[**rabbitmq channel参数详解**](http://www.cnblogs.com/piaolingzxh/p/5448927.html)

**1、Channel**

**1.1 channel.exchangeDeclare()：**

type：有direct、fanout、topic三种

durable：true、false true：服务器重启会保留下来Exchange。警告：仅设置此选项，不代表消息持久化。即不保证重启后消息还在。原文：true if we are declaring a durable exchange (the exchange will survive a server restart)

autoDelete:true、false.true:当已经没有消费者时，服务器是否可以删除该Exchange。原文1：true if the server should delete the exchange when it is no longer in use。

copycode.gif

复制代码

/\*\*
\* Declare an exchange.
\* @see com.rabbitmq.client.AMQP.Exchange.Declare
\* @see com.rabbitmq.client.AMQP.Exchange.DeclareOk
\* @param exchange the name of the exchange
\* @param type the exchange type
\* @param durable true if we are declaring a durable exchange (the exchange will survive a server restart)
\* @param autoDelete true if the server should delete the exchange when it is no longer in use
\* @param arguments other properties (construction arguments) for the exchange
\* @return a declaration-confirm method to indicate the exchange was successfully declared
\* @throws java.io.IOException if an error is encountered
\*/
Exchange.DeclareOk exchangeDeclare(String exchange, String type, boolean durable, boolean autoDelete,
Map<String, Object> arguments) throws IOException;

copycode.gif

复制代码

**1.2 chanel.basicQos()**

prefetchSize：0

prefetchCount：会告诉RabbitMQ不要同时给一个消费者推送多于N个消息，即一旦有N个消息还没有ack，则该consumer将block掉，直到有消息ack

global：true\false 是否将上面设置应用于channel，简单点说，就是上面限制是channel级别的还是consumer级别

备注：据说prefetchSize 和global这两项，rabbitmq没有实现，暂且不研究

copycode.gif

复制代码

/\*\*
\* Request specific "quality of service" settings.
\*
\* These settings impose limits on the amount of data the server
\* will deliver to consumers before requiring acknowledgements.
\* Thus they provide a means of consumer-initiated flow control.
\* @see com.rabbitmq.client.AMQP.Basic.Qos
\* @param prefetchSize maximum amount of content (measured in
\* octets) that the server will deliver, 0 if unlimited
\* @param prefetchCount maximum number of messages that the server
\* will deliver, 0 if unlimited
\* @param global true if the settings should be applied to the
\* entire channel rather than each consumer
\* @throws java.io.IOException if an error is encountered
\*/
void basicQos(int prefetchSize, int prefetchCount, boolean global) throws IOException;

copycode.gif

复制代码

**1.3 channel.basicPublish()**

routingKey：路由键，#匹配0个或多个单词，\*匹配一个单词，在topic exchange做消息转发用

mandatory：true：如果exchange根据自身类型和消息routeKey无法找到一个符合条件的queue，那么会调用basic.return方法将消息返还给生产者。false：出现上述情形broker会直接将消息扔掉

*immediate：true：如果exchange在将消息route到queue(s)时发现对应的queue上没有消费者，那么这条消息不会放入队列中。当与消息routeKey关联的所有queue(一个或多个)都没有消费者时，该消息会通过basic.return方法返还给生产者。*

*BasicProperties ：需要注意的是BasicProperties.deliveryMode，0:不持久化 1：持久化 这里指的是消息的持久化，配合channel(durable=true),queue(durable)可以实现，即使服务器宕机，消息仍然保留*

*简单来说：mandatory标志告诉服务器至少将该消息route到一个队列中，否则将消息返还给生产者；immediate标志告诉服务器如果该消息关联的queue上有消费者，则马上将消息投递给它，如果所有queue都没有消费者，直接把消息返还给生产者，不用将消息入队列等待消费者了。*

copycode.gif

复制代码

/\*\*
\* Publish a message.
\*
\* Publishing to a non-existent exchange will result in a channel-level
\* protocol exception, which closes the channel.
\*
\* Invocations of <code>Channel#basicPublish</code> will eventually block if a
\* <a href="http://www.rabbitmq.com/alarms.html">resource-driven alarm</a> is in effect.
\*
\* @see com.rabbitmq.client.AMQP.Basic.Publish
\* @see <a href="http://www.rabbitmq.com/alarms.html">Resource-driven alarms</a>.
\* @param exchange the exchange to publish the message to
\* @param routingKey the routing key
\* @param mandatory true if the 'mandatory' flag is to be set
\* @param immediate true if the 'immediate' flag is to be
\* set. Note that the RabbitMQ server does not support this flag.
\* @param props other properties for the message - routing headers etc
\* @param body the message body
\* @throws java.io.IOException if an error is encountered
\*/
void basicPublish(String exchange, String routingKey, boolean mandatory, boolean immediate, BasicProperties props, byte[] body)
throws IOException;

copycode.gif

复制代码

**1.4 channel.basicAck();**

deliveryTag:该消息的index

multiple：是否批量.true:将一次性ack所有小于deliveryTag的消息。

copycode.gif

复制代码

/\*\*
\* Acknowledge one or several received
\* messages. Supply the deliveryTag from the {@link com.rabbitmq.client.AMQP.Basic.GetOk}
\* or {@link com.rabbitmq.client.AMQP.Basic.Deliver} method
\* containing the received message being acknowledged.
\* @see com.rabbitmq.client.AMQP.Basic.Ack
\* @param deliveryTag the tag from the received {@link com.rabbitmq.client.AMQP.Basic.GetOk} or {@link com.rabbitmq.client.AMQP.Basic.Deliver}
\* @param multiple true to acknowledge all messages up to and
\* including the supplied delivery tag; false to acknowledge just
\* the supplied delivery tag.
\* @throws java.io.IOException if an error is encountered
\*/
void basicAck(long deliveryTag, boolean multiple) throws IOException;

copycode.gif

复制代码

**1.5 channel.basicNack(delivery.getEnvelope().getDeliveryTag(), false, true);**

deliveryTag:该消息的index

multiple：是否批量.true:将一次性拒绝所有小于deliveryTag的消息。

requeue：被拒绝的是否重新入队列

copycode.gif

复制代码

/\*\*
\* Reject one or several received messages.
\*
\* Supply the <code>deliveryTag</code> from the {@link com.rabbitmq.client.AMQP.Basic.GetOk}
\* or {@link com.rabbitmq.client.AMQP.Basic.GetOk} method containing the message to be rejected.
\* @see com.rabbitmq.client.AMQP.Basic.Nack
\* @param deliveryTag the tag from the received {@link com.rabbitmq.client.AMQP.Basic.GetOk} or {@link com.rabbitmq.client.AMQP.Basic.Deliver}
\* @param multiple true to reject all messages up to and including
\* the supplied delivery tag; false to reject just the supplied
\* delivery tag.
\* @param requeue true if the rejected message(s) should be requeued rather
\* than discarded/dead-lettered
\* @throws java.io.IOException if an error is encountered
\*/
void basicNack(long deliveryTag, boolean multiple, boolean requeue)
throws IOException;

copycode.gif

复制代码

**1.5 channel.basicReject(delivery.getEnvelope().getDeliveryTag(), false);**

deliveryTag:该消息的index

requeue：被拒绝的是否重新入队列

**channel.basicNack 与 channel.basicReject 的区别在于basicNack可以拒绝多条消息，而basicReject一次只能拒绝一条消息**

copycode.gif

复制代码

/\*\*
\* Reject a message. Supply the deliveryTag from the {@link com.rabbitmq.client.AMQP.Basic.GetOk}
\* or {@link com.rabbitmq.client.AMQP.Basic.Deliver} method
\* containing the received message being rejected.
\* @see com.rabbitmq.client.AMQP.Basic.Reject
\* @param deliveryTag the tag from the received {@link com.rabbitmq.client.AMQP.Basic.GetOk} or {@link com.rabbitmq.client.AMQP.Basic.Deliver}
\* @param requeue true if the rejected message should be requeued rather than discarded/dead-lettered
\* @throws java.io.IOException if an error is encountered
\*/
void basicReject(long deliveryTag, boolean requeue) throws IOException;

copycode.gif

复制代码

**1.6 channel.basicConsume(QUEUE\_NAME, true, consumer);**

autoAck：是否自动ack，如果不自动ack，需要使用channel.ack、channel.nack、channel.basicReject 进行消息应答

copycode.gif

复制代码

/\*\*
\* Start a non-nolocal, non-exclusive consumer, with
\* a server-generated consumerTag.
\* @param queue the name of the queue
\* @param autoAck true if the server should consider messages
\* acknowledged once delivered; false if the server should expect
\* explicit acknowledgements
\* @param callback an interface to the consumer object
\* @return the consumerTag generated by the server
\* @throws java.io.IOException if an error is encountered
\* @see com.rabbitmq.client.AMQP.Basic.Consume
\* @see com.rabbitmq.client.AMQP.Basic.ConsumeOk
\* @see #basicConsume(String, boolean, String, boolean, boolean, Map, Consumer)
\*/
String basicConsume(String queue, boolean autoAck, Consumer callback) throws IOException;

copycode.gif

复制代码

**1.7 chanel.exchangeBind()**

channel.queueBind(queueName, EXCHANGE\_NAME, bindingKey);

用于通过绑定bindingKey将queue到Exchange，之后便可以进行消息接收

copycode.gif

复制代码

/\*\*
\* Bind an exchange to an exchange, with no extra arguments.
\* @see com.rabbitmq.client.AMQP.Exchange.Bind
\* @see com.rabbitmq.client.AMQP.Exchange.BindOk
\* @param destination the name of the exchange to which messages flow across the binding
\* @param source the name of the exchange from which messages flow across the binding
\* @param routingKey the routine key to use for the binding
\* @return a binding-confirm method if the binding was successfully created
\* @throws java.io.IOException if an error is encountered
\*/
Exchange.BindOk exchangeBind(String destination, String source, String routingKey) throws IOException;

copycode.gif

复制代码

**1.8 channel.queueDeclare(QUEUE\_NAME, false, false, false, null);**

durable：true、false true：在服务器重启时，能够存活

exclusive ：是否为**当前连接**的专用队列，在连接断开后，会自动删除该队列，生产环境中应该很少用到吧。

autodelete：当没有任何消费者使用时，自动删除该队列。this means that the queue will be deleted when there are no more processes consuming messages from it.

copycode.gif

复制代码

/\*\*
\* Declare a queue
\* @see com.rabbitmq.client.AMQP.Queue.Declare
\* @see com.rabbitmq.client.AMQP.Queue.DeclareOk
\* @param queue the name of the queue
\* @param durable true if we are declaring a durable queue (the queue will survive a server restart)
\* @param exclusive true if we are declaring an exclusive queue (restricted to this connection)
\* @param autoDelete true if we are declaring an autodelete queue (server will delete it when no longer in use)
\* @param arguments other properties (construction arguments) for the queue
\* @return a declaration-confirm method to indicate the queue was successfully declared
\* @throws java.io.IOException if an error is encountered
\*/
Queue.DeclareOk queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete,
Map<String, Object> arguments) throws IOException;