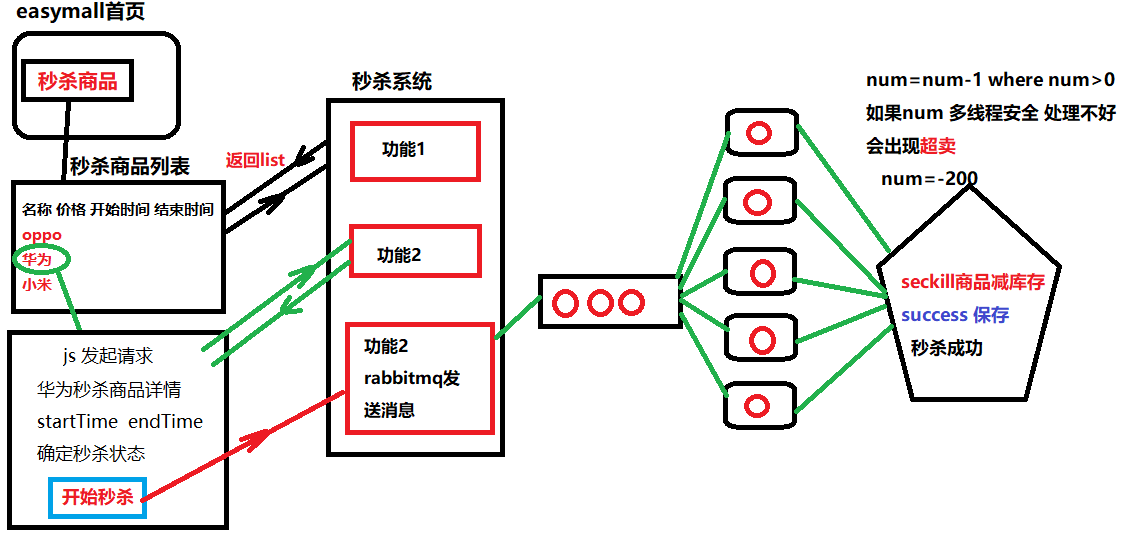
**编写秒杀功能代码**

**1.秒杀功能流程结构**



前端接收秒杀请求的系统的线程处理速度要快，这单位时间的并发量才会提高。

**2.数据库**

**2.1导入数据库**

在window上的mysql导入seckill数据库

seckill.txt文件

**2.2 表结构**

* 秒杀商品seckill(对应一个entity实体类Seckill)

**图形用户界面, 应用程序, 表格

描述已自动生成**

图表, 散点图

描述已自动生成

* 秒杀成功数据入库success(对应entity实体类Success)

表格

描述已自动生成

**3. 编写代码**

**3.1 页面**

秒杀功能一共需要2个html页面:

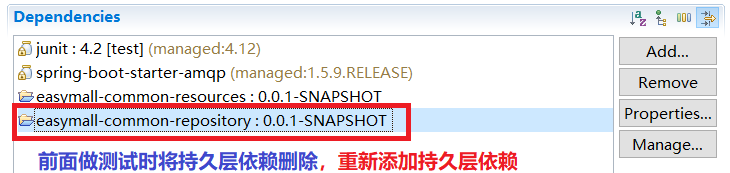
1）展示商品列表的页面

2）秒杀商品详情页面（点击秒杀）

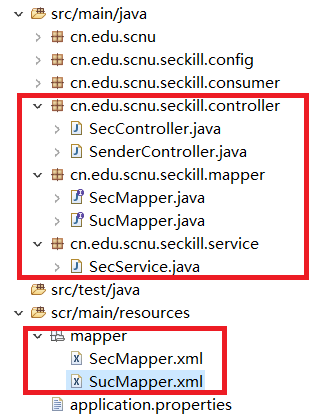
**3.2准备代码结构**

1.pom文件

添加**持久层**依赖



2.创建代码结构



**SecMapper.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.seckill.mapper.SecMapper"*>

</mapper>

**SucMapper.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.seckill.mapper.SucMapper"*>

</mapper>

**3.3 显示秒杀商品**

### 3.3.1功能需求演示

根据功能接口文件编写代码

|  |  |
| --- | --- |
| 后台接收 | seckill/manage/list |
| 请求方式 | get |
| 请求参数 | 空 |
| 返回数据 | List<Seckill>的查询数据 select \* from seckill |

### 3.3.2 SecController

文本

描述已自动生成

### 3.3.2 SecService

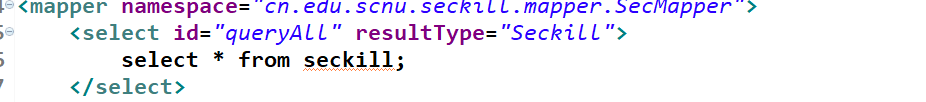
文本, 信件

描述已自动生成

### 3.3.3 SecMapper

List<Seckill> queryAll();

### 3.3.4 SecMapper.xml



### 3.3.5 各种配置

* 在application.properties配置文件中添加datasource mybatis 端口,注册eureka

端口10006

服务名称seckillservice

访问数据库seckill

**配置redis(超卖逻辑准备的)**

spring.rabbitmq.host=192.168.243.134

spring.rabbitmq.username=guest

spring.rabbitmq.password=guest

spring.rabbitmq.virtualHost=/

erver.port=10006

server.contextPath=/

spring.application.name=seckillservice

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

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

spring.datasource.url=jdbc:mysql:///seckill?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

#redis config

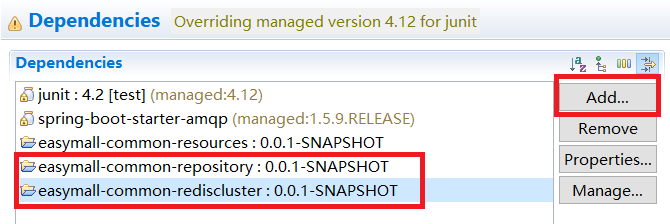
spring.redis.cluster.nodes=192.168.243.133:8000,192.168.243.133:8001,192.168.243.133:8002

spring.redis.cluster.maxTotal=200

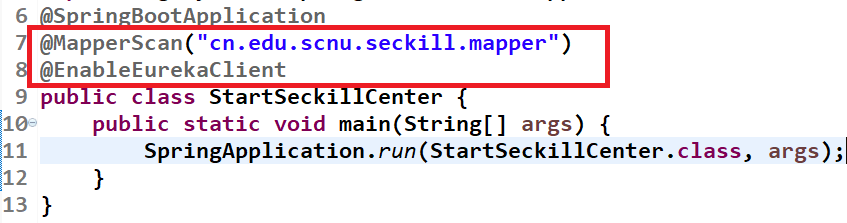
spring.redis.cluster.maxIdle=8

spring.redis.cluster.minIdle=2

* 在pom文件中依赖repository,redis



* 启动类
  + mapperScan
  + eurekaclient



* 和页面整合
  + **Nginx**

location /seckills {

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

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

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

}

* + **Zuul路由**

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

zuul.routes.zuul-seckill.serviceId=seckillservice

表格

描述已自动生成

**3.4单个秒杀商品数据访问**

|  |  |
| --- | --- |
| 后台接收 | /seckill/manage/detail?seckillId=1 |
| 请求方式 | Get |
| 请求参数 | Long seckillId |
| 返回数据 | Seckill根据id的查询对象 |

### 3.4.1 SecController

文本

描述已自动生成

### 3.4.2 SecService

文本

描述已自动生成

### 3.4.3 SecMapper

**public** Seckill queryOne(long seckillId);

### 3.4.4 SecMapper.xml

<select id="queryOne" parameterType="**long**" resultType="Seckill">

select \* from seckill where seckill\_id=#{seckillId}

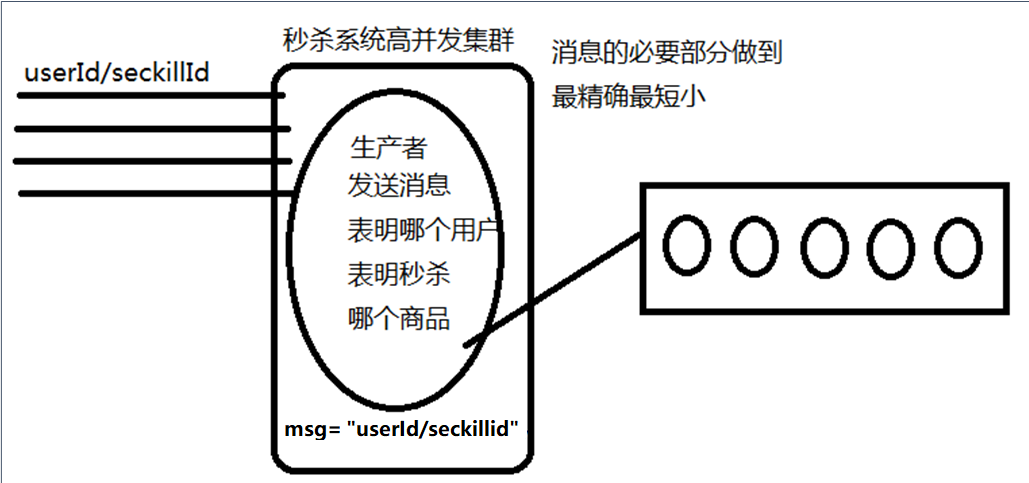
</select>



**4.秒杀核心功能**

**4.1请求迅速发送到消息队列**

在秒杀商品详情页面,海量用户在开始的一瞬间发起系统的高并发请求,每一个请求携带一个用户信息+秒杀商品信息(msg=userId+seckillId)



**4.2发起秒杀**

### 4.2.1功能需求演示

点击开始秒杀显示秒杀成功

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

### 4.2.2 js源码展示(seckill-detail.html)

**function** *startSeckill*(){

**$**.ajax({

[**url**:**"http://www.easymall.com/seckills/"**+***seckillId***](url:%22http://www.easymall.com/seckills/%22+seckillId),

**dataType**:**"json"**,

**type**:**"GET"**,

success:**function**(data){

**if**(data.**status**==200){

*alert*(**"秒杀成功"**);

}**else if**(data.**status**=202){

*alert*(**"兄弟你没有登录吧,那我哪知道你是谁?"**);

***window***.location.href=**"./login.html"**;

}**else**{

*alert*(**"秒杀失败,原因未知,算你倒霉"**);

windwo.**location**.**href**=**"./seckill-list.html"**;

}

},

error:**function**(){

*alert*(**"发送失败"**); }});}

### 4.2.3 js代码解释

将秒杀商品id携带到后台,拼接消息发送到rabbitmq,等待消费者处理。

### 4.2.4 接口文件

|  |  |
| --- | --- |
| js请求地址 | [http://www.easymall/seckills/{seckillId}](http://www.easymall/seckills/%7bseckillId%7d) |
| 后台接收 | /seckill/manage/{seckillId} |
| 请求方式 | Get |
| 请求参数 | Long seckillId 路径传参 |
| 返回数据 | **SysResult**的返回对象  Integer status 200表示秒杀成功  String msg:ok表示成功  Object data:其他数据 |
| 备注 | 根据cookie获取用户信息,拼接用户数据到消息中,绑定一个秒杀的商品 |

由于前端界面js没有实现只有登录用户才能秒杀，本例子此处的userId用随机生成的手机号代替

### 4.2.5 控制器接收请求

SecController作为发送消息的生产者引入RabbitTemplate进行msg的发送

图形用户界面, 文本, 应用程序

描述已自动生成

### 4.2.5 消费端测试代码

此处先在消费端打印出提示消息：

文本

描述已自动生成

### 4.2.6 启动rabitMQ

文本, 信件

描述已自动生成

访问rabbitMQ控制台：



进入mysql（window上的mysql）,修改秒杀开始时间

文本

描述已自动生成

### 4.2.7 访问详情页进入秒杀

启动Nginx、启动eureka、启动zuul、启动seckill。进入秒杀界面

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

查看eclipse中秒杀系统控制台，查看提示信息：

文本

低可信度描述已自动生成

查看RabitMQ控制台，看到队列：

图形用户界面, 文本, 应用程序

描述已自动生成

**4.3消费端代码**

实现seckill商品减库存

如果库存能减，则记录 秒杀成功者

如何避免超卖

在减库存的设计sql代码做限制

### 4.3.1.SecMapper.xml

文本

描述已自动生成

<update id="updateNum">

update seckill set number=number-1 where

#{nowTime} &lt;end\_time and #{nowTime} &gt;start\_time

and seckill\_id=#{seckillId} and number &gt;0;

</update>

### 4.3.2.SecMapper.java接口

在接口中添加方法：

**public** **int** updateNum(@Param("seckillId")Long seckillId, @Param("nowTime")Date nowTime);

### 4.3.3. SeckillConsumer.java 消费者

版本1，没防止超卖

消费端消费消息入库数据代码

**package** cn.edu.scnu.seckill.consumer;

**import** java.util.Date;

**import** org.springframework.amqp.rabbit.annotation.RabbitListener;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** com.easymall.common.pojo.Success;

**import** cn.edu.scnu.seckill.config.RabbitmqConfig;

**import** cn.edu.scnu.seckill.mapper.SecMapper;

**import** cn.edu.scnu.seckill.mapper.SucMapper;

@Component

**public** **class** SeckillConsumer {

@Autowired

**private** SecMapper secMapper;

@Autowired

**private** SucMapper sucMapper;

// 声明式注解,底层实现异步监听

@RabbitListener(queues = RabbitmqConfig.***qName***)

**public** **void** processMsg(String msg) {

// msg包含了用户信息,和商品信息

// 根据商品信息,秒杀的减库存。

// 将用户信息和秒杀商品入库保存作为成功秒杀数据 success

// 将用户userId和seckillId分离开

String userId = msg.split("/")[0];

Long seckillId = Long.*parseLong*(msg.split("/")[1]);

// 执行减库存 对当前商品的库存减1,条件number>0 nowTime<endTime,nowTime>startTime

Date nowTime = **new** Date();

**int** result = secMapper.updateNum(seckillId, nowTime);

**if** (result == 0) {// 库存没有减成功

// 说明卖完了,处理失败逻辑

System.***out***.println(userId + "用户秒杀失败");

**return**;

}

// 封装成功的数据success对象,新增到success表格

Success success = **new** Success();

success.setCreateTime(nowTime);

success.setSeckillId(seckillId);

success.setUserId(userId);

success.setState(1);

sucMapper.saveSuc(success);

}

}

### 4.3.4 测试

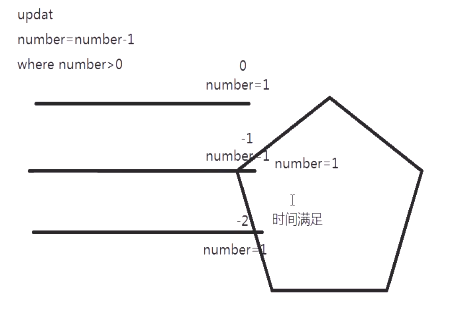
启动Nginx、启动eureka、启动zuul、启动seckill。进入秒杀界面

点击秒杀后，查看seckill表是否减少库存，查看success表是否添加了记录。

**4.4 防止超卖**

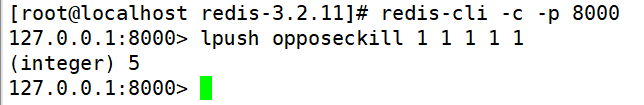
### 4.4.1 防止超卖

成功执行完秒杀,需要考虑超卖现象,可以使用redis中的数据类型list,将秒杀商品的个数对应元素提前存放到redis中,秒杀之前先rpop取出数据，取数据成功的人才能调用减库存逻辑



图示

描述已自动生成



### 4.4.2 消费端代码第2版

**package** cn.edu.scnu.seckill.consumer;

**import** java.util.Date;

**import** org.springframework.amqp.rabbit.annotation.RabbitListener;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** com.easymall.common.pojo.Success;

**import** cn.edu.scnu.seckill.config.RabbitmqConfig;

**import** cn.edu.scnu.seckill.mapper.SecMapper;

**import** cn.edu.scnu.seckill.mapper.SucMapper;

**import** redis.clients.jedis.JedisCluster;

@Component

**public** **class** SeckillConsumer {

@Autowired

**private** SecMapper secMapper;

@Autowired

**private** SucMapper sucMapper;

@Autowired

**private** JedisCluster cluster;

// 声明式注解,底层实现异步监听

@RabbitListener(queues = RabbitmqConfig.***qName***)

**public** **void** processMsg(String msg) {

// 假设当前方法的参数就是接受的rabbitmq的某个队列的消息

// System.out.println("消费端接收消息:" + msg);

// msg包含了用户信息,和商品信息

// 根据商品信息,秒杀的减库存。

// 将用户信息和秒杀商品入库保存作为成功秒杀数据 success

// 将用户userId和seckillId分离开

//判断redis数据获取成功的情况下才修改数据库

**if**(!cluster.exists("opposeckill")){

System.***out***.println("该商品已经秒杀完毕！");

**return**;

}

String rpop=cluster.rpop("opposeckill");

**if**(rpop==**null**){

//当前消费者没有从redis的list中获取导数据

//说明list中的元素为空，商品卖完

System.***out***.println("该商品已经秒杀完毕！");

**return**;

}

String userId = msg.split("/")[0];

Long seckillId = Long.*parseLong*(msg.split("/")[1]);

// 执行减库存 对当前商品的库存减1,条件number>0 nowTime<endTime,nowTime>startTime

Date nowTime = **new** Date();

**int** result = secMapper.updateNum(seckillId, nowTime);

**if** (result == 0) {// 库存没有减成功

// 说明卖完了,处理失败逻辑

System.***out***.println(userId + "用户秒杀失败");

**return**;

}

// 封装成功的数据success对象,新增到success表格

Success success = **new** Success();

success.setCreateTime(nowTime);

success.setSeckillId(seckillId);

success.setUserId(userId);

success.setState(1);

**try**{

sucMapper.saveSuc(success);

}**catch**(Exception e){

e.printStackTrace();

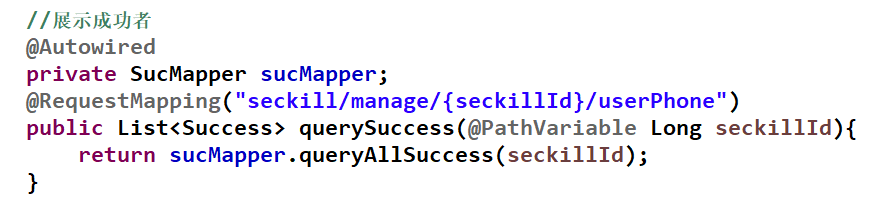
**return**;

}

}

}

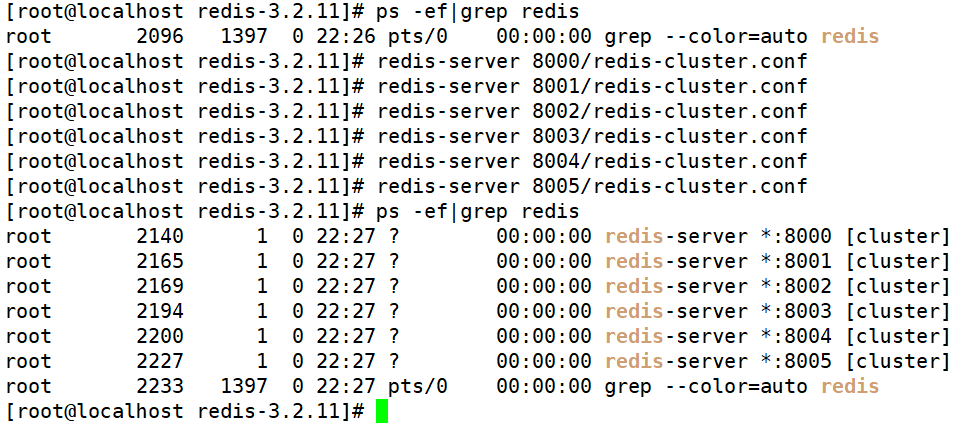
**4.4成功秒杀者信息展示**



表格

描述已自动生成**5 秒杀功能测试**

**5.1 启动redis**



**5.2 启动Nginx、eureka、zuul工程和seckill工程**