

# 《锋迷商城》项目

## 1. 《锋迷商城》设计及实现---用户管理

---

### 1.1 前端页面登录的实现

```
<!DOCTYPE html>
<html>
  <head lang="en">
    <meta charset="UTF-8" />
    <title>登录</title>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0, minimum-
scale=1.0, maximum-scale=1.0, user-scalable=no"
    />
    <meta name="format-detection" content="telephone=no" />
    <meta name="renderer" content="webkit" />
    <meta http-equiv="Cache-Control" content="no-siteapp" />

    <link rel="stylesheet" href="static/css/amazeui.css" />
    <link href="static/css/dlstyle.css" rel="stylesheet"
type="text/css" />
    <link rel="stylesheet" href="static/css/bootstrap.min.css"
/>
  </head>

  <body>
    <div class="login-boxtitle" style="height: 100px">
      <div class="logoBig">
        
      </div>
    </div>
  </div>
```

```
<div class="login-banner">
  <div class="login-main">
    <div class="login-banner-bg">
      <span></span>
    </div>
    <div class="login-box" style="margin-top: 20px"
id="container">
      <h3 class="title">登录商城</h3>
      <h5 style="color: red" id="tips">{{tips}}</h5>

      <div class="clear"></div>

      <div class="login-form" style="background: none;
margin-top: 15px">
        <form>
          <div class="user-name" style="margin-top: 20px">
            <label for="user"
              ><span
                class="glyphicon glyphicon-user"
                aria-hidden="true"
              ></span>
            ></label>
            <input
              type="text"
              name=""
              id="username"
              v-model="username"
              placeholder="邮箱/手机/用户名"
              @keyup="checkInfo"
            />
          </div>
          <div class="user-pass" style="margin-top: 20px">
            <label for="password"
              ><span
                class="glyphicon glyphicon-lock"
                aria-hidden="true"
              ></span>
            ></label>
            <input
              type="password"
```

```

        name=""
        id="password"
        v-model="password"
        placeholder="请输入密码"
        @keyup="checkInfo"
    />
</div>
</form>
</div>

<div class="login-links">
    <label for="remember-me"
        ><input id="remember-me" type="checkbox" />记住
密码</label
    >
    <a href="#" class="am-fr">忘记密码</a>
    <br />
</div>
<div class="am-cf">
    <input
        type="button"
        id="submitBtn"
        name=""
        @click="doSubmit"
        value="登 录"
        class="am-btn am-btn-primary am-btn-sm"
    />
</div>
<div class="am-cf">
    <input
        type="button"

onclick="javascript:window.location.href='register.html'"
        value="注 册"
        class="am-btn am-btn-default am-btn-sm"
    />
</div>
<div class="partner"></div>
</div>
</div>

```

```

<div class="footer">
  <div class="footer-hd"></div>
  <div class="footer-bd">
    <p>
      <a href="#">联系我们</a>
      <a href="#">网站地图</a>
    </p>
  </div>
</div>
</body>
<script type="text/javascript" src="static/js/vue.js">
</script>
<script type="text/javascript" src="static/js/axios.min.js">
</script>
<script type="text/javascript" src="static/js/base.js">
</script>
<script>
  //创建vue实例   var(全局 、局部) let(局部变量定义)   const(常量
一般定义对象)
  const vm = new Vue({
    el: "#container",
    data: {
      username: "",
      password: "",
      tips: "",
      isRight: false,
    },
    methods: {
      doSubmit: function () {
        if (vm.isRight) {
          var url = baseUrl + "user/login";
          axios
            .get(url, {
              params: {
                username: vm.username,
                password: vm.password,
              },
            })
            .then((result) => {
              var vo = result.data;

```

面跳转

```
        if (vo.code == 10000) {
            //登录成功
            window.location.href = "index.html"; //js 页面跳转
        } else {
            vm.tips = "登录失败，账号或者密码错误!";
        }
    });
} else {
    vm.tips = "请正确输入账号和密码! ";
}
},
//验证用户名和密码
checkInfo: function () {
    if (vm.username == "") {
        vm.tips = "请输入账号";
        vm.isRight = false;
    } else if (vm.username.length < 6 ||
vm.username.length > 10) {
        vm.tips = "账号长度必须为6-10个字符";
        vm.isRight = false;
    } else {
        //账号合法，校验密码
        if (vm.password == "") {
            vm.tips = "请输入密码";
            vm.isRight = false;
        } else if (vm.password.length < 6 ||
vm.password.length > 10) {
            vm.tips = "密码长度必须为6-10个字符";
            vm.isRight = false;
        } else {
            vm.tips = "";
            vm.isRight = true;
        }
    }
},
},
});
</script>
</html>
```

## 1.2 后端状态码工具类

```
package org.qf.utils;

/**
 * 状态码
 */
public class ResStatus {

    public static final int OK=10000;

    public static final int NO=10001;

    public static final int LOGIN_SUCCESS=20000; //认证成功

    public static final int LOGIN_NOT=20001; //用户未登录

    public static final int LOGIN_Fail_OVERDUE=20002; //用户登录失效

}
```

## 1.3 前端页面注册的实现

```
<!DOCTYPE html>
<html>
  <head lang="en">
    <meta charset="UTF-8" />
    <title>注册</title>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0, minimum-
scale=1.0, maximum-scale=1.0, user-scalable=no"
    />
    <meta name="format-detection" content="telephone=no" />
    <meta name="renderer" content="webkit" />
    <meta http-equiv="Cache-Control" content="no-siteapp" />

    <link rel="stylesheet" href="static/css/amazeui.css" />
```

```

    <link href="static/css/dlstyle.css" rel="stylesheet"
type="text/css" />

    <link rel="stylesheet" href="static/css/bootstrap.min.css"
/>
</head>

<body>
    <div class="login-boxtitle" style="height: 100px">
        <div class="logoBig">
            
        </div>
    </div>

    <div class="login-banner" style="background: rgb(142, 213,
21)">
        <div class="login-main">
            <div class="login-banner-bg">
                <span></span>
            </div>
            <div class="login-box" id="container">
                <h3 class="title">用户注册</h3>
                <h5 :style="colorStyle" id="tips">{{tips}}</h5>
                <div class="clear"></div>

                <div class="login-form" style="background: none">
                    <form>
                        <div class="user-name">
                            <label for="user"
                                ><span
                                    class="glyphicon glyphicon-user"
                                    aria-hidden="true"
                                ></span>
                            ></label>
                            <input
                                type="text"
                                name=" "

```

```

        id="user"
        v-model="user.username"
        placeholder="邮箱/手机/用户名"
        @keyup="checkRegitInfo"
    />
</div>
<div class="user-pass" style="margin-top: 15px">
    <label for="password"
        ><span
            class="glyphicon glyphicon-lock"
            aria-hidden="true"
        ></span>
    ></label>
    <input
        type="password"
        name=""
        id="password"
        placeholder="请输入密码"
        v-model="user.password"
        @keyup="checkRegitInfo"
    />
</div>
<div class="user-pass" style="margin-top: 15px">
    <label for="password"
        ><span
            class="glyphicon glyphicon-lock"
            aria-hidden="true"
        ></span>
    ></label>
    <input
        type="password"
        name=""
        id="repassword"
        placeholder="请再次输入密码"
        v-model="user.repassword"
        @keyup="checkRegitInfo"
    />
</div>
</form>
</div>

```



```
<div class="login-links">
  <br />
</div>
<div class="am-cf">
  <input
    type="button"
    name=""
    value="提交注册"
    class="am-btn am-btn-primary am-btn-sm"
    @click="doRegist"
  />
</div>
<div class="partner"></div>
</div>
</div>

<div class="footer">
  <div class="footer-hd"></div>
  <div class="footer-bd">
    <p>
      <a href="#">联系我们</a>
      <a href="#">网站地图</a>
    </p>
  </div>
</div>
</body>
<script type="text/javascript" src="static/js/vue.js">
</script>
<script type="text/javascript" src="static/js/axios.min.js">
</script>
<script type="text/javascript" src="static/js/base.js">
</script>
<script>
  const vm = new Vue({
    el: "#container",
    data: {
      user: {
        username: "",
        password: "",
        repassword: "",

```

```

    },
    tips: "",
    colorStyle: "color:red",
    isRight: false,
  },
  methods: {
    doRegist: function () {
      if (!vm.isRight) {
        vm.tips = "请确认注册信息输入完整且正确! ";
      } else {
        var url = baseUrl + "user/regist";
        axios
          .post(url, vm.user)
          .then((result) => {
            var vo = result.data;
            if (vo.code == 10000) {
              //注册成功
              vm.tips = "恭喜您, 注册成功!";
              vm.colorStyle = "color:green"; //定时器 3秒之后跳转

              setTimeout(function () {
                window.location.href = "login.html";
              }, 3000);
            } else {
              vm.tips = "很遗憾, 注册失败! ";
            }
          })
          .catch((err) => {});
      }
    },
    //校验
    checkRegitInfo: function () {
      //校验账号
      if (vm.user.username == "") {
        (vm.tips = "请输入账号"), (vm.isRight = false);
      } else if (
        vm.user.username.length < 6 ||
        vm.user.username.length > 10
      ) {
        vm.tips = "账号长度必须为6-10个字符";
        vm.isRight = false;
      }
    }
  }
}

```

```

    } else {
      //校验密码
      if (vm.user.password == "") {
        vm.tips = "请输入密码";
        vm.isRight = false;
      } else if (
        vm.user.password.length < 6 ||
        vm.user.password.length > 10
      ) {
        vm.tips = "密码长度必须为6-10个字符";
        vm.isRight = false;
      } else {
        //校验重复密码
        if (vm.user.repassword == "") {
          vm.tips = "请再次输入密码";
          vm.isRight = false;
        } else if (vm.user.repassword !=
vm.user.password) {
          vm.tips = "两次密码不一致";
          vi.isRight = false;
        } else {
          vm.tips = "";
          vm.isRight = true;
        }
      }
    }
  },
},
});
</script>
</html>

```

## 2. 前后端分离用户认证-JWT

### 2.1 基于Session实现单体项目用户认证

在单体项目中如何保证受限资源在用户未登录的情况下允许访问？

在单体项目中，视图资源（页面）和接口（控制器）都在一台服务器上，用户的多次请求都是基于同一个会话（session），因此可以借助session来进行用户判断。

1. 当用户登录成功之后，将用户信息存放到session中
2. 当用户再次访问受限资源时，验证session中是否存在用户对象，可以通过用户信息来判断。

## 2.2 基于token实现前后端分离用户认证

由于在前后端分离项目开发中，前后端之间是通过异步交互完成数据访问的，请求时无状态的，因此不能基于 session实现用户的认证。

## 2.3 基于token的用户认证实现

### 1. A页面

```
var url = baseUrl + "user/login";
axios
  .get(url, {
    params: {
      username: vm.username,
      password: vm.password,
    },
  })
  .then((result) => {
    var vo = result.data;
    if (vo.code == 10000) {
      //登录成功
      //放到cookie localStorage
      localStorage.setItem("token", JSON.stringify(vo.msg));
      window.location.href = "index.html"; //js 页面跳转
    } else {
      vm.tips = "登录失败，账号或者密码错误!";
    }
  });
} else {
  vm.tips = "请正确输入账号和密码!";
}
```

### 2. B页面

```

</script>
    window.jQuery || document.write('<script src="static/js/jquery.min.js "><
</script>
<script>
    var jsonStr=localStorage.getItem("token");
    var token=eval("(" +jsonStr+")");
    if(token==null){
        //跳转到登录页面,
    }else{
        //跳转到服务器查询 数据并把数据展示到页面中
    }
</script>

```

### 3. 后端（业务实现类）

```

47 //登录
48 @Override
49 public ResultVO checkLogin(String name, String pwd) {
50     Users user = usersMapper.query(name);
51     if(user==null){
52         return new ResultVO(ResStatus.NO, msg: "登录失败, 用户名不存在!", data: null);
53     }else{
54         String md5Pwd=MD5Utils.md5(pwd);
55         if(md5Pwd.equals(user.getPassword())){
56             //登录成功, 则需要生成token(token就是按照特定规则生成的字符串) md5 (加密) base64(加密, 解密)
57             String token= Base64Utils.encode( msg: name+"QIANfeng666");
58             return new ResultVO(ResStatus.OK,token,user);
59         }else{
60             return new ResultVO(ResStatus.NO, msg: "登录失败, 密码错误!", data: null);
61         }
62     }
63 }

```

## 2.4 JWT

如果按照上述规则生成token:

1. 简易的token生成规则安全性较差, 如果要生成安全性很高的token对加密算法要求较高。
2. 无法完成时效性的校验 (登录过期)

### 2.4.1 JWT简介

- JWT: Json Web Token
- 官网地址: <https://jwt.io/>

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG91IiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret ☐ base64 ☐ encoded
```



## 2.4.2 生成JWT

- 添加依赖

```
<dependency>
  <groupId>com.auth0</groupId>
  <artifactId>java-jwt</artifactId>
  <version>4.0.0</version>
</dependency>

<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt</artifactId>
  <version>0.9.1</version>
</dependency>
```

- 生成token

```

//使用jwt规则生成token字符串
        JwtBuilder jwtBuilder= Jwts.builder();
        String token=jwtBuilder.setSubject(name).
//主题，就是token携带的数据
        setIssuedAt(new Date()). //设置token的生成时间
        setId(user.getUserId()+"") // 设置用户id token
id
        .setExpiration(new
Date(System.currentTimeMillis()+24*60*60*1000)) //设置token过
期时间

        .signWith(SignatureAlgorithm.ES256,"QIANfeng666") //设置加密方
式和加密密码

        .compact();

```

### 2.4.3 拦截器校验Token

- 创建拦截器

```

package org.qf.interceptor;

import com.fasterxml.jackson.databind.ObjectMapper;
import io.jsonwebtoken.*;
import org.qf.utils.ResStatus;
import org.qf.utils.ResultVO;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.HandlerInterceptor;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.PrintWriter;

/**
 * 拦截校验token
 */
@Component
public class CheckTokenInterceptor implements
HandlerInterceptor {

    @Override

```

```

    public boolean preHandle(HttpServletRequest request,
        HttpServletResponse response, Object handler) throws Exception
    {
        //校验token
        String method=request.getMethod();
        if("OPTIONS".equalsIgnoreCase(method)){
            return true;
        }
        String token=request.getHeader("token");
        if(token==null){
            ResultVO resultVO=new
ResultVO(ResStatus.LOGIN_NOT,"请先登录",null);
            doResponse(response,resultVO);
        }else{
            try {
                JwtParser parser= Jwts.parser();
                parser.setSigningKey("QIANfeng666"); //解析
token的setSigningKey必须和生成token时设置密码一致
                //如果正确（有效期内）则正常 执行，否则抛异常
                Jws<Claims>
claimsJws=parser.parseClaimsJws(token);
                return true;
            }catch(ExpiredJwtException e){
                ResultVO resultVO=new
ResultVO(ResStatus.LOGIN_Fail_OVERDUE,"登录过期，请重新登
录",null);
                doResponse(response,resultVO);
            }catch (UnsupportedJwtException e){
                ResultVO resultVO=new
ResultVO(ResStatus.LOGIN_NOT,"Token不合法，请自动",null);
                doResponse(response,resultVO);
            }catch (Exception e){
                ResultVO resultVO=new
ResultVO(ResStatus.LOGIN_NOT,"请先登录！ ",null);
                doResponse(response,resultVO);
            }
        }
        return false;
    }
}

```



```

        private void doResponse(HttpServletResponse
response, ResultVO resultVO) throws Exception{
            response.setContentType("application/json");
            response.setCharacterEncoding("utf-8");
            PrintWriter out=response.getWriter();
            String s=new
ObjectMapper().writeValueAsString(resultVO);
            out.print(s);
            out.flush();
            out.close();
        }
    }
}

```

- 配置拦截器

```

package org.qf.config;

import org.qf.interceptor.CheckTokenInterceptor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.InterceptorR
egistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfig
urer;

@Configuration
public class InterceptorConfig implements WebMvcConfigurer {

    @Autowired
    private CheckTokenInterceptor checkTokenInterceptor;

    @Override
    public void addInterceptors(InterceptorRegistry registry)
    {
        registry.addInterceptor(checkTokenInterceptor)
            .addPathPatterns("/**")
            .addPathPatterns("/user/**"); //拦截路径

    }
}

```

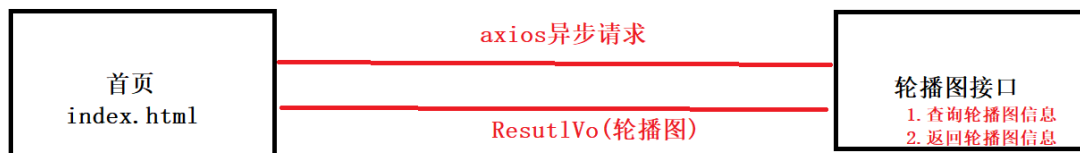
## 2.5 请求头传递token

前端但凡访问受限资源，都必须携带token发送请求；token可以通过请求行（params）、请求头（header）以及请求体（data）数据，但是习惯性使用header传递

## 3. 首页-轮播图

### 3.1 实现流程分析

- 实现流程图



- 接口

- 查询轮播图信息返回

### 3.2 数据库设计

保存

添加字段

插入字段

删除字段

主键

上移

下移

字段

索引

外键


检查

触发器

选项

注释

SQL 预览

名	类型	长度	小数点	不是 null	虚拟	键	注释
img_id	varchar	64		<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	主键
img_url	varchar	255		<input checked="" type="checkbox"/>	<input type="checkbox"/>		图片地址
img_bg_color	varchar	32		<input type="checkbox"/>	<input type="checkbox"/>		背景颜色
prod_id	varchar	64		<input type="checkbox"/>	<input type="checkbox"/>		商品id
category_id	varchar	64		<input type="checkbox"/>	<input type="checkbox"/>		商品分类id
index_type	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>		轮播图的类型，用于判断，可以根据商品id或者分类进行页面跳转 1: 商品 2
seq	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>		轮播图展示顺序，从小到大
status	int			<input checked="" type="checkbox"/>	<input type="checkbox"/>		是否展示，1 表示正常显示 0 表示下线 1: 展示 0: 不展示
create_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		创建时间
update_time	datetime			<input type="checkbox"/>	<input type="checkbox"/>		更新时间

开始事务

文本

筛选

排序

导入

导出

数据生成

创建图表

img_id	img_url	img_bg_color	prod_id	category_id	index_type	seq	status	create_time	update_time
1	ad1.jpg	green	101	(Null)		1	1	1 2022-07-26 10:5	2022-07-26 10:50
2	ad2.jpg	pink		1		2	2	1 2022-07-26 10:5	2022-07-26 10:52
3	ad3.jpg	black	103	(Null)		1	3	1 2022-07-26 10:5	2022-07-26 10:54
4	ad4.jpg	orange		2		2	4	1 2022-07-26 10:5	2022-07-26 10:54
5	ad5.jpg	green	101	(Null)		1	1	0 2022-07-26 10:5	2022-07-26 10:58

## 3.3 完成后台接口开发

### 3.3.1 实体类

```
package org.qf.entity;

import io.swagger.annotations.ApiModel;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Date;

@Data
@AllArgsConstructor
@NoArgsConstructor
@ApiModel(value = "IndexImg对象",description = "首页轮播图")
public class IndexImg {
    private String imgId;
    private String imgUrl;
    private String imgBgColor;
    private String prodId;
    private String categoryId;
    private Integer indexType;
    private Integer seq;
    private Integer status;
    private Date create_time;
    private Date update_time;
}
```

### 3.3.2 dao接口

```
package org.qf.dao;

import org.qf.entity.IndexImg;
import org.springframework.stereotype.Component;

import java.util.List;

/**
```

```

    * 首页轮播图接口
    */
@Component
public interface IndexImgMapper {
    /**
     * 查询轮播图信息  查询status=1（显示） 且按照seq进行排序
     * @return
     */
    public List<IndexImg> listIndexImgs();
}

```

### 3.3.3 mapper映射文件

```

<?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="org.qf.dao.IndexImgMapper">

    <resultMap id="BaseMap" type="indexImg">
        <id property="imgId" column="img_id"></id>
        <result property="imgUrl" column="img_url"></result>
        <result property="imgBgColor" column="img_bg_color">
    </result>
        <result property="prodId" column="prod_id"></result>
        <result property="categoryId" column="category_id">
    </result>
        <result property="indexType" column="index_type">
    </result>
        <result property="seq" column="seq"></result>
        <result property="status" column="status"></result>
        <result property="createTime" column="create_time">
    </result>
        <result property="updateTime" column="update_time">
    </result>
    </resultMap>

    <select id="listIndexImgs" resultMap="BaseMap">

```

```
        select img_id, img_url, img_bg_color, prod_id,
        category_id, index_type, seq, status, create_time, update_time
        from index_img where status=1 order by seq
    </select>

</mapper>
```

### 3.3.4 业务层接口

```
package org.qf.service;

import org.qf.utils.ResultVO;

/**
 * 轮播图业务层
 */
public interface IndexImgService {

    /**
     * 查询轮播图信息
     * @return
     */
    public ResultVO listIndexImgs();

}
```

### 3.3.5 业务层实现类

```
package org.qf.service.impl;

import org.qf.dao.IndexImgMapper;
import org.qf.entity.IndexImg;
import org.qf.service.IndexImgService;
import org.qf.utils.ResStatus;
import org.qf.utils.ResultVO;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
```

```

@Service
public class IndexImgServiceImpl implements IndexImgService {

    @Autowired
    private IndexImgMapper indexImgMapper;

    @Override
    public ResultVO listIndexImgs() {

        List<IndexImg> list = indexImgMapper.listIndexImgs();
        if(list.size()==0){
            return new ResultVO(ResStatus.NO,"fail",null);
        }else{
            return new ResultVO(ResStatus.OK,"success",list);
        }
    }
}

```

### 3.3.6 控制器

```

package org.qf.web;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiImplicitParam;
import io.swagger.annotations.ApiImplicitParams;
import io.swagger.annotations.ApiOperation;
import org.qf.service.IndexImgService;
import org.qf.utils.ResultVO;
import org.springframework.web.bind.annotation.*;

import javax.annotation.Resource;

@RestController
@RequestMapping("/index")
@CrossOrigin    //允许跨域
@Api(value = "提供首页数据显示所需的接口",tags = "首页管理")
public class IndexImgController {

    @Resource

```

```

private IndexImgService indexImgService;

@ApiOperation("首页轮播图接口")
@GetMapping("/indexImg")
public ResultVO listIndexImgs(){
    return indexImgService.listIndexImgs();
}

}

```

### 3.4 完成前端功能

当进入index.html，在进行页面初始化之后，就需要请求轮播图数据进行轮播图的展示

```

<div class="banner">
  <!--轮播 -->
  <div class="am-slider am-slider-default scoll" data-am-flexslider id="demo-slider-0">
    <ul class="am-slides">
      <li v-for="img,index in indexImg" :class=" 'banner'+(index+1)">
        <a href="introduction.html">
          
        </a>
      </li>
    </ul>
  </div>
  <div class="clear"></div>
</div>

```

```

el: "#container",
data:{
  username:"",
  userImg:"",
  isLogin:false,
  indexImg:[]
},
created () {
  //获取token
  var jsonStr=localStorage.getItem("vo");
  var vo=eval("(" +jsonStr+")");
  var token=vo.msg;
  if(token !==null && token !==""){
    this.isLogin=true
    this.username=vo.data.username;
    this.userImg=vo.data.userImg;
  }

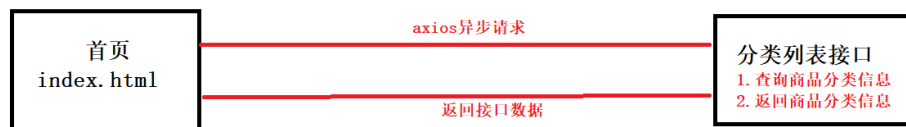
  //轮播图
  var indexUrl = baseUrl + "index/indexImg";
  axios.get(indexUrl).then((result) => {
    this.indexImg=result.data.data;
    //渲染轮播图
    //初始化轮播图动画效果
    setTimeout(function(){
      $(".am-slider").flexslider();
    },10)
  })
}

```

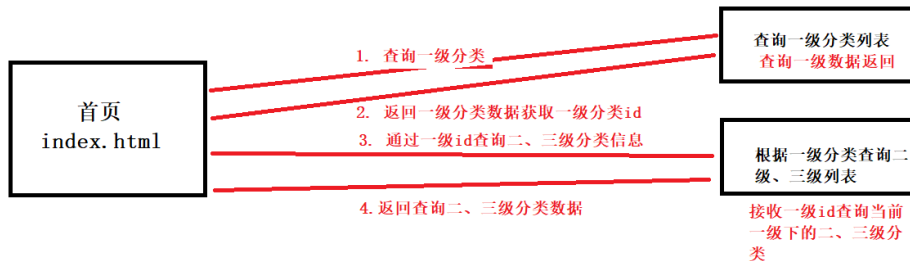
## 4. 首页分类列表

## 4.1 实现流程分析

### 方案1：一次性查询三级分类



方案2：先查询一级，再动态查询二级、三级分类



- 方案一：一次性查询三级分类
  - 优点：只需要查询一次，根据一级分类显示二级分类时响应速度快
  - 缺点：数据库查询效率较低，页面首次加载的速度也相对比较慢
- 方案二：先只查询一级分类，用户点击/鼠标移动到一级分类，动态加载二级分类
  - 优点：数据库查询效率提高，页面首次加载速度也提高
  - 缺点：需要多次连接数据库

## 4.2 数据库设计

字段	索引	外键	检查	触发器	选项	注释	SQL 预览					
名						类型	长度	小数点	不是 null	虚拟	键	注释
category_id						int			<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1	主键 分类id
category_name						varchar	255		<input checked="" type="checkbox"/>	<input type="checkbox"/>		分类名称
category_level						int			<input type="checkbox"/>	<input type="checkbox"/>		分类层次：一级分类 二级分类 三级分类
parent_id						int			<input type="checkbox"/>	<input type="checkbox"/>		父层次，上一级
category_icon						varchar	255		<input type="checkbox"/>	<input type="checkbox"/>		图标
category-slogan						varchar	255		<input type="checkbox"/>	<input type="checkbox"/>		口号
category_pic						varchar	255		<input type="checkbox"/>	<input type="checkbox"/>		分类图
category_bg_color						varchar	255		<input type="checkbox"/>	<input type="checkbox"/>		背景颜色

category_id	category_name	category_level	parent_id	category_icon	category-slogan	category_pic	category_bg_color
7	巧克力	1	0	chocolate.png	(Null)	act1.png	black
8	海味/河鲜	1	0	fish.png	(Null)	act1.png	black
9	花茶/果茶	1	0	tea.png	(Null)	act1.png	black
10	品牌/礼包	1	0	package.png	(Null)	act1.png	black
11	蛋糕	2	1	(Null)	(Null)	(Null)	(Null)
12	点心	2	1	(Null)	(Null)	(Null)	(Null)
13	饼干	2	2	(Null)	(Null)	(Null)	(Null)
14	薯片	2	2	(Null)	(Null)	(Null)	(Null)
15	虾条	2	2	(Null)	(Null)	(Null)	(Null)
16	猪肉脯	2	3	(Null)	(Null)	(Null)	(Null)
17	牛肉丝	2	3	(Null)	(Null)	(Null)	(Null)
18	小香肠	2	3	(Null)	(Null)	(Null)	(Null)

## 4.3 完成后台接口开发



### 4.3.1 实体类

```
package org.qf.entity;

import io.swagger.annotations.ApiModel;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * 首页分类
 */
@Data
@AllArgsConstructor
@NoArgsConstructor
@ApiModel(value = "category对象",description = "首页分类")
public class Category {

    private Integer categoryId;
    private String categoryName;
    private Integer categoryLevel;
    private Integer parentId;
    private String categoryIcon;
    private String categorySlogan;
    private String categoryPic;
    private String categoryBgColor;
}
```

### 4.3.2 dao接口

```
/**
 * 方案1：一次性查询所有的分类
 * @return
 */
public List<Category> selectAllCategory();
```

### 4.3.3 mapper映射文件

```
<?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="org.qf.dao.CategoryMapper">

    <resultMap id="categoryMap" type="category">
        <id property="categoryId" column="category_id"></id>
        <result property="categoryName"
column="category_name"></result>
        <result property="categoryLevel"
column="category_level"></result>
        <result property="parentId" column="parent_id">
</result>
        <result property="categoryIcon"
column="category_icon"></result>
        <result property="categorySlogan"
column="category_slogan"></result>
        <result property="categoryPic" column="category_pic">
</result>
        <result property="categoryBgColor"
column="category_bg_color"></result>
    </resultMap>

    <select id="selectAllCategory" resultMap="categoryMap">
        select * from category c1 INNER JOIN category c2 on
c1.category_id=c2.parent_id
        left JOIN category c3 on c3.parent_id=c2.category_id
    </select>

</mapper>

```

#### 4.3.4 业务层接口

```

package org.qf.service;

import org.qf.utils.ResultVO;

public interface CategoryService {
    /**
     * 查询所有的分类
     * @return

```

```

        */
        public ResultVO listCategories();
    }

```

#### 4.3.5 业务层实现类

```

package org.qf.service.impl;

import org.qf.dao.CategoryMapper;
import org.qf.entity.Category;
import org.qf.service.CategoryService;
import org.qf.utils.ResStatus;
import org.qf.utils.ResultVO;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class CategoryServiceImpl implements CategoryService {

    @Autowired
    private CategoryMapper categoryMapper;

    //查询所有的三级分类
    @Override
    public ResultVO listCategories() {
        List<Category> list =
categoryMapper.selectAllCategory();
        if(list.size()==0){
            return new ResultVO(ResStatus.NO,"fail",null);
        }else{
            return new ResultVO(ResStatus.OK,"success",list);
        }
    }
}

```

## 4.3.6 控制器

```
@RestController
@RequestMapping("/index")
@CrossOrigin //允许跨域
@Api(value = "提供首页数据显示所需的接口",tags = "首页管理")
public class IndexImgController {

    @Resource
    private IndexImgService indexImgService;

    @Resource
    private CategoryService categoryService;

    @ApiOperation("首页轮播图接口")
    @GetMapping("/indexImg")
    public ResultVO listIndexImgs() { return indexImgService.listIndexImgs(); }

    @ApiOperation("首页分类查询接口")
    @GetMapping("/listCategory")
    public ResultVO listCategory(){
        return categoryService.listCategories();
    }
}
```