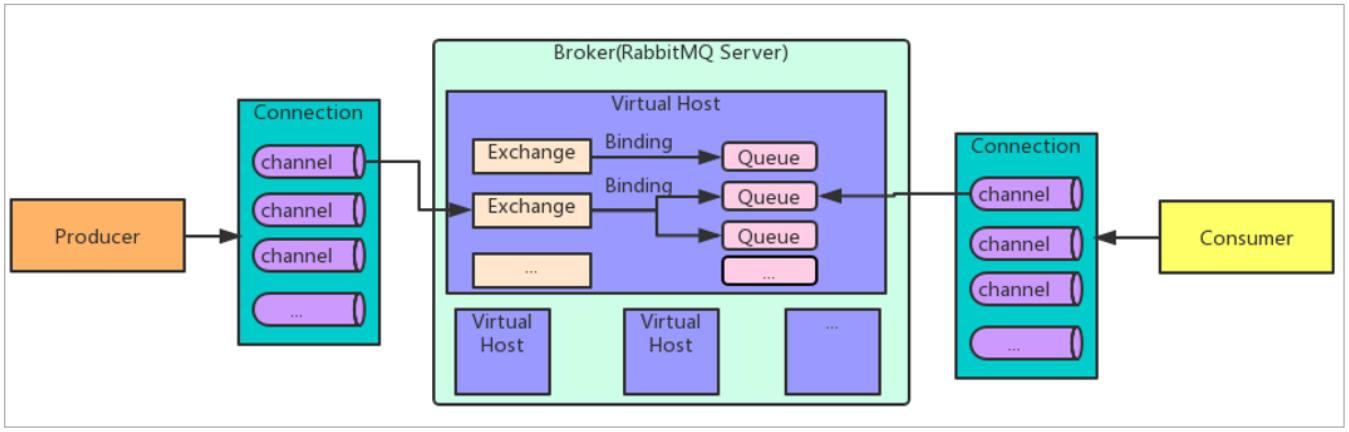
RabbitMQ基础

RabbitMQ 简介

AMQP，即Advanced Message Queuing Protocol（高级消息队列协议），是一个网络协议，是应用层协议的一个开放标准，为面向消息的中间件设计。基于此协议的客户端与消息中间件可传递消息，并不受客户端/中间件不同产品，不同的开发语言等条件的限制。2006年，AMQP规范发布。类比HTTP。

RabbitMQ 基础架构如下图:



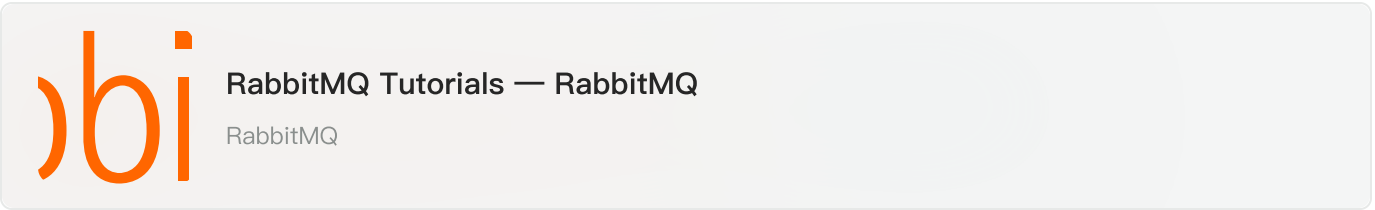
相关概念

- Broker：接收和分发消息的应用，RabbitMQ Server就是 Message Broker
- Virtual host：出于多租户和安全因素设计的，把 AMQP 的基本组件划分到一个虚拟的分组中，类似于网络中的 namespace 概念。当多个不同的用户使用同一个 RabbitMQ server 提供的服务时，可以划分出多个vhost，每个用户在自己的 vhost 创建 exchange / queue 等
- Connection：publisher / consumer 和 broker 之间的 TCP 连接
- Channel：如果每一次访问 RabbitMQ 都建立一个 Connection，在消息量大的时候建立 TCP Connection的开销将是巨大的，效率也较低。Channel 是在 connection 内部建立的逻辑连接，如果应用程序支持多线程，通常每个thread创建单独的 channel 进行通讯，AMQP method 包含了channel id 帮助客户端和message broker 识别 channel，所以 channel 之间是完全隔离的。Channel 作为轻量级的 Connection 极大减少了操作系统建立 TCP connection 的开销
- Exchange：message 到达 broker 的第一站，根据分发规则，匹配查询表中的 routing key，分发消息到queue 中去。常用的类型有：direct (point-to-point), topic (publish-subscribe) and fanout (multicast)
- Queue：消息最终被送到这里等待 consumer 取走
- Binding：exchange 和 queue 之间的虚拟连接，binding 中可以包含 routing key。Binding 信息被保存到 exchange 中的查询表中，用于 message 的分发依据

工作模式

RabbitMQ提供了6种工作模式：简单模式、work queues、Publish/Subscribe发布与订阅模式、Routing路由模式、Topics主题模式、RPC远程调用模式（远程调用，不太算MQ；暂不作介绍）。

官网对应模式介绍：



1.简单模式HelloWorld

一个生产者、一个消费者，不需要设置交换机(使用默认的交换机)。

2.工作队列模式 Work Queue

一个生产者、多个消费者(竞争关系)，不需要设置交换机(使用默认的交换机)。

3.发布订阅模式Publish/subscribe

需要设置类型为fanout的交换机，并且交换机和队列进行绑定，当发送消息到交换机后，交换机会将消息发送到绑定的队列。

4.路由模式Routing

需要设置类型为direct 的交换机，交换机和队列进行绑定，并且指定routing key，当发送消息到交换机后，交换机会根据routing key将消息发送到对应的队列。

5.通配符模式Topic

需要设置类型为topic的交换机，交换机和队列进行绑定，并且指定通配符方式的 routing key，当发送消息到交换机后，交换机会根据routing key将消息发送到对应的队列。

JMS

- JMS即Java消息服务（JavaMessage Service）应用程序接口，是一个Java平台中关于面向消息中间件的API
- JMS 是 JavaEE规范中的一种，类比JDBC
- 很多消息中间件都实现了JMS规范，例如：ActiveMQ。RabbitMQ 官方没有提供 JMS 的实现包，但是开源社区有

工作模式–简单模式

在上图的模型中，有以下概念：

P：生产者，也就是要发送消息的程序

C：消费者：消息的接收者，会一直等待消息到来

queue：消息队列，图中红色部分。类似一个邮箱，可以缓存消息；生产者向其中投递消息，消费者从其中取出消息

Java | 复制代码

```java
/**
 * 简单模式-发送消息
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/21 15:57
 */
public class ProducerHelloWorld {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.37.37");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        //5.创建队列
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete, Map<String, Object> arguments)
          1.queue：队列名称
          2.durable：是否持久化，mq重启后是否存在
          3.exclusive：
                  是否独占，只能有一个消费者监听这队列
                  当connection关闭时，是否删除队列
          4.autoDelete：是否自动删除
          5.arguments：其他参数
         */
        //没有才会创建
        channel.queueDeclare("hello_world",true,false,false,null);
        /*
         basicPublish(String exchange, String routingKey, BasicProperties props, byte[] body)
           1.exchange: 交换机名称。简单模式会使用默认的""
```

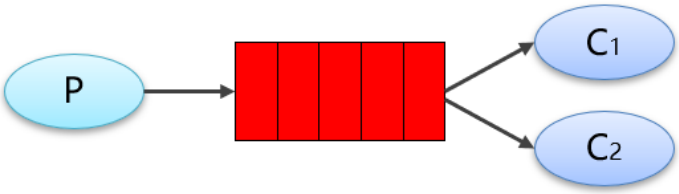
```
            2.routingKey：路由键名称要和对应的队列名称一样才能路由到
            3.props：配置信息
            4.body：主要发送体
         */
        //6.发送消息
        String body = "hello rabbitmq ~~~";
        channel.basicPublish("","hello_world",null,body.getBytes(StandardC
harsets.UTF_8));
        channel.close();
        connection.close();

    }

}
```

Java | 复制代码

```java
/**
 * 简单模式-接收消息
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/21 15:57
 */
public class ConsumerHelloWorld {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.37.37");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        //5.创建队列
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete, Map<String, Object> arguments)
          1.queue：队列名称
          2.durable：是否持久化，mq重启后是否存在
          3.exclusive：
                  是否独占，只能有一个消费者监听这队列
                  当connection关闭时，是否删除队列
          4.autoDelete：是否自动删除
          5.arguments：其他参数
         */
        //没有才会创建
        channel.queueDeclare("hello_world",true,false,false,null);
        /*
         basicConsume(String queue,boolean autoAck,Consumer consumer)
           1.queue: 队列名称。
           2.autoAck: 是否自动确认
```

```
            3.consumer：回调对象
         */
        //6.接收消息
        Consumer consumer =new DefaultConsumer(channel){
            @Override
            public void handleDelivery(String s, Envelope envelope, AMQP.BasicProperties basicProperties, byte[] body) {
                System.out.println("consumerTag:"+s);
                System.out.println("routingKey:"+envelope.getRoutingKey());
                System.out.println("exchange:"+envelope.getExchange());
                System.out.println("properties:"+basicProperties);
                System.out.println("body:"+new String(body));
            }
        };
        channel.basicConsume("hello_world",true,consumer);

    }

}
```

工作模式–WorkQueue(工作队列)



Work Queues：与入门程序的简单模式相比，多了一个或一些消费端，多个消费端共同消费同一个队列中的消息。

应用场景：对于任务过重或任务较多情况使用工作队列可以提高任务处理的速度。例如：短信服务部署多个，只需要有一个节点成功发送即可。

在一个队列中如果有多个消费者，那么消费者之间对于同一个消息的关系是**竞争**的关系。

Java | 复制代码

```java
/**
 * workQueue模式-发送消息
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/21 15:57
 */
public class ProducerWorkQueue {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        //5.创建队列
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete, Map<String, Object> arguments)
          1.queue：队列名称
          2.durable：是否持久化，mq重启后是否存在
          3.exclusive：
                  是否独占，只能有一个消费者监听这队列
                  当connection关闭时，是否删除队列
          4.autoDelete：是否自动删除
          5.arguments：其他参数
         */
        //没有才会创建
        channel.queueDeclare("workQueue",true,false,false,null);
        /*
         basicPublish(String exchange, String routingKey, BasicProperties props, byte[] body)
           1.exchange: 交换机名称。简单模式会使用默认的""
```

```
            2.routingKey：路由键名称要和对应的队列名称一样才能路由到
            3.props：配置信息
            4.body：主要发送体
         */
        //6.发送消息
        for (int i = 0; i < 10; i++) {
            String body = i+"、hello rabbitmq workQueue";
            channel.basicPublish("","workQueue",null,body.getBytes(StandardCharsets.UTF_8));
        }
        channel.close();
        connection.close();

    }

}
```

Java | 复制代码

```java
/**
 * workQueue模式-接收消息
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/24 9:57
 */
public class ConsumerWorkQueue {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        //5.创建队列
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete, Map<String, Object> arguments)
          1.queue：队列名称
          2.durable：是否持久化，mq重启后是否存在
          3.exclusive：
                  是否独占，只能有一个消费者监听这队列
                  当connection关闭时，是否删除队列
          4.autoDelete：是否自动删除
          5.arguments：其他参数
         */
        //没有才会创建
        channel.queueDeclare("hello_world",true,false,false,null);
        /*
         basicConsume(String queue,boolean autoAck,Consumer consumer)
           1.queue: 队列名称。
           2.autoAck: 是否自动确认
```

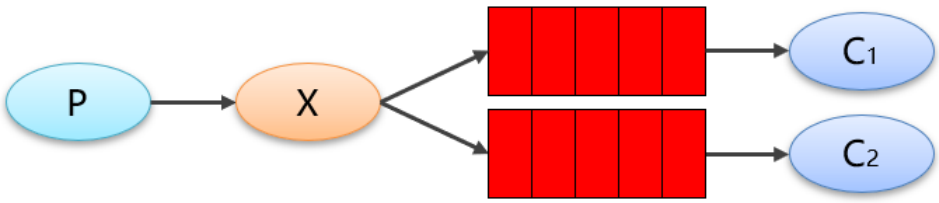
```
            3.consumer: 回调对象
         */
        //6.接收消息
        Consumer consumer =new DefaultConsumer(channel){
            @Override
            public void handleDelivery(String s, Envelope envelope, AMQP.BasicProperties basicProperties, byte[] body) {
                System.out.println("consumerTag:"+s);
                System.out.println("routingKey:"+envelope.getRoutingKey());
                System.out.println("exchange:"+envelope.getExchange());
                System.out.println("properties:"+basicProperties);
                System.out.println("body:"+new String(body));
            }
        };
        channel.basicConsume("workQueue",true,consumer);

    }

}
```

Java | 复制代码

```java

/**
 * workQueue模式-接收消息2
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/24 9:57
 */
public class ConsumerWorkQueue2 {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        //5.创建队列
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete, Map<String, Object> arguments)
          1.queue：队列名称
          2.durable：是否持久化，mq重启后是否存在
          3.exclusive：
                  是否独占，只能有一个消费者监听这队列
                  当connection关闭时，是否删除队列
          4.autoDelete：是否自动删除
          5.arguments：其他参数
         */
        //没有才会创建
        channel.queueDeclare("workQueue",true,false,false,null);
        /*
         basicConsume(String queue,boolean autoAck,Consumer consumer)
           1.queue: 队列名称。
```

```
            2.autoAck: 是否自动确认
            3.consumer: 回调对象
         */
        //6.接收消息
        Consumer consumer =new DefaultConsumer(channel){
            @Override
            public void handleDelivery(String s, Envelope envelope, AMQP.BasicProperties basicProperties, byte[] body) {
                System.out.println("consumerTag:"+s);
                System.out.println("routingKey:"+envelope.getRoutingKey());
                System.out.println("exchange:"+envelope.getExchange());
                System.out.println("properties:"+basicProperties);
                System.out.println("body:"+new String(body));
            }
        };
        channel.basicConsume("workQueue",true,consumer);

    }

}
```

工作模式–Pub/Sub 订阅模式



在订阅模型中，多了一个 Exchange 角色，而且过程略有变化：

P：生产者，也就是要发送消息的程序，但是不再发送到队列中，而是发给X（交换机）

C：消费者，消息的接收者，会一直等待消息到来

Queue：消息队列，接收消息、缓存消息

Exchange：交换机（X）。一方面，接收生产者发送的消息。另一方面，知道如何处理消息，例如递交给某个特别队列、递交给所有队列、或是将消息丢弃。到底如何操作，取决于Exchange的类型。Exchange有常见以下3种类型：

- Fanout：广播，将消息交给所有绑定到交换机的队列

- Direct：定向，把消息交给符合指定routing key 的队列
- Topic：通配符，把消息交给符合routing pattern（路由模式） 的队列

Exchange（交换机）只负责转发消息，不具备存储消息的能力，因此如果没有任何队列与 Exchange 绑定，或者没有符合路由规则的队列，那么消息会丢失！

Java | 复制代码

```java

/**
 * 订阅模式-发送消息
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/21 15:57
 */
public class ProducerPubSub {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection = connectionFactory.newConnection();
        //4.创建Channel
        Channel channel = connection.createChannel();
        //5.创建交换机
        /*
        exchangeDeclare(String exchange, BuiltinExchangeType type, boolean durable, boolean autoDelete, boolean internal, Map<String, Object> arguments)
          1.exchange：交换机名称
          2.type：交换机类型
             DIRECT("direct")：定向
             FANOUT("fanout")：扇形（广播）
             TOPIC("topic")：通配符的方式
             HEADERS("headers")：参数匹配
          3.durable：是否持久化
          4.autoDelete：自动删除
          5.internal：内部使用，一般使用false
          6.arguments：参数
         */
        String exchangeName = "test_fanout";
```

```
        channel.exchangeDeclare(exchangeName, BuiltinExchangeType.FANOUT,true,false,false,null);
        //6.创建队列
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete, Map<String, Object> arguments)
          1.queue：队列名称
          2.durable：是否持久化，mq重启后是否存在
          3.exclusive：
                  是否独占，只能有一个消费者监听这队列
                  当connection关闭时，是否删除队列
          4.autoDelete：是否自动删除
          5.arguments：其他参数
         */
        //创建队列，没有才会创建
        String queue1Name = "test_fanout_queue1";
        String queue2Name = "test_fanout_queue2";
        channel.queueDeclare(queue1Name, true, false, false, null);
        channel.queueDeclare(queue2Name, true, false, false, null);
        //7.绑定队列和交换机
        /*
        queueBind(String queue, String exchange, String routingKey)
        参数：
            1.queue：队列名称
            2.exchange：交换机名称
            3.routingKey：路由键，绑定规则
              如果交换机绑定的是fanout，设置为""
         */
        channel.queueBind(queue1Name,exchangeName,"");
        channel.queueBind(queue2Name,exchangeName,"");
        /*
         basicPublish(String exchange, String routingKey, BasicProperties props, byte[] body)
           1.exchange: 交换机名称。简单模式会使用默认的""
           2.routingKey：路由键名称要和对应的队列名称一样才能路由到
           3.props：配置信息
           4.body：主要发送体
         */
        //8.发送消息
        String body = "日志信息：张三调用了fanout方法。。。日志级别：info。。。";
        channel.basicPublish(exchangeName, "", null, body.getBytes(StandardCharsets.UTF_8));
        //9.释放资源
        channel.close();
        connection.close();

    }
```

```
87  }
88
```

Java | 复制代码

```java
/**
 * 订阅模式-接收消息1
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/24 9:57
 */
public class ConsumerPubSub1 {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        String queue1Name = "test_fanout_queue1";
        /*
         basicConsume(String queue,boolean autoAck,Consumer consumer)
           1.queue: 队列名称。
           2.autoAck: 是否自动确认
           3.consumer: 回调对象
         */
        //6.接收消息
        Consumer consumer =new DefaultConsumer(channel){
            @Override
            public void handleDelivery(String s, Envelope envelope, AMQP.BasicProperties basicProperties, byte[] body) {
                System.out.println("body:"+new String(body));
                System.out.println("将日志打印到控制台");
            }
        };
        channel.basicConsume(queue1Name,true,consumer);

```

```
44          }
45
46      }
```

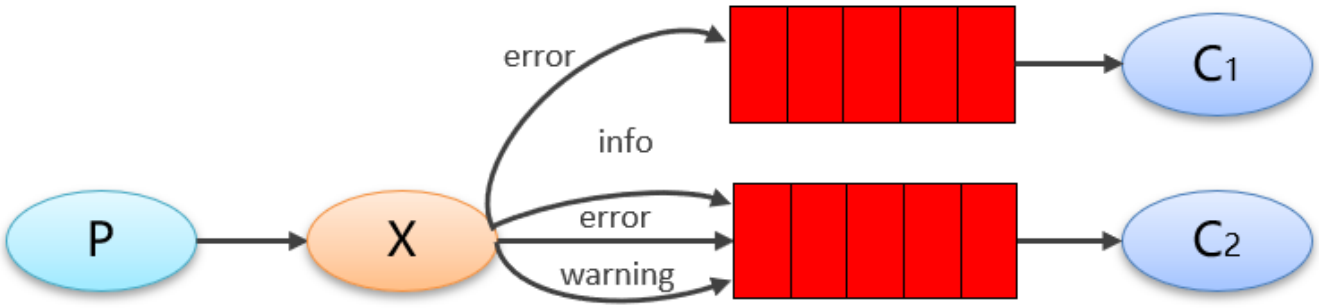
Java | 复制代码

```java
/**
 * 订阅模式-接收消息2
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/24 9:57
 */
public class ConsumerPubSub2 {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        String queue2Name = "test_fanout_queue2";
        /*
         basicConsume(String queue,boolean autoAck,Consumer consumer)
           1.queue: 队列名称。
           2.autoAck: 是否自动确认
           3.consumer: 回调对象
         */
        //6.接收消息
        Consumer consumer =new DefaultConsumer(channel){
            @Override
            public void handleDelivery(String s, Envelope envelope, AMQP.BasicProperties basicProperties, byte[] body) {
                System.out.println("body:"+new String(body));
                System.out.println("将日志保存至数据库");
            }
        };
        channel.basicConsume(queue2Name,true,consumer);

```

```
        }

    }
```

工作模式–Routing 路由模式

- 队列与交换机的绑定，不能是任意绑定了，而是要指定一个 RoutingKey（路由key）
- 消息的发送方在向 Exchange 发送消息时，也必须指定消息的 RoutingKey
- Exchange 不再把消息交给每一个绑定的队列，而是根据消息的 Routing Key 进行判断，只有队列的Routingkey 与消息的 Routing key 完全一致，才会接收到消息



P：生产者，向 Exchange 发送消息，发送消息时，会指定一个routing key

X：Exchange（交换机），接收生产者的消息，然后把消息递交给与 routing key 完全匹配的队列

C1：消费者，其所在队列指定了需要 routing key 为 error 的消息

C2：消费者，其所在队列指定了需要 routing key 为 info、error、warning 的消息

Java | 复制代码

```java

/**
 * RoutingKey模式（路由）-发送消息
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/21 15:57
 */
public class ProducerRouting {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection = connectionFactory.newConnection();
        //4.创建Channel
        Channel channel = connection.createChannel();
        //5.创建交换机
        /*
        exchangeDeclare(String exchange, BuiltinExchangeType type, boolean durable, boolean autoDelete, boolean internal, Map<String, Object> arguments)
          1.exchange：交换机名称
          2.type：交换机类型
             DIRECT("direct")：定向
             FANOUT("fanout")：扇形（广播）
             TOPIC("topic")：通配符的方式
             HEADERS("headers")：参数匹配
          3.durable：是否持久化
          4.autoDelete：自动删除
          5.internal：内部使用，一般使用false
          6.arguments：参数
         */
        String exchangeName = "test_direct";
```

```
        channel.exchangeDeclare(exchangeName, BuiltinExchangeType.DIRECT,true,false,false,null);
        //6.创建队列
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete, Map<String, Object> arguments)
          1.queue：队列名称
          2.durable：是否持久化，mq重启后是否存在
          3.exclusive：
                  是否独占，只能有一个消费者监听这队列
                  当connection关闭时，是否删除队列
          4.autoDelete：是否自动删除
          5.arguments：其他参数
         */
        //创建队列，没有才会创建
        String queue1Name = "test_direct_queue1";
        String queue2Name = "test_direct_queue2";
        channel.queueDeclare(queue1Name, true, false, false, null);
        channel.queueDeclare(queue2Name, true, false, false, null);
        //7.绑定队列和交换机
        /*
        queueBind(String queue, String exchange, String routingKey)
        参数：
            1.queue：队列名称
            2.exchange：交换机名称
            3.routingKey：路由键，绑定规则
              如果交换机绑定的是fanout，设置为""
         */
        //队列一绑定：error
        channel.queueBind(queue1Name,exchangeName,"error");
        //队列二绑定：error，info，warning
        channel.queueBind(queue2Name,exchangeName,"info");
        channel.queueBind(queue2Name,exchangeName,"error");
        channel.queueBind(queue2Name,exchangeName,"warning");
        /*
         basicPublish(String exchange, String routingKey, BasicProperties props, byte[] body)
           1.exchange: 交换机名称。简单模式会使用默认的""
           2.routingKey：路由键名称要和对应的队列名称一样才能路由到
           3.props：配置信息
           4.body：主要发送体
         */
        //8.发送消息
        String body = "日志信息：张三调用了delete方法。。报错。日志级别：error。。。";
        channel.basicPublish(exchangeName, "error", null, body.getBytes(StandardCharsets.UTF_8));
        //9.释放资源
```

```
86            channel.close();
87            connection.close();
88
89        }
90
91    }
92
```

Java | 复制代码

```java
/**
 * RoutingKey模式（路由）-接收消息1
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/24 9:57
 */
public class ConsumerRouting1 {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        String queue1Name = "test_direct_queue1";
        /*
         basicConsume(String queue,boolean autoAck,Consumer consumer)
           1.queue: 队列名称。
           2.autoAck: 是否自动确认
           3.consumer: 回调对象
         */
        //6.接收消息
        Consumer consumer =new DefaultConsumer(channel){
            @Override
            public void handleDelivery(String s, Envelope envelope, AMQP.BasicProperties basicProperties, byte[] body) {
                System.out.println("body:"+new String(body));
                System.out.println("将日志保存至数据库");
            }
        };
        channel.basicConsume(queue1Name,true,consumer);

```

```
44          }
45
46      }
47
```

Java | 复制代码

```java
/**
 * RoutingKey模式（路由）-接收消息2
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/24 9:57
 */
public class ConsumerRouting2 {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        String queue2Name = "test_direct_queue2";
        /*
         basicConsume(String queue,boolean autoAck,Consumer consumer)
           1.queue: 队列名称。
           2.autoAck: 是否自动确认
           3.consumer: 回调对象
         */
        //6.接收消息
        Consumer consumer =new DefaultConsumer(channel){
            @Override
            public void handleDelivery(String s, Envelope envelope, AMQP.BasicProperties basicProperties, byte[] body) {
                System.out.println("body:"+new String(body));
                System.out.println("将日志打印到控制台");
            }
        };
        channel.basicConsume(queue2Name,true,consumer);

```

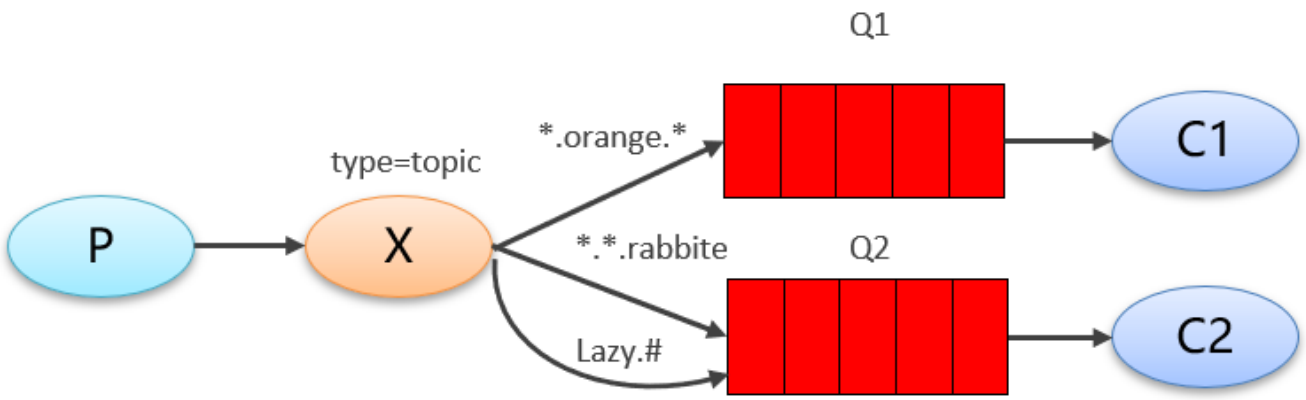
```
44          }
45
46      }
47
```

工作模式–通配符模式

Topic 类型与 Direct 相比，都是可以根据 RoutingKey 把消息路由到不同的队列。只不过 Topic 类型Exchange 可以让队列在绑定 Routing key 的时候使用**通配符**!

Routingkey 一般都是有一个或多个单词组成，多个单词之间以”.”分割，例如： item.insert

通配符规则：# 匹配一个或多个词，* 匹配不多不少恰好1个词，例如：item.# 能够匹配 item.insert.abc 或者 item.insert，item.* 只能匹配 item.insert



红色 Queue：绑定的是 usa.# ， 因此凡是以 usa. 开头的 routing key 都会被匹配到

黄色 Queue：绑定的是 #.news ， 因此凡是以 .news 结尾的 routing key 都会被匹配

Java | 复制代码

```java
/**
 * 通配符模式-发送消息
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/21 15:57
 */
public class ProducerTopic {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection = connectionFactory.newConnection();
        //4.创建Channel
        Channel channel = connection.createChannel();
        //5.创建交换机
        /*
        exchangeDeclare(String exchange, BuiltinExchangeType type, boolean durable, boolean autoDelete, boolean internal, Map<String, Object> arguments)
          1.exchange：交换机名称
          2.type：交换机类型
             DIRECT("direct")：定向
             FANOUT("fanout")：扇形（广播）
             TOPIC("topic")：通配符的方式
             HEADERS("headers")：参数匹配
          3.durable：是否持久化
          4.autoDelete：自动删除
          5.internal：内部使用，一般使用false
          6.arguments：参数
         */
        String exchangeName = "test_topic";

```

```
        channel.exchangeDeclare(exchangeName, BuiltinExchangeType.TOPIC,true,false,false,null);
        //6.创建队列
        /*
        queueDeclare(String queue, boolean durable, boolean exclusive, boolean autoDelete, Map<String, Object> arguments)
          1.queue：队列名称
          2.durable：是否持久化，mq重启后是否存在
          3.exclusive：
                  是否独占，只能有一个消费者监听这队列
                  当connection关闭时，是否删除队列
          4.autoDelete：是否自动删除
          5.arguments：其他参数
         */
        //创建队列，没有才会创建
        String queue1Name = "test_topic_queue1";
        String queue2Name = "test_topic_queue2";
        channel.queueDeclare(queue1Name, true, false, false, null);
        channel.queueDeclare(queue2Name, true, false, false, null);
        //7.绑定队列和交换机
        /*
        queueBind(String queue, String exchange, String routingKey)
        参数：
            1.queue：队列名称
            2.exchange：交换机名称
            3.routingKey：路由键，绑定规则
              如果交换机绑定的是fanout，设置为""
         */
        //所有error级别的日志存入数据库，所有order系统的日志存入数据库
        channel.queueBind(queue1Name,exchangeName,"#.error");
        channel.queueBind(queue1Name,exchangeName,"order.*");
        //打印所有信息
        channel.queueBind(queue2Name,exchangeName,"*.*");
        /*
         basicPublish(String exchange, String routingKey, BasicProperties props, byte[] body)
           1.exchange: 交换机名称。简单模式会使用默认的""
           2.routingKey：路由键名称要和对应的队列名称一样才能路由到
           3.props：配置信息
           4.body：主要发送体
         */
        //8.发送消息
        String body = "日志信息：张三调用了findAll方法。。报错。日志级别：error。。。";
        channel.basicPublish(exchangeName, "goods.error", null, body.getBytes(StandardCharsets.UTF_8));
        //9.释放资源
        channel.close();
```

```
86              connection.close();
87
88          }
89
    }
```

Java | 复制代码

```java
/**
 *  * 通配符模式-接收消息1
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/24 9:57
 */
public class ConsumerTopic1 {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        String queue1Name = "test_topic_queue1";
        /*
         basicConsume(String queue,boolean autoAck,Consumer consumer)
           1.queue: 队列名称。
           2.autoAck: 是否自动确认
           3.consumer: 回调对象
         */
        //6.接收消息
        Consumer consumer =new DefaultConsumer(channel){
            @Override
            public void handleDelivery(String s, Envelope envelope, AMQP.BasicProperties basicProperties, byte[] body) {
                System.out.println("body:"+new String(body));
                System.out.println("将日志保存至数据库");
            }
        };
        channel.basicConsume(queue1Name,true,consumer);

```

```
44          }
45
46      }
```

Java | 复制代码

```java
/**
 *  * 通配符模式-接收消息2
 *
 * @author zhichao.jiang01@hand-china.com 2022/10/24 9:57
 */
public class ConsumerTopic2 {

    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory connectionFactory = new ConnectionFactory();
        //2.设置参数
        // ip 默认localhost
        connectionFactory.setHost("172.26.41.194");
        // amqp端口默认5672
        connectionFactory.setPort(5672);
        // 虚拟机名称 默认/
        connectionFactory.setVirtualHost("/hzero");
        // 用户 默认guest
        connectionFactory.setUsername("admin");
        // 密码 默认guest
        connectionFactory.setPassword("admin");
        //3.创建连接 Connection
        Connection connection =connectionFactory.newConnection();
        //4.创建Channel
        Channel channel =connection.createChannel();
        String queue2Name = "test_topic_queue2";
        /*
         basicConsume(String queue,boolean autoAck,Consumer consumer)
           1.queue: 队列名称。
           2.autoAck: 是否自动确认
           3.consumer: 回调对象
         */
        //6.接收消息
        Consumer consumer =new DefaultConsumer(channel){
            @Override
            public void handleDelivery(String s, Envelope envelope, AMQP.BasicProperties basicProperties, byte[] body) {
                System.out.println("body:"+new String(body));
                System.out.println("将日志打印到控制台");
            }
        };
        channel.basicConsume(queue2Name,true,consumer);

```

```
44          }
45
46      }
```

Topic 主题模式可以实现 Pub/Sub 发布与订阅模式和 Routing 路由模式的功能，只是 Topic 在配置routing key 的时候可以使用通配符，显得更加灵活。