**4-15｜延迟双删功能实现**

本章节我们主要讲解如何通过使用RocketMQ的延迟消息去实现延迟双删功能。

相关依赖引入：

|  |
| --- |
| Java <rocketmq.client.version>4.8.0</rocketmq.client.version>   <dependency>  <groupId>org.apache.rocketmq</groupId>  <artifactId>rocketmq-client</artifactId>  <version>${rocketmq.client.version}</version> </dependency> |

首先是我们的RocketMQ配置类，定义我们的消费者配置类和生产者配置类：

生产者配置类：

|  |
| --- |
| Java package org.qiyu.live.user.provider.config;  import org.springframework.boot.context.properties.ConfigurationProperties; import org.springframework.context.annotation.Configuration;  /\*\*  \* 生产者的配置信息  \*  \* @Author idea  \* @Date: Created in 16:39 2023/5/21  \* @Description  \*/ @ConfigurationProperties(prefix="qiyu.rmq.producer") @Configuration public class RocketMQProducerProperties {   //rocketmq的nameSever地址  private String nameSrv;  //分组名称  private String groupName;  //消息重发次数  private int retryTimes;  //发送超时时间  private int sendTimeOut;   public String getNameSrv() {  return nameSrv;  }   public void setNameSrv(String nameSrv) {  this.nameSrv = nameSrv;  }   public String getGroupName() {  return groupName;  }   public void setGroupName(String groupName) {  this.groupName = groupName;  }   public int getRetryTimes() {  return retryTimes;  }   public void setRetryTimes(int retryTimes) {  this.retryTimes = retryTimes;  }   public int getSendTimeOut() {  return sendTimeOut;  }   public void setSendTimeOut(int sendTimeOut) {  this.sendTimeOut = sendTimeOut;  }   @Override  public String toString() {  return "RocketMQProducerProperties{" +  "nameSrv='" + nameSrv + '\'' +  ", groupName='" + groupName + '\'' +  ", retryTimes=" + retryTimes +  ", sendTimeOut=" + sendTimeOut +  '}';  } } |

消费者配置类：

|  |
| --- |
| Java package org.qiyu.live.user.provider.config;  import org.springframework.boot.context.properties.ConfigurationProperties; import org.springframework.context.annotation.Configuration;  /\*\*  \* @Author idea  \* @Date: Created in 16:48 2023/5/21  \* @Description  \*/ @ConfigurationProperties(prefix = "qiyu.rmq.consumer") @Configuration public class RocketMQConsumerProperties {   //rocketmq的nameSever地址  private String nameSrv;  //分组名称  private String groupName;   public String getNameSrv() {  return nameSrv;  }   public void setNameSrv(String nameSrv) {  this.nameSrv = nameSrv;  }   public String getGroupName() {  return groupName;  }   public void setGroupName(String groupName) {  this.groupName = groupName;  }   @Override  public String toString() {  return "RocketMQConsumerProperties{" +  "nameSrv='" + nameSrv + '\'' +  ", groupName='" + groupName + '\'' +  '}';  } } |

上述的两个配置类，主要是用于映射我们在SpringBoot的配置文件中编写的配置项，下边两个类是对于的启动配置：

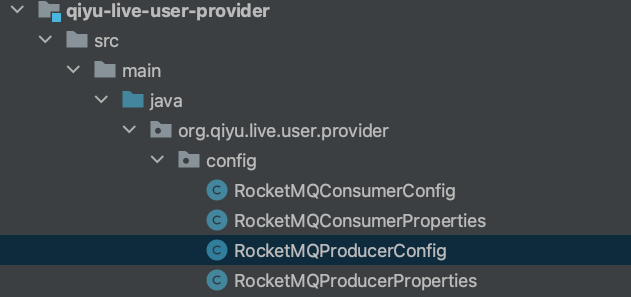
首先是我们的消费者启动配置：

|  |
| --- |
| Java package org.qiyu.live.user.provider.config;  import com.alibaba.fastjson.JSON; import jakarta.annotation.Resource; import org.apache.rocketmq.client.consumer.DefaultMQPushConsumer; import org.apache.rocketmq.client.consumer.listener.ConsumeConcurrentlyContext; import org.apache.rocketmq.client.consumer.listener.ConsumeConcurrentlyStatus; import org.apache.rocketmq.client.consumer.listener.MessageListenerConcurrently; import org.apache.rocketmq.client.exception.MQClientException; import org.apache.rocketmq.common.consumer.ConsumeFromWhere; import org.apache.rocketmq.common.message.MessageExt; import org.idea.qiyu.live.framework.redis.starter.key.UserProviderCacheKeyBuilder; import org.qiyu.live.user.dto.UserDTO; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.InitializingBean; import org.springframework.context.annotation.Configuration; import org.springframework.data.redis.core.RedisTemplate;  import java.util.List;  /\*\*  \* RocketMQ的消费者bean配置类  \*  \* @Author idea  \* @Date: Created in 16:50 2023/5/21  \* @Description  \*/ @Configuration public class RocketMQConsumerConfig implements InitializingBean {   private static final Logger LOGGER = LoggerFactory.getLogger(RocketMQConsumerConfig.class);   @Resource  private RocketMQConsumerProperties consumerProperties;  @Resource  private RedisTemplate<String, Object> redisTemplate;  @Resource  private UserProviderCacheKeyBuilder userProviderCacheKeyBuilder;   @Override  public void afterPropertiesSet() throws Exception {  initConsumer();  }   public void initConsumer() {  try {  //初始化我们的RocketMQ消费者  DefaultMQPushConsumer defaultMQPushConsumer = new DefaultMQPushConsumer();  defaultMQPushConsumer.setNamesrvAddr(consumerProperties.getNameSrv());  defaultMQPushConsumer.setConsumerGroup(consumerProperties.getGroupName());  defaultMQPushConsumer.setConsumeMessageBatchMaxSize(1);  defaultMQPushConsumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME\_FROM\_FIRST\_OFFSET);  defaultMQPushConsumer.subscribe("user-update-cache", "\*");  defaultMQPushConsumer.setMessageListener(new MessageListenerConcurrently() {  @Override  public ConsumeConcurrentlyStatus consumeMessage(List<MessageExt> msgs, ConsumeConcurrentlyContext context) {  String msgStr = new String(msgs.get(0).getBody());  UserDTO userDTO = JSON.parseObject(msgStr, UserDTO.class);  if (userDTO == null || userDTO.getUserId() == null) {  LOGGER.error("用户id为空，参数异常，内容:{}", msgStr);  return ConsumeConcurrentlyStatus.CONSUME\_SUCCESS;  }  //延迟消息的回调，处理相关的缓存二次删除  redisTemplate.delete(userProviderCacheKeyBuilder.buildUserInfoKey(userDTO.getUserId()));  LOGGER.error("延迟删除处理，userDTO is {}", userDTO);  return ConsumeConcurrentlyStatus.CONSUME\_SUCCESS;  }  });  defaultMQPushConsumer.start();  LOGGER.info("mq消费者启动成功,nameSrv is {}", consumerProperties.getNameSrv());  } catch (MQClientException e) {  throw new RuntimeException(e);  }  }  } |

接着是生产者的启动配置：

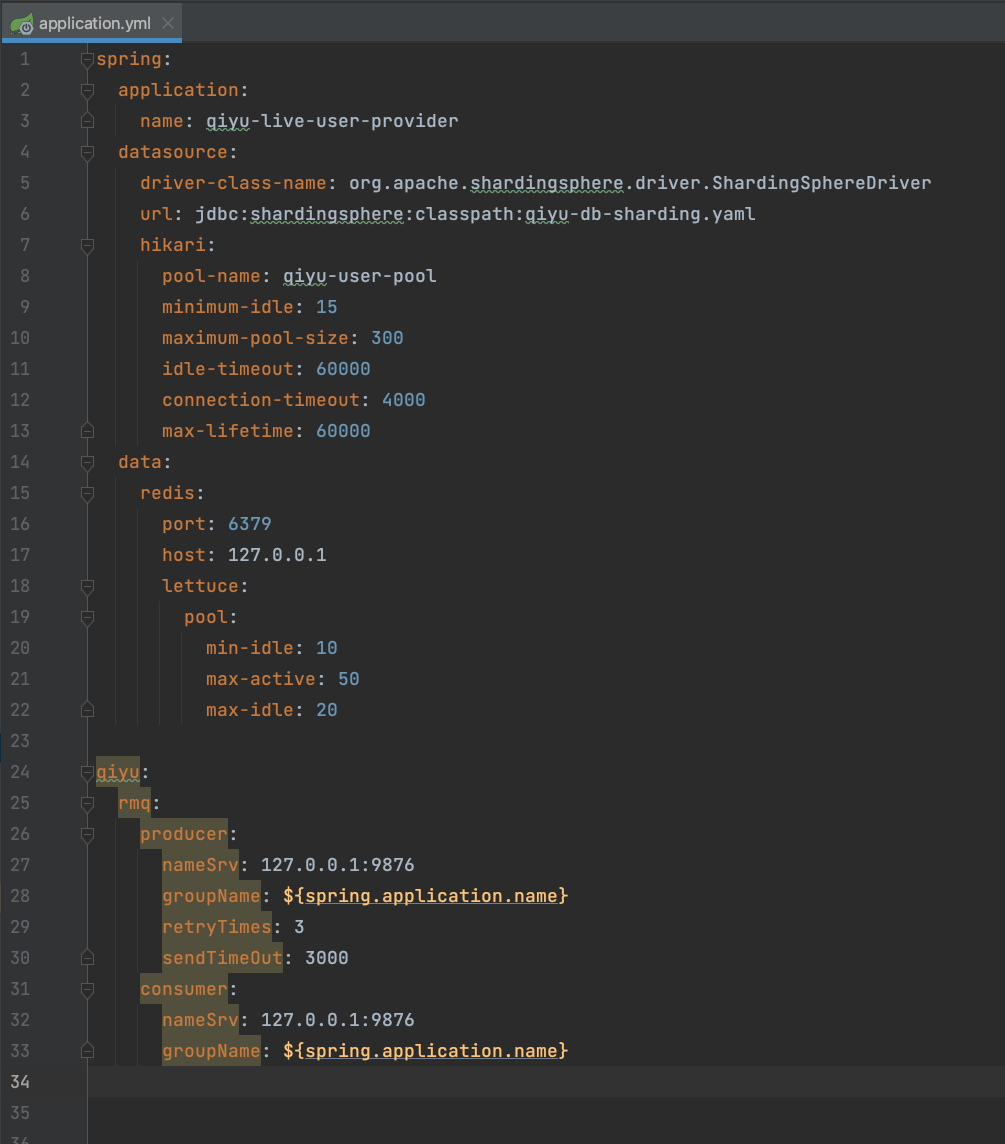
|  |
| --- |
| Java package org.qiyu.live.user.provider.config;  import jakarta.annotation.Resource; import org.apache.rocketmq.client.exception.MQClientException; import org.apache.rocketmq.client.producer.DefaultMQProducer; import org.apache.rocketmq.client.producer.MQProducer; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Value; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration;  import java.util.concurrent.\*;  /\*\*  \* RocketMQ的生产者bean配置类  \*  \* @Author idea  \* @Date: Created in 16:42 2023/5/21  \* @Description  \*/ @Configuration public class RocketMQProducerConfig {   private final static Logger LOGGER = LoggerFactory.getLogger(RocketMQProducerConfig.class);   @Resource  private RocketMQProducerProperties producerProperties;  @Value("${spring.application.name}")  private String applicationName;   @Bean  public MQProducer mqProducer() {  ThreadPoolExecutor asyncThreadPoolExecutor = new ThreadPoolExecutor(100, 150, 3, TimeUnit.MINUTES,  new ArrayBlockingQueue<>(1000), new ThreadFactory() {  @Override  public Thread newThread(Runnable r) {  Thread thread = new Thread(r);  thread.setName(applicationName + ":rmq-producer:" + ThreadLocalRandom.current().nextInt(1000));  return thread;  }  });  //初始化rocketmq的生产者  DefaultMQProducer defaultMQProducer = new DefaultMQProducer();  try {  defaultMQProducer.setNamesrvAddr(producerProperties.getNameSrv());  defaultMQProducer.setProducerGroup(producerProperties.getGroupName());  defaultMQProducer.setRetryTimesWhenSendFailed(producerProperties.getRetryTimes());  defaultMQProducer.setRetryTimesWhenSendAsyncFailed(producerProperties.getRetryTimes());  defaultMQProducer.setRetryAnotherBrokerWhenNotStoreOK(true);  //设置异步发送的线程池  defaultMQProducer.setAsyncSenderExecutor(asyncThreadPoolExecutor);  defaultMQProducer.start();  LOGGER.info("mq生产者启动成功,nameSrv is {}", producerProperties.getNameSrv());  } catch (MQClientException e) {  throw new RuntimeException(e);  }  return defaultMQProducer;  } } |

上述的四个类，全部都放在qiyu-live-user-provider模块的 org.qiyu.live.user.provider.config 包里面。



最后启动的时候要设置好相关的application.yml配置：

|  |
| --- |
| Java qiyu:  rmq:  producer:  nameSrv: 127.0.0.1:9876  groupName: ${spring.application.name}  retryTimes: 3  sendTimeOut: 3000  consumer:  nameSrv: 127.0.0.1:9876  groupName: ${spring.application.name} |



当我们进行缓存更新的时候，要进行缓存的及时删除，然后再发送mq的延迟消息，重点见org.qiyu.live.user.provider.service.impl.UserServiceImpl#updateUserInfo方法，修改如下：

|  |
| --- |
| Java @Override public boolean updateUserInfo(UserDTO userDTO) {  if (userDTO == null || userDTO.getUserId() == null) {  return false;  }  userMapper.updateById(ConvertBeanUtils.convert(userDTO, UserPO.class));  String key = userProviderCacheKeyBuilder.buildUserInfoKey(userDTO.getUserId());  //立即删除缓存  redisTemplate.delete(key);  try {  //发送延迟消息，触发二次删除缓存操作  Message message = new Message();  message.setBody(JSON.toJSONString(userDTO).getBytes());  message.setTopic("user-update-cache");  //延迟级别，1代表延迟一秒发送  message.setDelayTimeLevel(1);  mqProducer.send(message);  } catch (Exception e) {  throw new RuntimeException(e);  }  return true; } |