一、Vue前端

## 与jquery的对比

jQuery对DOM对象进行编辑。

Vue对值进行编辑，而这个值绑定了DOM对象。

例：

JQuery修改：

<div>

<p>大家好，我是<span id="name">张三<span>!</p>

<p>我是一名<span id="jop">医生</span>.</p>

<button id = "modifyBtn">修改</button>

</div>

<script type="text/javascript">

$("#modifyBtn").click(function(){

$("#name").text("李四");

$("#jop").text("老师");

});

</script>

Vue修改：

<div id="app">

<p>大家好，我是<span>{{name}}<span>!</p>

<p>我是一名<span>{{jop}}</span>.</p>

<button v-on:click="modifyInfo">修改</button>

</div>

<script>

new Vue({

el: '#app',

data:{

name:"张三",

jop:"医生"

},

methods:{

modifyInfo:function(){

this.name = "李四";

this.jop = "老师";

}

}

})

</script>

## 基本实现逻辑

2.1只有一个首页，根目录下的index.html,其他所有页面均可通过路由”router”来实现页面元素的替换。



最关键的是<div id="app"></div> 此部分将被自动替换。

2.2由src/App.vue实现首页页面元素的展现



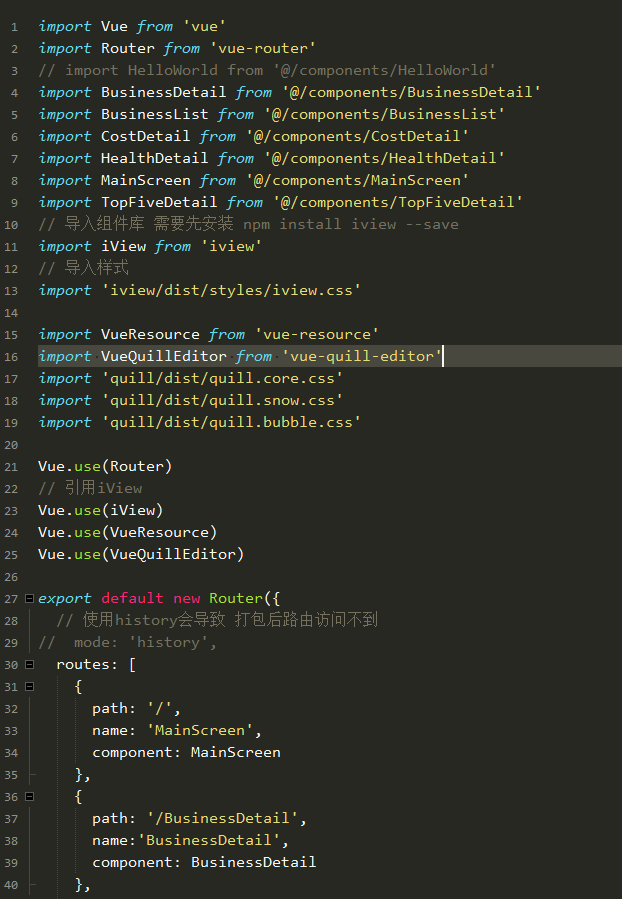
关键为<router-view/>,此部分将由路由自动替换。

2.3由src/main.js自动完成自动替换标签的绑定及全局环境的配置。



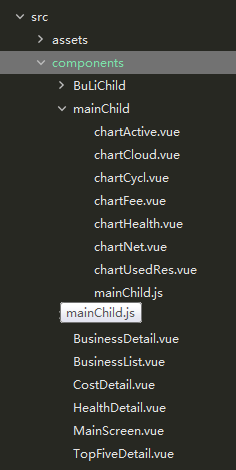
Id为app的标签将被替换为components内的对象APP，这个对象在main.js文件上方导入，路径为./App，此为缺损写法，实际文件为./App.vue。通过这个配置将index.html内的标签替换为App.vue内的内容

2.4路由的设置，src/router/index.js内



分两步：1-导入页面路径。2-配置路由。

2.5实际页面内容，路径可自定义，一般放在src/ components下。



2.6文件之间的引用。Vue内的文件间引用主要思路为，导出被引用文件，引用处导入文件。

例：2.5中components/MainScreen.vue引用了components/mainChild/chartActive.vue的内容。

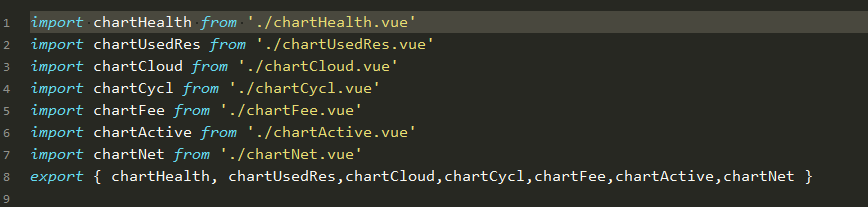
实现步骤：

1. chartActive.vue导出文件内容，被导出内容可以视作一个叫做chartActive的对象。

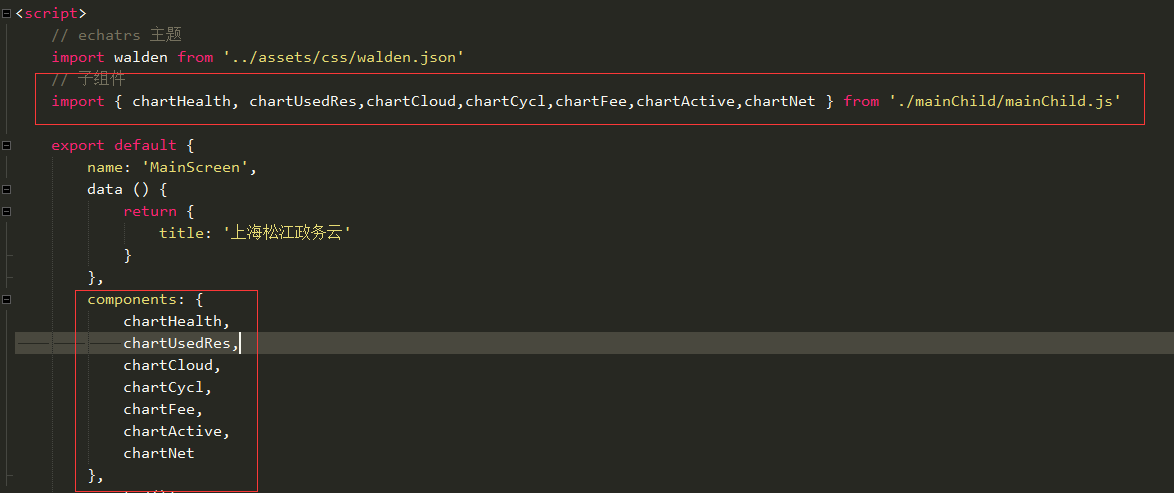




1. (可选步骤)将多个被引用的文件内容集成，方便统一引用。此处使用components/mainChild/mainChild.js作为整合文件



3) MainScreen.vue引用。



注意，红框选中的内容均不可少。Components相当于将导入的内容转换为当前文件内的元素。

4) 使用被引用内容。



此为替换页面元素，将chartActive作为一个子组件，替换了父组件的一个区域。

标签名chart-Active对应，子组件chartActive.vue内的class。



此为调用子组件方法的方式。此处的chartActive对应的是父组件标签的ref属性值。

2.7 dom元素文本内容修改

Html内使用{{参数名}}。

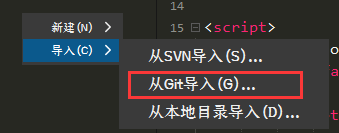
js部分在data内定义参数，在方法内为参数赋值。

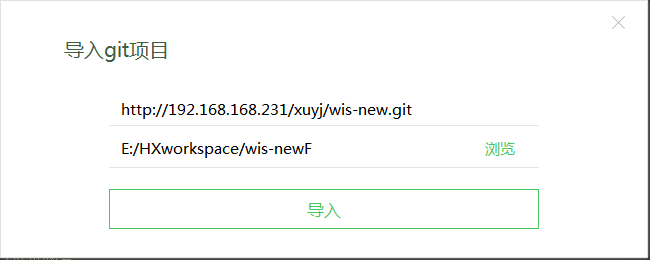
## 3.框架项目搭建

Git地址：<http://192.168.168.231/xuyj/wis-new.git>

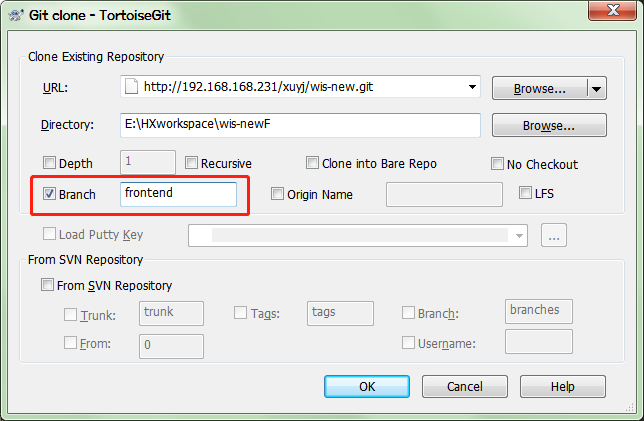
3.1导入项目

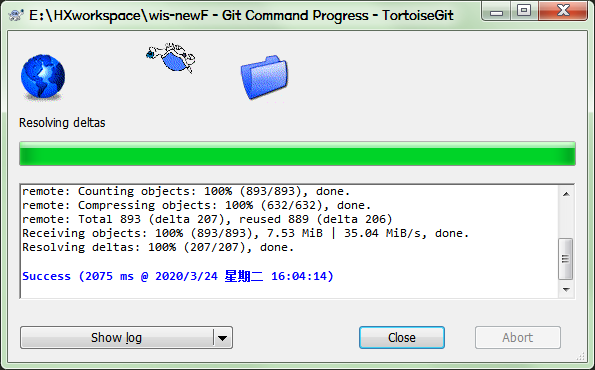
左侧项目树空白处右键，选择导入方式，此处为git





注意项目所在分支，如下图：

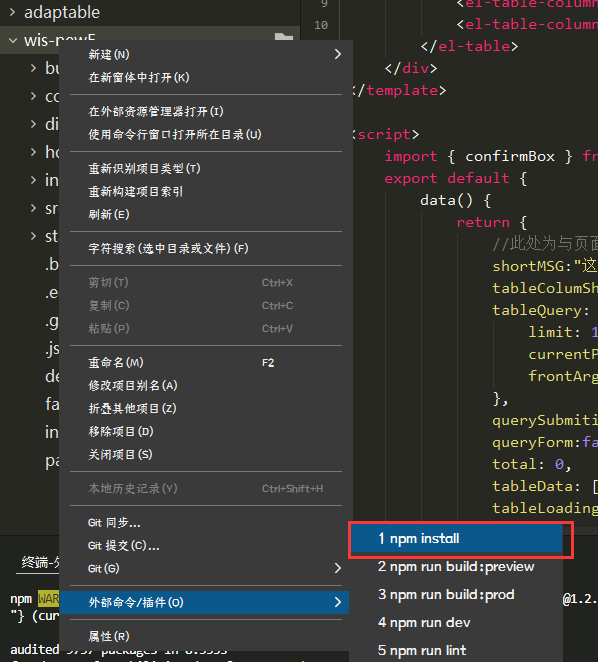




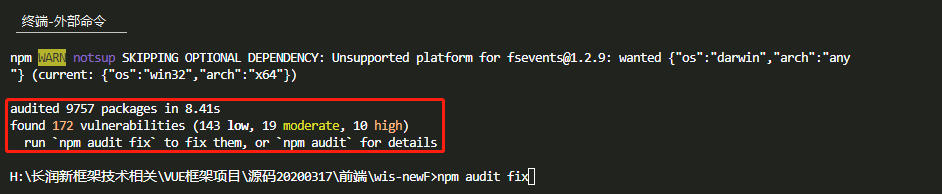
项目导入完毕。

3.2运行项目

右键项目-外部命令/插件-npm install，自动安装项目开发环境：

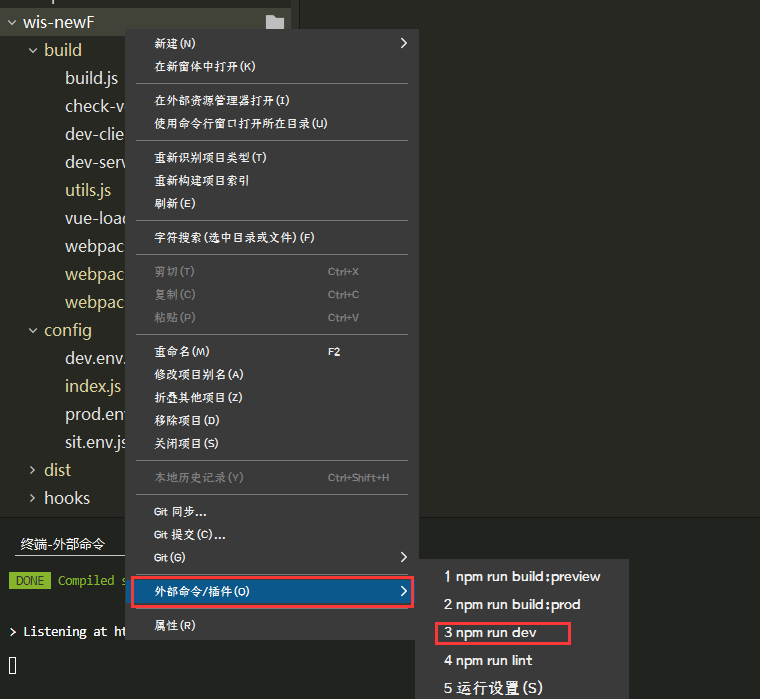


出现如下提示时，根据提示输入npm audit fix进行修复。



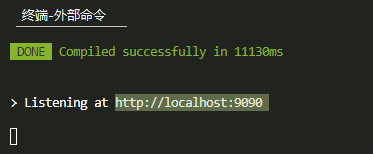
看到如下提示，说明修复完成：

右键项目-外部命令/插件-npm run dev



3.3验证是否运行

在浏览器输入终端显示的地址



看到登录页面，说明启动正常

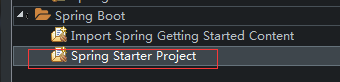


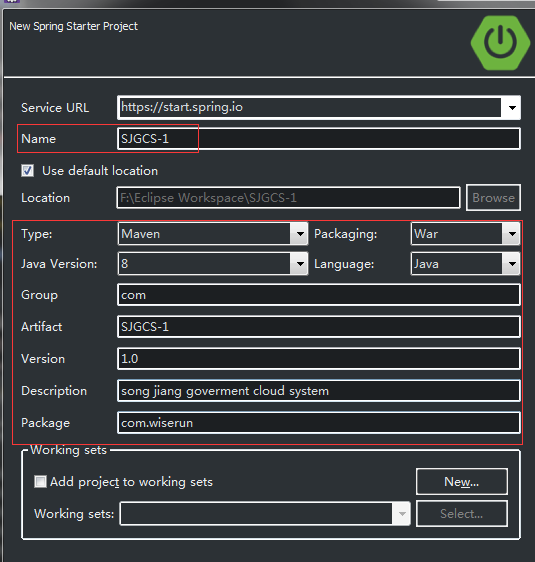
二、Springboot

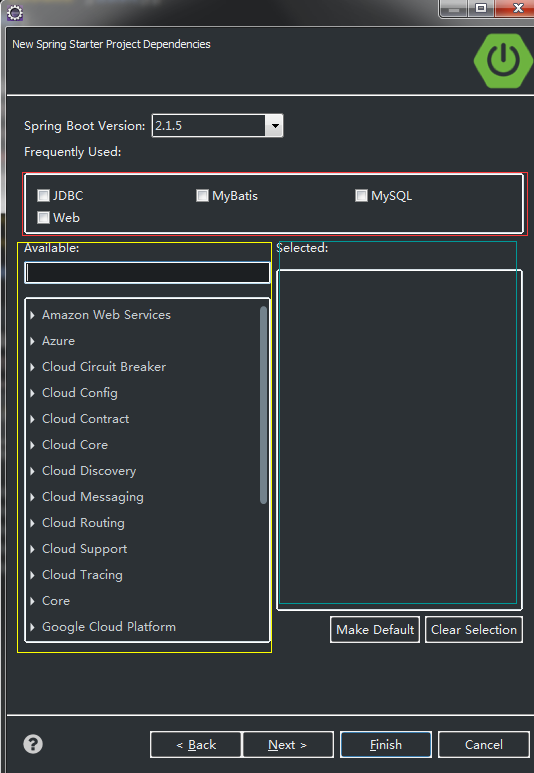
## 基础使用

1.1必要支持Maven。在创建springboot项目时，可自由选择扩展功能，以eclipse为例。

New ->other->选择下方的spring starter project->填写项目信息->选择扩展功能->等待自动构建springboot项目。

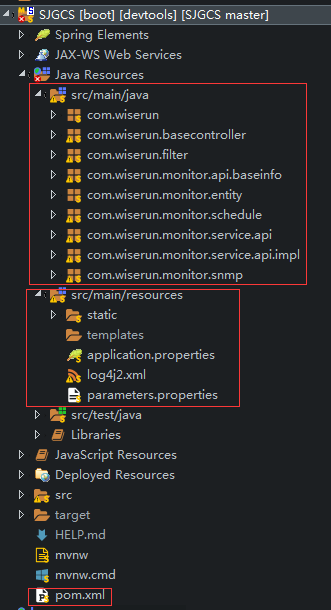






红色框为增加使用过的功能，黄色框为所有可选功能，绿色框为本次已选功能。

1.2 目录结构



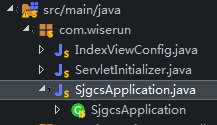
红框从上往下，依次为后台文件，资源文件及maven配置文件。

后台文件存放实际项目实现代码。

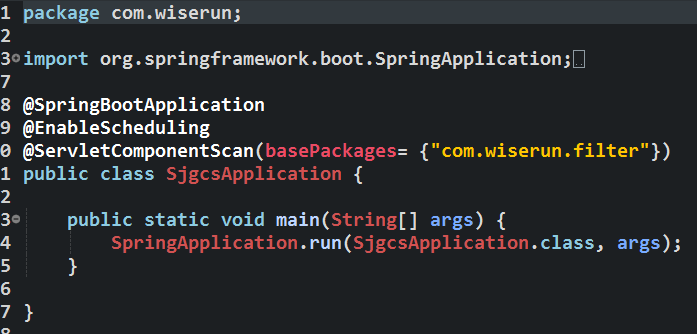
资源文件存放静态资源、页面、配置文件等。

Maven配置文件，管理项目所用的各类扩展包。

1.3 开发时的项目启动。



创建项目时填写的package名下，项目名Application.java文件为主启动文件。



拦截器的调用，定时工具的开启均在此处设置。

1.4框架文件结构

分为：

1. 引导，即application.java文件及ServletInitializer文件。
2. 控制，即contoroller。使用@RestController或者@Controller进行注解。
3. 业务逻辑实现，即service及service接口，使用@Service进行注解。
4. 实体类，使用 @Component或@Entity进行注解。
5. 持久层，即dao层。通过@Mapper注解和实体类.xml文件可自动完成实体类的数据库操作，无需手动创建dao文件。

1.5 实现过程：

1. 引导层启动项目
2. 接收到请求被controller解析，确定使用哪个业务逻辑实现方式去实现
3. 业务逻辑实现时，通过绑定的mapper文件，调用实体类的持久化操作
4. Mapper文件被调用后，根据被使用的方法，在对应的xml文件中找到数据库操作语句
5. xml中将根据实体类字段及sql语句自动生成完整的sql语句，并执行
6. 执行后的返回结果，自动封装为xml中指定的返回类型的格式，再依次返回至controller，最后返回响应内容给请求方。

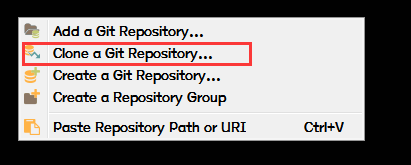
1.6 springboot项目自动创建后，可能无法直接部署至tomcat。如果想要在eclipse内将项目添加到tomcat内，需在Navigator窗口，项目名/.settings/org.eclipse.wst.common.project.facet.core.xml文件内，将 <installed facet="jst.web" version="4.0"/>修改版本为3.0。

## 2.框架项目搭建

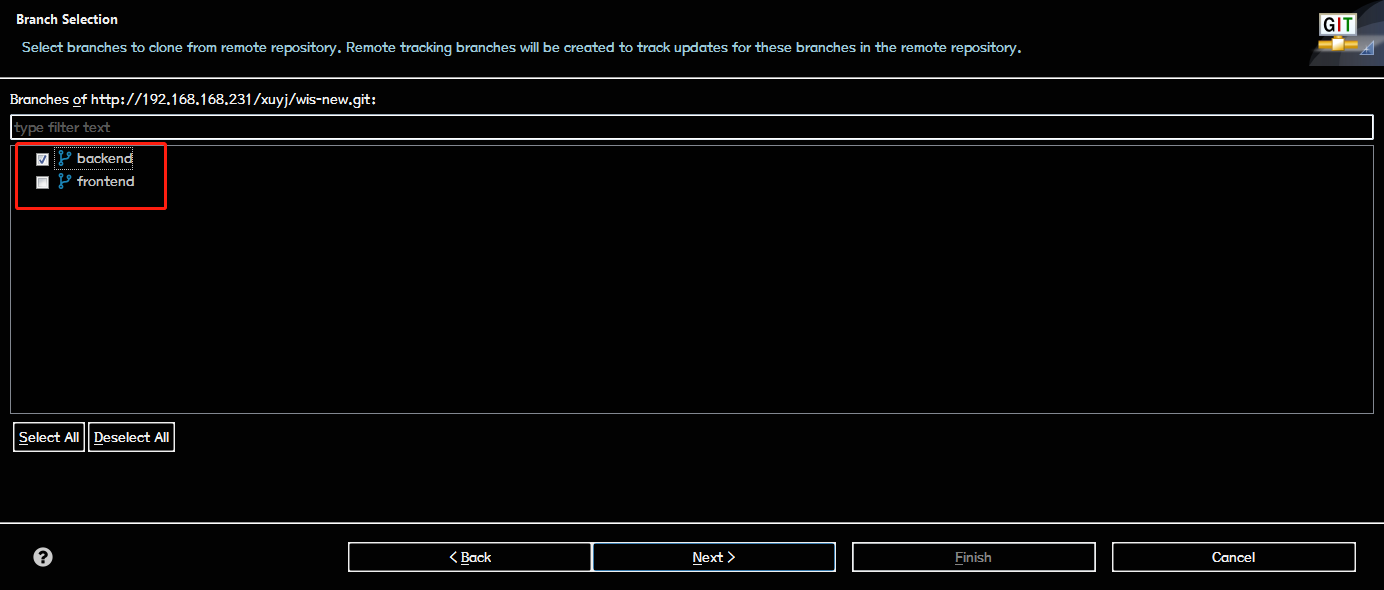
Git地址：<http://192.168.168.231/xuyj/wis-new.git>

2.1 eclipse使用git导入到本地库，再从本地库拉取到项目内

从复制的git地址克隆



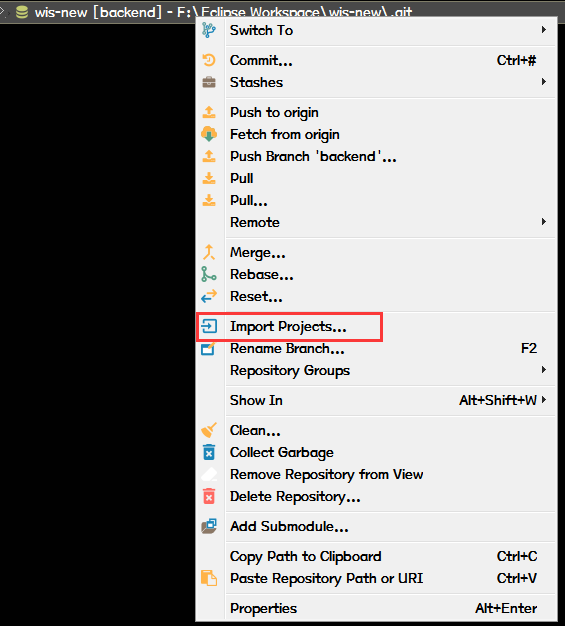
选择分支



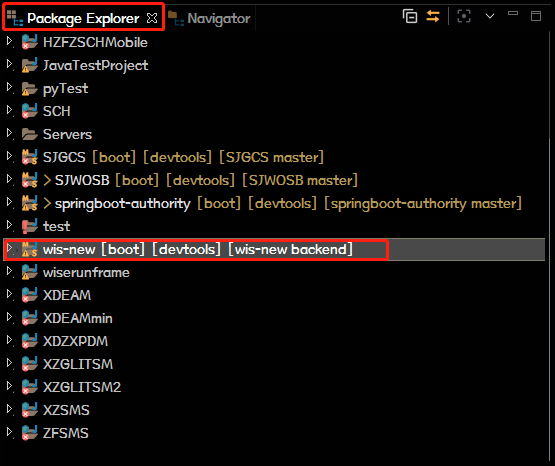
选择本地目录



将项目从本地库拉取到workspace



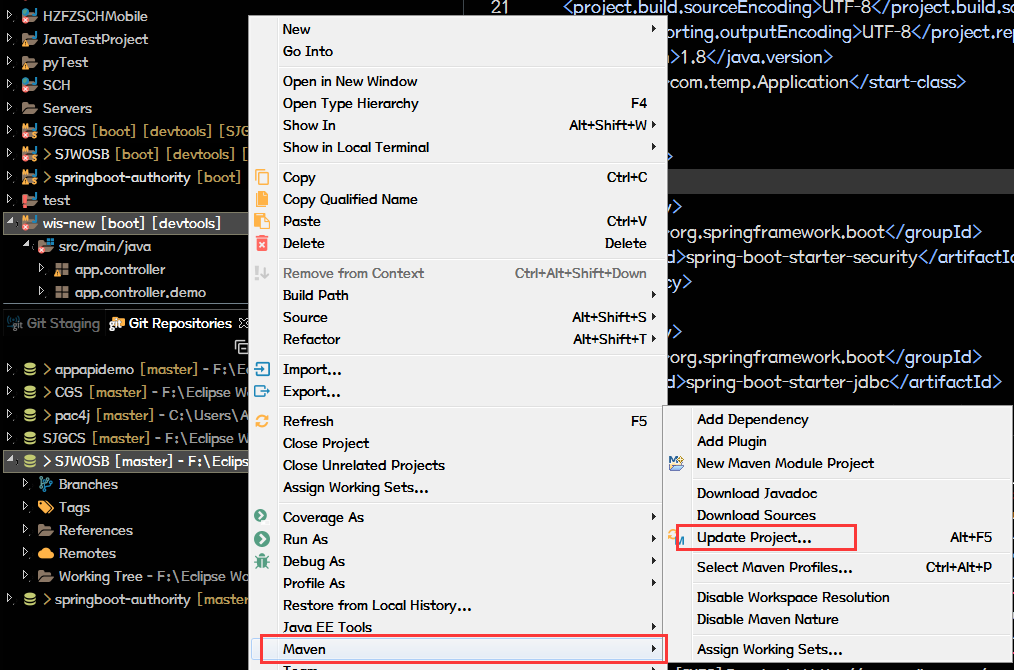
项目在包浏览区域出现，导入成功



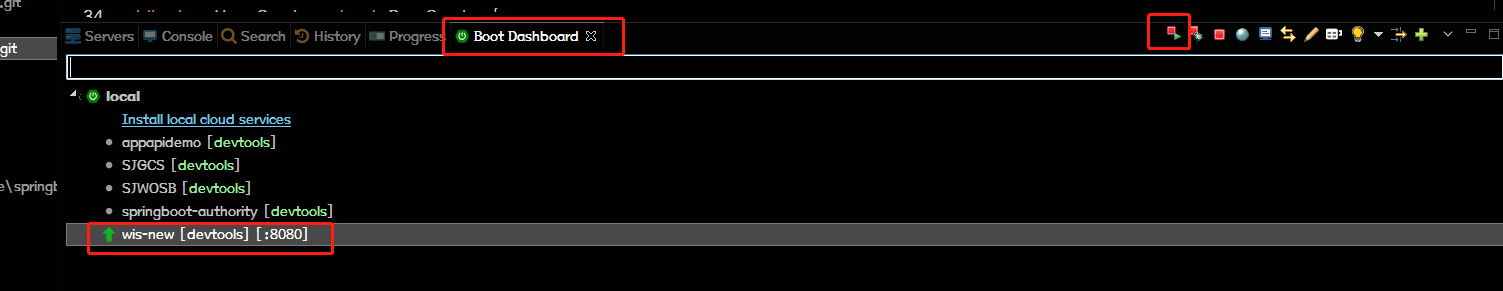
2.2更新项目的java及service环境版本(如有需要)

右键项目-properties-java build path内

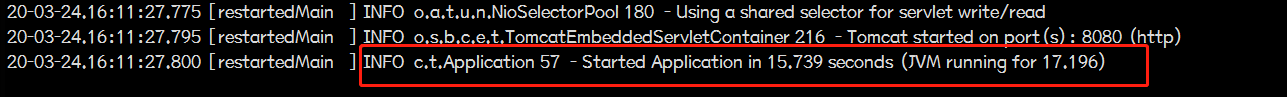
2.3更新maven包，右键项目，maven-update project，等待项目jar包更新(如有需要)



2.4启动项目



看到如下提示，说明启动成功：

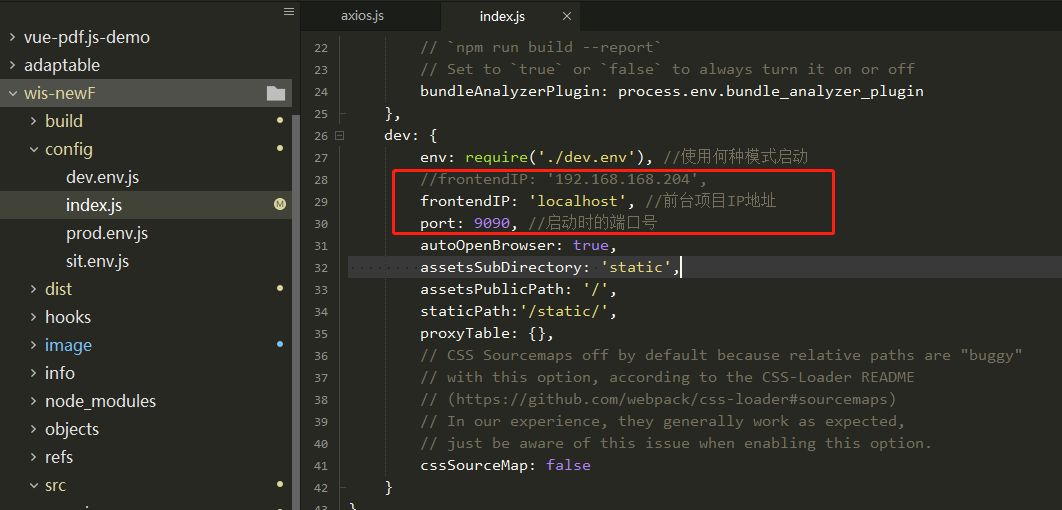


# 三、前后端对接

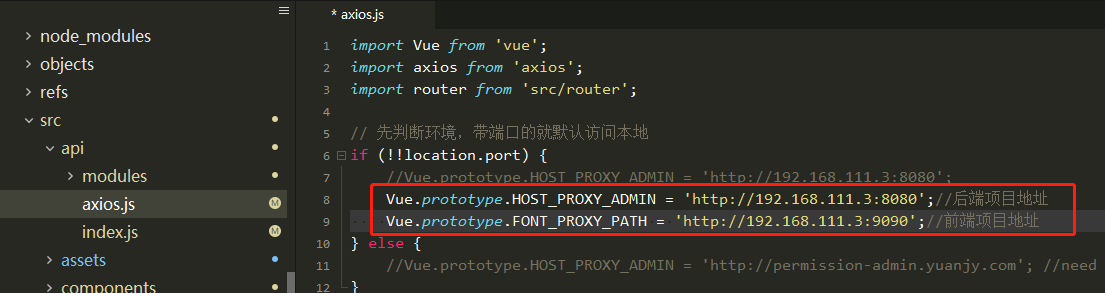
## 前端配置

1.1 IP修改，以下文件的IP地址根据实际情况修改

#前端项目启动配置



#前端项目与后端项目交互的ajax请求配置



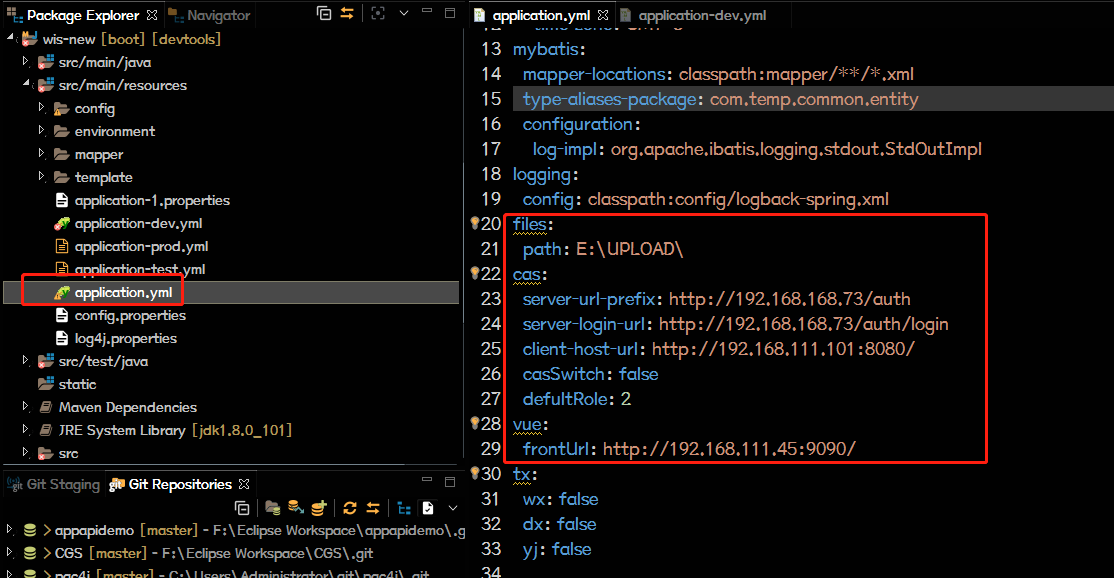
## 后端配置

2.1 IP修改，根据实际项目地址修改

Files为文件上传路径

Cas为单点登录配置

Vue为前端项目配置



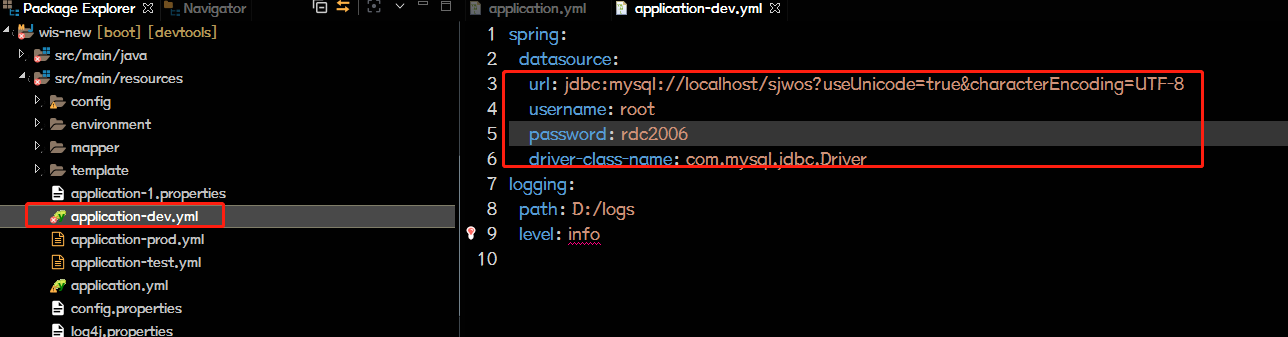
2.2创建数据库

使用mysql管理工具执行sql语句，注意修改数据库名



2.3配置数据库

修改下方红框内数据库URL及账号配置

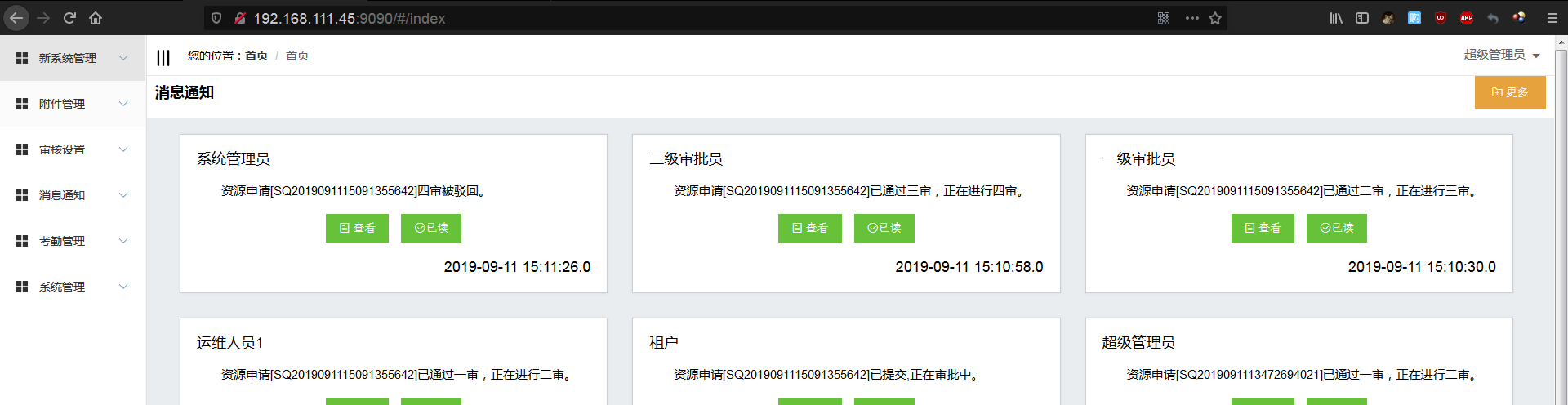


## 验证

3.1将前后端项目全部重启

3.2在登陆界面使用默认的管理员(账号U13800000000，密码123456)进行登录

3.3进入首页，说明登录成功

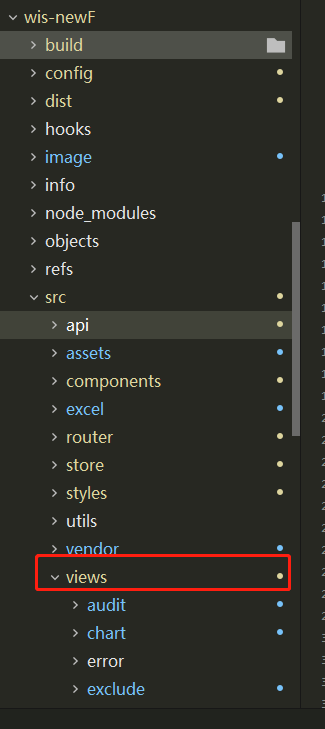


# 创建新页面

## 1.前端增加页面文件

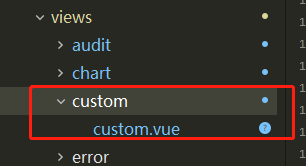
1.1前端项目添加文件

页面文件路径在src/views下

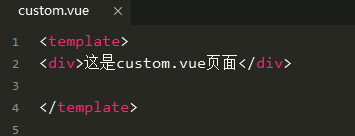


比如：

新建目录custom，新建vue文件custom.vue

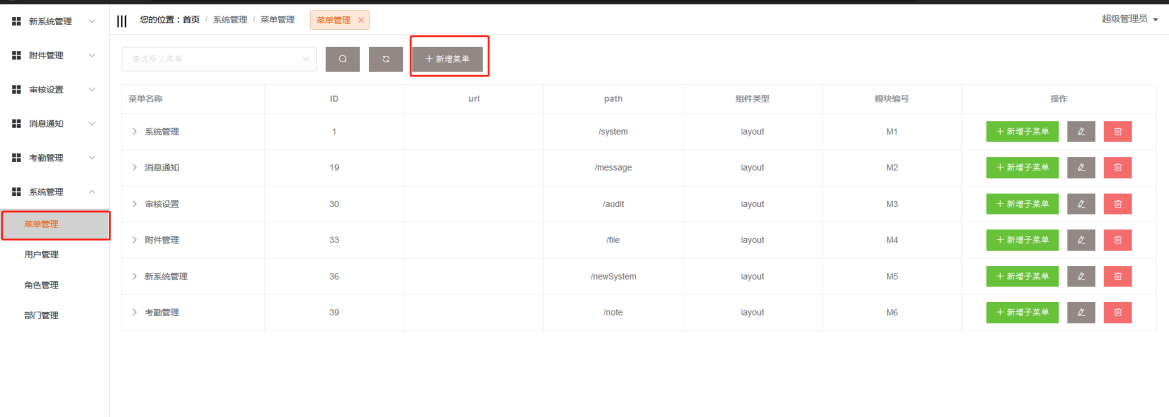


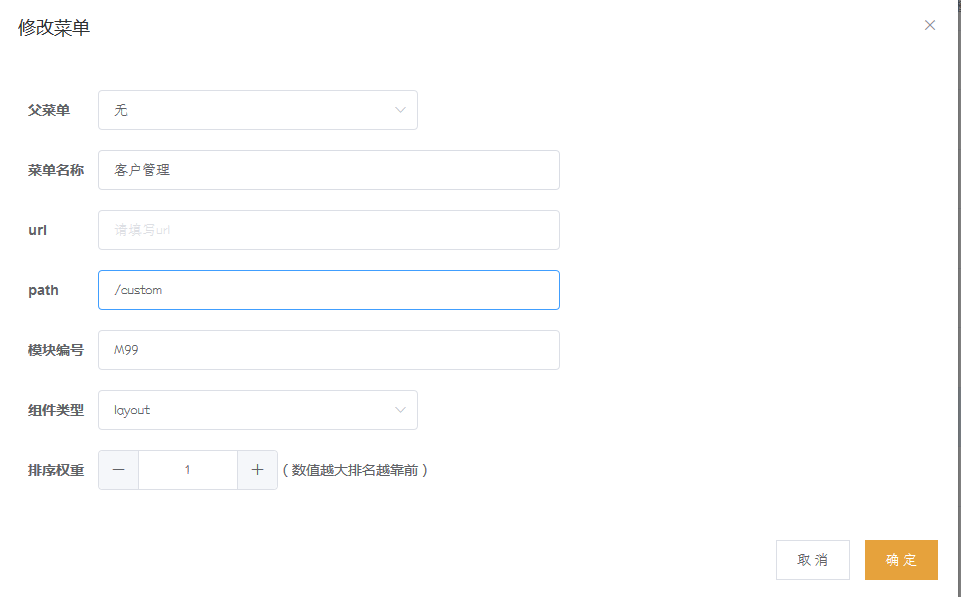
在template标签内添加随意的html代码，比如：



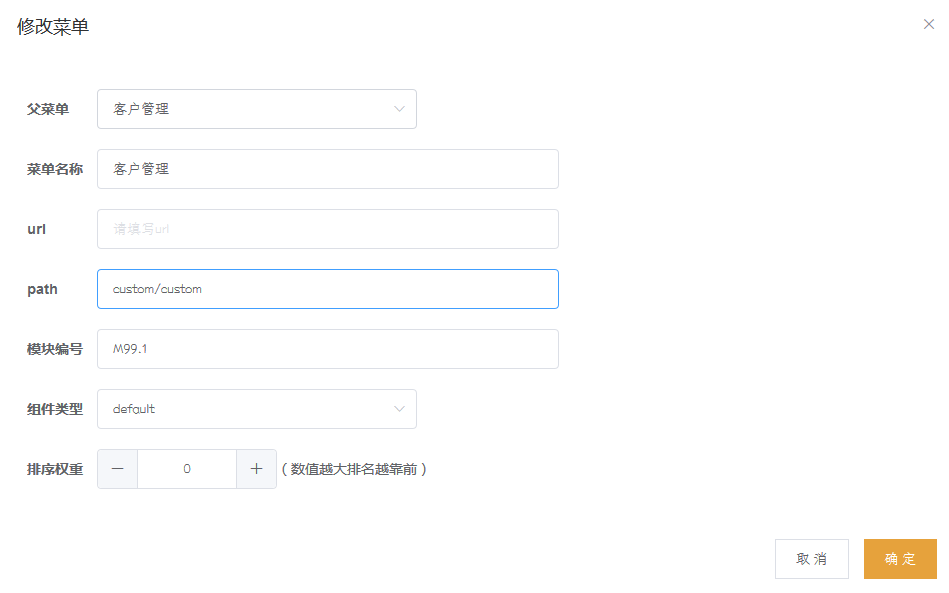
1.2界面添加系统菜单

打开系统管理-菜单管理，点击新增菜单按钮，并填写相关信息，可参考现有菜单，注意菜单层级

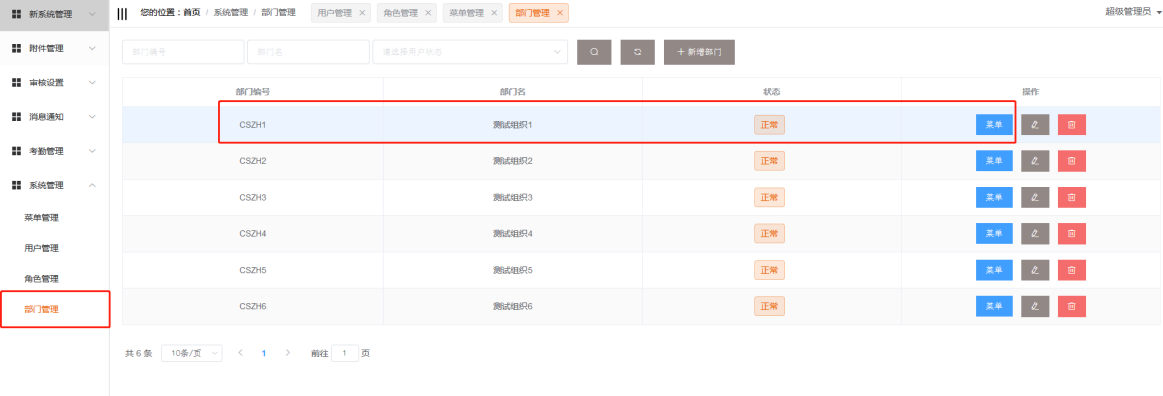




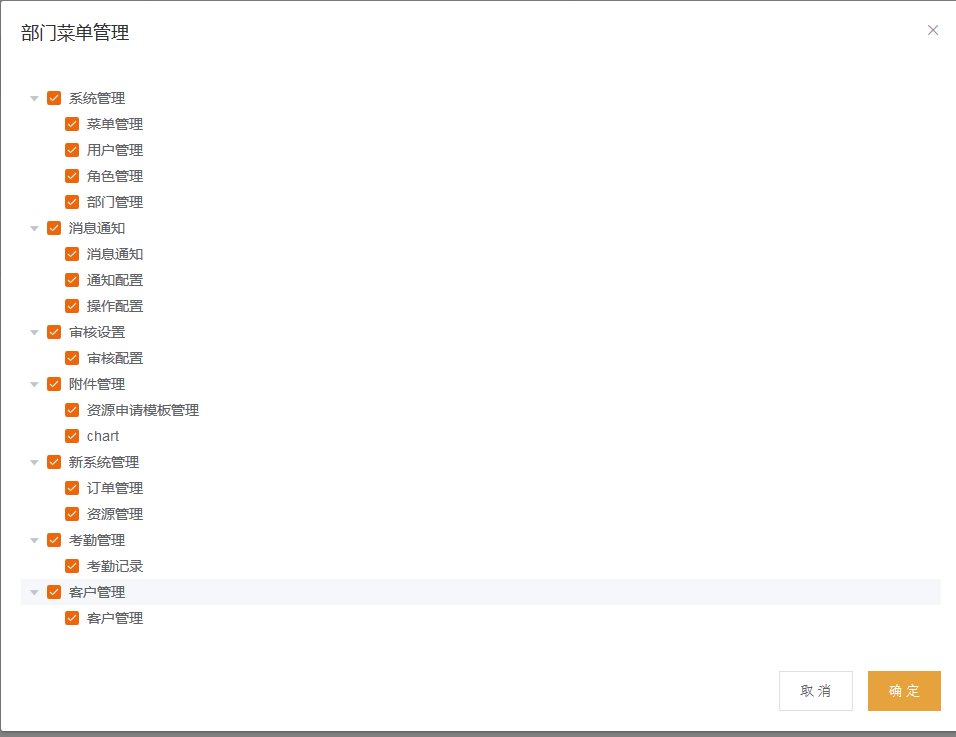
然后再次点击新增菜单 按钮，此次需选择父菜单，如图：



为需要查看菜单的部门配置菜单，如图：



点击菜单按钮，勾选新建的页面。



刷新页面，看到左侧菜单：

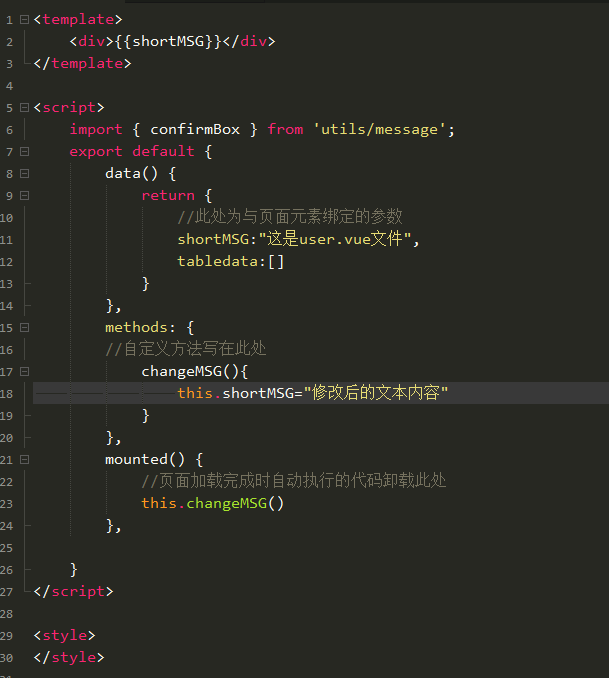


点击菜单，看到div内容，说明页面路径正常

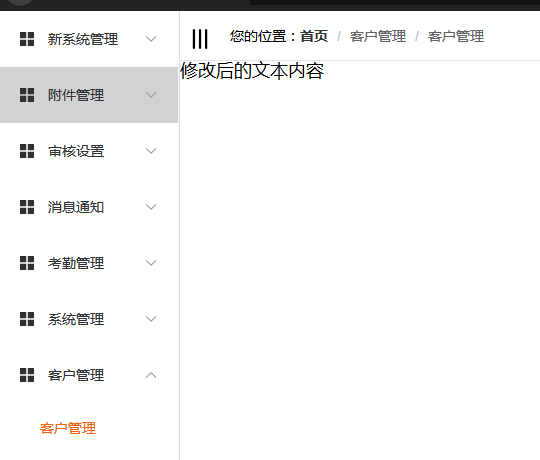


接下来可以添加自定义的界面内容。

例：



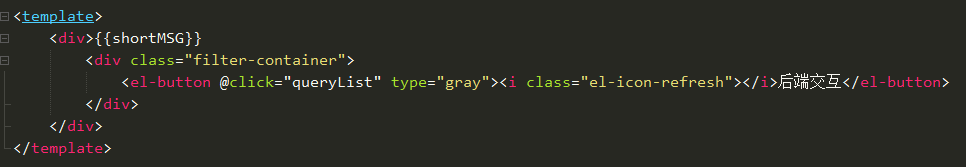
再次查看页面：



## 2.前端添加后端交互方法

2.1添加ajax请求方法

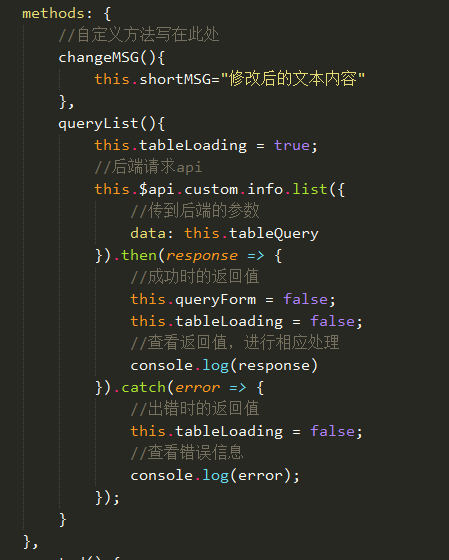
Html元素



Data参数

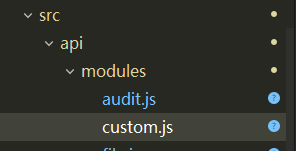


页面方法

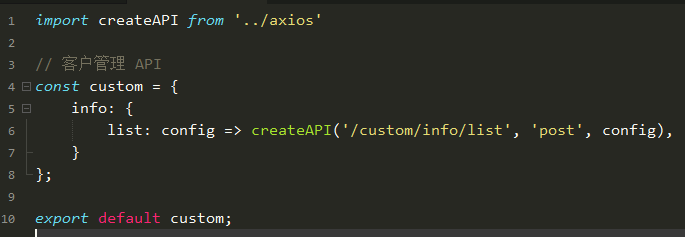


2.2配置ajax请求的后端地址

在src/api/modules下新建文件custom.js

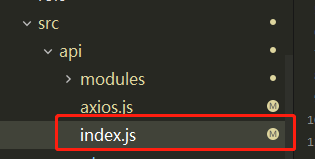


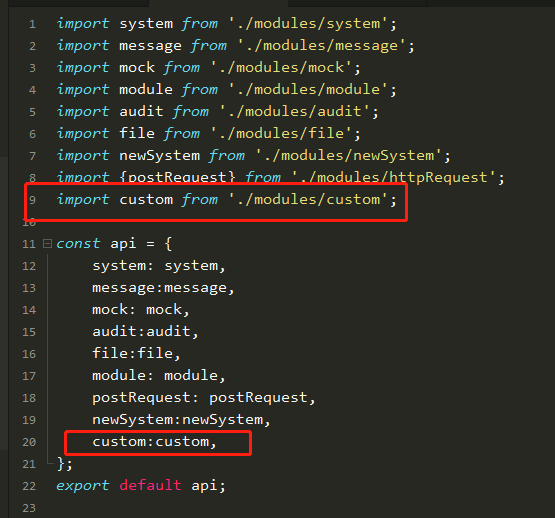
文件内容如下：



‘/custom/info/list’为指向后端的请求连接，自动拼接后端项目IP及端口。

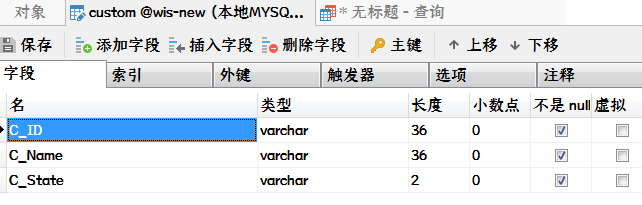
修改src/api/index.js，引入接口文件：



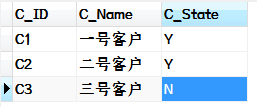


## 3.数据库创建对应表

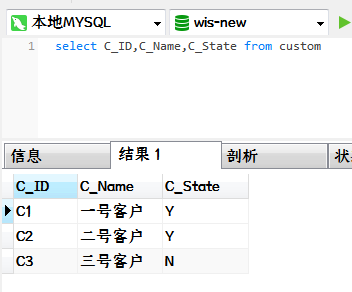
表结构如下：



填入测试数据：

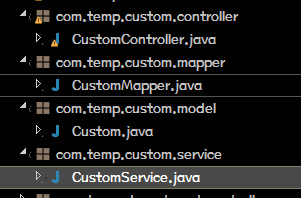


测试查询语句：



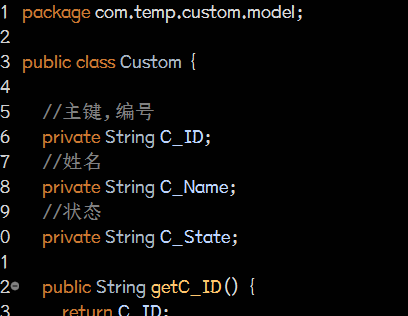
## 4.后端项目配置接受请求

3.1在src/main/java目录下创建文件，如下：

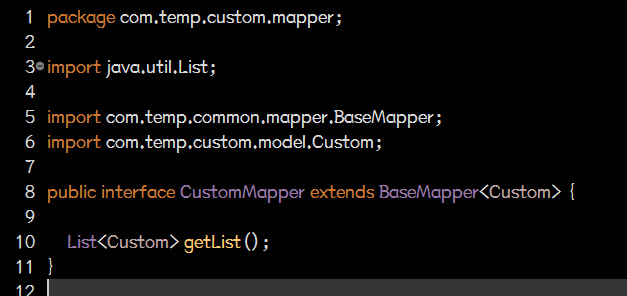


此为基础文件，在实际使用过程中根据需求可扩展或修改。

Custom.java中添加与数据库表对应的字段：



CustomMapper.java，接口文件，所有数据库接口都在此出现：



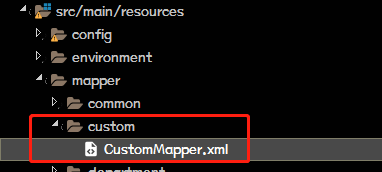
CustomService.java，数据库接口调用文件，此处将数据库数据转换为业务需要的格式：



CustomController.java，请求接收及处理文件，定义接收哪种前端请求：



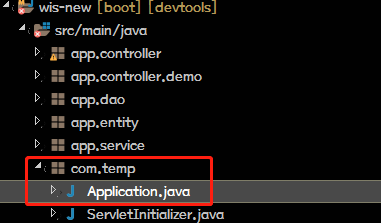
3.2在src/main/resources 目录下创建文件，如下：



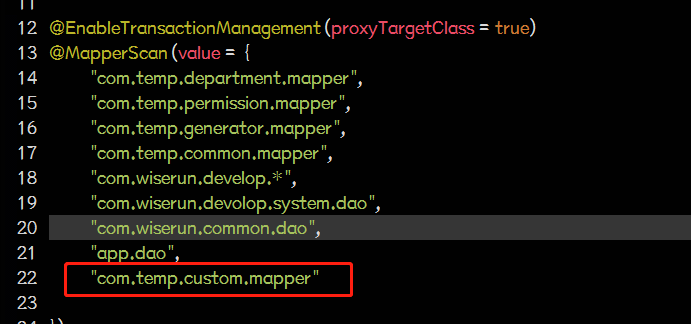
CustomMapper.xml为数据库查询语句配置：



添加Mapper扫描路径：



如果路径已被包含，则不需要添加



## 5.查看效果

5.1前台输出结果



打开调试模式，点击“后端交互”按钮。



5.2根据实际业务需求，获取信息并展示，如下图：



修改了custom.vue的内容：

