

工具类：

```
1 import com.rabbitmq.client.ConnectionFactory;
2
3 /**
4  * RabbitMQ连接工厂工具类
5  * @author qiyang
6  */
7 public class FactoryConnectionUtil {
8     private static final String IP_ADDRESS = "192.168.230.128";
9     private static final int port = 5672;
10    private static final String username = "root";
11    private static final String password = "root";
12    private static ConnectionFactory factory = null;
13
14    static {
15        factory = new ConnectionFactory();
16        factory.setHost(IP_ADDRESS);
17        factory.setPort(port);
18        factory.setUsername(username);
19        factory.setPassword(password);
20    }
21
22    public static ConnectionFactory getFactory(){
23        return factory;
24    }
25 }
```

场景1：单发送单接收



生产者：

```
1 /**
2  * 发送消息
3  * @param exchangeName 交换器名称
4  * @param queueName 队列名称
5  * @param routingKey 路由键
6  */
7 public static void send01(String exchangeName, String queueName, String routingKey){
8     try {
9         Connection connection = FactoryConnectionUtil.getFactory().newConnection();//创建连接
10        Channel channel = connection.createChannel();//创建信道
11        //创建一个type="direct"、持久化、非自动删除的交换器
12        channel.exchangeDeclare(exchangeName, BuiltinExchangeType.DIRECT, true, false, null);
13        //创建一个持久化、非排他的、非自动删除的队列
14        channel.queueDeclare(queueName, true, false, false, null);
15        //将交换器与队列通过路由键绑定
```

```

16     channel.queueBind(queueName, exchangeName, routingKey);
17     //发送一条持久化的消息: hello world
18     String message = "hello world";
19     channel.basicPublish(exchangeName, routingKey, MessageProperties.PERSISTENT_TEXT_PLAIN, mes
20     channel.close();
21 } catch (Exception e) {
22     e.printStackTrace();
23 }
24 }

```

消费者:

```

1  /**
2   * 接收消息
3   * @param queueName 队列名称
4   */
5  public static void receive_01(String queueName){
6      try {
7          Connection connection = FactoryConnectionUtil.getFactory().newConnection();/* 创建连接 */
8          Channel channel = connection.createChannel();/*创建信道
9          Consumer consumer = new DefaultConsumer(channel){
10
11              @Override
12              public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties
13                  System.out.println("收到消息 : " + new String(body));
14                  channel.basicAck(envelope.getDeliveryTag(), false);
15              }
16          };
17
18          channel.basicConsume(queueName, consumer);
19
20          //休眠60秒后, 信道关闭, 关闭之后无法接收消息
21          TimeUnit.SECONDS.sleep(60);
22          channel.close();
23      } catch (Exception e) {
24          e.printStackTrace();
25      }
26  }

```

测试代码:

发送端:

```

1  @Test
2  public void testSend01() {
3      Producer.send01("ex1", "q1", "k1");
4  }

```

接收端:

```

1  @Test
2  public void testReceive_01(){
3      MyConsumer.receive_01("q1");
4  }

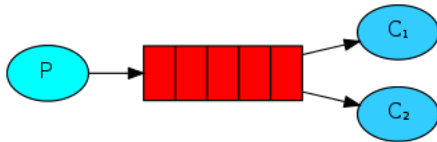
```

结果:

- 1 生产者发送的消息, 发送几条, 都会经过RabbitMQ

- 2 如果发送的消息，消费者还没有消费掉，那么在RabbitMQ中会记录有几条消息，如果已经被消费者消费掉了，那么消息就不会在RabbitMQ中记录
- 3 如果消费者连接之后，会监听RabbitMQ中的消息，如果有就会获取到

场景2：单发送多接收



生产者：

```
1  /**
2   * 发送消息
3   * @param exchangeName 交换器名称
4   * @param queueName 队列名称
5   * @param routingKey 路由键
6   */
7  public static void send01(String exchangeName, String queueName, String routingKey){
8      try {
9          Connection connection = FactoryConnectionUtil.getFactory().newConnection();//创建连接
10         Channel channel = connection.createChannel();//创建信道
11         //创建一个type="direct"、持久化、非自动删除的交换器
12         channel.exchangeDeclare(exchangeName, BuiltinExchangeType.DIRECT, true, false, null);
13         //创建一个持久化、非排他的、非自动删除的队列
14         channel.queueDeclare(queueName, true, false, false, null);
15         //将交换器与队列通过路由键绑定
16         channel.queueBind(queueName, exchangeName, routingKey);
17         //发送一条持久化的消息: hello world
18         String message = "hello world";
19         channel.basicPublish(exchangeName, routingKey, MessageProperties.PERSISTENT_TEXT_PLAIN, message);
20         channel.close();
21     } catch (Exception e) {
22         e.printStackTrace();
23     }
24 }
```

消费者：

```
1  /**
2   * 接收消息
3   * @param queueName 队列名称
4   * @param name 名称
5   */
6  public static void receive_02(String queueName, String name ){
7      try {
8          Connection connection = FactoryConnectionUtil.getFactory().newConnection();// 创建连接 */
9          Channel channel = connection.createChannel();//创建信道
10         Consumer consumer = new DefaultConsumer(channel){
11
12             @Override
13             public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties
14                 properties, byte[] body) throws IOException {
15                 System.out.println( name + "收到消息 : " + new String(body));
16                 channel.basicAck(envelope.getDeliveryTag(), false);
17             }
18         };
19     }
20 }
```

```

16         }
17     };
18
19     channel.basicConsume(queueName, consumer);
20
21     //休眠60秒后，信道关闭，关闭之后无法接收消息
22     TimeUnit.SECONDS.sleep(60);
23     channel.close();
24 } catch (Exception e) {
25     e.printStackTrace();
26 }
27 }

```

测试代码：

生产者

```

1 @Test
2 public void testSend02(){
3     for (int i = 0; i < 5; i++) {
4         Producer.send01("ex1", "q1", "k1");
5     }
6 }

```

消费者

```

1 @Test
2 public void testReceive_02_A(){
3     MyConsumer.receive_02("q1", "name1");
4 }
5
6 @Test
7 public void testReceive_02_B(){
8     MyConsumer.receive_02("q1", "name2");
9 }

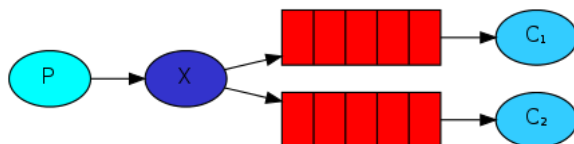
```

先运行消费者代码：分别先运行testReceive_02_A、testReceive_02_B
之后运行生产者代码

测试结果：

生产者发送了五条信息，结果发现有的数据被testReceive_02_A 接收到了，有的被testReceive_02_B接收到了

场景3：发布/订阅模式



生产者：

```

1 /**
2  * 发布订阅消息
3  * @param exchangeName 交换器名称
4  * @param queueName1 队列1名称
5  * @param queueName2 队列2名称

```

```

6  * @param routingKey 路由键
7  */
8  public static void send_02(String exchangeName, String queueName1, String queueName2, String routingKey) {
9      try {
10         Connection connection = FactoryConnectionUtil.getFactory().newConnection();//创建连接
11         Channel channel = connection.createChannel();//创建信道
12         //创建一个type="FANOUT"、持久化、非自动删除的交换器
13         channel.exchangeDeclare(exchangeName, BuiltinExchangeType.FANOUT, true, false, null);
14         //创建一个持久化、非排他的、非自动删除的队列
15         channel.queueDeclare(queueName1, true, false, false, null);
16         //将交换器与队列通过路由键绑定
17         channel.queueBind(queueName1, exchangeName, routingKey);
18         channel.queueDeclare(queueName2, true, false, false, null);
19         channel.queueBind(queueName2, exchangeName, routingKey);
20         //发送一条持久化的消息: hello world
21         String message = "hello world";
22         channel.basicPublish(exchangeName, routingKey, MessageProperties.PERSISTENT_TEXT_PLAIN, message);
23         channel.close();
24     } catch (Exception e) {
25         e.printStackTrace();
26     }
27 }

```

消费者:

```

1  /**
2   * 接收消息
3   * @param queueName 队列名称
4   * @param name 名称
5   */
6  public static void receive_02(String queueName, String name) {
7      try {
8         Connection connection = FactoryConnectionUtil.getFactory().newConnection();//创建连接 */
9         Channel channel = connection.createChannel();//创建信道
10         Consumer consumer = new DefaultConsumer(channel){
11
12             @Override
13             public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties
14                 properties, byte[] body) throws Exception {
15                 System.out.println(name + "收到消息 : " + new String(body));
16                 channel.basicAck(envelope.getDeliveryTag(), false);
17             }
18         };
19         channel.basicConsume(queueName, consumer);
20
21         //休眠60秒后, 信道关闭, 关闭之后无法接收消息
22         TimeUnit.SECONDS.sleep(60);
23         channel.close();
24     } catch (Exception e) {
25         e.printStackTrace();
26     }
27 }

```

测试代码:

生产者：

发送消息到两个队列上

```
1 @Test
2 public void testSend03(){
3     for (int i = 0; i < 5; i++) {
4         Producer.send_02("ex2", "q_01", "q_02", "rKey");
5     }
6 }
```

消费者：

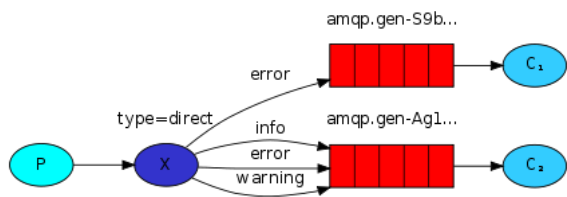
分别接收两个不同的队列

```
1 @Test
2 public void testReceive_03_A(){
3     MyConsumer.receive_02("q_01", "name1");
4 }
5
6 @Test
7 public void testReceive_03_B(){
8     MyConsumer.receive_02("q_02", "name2");
9 }
```

测试结果：

两个消费者，都可以接收到5条信息

场景4：按线路(routing)发送



生产者：

如上图所示发送消息到队列A和B上，结果如图所示

```
1 public static void send_03(String exchangeName){
2     try{
3         String queueA = "queueA";
4         String queueB = "queueB";
5         String routingKeyError = "error";
6         String routingKeyInfo = "info";
7         String routingKeyWarning="warning";
8         Connection connection = FactoryConnectionUtil.getFactory().newConnection();//创建连接
9         Channel channel = connection.createChannel();//创建信道
10        //创建一个type="direct"、持久化、非自动删除的交换器
11        channel.exchangeDeclare(exchangeName, BuiltinExchangeType.DIRECT, true, false, null);
12        //创建一个持久化、非排他的、非自动删除的队列
13        channel.queueDeclare(queueA, true, false, false, null);
14        channel.queueBind(queueA, exchangeName, routingKeyError);
15
16        channel.queueDeclare(queueB, true, false, false, null);
```

```

17     channel.queueBind(queueB, exchangeName, routingKeyError);
18     channel.queueBind(queueB, exchangeName, routingKeyInfo);
19     channel.queueBind(queueB, exchangeName, routingKeyWarning);
20
21     String messageInfo = "info xxx";
22     String messageError = "error xxx";
23     String messageWarning = "warning xxx";
24     channel.basicPublish(exchangeName, routingKeyError, MessageProperties.PERSISTENT_TEXT_PLAIN, messageInfo);
25     channel.basicPublish(exchangeName, routingKeyInfo, MessageProperties.PERSISTENT_TEXT_PLAIN, messageInfo);
26     channel.basicPublish(exchangeName, routingKeyWarning, MessageProperties.PERSISTENT_TEXT_PLAIN, messageWarning);
27     channel.close();
28 } catch (Exception e) {
29     e.printStackTrace();
30 }
31 }

```

消费者:

```

1  /**
2   * 接收消息
3   * @param queueName 队列名称
4   */
5  public static void receive_01(String queueName) {
6      try {
7          Connection connection = FactoryConnectionUtil.getFactory().newConnection(); /* 创建连接 */
8          Channel channel = connection.createChannel(); /* 创建信道 */
9          Consumer consumer = new DefaultConsumer(channel);
10
11          @Override
12          public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties
13              System.out.println("收到消息 : " + new String(body));
14              channel.basicAck(envelope.getDeliveryTag(), false);
15          }
16      };
17
18      channel.basicConsume(queueName, consumer);
19
20      //休眠60秒后, 信道关闭, 关闭之后无法接收消息
21      TimeUnit.SECONDS.sleep(60);
22      channel.close();
23  } catch (Exception e) {
24      e.printStackTrace();
25  }
26 }

```

测试代码:

生产者

```

1  @Test
2  public void testSend04() {
3      Producer.send_03("myexchange");
4  }

```

消费者

```

1  @Test

```

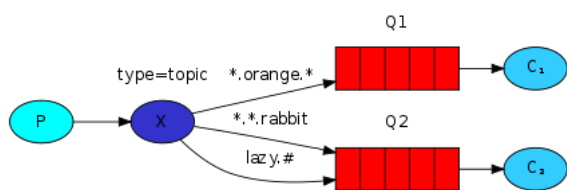
```

2 public void testReceive_04(){
3     MyConsumer.receive_01("queueA");
4 }
5
6 @Test
7 public void testReceive_05(){
8     MyConsumer.receive_01("queueB");
9 }

```

测试结果 如图所示

场景4：Topic方式发送



生产者：

```

1 public static void send05(String exChangeName){
2     try{
3         String queueA = "queueA";
4         String queueB = "queueB";
5         Connection connection = FactoryConnectionUtil.getFactory().newConnection();//创建连接
6         Channel channel = connection.createChannel();//创建信道
7         channel.exchangeDeclare(exChangeName,BuiltinExchangeType.TOPIC);
8
9         channel.queueDeclare(queueA, true, false, false, null);
10        channel.queueBind(queueA, exChangeName, "*.orange.*");
11
12        channel.queueDeclare(queueB, true, false, false, null);
13        channel.queueBind(queueB, exChangeName, ".*.rabbit");
14        channel.queueBind(queueB, exChangeName, "lazy.#");
15
16        channel.basicPublish(exChangeName, "quick.orange.rabbit", null, "quick.orange.rabbit msg!!!");
17        channel.basicPublish(exChangeName, "lazy.write.elephant", null, "lazy.write.elephant msg !!!");
18
19        channel.close();
20    }catch (Exception e){
21        e.printStackTrace();
22    }
23 }

```

消费者：

```

1 public static void receive_01(String queueName){
2     try {
3         Connection connection = FactoryConnectionUtil.getFactory().newConnection();// 创建连接 */
4         Channel channel = connection.createChannel();//创建信道
5         Consumer consumer = new DefaultConsumer(channel){
6
7             @Override

```



```

8         public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties p
9             System.out.println("收到消息 : " + new String(body));
10            channel.basicAck(envelope.getDeliveryTag(), false);
11        }
12    };
13
14    channel.basicConsume(queueName, consumer);
15
16    //休眠60秒后, 信道关闭, 关闭之后无法接收消息
17    TimeUnit.SECONDS.sleep(60);
18    channel.close();
19 } catch (Exception e) {
20     e.printStackTrace();
21 }
22 }

```

测试代码:

生产者

```

1 @Test
2 public void testSent05(){
3     Producer.send05("topicExchange");
4 }

```

消费者

```

1 @Test
2 public void testReceive_04(){
3     MyConsumer.receive_01("queueA");
4 }
5
6 @Test
7 public void testReceive_05(){
8     MyConsumer.receive_01("queueB");
9 }

```

测试结果:

如图所示: queueA只有一个消息, 而queueB有两个消息
 "quick.orange.rabbit msg" 这条消息两边都有收到