# SpringBoot消息重试机制

### 消息重试机制幂等性

#### 如何合适选择重试机制

情况1:  消费者获取到消息后，调用第三方接口，但接口暂时无法访问，是否需要重试?      需要重试

情况2:  消费者获取到消息后，抛出数据转换异常，是否需要重试?                                      不需要重试

总结：对于情况2，如果消费者代码抛出异常是需要发布新版本才能解决的问题，那么不需要重试，重试也无济于事。应该采用日志记录+定时任务job健康检查+人工进行补偿

#### 消费者如果保证消息幂等性，不被重复消费

产生原因:网络延迟传输中，会造成进行MQ重试中，在重试过程中，可能会造成重复消费。

解决办法:

使用全局MessageID判断消费方使用同一个，解决幂等性。

#### 基于全局消息id区分消息，解决幂等性

##### 生产者:

请求头设置消息id（messageId）

|  |
| --- |
| @Component  **public** **class** FanoutProducer {  @Autowired  **private** AmqpTemplate amqpTemplate;  **public** **void** send(String queueName) {  String msg = "my\_fanout\_msg:" + System.*currentTimeMillis*();  Message message = MessageBuilder.*withBody*(msg.getBytes()).setContentType(MessageProperties.***CONTENT\_TYPE\_JSON***)  .setContentEncoding("utf-8").setMessageId(UUID.*randomUUID*() + "").build();  System.***out***.println(msg + ":" + msg);  amqpTemplate.convertAndSend(queueName, message);  }  } |

##### 消费者:

###### 核心代码

|  |
| --- |
| @Component  **public** **class** FanoutEamilConsumer {  @RabbitListener(queues = "fanout\_email\_queue")  **public** **void** process(Message message) **throws** Exception {  System.***out***  .println(Thread.*currentThread*().getName() + ",邮件消费者获取生产者消息msg:" + **new** String(message.getBody(), "UTF-8")  + ",messageId:" + message.getMessageProperties().getMessageId());  // int i = 1 / 0;  }  } |

###### application配置

|  |
| --- |
| spring:  rabbitmq:  ####连接地址  host: 127.0.0.1  ####端口号  port: 5672  ####账号  username: guest  ####密码  password: guest  ### 地址  virtual-host: /admin\_host  listener:  simple:  retry:  ####开启消费者重试  enabled: **true**  ####最大重试次数  max-attempts: 5  ####重试间隔次数  initial-interval: 3000    server:  port: 8081 |

#### RabbitMQ消费者重试调用接口

|  |
| --- |
| //邮件队列  @Component  **public** **class** FanoutEamilConsumer {  @RabbitListener(queues = "fanout\_email\_queue")  **public** **void** process(String msg) **throws** Exception {  System.***out***.println("邮件消费者获取生产者消息msg:" + msg);  JSONObject jsonObject = JSONObject.*parseObject*(msg);  // 获取email参数  String email = jsonObject.getString("email");  // 请求地址  String emailUrl = "http://127.0.0.1:8083/sendEmail?email=" + email;  JSONObject result = HttpClientUtils.*httpGet*(emailUrl);  **if** (result == **null**) {  // 因为网络原因,造成无法访问,继续重试  **throw** **new** Exception("调用接口失败!");  }  System.***out***.println("执行结束....");  }  } |

|  |
| --- |
| @RabbitListener(queues = "fanout\_email\_queue")  **public** **void** process(Message message) **throws** Exception {  // 获取消息Id  String messageId = message.getMessageProperties().getMessageId();  String msg = **new** String(message.getBody(), "UTF-8");  System.***out***.println("邮件消费者获取生产者消息" + "messageId:" + messageId + ",消息内容:" + msg);  JSONObject jsonObject = JSONObject.*parseObject*(msg);  // 获取email参数  String email = jsonObject.getString("email");  // 请求地址  String emailUrl = "http://127.0.0.1:8083/sendEmail?email=" + email;  JSONObject result = HttpClientUtils.*httpGet*(emailUrl);  **if** (result == **null**) {  // 因为网络原因,造成无法访问,继续重试  **throw** **new** Exception("调用接口失败!");  }  System.***out***.println("执行结束....");  } |

# RabbitMQ签收模式

|  |
| --- |
| //邮件队列  @Component  **public** **class** FanoutEamilConsumer {  @RabbitListener(queues = "fanout\_email\_queue")  **public** **void** process(Message message, @Headers Map<String, Object> headers, Channel channel) **throws** Exception {  System.***out***  .println(Thread.*currentThread*().getName() + ",邮件消费者获取生产者消息msg:" + **new** String(message.getBody(), "UTF-8")  + ",messageId:" + message.getMessageProperties().getMessageId());  // 手动ack  Long deliveryTag = (Long) headers.get(AmqpHeaders.***DELIVERY\_TAG***);  // 手动签收  channel.basicAck(deliveryTag, **false**);  }  } |

开启手动应答

|  |
| --- |
| spring:  rabbitmq:  ####连接地址  host: 127.0.0.1  ####端口号  port: 5672  ####账号  username: guest  ####密码  password: guest  ### 地址  virtual-host: /admin\_host  listener:  simple:  retry:  ####开启消费者异常重试  enabled: **true**  ####最大重试次数  max-attempts: 5  ####重试间隔次数  initial-interval: 2000  ####开启手动ack  acknowledge-mode: manual |