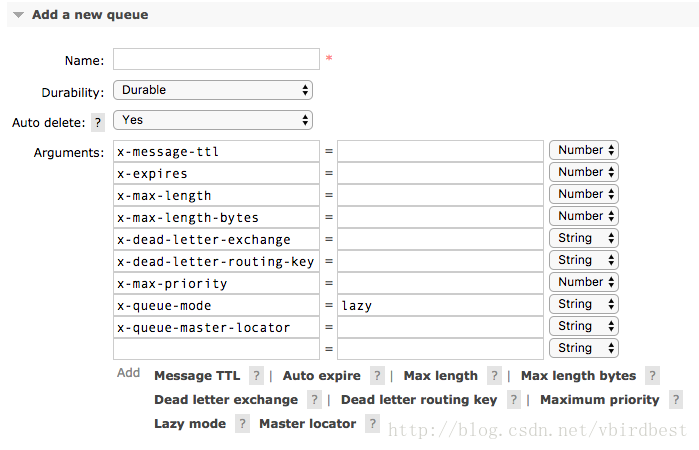
**RabbitMq各个参数含义**

论队列声明的各个参数



1. queueDeclare(String queue,
2. boolean durable,
3. boolean exclusive,
4. Map<String, Object> arguments);

* 1
* 2
* 3
* 4
* queue: 队列名称
* durable： 是否持久化, 队列的声明默认是存放到内存中的，如果rabbitmq重启会丢失，如果想重启之后还存在就要使队列持久化，保存到Erlang自带的Mnesia数据库中，当rabbitmq重启之后会读取该数据库
* exclusive：是否排外的，有两个作用，一：当连接关闭时connection.close()该队列是否会自动删除；二：该队列是否是私有的private，如果不是排外的，可以使用两个消费者都访问同一个队列，没有任何问题，如果是排外的，会对当前队列加锁，其他通道channel是不能访问的，如果强制访问会报异常：com.rabbitmq.client.ShutdownSignalException: channel error; protocol method: #method<channel.close>(reply-code=405, reply-text=RESOURCE\_LOCKED - cannot obtain exclusive access to locked queue 'queue\_name' in vhost '/', class-id=50, method-id=20)一般等于true的话用于一个队列只能有一个消费者来消费的场景
* autoDelete：是否自动删除，当最后一个消费者断开连接之后队列是否自动被删除，可以通过RabbitMQ Management，查看某个队列的消费者数量，当consumers = 0时队列就会自动删除
* arguments：   
  队列中的消息什么时候会自动被删除？
  + Message TTL(x-message-ttl)：设置队列中的所有消息的生存周期(统一为整个队列的所有消息设置生命周期), 也可以在发布消息的时候单独为某个消息指定剩余生存时间,单位毫秒, 类似于redis中的ttl，生存时间到了，消息会被从队里中删除，注意是消息被删除，而不是队列被删除， 特性Features=TTL, 单独为某条消息设置过期时间AMQP.BasicProperties.Builder properties = new AMQP.BasicProperties().builder().expiration(“6000”);   
    channel.basicPublish(EXCHANGE\_NAME, “”, properties.build(), message.getBytes(“UTF-8”));
  + Auto Expire(x-expires): 当队列在指定的时间没有被访问(consume, basicGet, queueDeclare…)就会被删除,Features=Exp
  + Max Length(x-max-length): 限定队列的消息的最大值长度，超过指定长度将会把最早的几条删除掉， 类似于mongodb中的固定集合，例如保存最新的100条消息, Feature=Lim
  + Max Length Bytes(x-max-length-bytes): 限定队列最大占用的空间大小， 一般受限于内存、磁盘的大小, Features=Lim B
  + Dead letter exchange(x-dead-letter-exchange)： 当队列消息长度大于最大长度、或者过期的等，将从队列中删除的消息推送到指定的交换机中去而不是丢弃掉,Features=DLX
  + Dead letter routing key(x-dead-letter-routing-key)：将删除的消息推送到指定交换机的指定路由键的队列中去, Feature=DLK
  + Maximum priority(x-max-priority)：优先级队列，声明队列时先定义最大优先级值(定义最大值一般不要太大)，在发布消息的时候指定该消息的优先级， 优先级更高（数值更大的）的消息先被消费,
  + Lazy mode(x-queue-mode=lazy)： Lazy Queues: 先将消息保存到磁盘上，不放在内存中，当消费者开始消费的时候才加载到内存中
  + Master locator(x-queue-master-locator)

**注意**

关于队列的声明，如果使用同一套参数进行声明了，就不能再使用其他参数来声明，要么删除该队列重新删除，可以使用命令行删除也可以在RabbitMQ Management上删除，要么给队列重新起一个名字。

**队列持久化**

重启RabbitMQ服务器(可以通过rabbitmqctl stop\_app关闭服务器，rabbitmqctl start\_app重启服务器)，可以登录RabbitMQ Management—> Queues中可以看到之前声明的队列还存在

1. boolean durable = true;
2. channel.queueDeclare(QUEUE\_NAME, durable, false, false, arguments);

* 1
* 2

**消息持久化**

设置消息持久化必须先设置队列持久化，要不然队列不持久化，消息持久化，队列都不存在了，消息存在还有什么意义。消息持久化需要将交换机持久化、队列持久化、消息持久化，才能最终达到持久化的目的

方式一：设置deliveryMode=2

1. channel.exchangeDeclare(EXCHANGE\_NAME, BuiltinExchangeType.DIRECT, true);
2. channel.queueDeclare(QUEUE\_NAME, true, false, false, arguments);
3. channel.queueBind(QUEUE\_NAME, EXCHANGE\_NAME, "");
5. String message = "Hello RabbitMQ: ";
6. *// 设置消息持久化*
7. AMQP.BasicProperties.Builder properties = new AMQP.BasicProperties().builder();
8. properties.deliveryMode(2); *// 设置消息是否持久化，1： 非持久化 2：持久化*
10. channel.basicPublish(EXCHANGE\_NAME, "", properties.build(), message.getBytes("UTF-8"));

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

方式二：设置BasicProperties为MessageProperties.PERSISTENT\_TEXT\_PLAIN

1. channel.queueDeclare(QUEUE\_NAME, true, false, false, arguments);
2. channel.queueBind(QUEUE\_NAME, EXCHANGE\_NAME, "");
4. String message = "Hello RabbitMQ: ";
5. channel.basicPublish(EXCHANGE\_NAME, "", MessageProperties.PERSISTENT\_TEXT\_PLAIN, message.getBytes("UTF-8"));

* 1
* 2
* 3
* 4
* 5

**Message TTL消息剩余生存时间**

统一设置队列中的所有消息的过期时间，例如设置10秒，10秒后这个队列的消息清零

方式一：为该队列的所有消息统一设置相同的声明周期

1. Map<String, Object> arguments = new HashMap<String, Object>();
2. arguments.put("x-message-ttl", 10000);
4. *// 声明队列时指定队列中的消息过期时间*
5. channel.queueDeclare(QUEUE\_NAME, false, false, false, arguments);
6. channel.basicPublish(EXCHANGE\_NAME, "", null, message.getBytes("UTF-8"));

* 1
* 2
* 3
* 4
* 5
* 6

方式二：单独为某条消息单独设置时间

// expiration: 设置单条消息的过期时间

1. channel.queueDeclare(QUEUE\_NAME, false, false, false, null);
2. channel.basicPublish(EXCHANGE\_NAME, "", null, message.getBytes("UTF-8"));
4. for(int i = 1; i <= 5; i++) {
5. AMQP.BasicProperties.Builder properties = new AMQP.BasicProperties()
6. .builder().expiration( i \* 1000 + "");
8. channel.basicPublish(EXCHANGE\_NAME, "", properties.build(), (message + i).getBytes("UTF-8"));
9. }

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9

**Auto Expire自动过期**

x-expires用于当多长时间没有消费者访问该队列的时候，该队列会自动删除，可以设置一个延迟时间，如仅启动一个生产者，10秒之后该队列会删除，或者启动一个生产者，再启动一个消费者，消费者运行结束后10秒，队列也会被删除

1. Map<String, Object> arguments = new HashMap<String, Object>();
2. arguments.put("x-expires", 10000);
4. channel.queueDeclare(QUEUE\_NAME, false, false, false, arguments);
5. channel.basicPublish(EXCHANGE\_NAME, "", null, message.getBytes("UTF-8"));

* 1
* 2
* 3
* 4
* 5

**Max Length最大长度**

x-max-length:用于指定队列的长度，如果不指定，可以认为是无限长，例如指定队列的长度是4，当超过4条消息，前面的消息将被删除，给后面的消息腾位

1. Map<String, Object> arguments = new HashMap<String, Object>();
2. arguments.put("x-max-length", 4);
4. channel.queueDeclare(QUEUE\_NAME, false, false, false, arguments);
5. for(int i = 1; i <= 5; i++) {
6. channel.basicPublish(EXCHANGE\_NAME, "", MessageProperties.PERSISTENT\_TEXT\_PLAIN, (message + i).getBytes("UTF-8"));
7. }

* 1
* 2
* 3
* 4
* 5
* 6
* 7

**Max Length Bytes代码片段**

x-max-length-bytes: 用于指定队列存储消息的占用空间大小，当达到最大值是会删除之前的数据腾出空间

1. Map<String, Object> arguments = new HashMap<String, Object>();
2. rguments.put("x-max-length-bytes", 1024);
4. channel.queueDeclare(QUEUE\_NAME, false, false, false, arguments);
5. channel.basicPublish(EXCHANGE\_NAME, "", null, message.getBytes("UTF-8"));

* 1
* 2
* 3
* 4
* 5

**Maximum priority最大优先级**

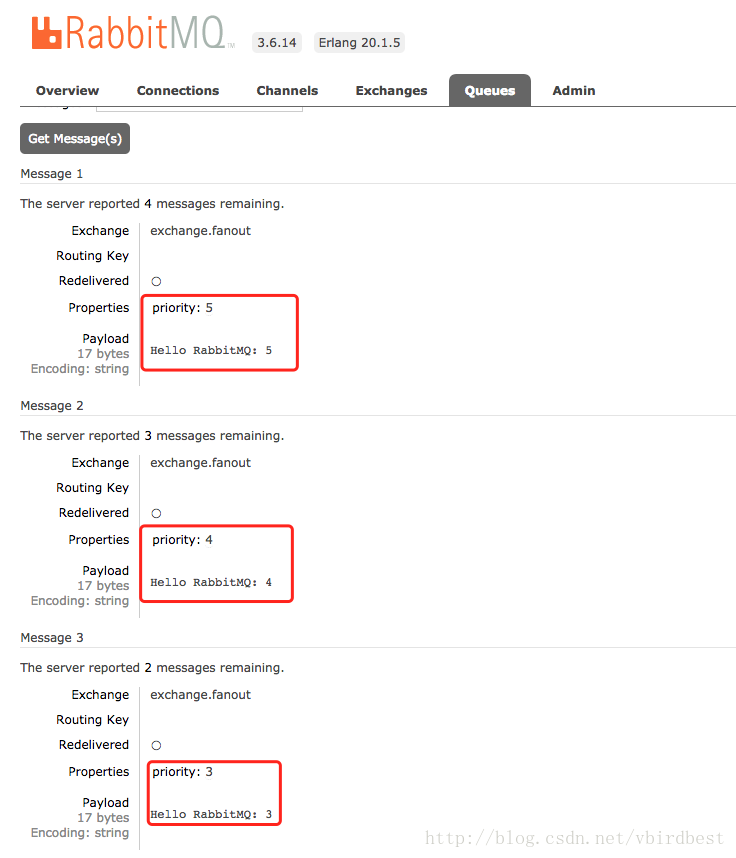
x-max-priority: 设置消息的优先级，优先级值越大，越被提前消费。

正常情况下不适用优先级   
Hello RabbitMQ: 1   
Hello RabbitMQ: 2   
Hello RabbitMQ: 3   
Hello RabbitMQ: 4   
Hello RabbitMQ: 5

使用优先级顺序正好相反   
Hello RabbitMQ: 5   
Hello RabbitMQ: 4   
Hello RabbitMQ: 3   
Hello RabbitMQ: 2   
Hello RabbitMQ: 1

1. Map<String, Object> arguments = new HashMap<String, Object>();
2. arguments.put("x-max-priority", 10);
4. channel.queueDeclare(QUEUE\_NAME, false, false, false, arguments);
5. for(int i = 1; i <= 5; i++) {
6. AMQP.BasicProperties.Builder properties = new AMQP.BasicProperties()
7. .builder().priority(i);
8. channel.basicPublish(EXCHANGE\_NAME, "", properties.build(), (message + i).getBytes("UTF-8"));
9. }

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9



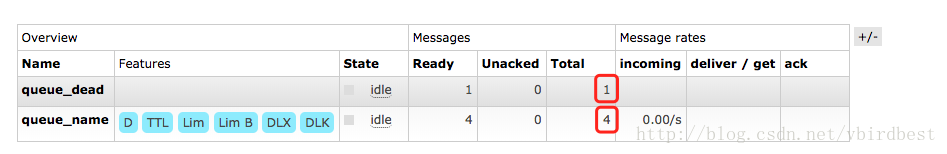
**Dead letter exchange(死亡交换机) 和 Dead letter routing key(死亡路由键)**

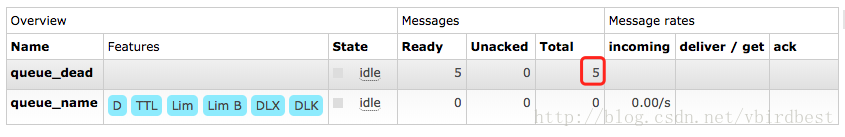
当队列中的消息过期，或者达到最大长度而被删除，或者达到最大空间时而被删除时，可以将这些被删除的信息推送到其他交换机中，让其他消费者订阅这些被删除的消息，处理这些消息

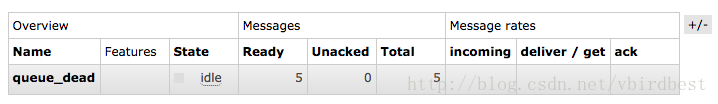
1. public void testBasicPublish() throws IOException, TimeoutException, InterruptedException {
2. ConnectionFactory factory = new ConnectionFactory();
3. factory.setHost("127.0.0.1");
4. factory.setPort(AMQP.PROTOCOL.PORT);
5. factory.setUsername("mengday");
6. factory.setPassword("mengday");
8. Connection connection = factory.newConnection();
9. Channel channel = connection.createChannel();
11. *// 声明一个接收被删除的消息的交换机和队列*
12. String EXCHANGE\_DEAD\_NAME = "exchange.dead";
13. String QUEUE\_DEAD\_NAME = "queue\_dead";
14. channel.exchangeDeclare(EXCHANGE\_DEAD\_NAME, BuiltinExchangeType.DIRECT);
15. channel.queueDeclare(QUEUE\_DEAD\_NAME, false, false, false, null);
16. channel.queueBind(QUEUE\_DEAD\_NAME, EXCHANGE\_DEAD\_NAME, "routingkey.dead");
18. String EXCHANGE\_NAME = "exchange.fanout";
19. String QUEUE\_NAME = "queue\_name";
20. channel.exchangeDeclare(EXCHANGE\_NAME, BuiltinExchangeType.FANOUT);
22. Map<String, Object> arguments = new HashMap<String, Object>();
23. arguments.put("x-message-ttl", 15000);
24. arguments.put("x-max-length", 4);
25. arguments.put("x-max-length-bytes", 1024);
26. arguments.put("x-expires", 30000);
28. arguments.put("x-dead-letter-exchange", "exchange.dead");
29. arguments.put("x-dead-letter-routing-key", "routingkey.dead");
30. channel.queueDeclare(QUEUE\_NAME, true, false, false, arguments);
31. channel.queueBind(QUEUE\_NAME, EXCHANGE\_NAME, "");
33. String message = "Hello RabbitMQ: ";
34. for(int i = 1; i <= 5; i++) {
35. channel.basicPublish(EXCHANGE\_NAME, "", null, (message + i).getBytes("UTF-8"));
36. }

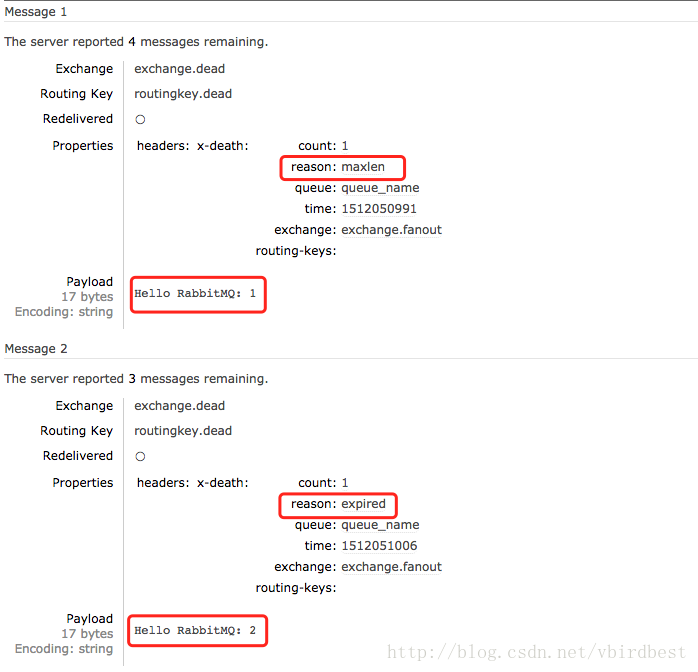
39. channel.close();
40. connection.close();
41. }

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41

刚开始由于队列长度是4，总共发送5条消息，所以最早进入队列的消息1将被删除掉，被推送到“死亡队列”中，所以看到普通队列的消息为4条，死亡队列的消息为1条，是消息1   


随着时间的流逝，普通队列中的消息都该过期了，所以消息2、3、4、5都被推送到死亡队列，所以死亡队列消息是5条，普通队列的消息条数为0   


再随着时间的流逝，普通队列过了指定时间没有被消费者访问，这个队列自动被删除了，所以看不到普通队列了，只有死亡队列   


查看死亡队列的消息可以得知，消息一死亡的原因是maxlen达到了最大长度，消息2、3、4、5都是因为生存时间到了导致死亡的   
   


**一个比较杂的综合示例**

关于消费者就不用代码来获取消息了，直接在RabbitMQ Management点击某个队列的名字，然后Get Message(s) 即可获取

该示例使用很多参数配置，可能实际使用不会像这样用，因为这样好像不太配套。

1. public class Producer {
2. @Test
3. public void testBasicPublish() throws IOException, TimeoutException, InterruptedException {
4. ConnectionFactory factory = new ConnectionFactory();
5. factory.setHost("127.0.0.1");
6. factory.setPort(AMQP.PROTOCOL.PORT);
7. factory.setUsername("mengday");
8. factory.setPassword("mengday");
10. Connection connection = factory.newConnection();
11. Channel channel = connection.createChannel();
13. *// 声明一个接收被删除的消息的交换机和队列*
14. String EXCHANGE\_DEAD\_NAME = "exchange.dead";
15. String QUEUE\_DEAD\_NAME = "queue\_dead";
16. channel.exchangeDeclare(EXCHANGE\_DEAD\_NAME, BuiltinExchangeType.DIRECT);
17. channel.queueDeclare(QUEUE\_DEAD\_NAME, false, false, false, null);
18. channel.queueBind(QUEUE\_DEAD\_NAME, EXCHANGE\_DEAD\_NAME, "routingkey.dead");
20. String EXCHANGE\_NAME = "exchange.fanout";
21. String QUEUE\_NAME = "queue\_name";
22. channel.exchangeDeclare(EXCHANGE\_NAME, BuiltinExchangeType.FANOUT);

25. Map<String, Object> arguments = new HashMap<String, Object>();
26. *// 统一设置队列中的所有消息的过期时间*
27. arguments.put("x-message-ttl", 30000);
28. *// 设置超过多少毫秒没有消费者来访问队列，就删除队列的时间*
29. arguments.put("x-expires", 20000);
30. *// 设置队列的最新的N条消息，如果超过N条，前面的消息将从队列中移除掉*
31. arguments.put("x-max-length", 4);
32. *// 设置队列的内容的最大空间，超过该阈值就删除之前的消息*
33. arguments.put("x-max-length-bytes", 1024);
34. *// 将删除的消息推送到指定的交换机，一般x-dead-letter-exchange和x-dead-letter-routing-key需要同时设置*
35. arguments.put("x-dead-letter-exchange", "exchange.dead");
36. *// 将删除的消息推送到指定的交换机对应的路由键*
37. arguments.put("x-dead-letter-routing-key", "routingkey.dead");
38. *// 设置消息的优先级，优先级大的优先被消费*
39. arguments.put("x-max-priority", 10);
40. channel.queueDeclare(QUEUE\_NAME, false, false, false, arguments);
41. channel.queueBind(QUEUE\_NAME, EXCHANGE\_NAME, "");

44. String message = "Hello RabbitMQ: ";
45. for(int i = 1; i <= 5; i++) {
46. *// expiration: 设置单条消息的过期时间*
47. AMQP.BasicProperties.Builder properties = new AMQP.BasicProperties().builder().priority(i).expiration( i \* 1000 + "");
48. channel.basicPublish(EXCHANGE\_NAME, "", properties.build(), (message + i).getBytes("UTF-8"));
49. }

52. channel.close();
53. connection.close();
54. }
55. }

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56

**运行效果**

