**电商Web系统部分项目指导说明书**

目录

[任务1. Web端系统架构设计 5](#_Toc44266106)

[任务描述 5](#_Toc44266107)

[任务指导 5](#_Toc44266108)

[任务实现 8](#_Toc44266109)

[任务2. 图书推荐系统首页 - 布局 22](#_Toc44266110)

[任务描述 22](#_Toc44266111)

[授课思路 22](#_Toc44266112)

[任务指导 23](#_Toc44266113)

[任务实现 23](#_Toc44266114)

[任务3. 图书推荐系统首页 - 登录 27](#_Toc44266115)

[任务描述 27](#_Toc44266116)

[授课思路 28](#_Toc44266117)

[任务指导 28](#_Toc44266118)

[任务实现 28](#_Toc44266119)

[任务4. 图书推荐系统首页 - 注册 31](#_Toc44266120)

[任务描述 31](#_Toc44266121)

[授课思路 31](#_Toc44266122)

[任务指导 31](#_Toc44266123)

[任务实现 31](#_Toc44266124)

[任务5. 图书详情页面开发 34](#_Toc44266125)

[任务描述 34](#_Toc44266126)

[授课思路 34](#_Toc44266127)

[任务指导 34](#_Toc44266128)

[任务实现 34](#_Toc44266129)

[任务6. 我的购物车页面开发 37](#_Toc44266130)

[任务描述 37](#_Toc44266131)

[授课思路 37](#_Toc44266132)

[任务指导 37](#_Toc44266133)

[任务实现 38](#_Toc44266134)

[任务7. 订单支付功能完成 40](#_Toc44266135)

[任务描述 40](#_Toc44266136)

[授课思路 41](#_Toc44266137)

[任务指导 41](#_Toc44266138)

[任务实现 41](#_Toc44266139)

[任务8. 订单列表功能页面开发 44](#_Toc44266140)

[任务描述 44](#_Toc44266141)

[授课思路 44](#_Toc44266142)

[任务指导 44](#_Toc44266143)

[任务实现 45](#_Toc44266144)

[任务9. 权限控制之shiro学习 46](#_Toc44266145)

[任务描述 46](#_Toc44266146)

[授课思路 47](#_Toc44266147)

[任务指导 47](#_Toc44266148)

[任务实现 47](#_Toc44266149)

[任务10. 用户权限管理设计思路 51](#_Toc44266150)

[任务描述 51](#_Toc44266151)

[授课思路 51](#_Toc44266152)

[任务指导 51](#_Toc44266153)

[任务实现 51](#_Toc44266154)

[任务11. 用户管理 - 用户列表 52](#_Toc44266155)

[任务描述 52](#_Toc44266156)

[授课思路 52](#_Toc44266157)

[任务指导 52](#_Toc44266158)

[任务实现 53](#_Toc44266159)

[任务12. 用户管理 - 添加用户 55](#_Toc44266160)

[任务描述 55](#_Toc44266161)

[授课思路 55](#_Toc44266162)

[任务指导 56](#_Toc44266163)

[任务实现 56](#_Toc44266164)

[任务13. 用户管理 - 用户编辑及删除 61](#_Toc44266165)

[任务描述 61](#_Toc44266166)

[授课思路 62](#_Toc44266167)

[任务指导 62](#_Toc44266168)

[任务实现 62](#_Toc44266169)

[任务14. 云店铺管理 - 云店铺列表及添加 66](#_Toc44266170)

[任务描述 66](#_Toc44266171)

[授课思路 66](#_Toc44266172)

[任务指导 66](#_Toc44266173)

[任务实现 66](#_Toc44266174)

[任务15. 云店铺管理 - 云店铺编辑及删除 71](#_Toc44266175)

[任务描述 71](#_Toc44266176)

[授课思路 71](#_Toc44266177)

[任务指导 71](#_Toc44266178)

[任务实现 71](#_Toc44266179)

[任务16. 角色权限管理 - 角色列表及编辑与删除 75](#_Toc44266180)

[任务描述 75](#_Toc44266181)

[授课思路 75](#_Toc44266182)

[任务指导 75](#_Toc44266183)

[任务实现 75](#_Toc44266184)

[任务17. 角色权限管理 - 角色添加 79](#_Toc44266185)

[任务描述 79](#_Toc44266186)

[授课思路 80](#_Toc44266187)

[任务指导 80](#_Toc44266188)

[任务实现 80](#_Toc44266189)

[任务18. 角色权限管理 - 权限列表 82](#_Toc44266190)

[任务描述 82](#_Toc44266191)

[授课思路 82](#_Toc44266192)

[任务指导 82](#_Toc44266193)

[任务实现 82](#_Toc44266194)

[任务19. 我的店铺 - 店铺信息查看 84](#_Toc44266195)

[任务描述 84](#_Toc44266196)

[授课思路 84](#_Toc44266197)

[任务指导 85](#_Toc44266198)

[任务实现 85](#_Toc44266199)

[任务20. 我的店铺 - 图书列表及编辑 87](#_Toc44266200)

[任务描述 87](#_Toc44266201)

[授课思路 88](#_Toc44266202)

[任务指导 88](#_Toc44266203)

[任务实现 88](#_Toc44266204)

[任务21. 我的店铺 - 图书添加 90](#_Toc44266205)

[任务描述 90](#_Toc44266206)

[授课思路 91](#_Toc44266207)

[任务指导 91](#_Toc44266208)

[任务实现 91](#_Toc44266209)

[任务22. 订单管理 - 查看订单 94](#_Toc44266210)

[任务描述 94](#_Toc44266211)

[授课思路 94](#_Toc44266212)

[任务指导 94](#_Toc44266213)

[任务实现 95](#_Toc44266214)

[任务23. 个人中心管理 - 个人信息编辑 97](#_Toc44266215)

[任务描述 97](#_Toc44266216)

[授课思路 97](#_Toc44266217)

[任务指导 97](#_Toc44266218)

[任务实现 97](#_Toc44266219)

[任务24. 前端—个性化展示（词云、聚类、热点图、Top20展示） 101](#_Toc44266220)

[任务描述 101](#_Toc44266221)

[任务指导 102](#_Toc44266222)

[任务实现 102](#_Toc44266223)

[任务25. 项目测试、整合、发布 102](#_Toc44266224)

[任务描述 102](#_Toc44266225)

[授课思路 103](#_Toc44266226)

[任务指导 103](#_Toc44266227)

[任务实现 103](#_Toc44266228)

# Web端系统架构设计

## 任务描述

**内 容**：完成Web端系统架构设计SpringBoot+Vue

**学 时**：2学时

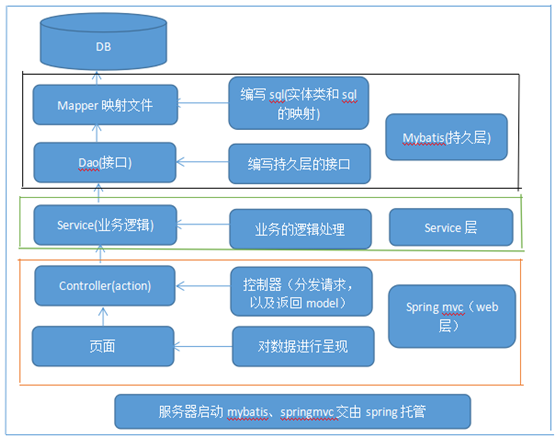
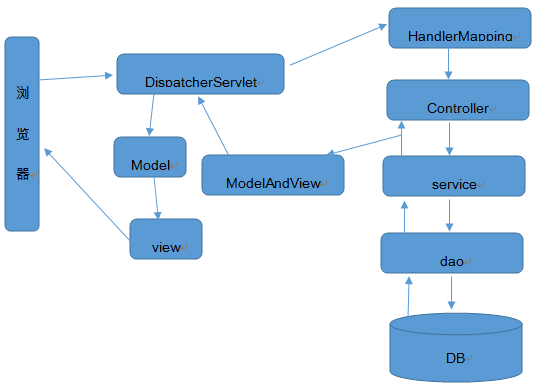
**知识点**：SpringBoot、Spring、Sparing MVC、MyBatis、JQuery、Vue 、MySql

**重点**：Spring、Sparing MVC、MyBatis、JQuery、Vue 、MySql

**难点**：SpringBoot + Vue 实现前后端分离并支持跨域

## 任务指导

1. 搭建Web端系统架构



1. 演示一个基于SpringBoot SSM + Vue的Demo示例

输入http://localhost:8080回车浏览器自动跳转到http://localhost:8080/login登录页面，登录页面包含页面头部、登录信息页面、页面尾部。输入用户名、密码点击登录功能实现效果如下：

* + 输入服务器不正确的用户名密码，如输入用户名zhangwuji登录，会在登录验证情况文本框中显示后端端验证后的错误信息。
  + 输入正确的用户名密码，如输入用户名zhangsanfeng密码123456登录，则服务端验证正确后前端页面自动跳转到首页<http://localhost:8080/index>。

1. DEMO技术栈描述
   * 前端技术栈
     + 编程语言：html5、js、css
     + 开发工具：IDEA或Visual Studio Code
     + 开发框架：vue + axios
     + 包管理工具:npm
     + 打包工具：webpack
   * 后端技术栈
     + 编程语言：java
     + 开发工具：IDEA
     + 开发框架：spring boot
     + 包管理工具: maven资源库
     + 打包工具：gradle
2. 开发流程和步骤
   * 前端开发流程
     + 安装nodejs并初始化Vue项目
     + 在Vue项目中的开发页面头、页面尾等公共组件
     + 开发登录页面组件
     + 开发首页页面组件
     + 支持跨域，请求路由，页面路由开发
     + 单独运行Vue项目查看效果
   * 后端开发流程
     + 初始化springboot项目
     + 开发restful控制器
     + 支持跨域
     + 单独运行后端springboot项目查看效果
   * 运行部署项目
     + 使用webpack将Vue项目打包。
     + 将打包的Vue项目集成到springboot项目中。
     + 使用gradle将springboot打包成jar文件。
     + 使用jdk运行jar包来启动demo项目服务，请访问地址查看效果。
     + 前端项目由于启用了eslint语法检测，所以有时候多个空格或者少个空格或者少个空行，都会运行不起来前端项目，对应提示信息改下即可。
     + 前端发送请求的数据格式需要与后端接收请求数据对象格式要约定一致。
     + 在前后端未集成的时候需要跨域支持。

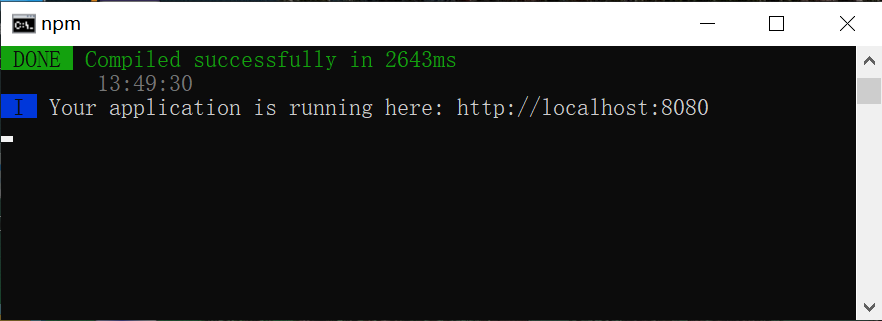
## 任务实现

1. **前端开发流程**
   * 安装nodejs并初始化Vue项目

* 进入Node.js官网：https://nodejs.org/en/，选择下载并安装Node.js。
* 在cmd命令行窗口，输入node -v查看Node.js版本。
* 由于Node.js自带了npm，所以输入npm -v 可以查看到npm的版本。
* 输入npm -g install npm更新npm至最新版本。
* 执行命令 npm install -g cnpm --registry=https://registry.npm.taobao.org （使用国内镜像）。cnpm 命令代替默认的npm命令，增加依赖包加载速度且避免资源限制。
* 使用cnpm安装vue-cli脚手架：使用cnpm install -g vue-cli命令。
* 构建Vue项目：创建一个目录并使用cd命令进入该目录，再输入vue init webpack bx\_web\_vue命令，生成Vue项目。



* 进入Vue项目根目录，使用cnpm install命令，安装项目所需的依赖包资源。（有时会遇到奇怪的报错，然后通过先执行cnpm rebuild node-sass，后执行cnpm install解决，此步骤不是必须的）。
* 在Vue项目根目录下运行npm run dev命令来运行项目。



* 在浏览器中可以使用<http://localhost:8080>查看效果。
* 经过以上步骤完成了搭建vue的脚手架项目，可以简单看下项目目录，后续我们开发项目的时候只需在src目录下进行。
* Visual Studio Code工具作为前端开发工具，真心好用，开发工具下载官网地址：https://code.visualstudio.com/
  + 在Vue项目中的开发页面头、页面尾等公共组件
* 开发页面头部的公共组件，在Vue项目的src/components目录下新建common/BaseHeader.vue文件

<template>

<div>

页面头部

</div>

</template>

<script>

export default {

name: 'BlogHeader'

}

</script>

* 开发页面尾部的公共组件，在Vue项目的src/components目录下新建common/BaseFooter.vue文件

<template>

<div>

页面尾部

</div>

</template>

<script>

export default {

name: 'BlogFooter'

}

</script>

* + 开发登录页面组件，在Vue项目的src/components目录下新建manage/UserLogin.vue文件

<template>

<div>

<base-header></base-header>

<hr/>

<div>

用户名:<input type="text" v-model="loginInfoVo.username" placeholder="请输入用户名" />

<br/>

密码：<input type="password" v-model="loginInfoVo.password" placeholder="请输入密码" />

<br/>

<button v-on:click="login">登录</button>

<br/>

登录验证情况：<textarea cols="30" rows="10" v-model="responseResult"></textarea>

</div>

<hr/>

<base-footer></base-footer>

</div>

</template>

<script>

import baseHeader from '@/components/common/BaseHeader.vue'

import baseFooter from '@/components/common/BaseFooter.vue'

export default {

name: 'UserLogin',

// baseHeader、baseFooter组件给申明到components里面然后在template里面使用

components: { baseHeader, baseFooter },

data () {

return {

loginInfoVo: { username: '', password: '' },

responseResult: []

}

},

methods: {

login () {

this.$axios

.post('/login', {

username: this.loginInfoVo.username,

password: this.loginInfoVo.password

})

.then(successResponse => {

this.responseResult = JSON.stringify(successResponse.data)

if (successResponse.data.code === 200) {

this.$router.replace({path: '/index'})

}

})

.catch(failResponse => {})

}

}

}

</script>

* + 开发首页页面组件，在Vue项目的src/components目录下新建home/UserIndex.vue文件

<template>

<div>

<base-header></base-header>

<hr/>

<div>

这是首页。

</div>

<hr/>

<base-footer></base-footer>

</div>

</template>

<script>

import baseHeader from '@/components/common/BaseHeader.vue'

import baseFooter from '@/components/common/BaseFooter.vue'

export default {

name: 'UserIndex',

// blogHeader/blogFooter组件给申明到components里面然后在template里面使用

components: { baseHeader, baseFooter }

}

</script>

* + 支持跨域，请求路由，页面路由开发，
    - 在main.js文件中修改内容：

import Vue from 'vue'

import App from './App'

import router from './router'

// 引用axios，并设置基础URL为后端服务api地址

var axios = require('axios')

axios.defaults.baseURL = 'http://localhost:8081/bx\_web/api

// 将API方法绑定到全局

Vue.prototype.$axios = axios

Vue.config.productionTip = false

/\* eslint-disable \*/

new Vue({

el: '#app',

router,

components: { App },

template: '<App/>'

})

* + - 在router/index.js页面修改路由代码如下：

import Vue from 'vue'

import Router from 'vue-router'

import UserLogin from '@/components/manage/UserLogin.vue'

import UserIndex from '@/components/home/UserIndex.vue'

Vue.use(Router)

export default new Router({

routes: [

{

path: '/',

redirect: '/login'

},

{

path: '/index',

name: 'UserIndex',

component: UserIndex

},

{

path: '/manage',

redirect: '/login'

},

{

path: '/login',

name: 'UserLogin',

component: UserLogin

}

]

})

* + - 在config/index.js文件中修改

assetsSubDirectory: 'static'要改变为其它字符，比如：'content', 'vue', 'api'等等。

assetsPublicPath: '/' 改为 assetsPublicPath: './'

* + - 在config/index.js文件中添加跨域支持代码：

// 路由接口代理配置

proxyTable: {

'/api': {

target: 'http://localhost:8081/bx\_web',

changeOrigin: true,

pathRewrite: {

'^/api': ''

}

}

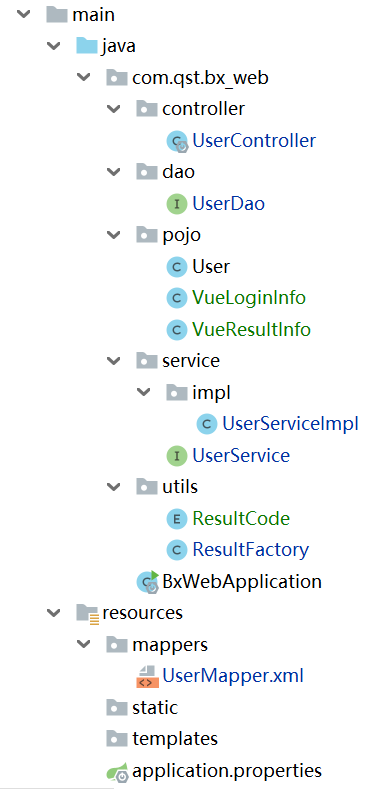
}

* + 单独运行Vue项目查看效果,访问地址：<http://localhost:8080> 。



1. **后端开发流程**
   * 初始化springboot项目( **bx\_web** )

按照前面小节中的步骤创建Spring Boot SSM项目。



* + 开发restful控制器
    - 定义一个用于封装请求数据的类VueLoginInfo.java

public class VueLoginInfo {

@NotNull(message="用户名不允许为空")

private String username;

@NotNull(message="密码不允许为空")

private String password;

public String getUsername() {return username;}

public String getPassword() {return password;}

public void setUsername(String username) {this.username = username;}

public void setPassword(String password) {this.password = password;}

}

* + - 定义一个用于封装响应结果的类VueResultInfo.java

public class VueResultInfo {

//响应状态码

private int code;

//响应提示信息

private String message;

//响应结果对象

private Object data;

public VueResultInfo(int code, String message, Object data) {

this.code = code;

this.message = message;

this.data = data;

}

public int getCode() {return code;}

public void setCode(int code) {this.code = code;}

public String getMessage() {return message;}

public void setMessage(String message) {this.message = message;}

public Object getData() { return data;}

public void setData(Object data) {this.data = data;}

}

* + - 通过一个枚举类ResultCode来定义响应码（参考HTTP状态码语义）

public enum ResultCode {

/\*\*成功\*/

SUCCESS(200),

/\*\* 失败\*/

FAIL(400),

/\*\* 未认证（签名错误）\*/

UNAUTHORIZED(401),

/\*\*接口不存在\*/

NOT\_FOUND(404),

/\*\* 服务器内部错误\*/

INTERNAL\_SERVER\_ERROR(500);

public int code;

ResultCode(int code) {

this.code = code;

}

}

* + - 定义一个用于生成响应结果的工厂类

public class ResultFactory {

public static VueResultInfo buildSuccessResult(Object data) {

return buidResult(ResultCode.SUCCESS, "成功", data);

}

public static VueResultInfo buildFailResult(String message) {

return buidResult(ResultCode.FAIL, message, null);

}

public static VueResultInfo buidResult(ResultCode resultCode, String message, Object data) {

return buidResult(resultCode.code, message, data);

}

public static VueResultInfo buidResult(int resultCode, String message, Object data) {

return new VueResultInfo(resultCode, message, data);

}

}

* + - 创建数据访问接口UserDao.java

@Mapper

public interface UserDao {

User userLogin(String username,String password);

}

* + - 创建数据访问接口的映射文件UserMapper.xml

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >

<mapper namespace="com.qst.bx\_web.dao.UserDao">

<!--普通用户登录-->

<select id="userLogin" resultType="com.qst.bx\_web.pojo.User">

SELECT \*

FROM bx\_users

where username = #{username}

and `password` = #{password}

limit 0,1

</select>

</mapper>

* + - 创建服务接口UserService.java

public interface UserService {

User userLogin(User user);

}

* + - 创建服务接口的实现类UserServiceImpl.java

@Service("userService")

public class UserServiceImpl implements UserService {

@Autowired

private UserDao userDao;

@Override

public User userLogin(User user) {

user = userDao.userLogin(user.getUsername(), user.getPassword());

return user;

}

}

* + - 开发登录控制器，并支持跨域

@Controller

@EnableAutoConfiguration

public class UserController {

@Autowired

public UserService userService;

/\*\*

\* 登录控制器，前后端分离用的不同协议和端口，所以需要加入@CrossOrigin支持跨域。

\* 给VueLoginInfoVo对象加入@Valid注解，并在参数中加入BindingResult来获取错误信息。

\* 在逻辑处理中我们判断BindingResult知否含有错误信息，如果有错误信息，则直接返回错误信息。

\*/

@CrossOrigin

@RequestMapping(value = "/api/login", method = RequestMethod.POST, produces = "application/json; charset=UTF-8")

@ResponseBody

public VueResultInfo userLogin(@Valid @RequestBody VueLoginInfo loginInfo, BindingResult bindingResult) throws IOException {

User user = new User();

user.setUsername(loginInfo.getUsername());

user.setPassword(loginInfo.getPassword());

user = userService.userLogin(user);

VueResultInfo vueResultInfo = null;

if (bindingResult.hasErrors()) {

String message = String.format("登陆失败，详细信息[%s]。", bindingResult.getFieldError().getDefaultMessage());

vueResultInfo = ResultFactory.buildFailResult(message);

} else {

if (user != null&&Objects.equals(user.getPassword(),loginInfo.getPassword())) {

vueResultInfo = ResultFactory.buildSuccessResult("登陆成功。");

} else {

String message = String.format("登陆失败，详细信息[用户名、密码信息不正确]。");

vueResultInfo = ResultFactory.buildFailResult(message);

}

}

return vueResultInfo;

}

}

* + 单独运行后端springboot项目查看效果
    - 由于Vue前端使用了8080端口，所以服务端的端口需要进行修改，不能再使用8080

1. 配置使用外置Tomcat
   * 修改pom.xml文件，添加如下内容
     + 在dependencies标签中添加：

<!--tomcat-embed-jasper中scope必须是provided，因为SpringBootServletInitializer需要依赖 javax.servlet，而tomcat-embed-jasper下面的tomcat-embed-core中就有这个javax.servlet，如果没用provided，最终打好的war里面会有servlet-api这个jar，这样就会跟tomcat本身的冲突了。-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<scope>provided</scope>

</dependency>

* + - 配置打包类型为war文件：

<packaging>war</packaging>

* + - 在build标签中添加：

<build>

<!--项目打包成的包的名称，应该与application.properties的server.context-path=/bx\_web保持一致-->

<finalName>bx\_web</finalName>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

<!--项目最终需要进行打包成war，所以还需要设置一个 maven 的 war 文件的打包插件-->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<configuration>

<warName>bx\_web</warName>

</configuration>

</plugin>

</plugins>

</build>

* + 修改application.properties文件
    - 添加以下内容：

server.servlet.context-path=/bx\_web

* + 让Application继承SpringBootServletInitializer，并实现configure方法

@SpringBootApplication

public class BxWebApplication extends SpringBootServletInitializer {

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {

return application.sources(BxWebApplication.class);

}

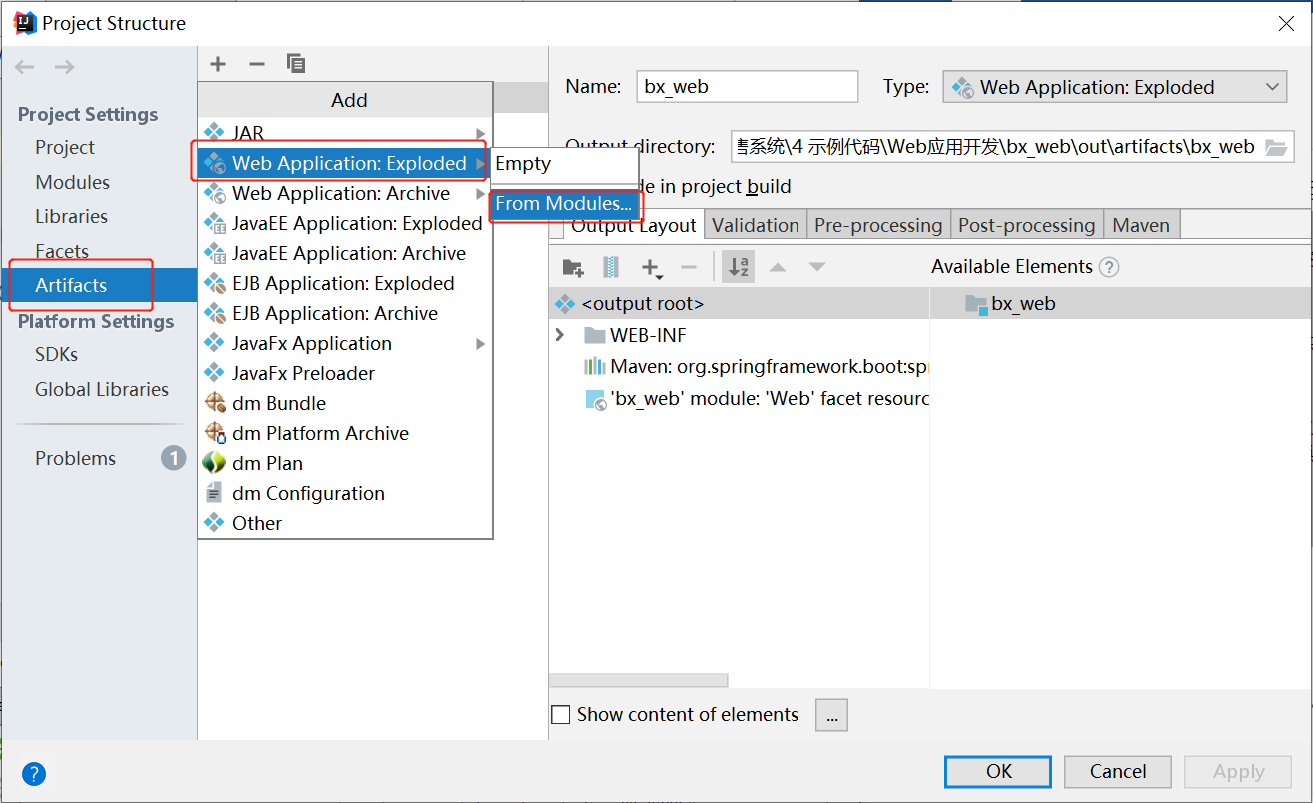
public static void main(String[] args) {

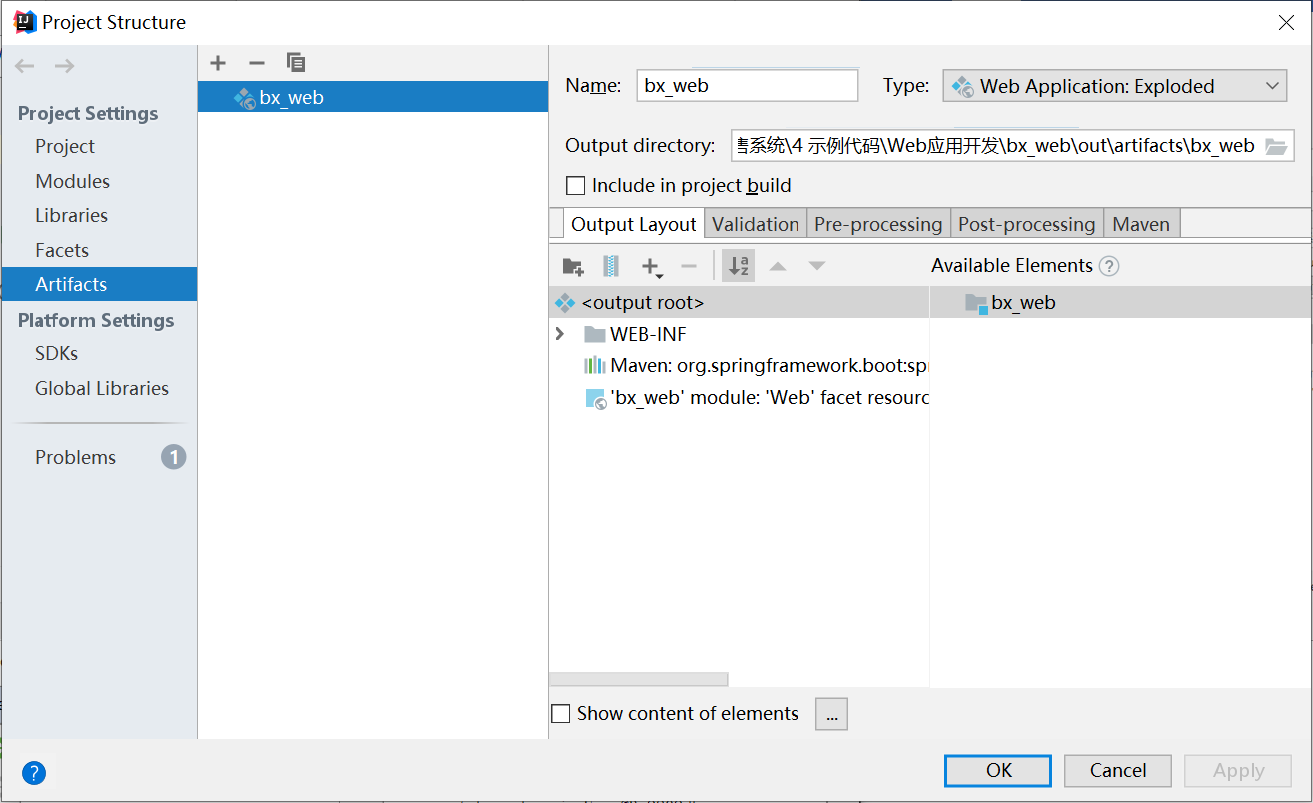
SpringApplication.run(BxWebApplication.class, args);

}

}

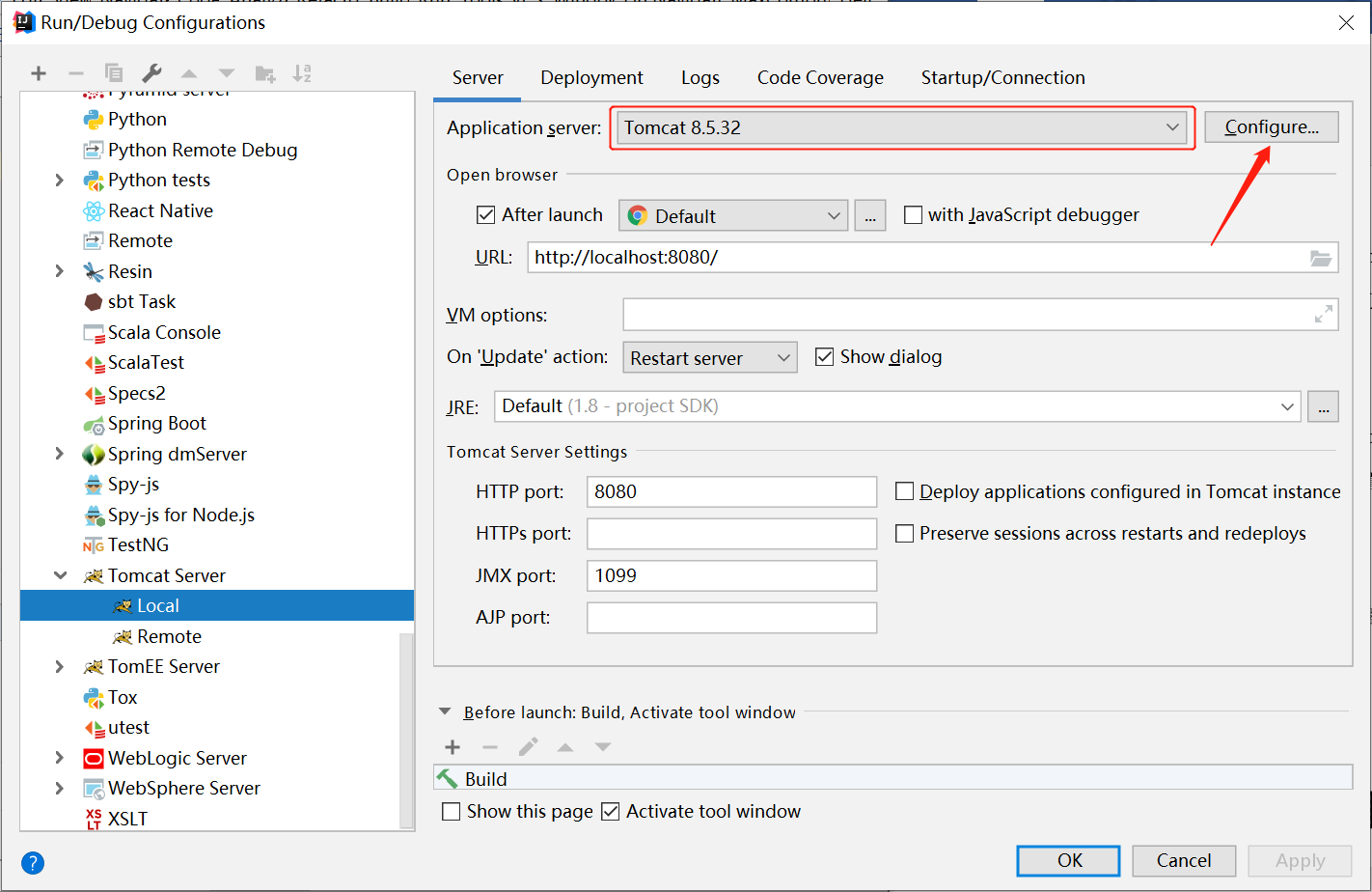
* + 打包项目



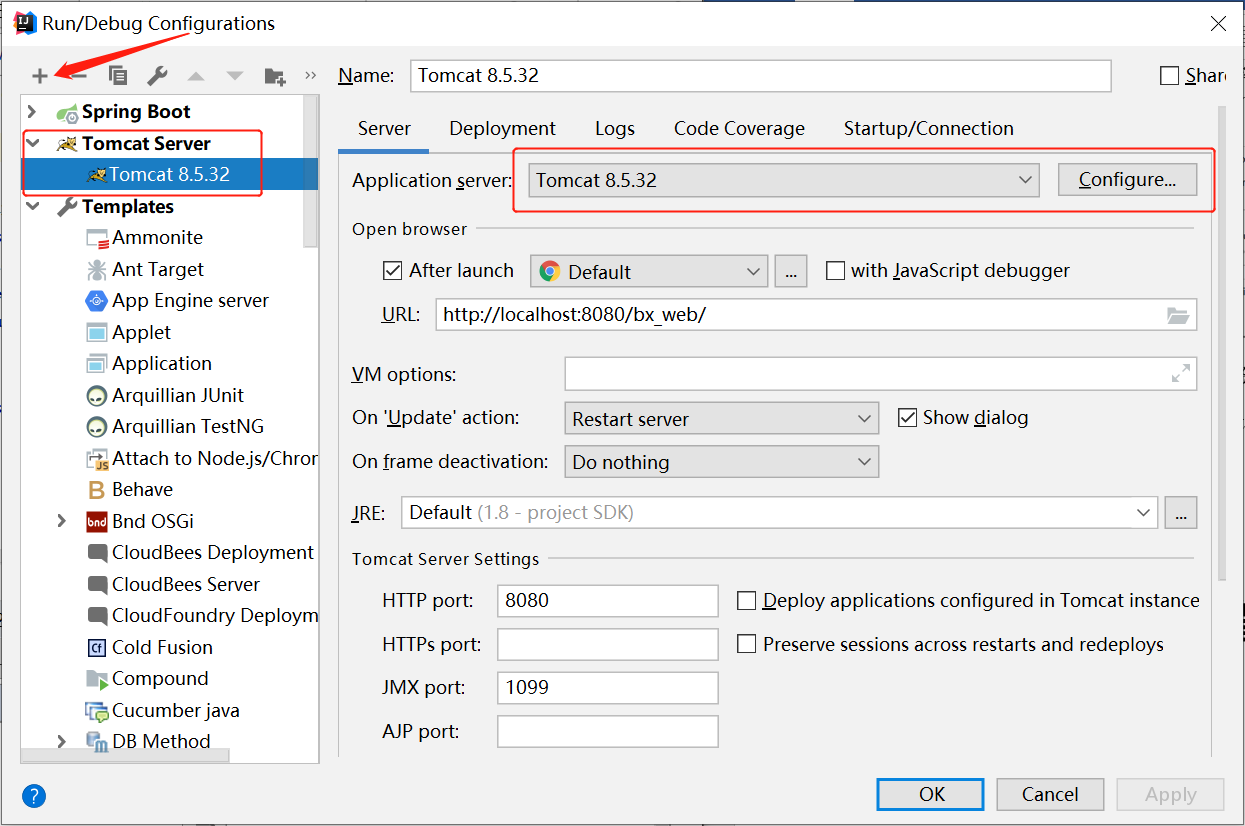


* + 并部署到Tomcat，随后启动Tomcat容器
    - 方式一：将打包好的war包，直接拷贝到Tomcat中，启动Tomcat，打开浏览器访问。
    - 方式二：在IDEA中配置Tomcat服务器，并在IDEA中进行部署（在开发阶段适用）。
      1. 在IDEA中配置Tomcat

点击菜单Run 🡪 Edit Configurations，在左侧依次点击Templates 🡪 Tomcat Server 🡪 Local，然后进行配置。

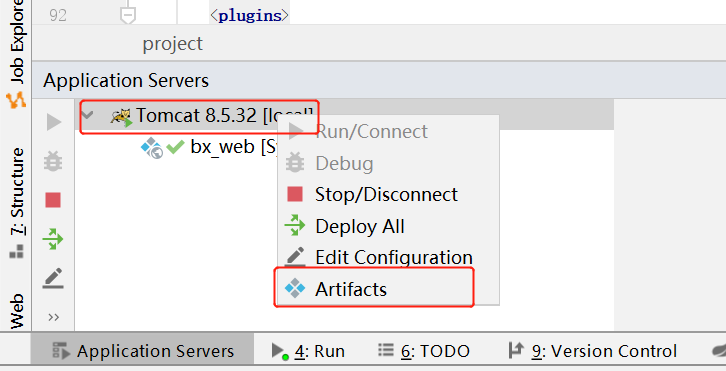


* + - 1. 新建一个Tomcat服务器



* + - 1. 将项目部署到Tomcat服务器





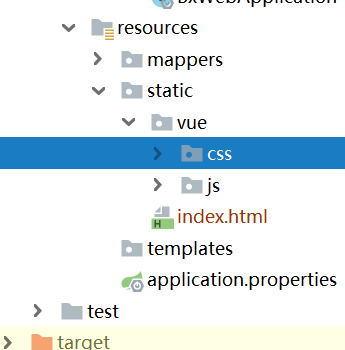
* + - 1. 启动Tomcat，打开浏览器访问

1. **运行部署项目**
   * 使用webpack将Vue项目打包。

进入项目目录下，执行 npm run build命令进行打包，会自动生成打包后的dist目录文件。

* + 将打包的Vue项目集成到springboot项目中。

把dist里面所有文件都拷贝到springboot项目的resources/static目录下。



* + 测试观察结果

重启springboot项目，浏览器访问后台服务地址：https://localhost:8081，会发现页面显示的就是vue开发的前端页面，然后输入用户名密码登录正常。



# 图书推荐系统首页 - 布局

## 任务描述

**内 容**： 图书推荐系统首页vue开发

**学 时**： 2学时

**知识点**：图书推荐系统首页布局，使用vue和elementui

**重点**：

**难点**：

## 授课思路

此页面包括图书类别展示，新书上架，登录，注册，个性化推荐，个性化聚类，图书推荐，图书top。

主要通过此页面让学生学习如何使用vue+elementui来布局页面，了解vue的工作过程，并对图书推荐的功能点有深入了解。

## 任务指导

1. 新建vue页面
2. 布局elementui
3. Vue数据绑定

# 图书推荐系统首页 - 登录

## 任务描述

**内 容**： 系统登录功能实现

**学 时**： 2学时

**知识点**：登录功能实现，vue和elementui熟悉

**重点**：

**难点**：

## 授课思路

首先登录页面的设计，设计成form表单，当用户填写好用户名和密码之后，进行表单必填验证，验证成功提交后端进行登录验证，用户名密码验证，验证成功之后，返回成功信息，跳到首页

## 任务指导

1. 登录form表单的编写设计
2. Form通过vue提交后台进行验证
3. 后端通过验证实现跳转

# 图书推荐系统首页 - 注册

## 任务描述

**内 容**： 图书推荐系统首页vue开发注册功能

**学 时**： 2学时

**知识点**：图书推荐系统注册功能实现，vue学习

**重点**：

**难点**：

## 授课思路

此功能是用户首次登录需要账号注册，注册的过程首先需要用vue编写注册页面，生成form表单提交到后端进行验证，如果用户名已经存在需要给出提示，邮箱验证等过程。如果注册成功往数据库中插入新用户，注册失败给出失败原因。

## 任务指导

1. 新建Register.vue页面,设计登注册页面的form表单
2. Form表单参数验证以及vue编写
3. 提交后端验证成功并写库

# 图书详情页面开发

## 任务描述

**内 容**： 首页通过点击谋一本书进入图书详情页面

**学 时**： 2学时

**知识点**：图书详情的页面开发以及实现思路

**重点**：

**难点**：

## 授课思路

通过首页点击某一本书，获取到某一本书的id之后在vue面访问后端，访问数据库拿到书的详情信息，进行页面跳转，跳转到详情页面进行展示。

## 任务指导

1. 首页中处理一本书的点击事件
2. 跳转的详情页面vue+elementui处理
3. 后端获取一本书的详情信息，并返回到页面进行展示

# 我的购物车页面开发

## 任务描述

**内 容**： 页面点击我的购物车或者点击图书详情页面，会进入我的购物车详情页面

**学 时**： 2学时

**知识点**： 我的购物车列表以及加入购物车实现

**重点**：

**难点**：

## 授课思路

购物车功能主要是将书本或者购买的物品中加入到购物车，实现的思路首先是编写购物车页面，然后将当前的用户信息以及购买的物品的信息，插入到购物车表中，当用户再次登录的时候 我的购物车功能，就是根据当前登录账户的userid来查询购物车表，将购物信息一一列出。

## 任务指导

1. 加链接链接到加入购物车
2. 购物车详情页面开发
3. 加入购物车代码实现
4. 购物车列表功能实现

# 订单支付功能完成

## 任务描述

**内 容**： 对支付页面的开发以及支付之后的后端处理

**学 时**： 2学时

**知识点**： 支付功能完成

**重点**：

**难点**：

## 授课思路

支付主要是对已经加入购物车的数据进行支付提交，主要是调用阿里的支付宝接口，当支付完成后无论是成功还是失败都会去修改订单的状态。

## 任务指导

* + - 1. 支付页面的前端vue开发

主要是在相关页面增加支付按钮

* + - 1. 支付提交的调用阿里的支付宝

支付需要调用阿里的支付宝api，需要到网上自己研究学习，如何在项目中应用

* + - 1. 支付后端逻辑代码开发

当支付成功之后需要去修改订单的状态

# 订单列表功能页面开发

## 任务描述

**内 容**： 订单列表页开发，以及后端逻辑实现

**学 时**： 2学时

**知识点**： 订单列表页开发，取消订单

**重点**：

**难点**：

## 授课思路

通过点击我的订单或者通过下订单按钮进入订单列表中，展示出所有订单列表，可以进行取消订单的操作，以及后端实现对订单的状态修改。

## 任务指导

1. 订单列表详情页面开发

主要是用vue编写订单列表页面，并且增加取消按钮，实现取消按钮的前端逻辑

1. 订单后端功能实现

通过当前登录的用户去查看用户的订单，当取消订单时需要实现订单的状态转化

# 权限控制之shiro学习

## 任务描述

**内 容**：了解什么是shiro，以及shiro的工作流程

**学 时**：2学时

**知识点**： shiro的工作原理以及工作流程

**重点**：

**难点**： 如何在项目中集成shiro

## 授课思路

Shiro是什么，解决了什么问题，我们在此项目中是如何应用的，shiro的原理又是什么

## 任务指导

1. shiro是什么
2. Shiro的系统架构是什么
3. 程序中如何使用shiro
4. Shiro的原理
5. Shiro应用中应该注意的事项

# 用户权限管理设计思路

## 任务描述

**内 容**：了解用户权限管理的设计思路

**学 时**：2学时

**知识点**：表的设计思路，以及设计流程

**重点**：

**难点**：

## 授课思路

首先我们要明确用户，角色，权限之间的关系，我们的权限管理是这样的，用户拥有角色，角色拥有权限，所以我们在设计表的时候应该至少有5张表，用户表，角色表，权限表，用户角色关联表，角色权限关联表，

## 任务指导

数据表的设计，详见“数据库设计说明”文档

# 用户管理 - 用户列表

## 任务描述

**内 容**：后端页面用的jquery技术，后端springboot +ssm

**学 时**：2学时

**知识点**：查询用户列表，配置用户列表页面权限

**重点**：

**难点**：

## 授课思路

用户列表功能主要是将用户信息一一列出并提供编辑等功能的入口。实现的思路首先是编写用户页面，配置页面的角色权限，查询用户表信息，关联用户详细信息展示在列表上。

## 任务指导

* + - 1. 新建用户列表界面

对用户主要的信息简要展示，以及相应的vue逻辑实现

* + - 1. 用户列表代码实现

后端主要是查询当前user表中的所有用户，传到前端展示

* + - 1. 配置页面角色权限

给用户设置角色，给相应的角色绑定权限

# 用户管理 - 添加用户

## 任务描述

**内 容**：后端页面用的jquery技术，后端springboot +ssm

**学 时**：2学时

**知识点**：增加新的用户信息

**重点**：

**难点**：

## 授课思路

在后端管理平台，用户管理菜单下增加添加用户菜单，编写添加用户的功能代码，

添加时对必填信息进行校验，添加用户完成后可在用户列表中查询到。

## 任务指导

* + - 1. 新建添加用户界面，利用jsp绘制须填写的form表单
      2. 用户必填信息进行校验
      3. 添加用户代码实现
      4. 配置添加用户页面权限

# 用户管理 - 用户编辑及删除

## 任务描述

**内 容**：后端页面用的jquery技术，后端springboot +ssm

**学 时**：2学时

**知识点**：对用户信息进行编辑修改和删除

**重点**：

**难点**：

## 授课思路

在用户列表页面中，增加编辑和删除按钮，实现用户信息的编辑修改和删除功能。

## 任务指导

* + - 1. 用户列表界面中增加编辑和删除按钮，实现用户的增加和删除
      2. 用户信息编辑和删除代码实现
      3. 配置编辑和删除功能权限

# 云店铺管理 - 云店铺列表及添加

## 任务描述

**内 容**：图书推荐系统对店铺的查询及添加功能

**学 时**：2学时

**知识点**：云店铺的查询及添加功能

**重点**：

**难点**：

## 授课思路

在店铺列表页面中，增加编辑和删除按钮，实现店铺信息的编辑修改和删除功能。

## 任务指导

* + - 1. 编写店铺信息的列表展示页面
      2. 店铺列表代码实现
      3. 编写添加店铺页面
      4. 添加店铺代码实现
      5. 配置店铺列表和添加店铺的页面权限

# 云店铺管理 - 云店铺编辑及删除

## 任务描述

**内 容**：图书推荐系统对店铺的编辑和删除功能

**学 时**：2学时

**知识点**：云店铺的编辑和删除

**重点**：

**难点**：

## 授课思路

在店铺列表中增加编辑和删除的功能按钮。

## 任务指导

* + - 1. 店铺列表界面中增加编辑和删除按钮
      2. 编辑店铺信息和删除店铺信息代码实现
      3. 配置编辑和删除功能权限

# 角色权限管理 - 角色列表及编辑与删除

## 任务描述

**内 容：**系统后端开发进行角色列表的开发

**学 时：** 2学时

**知识点：**角色列表查询及编辑，删除功能

**重 点:**

**难 点:**

## 授课思路

在后端管理平台，角色权限菜单下增加角色列表菜单，编写角色列表和编辑删除功能代码。

编辑时对必填信息进行校验。功能完成后配置菜单权限。

## 任务指导

* + - 1. 编写角色列表展示页面
      2. 角色列表代码实现
      3. 列表页面增加编辑与删除按钮
      4. 编辑与删除代码实现
      5. 配置角色列表和添加角色的页面权限

# 角色权限管理 - 角色添加

## 任务描述

**内 容**：图书推荐系统添加新角色功能

**学 时**：2学时

**知识点**：角色添加

**重点**：

**难点**：

## 授课思路

增加添加角色的菜单及页面，实现添加角色的功能代码，保存时对必填信息进行校验。功能完成后配置角色菜单权限。

## 任务指导

* + - 1. 编写添加角色页面，对角色进行新增
      2. 添加角色代码实现，将form中的信息保存到角色表中
      3. 配置角色列表和添加角色的页面权限

# 角色权限管理 - 权限列表

## 任务描述

**内 容**：图书推荐系统对权限列表的管理

**学 时**：2学时

**知识点**：权限树的构成，以及后端代码的编写

**重点**：

**难点**：

## 授课思路

对权限树的构造实质上对数据的构造，也就是说我们在设计表的时候就要根据表的构成来取数据，比如parentid，child这种构成，程序中处理成树结构

## 任务指导

1. 新建权限页面，主要是用ztree构造树形结构，注意引用ztree的js包
2. 后端构造数据结构，根据parentId来查出所有的下面的孩子节点，来构造成树形的结构

# 我的店铺 - 店铺信息查看

## 任务描述

**内 容：**主要是维护个人的店铺的基本信息

**学 时：** 2学时

**知识点：** 店铺信息的更新保存

**重 点:**

**难 点:**

## 授课思路

通过超级管理员或者云店铺管理员创建的拥有店铺权限的人员进入后台管理会看到我的店铺信息，但是只能修改并不能新增，也可以看到店铺的基本信息

## 任务指导

1. 店铺信息页面的开发
2. 对店铺信息的修改以及保存

# 我的店铺 - 图书列表及编辑

## 任务描述

**内 容：**对已经上架的图书进行管理列表页展示，以及某一本图书进行编辑保存

**学 时：** 2学时

**知识点：** 图书的列表以及更新功能的开发

**重 点:**

**难 点:**

## 授课思路

首先根据当前登录用户的id来确认当前用户拥有哪些书本，然后调用接口到前端列表展示，当点击某一本书的编辑按钮时，根据图书id获取书本信息，点击保存进行书本的更新

## 任务指导

1. 新建图书列表页面，展示图书的主要相关信息
2. 获取当前用户创建的图书，列表展示
3. 对编辑书本进行更新到数据库

# 我的店铺 - 图书添加

## 任务描述

**内 容：**主要是图书上新，即添加书本

**学 时：** 2学时

**知识点：** 对书本的添加

**重 点:**

**难 点:**

## 授课思路

当前登录的用户身份必须是商铺的管理员，可以对新书进行上架，当新增书本的时候会带着当前登录用户的id，传到后端存到数据库，传到数据库的书也会实时的在首页上进行显示。

## 任务指导

1. 新建书本添加页面，在相应的vue目录下
2. 新增书本的后端开发以及功能实现

# 订单管理 - 查看订单

## 任务描述

**内 容**：对生成的订单的管理包括取消订单，查看编辑订单，查看买家信息等功能学习开发

**学 时**：2学时

**知识点**：

**重点**：

**难点**：

## 授课思路

当我们购买了图书之后会生成我们的订单，订单模块的主要功能有订单的查看，订单的取消，订单的取消主要是后端更新下订单的状态字段即可，以及查看买家的相关信息

## 任务指导

1. 新建订单列表页面以及相关页面
2. 订单页面访问后台返回列表

订单取消的后端逻辑

# 个人中心管理 - 个人信息编辑

## 任务描述

**内 容**：对个人信息进行修改，包括修改密码功能

**学 时**：2学时

**知识点**：个人信息的修改

**重点**：

**难点**：

## 授课思路

当我们以普通用户的身份登录的时候，我们只能对自己信息进行修改，以及密码重置的功能，对个人信息以及密码修改，提交表单，然后更新到数据库

## 任务指导

1. 对我的个人信息页面的开发，主要是根据用户的id查询用户信息返回到前端
2. 将个人信息更新到数据库，包括密码修改

# 前端—个性化展示（词云、聚类、热点图、Top20展示）

## 任务描述

在主页面的右侧导航位置展示图书推荐、图书词云、Top20。



在每个图书类别下，展示该类别销量最高的20本图书。



## 任务指导

无

# 项目测试、整合、发布

## 任务描述

**内 容**：

**学 时**：

**知识点**：

**重点**：

**难点**：

## 授课思路

本模块由学生独立完成。

## 任务指导

无