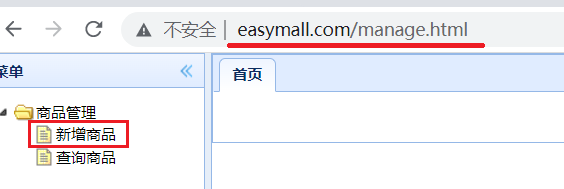
# 商品新增（没有上传图片）

## 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

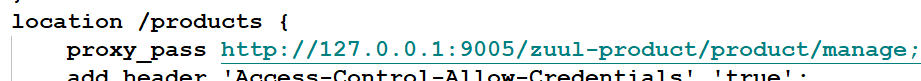
描述已自动生成

### 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

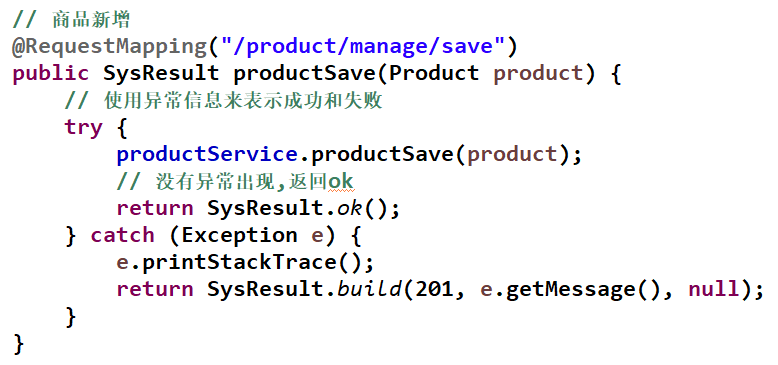
引入springcloud资源dependencyManagement import导入

|  |  |
| --- | --- |
| 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

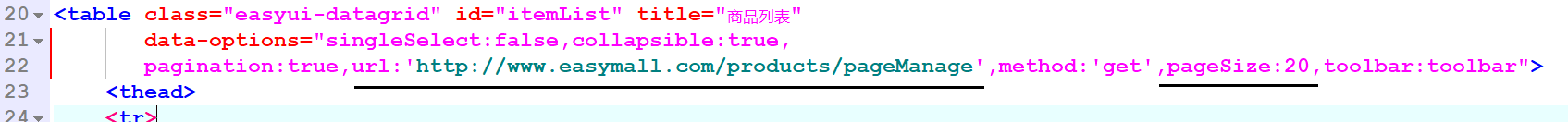
### 5.1.1 功能演示

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：

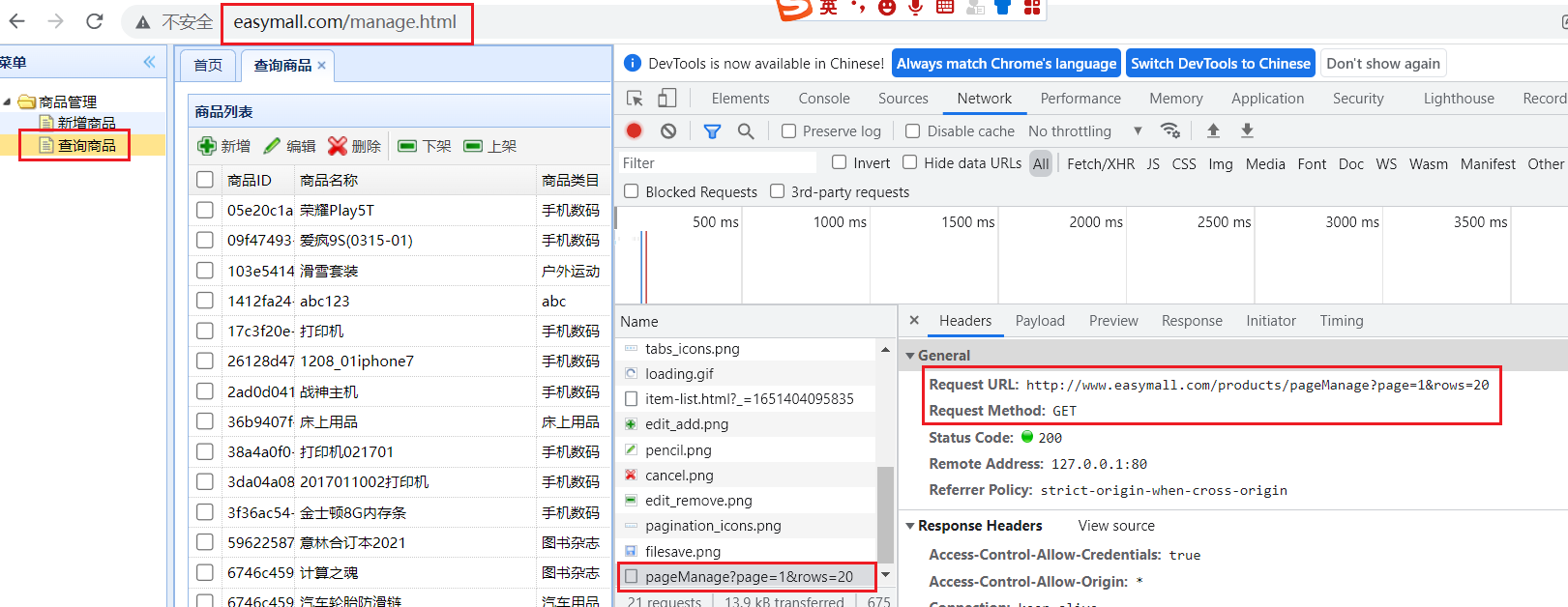
文本

描述已自动生成

打开item-list.html，看到js的请求的url地址：

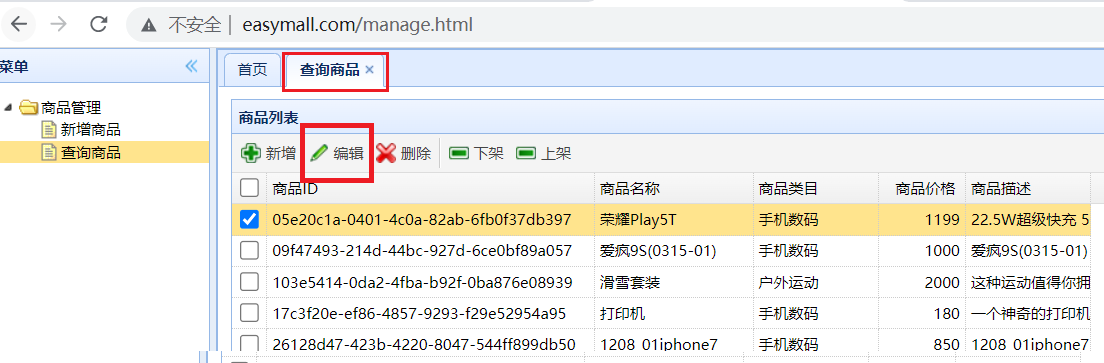


在页面点击“查询商品”，按F12，可以看到：



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

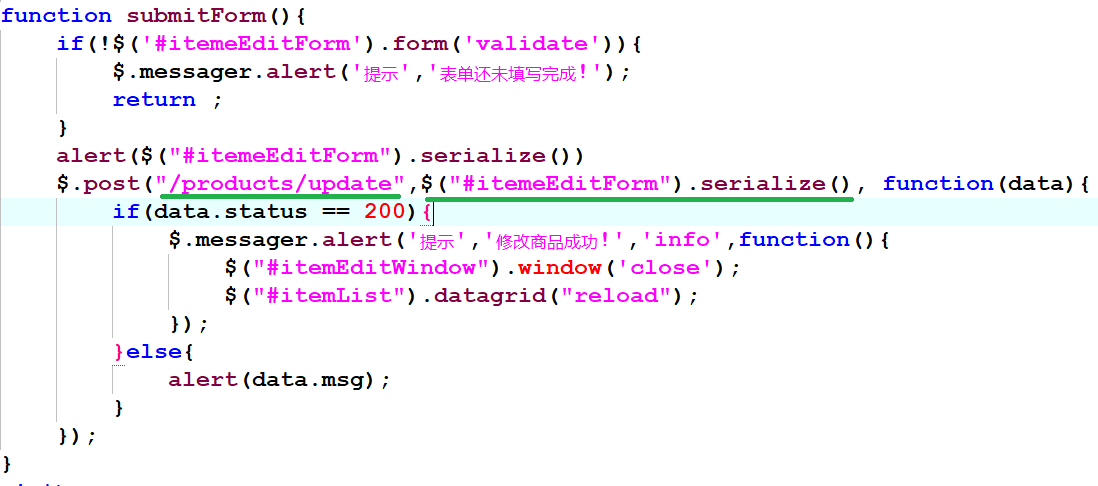


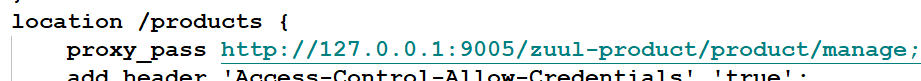
* 在弹出页面修改所有想要改的内容



### 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

# 图片上传

**单独搭建图片上传系统**

**分析:**

图示

描述已自动生成

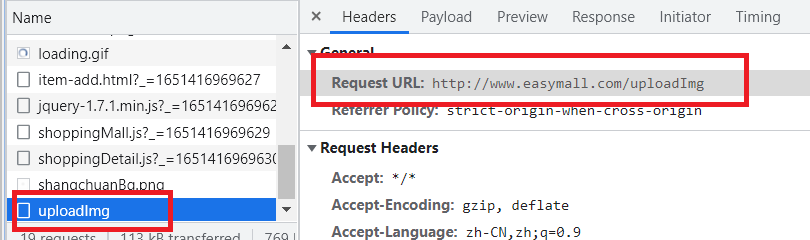
## 6.1调用流程

查看item-add.html中的图片的change事件：

图形用户界面, 文本, 应用程序

描述已自动生成

**查看商品添加页面，选择图片，通过F12可以查看到js图片上传请求地址：**



**Nginx配置：**

图形用户界面, 文本, 应用程序

描述已自动生成

**网关配置：**

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

zuul.routes.zuul-pic.serviceId=picservice

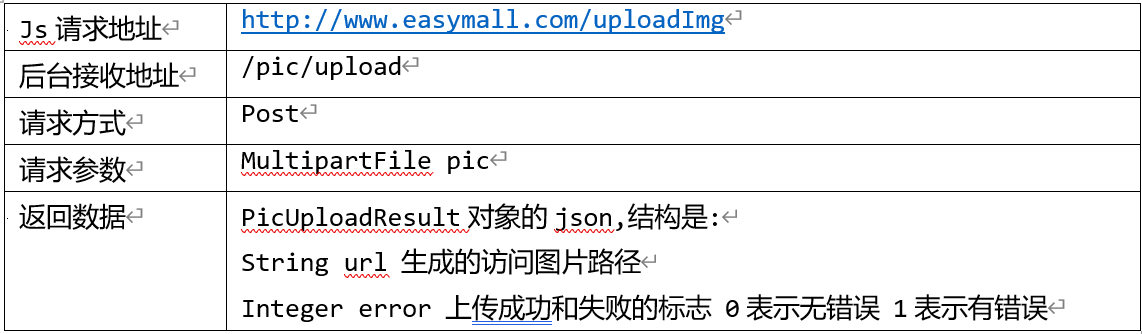
**替换：**

[**http://picservice/pic/upload**](http://picservice/pic/upload)

**ribbon抓取服务提供者地址**

[**http://127.0.0.1:10002/pic/upload**](http://127.0.0.1:10002/pic/upload)

**接口文件：**

****

## 6.2搭建图片系统

没有持久层内容（不用访问数据库）

### 6.2.1 搭建步骤

#### 1.搭建一个quikstart

**easymall-microservice-pic**

#### 2.编辑pom文件

1)继承parent

<!-- 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>

2)依赖common-resources

<dependency>

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

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

<version>0.0.1-SNAPSHOT</version>

</dependency>

#### 3.编辑application.properties

server.port=10002

server.contextPath=/

spring.application.name=picservice

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

pic.pathDirPrefix=d://java//easymall\_image/

pic.urlPrefix=http://image.easymall.com/

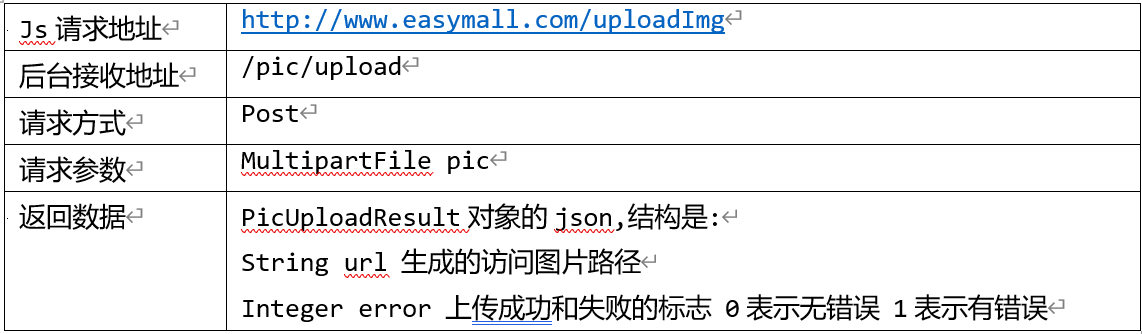
#### 创建启动类(eureka客户端)

文本, 信件

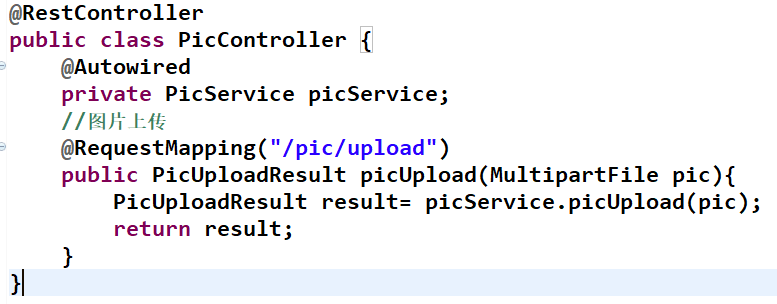
描述已自动生成

### 6.2.2根据接口文件完成图片上传功能

#### 接口文件

****

#### 控制器层



#### Service层

1)判断后缀合法

2判断是不是木马

3)创建以upload开始的路径

4)创建nginx访问的静态目录,pathDir, 通过配置文件中的变量pathDirPrefix 创建(D:\java\easymall\_image**\upload\2\2\c\a\b\0\e\b\**)

5)创建urlPrefix （http://www.easymalll.com/upload/2/2/c/a/b/0/e/b/)

6)拼接图片名称,将图片重命名 uuid表示图片存储访问的名称

7上传文件到磁盘路径

8)返回urlPrefix+图片名称的路径



### 6.2.3 配置zuul网关路由 配置Nginx.conf的转发

#### 1.修改nginx配置文件

修改localhost /uploadImg指向zuul网关路径

图形用户界面, 文本, 应用程序

描述已自动生成

#### 2.定义zuul路由

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

zuul.routes.zuul-pic.serviceId=picservice

### 6.2.4 与页面整合测试

1.启动Eureka

2.启动zuul网关

3.启动商品系统

4.启动文件上传系统

5启动Nginx