**搭建商品系统**

# 导入数据库

## 1.1导入数据库

文本, 信件

描述已自动生成

## 1.2商品表t\_product的表结构

表格

描述已自动生成

product\_id:商品Id

String类型(char 36类型).代码中使用UUID来完成主键id的值设置

product\_name:商品名称 String

product\_price:商品价钱 Double

product\_category:商品类型String

product\_imgurl: 商品图片链接地址 String

例如:http://image.easymall.com/upload/3/d/5/f/g/3/e/2/uuid.jpg|png|gif

图片(作为数据)能存储在数据库,类型blob,大字段类型的数据不建议存储在数据库,会破坏数据库的索引结构,导致查询数据绕过索引,遍历全表。

在easymall中图片作为静态资源存储在nginx服务器中,利用静态资源访问,交给nginx处理图片内容,但是图片资源量非常庞大,单个nginx无法维护所有的商品图片(分布式存储处理图片,最终显示交给nginx访问分布式存储图片的系统显示)

product\_num: 商品库存 Integer

product\_description: 商品详情描述 String

t\_product表中的图片数据如下

信件

中度可信度描述已自动生成

# 商品系统 项目搭建

## 2.1搭建步骤

1.搭建一个quickstart工程

**easymall-microservice-product**

2.添加src/main/resources文件夹

## 2.2 pom文件

* 继承easymall-parent spring-boot-starter-parent,spring-boot-starter-web,spring-cloud-starter-eureka,spring-cloud-starter-ribbon

<!-- 继承自己的父工程 -->

<parent>

<groupId>cn.edu.scnu</groupId>

<artifactId>easymall-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

* 持久层

依赖[easymall-common-repository] jdbc,mysql,mybatis,druid连接池，

<dependency>

<groupId>cn.edu.scnu</groupId>

<artifactId>easymall-common-repository</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

* 使用工具公用资源代码

依赖easymall-common-resources

<dependency>

<groupId>cn.edu.scnu</groupId>

<artifactId>easymall-common-resources</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

## 2.3 配置properties文件

* 配置数据库连接
* spring.datasource.type=德鲁伊数据源

为单独自定义的数据源对象创建4个属性

spring.datasrouce.initialSize=5

spring.datasrouce.maxActive=50

spring.datasrouce.maxIdle=10

spring.datasrouce.minIdle=5

* mybatis相关配置

别名包:com.easymall.common.pojo

mapper映射扫描: mapper/\*.xml

驼峰命名开启/关闭持久层框架缓存

* 微服务框架配置

product相关服务名称productservice

注册指向的eureka注册中心8761

server.port=10001

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql:///easydb?useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true

spring.datasource.username=root

spring.datasource.password=12345678

spring.datasource.type=com.alibaba.druid.pool.DruidDataSource

spring.datasource.initialSize=5

spring.datasource.maxActive=50

spring.datasource.maxIdle=10

spring.datasource.minIdle=5

mybatis.typeAliasesPackage=com.easymall.common.pojo

mybatis.mapperLocations=classpath:mapper/\*.xml

mybatis.configuration.mapUnderscoreToCamelCase=true

mybatis.configuration.cacheEnabled=false

spring.application.name=productservice

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/

## 2.3 创建启动类

标准的springboot启动类

@MapperScan扫描接口

@EnableEurekaClient

@Bean

@LoadBalanced

创建一个可能用到的RestTemplate对象

图形用户界面, 文本, 应用程序

描述已自动生成

# 商品系统 功能创建

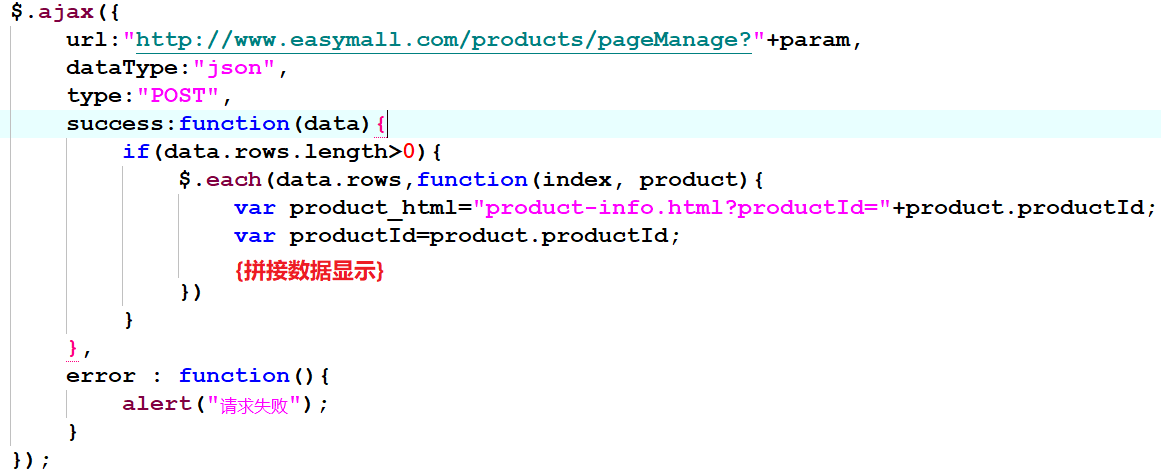
## 3.1商品的分页查询

### 3.1.1 功能需求

#### 1.前台“全部商品”分页显示



#### 2. js源码展示(product-list.html)

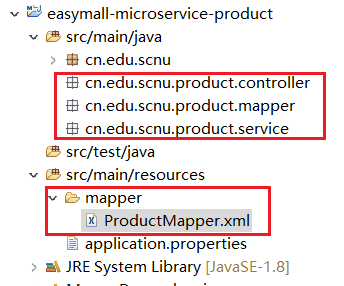


#### 3.js代码解释

页面通过js发起ajax请求,按照ajax的要求返回对象EasyUIResult的json进行解析.

### 3.1.2 实现步骤：

#### 1.搭建代码框架结构

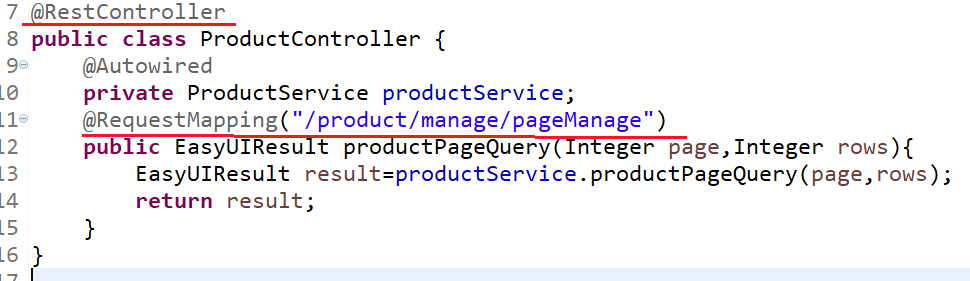


#### 2.控制器层ProductController

接口文件：

|  |  |
| --- | --- |
| js请求地址 | http://www.easymall.com/products/pageManage?**page=1&rows=1** |
| 后台接收 | /product/manage/pageManage?page=1&rows=1 |
| 请求方式 | Get |
| 请求参数 | Get提交参数 Integer page,Integer rows |
| 返回数据 | 根据查询结果封装2个数据到EasyUIResult对象中:  Integer total:查询的总条数;  List<Product> rows:查询分页的数据结果; |
| 备注 | 商品总数 SELECT COUNT(\*) FROM t\_product  分页数据  (第一页的5条数据) SELECT \* FROM t\_product LIMIT 0,5  (第二页的5条数据) select \* from t\_product LIMIT 5,5  (第三页的5条数据) select \* from t\_product LIMIT 10,5 |

代码编写：



#### 3.Service层ProductService

// 封装数据到EasyUIResult对象

// 1.创建一个返回的对象,将查询数据set进去然后返回

// 2封装第一个属性 total 的数量

// 3封装第二个属性List<Product> pList

// 根据页数计算起始位置

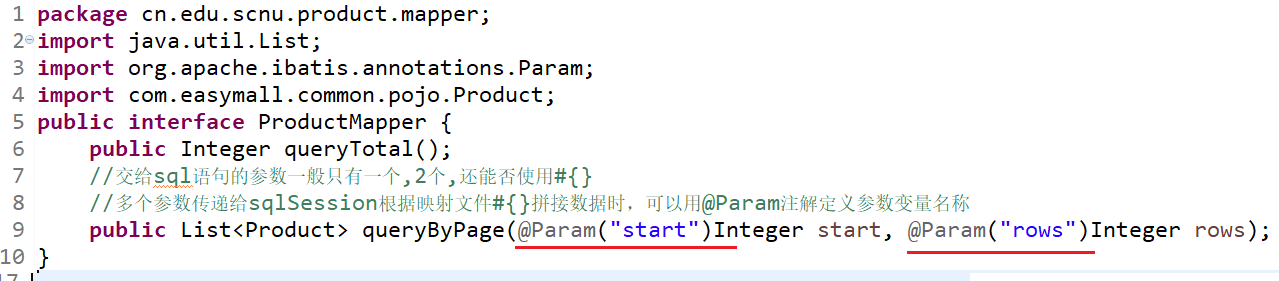
// 4封装对象属性

文本

描述已自动生成

#### 4.Dao层ProductMapper

多个参数时,可以使用@Param注解,定义参数在sql预编译时使用#{value值}



#### 5.映射文件ProductMapper.xml

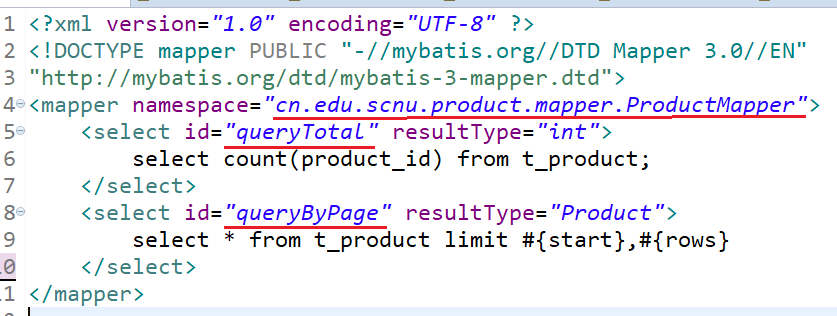
<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*""*>

</mapper>



### 3.1.3 测试

#### 1.单独测试：

http://localhost:10001/product/manage/pageManage?page=1&rows=2

返回结果为EasyUIResult对象，类似下面的结构：



返回结果：

{

"total": 29,

"rows": [{

"productId": "05e20c1a-0401-4c0a-82ab-6fb0f37db397",

"productName": "荣耀Play5T",

"productPrice": 1199.0,

"productCategory": "手机数码",

"productImgurl": "http://image.easymall.com/upload/5/e/d/5/4/5/e/b/5f0d34dc-157f-49ba-ad39-1b28927ba6ae\_1005714.jpg",

"productNum": 206,

"productDescription": "22.5W超级快充 5000mAh大电池 6.5英寸护眼屏"

}, {

"productId": "09f47493-214d-44bc-927d-6ce0bf89a057",

"productName": "爱疯9S(0315-01)",

"productPrice": 1000.0,

"productCategory": "手机数码",

"productImgurl": "http://image.easymall.com/upload/5/2/3/4/7/8/d/c/1838eaa6-6459-420f-b8e2-6ea9f43c4b5e\_dfd259ab-bcc7-43f6-a9d5-62872ff5671e.jpg",

"productNum": 185,

"productDescription": "爱疯9S(0315-01)"

}]

}

#### 2.页面整合测试

* 1. 复制一个网关工程

**easymall-micro-zuul**

**配置properties**

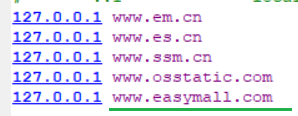
server.port=9005

spring.application.name=serviceZuul

zuul.routes.**zuul-product**.path=/**zuul-product**/\*\*

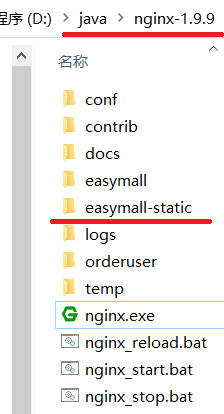
zuul.routes.**zuul-product**.serviceId=productservice

* 1. 在host文件中配置www.easymall.com

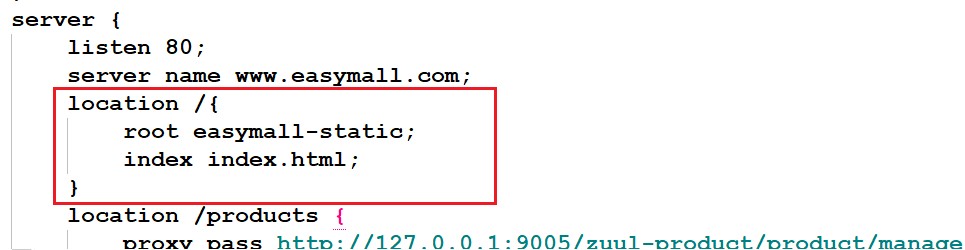


* 1. 在Nginx中配置静态界面

将easymall-static复制到**d:\java\nginx-1.9.9**中



在nginx.conf中做location配置



* 1. 根据js请求地址,在nginx中做商品系统的location匹配

<http://www.easymall.com/products/pageManage>

nginx.conf的配置如下：

server {

listen 80;

server\_name www.easymall.com;

location /{

root easymall-static;

index index.html;

}

location /products {

proxy\_pass http://127.0.0.1:9005/zuul-product/product/manage;

add\_header 'Access-Control-Allow-Credentials' 'true';

add\_header 'Access-Control-Allow-Origin' '\*';

}

}

* 1. 启动
* 启动Eureka-Server和Eureka-Server2。查看结果
* 启动网关**easymall-micro-zuul**
* 启动easymall-microservice-product

查看结果

* 1. 运行过程分析

起始访问地址，通过js:

<http://www.easymall.com/products/pageManage?page=1&rows=5>

<http://www.easymall.com/products/pageManage>;

**http://127.0.0.1:9005/zuul-product/product/manage/pageManage?page=1&rows=5**

|nginx监听 location匹配

将www.easymall.com/products去掉

剩下/pageManage;拼接proxy\_pass的路径

**|**nginx拼接转发后发送的地址

<http://localhost:9005/zuul-product/product/manage/pageManage>

**|**zuul网关接收到地址

**|**满足路由匹配/zuul-product/\*\*

**|**路径过滤http://localhost:9005/zuul-prod/

**|**拼接底层restTemplate发起向服务的请求

<http://productservice/product/manage/pageManage>

**|**经过ribbon抓取的服务list进行正确服务节点的拼接

<http://localhost:10001/product/manage/pageManage>

## 3.2 访问图片资源

### 3.2.1 配置host文件

**http://image.easymall.com**/upload/5/2/3/4/7/8/d/c/1838eaa6-6459-420f-b8e2-6ea9f43c4b5e\_dfd259ab-bcc7-43f6-a9d5-62872ff5671e.jpg



### 3.2.2 配置nginx.conf

#### 1.将图片文件夹upload复制到d:/java/easymall\_image下。

图形用户界面, 应用程序

描述已自动生成

#### 2.通过nginx维护所有图片,请求地址image.easymall.com

配置nginx.conf中一个server的内容

server {

listen 80;

server\_name image.easymall.com;

location / {

root D:\\java\\easymall\_image;

}

}

<http://image.easymall.com/upload/2/6/4/a/a/5/2/3/ee6c796a-6333-4cd5-a06e-271d876aac8c_589577.jpg>

#### 3.运行结果

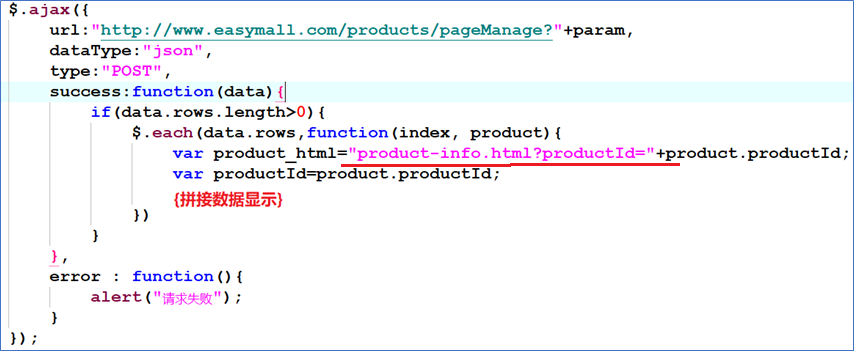
图形用户界面, 网站

描述已自动生成

## 3.3 单个商品的查询

### 3.3.1 功能描述

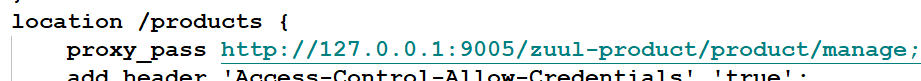
#### 首页—全部商品—点击图片或者名称，跳转到详细信息展示页面展示返回数据



#### js源码展示(product-info.html)

图形用户界面, 文本

描述已自动生成



**http://www.easymall.com/products/item/{productId}**

**http://127.0.0.1:9005/zuul-product/product/manage/item/{productId}**

**http://productservice/product/manage/item/{productId}**

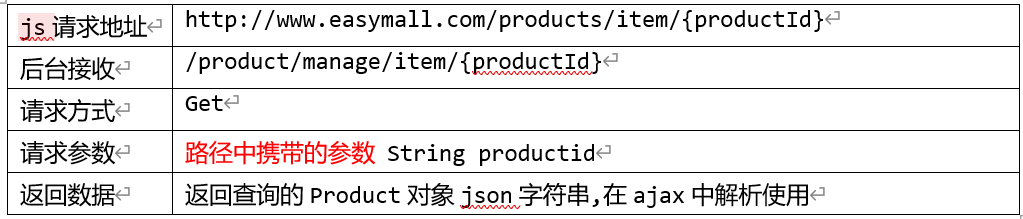
**http://127.0.0.1:10001/product/manage/item/{productId}**

**http://127.0.0.1:10001/product/manage/item/{05e20c1a-0401-4c0a-82ab-6fb0f37db397}**

#### js代码解释

页面加载后直接调用ajax请求,将返回的对象数据,解析,调用js代码拼接到对应的元素标签中

#### 接口文件



### 3.3.2 @PathVariable注解

springmvc的功能之一 @PathVariable

例如下面路径:/page/zhangsan/58

路径中直接传值，没有参数名

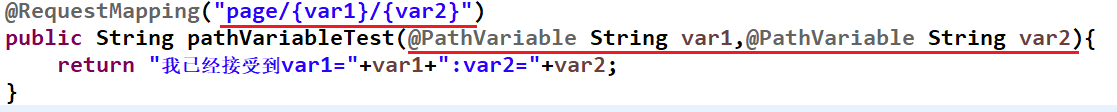
**如何获取路径中的 zhangsan值 和58值？**

在controller接收时RequestMapping注解格式：

@RequestMapping("page/{变量名1}/{变量名2}")

在控制器层处理方法中，按照**前后位置顺序**

@PathVariable String 变量名1,@PathVariable String 变量名2



### 3.3.3代码实现

#### 控制器层 （ProductController）

文本

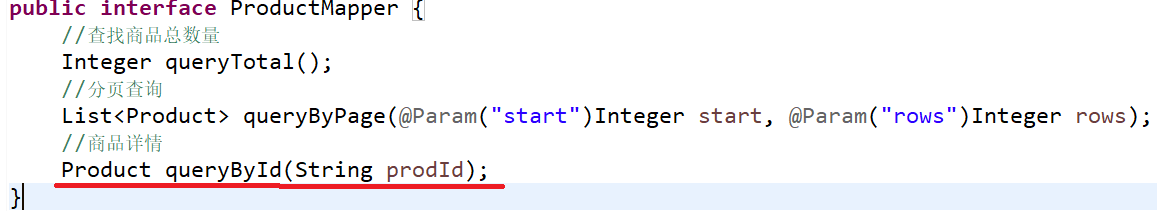
中度可信度描述已自动生成

#### Service层（ProductService）

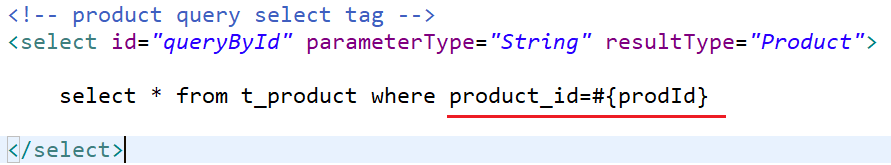
文本

描述已自动生成

#### Dao层（ProductMapper）



#### 映射文件（ProductMapper.xml）



### 3.3.4 测试

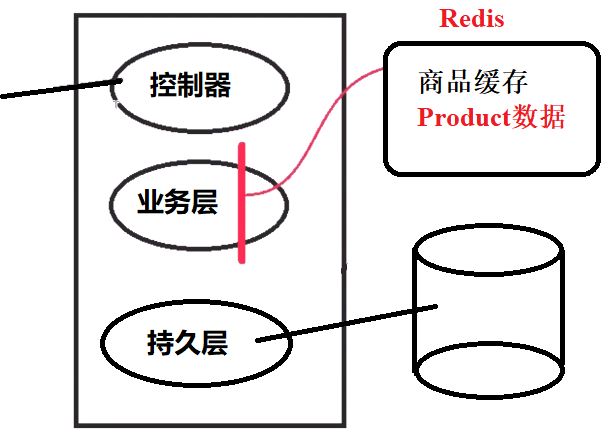
页面整合商品单个查询的效果

* + nginx的配置保持和上个功能一致
  + zuul的网关路由也不用发生变动
  + 流转过程(理解清楚)

从js发起请求(接口)

到最后功能提供者的requestMapping接收

### 3.3.5 商品缓存

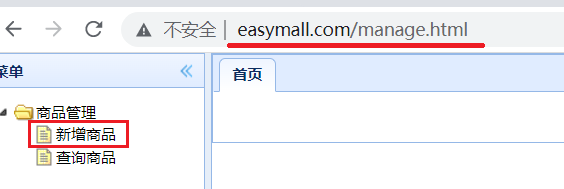


# 商品新增（没有上传图片）

## 4.1 功能需求

### 4.1.1 功能演示

http://www.easymall.com/manage.html



文本

描述已自动生成

在后台页面点击 新增商品 ,进入界面：

图形用户界面

描述已自动生成

填写数据点击提交

图形用户界面, 文本, 应用程序

描述已自动生成

继续点击“确定”会提示新增商品成功（**这一步我们还没做**）

图形用户界面, 文本, 应用程序

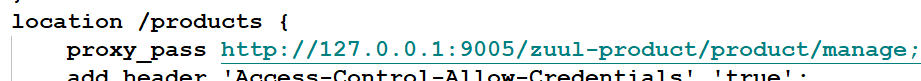
描述已自动生成

### 4.1.2 js源码展示

查看item-add.html的js代码

图片包含 文本

描述已自动生成



### js 代码解释

**$("#itemAddForm").serialize()**

新增商品的表单数据作为参数,**jquery将其序列化成了key=value&key=value的字符串**,在请求体中提交,接收返回的数据,判断执行状态结果弹框提示.

js的请求地址：

[http://www.easymall.com**/products**/save](http://www.easymall.com/products/save)

[**http://127.0.0.1:9005/zuul-product/product/manage/****save**](http://127.0.0.1:9005/zuul-product/product/manage/save)

图片包含 文本

描述已自动生成

**http://productservice/product/manage/save**

[**http://127.0.0.1:10001/product/manage/save**](http://127.0.0.1:10001/product/manage/save)

## 4.2 接口文件

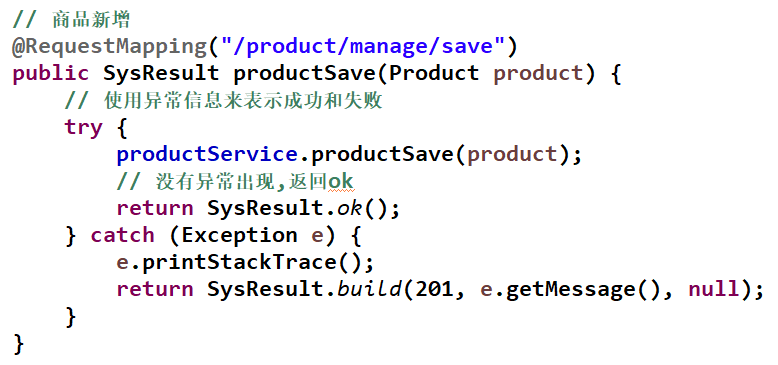
引入springcloud资源dependencyManagement import导入

|  |  |
| --- | --- |
| js请求地址 | <http://www.easymall.com/products/save> |
| 后台接收 | /product/manage/save |
| 请求方式 | Post |
| 请求参数 | Product product对象接参 缺少id |
| 返回数据 | 返回SysResult对象的json,其结构:  Integer status; 200表示成功,其他表示失败  String msg;成功返回 “ok”,失败返回其他信息  Object data;根据需求携带其他数据 |
| 备注 | 返回的SysResult对象是一个标准的和ajax对话的后台vo对象,它可以将当前的操作结果通过status传递给ajax,可以将操作中出现任何后台的消息,例如错误信息,以msg传递给前台,也可以将很多其他数据封装到data传递给前台使用 |

## 4.3功能编写

### 4.3.1 控制器层

Product product对象接参 缺少id



### 4.3.2 Service层

文本

描述已自动生成

### 4.3.3 Dao层

图片包含 徽标

描述已自动生成

### 4.3.4 映射文件

文本

描述已自动生成

## 4.4 和页面整合

1. 启动nginx
2. 启动两个Eureka注册中心
3. 启动zuul
4. 启动商品微服务

## 4.5 pojo分类（JavaBean规范）

* entity:严格按照一个字段对应一个属性，对应表格与类关系的javaBean

文本

描述已自动生成

* domain:具有一定业务意义的后台封装对象,一般代表具有关系的内容;例如:Order中封装一个orderItemList，一个订单可以包括多个商品。但是order表中设有orderItem字段。

文本

描述已自动生成

* dto:从页面传递给服务器的数据的格式对象，比如，京东商品输入，输入的信息特别细，比如说输入“TCL大彩电”，还会被要求输入尺寸，大小，长度高度，颜色，型号，重量等等信息，这些信息可以被封装在一个dto对象，但是这些数据将会被拆分开来，分别添加到不同的表格里面。
* vo:后台传递给js解析的对象,vo是view object的简写

图示

描述已自动生成 图示

描述已自动生成

# 商品修改

## 5.1功能需求

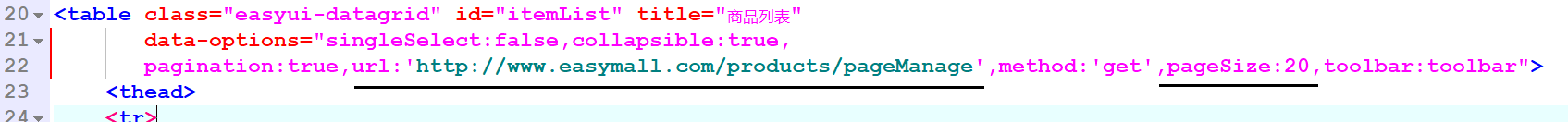
### 5.1.1 功能演示

后台的查询页面，跟前面的**分页显示**功能使用相同代码实现。查看请求地址：

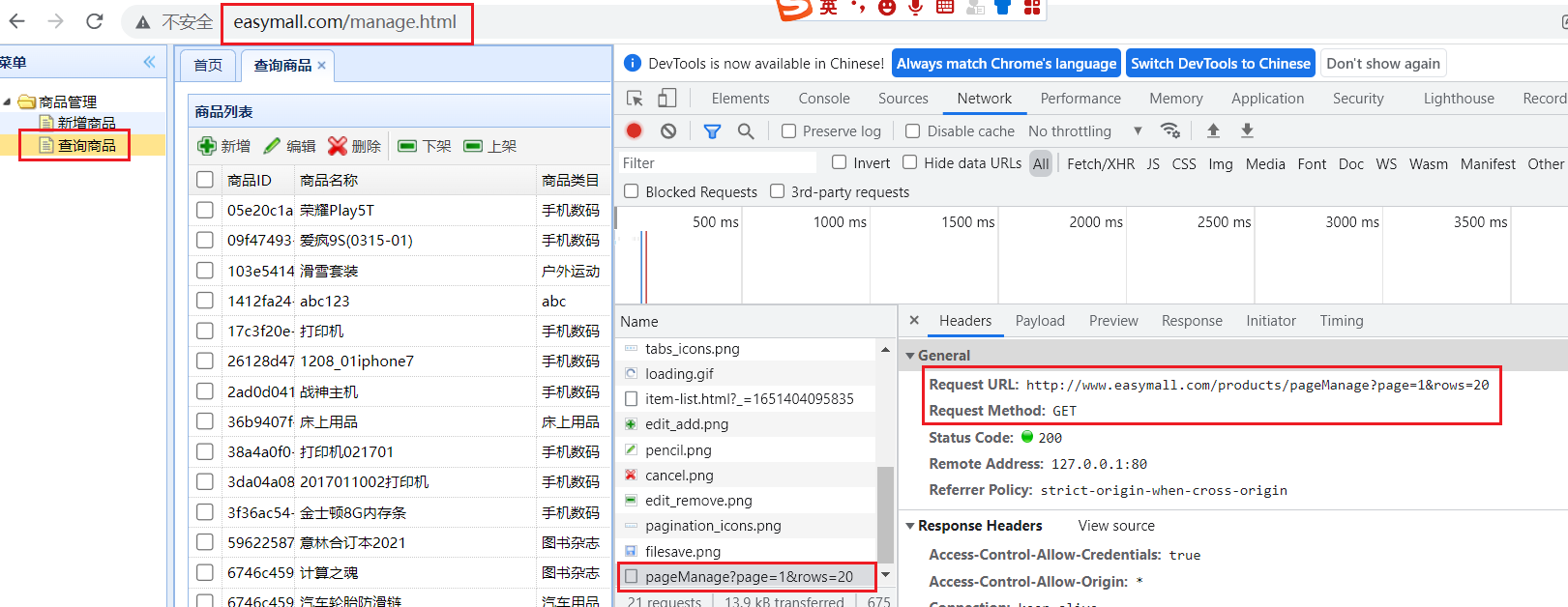
文本

描述已自动生成

打开item-list.html，看到js的请求的url地址：

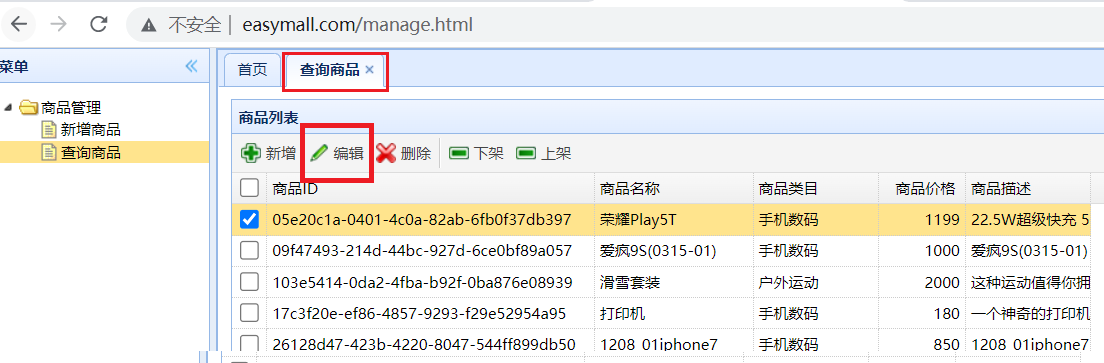


在页面点击“查询商品”，按F12，可以看到：



可以看到js发出的请求地址跟前面“商品分页查询”功能的请求地址一样，这里是一页显示20条记录。所以我们点击查询商品，可以直接查看到数据而不用重新编写代码。

* 在页面选择一个商品点击编辑

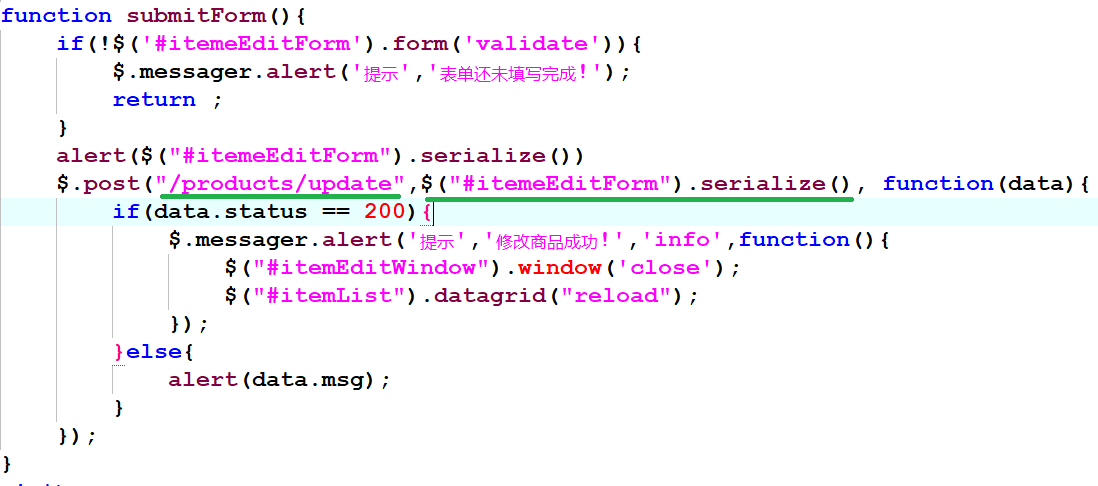


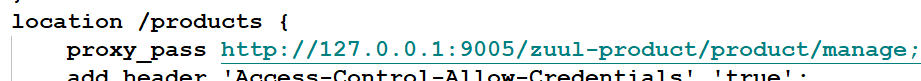
* 在弹出页面修改所有想要改的内容



### 5.1.2 js源码展示

查看item-edit.html中的js 代码





### 5.1.3 js代码解释

修改商品的表单数据作为参数,**jquery将其序列化成了key=value&key=value的字符串**,在请求体中提交,接收返回的数据,判断执行状态结果弹框提示

**js的请求地址：**

[http://www.easymall.com**/products**/**update**](http://www.easymall.com/products/update)

[**http://127.0.0.1:9005/zuul-product/product/manage/update**](http://127.0.0.1:9005/zuul-product/product/manage/update%20)

图片包含 文本

描述已自动生成

**http://productservice/product/manage/update**

[**http://127.0.0.1:10001/product/manage/update**](http://127.0.0.1:10001/product/manage/save)

## 5.2接口文件

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

## 5.3功能编写

### 5.3.1 控制器层

Product product对象接参

文本

描述已自动生成

### 5.3.2 Service层

图形用户界面, 文本

描述已自动生成

### 5.3.3 Dao层

徽标

描述已自动生成

### 5.3.4 映射文件

文本, 信件

描述已自动生成

## 5.4 与页面整合测试

1. 启动nginx
2. 启动两个Eureka注册中心
3. 启动zuul
4. 启动商品微服务