# 备份交换器

备份交换器，英文名称为Altemate Exchange ，简称AE，或者更直白地称之为"备胎交换器"。

生产者在发送消息的时候如果不设置mandatory 参数， 那么消息在未被路由的情况下将会丢失:

如果设置了mandatory 参数，那么需要添加ReturnListener 的编程逻辑，生产者的代码将

变得复杂。如果既不想复杂化生产者的编程逻辑，又不想消息丢失，那么可以使用备份交换器，

这样可以将未被路由的消息存储在RabbitMQ 中，再在需要的时候去处理这些消息。可以通过在声明交换器(调用channel.exchangeDeclare 方法)的时候添加alternate-exchange 参数来实现

# 队列

## 设置消息的TTL

1. 设置队列的属性，队列中的所有消息都有相同的过期时间

方法一：在channel.queueDeclare()方法设置参数：x-message –ttl

方法二：使用Policy方式设置

rabbitmqctl set\_policy TTL “.\*” ‘{"message-ttl":60000}’ –apply-to <queueName>

1. 对消息本身进行单独设置，每条消息的TTL可以不同。

消息在队列中的生存时间一旦超过设置的过期时间，就会变成“死信”。

# 设置队列的TTL

通过channel . queueDeclare 方法中的x-expires 参数可以控制队列被自动删除前处，于未使用状态的时间。未使用的意思是队列上没有任何的消费者，队列也没有被重新声明，并且在过期时间段内也未调用过Basic.Get命令。

## 死信队列

DLX，全称为Dead-Letter-Exchange，可以称为死信交换器。当消息在一个队列中编程死信之后，它能够被重新发送到另一个交换器中，这个交换器就是DLX，绑定DLX的队列就称为死信队列。

死信队列一般由于以下几种情况：

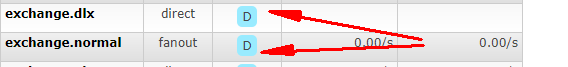
1. 消息被拒绝（Basic.Reject/Basic.Nack），并设置requeue参数为false。
2. 消息过期
3. 队列达到最大长度

x-dead-letter-exchange：

x-dead-letter-routing-key：

|  |
| --- |
| public class RabbitDlxProducer {  private static final String *IP\_ADDR* = "192.168.253.120" ;  private static final int *PORT*= 5672 ;  public static void main(String[] args) throws Exception  {  ConnectionFactory factory = new ConnectionFactory() ;  factory.setHost(*IP\_ADDR*);  factory.setPort(*PORT*);  factory.setUsername("root");  factory.setPassword("root");  Connection connection = factory.newConnection() ;  Channel channel = connection.createChannel() ;  System.*out*.println(channel.isOpen());  //创建一个死信交换器  channel .exchangeDeclare("exchange.dlx" ,"direct",true);  channel.queueDeclare("queue.dlx",true,false,false,null);//死信队列  channel.queueBind("queue.dlx","exchange.dlx","routingkey") ;//板顶  channel . exchangeDeclare( "exchange.normal" ,"fanout" , true);  Map<String , Object> argss = new HashMap<String, Object>( );  argss.put("x-message- ttl",10000);//队列信息过期时间  argss.put( "x-dead-letter-exchange","exchange.dlx");//配置死信队列  argss.put( "x-dead-letter-routing-key" ,"routingkey");//配置死信路由  channel.queueDeclare("queue.norma1",true,false,false ,argss);  channel.queueBind("queue.norma1" , "exchange.normal" , "");  //发送消息  channel . basicPublish( "exchange.normal" ," rk " ,MessageProperties.*PERSISTENT\_TEXT\_PLAIN*, "dlx " .getBytes()) ;  } |

Exchange属性：



D：durable的缩写，即设置队列的持久化

Queue属性：



Queue.normal1

## 延迟队列

## 优先级队列

具有高优先级的队列具有高的优先权，优先级高的消息具备优先被消费的特权。

x-rnax-priority：设置队列的优先级

Map<String , Object> argss = new HashMap<String, Object>( );

argss.put("x-max-priority",10);  
channel.queueDeclare("queue.norma1",true,false,false ,argss);

在发送消息中设置消息当前的优先级：

AMQP.BasicProperties.Builder builder = new AMQP.BasicProperties.Builder() ;  
builder.priority(5) ;

## 镜像队列

## 生产者确定

### 事物机制

try{  
 channel.txSelect();//开启事物  
 String message = "Hello World2" ;  
 channel.basicPublish(*EXCHANGE\_NAME*,*ROUTING\_KEY*,MessageProperties.*PERSISTENT\_TEXT\_PLAIN*,message.getBytes());  
 channel.txCommit(); //提交事物  
}catch(IOException e){  
 channel.txRollback();//事物回滚  
}

### 发送方确认机制

try{  
 channel.confirmSelect() ; //将信道置为publisher confirm 模式  
 String message = "Hello World2" ;  
 channel.basicPublish(*EXCHANGE\_NAME*,*ROUTING\_KEY*,MessageProperties.*PERSISTENT\_TEXT\_PLAIN*,message.getBytes());  
 if(!channel.waitForConfirms())  
 {  
 System.*out*.println("send message fail");  
 }  
}catch(IOException e){  
 e.printStackTrace();  
}