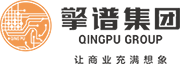
|  |
| --- |
| 智趣Mall系统 |
| 开发规范文档 |

|  |
| --- |
| 后台程序猿  2017-8-18 |



[1 项目svn地址 1](#_Toc491332258)

[2 开发软件 1](#_Toc491332259)

[2.1 IntelliJ IDEA 1](#_Toc491332260)

[2.2 Navicat for MySQL 1](#_Toc491332261)

[2.3 apache-maven-3.0.3 2](#_Toc491332262)

[2.4 jdk1.7.0\_67 4](#_Toc491332263)

[2.5 apache-tomcat-8.0.41 6](#_Toc491332264)

[3 项目文件结构说明 7](#_Toc491332265)

[4 功能开发流程 8](#_Toc491332266)

[4.1 决定功能归属点 8](#_Toc491332267)

[4.2 创建表（domain） 8](#_Toc491332268)

[4.2.1 表文件夹声明 9](#_Toc491332269)

[4.2.2 头部申明 9](#_Toc491332270)

[4.2.3 常用表字段 10](#_Toc491332271)

[4.2.3.1 主键id 10](#_Toc491332272)

[4.2.3.2 创建时间 10](#_Toc491332273)

[4.2.3.3 是否删除 10](#_Toc491332274)

[4.2.4 表字段常用约束 10](#_Toc491332275)

[4.3 创建查询Condition层 11](#_Toc491332276)

[4.4 创建Service，业务逻辑处理 11](#_Toc491332277)

[4.5 创建Controller层，提供接口 12](#_Toc491332278)

[4.5.1 使用规则 12](#_Toc491332279)

[4.5.2 Controller层返回json数据格式如下 13](#_Toc491332280)

[4.5.3 controller注册 13](#_Toc491332281)

[4.5.4 接口请求拦截 14](#_Toc491332282)

[4.5.4.1 逻辑实现文件位置 14](#_Toc491332283)

[4.5.4.2 不同客户端接口区分 14](#_Toc491332284)

[4.5.4.3 接口命名规则 15](#_Toc491332285)

[5 平台前端网页说明 15](#_Toc491332286)

[5.1 目录 15](#_Toc491332287)

[5.2 点击菜单事件 15](#_Toc491332288)

[5.3 http请求 16](#_Toc491332289)

# 项目svn地址

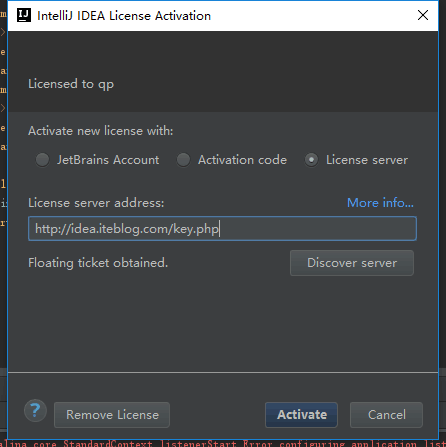
https://192.168.0.188/svn/QingPu/程序/mall/server/mall

# 开发软件

## IntelliJ IDEA

下载网页：<https://www.jetbrains.com/idea/download/#section=windows>

授权地址：http://idea.iteblog.com/key.php



## Navicat for MySQL

主机：192.168.0.29

用户名：root

端口：3306

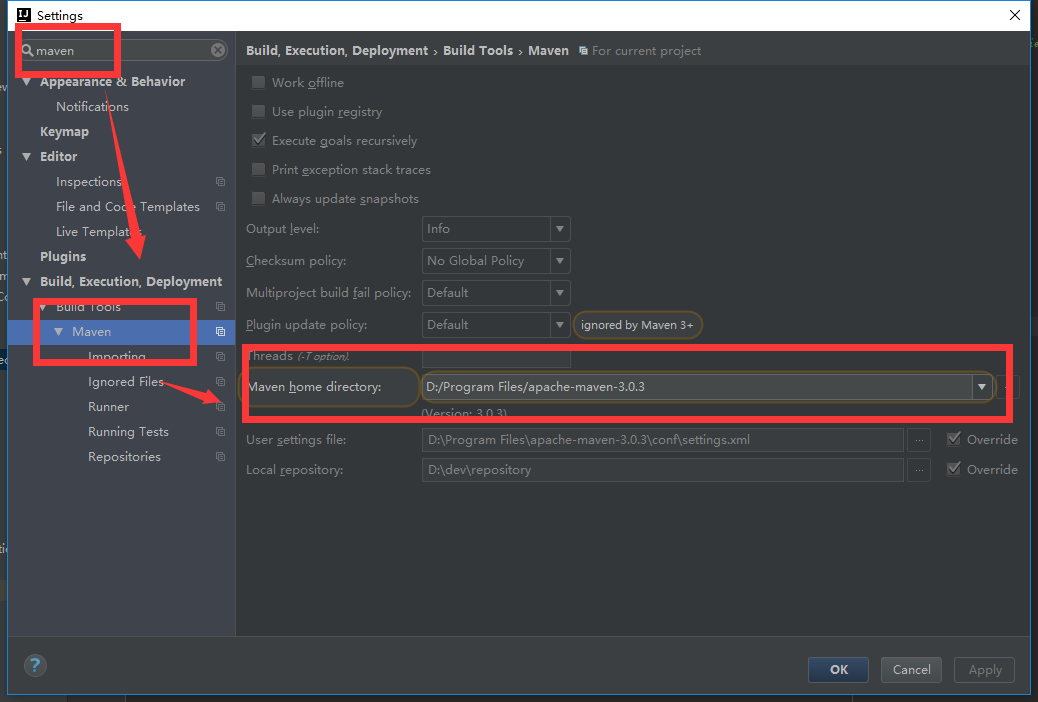
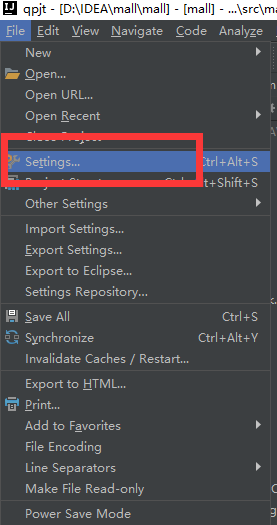
密码：123456



## apache-maven-3.0.3

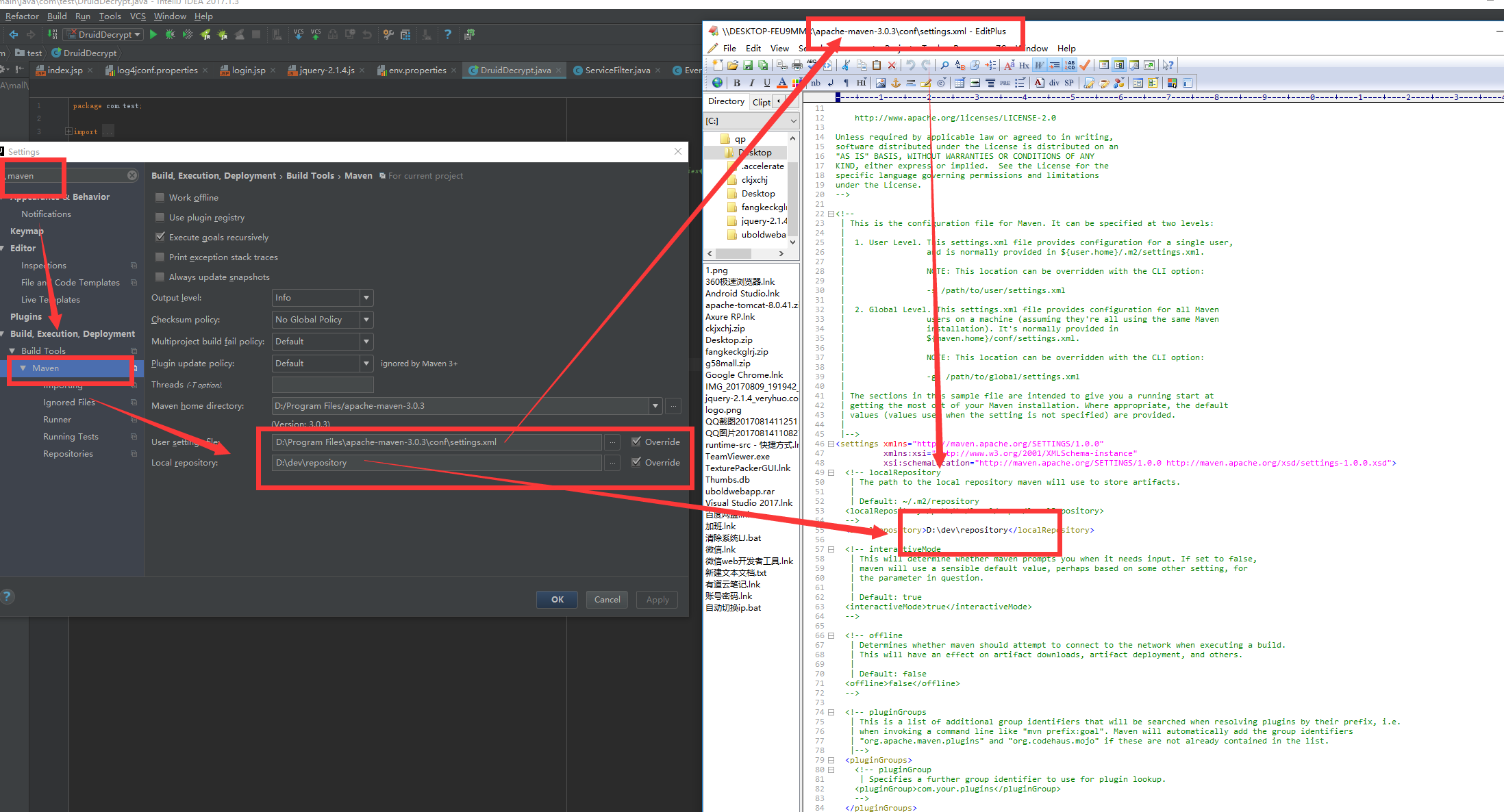
局域网路径：[\\DESKTOP-FEU9MMR\apache-maven-3.0.3](file:///\\DESKTOP-FEU9MMR\apache-maven-3.0.3)

配置路径：



架包库文件配置：

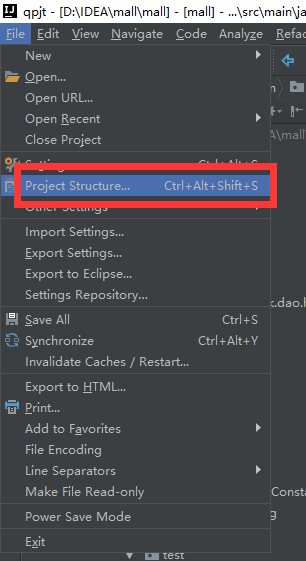
局域网路径：[\\DESKTOP-FEU9MMR\repository](file:///\\DESKTOP-FEU9MMR\repository)

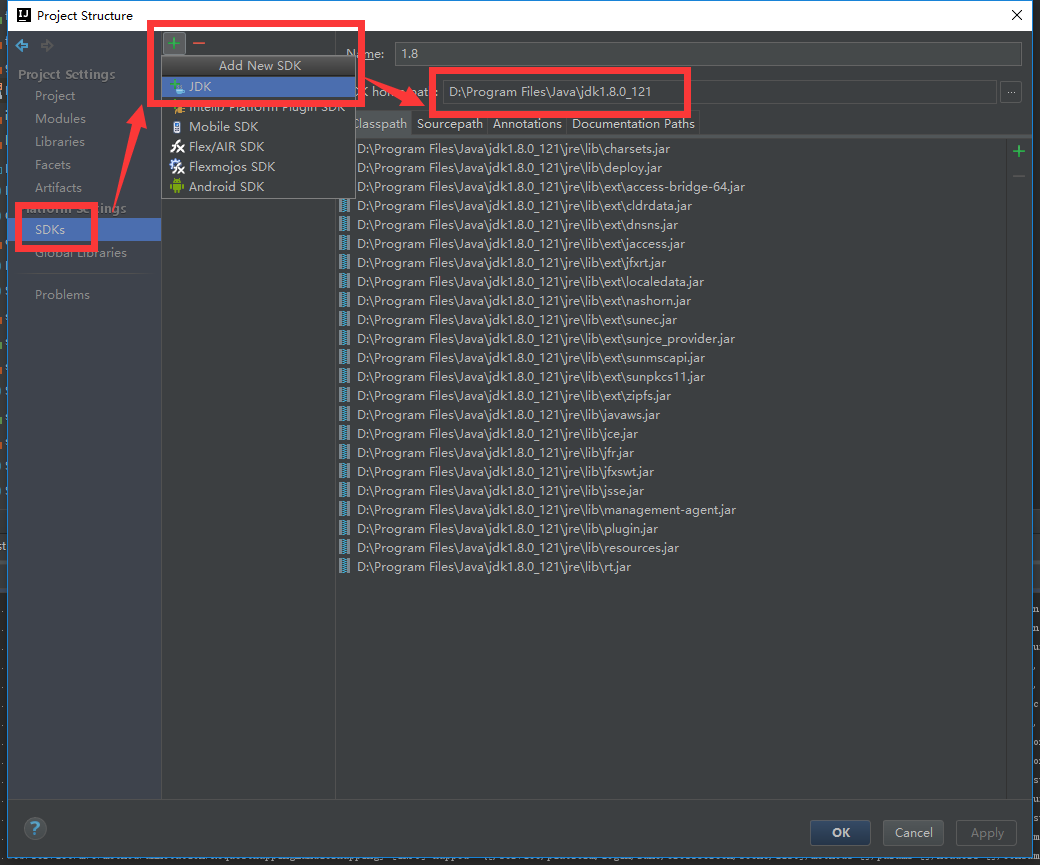


## jdk1.7.0\_67

局域网路径：[\\DESKTOP-FEU9MMR\jdk1.8.0\_121](file:///\\DESKTOP-FEU9MMR\jdk1.7.0_67)

配置：

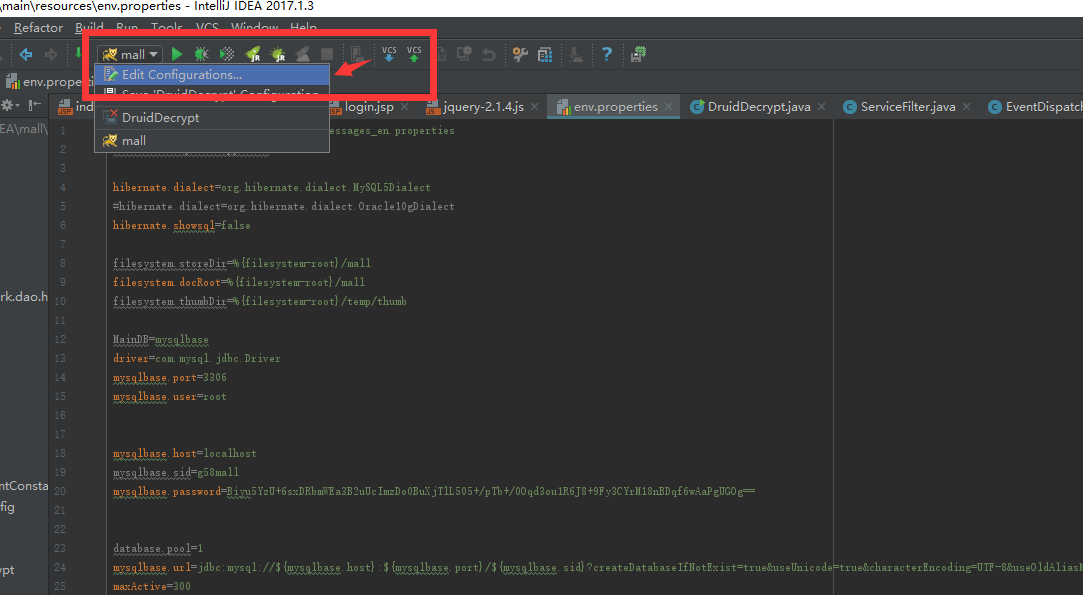


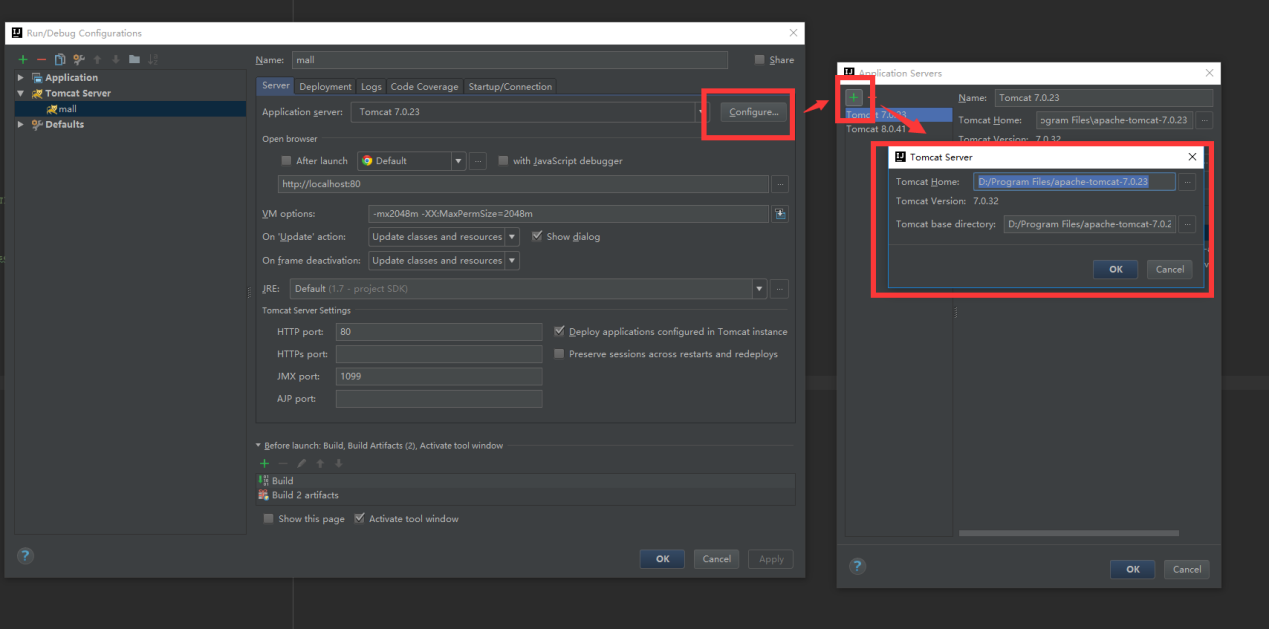


## apache-tomcat-8.0.41

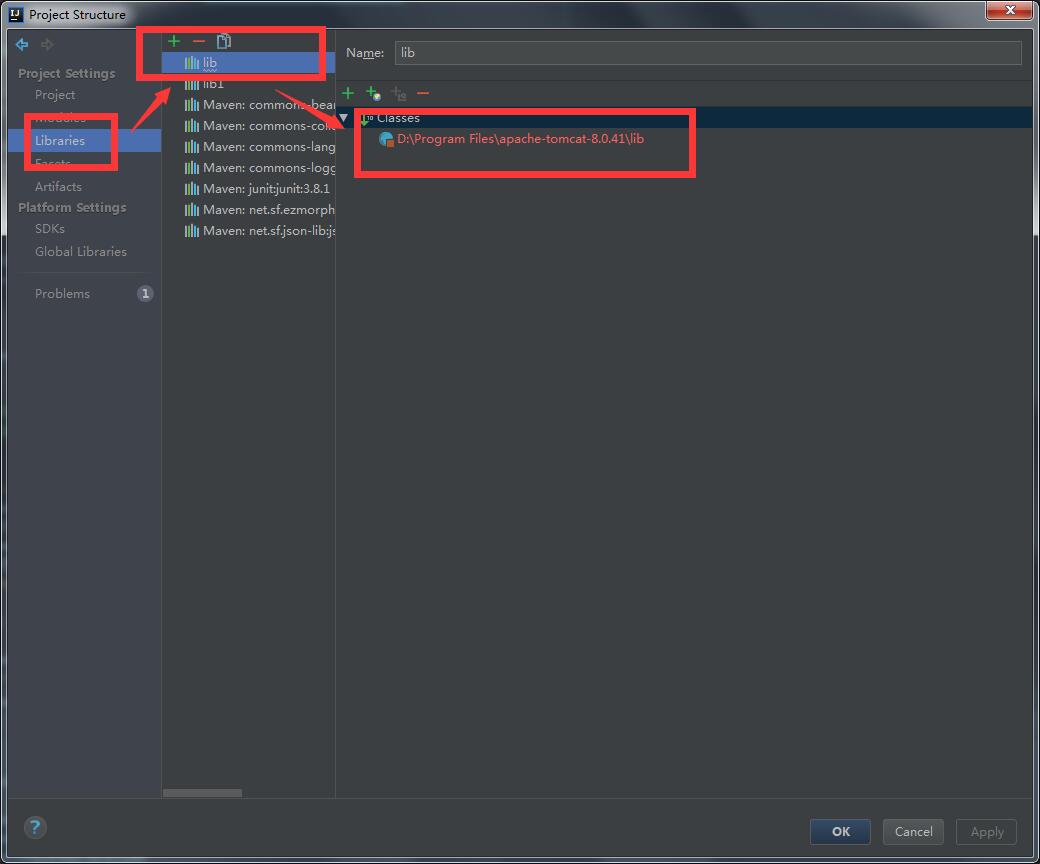
局域网路径：[\\DESKTOP-FEU9MMR\apache-tomcat-8.0.41](file:///\\DESKTOP-FEU9MMR\apache-tomcat-7.0.23)

配置：

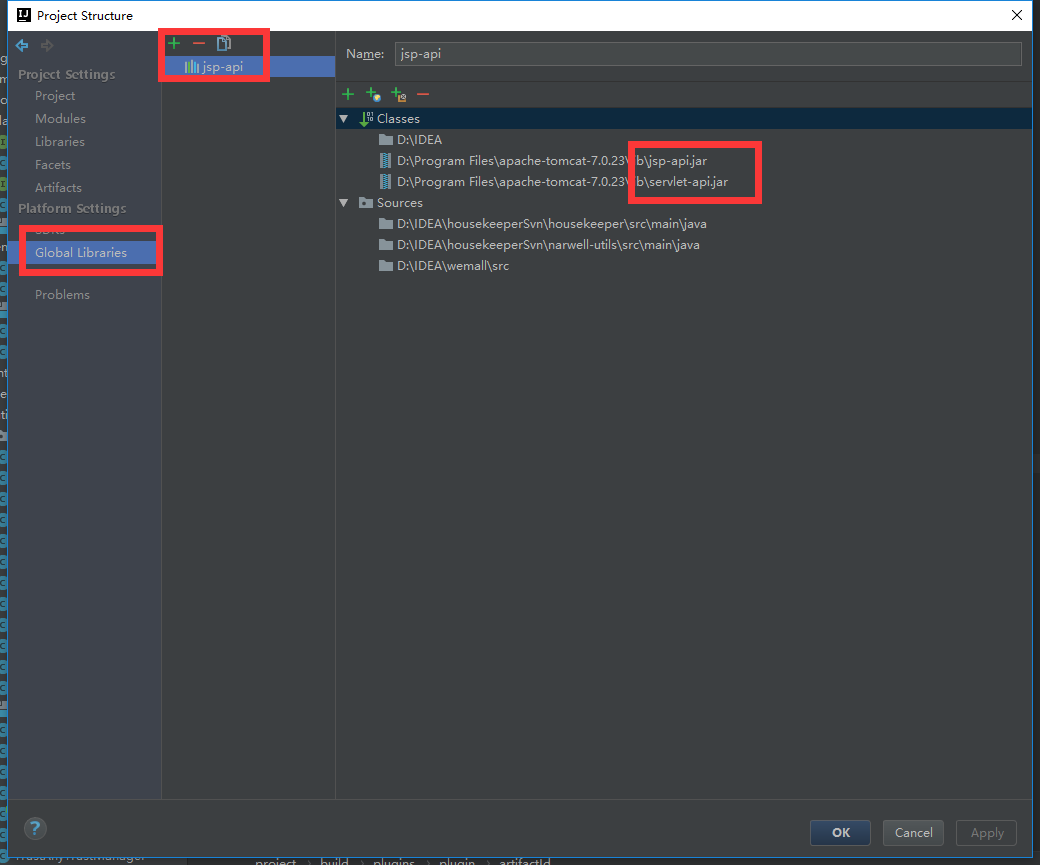




修改tomcat架包位置

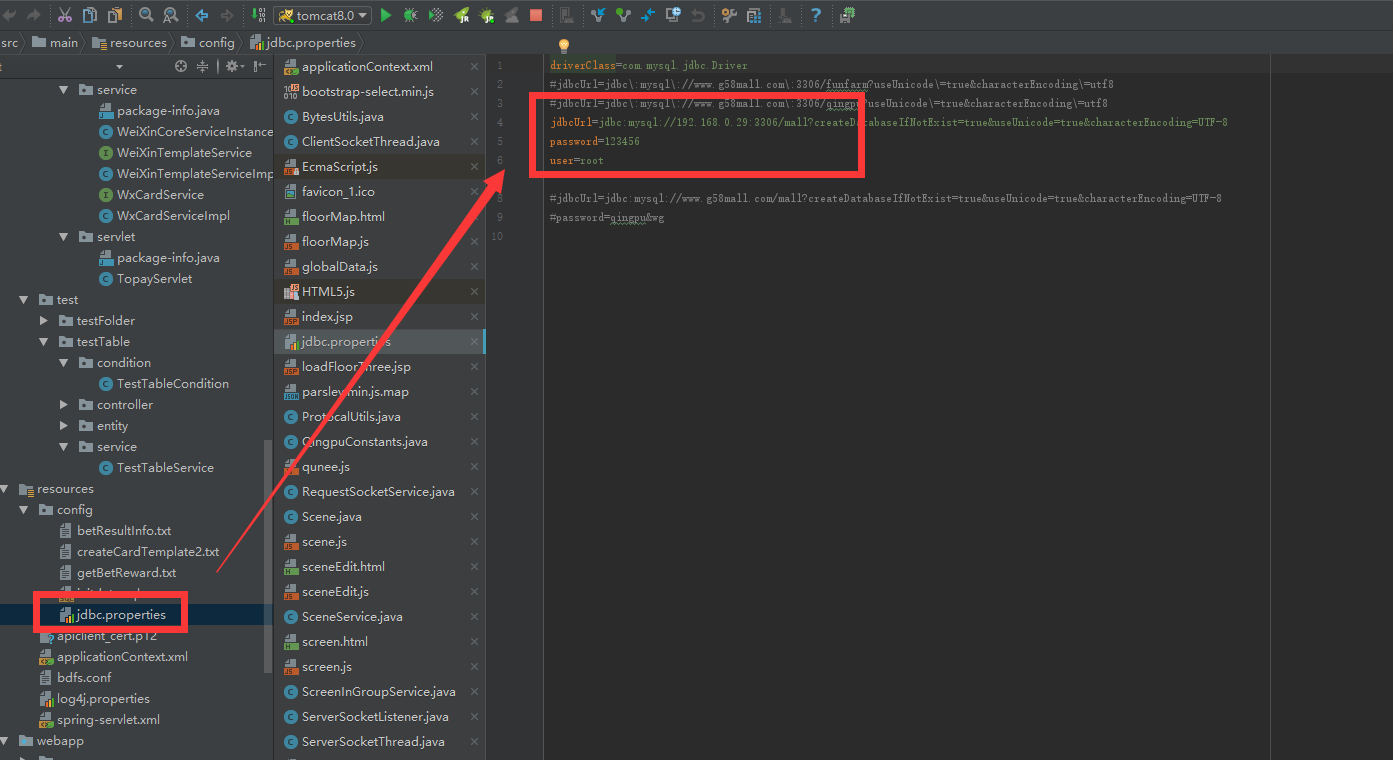


## 全局架包



# 项目文件结构说明

数据库连接：



# 功能开发流程

参考Test/testTable的文件结构

## 决定功能归属点

根据开发文档，知道功能点具体归属项目结构的位置。

## 创建表（entity）

### 头部申明

@Entity // 任何hibernte 映射对象都要有此注解

@Table(name = "xxx\_xxx") // 声明此对象映射到哪个表



### 常用表字段

#### 主键id

@Id

@Column(length = 32)

private String id; // 主键Id

#### 创建时间

private Date create\_time = new Date();

#### 是否删除

private Boolean is\_del = false; // 一般不要去删除数据，只是设置为删除字段为true。

### 表字段常用约束

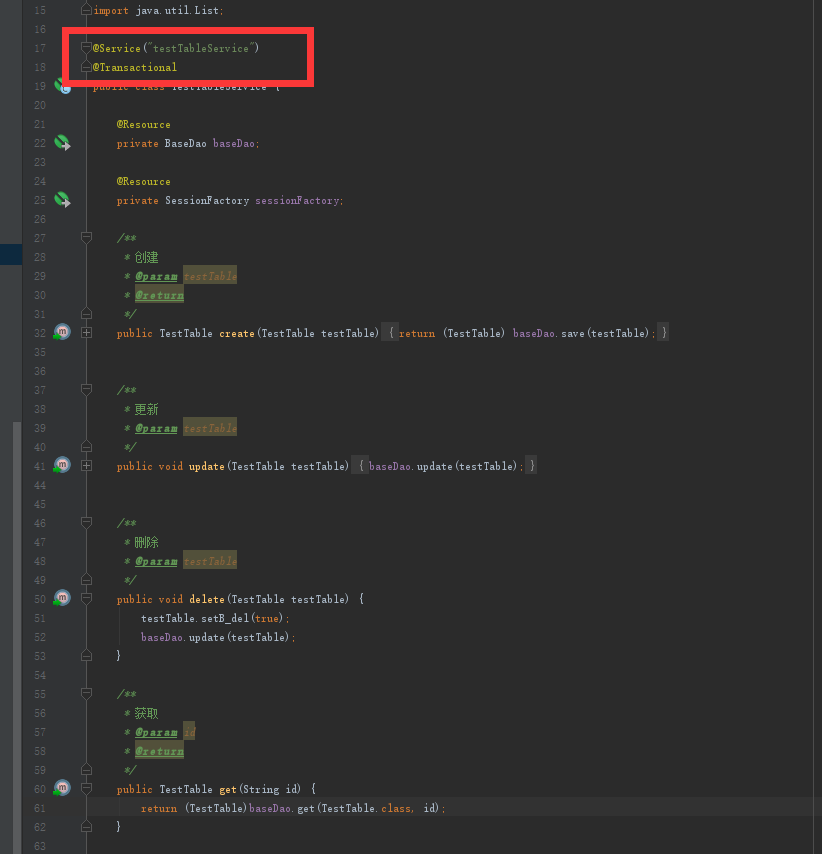
1. 索引：@Index(name = "I\_mac")
2. 长度：@Column(length = 32)
3. 主键：@Id
4. 字符串枚举类型：@Enumerated(value = EnumType.STRING)

## 创建查询Condition层



1. 继承于QPCondition
2. 写入一些需要的查询的字段。
3. 注意：super.initCriteria一定要写在initCriteria函数最后面。
4. 使用方法参考service对其的使用。

## 创建Service，业务逻辑处理



1. 注意红框中两个必须加上，缺一不可。
2. 实现增删改查基本四个函数。
3. 函数抛出异常调用方法为：

函数头部添加@Transactional(rollbackFor = Exception.class)

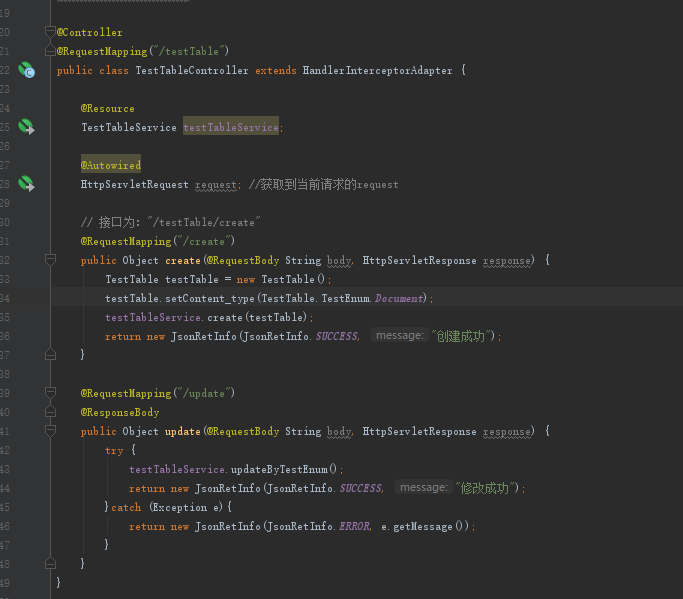
函数尾部添加throws Exception

函数抛出地方添加throw new Exception("未找到数据");

## 创建Controller层，提供接口

### 使用规则

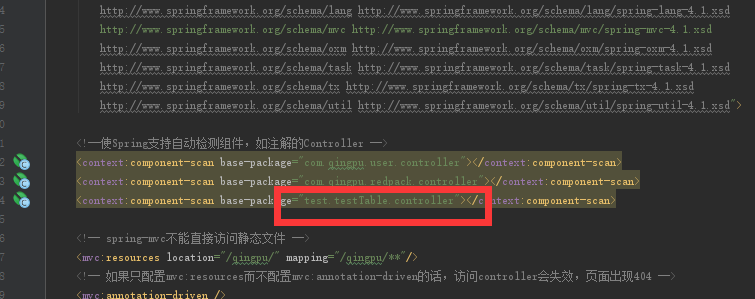
1. 使用@Controller注解声明一个Controller对象
2. @RequestMapping(“/path”)声明类的访问路径，访问路径命名规则参考接口拦截。
3. 类中的方法也使用@RequestMapping声明每个方法各自的访问路径
4. 方法参数中使用@RequestBody注解可以将封装过来的数据存储在所修饰的参数中
5. @ResponseBody指定方法返回值是json格式数据而不是被默认解析成跳转路径



### Controller层返回json数据格式如下

返回数据使用JsonRetInfo对象进行封装，需要填充code，message和Object data

### controller注册

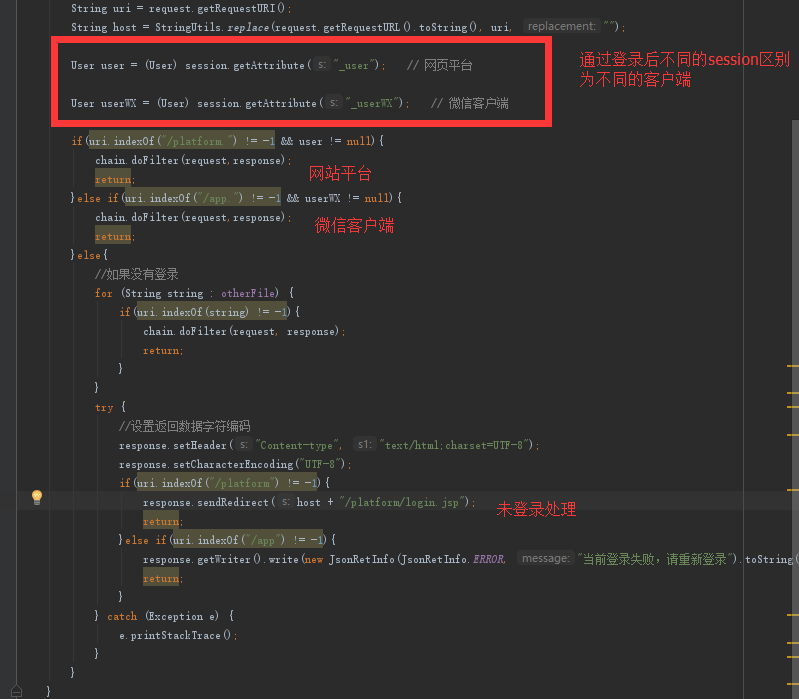


### 接口请求拦截

#### 逻辑实现文件位置

filter下的ServiceFilter

#### 不同客户端接口区分



#### 接口命名规则

1. 未登录：

文件命名：NoLogin + 文件夹名 + Controller，如：NoLoginUserController。

接口命名：@RequestMapping("/service/platform/noLogin/user")

未登录接口要去ServiceFilter文件24行声明，otherFile.add("/service/platform/noLogin");

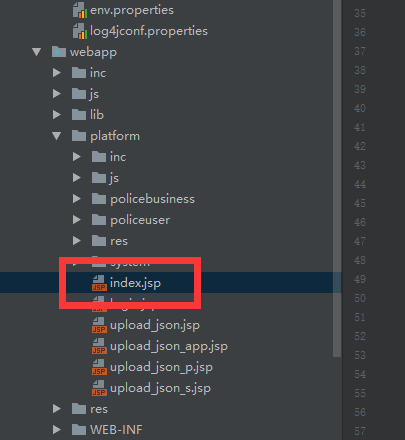
1. 登录

文件命名：文件夹名 + Controller，如：UserController。

接口命名：@RequestMapping("/service/platform/login/user")

# 平台前端网页说明

## 目录



## 点击菜单事件

1. 由index.jsp作为主页面，点击菜单时，通过菜单url值如：user/user。说明是：加载platform/page下的user文件夹中的user.js和user.html。只需关心这两个文件业务逻辑的实现。
2. demo.html和demo.js有一些常用的控件定义。

## http请求

### js代码

getDataByURL("/service/platform/login/mall/screen/delete", {screenIds:screenIds}, function (data) {

$("#editScreen").modal("hide");

$('#table').bootstrapTable("refresh");

})

### java接口规则

@RequestMapping("/create")

public void create(TestTable testTable, HttpServletResponse response) throws IOException {

try{

testTableService.create(testTable);

JsonRetInfo.returnSuccess(response, "操作成功");

}catch (Exception e){

String error = e.getMessage();

if(CommonUtils.checkIsNotUniqueForEntity(error, "num")){

error = "编号已存在，不能重复创建";

}

JsonRetInfo.returnFail(response, "操作失败：" + error);

}

}

注意参数传入，以及接口返回规则。

## table显示

### js代码

TableFunc.showTable({

url:"/service/platform/login/mall/screenGroup/list",

queryParams:queryScreenGroupParams,

elm:"screenGroupTable",

columns: [

{

field : 'name',

title : '组名',

align : 'center',

valign : 'middle',

},

{

field : 'des',

title : '描述',

align : 'center',

valign : 'middle',

},

{

field : 'screen\_count',

title : '屏幕数',

align : 'center',

valign : 'middle',

},

{

title : '编辑',

align : 'center',

valign : 'middle',

formatter: function (value, row, index) {

var a = '<button type="button" class="btn btn-success waves-effect waves-light" onclick="editGroup(' + index + ')">编辑</button>';

return a;

}

}]

});

### java接口规则

@RequestMapping("/list")

@ResponseBody

public Object list(@RequestBody TableParams tableParams, HttpServletResponse response) {

PaginationParams paginationParams = tableParams.getPaginationParams();

Range range = paginationParams.getRange();

Sorter sorter = paginationParams.getSorter();

TestTableCondition testTableCondition = new TestTableCondition();

PaginationSupport<TestTable> testTablePaginationSupport = testTableService.list(testTableCondition, range, sorter);

return new JsonRetInfo(JsonRetInfo.SUCCESS, "操作成功", CommonUtils.getPageMap(testTablePaginationSupport));

}

注意参数传入固定为TableParams，返回规则

## input的select加载

### 表中枚举类型

1. index.jsp修改
2. 顶部加入：String deviceTypesStr = CommonUtils.getEnumKeyValueString("deviceType");
3. 下面加入GlobalData.deviceTypes = JSON.parse('<%= deviceTypesStr%>');
4. utils.js修改

TableFunc加入函数

screenType:function (value, row, index) {

return TableFunc.formatter.\_enumType(value, "screenType")

},

1. 调用：

input调用：

SelectPickerFunc.loadLocalData('screenType', GlobalData.screenTypes)

table调用：

{

field : 'type',

title : '设备种类',

align : 'center',

formatter: TableFunc.formatter.deviceType

},

### 表数据加入select

1. java代码

@RequestMapping("/getFloorSelect")

public void getFloorSelect(HttpServletResponse response) throws IOException {

try{

List<Floor> floorList = floorService.list( new FloorCondition());

Map<String, Integer> map = new HashMap<>();

for(Floor floor : floorList){

map.put(floor.getId(), floor.getNum());

}

List<Map.Entry<String, Integer>> infoIds =

new ArrayList<>(map.entrySet());

Collections.sort(infoIds, new Comparator<Map.Entry<String, Integer>>() {

public int compare(Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2) {

return o1.getValue().compareTo(o2.getValue());

}

});

JsonRetInfo.returnSuccess(response, "获取成功", infoIds);

}catch (Exception e){

JsonRetInfo.returnFail(response, "获取失败：" + e.getMessage());

}

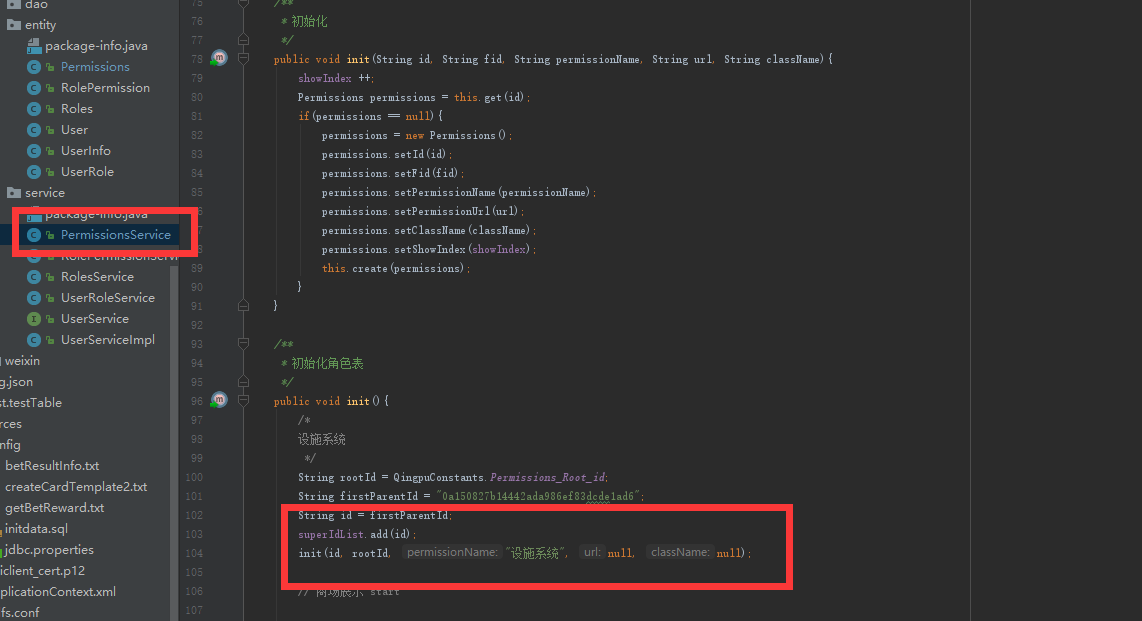
}

1. js调用

SelectPickerFunc.loadUrlData('floorIds', "/service/platform/login/mall/floor/getFloorSelect", true)

## 添加菜单页面

### 注册菜单

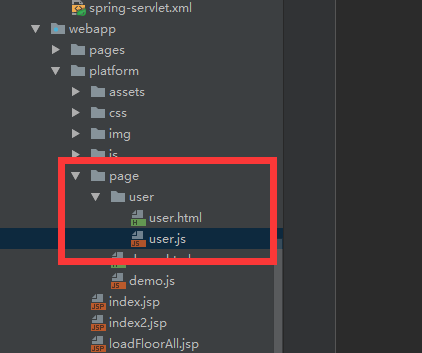


id：菜单uuid

fid：父级菜单uuid

className：一级菜单图标

url：统一放在webapp/platform/page里面



图中url则为：user/user（不需要包括html或者js）