79_高阶实战: 登录模块基于 RocketMQ 发送消息替代 Spring Event 事件

儒猿架构官网上线,内有石杉老师架构课最新大纲,儒猿云平台详细介绍,敬请浏 览

官网: <u>www.ruyuan2020.com</u> (建议 PC 端访问)

1、开篇

上节课介绍了主动取消订单和延迟取消订单的测试过程,在小程序上进行下单、取消、等待操作模拟了两个场景,通过 log 日志的查看正式了我们的想法。本节课将针对登录模块进行改造,使用 RocketMQ 替代 Spring Event 事件完成发送消息的功能。今天包括如下内容:

- 定义登陆消息相关组件
- 发送以及消费登陆消息

2、定义登陆消息相关组件

之前对登陆的操作时通过 Spring Event 来实现的现在使用 RocketMQ 替换之,因此需要定义消息生产者、消息消费者和消息监听器。

如图 1 所示,在 spring/jms 目录下面建立 LoginProducerConfiguration 类,用来 定义登陆消息的生产者。在类中定义了 producerGroupName(生产者组名字), nameServerAddr(名字服务器地址)。在初始化方法 init 中设置上述两个值,并且启动消息生产者。

```
@author <a href="mailto:little@163.com">little</a>
version: 1.0
event
                                            8
9
10
11
12
13
14
15
16
17
18

    LogonEvent

                                                       * Description:mg 生产者配置类
   C LogonEventPublisher
                                                     public class LoginProducerConfiguration {
  C LogonListener
exception
                                                               日志管理组件
nandler handler
                                                          private static final Logger LOGGER = LoggerFactory.getLogger(LoginProducerConfiguration.class);
interceptor
ims ims
                                                            * 生产者group名称

    LoginConsumerConfiguration

    LoginMessageListener

                                            20
21
22
23
24
25
26
27
28
30
31
33
33
34
35
36
37
38
40
41
42
43
44
44
45
46
47
                                                          private String producerGroupName;

    LoginProducerConfiguration

                                                            * rocketmg的namesrv的地址

    MqConsumerConfiguration

  © MqDelayConsumerConfiguration
                                                          private String nameServerAddr;
  © MqProducerConfiguration
                                                            * 生产者

    OrderDelayMessageListener

                                                          */
private DefaultMQProducer producer;
public LoginProducerConfiguration(String producerGroupName, String nameServerAddr) {
    this.producerGroupName = producerGroupName;
    this.nameServerAddr = nameServerAddr;

    OrderMessageListener

mapper mapper
service
     ConsumerServiceImpl
                                                            * bean初始化执行方法
     C CouponServiceImpl
     c LoginEvenManagerImpl
                                                            * @throws Exception
     © OrderEvenManagerImpl
                                                          public void init() throws Exception {
     OrderServiceImpl
                                                               LOGGER.info("开始启动消息生产者服务...'
//创建一个消息生产者,并设置一个消息生产者
     C TeacherServiceImpl
                                                                producer = new DefaultMQProducer(producerGroupName);
  ConsumerService
                                                                 /指定 NameServer 地址

    CouponService

                                                                producer.setNamesrvAddr(nameServerAddr);
  I LoginEventManager
                                                                 //初始化 SpringProducer, 整个应用生命周期内只需要初始化一次
  OrderEventManager
                                                                producer.start():
                                                                LOGGER.info("消息生产者服务启动成功.");
  OrderService
  TeacherService
                                            48
utils
                                                            * bean消费时执行方法
validator
```

图 1

有了消息生产者,再定义消息的消费者。在同一目录下面,创建 LoginConusmerConfiguration 类,如图 2 所示,在类中也需要定义 consumerGroupName(消费者群名字),nameServerAddr(名字服务器地址),通过 setConsumeFromwhere 方法定义队列消费的方式,通过 subscribe 方 法定义消费 topic 下所有的消息以及 registerMessageListener 方法定义对应的消息 监听器,最后启动消息消费者。

```
□ dto
                                                      private DefaultMQPushConsumer consumer;
enums
                                                       * 订单消息的listener组件
event
                                                     exception
interceptor
                                        43
44
45
46
47
48
49
50
51
52
53
54
55
56
67
68
68
66
66
ims

    LoginConsumerConfiguration

    LoginMessageListener

    LoginProducerConfiguration

   © MqConsumerConfiguration
                                                       * bean加载是init的方法
   MqDelayConsumerConfiguration
                                                        @throws Exception
   © MaProducerConfiguration
                                                     public void init() throws Exception {
    LOGGER.info("开始启动消息消费者服务..."
    //创建一个消息消费者,并设置一个消息消费者;
   OrderMessageListener
                                                           consumer = new DefaultMQPushConsumer(consumerGroupName);
service
                                                           consumer.setNamesrvAddr(nameServerAddr);
▼ Dimimpl
                                                                               次启动是从队列头部开始消费还是队列屋部开始消费
     ConsumerServiceImpl
                                                           consumer.setConsumeFromWhere(ConsumeFromWhere.CONSUME_FROM_FIRST_OFFSET);
     CouponServiceImpl
                                                           consumer.subscribe(topicName, subExpression: "*");
     C LoginEvenManagerImpl

    OrderEvenManagerImpl

                                                          ///エ加州の通知時間
consumer.registerMessageListener(loginMessageListener);
// 消費者対象在使用之前必须要调用 start 初始化
consumer.start();
LOGGER.info("消息消费者服务启动成功.");
     C OrderServiceImpl
     C TeacherServiceImp

    ConsumerService

CamanCandaa
```

图 2

再来看看消费者的监听器,还是在 spring/jms 目录下创建 LoginMessageListener 类,该类实现了 MessageListenerOrderly 接口,特别是 consumeMessage 方法。如图 3 所示,该方法中首先将监听到的 consumer 消息从 jason 对象解析成 Consumer 对象,依次判断首次登陆、生日优惠券等逻辑,最后更新登陆次数。

```
@Override
public ConsumeOrderlyStatus consumeMessage(List<MessageExt> list, ConsumeOrderlyContext consumeOrderlyContext) {
    for (MessageExt ext : list) {
        String content = new String(ext.getBody(), StandardCharsets.UTF_8);
        LOGGER.info("received login message:{}", content);
        // पार्टा मार्गिक प्राथम विशेषा कर्णां क्षिति कर्णां 
     annotion
                                                                                                                          ▶ aspect
      ▶ 🖿 cache
                                                                                                                                                                                                // 用产信息消息
Consumer consumer = JSON.parseObject(content, Consumer.class);
// 模拟异步场景
      ▶ ☐ controller
                                                                                                                                                                                               ▶ ■ event
      exception
     ► I handler
                                                                                                                                                                                                // 画次質束學件,分及匹陽旁
if (consumer.getLoginTimes() == null || consumer.getLoginTimes() == 0) {
    LOGGER.info("消费者首次登录, 发放优惠券, 消费者信息{}",consumer);
    this.distributeCoupon(consumer, loginCouponId);
      ▶ interceptor
     ▼ 🛅 jms
                                                                                                                                                                                                 // 生日事件
if (!StringUtils.isEmpty(consumer.getBirthday())) {
C LoginMessageListener
                    © LoginProducerConfiguration
                                                                                                                                                                                                             MgConsumerConfiguration
                    © MqDelayConsumerConfiguration

    MgProducerConfiguration

                    © OrderDelayMessageListener
                    OrderMessageListener
      ▶ ➡ mapper
                                                                                                                                                                                                                                        // 次判別及以上ロ牙
LOGGER.info("消費者生日当月登录, 发放优惠券, 消费者信息{}",consumer);
this.distributeCoupon(consumer, birthdayLoginCouponId);
      ▼ 🖿 service
             ▼ 🛅 impl
                           ConsumerServiceImpl
                                                                                                                                                                                                /
// 更新登录次数
consumerService.updateLoginTimes(consumer.getId());
LOGGER.info("登录事件监听完成");
                           © CouponServiceImpl
                           © LoginEvenManagerImpl
                            C OrderEvenManagerImpl
                           © OrderServiceImpl
© TeacherServiceImpl
                                                                                                                                                                                     return ConsumeOrderlyStatus.SUCCESS;
```

图 3

最后将上面定义的消息生产者、消费者和监听器都注入到 Spring IoC 容器中进行管理。

如图 4 所示,在 resources 目录下面的 spring-jms.xml 文件中创建 3 个 bean 节点。分别定义了 loginMessageListener、loginConsumerConfiguration 和 loginProducerConfiguration 其中关于初始化的配置信息都从 application.properties 文件中读取。

```
</bean>

    CouponService

    LoginEventManager

    OrderEventManager

             OrderService
              TeacherService
        ▶ 🛅 utils
        ▶ 🖿 validator
                                                                <!-- 登录消息Listener-->
<bean id="loginMessageListener" class="com.ruyuan.little.project.spring.jms.LoginMessageListener" />
   resources
        mappe
        application.properties
                                                               idbc.properties
        a logback.properties
        alogback.xml
        amybatis-config.xml
         spring-jdbc.xml
        spring-jms.xml
                                                                (一) 甲周基生产者配置一>
-(中) 甲周基生产者配置一>
-(bean id="TogainProducerConfiguration" class="com.ruyuan.little.project.spring.jms.LoginProducerConfiguration" init-method="init" destroy"nethod="destroy">
-(constructor-arg name="ammeServerAddr" value="${rocketmq.namesrv.address}"/>
-(constructor-arg name="producerGroupName" value="${rocketmq.login.producer.group}"/>
-(bean>
web
gitignore
alittle-project-spring.iml
m pom.xml
README.md
External Libraries
```

图 4

3、发送以及消费登陆消息

介绍完了登陆消息的生产者、消费者和监听器的定义以后,再来看看它是如何应用 到登陆操作中的。

如图 5 所示,在 spring/controller 的目录下面,ConsumerController 作为后台 API 的入口类,其中有一个 login 方法,在方法体中条用了 consumerService 接口的 login 实现方法,并且传入 consumer 对象。



图 5

如图 6 所示,顺势看 spring/service/impl 下面的 ConsumerServiceImpl 类,其中实现了 login 方法,方法体中先通过传入的消费者 ID 获取消费者对象,然后通过 loginEventManager 的 loginEvent 方法发布登陆事件。

```
enums
                                                          48 🜒 @
▶ 🖿 event
                                          49
exception
                                          50
51
                                                               if (findConsumer == null) {
▶ □ handler
                                         52
53
54
55
56
57
58
59
60
61
62 1 @
                                                                       注册消费者信息
▶ interceptor
                                                                    consumerMapper.add(consumer);
▶ ims
                                                                    findConsumer = consumer;
▶ ➡ mapper
                                                                / 发布登录事件
service
                                                               loginEventManager.loginEvent(findConsumer);
   ▼ limpl
                                                               return findConsumer;
        C ConsumerServiceIm
        CouponServiceImpl

    LoginEvenManagerImpl

                                                          public int orderCreateInformDeductCredits(Order order) {
   if (order.getDeductCredits() <= 0) {</pre>

    OrderEvenManagerImpl

    OrderServiceImpl

                                                                    return 0:
                                          65
66
        C TeacherServiceImpl
                                                               Consumer consumer = consumerMapper.findById(order.getConsumerId());
if (consumer.getCredits() < order.getDeductCredits()) {
   throw new BusinessException(EducationBusinessErrorCodeEnum.CREDITS_NOT_ENOUGH.
      ConsumerService
                                          67
                                          68
      CouponService
                                          69
      LoginEventManager
                                                               return consumerMapper.deductCredits(order.getDeductCredits(), order.getConsumerId(
```

图 6

顺着 login 方法中调用 loginEvent 方法往下看,如图 7 所示,在 spring/service/impl 目录下面的 loginEventManagerImpl 类文件中实现了 loginEvent 方法,在方法中调用了 sendLoginMessage 方法,这个方法对传入的消息进行 setTopic 和 setBody 的处理然后通过 loginProducerConfiguration 中的

```
send 方法将消息发送出去。
event
exception
                                           @Override
                              39 ■↑
                                          public void loginEvent(Consumer consumer) { this.sendLoginMessage(consumer); }
nandler
                              42
ntercepto
                              43
                                            * 发布登录信息
                              46
                                              @param consumer 用户信息
service
                              47
                                           private void sendLoginMessage(Consumer consumer) {
                              49
                                               Message message = new Message();
message.setTopic(loginTopic);

    ConsumerServiceImpl

                              50
  CouponServiceImpl
                                               message.setBody(JSON.toJSONString(consumer).getBytes(StandardCharsets.UTF_8));

    LoginEvenManagerImpl

                              52
                                                    loginProducerConfiguration.getProducer().send(message);

    OrderEvenManagerImpl

                                                   LOGGER.info("send login message finished consumer:{}", consumer);

    OrderServiceImpl

                              55
                                               } catch (Exception e) {
                              56

    TeacherServiceImpl

                                                    LOGGER.error("send login message fail,error message:{}", e.getMessage());
ConsumerService
                                               }
CouponService
                              59
LoginEventManager
                              61
OrderEventManager
                              62
OrderService
```

图 7

4、总结

本节课使用 RocketMQ 替代 Spring Event 发送登陆消息,针对 RocketMQ 的消息 生产者、消费者和监听器进行了定义和使用等操作。下节课会测试一下 RocketMQ 改造之后的代码,看看执行效果。这里将代码给大家,下期见,拜拜。