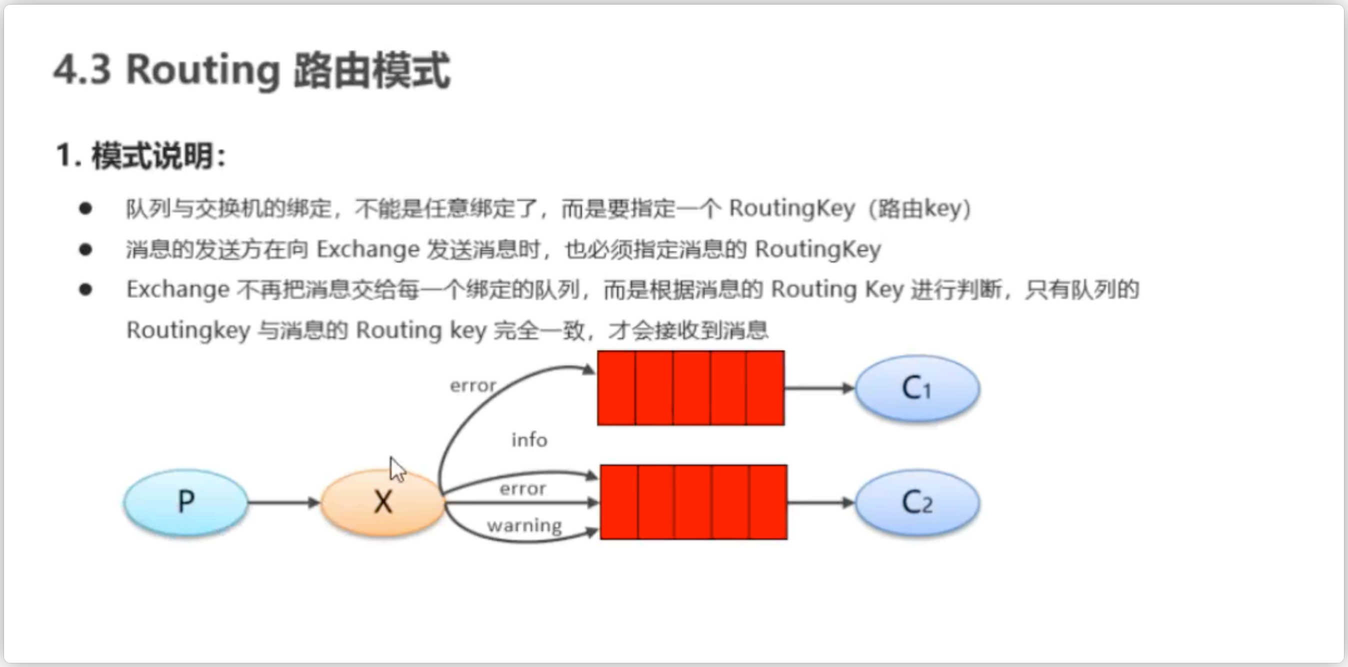
路由模式



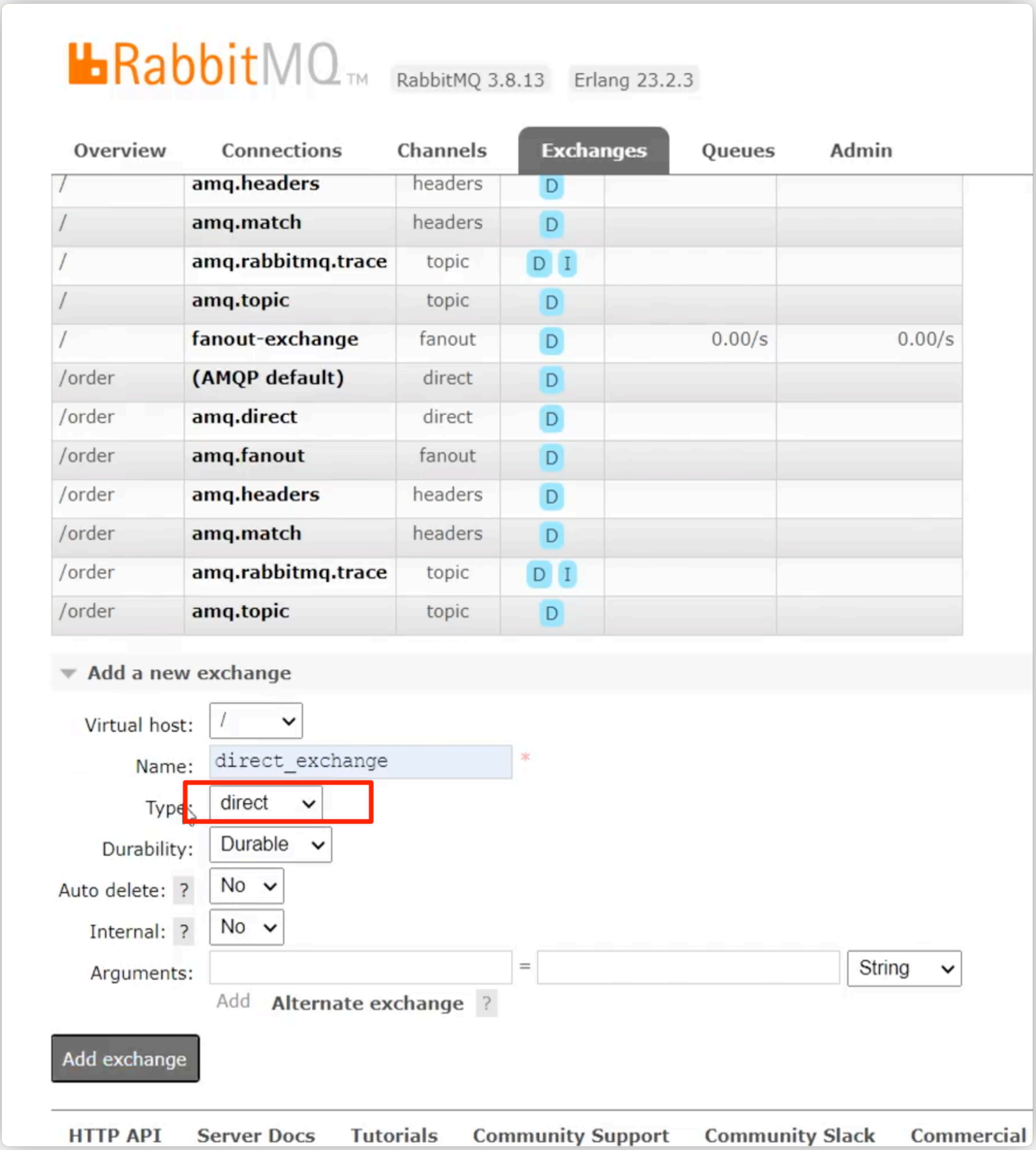
4.3 Routing 路由模式

1. 模式说明：

- 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

类似朋友圈指定人可见

界面模拟



RabbitMQ TM RabbitMQ 3.8.13 Erlang 23.2.3

Overview Connections Channels **Exchanges** Queues Admin

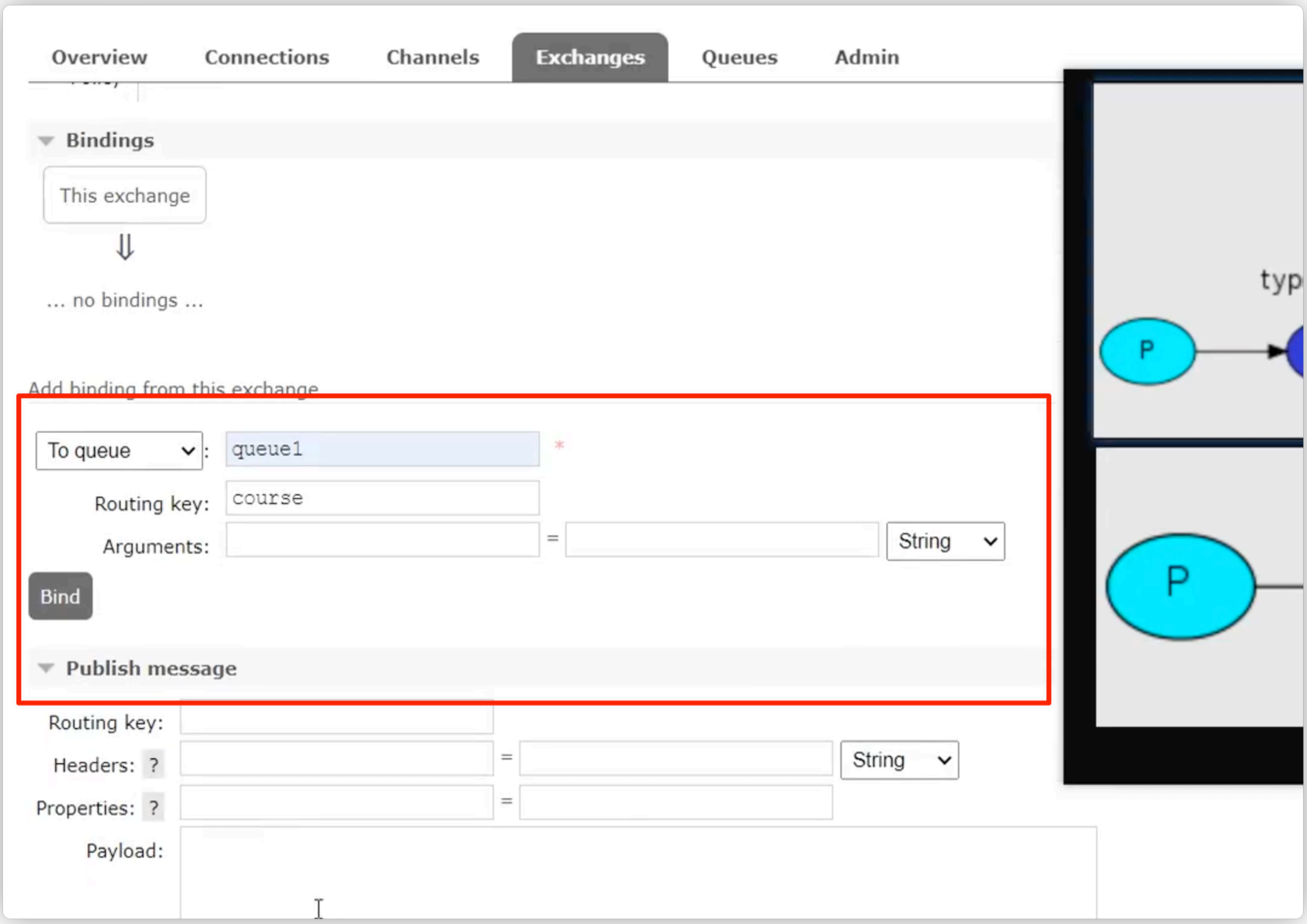
/	**amq.headers**	headers	D		
/	**amq.match**	headers	D		
/	**amq.rabbitmq.trace**	topic	D I		
/	**amq.topic**	topic	D		
/	**fanout-exchange**	fanout	D	0.00/s	0.00/s
/order	**(AMQP default)**	direct	D		
/order	**amq.direct**	direct	D		
/order	**amq.fanout**	fanout	D		
/order	**amq.headers**	headers	D		
/order	**amq.match**	headers	D		
/order	**amq.rabbitmq.trace**	topic	D I		
/order	**amq.topic**	topic	D		

Add a new exchange

Virtual host:	/
Name:	direct_exchange *
Type:	direct
Durability:	Durable
Auto delete: ?	No
Internal: ?	No
Arguments:	= String
	Add **Alternate exchange** ?

Add exchange

HTTP API Server Docs Tutorials Community Support Community Slack Commercial



Bindings

This exchange

⇓

… no bindings …

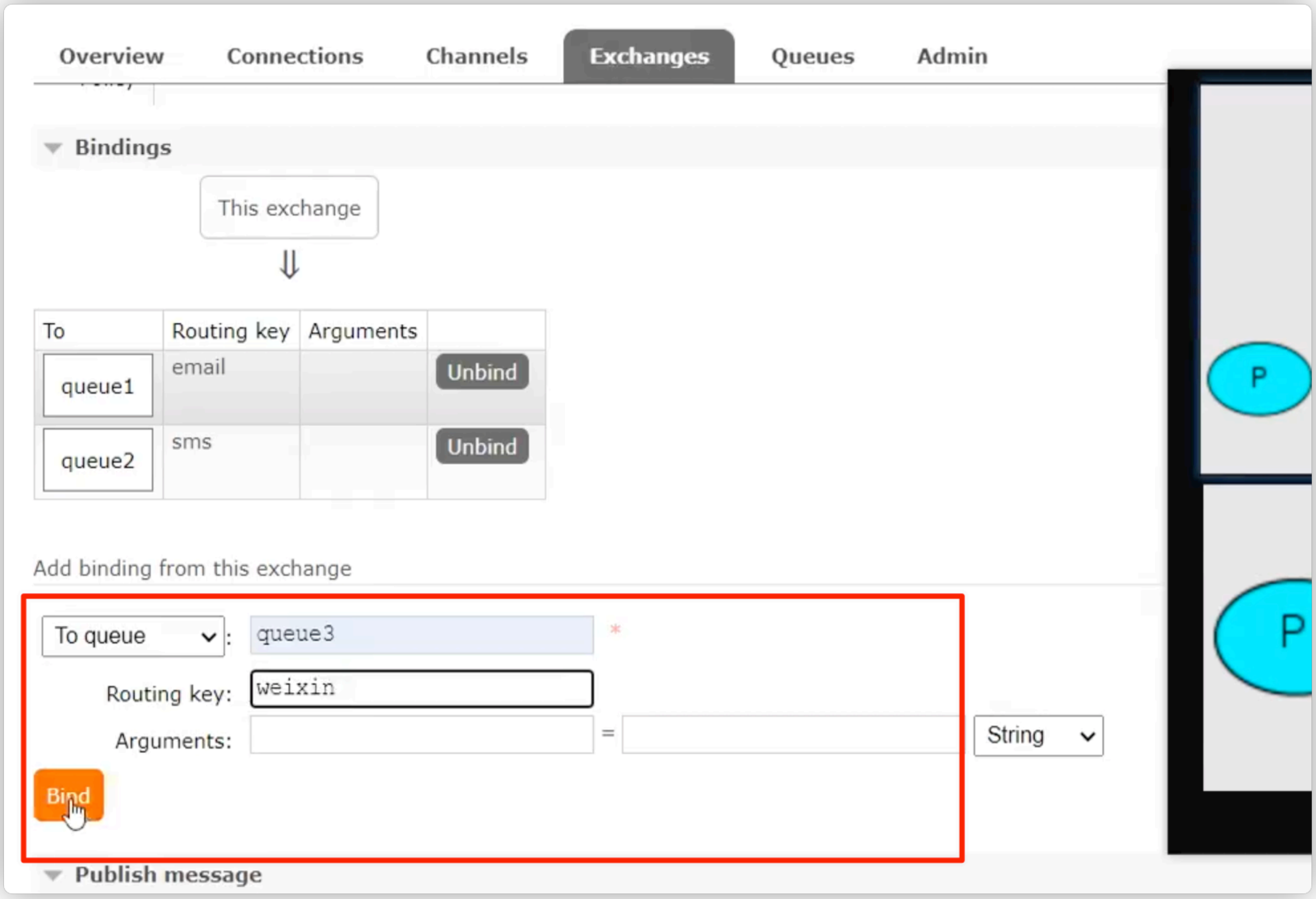
Add binding from this exchange

To queue:	queue1
Routing key:	course
Arguments:	= String

Bind

Publish message

Routing key:	
Headers: ?	= String
Properties: ?	=
Payload:	



Bindings

This exchange

⇓

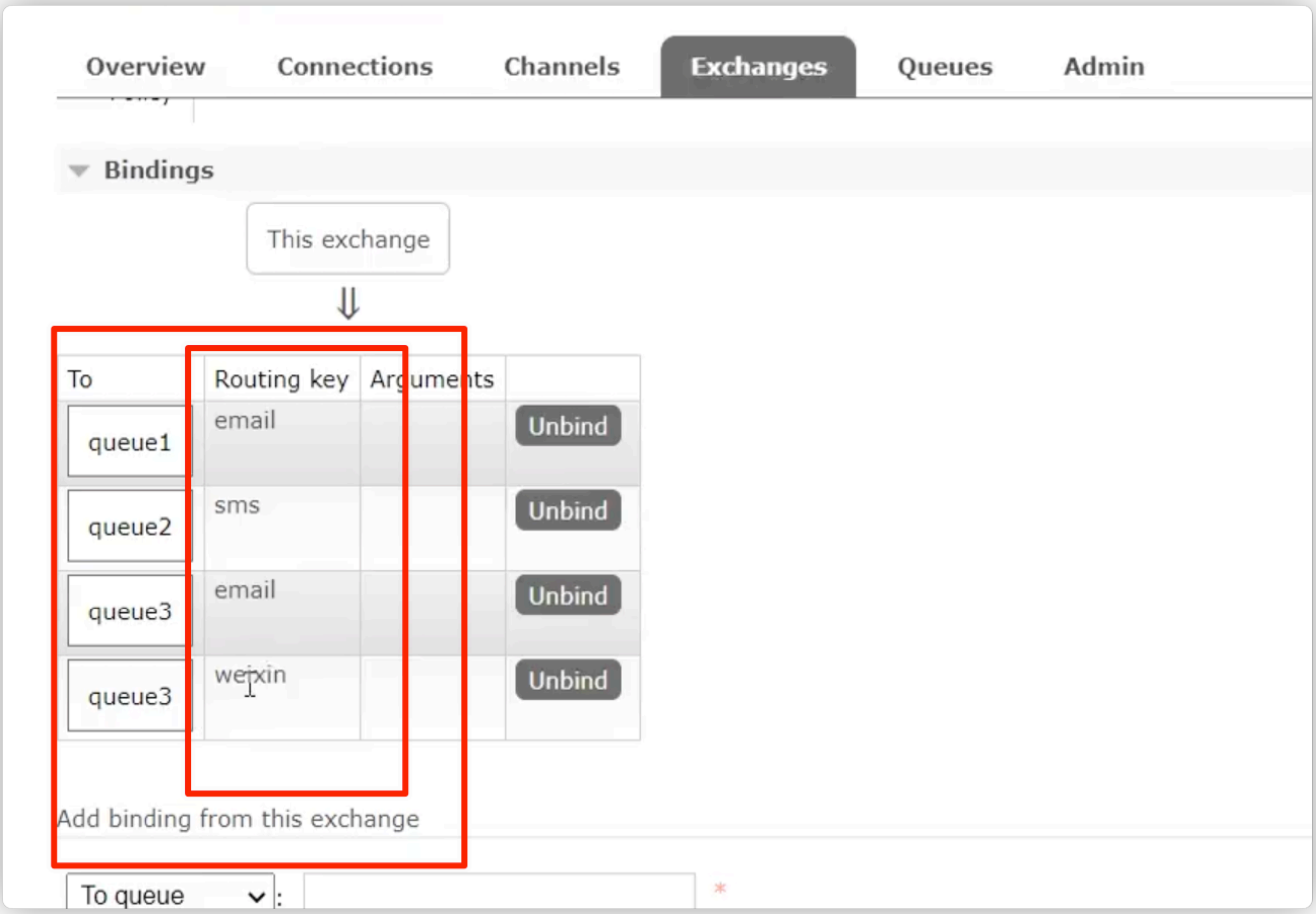
To	Routing key	Arguments	
queue1	email		Unbind
queue2	sms		Unbind

Add binding from this exchange

To queue:	queue3
Routing key:	weixin
Arguments:	= String

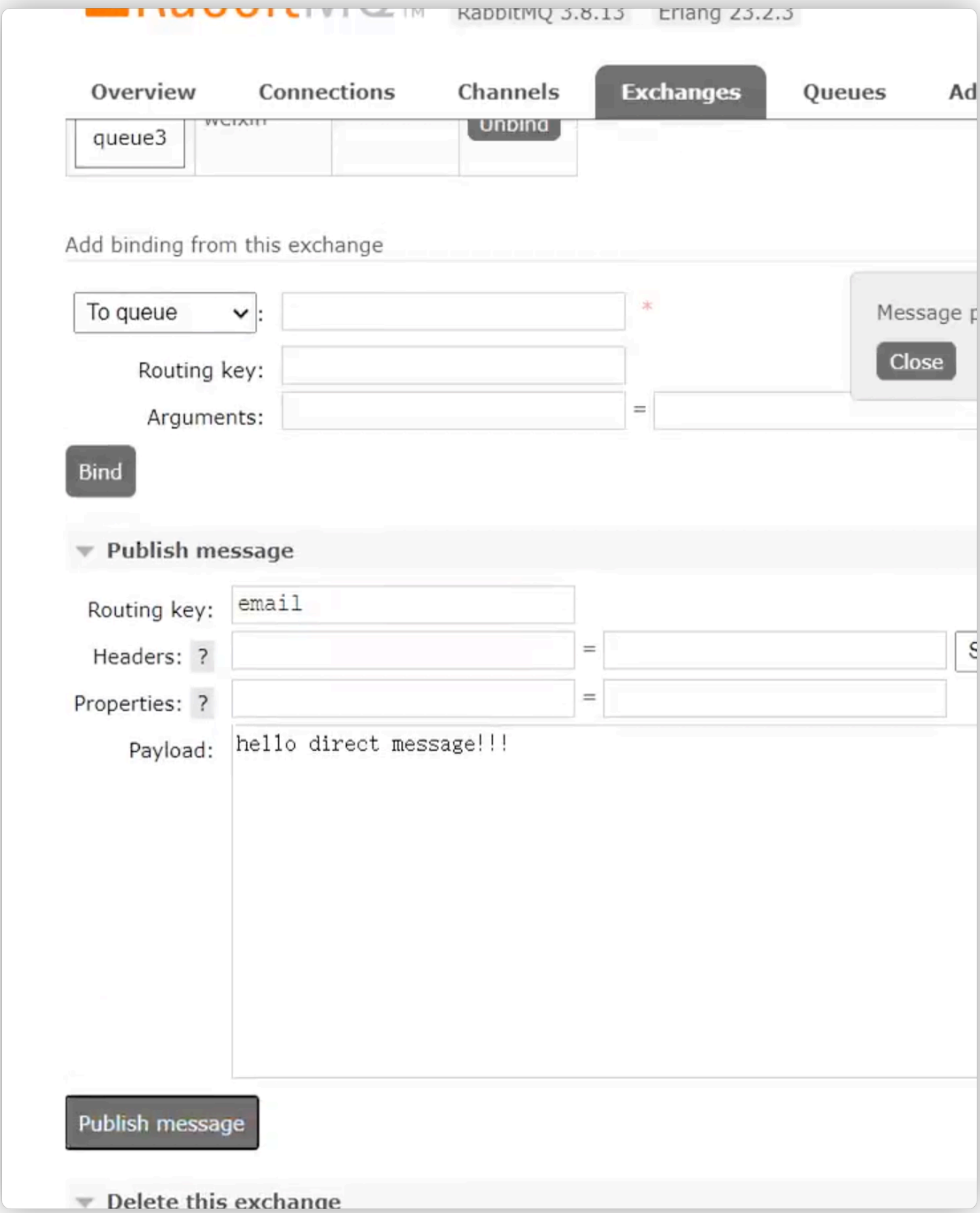
Bind

Publish message



To	Routing key	Arguments	
queue1	email		Unbind
queue2	sms		Unbind
queue3	email		Unbind
queue3	weixin		Unbind

发送消息



Add binding from this exchange

To queue:

Routing key:

Arguments: =

Publish message

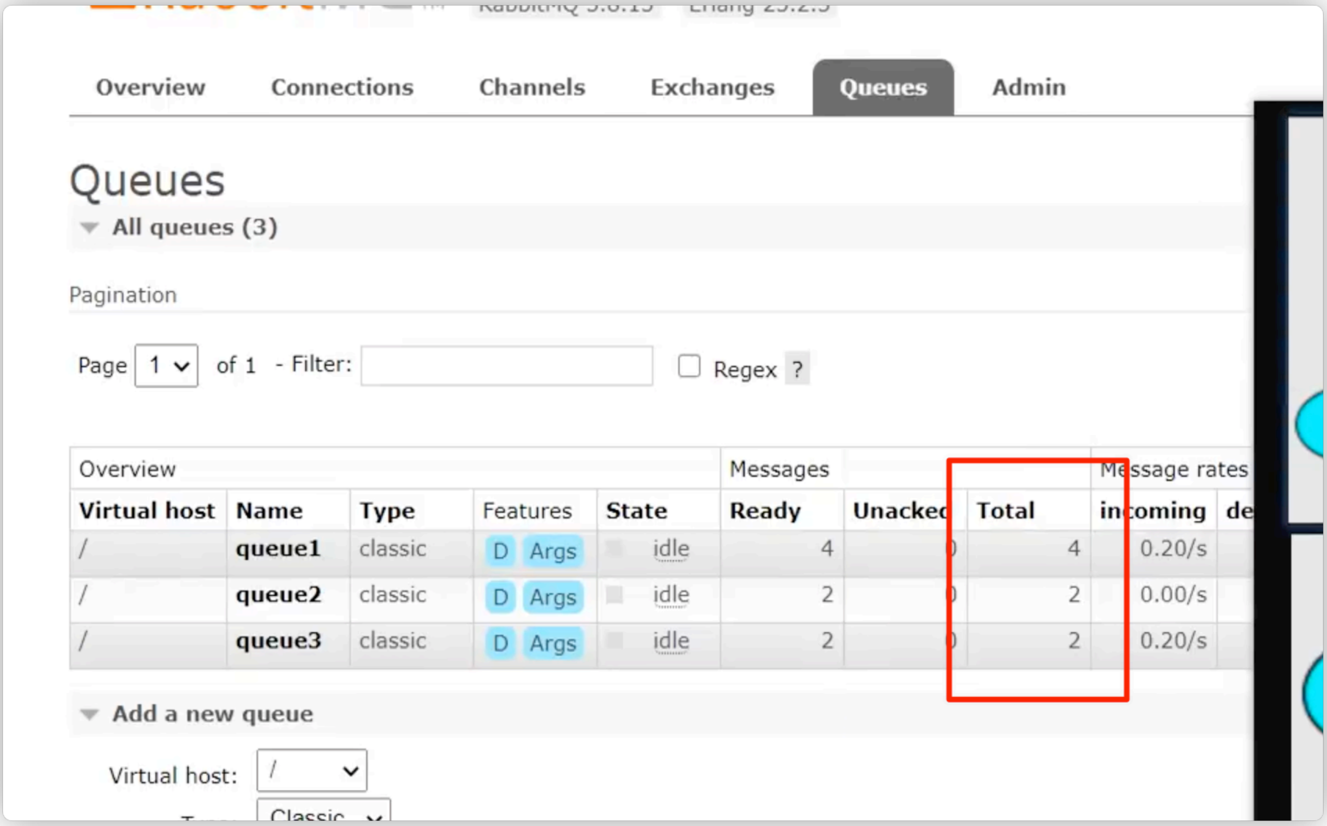
Routing key: email

Headers: ? =

Properties: ? =

Payload: hello direct message!!!

队列查看消息增加记录


Overview Connections Channels Exchanges Queues Admin

Queues

All queues (3)

Pagination

Page 1 of 1 - Filter: Regex ?

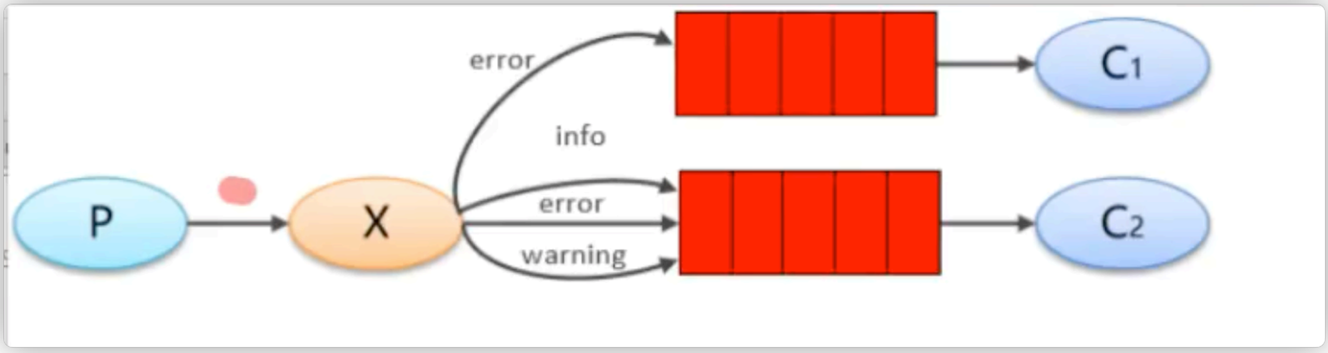
Overview					Messages			Message rates	
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	de
/	queue1	classic	D Args	idle	4	0	4	0.20/s	
/	queue2	classic	D Args	idle	2	0	2	0.00/s	
/	queue3	classic	D Args	idle	2	0	2	0.20/s	

Add a new queue

Virtual host: /

Classic

代码



Producer_Routing

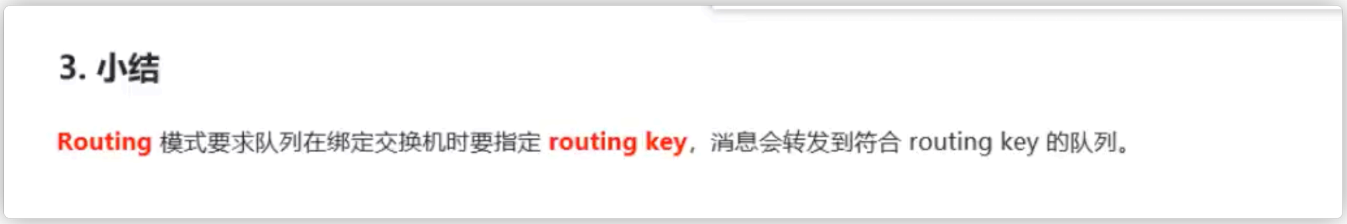
```
package com.itheima.producer;

import com.rabbitmq.client.BuiltinExchangeType;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;

import java.io.IOException;
import java.util.concurrent.TimeoutException;

/**
 * 发送消息
 */
public class Producer_Routing {
    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory factory = new ConnectionFactory();
        //2. 设置参数
        factory.setHost("172.16.98.133");//ip  默认值 localhost
        factory.setPort(5672); //端口  默认值 5672
        factory.setVirtualHost("/itcast");//虚拟机 默认值/
        factory.setUsername("heima");//用户名 默认 guest
        factory.setPassword("heima");//密码 默认值 guest
        //3. 创建连接 Connection
        Connection connection = factory.newConnection();
        //4. 创建Channel
        Channel channel = connection.createChannel();
       /*

       exchangeDeclare(String exchange, BuiltinExchangeType type, boolean durable, boolean
autoDelete, boolean internal, Map<String, Object> arguments)
       参数：
        1. exchange:交换机名称
        2. type:交换机类型
            DIRECT("direct"),：定向
            FANOUT("fanout"),：扇形（广播），发送消息到每一个与之绑定队列。
            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false
        6. arguments：参数
        */

       String exchangeName = "test_direct";
        //5. 创建交换机
        channel.exchangeDeclare(exchangeName, BuiltinExchangeType.DIRECT,true,false,false,null);
        //6. 创建队列
        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
                如果交换机的类型为fanout ，routingKey设置为""
         */
        //队列1绑定 error
        channel.queueBind(queue1Name,exchangeName,"error");
        //队列2绑定 info  error  warning
        channel.queueBind(queue2Name,exchangeName,"info");
        channel.queueBind(queue2Name,exchangeName,"error");
        channel.queueBind(queue2Name,exchangeName,"warning");

        String body = "日志信息：张三调用了delete方法...出错误了。。。日志级别：error...";
        //8. 发送消息
        channel.basicPublish(exchangeName,"warning",null,body.getBytes());

        //9. 释放资源
        channel.close();
        connection.close();

    }
}
```

Consumer_Routing1

```
package com.itheima.consumer;

import com.rabbitmq.client.*;

import java.io.IOException;
import java.util.concurrent.TimeoutException;

public class Consumer_Routing1 {
    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory factory = new ConnectionFactory();
        //2. 设置参数
        factory.setHost("172.16.98.133");//ip  默认值 localhost
        factory.setPort(5672); //端口  默认值 5672
        factory.setVirtualHost("/itcast");//虚拟机 默认值/
        factory.setUsername("heima");//用户名 默认 guest
        factory.setPassword("heima");//密码 默认值 guest
        //3. 创建连接 Connection
        Connection connection = factory.newConnection();
        //4. 创建Channel
        Channel channel = connection.createChannel();


        String queue1Name = "test_direct_queue1";
        String queue2Name = "test_direct_queue2";

        /*
        basicConsume(String queue, boolean autoAck, Consumer callback)
        参数：
            1. queue：队列名称
            2. autoAck：是否自动确认
            3. callback：回调对象

         */
        // 接收消息
        Consumer consumer = new DefaultConsumer(channel){
            /*
                回调方法，当收到消息后，会自动执行该方法

                1. consumerTag：标识
                2. envelope：获取一些信息，交换机，路由key...
                3. properties:配置信息
                4. body：数据

             */
            @Override
            public void handleDelivery(String consumerTag, Envelope envelope,
AMQP.BasicProperties properties, byte[] body) throws IOException {
              /*  System.out.println("consumerTag："+consumerTag);
                System.out.println("Exchange："+envelope.getExchange());
                System.out.println("RoutingKey："+envelope.getRoutingKey());
                System.out.println("properties："+properties);*/
                System.out.println("body："+new String(body));
                System.out.println("将日志信息打印到控制台.....");
            }
        };
        channel.basicConsume(queue2Name,true,consumer);


        //关闭资源？不要

    }
}
```

Consumer_Routing2

```
package com.itheima.consumer;

import com.rabbitmq.client.*;

import java.io.IOException;
import java.util.concurrent.TimeoutException;

public class Consumer_Routing2 {
    public static void main(String[] args) throws IOException, TimeoutException {

        //1.创建连接工厂
        ConnectionFactory factory = new ConnectionFactory();
        //2. 设置参数
        factory.setHost("172.16.98.133");//ip  默认值 localhost
        factory.setPort(5672); //端口  默认值 5672
        factory.setVirtualHost("/itcast");//虚拟机 默认值/
        factory.setUsername("heima");//用户名 默认 guest
        factory.setPassword("heima");//密码 默认值 guest
        //3. 创建连接 Connection
        Connection connection = factory.newConnection();
        //4. 创建Channel
        Channel channel = connection.createChannel();


        String queue1Name = "test_direct_queue1";
        String queue2Name = "test_direct_queue2";


        /*
        basicConsume(String queue, boolean autoAck, Consumer callback)
        参数:
            1. queue: 队列名称
            2. autoAck: 是否自动确认
            3. callback: 回调对象

         */
        // 接收消息
        Consumer consumer = new DefaultConsumer(channel){
            /*
                回调方法，当收到消息后，会自动执行该方法

                1. consumerTag: 标识
                2. envelope: 获取一些信息，交换机，路由key...
                3. properties:配置信息
                4. body: 数据

             */
            @Override
            public void handleDelivery(String consumerTag, Envelope envelope,
AMQP.BasicProperties properties, byte[] body) throws IOException {
              /*  System.out.println("consumerTag: "+consumerTag);
                System.out.println("Exchange: "+envelope.getExchange());
                System.out.println("RoutingKey: "+envelope.getRoutingKey());
                System.out.println("properties: "+properties);*/
                System.out.println("body: "+new String(body));
                System.out.println("将日志信息存储到数据库.....");
            }
        };
        channel.basicConsume(queue1Name,true,consumer);
        //关闭资源? 不要

    }
}
```



3. 小结

Routing 模式要求队列在绑定交换机时要指定 **routing key**，消息会转发到符合 routing key 的队列。