消息队列 RabbitMQ 版

开发指南

腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

开发指南
延时消息

最近更新时间：2024-01-22 15:28:41

本文主要介绍消息队列 TDMQ RabbitMQ 版中延迟消息的概念、使用场景和使用方式。

名词解释

延时消息：消息在发送至服务端后，实际业务并不希望消费端马上收到这条消息，而是推迟一段时间后再被消费，这类消息统称为延时消息。

使用场景

- 场景 1：对于消息的生产消费时间有要求的场景。例如在电商系统中，若用户下单后 30 分钟不支付，自动取消订单。
- 场景 2：通过消息触发延时任务的场景。用户登录 App 浏览特定商品 20 分钟后还没下单，自动推送商品评测信息的消息并发放商品相关优惠券。

实现方式

方式一：通过设置消息的过期时间和死信队列实现延时消息

原理概述

- 发送端通过设置消息过期时间，触发过期未消费的消息被投递到死信交换机下的死信队列。
- 消费端通过消费死信队列中的消息，实现延时消息的消费。

使用示例

代码仅示例使用，交换机类型、routingKey 等参数请结合实际场景替换为业务预期的值。

- 发送端代码示例如下，消费端订阅死信队列即可。

```
// 声明死信交换机
channel.exchangeDeclare("${dlxExchangeName}", "direct", true);
// 声明用于发送延时消息的交换机
channel.exchangeDeclare("${delayExchangeName}", "direct", true);
// 声明用于发送延时消息的队列，并指定其死信交换机
Map<String, Object> args = new HashMap<>();
args.put("x-dead-letter-exchange", "${dlxExchangeName}");
channel.queueDeclare("${delayQueueName}", true, false, false, args);
channel.queueBind("${delayQueueName}", "${delayExchangeName}", "");
```

```
// 声明死信队列
channel.queueDeclare("${delayQueueName}", true, false, false, null);
channel.queueBind("${dlxQueueName}", "${dlxExchangeName}", "");

// 发送延时消息
int delayInSeconds = 10; // 消息延时 10s
AMQP.BasicProperties props = new
AMQP.BasicProperties.Builder().expiration(String.valueOf(delayInSecond
s * 1000));
channel.basicPublish("${delayExchangeName}", "", props.build(),
"delayed payload".getBytes());
```

- 参数说明如下

参数	说明
${dlxExchangeName}	用于延迟消息的死信交换机名称，请替换为可在控制台 Exchange 列表查询到的名称。
${delayExchangeName}	实际发送延时消息的交换机名称，请替换为可在控制台 Exchange 列表查询到的名称。
x-dead-letter-exchange	队列参数 key，用于设置其对应的死信交换机。
${dlxQueueName}	用于延迟消息的死信队列名称
${delayQueueName}	实际发送延时消息的队列名称

方式二：通过内置的 rabbitmq_delayed_message_exchange 插件实现延时消息

使用限制

> **说明：**
> 更多详细信息请参考 RabbitMQ 官方插件使用限制说明 。

- 当前插件的设计不适用于大量延迟消息（未调度的消息达数十万甚至数百万条）的场景，生产环境请谨慎评估消息量级，避免非预期的长时间延迟、消息丢失等问题。
- 延时消息在每个节点上只有一个持久化副本，如果节点无法正常运行（例如由于消息堆积导致持续 OOM 后重启且无法恢复），则该节点上的延时消息无法被消费端消费。

- 延时交换机不支持设置 `mandatory` ，生产者无法通过 `basic.return` 事件感知到无法路由的消息，因此发送延时消息前请务必保证对应的交换机、队列、路由关系存在。

使用示例

TDMQ RabbitMQ 版延时消息的使用方式和 RabbitMQ 官方支持的延时插件的使用方式完全一致，方便业务进行无改造迁移。

1. 声明 Exchange 并指定 Exchange 的路由类型。

```
// 声明延时交换机
Map<String, Object> args = new HashMap<String, Object>();
args.put("x-delayed-type", "direct");
channel.exchangeDeclare("${delayedExchangeName}", "x-delayed-message",
true, false, args);
```

   - 参数说明如下：

参数	说明
x-delayed-type	Exchange 的类型，指定路由规则。取值说明如下： • direct • fanout • topic 具体说明请参见 Exchange 。
${delayedExchangeName}	Exchange 的名称，请替换为可在控制台 Exchange 列表查询到的名称。
x-delayed-message	指定 Exchange 类型，以支持投递延时消息。

2. 发送延时消息。在消息的 Header 属性中增加一个键为 x-delay，值为毫秒数的键值对，并且指定发送的目标 Exchange 为上一步已声明的 Exchange。

```
byte[] messageBodyBytes = "delayed payload".getBytes("UTF-8");
Map<String, Object> headers = new HashMap<String, Object>();
headers.put("x-delay", 4000); // 消息延时 4s
AMQP.BasicProperties.Builder props = new
AMQP.BasicProperties.Builder().headers(headers);
channel.basicPublish("${delayedExchangeName}", "", props.build(),
messageBodyBytes);
```

   当消息到达 Exchange 后，会在4000毫秒后投递到对应的 Queue。