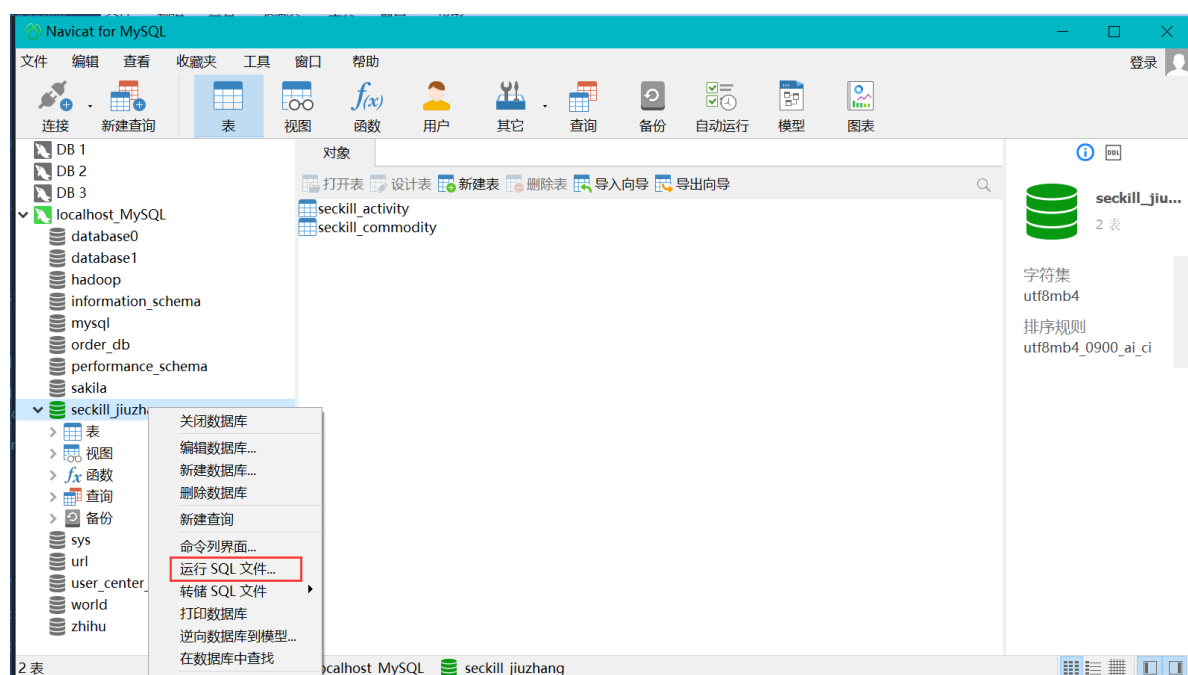
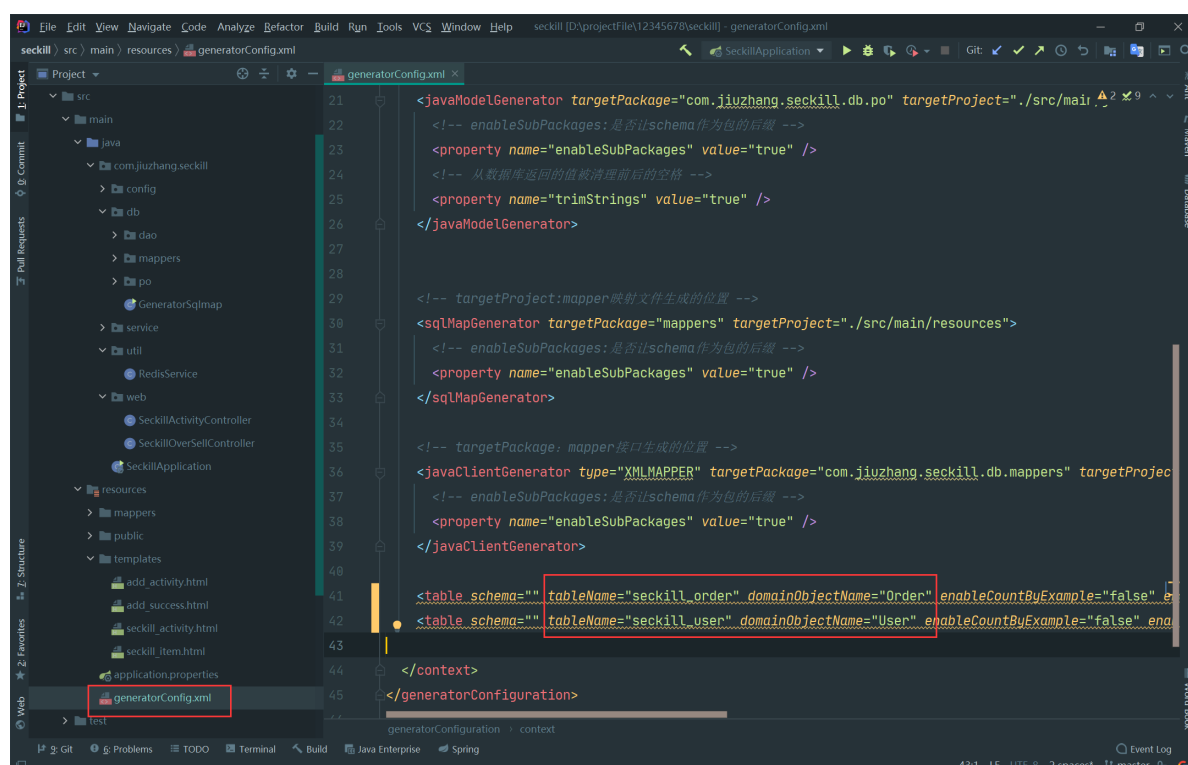


《保姆级教程》第七集 Order 和 User 表的引入 & RockerMQ 的发送、接收测试

1. 打开 navicat 运行 seckill_jiuzhang (下) .sql 文件



2. 修改 generatorConfig.xml 文件



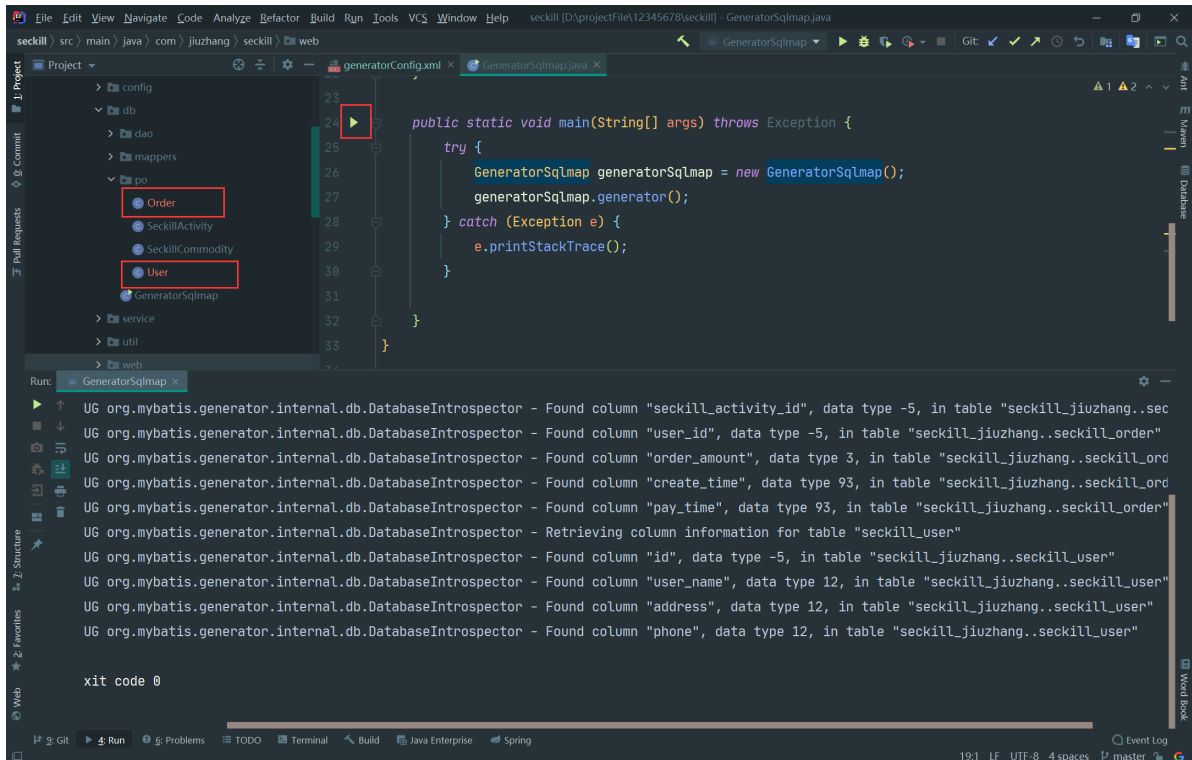
全网一手不加密IT课程，需要请加微信zzj97666九章来offer都有

```
<table schema="" tableName="seckill_order" domainObjectName="Order"
enableCountByExample="false" enableDeleteByExample="false"
enableSelectByExample="false" enableUpdateByExample="false"
selectByExampleQueryId="false"></table>

<table schema="" tableName="seckill_user" domainObjectName="User"
enableCountByExample="false" enableDeleteByExample="false"
enableSelectByExample="false" enableUpdateByExample="false"
selectByExampleQueryId="false"></table>
```

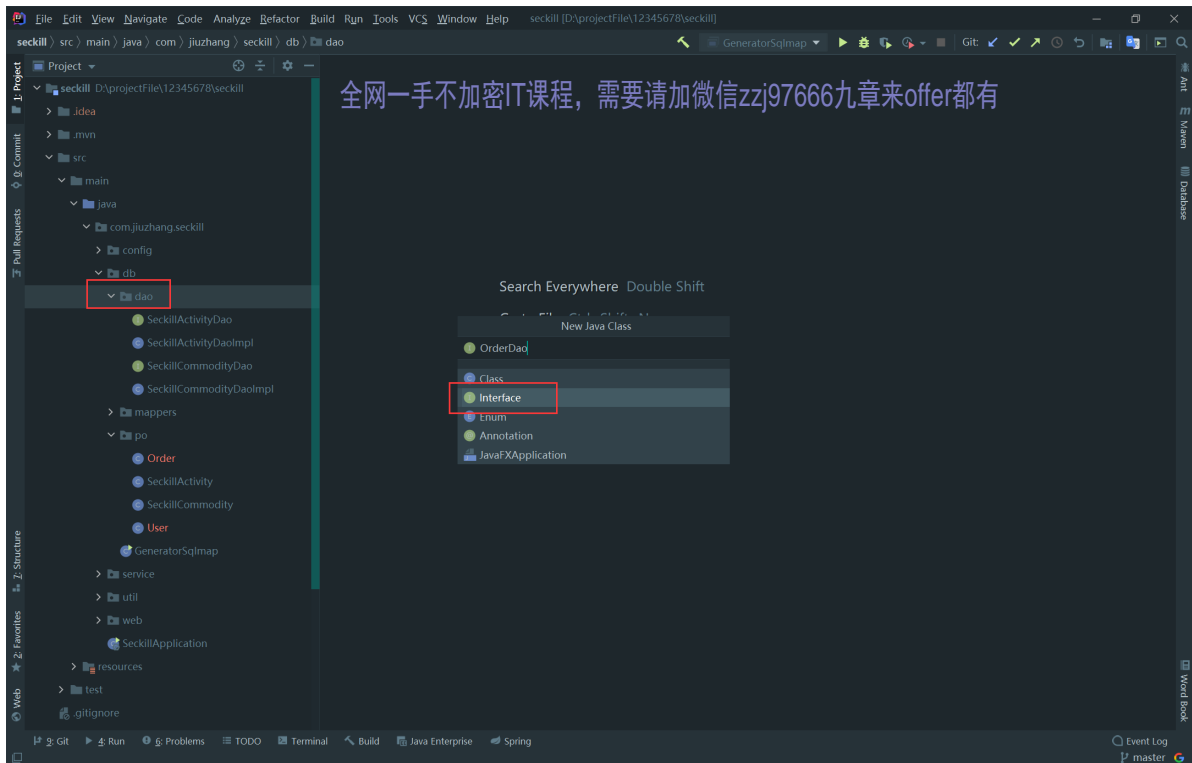
3. 执行 GeneratorSqlmap 的 main 方法，逆向生成相关文件

全网一手不加密IT课程，需要请加微信zzj97666九章来offer都有

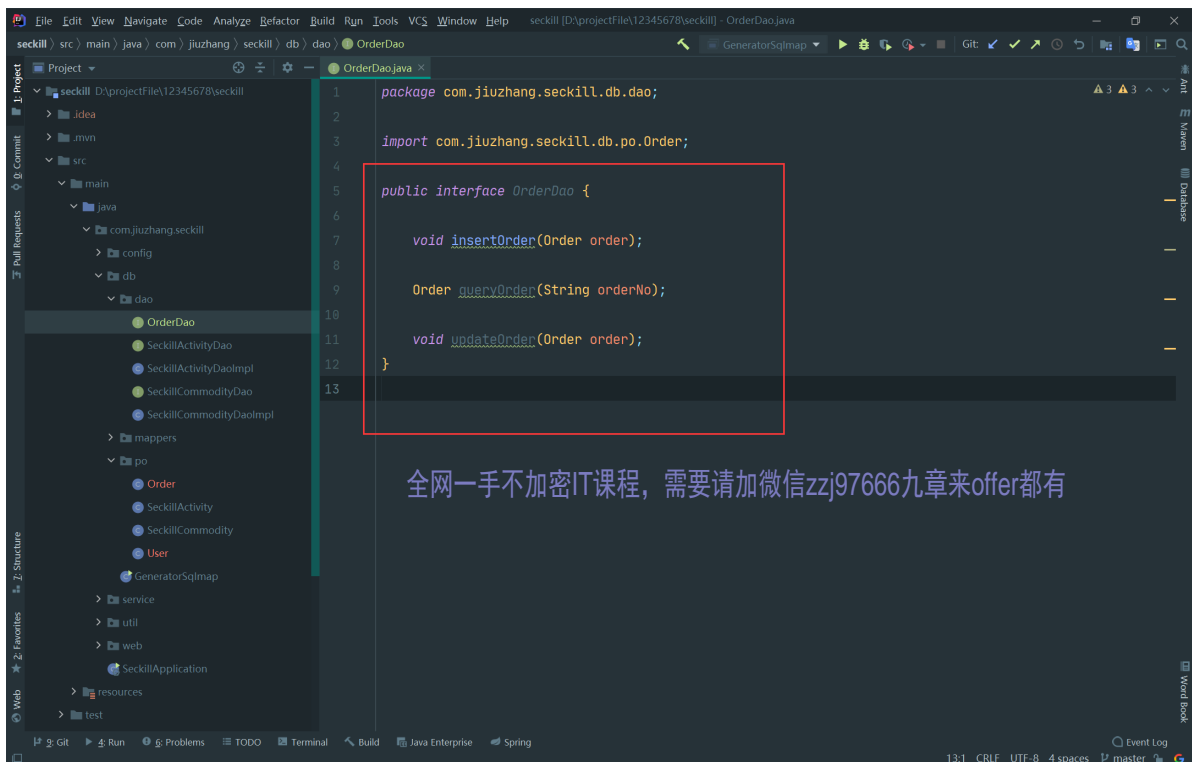


4. 创建 OrderDao 接口





5. 向 OrderDao 接口中添加三个抽象方法

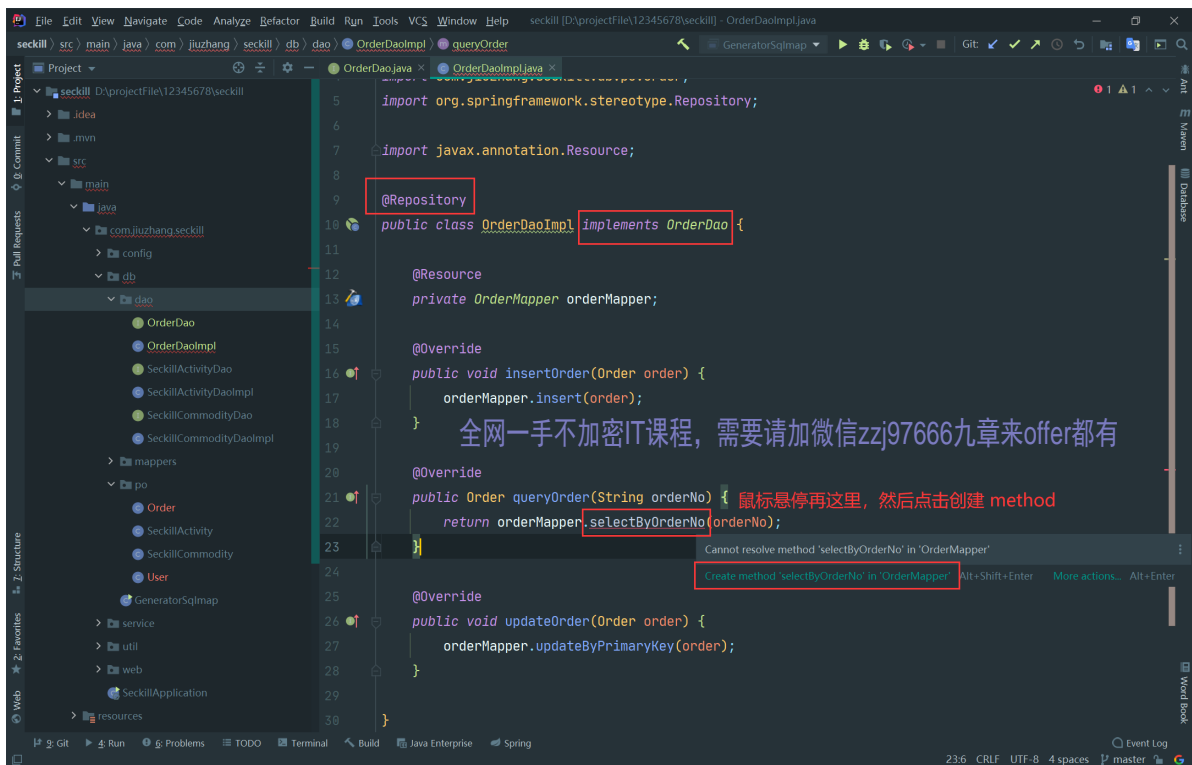


```
void insertOrder(Order order);

Order queryOrder(String orderNo);

void updateOrder(Order order);
```

6. 创建 OrderDao 接口的实现类 OrderDaoImpl & 实现 OrderDao 接口方法 & 添加 @Repository & 注入 OrderMapper 对象



```
@Repository
public class OrderDaoImpl implements OrderDao {

    @Resource
    private OrderMapper orderMapper;

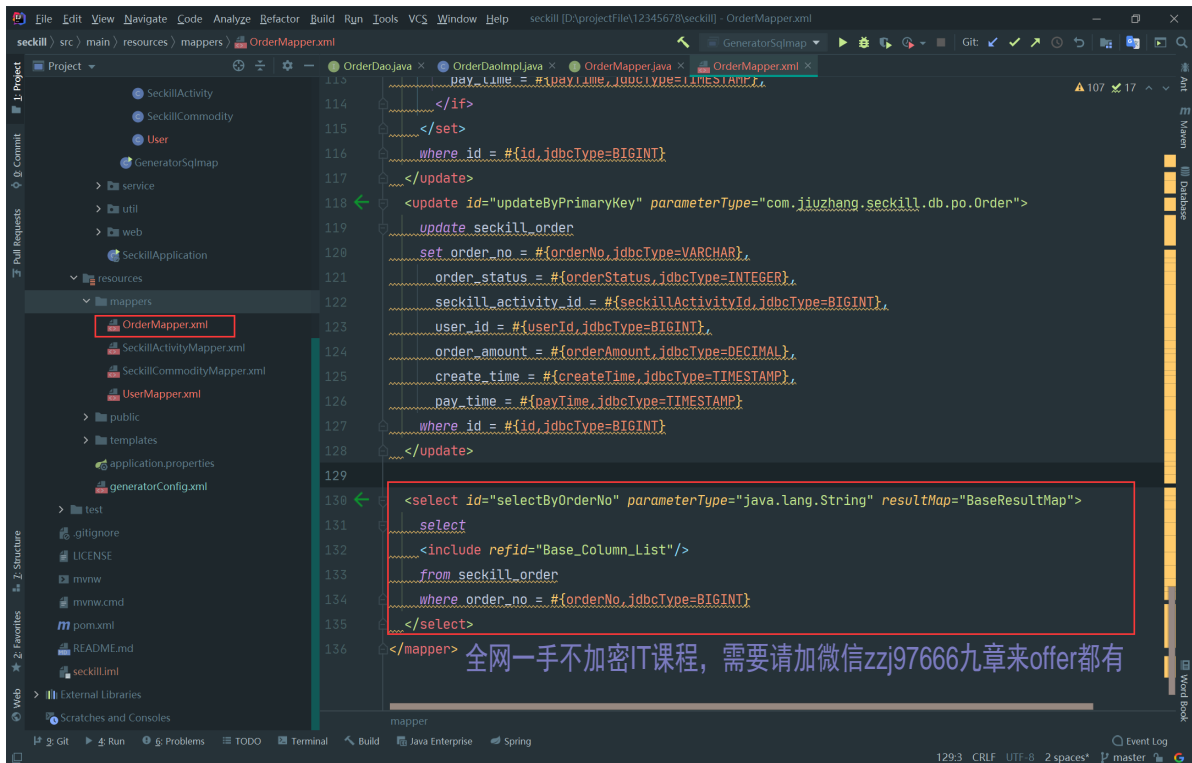
    @Override
    public void insertOrder(Order order) {
        orderMapper.insert(order);
    }

    @Override
    public Order queryOrder(String orderNo) {
        return orderMapper.selectByOrderNo(orderNo);
    }

    @Override
    public void updateOrder(Order order) {
        orderMapper.updateByPrimaryKey(order);
    }
}
```

7. 方法创建成功后，打开OrderMapper.xml文件，添加selectByOrderNo

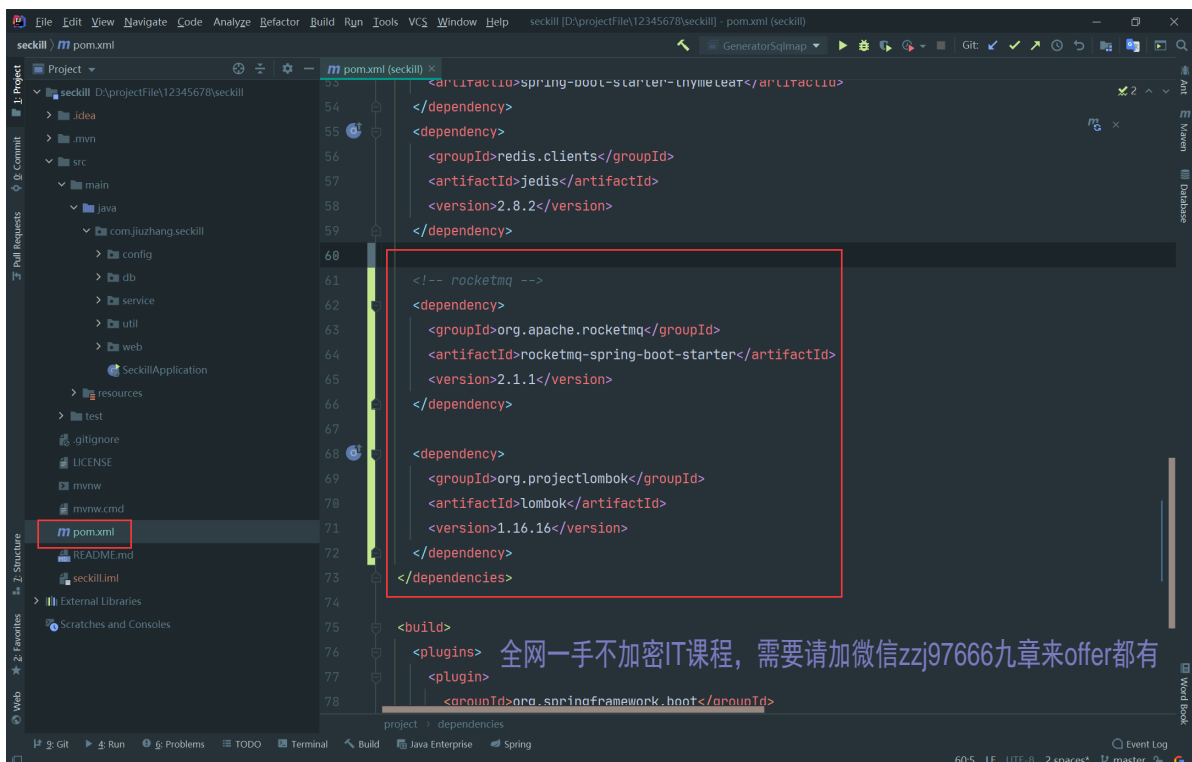




```
<select id="selectByOrderNo" parameterType="java.lang.String"
resultMap="BaseResultMap">
  select
  <include refid="Base_Column_List"/>
  from seckill_order
  where order_no = #{orderNo, jdbcType=BIGINT}
</select>
```

8. 下面我们来引入 RocketMQ 的依赖包 & 同时引入一下 Lombok 的依赖

注意：添加完依赖后记得刷新一下 maven 依赖



```

<!-- rocketmq -->
<dependency>
    <groupId>org.apache.rocketmq</groupId>
    <artifactId>rocketmq-spring-boot-starter</artifactId>
    <version>2.1.1</version>
</dependency>

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.16.16</version>
</dependency>

```

9. 配置文件配置 RocketMQ

```

24 #spring.redis.password=
25 # 连接池最大连接数（使用负值表示没有限制）
26 spring.redis.pool.max-active=200
27 # 连接池最大阻塞等待时间（使用负值表示没有限制）
28 spring.redis.pool.max-wait=-1
29 # 连接池中的最大空闲连接
30 spring.redis.pool.max-idle=8
31 # 连接池中的最小空闲连接
32 spring.redis.pool.min-idle=0
33 # 连接超时时间（毫秒）
34 spring.redis.timeout=0
35 #spring-session 使用
36 spring.session.store-type=none
37
38 ### RocketMQ ###
39 ## Windows 启动 RocketMQ 命令
40 ## 1、CMD 输入：mqnamesrv （请勿关闭窗口）
41 ## 2、另开CMD窗口 输入：mqbroker -n 127.0.0.1:9876 autoCreateTopicEnable=true （请勿关闭窗口）
42
43 rocketmq.name-server=localhost:9876
44 rocketmq.producer.group=my-group

```

全网一手不加密IT课程，需要请加微信zzj97666九章来offer都有

```

### RocketMQ ###
## windows 启动 RocketMQ 命令
## 1、CMD 输入：mqnamesrv （请勿关闭窗口）
## 2、另开CMD窗口 输入：mqbroker -n 127.0.0.1:9876 autoCreateTopicEnable=true （请勿关闭窗口）

rocketmq.name-server=localhost:9876
rocketmq.producer.group=my-group

```

10. 创建 RocketMQService 类




```

@Autowired
private RocketMQTemplate rocketMQTemplate;

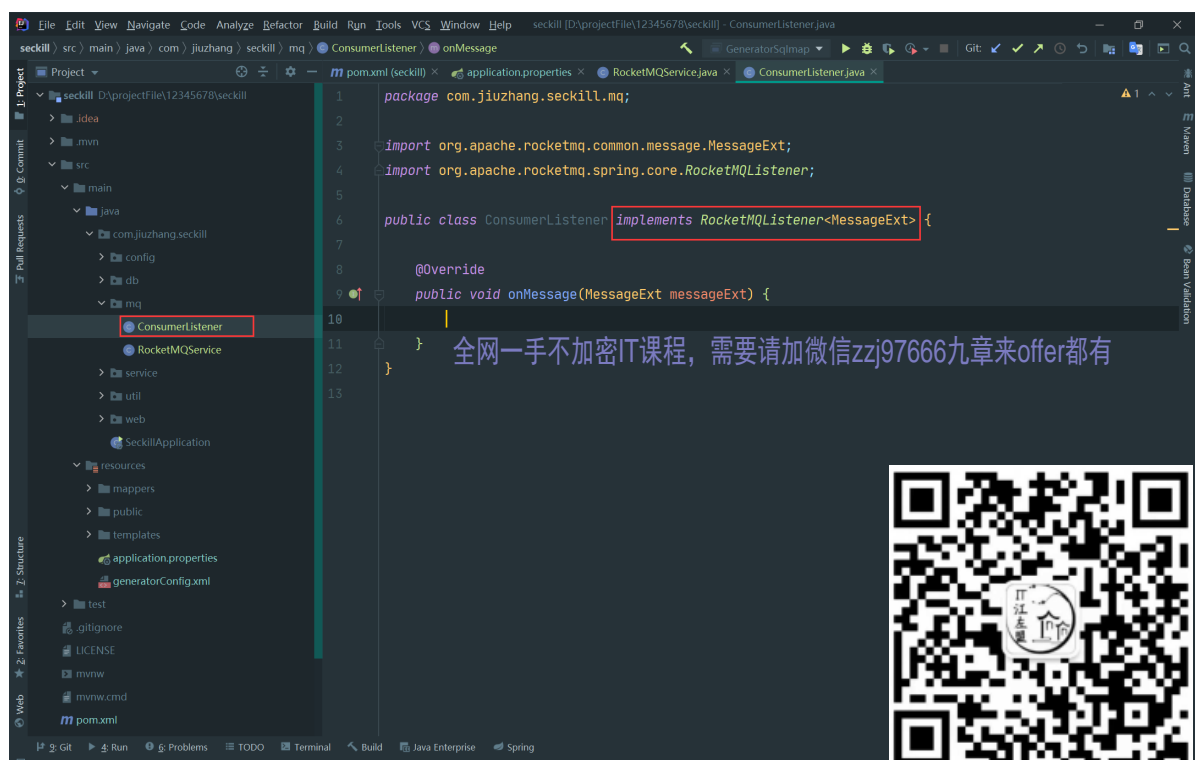
/**
 * 发送消息
 * @param topic
 * @param body
 * @throws Exception
 */
public void sendMessage(String topic,String body) throws Exception{
    Message message = new Message(topic,body.getBytes());
    rocketMQTemplate.getProducer().send(message);
}

/**
 * 发送延时消息
 *
 * @param topic
 * @param body
 * @param delayTimeLevel
 * @throws Exception
 */
public void sendDelayMessage(String topic, String body, int delayTimeLevel)
throws Exception {
    Message message = new Message(topic, body.getBytes());
    message.setDelayTimeLevel(delayTimeLevel);
    rocketMQTemplate.getProducer().send(message);
}

```

全网一手不加密IT课程，需要请加微信zzj97666九章来offer都有

12. 创建 ConsumerListener 消费者类 & 实现 RocketMQListener 接口



```

package com.jiuzhang.seckill.mq;


import org.apache.rocketmq.common.message.MessageExt;
import org.apache.rocketmq.spring.core.RocketMQListener;

public class ConsumerListener implements RocketMQListener<MessageExt> {

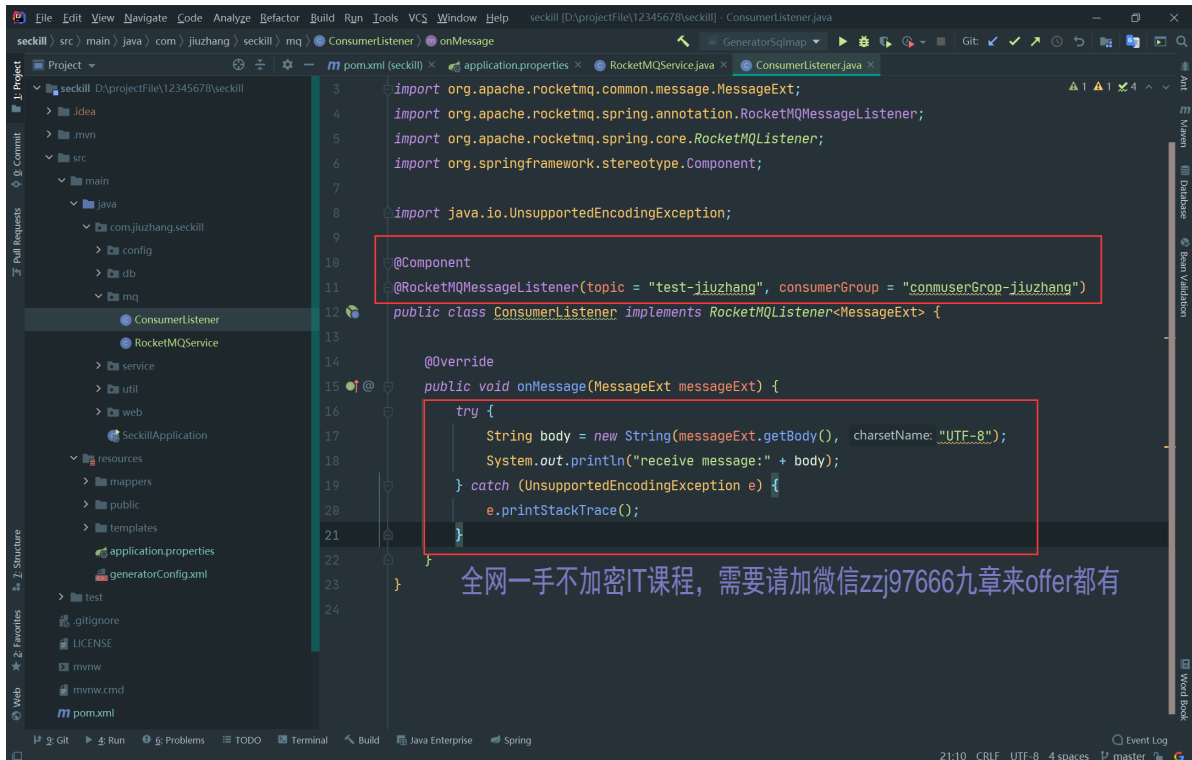
    @Override
    public void onMessage(MessageExt messageExt) {
    }
}

```

全网一手不加密IT课程，需要请加微信zzj97666九章来offer都有



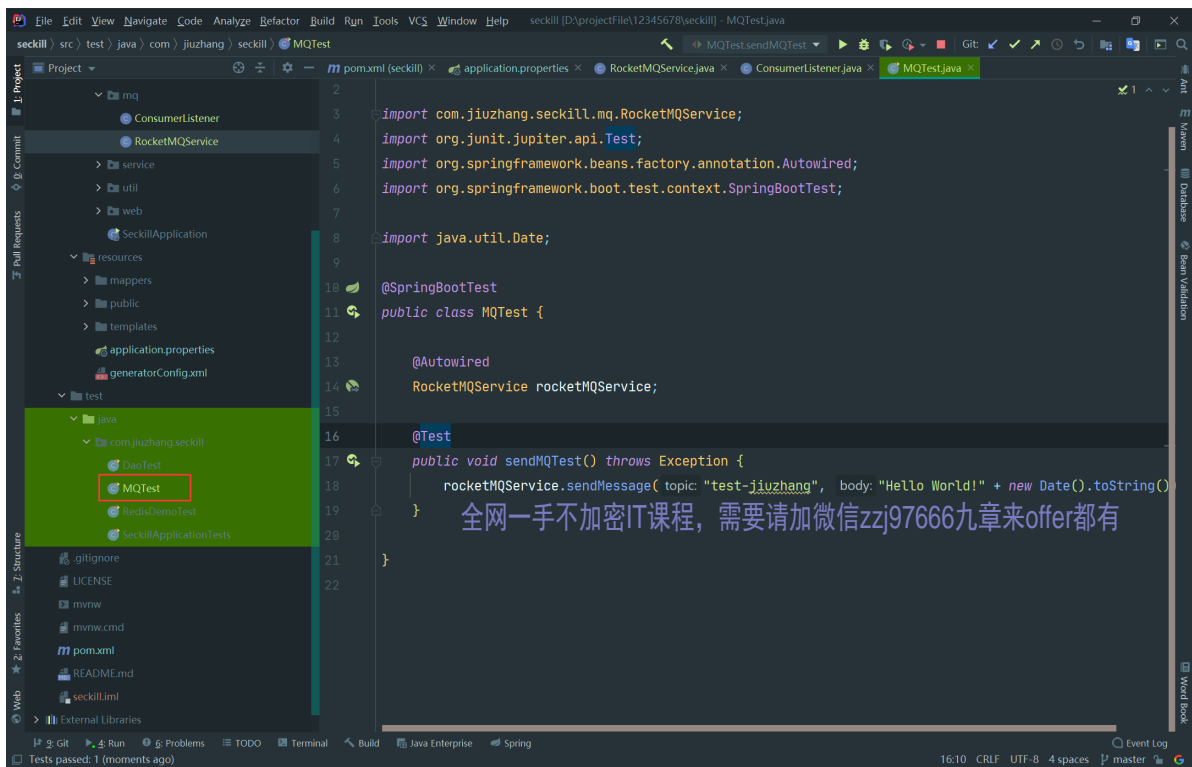
13. 添加Component, RocketMQMessageListener 注解 & 编写 onMessage 方法



```
@Component
@RocketMQMessageListener(topic = "test-jiuzhang", consumerGroup = "commuserGrop-jiuzhang")
public class ConsumerListener implements RocketMQListener<MessageExt> {
    @Override
    public void onMessage(MessageExt messageExt) {
        try {
            String body = new String(messageExt.getBody(), "UTF-8");
            System.out.println("receive message:" + body);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }
}
```

14. 编写 test 类, 测试 MQ 发送消息和接收消息





```
@SpringBootTest
public class MQTest {

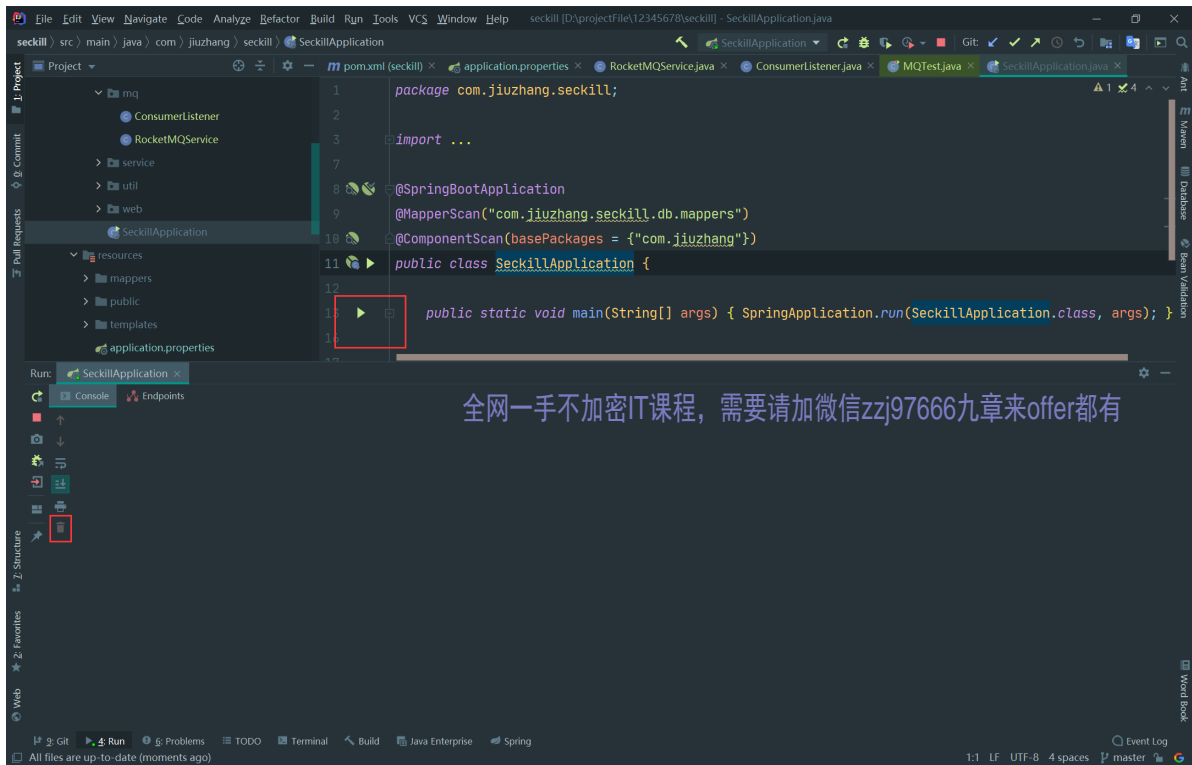
    @Autowired
    RocketMQService rocketMQService;

    @Test
    public void sendMQTest() throws Exception {
        rocketMQService.sendMessage("test-jiuzhang", "Hello World!" + new
Date().toString());
    }
}
```

15. 启动项目，清空控制台的日志输出

注意：需要先启动 Redis 和 RocketMQ





16. 运行 test 方法，检查控制台输出日志

