IDEA

前端框架：Bootstrap 4.3.1

后端框架：springMVC

js文件：javascript原生语言及jquery 3.2.1

数据库:oracle11g 11.2.0.1.0

1.2 数据库配置

db.properties

```
jdbc.driver=oracle.jdbc.OracleDriver
jdbc.url=jdbc:oracle:thin:@localhost:1521:orcl
jdbc.username=zth
jdbc.password=123456
```

1.3 web.xml配置

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>
  </context-param>

  <!--    3 使用spring提供的编码过滤器-->
  <filter>
    <filter-name>characterEncodingFilter</filter-name>
    <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
      <param-name>forceRequestEncoding</param-name>
      <param-value>true</param-value>
    </init-param>
    <init-param>
      <param-name>forceResponseEncoding</param-name>
      <param-value>true</param-value>
    </init-param>
  </filter>


  <!--    配置REST风格的转换-->
  <filter>
    <filter-name>hiddenHttpMethodFilter</filter-name>
    <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-
class>
  </filter>
  <filter>
    <filter-name>httpPutFormContentFilter</filter-name>
    <filter-
class>org.springframework.web.filter.HttpPutFormContentFilter</filter-class>
  </filter>
  <!--    编码过滤器映射-->
  <filter-mapping>
    <filter-name>characterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <!--    REST过滤器映射-->
  <filter-mapping>
    <filter-name>hiddenHttpMethodFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>httpPutFormContentFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>


  <!--    2 配置在web容器启动时,使spring的文件可以加载-->
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <!--  1 配置前端控制器,以后的用户访问都交给DispatcherServlet处理-->
  <servlet>
    <servlet-name>dispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:springmvc.xml</param-value>
    </init-param>
    <!--    启动服务器,创建该serlvet时即加载-->
    <load-on-startup>1</load-on-startup>
  </servlet>


  <servlet-mapping>
    <servlet-name>dispatcherServlet</servlet-name>
    <!--    所有的请求都交给该servlet处理-->
    <url-pattern>/</url-pattern>
  </servlet-mapping>
</web-app>
```

1.4 后端springMVC配置

1.4.1 依赖导入

```
<dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
      <version>3.5.10</version>
    </dependency>


    <dependency>
      <groupId>com.oracle</groupId>
      <artifactId>ojdbc6</artifactId>
      <version>11.2.0.1.0</version>
    </dependency>

    <dependency>
      <groupId>com.github.pagehelper</groupId>
      <artifactId>pagehelper</artifactId>
      <version>5.1.8</version>
    </dependency>

    <!--        json依赖-->
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.10.0</version>
    </dependency>
    <!--        导入spring的依赖-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-aop</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
    </dependency>
```

```xml
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-beans</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-expression</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!--        spring web 模块的包-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-web</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!--        jdbc-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!--        事务-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-tx</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!--        测试-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-test</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!--        织入包-->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-aspects</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.aspectj</groupId>
      <artifactId>aspectjweaver</artifactId>
      <version>1.6.12</version>
    </dependency>
    <!--        spring整合mybatis-->
    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis-spring</artifactId>
      <version>1.3.1</version>
    </dependency>
    <!--        数据库连接池-->
    <dependency>
```

```xml
      <groupId>c3p0</groupId>
      <artifactId>c3p0</artifactId>
      <version>0.9.1.2</version>
    </dependency>

    <dependency>
      <groupId>org.mybatis</groupId>
      <artifactId>mybatis</artifactId>
      <!--            根据目前的测试,3.4.5和3.5.5版本不会报错,其他版本还会出现错误提示-->
      <version>3.4.5</version>
    </dependency>

    <!--        mybatis的逆向工程-->
    <dependency>
      <groupId>org.mybatis.generator</groupId>
      <artifactId>mybatis-generator-core</artifactId>
      <version>1.3.7</version>
    </dependency>
    <!--        导入单元测试junit的坐标-->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.12</version>
      <!--            scope依赖形式-->
      <!--                compile  编译和发布时都会将该包放在构建路径中  mybatis-->
      <!--                test 测试时能够使用,不会发布    junit-->
      <!--                provider 不会发布      servlet-->
      <scope>compile</scope>
    </dependency>
    <!--        servlet和jsp依赖,EL表达式-->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jsp-api</artifactId>
      <version>2.0</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>jstl</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>
    <!--        日志功能-->


    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.12</version>
    </dependency>
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
```

```
        <version>1.7.7</version>
    </dependency>
    <!--         上传文件的依赖-->
    <dependency>
      <groupId>commons-fileupload</groupId>
      <artifactId>commons-fileupload</artifactId>
      <version>1.3.3</version>
    </dependency>
    <dependency>
      <groupId>commons-io</groupId>
      <artifactId>commons-io</artifactId>
      <version>2.2</version>
    </dependency>
    <!--         后端数据校验包,JSR303-->
    <dependency>
      <groupId>org.hibernate.validator</groupId>
      <artifactId>hibernate-validator</artifactId>
      <version>6.0.17.Final</version>
    </dependency>
    <dependency>
      <groupId>javax.validation</groupId>
      <artifactId>validation-api</artifactId>
      <version>2.0.1.Final</version>
    </dependency>
```

1.4.2 mybatis配置

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <properties resource="db.properties"></properties>
    <settings>
<!--         配置驼峰映射-->
        <setting name="mapUnderscoreToCamelCase" value="true"/>
            <!--    设置mybatis输出日志    STDOUT_LOGGING把日志输出到控制台   -->
        <setting name="logImpl" value="STDOUT_LOGGING" />
    </settings>
<!--    配置别名映射-->
    <typeAliases>
<!--         默认情况下别名是类名的小写开头-->
        <package name="file.zhang.entity"/>
    </typeAliases>
</configuration>
```

1.4.3 mybatis 托管数据库，Mapper及实体类生成配置

```
<!DOCTYPE generatorConfiguration PUBLIC
        "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
        "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
<generatorConfiguration>
    <context id="oracle11g" targetRuntime="MyBatis3">
```

```
        <commentGenerator>
            <property name="suppressAllComments" value="true"/>
        </commentGenerator>

<!--        数据库连接-->
        <jdbcConnection driverClass="oracle.jdbc.OracleDriver"
                        connectionURL="jdbc:oracle:thin:@localhost:1521:orcl"
userId="zth" password="123456" />
<!--    指定javaBean的生成位置-->
        <javaModelGenerator targetPackage="file.zhang.entity"
targetProject="./src/main/java">
            <property name="enableSubPackages" value="true"/>
        </javaModelGenerator>
<!--        指定mapper的接口文件的位置-->
        <sqlMapGenerator targetPackage="file.zhang.dao"
targetProject="./src/main/java">
            <property name="enableSubPackages" value="true"/>
        </sqlMapGenerator>

        <javaClientGenerator type="XMLMAPPER" targetPackage="file.zhang.dao"
targetProject="./src/main/java">
            <property name="enableSubPackages" value="true"/>
        </javaClientGenerator>
<!--        指定数据表的生成策略-->
        <table tableName="GUEST_INFO" domainObjectName="Guest_info" />
        <table tableName="RESERVE_INFO" domainObjectName="Reserve_info" />
        <table tableName="ROOM_INFO" domainObjectName="Room_info" />


    </context>
</generatorConfiguration>
```

1.4.4 springMVC 配置

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
https://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <!--    1 组件扫描,因为service层和持久层已经由spring负责管理,所以springmvc仅需要扫描
管理Controller即可-->
    <context:component-scan base-package="file.zhang">
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
        <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Service"/>
        <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Repository"/>
    </context:component-scan>
```

```xml
    <!--    2 配置视图解析器_网址前后缀-->
    <bean id="internalResourceViewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/pages/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>

    <!--    3 开启springMVC的注解驱动-->
    <mvc:annotation-driven/>

    <!--    4 配置静态资源过滤-->
    <!--    <mvc:default-servlet-handler />-->

    <mvc:resources mapping="/css/**" location="/css/"/>
    <mvc:resources mapping="/js/**" location="/js/"/>
    <mvc:resources mapping="/images/**" location="/images/"/>

    <!--    5 解决$.get从控制器返回的中文乱码问题-->
    <mvc:annotation-driven>
        <mvc:message-converters>
            <bean
class="org.springframework.http.converter.StringHttpMessageConverter">
                <property name="supportedMediaTypes">
                    <list>
                        <value>text/plain;charset=UTF-8</value>
                        <value>text/html;charset=UTF-8;</value>
                        <value>application/json;charset=UTF-8</value>
                    </list>
                </property>
            </bean>
        </mvc:message-converters>
    </mvc:annotation-driven>

    <!--    后端数据校验-->
    <bean id="validatorFactoryBean"
class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">
        <property name="providerClass"
value="org.hibernate.validator.HibernateValidator"/>
    </bean>
    <!--    数据类型转换-->
    <!-- 引入spring框架的类型转换工厂 -->
    <bean id="conversionService"
class="org.springframework.format.support.FormattingConversionServiceFactoryBean
">
        <!-- 注入 -->
        <property name="converters">
            <!-- 注册自己定义的字符串转日期类型的转换器 -->
            <bean class="file.zhang.utils.StringToDateConvert"></bean>
        </property>
    </bean>
    <!--    开启注解支持,使自定义的转换器生效-->
    <mvc:annotation-driven conversion-service="conversionService"/>


</beans>
```

1.4.5 spring整合各依赖及注册注解

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       https://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx.xsd
       http://www.springframework.org/schema/aop
       https://www.springframework.org/schema/aop/spring-aop.xsd">
<!--    </bean>-->
    <!--1 开启注解的扫描,使spring能够自动扫描到组件纳入到IOC容器中-->
    <context:component-scan base-package="file.zhang">
        <!--        将service和持久层的bean交给spring管理,表现层的bean交给springmvc管
理-->
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Service"/>
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Repository"/>
        <context:exclude-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>
    <!--    2 使用spring整合mybatis-->
    <!--    2.0 引入外部的数据连接信息-->
    <context:property-placeholder location="classpath:db.properties">
</context:property-placeholder>
    <!--    2.1 配置连接池(dbcp,c3p0,durid),通过连接池管理数据库的连接,可以保留mybatis的
配置文件,做一些spring配置不太方便的工作-->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="${jdbc.driver}"></property>
        <property name="jdbcUrl"
                  value="${jdbc.url}"></property>
        <property name="user" value="${jdbc.username}"></property>
        <property name="password" value="${jdbc.password}"></property>
    </bean>
    <!--    2.2 配置SqlSessionFactory工厂,使用的数据源就应该是 c3p0提供的数据源-->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
        <!--        引入独立的mybatis的配置文件,将一些在spring中不易的配置信息放在
mybatis中-->
        <property name="configLocation" value="classpath:mybatis-config.xml">
</property>

    </bean>
    <!--    2.3 配置动态代理指定接口-->
    <bean id="mapperScannerConfigurer"
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="file.zhang.dao"></property>
    </bean>
    <!--    3 配置spring框架的声明式事务-->
    <!--    3.1 配置事务管理器-->
    <bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="dataSource"></property>
    </bean>
    <!--    3.2 配置事务的增强-->
    <tx:advice transaction-manager="transactionManager"
id="transactionInterceptor">
        <tx:attributes>
            <!--            所有的方法都是事务方法-->
            <tx:method name="*" isolation="DEFAULT"/>
            <!--            对于查询方法,仅只读-->
            <tx:method name="select*" read-only="true"/>
        </tx:attributes>
    </tx:advice>
    <!--    3.3 配置AOP,事务如何切入(可以是注解的方式,本案例使用配置文件)-->
    <aop:config>
        <!--        表达式的写法:   访问权限[可省略] 返回值类型(*表示所有的返回值) 包.子
包.方法(参数)-->
        <aop:advisor advice-ref="transactionInterceptor"
                     pointcut="execution(*
file.zhang.service.impl.*ServiceImpl.*(..))"/>
    </aop:config>

<!--    上传文件-->
    <bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="defaultEncoding" value="UTF-8"></property>
        <property name="maxUploadSize" value="10000000"></property>
    </bean>
</beans>
```

二、前端功能点

2.1 html页面功能点

2.1.1 横向导航栏

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark fixed-top" >
    <!--        左对齐-->
    <!-- Brand/logo -->
    <a class="navbar-brand" href="./main.html">
        <img src="/images/logo.jpg" alt="logo" style="width:60px;">
    </a>
    <!-- 普通导航 -->
    <ul class="nav col-9 " id="function_opt">
        <li class="nav-item">
            <a class="nav-link" href="./book_room.html">预约房间</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="./check_room.html">查看预订记录</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/checkroom.html"></a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/checkroom.html"></a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/checkroom.html"></a>
        </li>

        <li class="nav-item dropdown" id="navbardrop5">
            <div class="nav-link dropdown-toggle"  data-toggle="dropdown"
id="item1"></div>
            <div class="dropdown-menu" id="userdrop">
                <div id="item2" class="dropdown-item"></div>
                <div id="item3" class="dropdown-item"></div>
                <div id="item4" class="dropdown-item"></div>
                <div id="item5" class="dropdown-item"></div>
            </div>
        </li>
    </ul>
    <!--        右对齐-->

    <div class="col-3 ">
        <button type="button" class="btn btn-outline-primary "
id="logout_button">退出登录</button>
        <button type="button" class="btn btn-outline-primary " data-
toggle="modal" data-target="#register_btn" id="register_button" >注册</button>
        <button type="button" class="btn btn-primary" data-toggle="modal" data-
target="#login_btn" id="login_button">登录</button>
    </div>
</nav>
```

2.1.2 模态框

```
<div class="modal fade" id="login_btn">
    <div class="modal-dialog">
        <div class="modal-content">

            <!-- 模态框头部 -->
            <div class="modal-header">
                <h4 class="modal-title">登录</h4>
<!--                关闭-->
                <button type="button" class="close" data-dismiss="modal">&times;
</button>
            </div>

            <!-- 模态框主体 -->
            <div class="modal-body">
                <form action="" class="needs-validation" name="form1" >
                    <div class="form-group">
                        <label for="uname">Username:</label>
                        <input type="text" class="form-control" id="uname"
placeholder="输入用户名" name="uname" required>

                        <div id="username-feedback">请输入用户名! </div>
                    </div>
                    <div class="form-group">
                        <label for="pwd">Password:</label>
                        <input type="password" class="form-control" id="pwd"
placeholder="输入密码" name="pwd" required>

                        <div id="password-feedback">请输入密码! </div>
                    </div>
                    <button type="button" class="btn btn-info" data-
dismiss="modal">取消</button>
                    <button type="button" class="btn btn-primary" id="login_1">
提交</button>
                </form>
            </div>
        </div>
    </div>
</div>
```

2.1.3 左侧导航栏

```
<nav class="col-sm-2 col-2" id="myScrollspy">
            <ul class="nav nav-pills flex-column" id="navigation">
                <li class="nav-item">
                    <a class="nav-link active" href="#section1">房间一览</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#section4">KTV图片</a>
                </li>
            </ul>
        </nav>
```

2.1.4 下拉框

```
<li class="nav-item dropdown" id="navbardrop5">
            <div class="nav-link dropdown-toggle"  data-toggle="dropdown"
id="item1"></div>
            <div class="dropdown-menu" id="userdrop">
                <div id="item2" class="dropdown-item"></div>
                <div id="item3" class="dropdown-item"></div>
                <div id="item4" class="dropdown-item"></div>
                <div id="item5" class="dropdown-item"></div>
            </div>
        </li>
```

2.1.5 鼠标悬停状态表格

```
<table class="table table-hover">
        <thead>
        <tr>
            <th>订单号</th>
            <th>房间号</th>
            <th>时间</th>
            <th>状态</th>
        </tr>
        </thead>
        <tbody id="book_form">
        <tr>
        </tr>
        <!--        <tr>-->
        <!--            <td>John</td>-->
        <!--            <td>Doe</td>-->
        <!--            <td>john@example.com</td>-->
        <!--        </tr>-->
        </tbody>
    </table>
```

2.1.6 输入框及按钮组

```
 <div class="input-group mb-3 col-3">
        <input type="text" class="form-control" placeholder="搜索空闲房间"
id="room_input">
        <div class="input-group-append">
            <button class="btn btn-success" type="button" id="room_button">查找
</button>
        </div>
    </div>
```

2.2 js文件功能点

2.2.1 ajax请求

```
$.ajax({
            url: "http://localhost:8888/guest/autologin",//超级链接
            method: "GET",
            error: function (xhr, status, error) {
                console.log(status);
            },
            success: function (result, status, xhr) {//xhr   xmlHttpRequest
                console.log(result);
                //操作DOM渲染页面
                if (result.statusCode == "200") {
                    $("#item1").text("个人信息").toggleClass("text-danger");
                    $("#item2").text("姓
名: "+result.dataZone.guest.name).toggleClass("text-danger");
                    $("#item3").text("手机
号:"+result.dataZone.guest.phone).toggleClass("text-danger");

 $("#item4").text("VIP: "+result.dataZone.guest.vip).toggleClass("text-danger");
```

```
                    $("#item5").text("用户名："+result.dataZone.guest.id).toggleClass("text-danger");
                    $("#function_opt").css("visibility", "visible");
                    $("#navbardrop5").css("visibility", "visible");
                    $("#login_btn").modal('hide');
                    $("#register_btn").modal('hide');
                    $("#register_button").css("visibility", "hidden");
                    $("#login_button").css("visibility", "hidden");
                    $("#logout_button").css("visibility", "visible");


                    alert(result.message);
                } else {
                    alert(result.message);
                }
            }});
```

2.2.2 绘制饼图

```
function init(data_arr, color_arr, text_arr,realdata_arr) {
        //绘制饼图
        //比例数据和颜色
         var data_arr = [0.2, 0.25, 0.45, 0.1];
         var color_arr = ["#00FF21", "#FFAA00", "#00AABB", "#FF4400"];
         var text_arr = ["小小房", "小房", "中房", "大房"];

        drawCircle("canvas_circle", data_arr, color_arr, text_arr);
    }

    function drawCircle(canvasId, data_arr, color_arr, text_arr)
    {
        var c = document.getElementById(canvasId);
        var ctx = c.getContext("2d");

        var radius = c.height / 2 - 20; //半径
        var ox = radius + 20, oy = radius + 20; //圆心

        var width = 30, height = 10; //图例宽和高
        var posX = ox * 2 + 20, posY = 30;   //
        var textX = posX + width + 5, textY = posY + 10;

        var startAngle = 0; //起始弧度
        var endAngle = 0;   //结束弧度
        for (var i = 0; i < data_arr.length; i++)
        {
            //绘制饼图
            endAngle = endAngle + data_arr[i] * Math.PI * 2; //结束弧度
            ctx.fillStyle = color_arr[i];
            ctx.beginPath();
            ctx.moveTo(ox, oy); //移动到到圆心
            ctx.arc(ox, oy, radius, startAngle, endAngle, false);
            ctx.closePath();
            ctx.fill();
            startAngle = endAngle; //设置起始弧度
            //绘制比例图及文字
            ctx.fillStyle = color_arr[i];
            ctx.fillRect(posX, posY + 20 * i, width, height);
            ctx.moveTo(posX, posY + 20 * i);
            ctx.font = 'bold 12px 微软雅黑';    //斜体 30像素 微软雅黑字体
            ctx.fillStyle = color_arr[i]; //"#000000";
            var percent = text_arr[i] + "： " +data_arr[i]*100 +"%";
            ctx.fillText(percent, textX, textY + 20 * i);
        }
    }
```

2.2.3 模态框登录及输入检测

```
function checkinformation() {
        var user_name=form1.uname.value;
            var user_pwd=form1.pwd.value;
            if((user_name=="")||(user_name==null)){
                $("#username-feedback").css("visibility", "visible");
                return false;
            }
            else if((user_pwd=="")||(user_pwd==null)){
                $("#password-feedback").css("visibility", "visible");
                return false;}
            else {
                return true;
            }
        }

    var login_bottom=document.querySelector("#login_1");
    login_bottom.onclick=function login() {
        if(checkinformation()){
            let username=form1.uname.value;
            let password=form1.pwd.value;
            $.ajax({
                url: " http://localhost:8888/guest/get",//普通登录
                method: "GET",

                data:{username:username,password:password},
                error: function (xhr, status, error) {
                    console.log(status);
                },
                success: function (result, status, xhr) {//xhr   xmlHttpRequest
                    console.log(result);
                    //操作DOM渲染页面
                    if (result.statusCode == "200") {
                        $("#item1").text("个人信息").toggleClass("text-danger");
                        $("#item2").text("姓
名： "+result.dataZone.guest.name).toggleClass("text-danger");
                        $("#item3").text("手机
号:"+result.dataZone.guest.phone).toggleClass("text-danger");

 $("#item4").text("VIP： "+result.dataZone.guest.vip).toggleClass("text-danger");
                        $("#item5").text("用户
名： "+result.dataZone.guest.id).toggleClass("text-danger");
```

```
                        $("#function_opt").css("visibility", "visible");
                        $("#navbardrop5").css("visibility", "visible");
                        $("#login_btn").modal('hide');
                        $("#register_btn").modal('hide');
                        $("#register_button").css("visibility", "hidden");
                        $("#login_button").css("visibility", "hidden");
                        $("#logout_button").css("visibility", "visible");
                        alert(result.message);
                    } else {
                        alert(result.message);
                    }
                },
            });
        }
        ;
    }
```

2.2.4 动态生成预约表格的内容，包括获取所有的房间信息

```
var room_bottom=document.querySelector("#room_button");
    room_bottom.onclick=function(){
        let text=document.getElementById("room_input").value;
        console.log(text);
        $.ajax({
            url: " http://localhost:8888/room/all",//普通登录
            method: "GET",

            data:{search:text},
            error: function (xhr, status, error) {
                console.log(status);
            },
            success: function (result, status, xhr) {//xhr   xmlHttpRequest
                console.log(result);
                //操作DOM渲染页面
                if (result.statusCode == "200") {
                    $("#book_form").html("");
                    var otbody=document.getElementsByTagName("tbody")[0];
                    var roomlist=result.dataZone.roomlist;
                    console.log(roomlist[0]);
                   for(room in roomlist){
                       var otr = document.createElement("tr");
                       var otd1= document.createElement("td");
                       var otd2= document.createElement("td");
                       var otd3= document.createElement("td");
                       var otd4= document.createElement("button");
                       otd4.setAttribute("class","btn btn-primary");

otd4.setAttribute("onclick","reserve('"+result.dataZone.roomlist[room].id+"')");
                       otd4.setAttribute("data-toggle","modal");
                       otd4.setAttribute("data-target","#reserve_btn")
                       otd1.innerText=result.dataZone.roomlist[room].id;
                       otd2.innerText=result.dataZone.roomlist[room].rsize;
                       otd3.innerText=result.dataZone.roomlist[room].price;
                       otd4.innerText="预约";
                       otr.appendChild(otd1);
                    otr.appendChild(otd2);
                    otr.appendChild(otd3);
                    otr.appendChild(otd4);
                    otbody.appendChild(otr);
                }

                 alert(result.message);
              } else {
                 alert(result.message);
              }
           },
       });

   }
```

2.2.5 动态获取对应房间的空闲时间段

```
function reserve(number){
    console.log(number);
    $.ajax({
        url: "http://localhost:8888/reserve/free",//查找空闲时间
        method: "GET",
        data: {room_id:number},
        error: function (xhr, status, error) {
            console.log(status);
        },
        success: function (result, status, xhr) {//xhr   xmlHttpRequest
            console.log(result);
            //操作DOM渲染页面
            if (result.statusCode == "200") {
                document.getElementById("room_id").innerText=number;
                var freetime=result.dataZone.free;
                console.log(freetime);
                var buttons=document.getElementsByClassName("form-check-input");
                // buttons[0].setAttribute("disabled","true");
                // buttons[1].setAttribute("disabled","true");
                for(free in freetime){
                    buttons[free].removeAttribute("disabled");
                    buttons[free].checked=false;
                    if(freetime[free]=='1'){
                        buttons[free].setAttribute("disabled","true");
                    }
                }
                alert(result.message);
            } else if (result.statusCode == "400") {
                alert(result.message);
            }
        },
    });
}
```

2.3 CSS文件功能点--各选择器

```
div.col-8 div:nth-child(1){
    background: #c1ffba;
}
#navigation li{
    margin-top: 2em;
}
```

三、后端功能点

3.1 Service接口类

```
public interface GuestService {
    public Guest_infoWithBLOBs getGuest(String guest_id);
    public void insertGuest(Guest_infoWithBLOBs guest);

}
```

3.2 ServiceImpl 实现类

3.2.1 GuestServiceImpl类

```
@Service(value="guest_infoService")
public class GuestServiceImpl implements GuestService {
    @Autowired
    private Guest_infoMapper guest_infoMapper;

    @Override
    public Guest_infoWithBLOBs getGuest(String guest_id) {

        return guest_infoMapper.selectByPrimaryKey(guest_id);
    }

    @Override
    public void insertGuest(Guest_infoWithBLOBs guest_info) {
          this.guest_infoMapper.insert(guest_info);
    }

}
```

3.2.2 ReserveServiceImpl类

```
@Service(value="reserve_infoService")
public class ReserveServiceImpl implements ReserveService {

    @Autowired
    private Reserve_infoMapper reserve_infoMapper;
    @Override
    public List<Reserve_info> findReserve(String room_id) {
        Reserve_infoExample reserve_infoExample=new
Reserve_infoExample(Reserve_info.class);
```

```
        Reserve_infoExample.Criteria
criteria=reserve_infoExample.createCriteria();
        criteria.andRIdEqualTo(room_id);
        List<Reserve_info> reserve_infoList =
reserve_infoMapper.selectByExample(reserve_infoExample);
        return reserve_infoList;
    }

    @Override
    public void insertReserve(Reserve_info reserve_info) {
     this.reserve_infoMapper.insert(reserve_info);
    }

    @Override
    public List<Reserve_info> getReserve(String guest_id) {
        Reserve_infoExample reserve_infoExample=new
Reserve_infoExample(Reserve_info.class);
        Reserve_infoExample.Criteria
criteria=reserve_infoExample.createCriteria();
        criteria.andGIdEqualTo(guest_id);
        List<Reserve_info> reserve_infoList =
reserve_infoMapper.selectByExample(reserve_infoExample);
        return reserve_infoList;
    }

    @Override
    public void deleteReserve(String id) {
        System.out.println(id);
        this.reserve_infoMapper.deleteByPrimaryKey(id);
    }
}
```

3.2.3 RoomServiceImpl类

```
@Service(value="room_infoService")
public class RoomServiceImpl implements RoomService {

    @Autowired
    private Room_infoMapper room_infoMapper;

    @Override
    public List<Room_info> getAllRoom() {
        return room_infoMapper.selectByExample(null);
    }

    @Override
    public Room_info getRoom(String room_id) {
        return room_infoMapper.selectByPrimaryKey(room_id);
    }
}
```

3.3 Controller类——后端功能点说明

3.3.1 客户登录

```
 @RequestMapping(value = "/get",method = RequestMethod.GET)
    public MessageAndData qetguestbyId(String username, String
password,HttpSession session){
        Guest_info guest_info=guestService.getGuest(username);
        System.out.println(username);
        System.out.println(guest_info);
        MessageAndData messageAndData;
        if(guest_info==null){
            messageAndData=MessageAndData.error();
            messageAndData.setMessage("未查到用户");
        }
        if(guest_info.getPassword().equals(password)) {
            messageAndData=MessageAndData.success();
            messageAndData.add("guest",guest_info).setMessage("密码正确");
            guest=guest_info;
        }
        else {
            messageAndData = MessageAndData.error();
            messageAndData.setMessage("查找到用户，密码错误");
        }

        return messageAndData;
    }
```

3.3.2 已登录后自动登录

```
@RequestMapping(value = "/autologin",method = RequestMethod.GET)
    public MessageAndData autologin(HttpSession session){
        MessageAndData messageAndData;
        System.out.println(guest);
        if (guest == null){
            messageAndData=MessageAndData.error();
            messageAndData.setMessage("您还未登录");
        }else{
            messageAndData = MessageAndData.success();
            messageAndData.add("guest",guest);
        }
        return messageAndData;
    }
```

3.3.3 登出

```
 @RequestMapping(value = "/logout",method = RequestMethod.GET)
    public MessageAndData logoutguest() {
        MessageAndData messageAndData;
        if (guest == null){
            messageAndData=MessageAndData.error();
            messageAndData.setMessage("您还未登录");
        }else{
            messageAndData = MessageAndData.success();
            messageAndData.setMessage("成功退出");
            guest=null;
        }
        return messageAndData;
    }
```

3.3.4 查询客户已预约订单信息

```
 @RequestMapping(value = "/all",method = RequestMethod.GET)
    public MessageAndData all() {
        MessageAndData messageAndData;
        System.out.println(guest);
        if (guest == null){
            messageAndData=MessageAndData.error();
            messageAndData.setMessage("您还未登录");
        }else{
            messageAndData = MessageAndData.success();
            messageAndData.add("guest",guest);
            List<Reserve_info>
reserve_list=reserveService.getReserve(guest.getId());
            messageAndData.add("reservelist",reserve_list).setMessage("查询客户预
约单成功");
        }
        return messageAndData;
    }
```

3.3.5 查询房间空闲时间

```
 @RequestMapping(value = "/free",method = RequestMethod.GET)
    public MessageAndData qetreservebyRId(String room_id){
        MessageAndData messageAndData;
        int[] freetime = {0, 0, 0, 0, 0, 0};
        List<Reserve_info> reserve_list=reserveService.findReserve(room_id);
        if(reserve_list!=null) {
            System.out.println(reserve_list);
            for (int i = 0; i < reserve_list.size(); i++) {
                Reserve_info reserve_info = reserve_list.get(i);
                System.out.println("预约数字串:" + reserve_info.getReserveFlag());
                String reservetime = reserve_info.getReserveFlag().toString();
                for (int j = 1; j < reservetime.length(); j++) {
                    if (reservetime.charAt(j) == '1') {
                        freetime[j - 1] = 1;
                    }
                }
            }
        }
```

3.3.6 预约信息插入

```
@RequestMapping(value = "/insert",method = RequestMethod.GET)
    public MessageAndData insert(String guest_id,String room_id,String 
reserve_flag){
        MessageAndData messageAndData;
        System.out.println(guest_id);
        System.out.println(room_id);
        System.out.println(reserve_flag);
        String id= String.valueOf(System.currentTimeMillis());
        String time= String.valueOf(LocalDateTime.now());
        System.out.println(id);
        System.out.println(time);

        Reserve_info reserve_info=new Reserve_info();
        reserve_info.setgId(guest_id);
        BigDecimal bd=new BigDecimal(reserve_flag);
        reserve_info.setReserveFlag(bd);
        reserve_info.setrId(room_id);
        reserve_info.setTime(time);
        reserve_info.setId(id);
        reserveService.insertReserve(reserve_info);
        messageAndData = MessageAndData.success();
        messageAndData.setMessage("预约成功");
        return messageAndData;
    }
```

3.3.7 删除预约记录

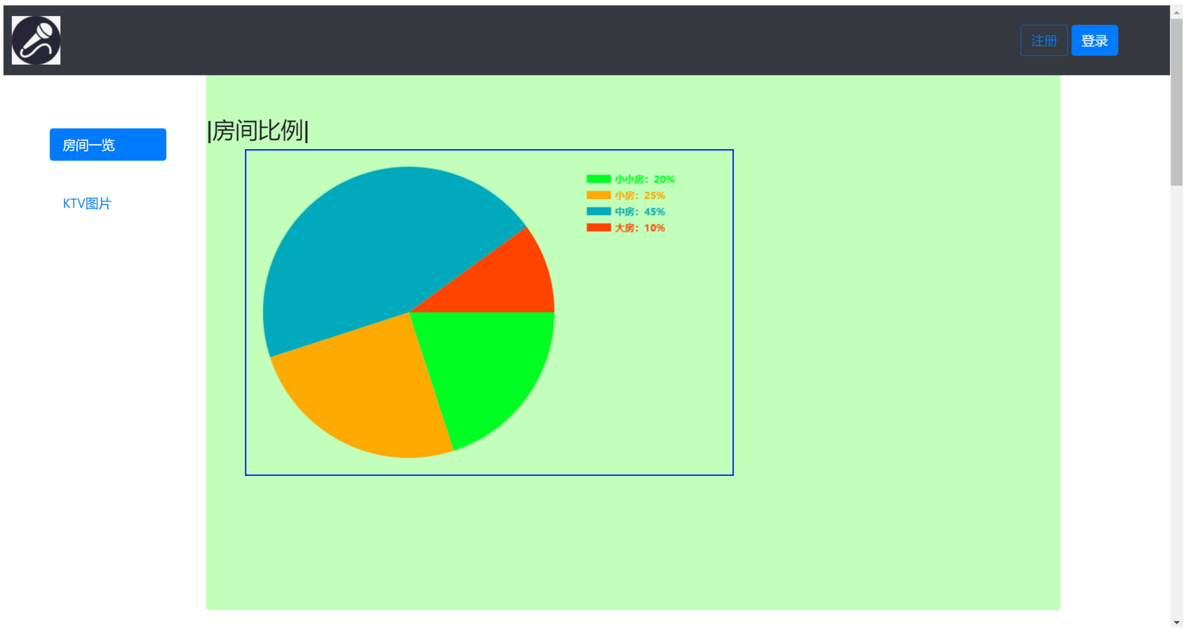
```
@RequestMapping(value = "/delete",method = RequestMethod.GET)
    public MessageAndData insert(String id){
        MessageAndData messageAndData;
        reserveService.deleteReserve(id);

        messageAndData = MessageAndData.success();
        messageAndData.setMessage("删除成功");
        return messageAndData;
    }
```

3.3.8 获取所有房间信息

```
@RequestMapping(value = "/all",method = RequestMethod.GET)
    public MessageAndData getallroom(String search){
        List<Room_info> room_list= roomService.getAllRoom();
        System.out.println(room_list);
        MessageAndData messageAndData;
        if(search=="")
        {
            if (room_list == null) {
                messageAndData = MessageAndData.error();
                messageAndData.setMessage("未查到房间");
            } else {
                messageAndData = MessageAndData.success();
                messageAndData.add("roomlist", room_list).setMessage("获取房间列表成功");
            }
        }
        else{
            Room_info room=roomService.getRoom(search);
            if(room==null){
                messageAndData=MessageAndData.error();
                messageAndData.setMessage("未查到房间");
            }
            else{
                List<Room_info> roomList=new ArrayList<>();
                roomList.add(room);
                messageAndData=MessageAndData.success();
                messageAndData.add("roomlist",roomList).setMessage("找到房间");
            }
        }
        return messageAndData;
    }
}
```

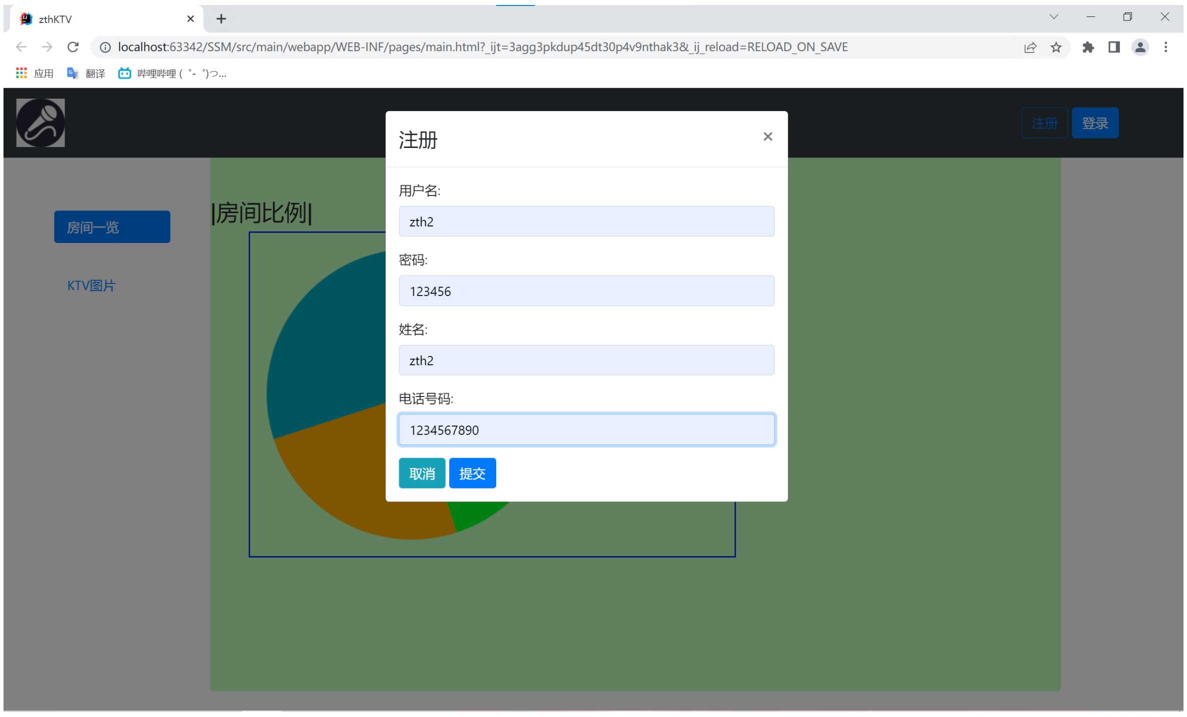
四、页面设计及使用说明

4.1 起始页面



4.2 注册

使用zth2作为演示


zthKTV

localhost:63342/SSM/src/main/webapp/WEB-INF/pages/main.html?_ijt=3agg3pkdup45dt30p4v9nthak3&_ij_reload=RELOAD_ON_SAVE

注册

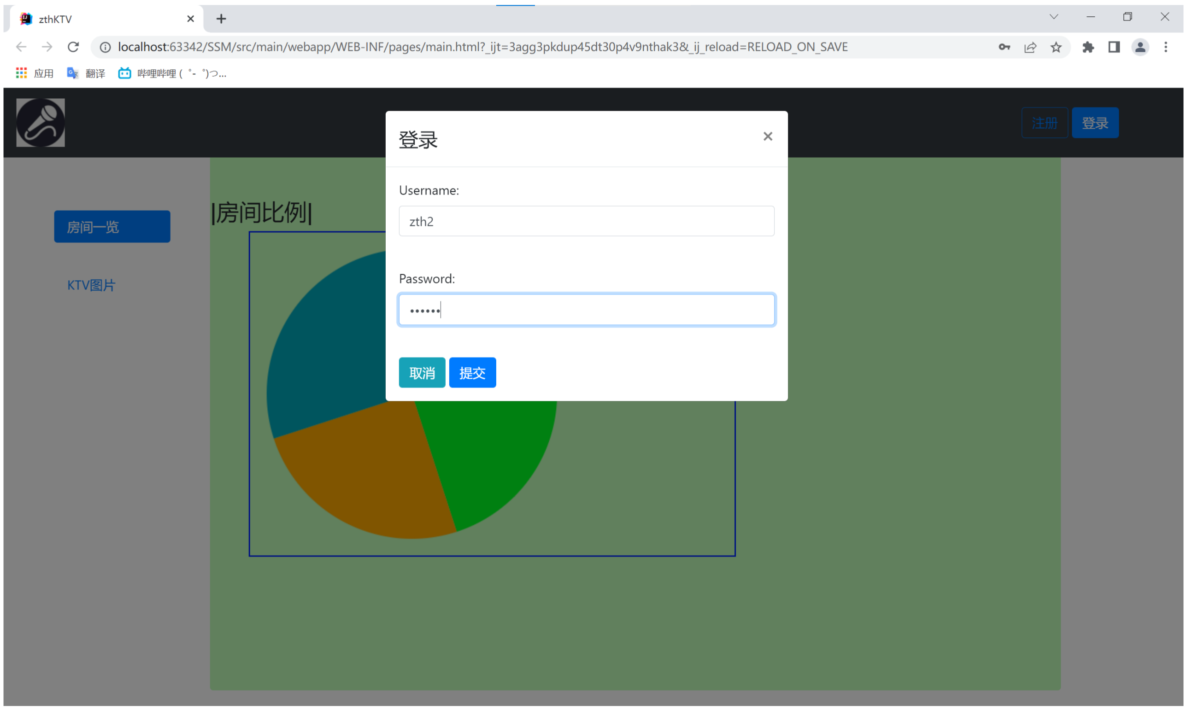
用户名: zth2

密码: 123456

姓名: zth2

电话号码: 1234567890

4.3 登录


zthKTV

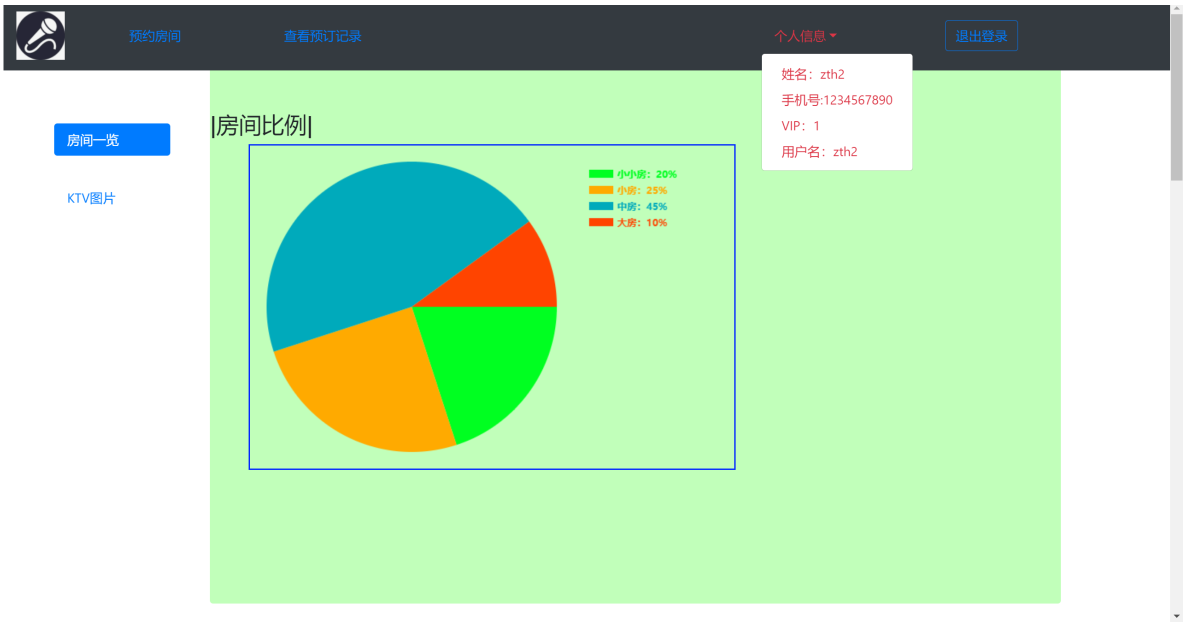
localhost:63342/SSM/src/main/webapp/WEB-INF/pages/main.html?_ijt=3agg3pkdup45dt30p4v9nthak3&_ij_reload=RELOAD_ON_SAVE

登录

Username: zth2

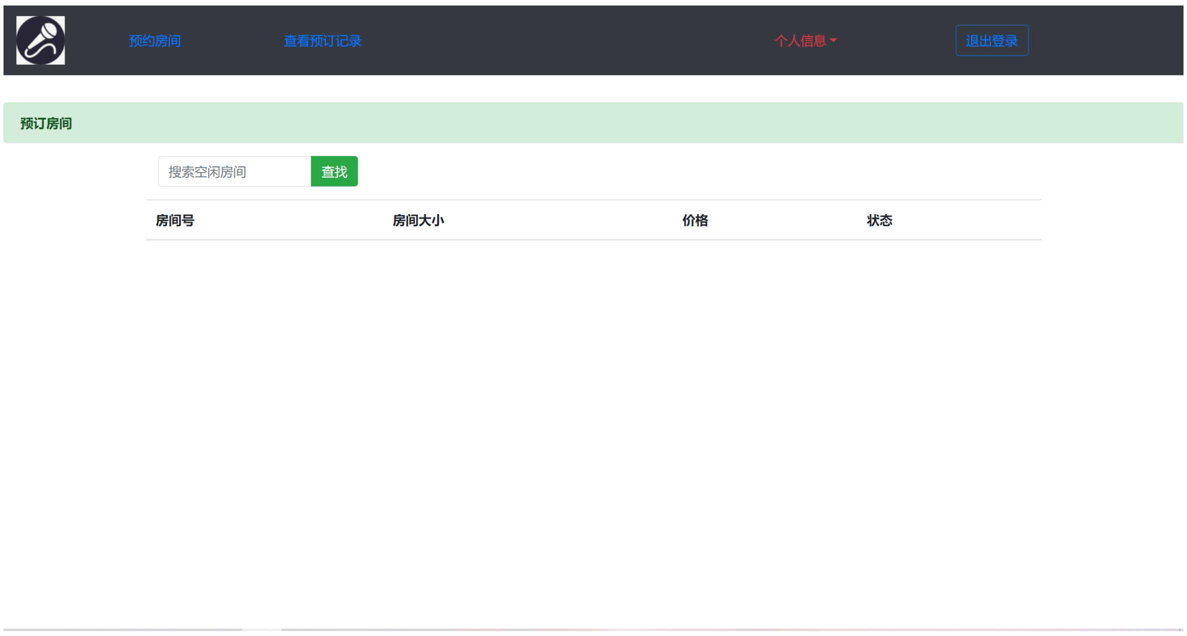
Password: ••••••

4.4 主页面



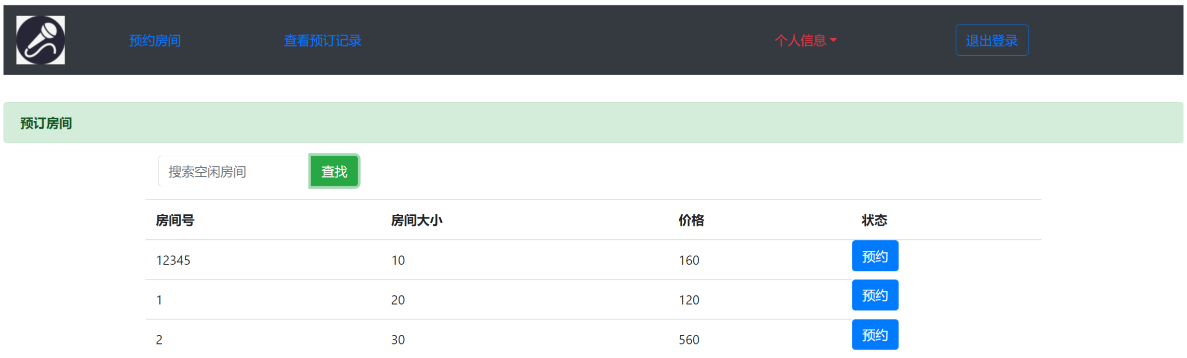
姓名：zth2
手机号:1234567890
VIP：1
用户名：zth2

4.5 预约房间主页面



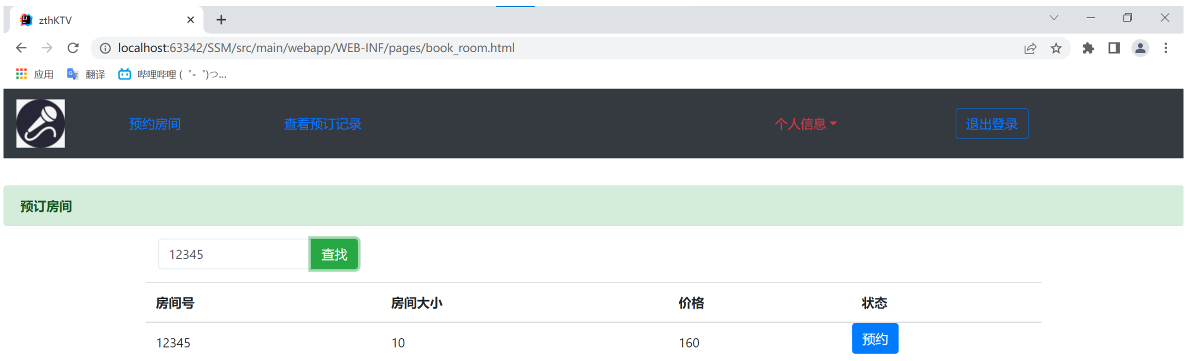
房间号	房间大小	价格	状态

4.6 空白查找房间



房间号	房间大小	价格	状态
12345	10	160	预约
1	20	120	预约
2	30	560	预约

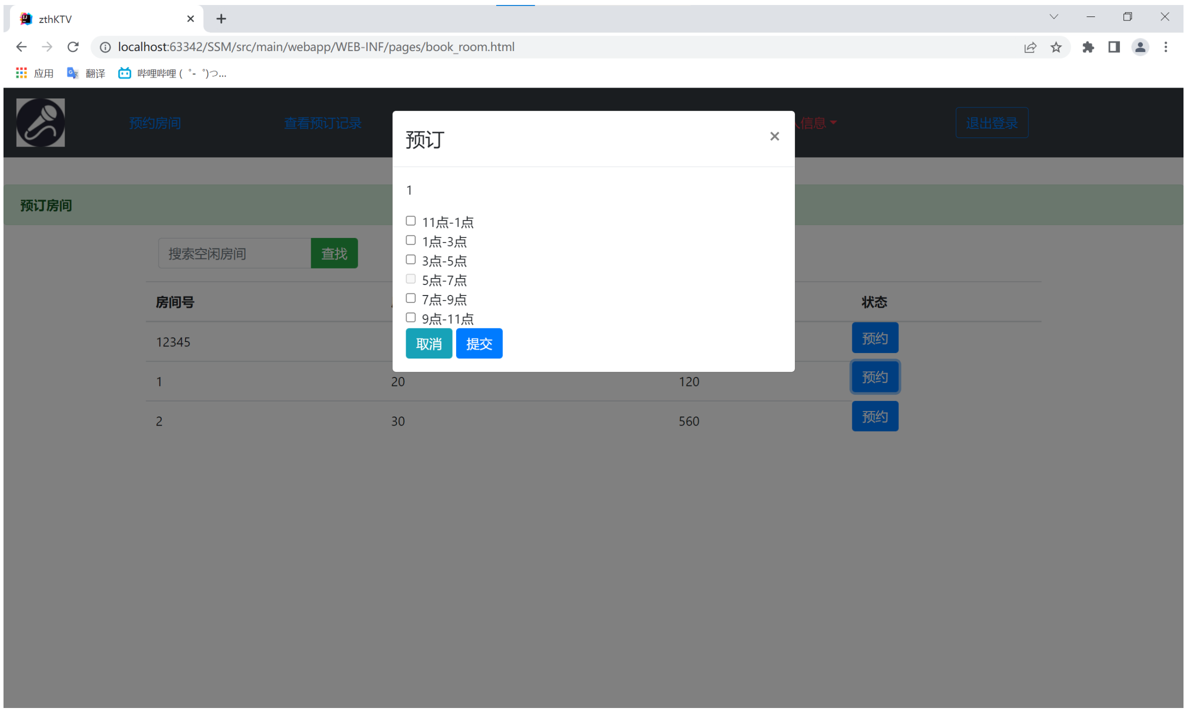
4.7 输入查找房间



房间号	房间大小	价格	状态
12345	10	160	预约

4.8 预约房间

无法选择的复选框代表已经被人占用。



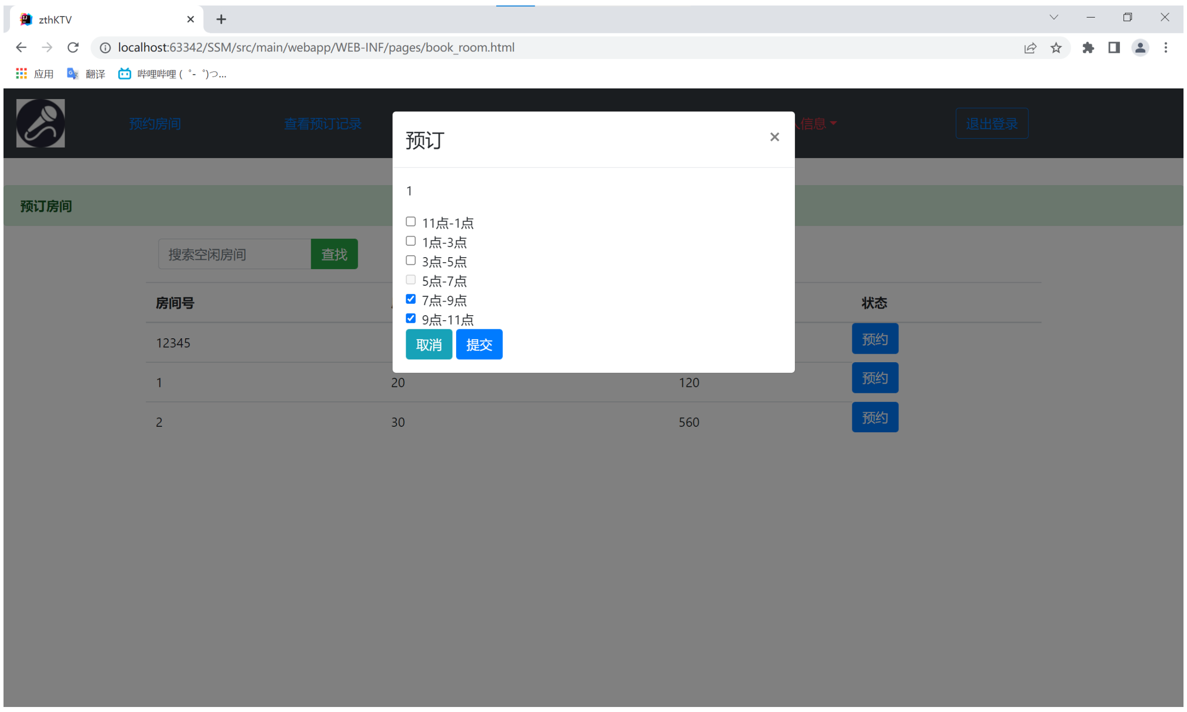
预订

1

- [ ] 11点-1点
- [ ] 1点-3点
- [ ] 3点-5点
- [ ] 5点-7点
- [ ] 7点-9点
- [ ] 9点-11点

房间号			状态
12345			预约
1	20	120	预约
2	30	560	预约

选择1号房间7-11点的框



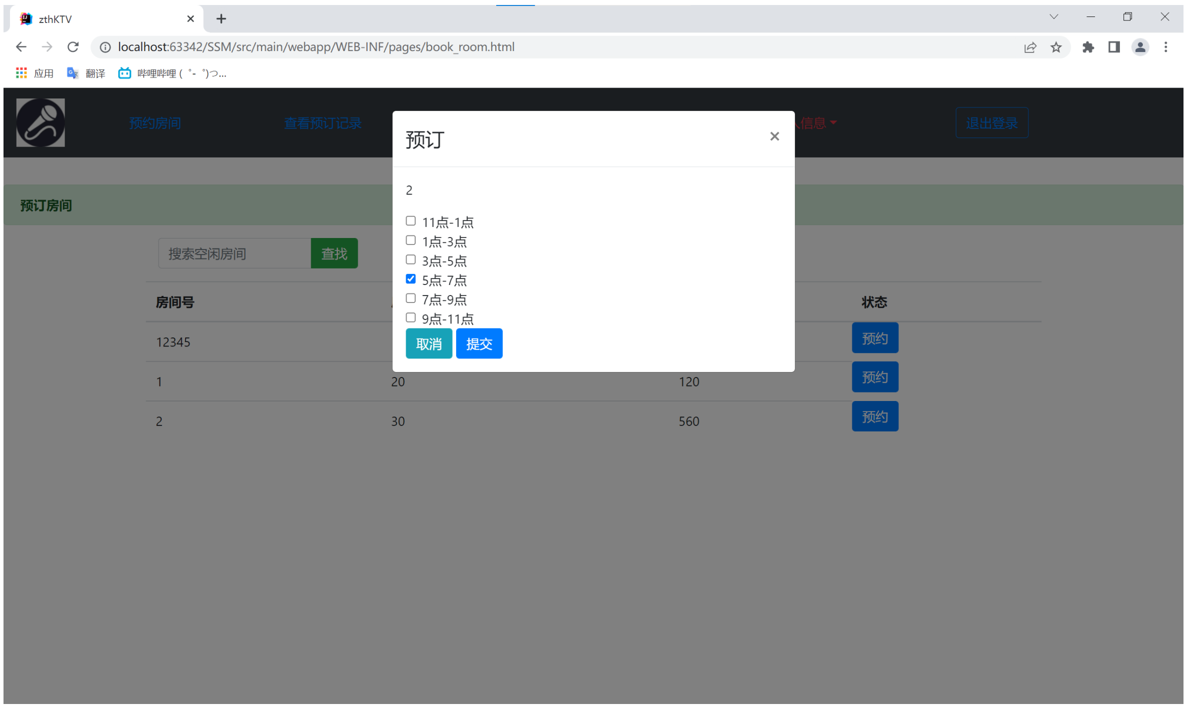
预订

1

- [ ] 11点-1点
- [ ] 1点-3点
- [ ] 3点-5点
- [ ] 5点-7点
- [x] 7点-9点
- [x] 9点-11点

房间号			状态
12345			预约
1	20	120	预约
2	30	560	预约

再选择2号房间5-7点的框


zthKTV
localhost:63342/SSM/src/main/webapp/WEB-INF/pages/book_room.html
预约房间
查看预订记录
退出登录
预订房间
搜索空闲房间
查找

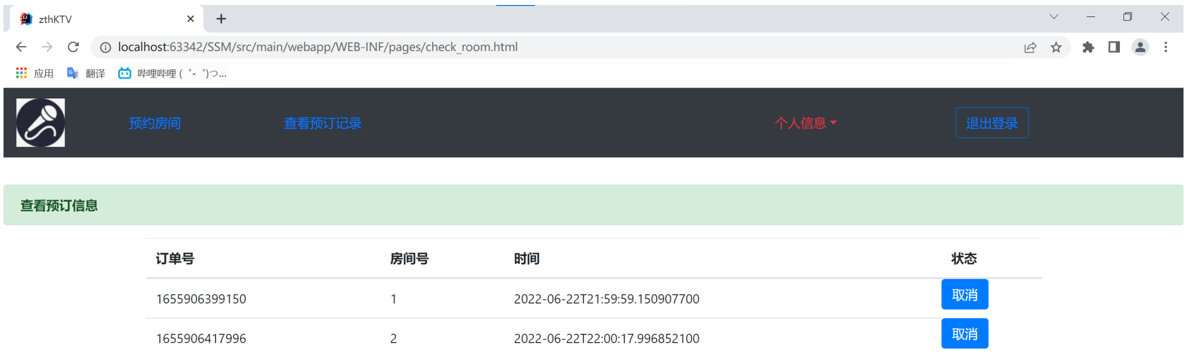
预订

2

- [ ] 11点-1点
- [ ] 1点-3点
- [ ] 3点-5点
- [x] 5点-7点
- [ ] 7点-9点
- [ ] 9点-11点

房间号			状态
12345			预约
1	20	120	预约
2	30	560	预约

4.9 查看预约信息

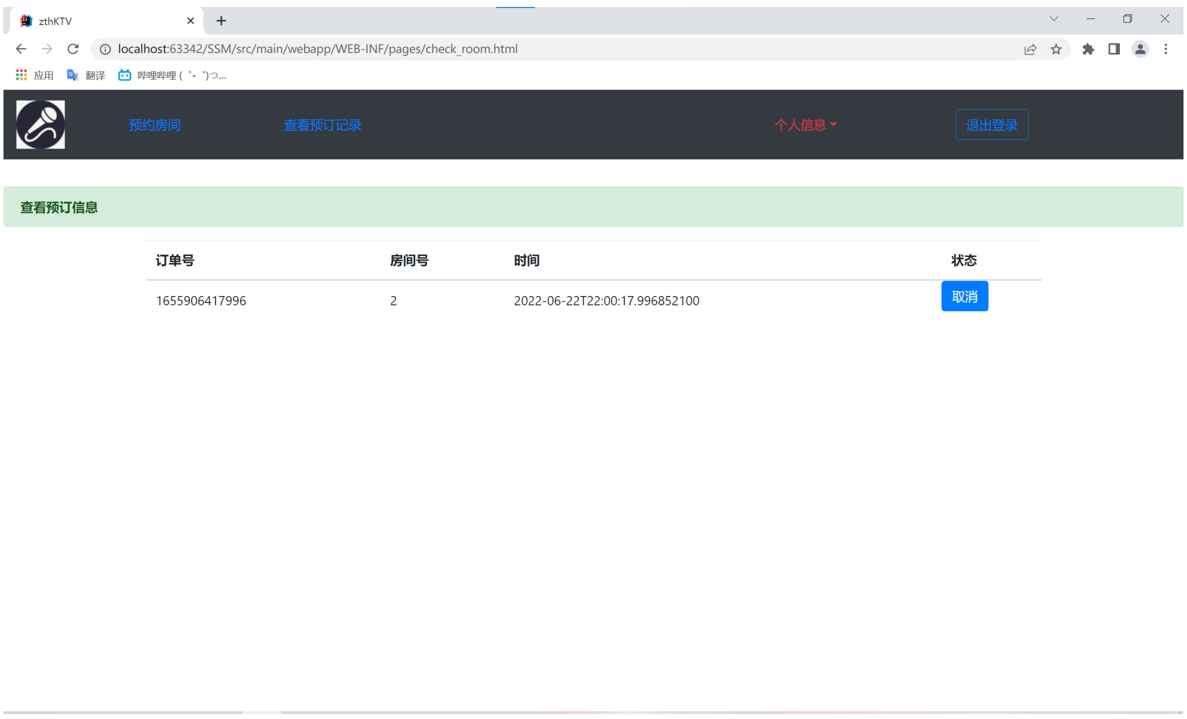

zthKTV
localhost:63342/SSM/src/main/webapp/WEB-INF/pages/check_room.html
预约房间
查看预订记录
个人信息
退出登录

查看预订信息

订单号	房间号	时间	状态
1655906399150	1	2022-06-22T21:59:59.150907700	取消
1655906417996	2	2022-06-22T22:00:17.996852100	取消

4.10 进行预约取消

取消1号房间的预约



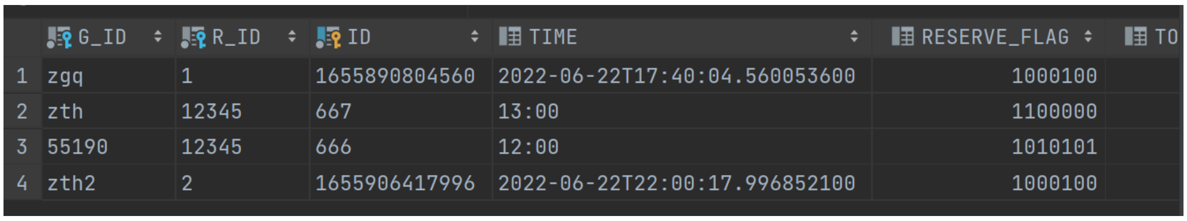
zthKTV

localhost:63342/SSM/src/main/webapp/WEB-INF/pages/check_room.html

查看预订信息

订单号	房间号	时间	状态
1655906417996	2	2022-06-22T22:00:17.996852100	取消

4.11 查看数据库中的2号房间预约



	G_ID	R_ID	ID	TIME	RESERVE_FLAG	TO
1	zgq	1	1655890804560	2022-06-22T17:40:04.560053600	1000100	
2	zth	12345	667	13:00	1100000	
3	55190	12345	666	12:00	1010101	
4	zth2	2	1655906417996	2022-06-22T22:00:17.996852100	1000100	

所有功能均正常运行，演示完毕。