**1.购物车表格结构**

t\_cart:一行数据表示某个用户的某个购物车商品

表格

描述已自动生成

id:主键 自增整数

user\_id:复合主键之一,表示某个用户

product\_id:复合主键之一,表示商品

product\_image:商品图片链接url(反范式设计的冗余字段)

product\_name:商品名称(反范式设计的冗余字段)

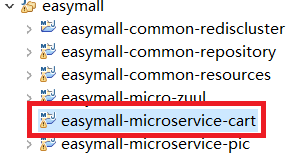
product\_price:商品的价钱(反范式设计的冗余字段)

num:购买数量

**2.搭建购物车系统**

**2.1创建工程**

* quickstart



* pom文件

继承easymall-parent，spring cloud环境，加入eureka客户端依赖ribbon

<parent>

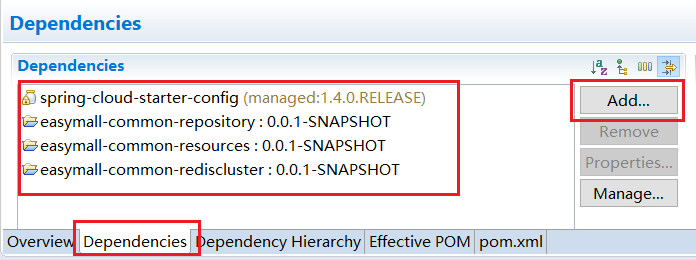
<groupId>cn.edu.scnu</groupId>

<artifactId>easymall-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

加入持久层工程和resouces工程依赖，加入redis工程依赖



手机屏幕截图

描述已自动生成

**2.2配置各种文件**

* **bootstrap.properties**

端口

注册eureka地址

访问开启服务config配置中心的configserver

application的名称服务名称

读取配置文件的前缀application datasource redis

图形用户界面, 文本, 应用程序

描述已自动生成

server.port=10004

spring.application.name=cartservice

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/,http://localhost:8762/eureka/

spring.cloud.config.label=master

spring.cloud.config.name=redis,datasource

spring.cloud.config.discovery.enabled=true

spring.cloud.config.discovery.serviceId=configserver

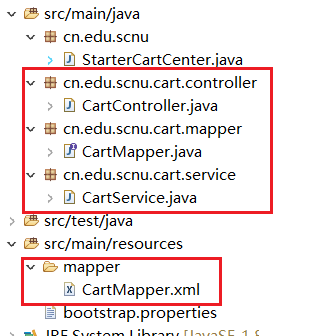
* **启动类**

图形用户界面, 文本, 应用程序

描述已自动生成

* **搭建一个项目代码架子**

**创建CartController-CartService-CartMapper-CartMapper.xml**



1. CartController

图形用户界面, 文本, 应用程序

描述已自动生成

1. CartService

图形用户界面, 文本, 应用程序

描述已自动生成

3）CartMapper接口

文本

描述已自动生成

4）CartMapper.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=*"cn.edu.scnu.cart.mapper.CartMapper"*>

</mapper>

**3.编写功能代码**

**3.1我的购物车数据查询**

### 3.1.1 功能需求演示

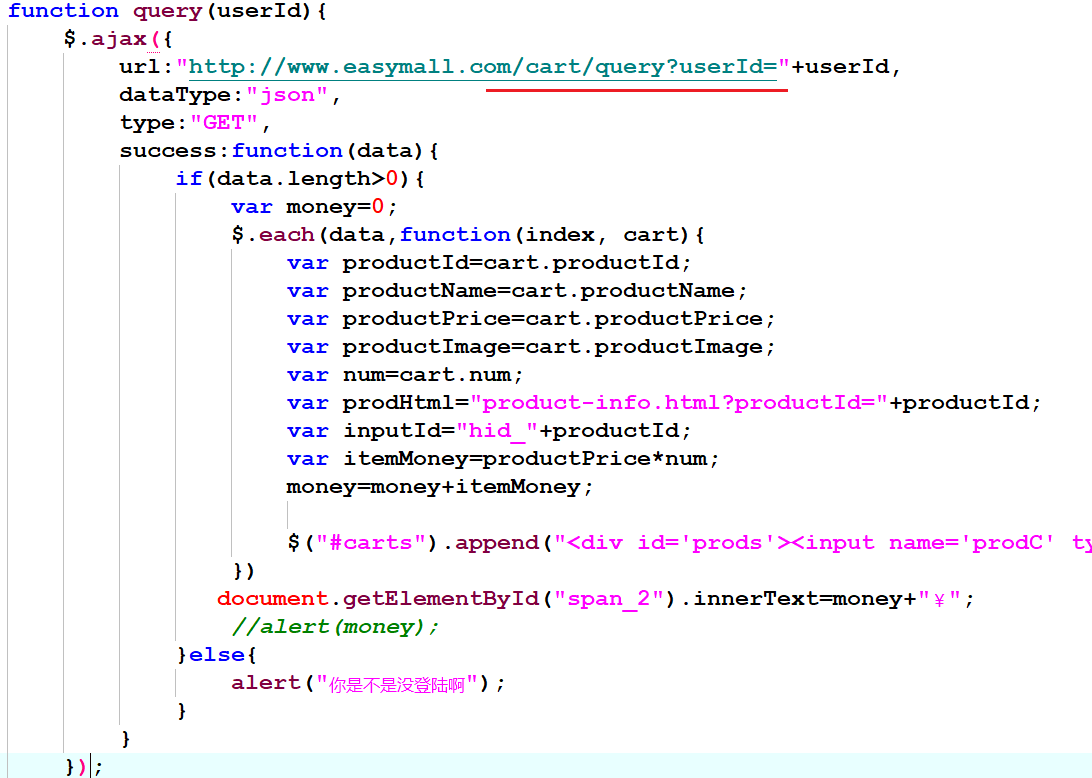
登录状态下点击我的购物车，转向购物车显示数据

图形用户界面

描述已自动生成

如果未登录转向登录页面

### 3.1.2 js源码展示(mycart.html)



### 3.1.3 js代码解释

页面加载,调用ajax方法,发起请求,根据返回结果,跳转页面,登录了直接拼接到页面位置展示购物车信息.没登录则跳转向登录页面

http://www.easymall.com/cart/query?userId=+userId,

Nginx配置：

图形用户界面, 文本, 应用程序

描述已自动生成

[http://www.easymall.com/cart/query](http://www.easymall.com/cart/query?userId=%22+userId)

<http://127.0.0.1:9005/zuul-cart/cart/manage/query>

zuul中的properties中配置cart网关相关配置

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

zuul.routes.zuul-cart.serviceId=cartservice

<http://cartservice/cart/manage/query>

**http://127.0.0.1:10004/**cart/manage/query

### 3.1.4 接口文件

表格

描述已自动生成

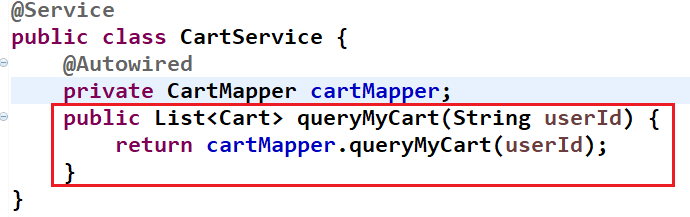
### 3.1.5 功能编写

1）CartController

手机屏幕截图

描述已自动生成

2）CartService



3）CartMapper

文本

描述已自动生成

4) CartMapper.xml

文本

描述已自动生成

**3.2整合页面zuul网关**

### 3.2.1 nginx配置

cart购物车访问zuul的location

**配置完重启nginx**

location /cart {

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

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

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

}

### 3.2.2 zuul网关的路由配置

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

zuul.routes.zuul-cart.serviceId=cartservice

### 3.2.3 测试

独立测试

http://localhost:10004/cart/manage/query?userId=f577f9f9-159e-4aaf-9332-fd7b294bc208



联合测试：

重启nginx

启动eureaka,zuul,config-server,product,user,cart观察页面查询我的购物车功能

文本

描述已自动生成

应用程序

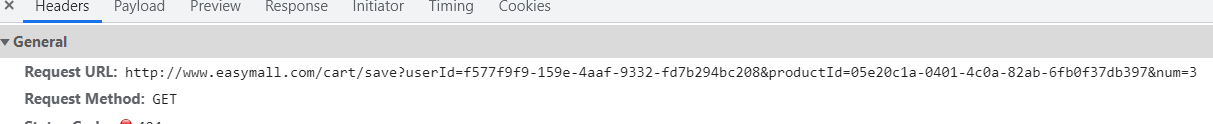
中度可信度描述已自动生成

图形用户界面

描述已自动生成

**3.3新增我的购物车**

### 3.3.1 js源码展示(product-list.html,product-info.html)



[http://www.easymall.com/cart/save?**userId**=f577f9f9-159e-4aaf-9332-fd7b294bc208&**productId**=05e20c1a-0401-4c0a-82ab-6fb0f37db397&**num**=3](http://www.easymall.com/cart/save?userId=f577f9f9-159e-4aaf-9332-fd7b294bc208&productId=05e20c1a-0401-4c0a-82ab-6fb0f37db397&num=3)

product\_info.html:



product\_list.html:

****

[http://www.easymall.com/cart/save?**userId**=f577f9f9-159e-4aaf-9332-fd7b294bc208&**productId**=05e20c1a-0401-4c0a-82ab-6fb0f37db397&**num**=3](http://www.easymall.com/cart/save?userId=f577f9f9-159e-4aaf-9332-fd7b294bc208&productId=05e20c1a-0401-4c0a-82ab-6fb0f37db397&num=3)

传递的参数只包括：userId，productId,num，缺少了product\_image,product\_name, product\_price.

t\_cart表结构

表格

描述已自动生成

逻辑关系:前端页面在新增传递的参数中,cart对象缺少属性值,我们可以利用购物车的微服务调用商品的微服务进行productId的商品信息查询.获取的数据生成一个完整的Cart购物车对象进行insert

步骤：

判断购物车商品是否需要真正的新增

userId productId,查询数据库表格是否已有数据

如果已经存在,将旧num和新num叠加重新update到表格中，如果不存在才是insert新增;

### 3.3.2 接口文件

表格

中度可信度描述已自动生成

### 代码实现

**1）CartController**

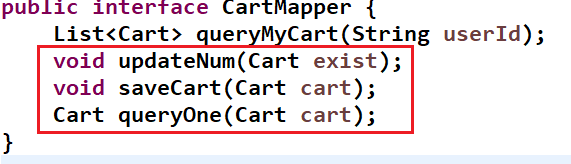
文本, 信件

描述已自动生成

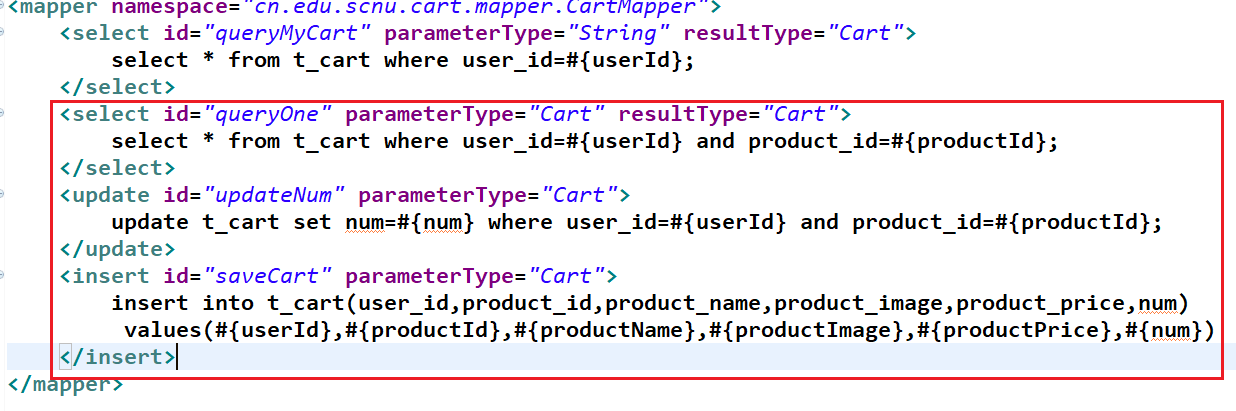
1. **CartService**

****

**3）CartMapper**

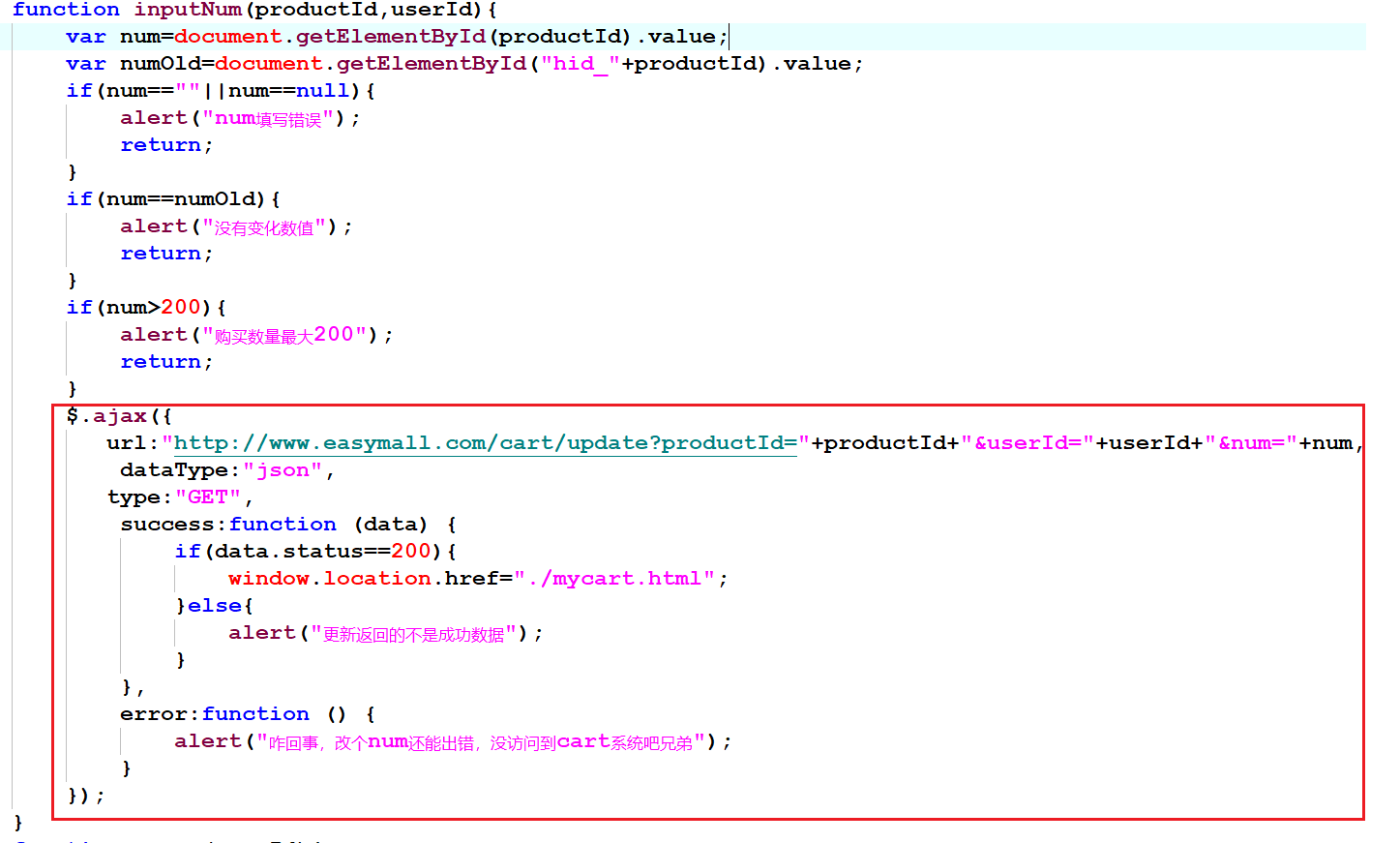


**4）CartMapper.xml**

****

**3.4购物车更新num功能**

### 3.4.1 js源码展示(mycart.html)



### 3.4.2 js代码解释

发起ajax请求,成功后返回数据,跳转页面重新查询一次

**url:"http://www.easymall.com/cart/update?productId="+productId+"&userId="+userId+"&num="+num**

图形用户界面, 文本, 应用程序

描述已自动生成

<http://www.easymall.com/cart/update>

[http://127.0.0.1:9005/zuul-cart/cart/manage/update](http://127.0.0.1:9005/zuul-cart/cart/manage/query)

### 3.4.3 接口文件

文本

描述已自动生成

### 3.4.4功能编写

1) CartController

文本

描述已自动生成

2) CartService

图形用户界面

中度可信度描述已自动生成

3)CartMapper(前面已经写了updateNum方法)

4)CartMapper.xml(前面已经写了)

**3.5删除商品购物车**

### 3.5.1 js源码展示



### 3.5.2 js代码解释

发起ajax请求,成功后返回数据,跳转页面重新查询一次

**url:"http://www.easymall.com/cart/delete?productId="+productId+"&userId="+userId**

图形用户界面, 文本, 应用程序

描述已自动生成

<http://www.easymall.com/cart/delete>

<http://127.0.0.1:9005/zuul-cart/cart/manage/delete>

### 3.5.3接口文件

文本

中度可信度描述已自动生成

### 3.5.4 代码编写

1) CartController

文本, 信件

描述已自动生成

2) CartService

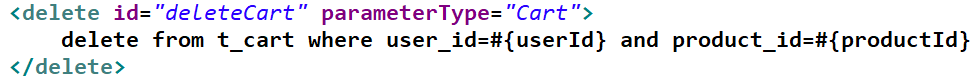
文本

中度可信度描述已自动生成

3)CartMapper

**void** deleteCart(Cart cart);

4) CartMapper.xml



**3.6 测试**

启动Nginx,启动七个工程：

文本

描述已自动生成