**41.登录功能整合JedisCluster**

**1.1构造一个抽取出来的公用工程common-rediscluster**

* 新建quistart工程

easymall-common-rediscluster

图形用户界面, 文本, 应用程序

描述已自动生成

* pom文件中继承springboot环境easymall-parent

图表, 条形图

描述已自动生成

* 单独依赖starter-redis，

删除掉junit依赖，添加starter-redis依赖

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-redis</artifactId>

<version>1.4.7.RELEASE</version>

</dependency>

</dependencies>

* 创建配置类

删除工程中自带的包“cn.edu.scnu.easymall\_common\_rediscluster”

创建包“cn.edu.scnu.config”，将测试工程的RedisClusterConfig类拷贝过来

**package** cn.edu.scnu.config;

**import** java.util.HashSet;

**import** java.util.List;

**import** java.util.Set;

**import** org.apache.commons.pool2.impl.GenericObjectPoolConfig;

**import** org.springframework.boot.context.properties.ConfigurationProperties;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** redis.clients.jedis.HostAndPort;

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

@Configuration

@ConfigurationProperties(prefix="spring.redis.cluster")

**public** **class** RedisClusterConfig {

**private** List<String> nodes;

**private** Integer maxTotal;

**private** Integer maxIdle;

**private** Integer minIdle;

//初始化JedisCluster的方法

@Bean

**public** JedisCluster initJedisCluster(){

//收集节点信息

Set<HostAndPort> set=**new** HashSet<HostAndPort>();

**for** (String node : nodes) {

//node="192.168.243.133:8000"

String host=node.split(":")[0];

Integer port=Integer.

*parseInt*(node.split(":")[1]);

set.add(**new** HostAndPort(host,port));

}

GenericObjectPoolConfig config=**new** GenericObjectPoolConfig();

config.setMaxTotal(maxTotal);

config.setMaxIdle(maxIdle);

config.setMinIdle(minIdle);

**return** **new** JedisCluster(set,config);

}

//自动生成getter和setter方法

}

**1.2 修改用户系统**

* 修改用户系统中配置文件application.properties,指定使用的cluster

添加以下属性值

#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

* 修改pom.xml，添加依赖，依赖上面新创建的公用工程

<dependency>

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

<artifactId>easymall-common-rediscluster</artifactId>

<version>0.0.1-SNAPSHOT</version>

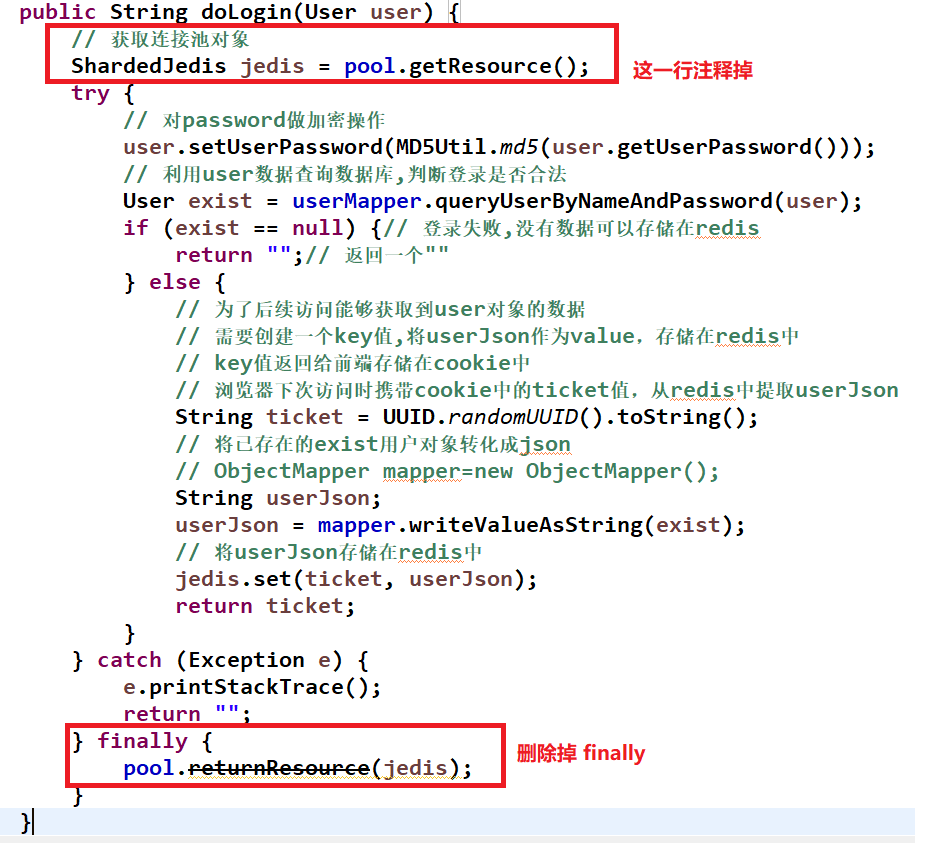
</dependency>

* 在UserService代码中直接注入JedisCluster,在方法中就可以实现缓存redis数据库使用。加上注入代码：

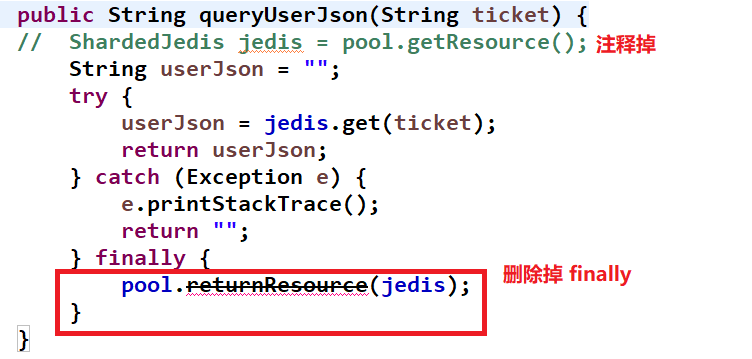
@Autowired

**private** JedisCluster jedis;

修改doLogin代码：



修改queryUserJson代码：



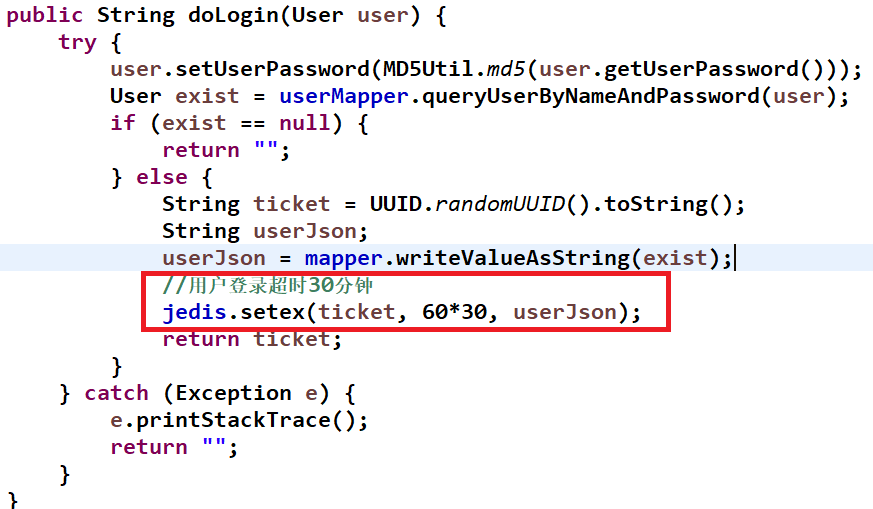
**1.3用户系统的业务逻辑修改**

### 1.3.1 登录超时

之前的代码，用户的ticket的值一直保存在redis中。登录状态一直存在，不符合业务逻辑，必须设置登录超时时间：

//用户登录超时30分钟

jedis.setex(ticket, 60\*30, userJson);

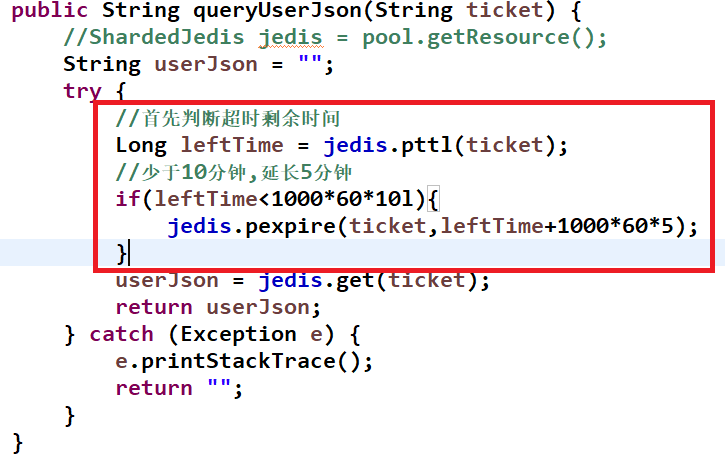


如果写死了用户登录后30分钟超时，当用户连续操作了30分钟时，到达超时时间，用户操作了过程**突然断开**，用户必须重新登录后才可以继续操作，**用户体验较差**。

解决办法：制定一个阈值，用户登录后不断的操作,一旦时间到达这个阈值，就续租;

一旦发现某次访问redis后.ticket剩余时间少于10分钟,自动给延长5分钟

在UserService中的queryUserJson方法中,校验登录状态时,添加对时间剩余的判断。



//首先判断超时剩余时间

Long leftTime = jedis.pttl(ticket);

//少于10分钟,延长5分钟

**if**(leftTime<1000\*60\*10l){

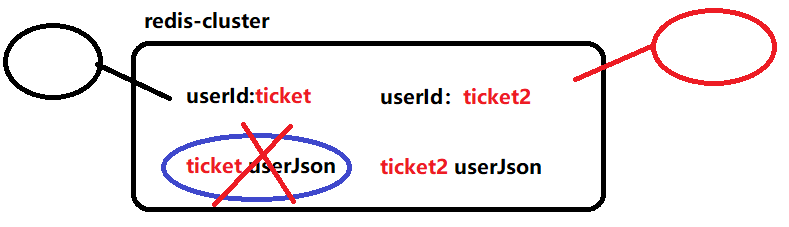
jedis.pexpire(ticket,leftTime+1000\*60\*5);

}

### 1.3.2 登录顶替操作(只允许一个用户登录)

一个用户在一个电脑登录后，接着用**同样账号**同时在另外一个电脑上登录，也就是说一个账号多处登录。对于收费网站，怎么防止一个账号多处登录的情况？

登录顶替：当一个账号在另外一个地方登录时，直接清除前面的登录状态。



实现方法：使用userId标识ticket， redis存储key=userId value=ticket。

用户使用userId登录时，通过userId判断ticket是否存在，如果的存在，则表示前面登录过，删除掉原先的ticket值,保存新的ticket2值，删除原先的ticket后，原先的登录状态就被清除，后面的登录顶替了前面的登录，这样就实现了登录顶替。



//判断当前用户是否曾经有人登录过

String existTicket=jedis.get("user\_logined\_"+exist.getUserId());

//顶替实现.不允许前一个登录的人ticket存在

**if**(StringUtils.*isNotEmpty*(existTicket)){

jedis.del(existTicket);

}

//定义当前客户端登录的信息 userId:ticket

jedis.setex("user\_logined\_"+exist.getUserId(), 60\*30,ticket);

### 1.3.3 实现一个账户最多3个用户同时登录

提示:使用redis的list结构的数据用userId相关的字符串做list的key值,元素内容存储ticket的值,判断list的长度,一旦大于3,将进行顶替,lpush新增使用rpop顶替删除,rpush新增使用lpop删除

**2.商品系统整合JedisCluster**

**2.1 使用redis完成商品数据的缓存**

**1）在pom文件中添加依赖（common-redistCluster工程）**

<dependency>

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

<artifactId>easymall-common-rediscluster</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

**2）修改商品系统中application.properties,指定使用的cluster**

#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

**3）注入jedisCluster对象实现商品查询缓存**

在ProductService中注入对象

@Autowired

private JedisCluster jedis;

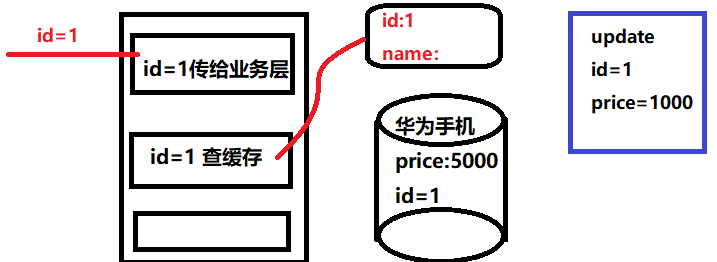
修改queryById方法文本

描述已自动生成

**3.高并发锁的使用**

**3.1 redis缓存需要和数据库数据保持一致**

缓存中存储的数据要根据对数据的删改做缓存的删改



例如:数据库中商品的价钱改变,由于缓存数据的存在.会导致至少2天用户查询的价钱是错误的;

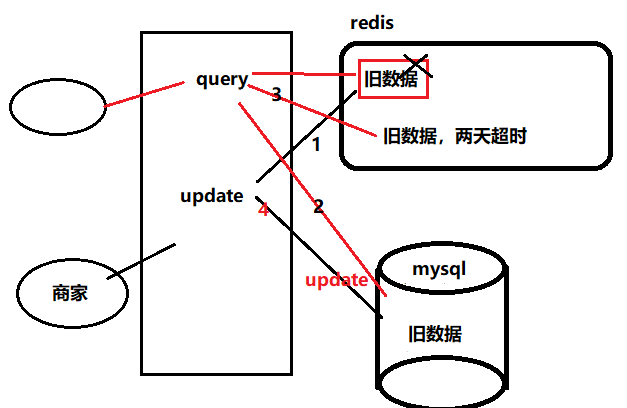
解决办法:只要商家对商品进行修改/删除,就将缓存数据删除;

文本

描述已自动生成

**3.2 高并发下的缓存问题**

描述问题:商家修改商品之前删除了缓存,但是由于高并发的查询在还没有修改数据库之前,用户就把数据库的旧数据重新利用查询的逻辑,放到缓存中.这样就导致,商家修改后的数据又和缓存不一致了;

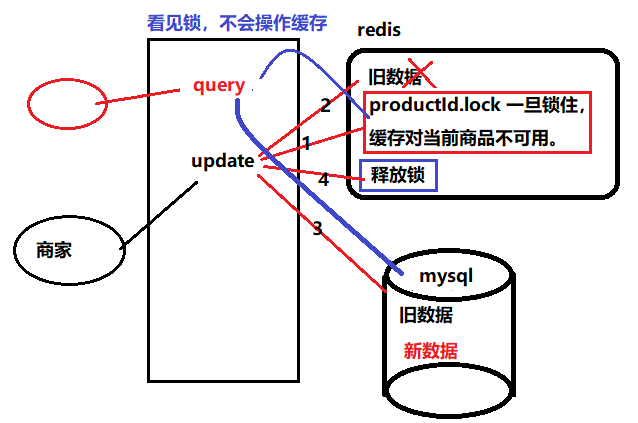


解决方法在修改完毕之前:缓存的新增要上锁(redis上锁)

### 3.2.1更新过程的加锁

手机屏幕截图

描述已自动生成



### 3.2.2查询商品逻辑

