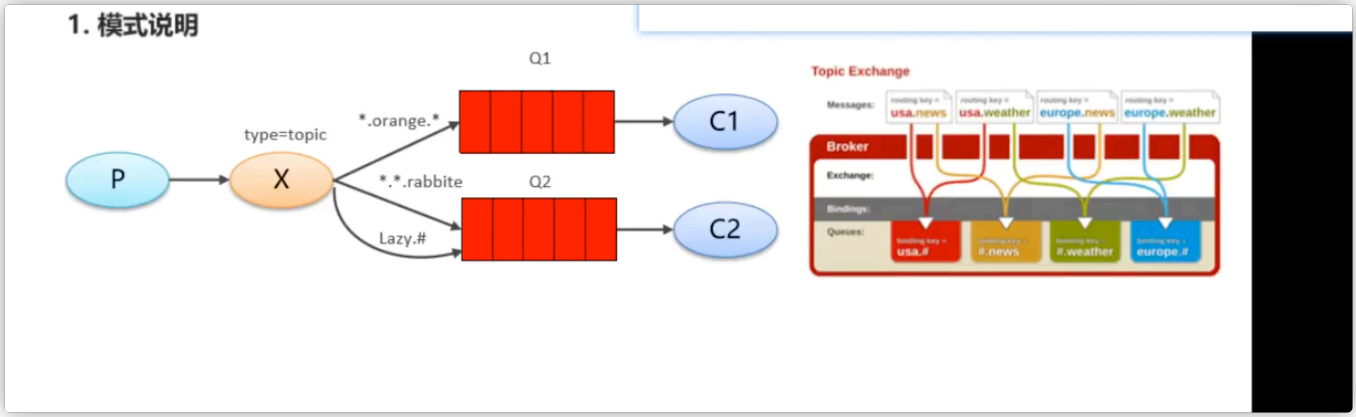
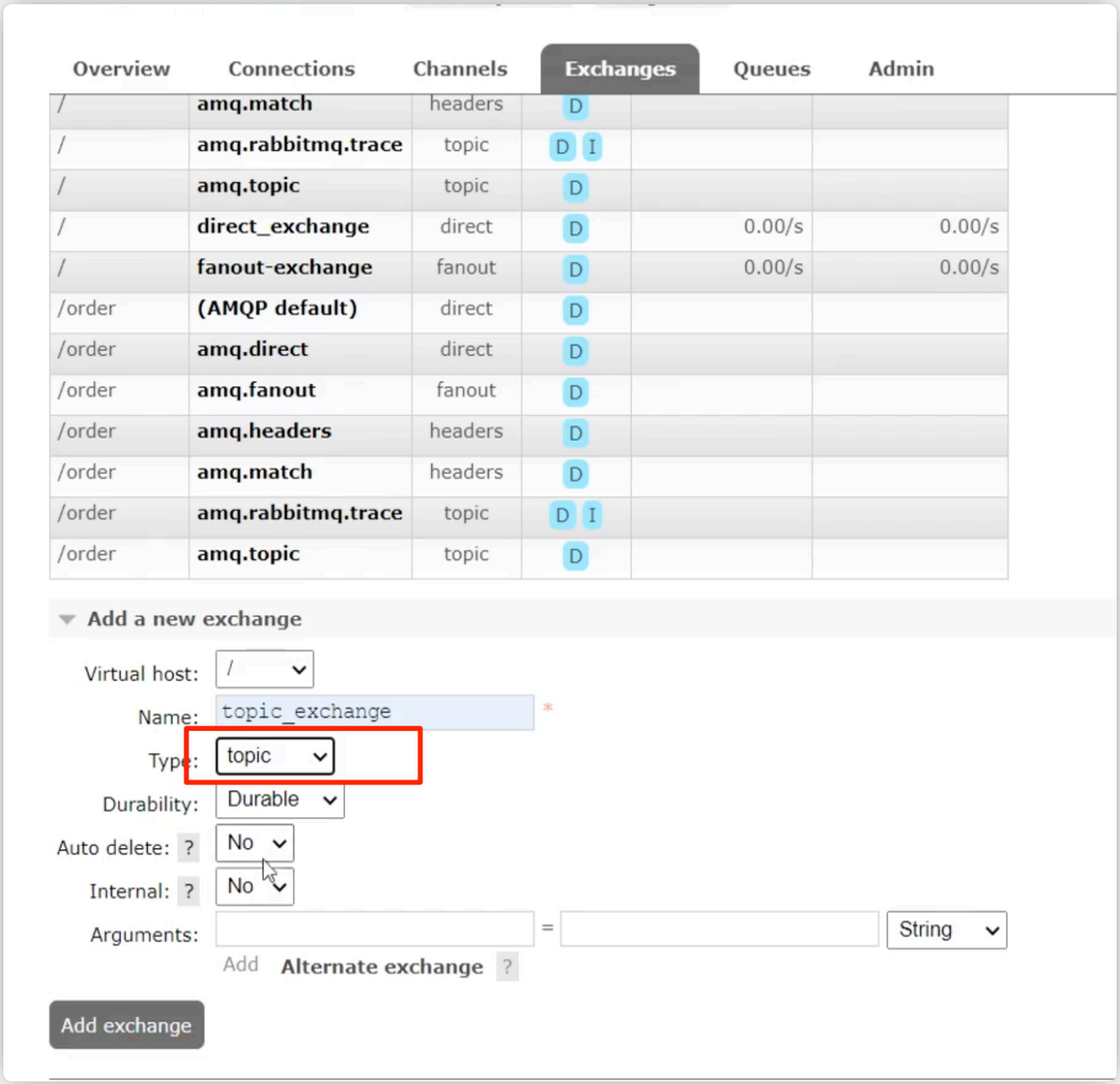
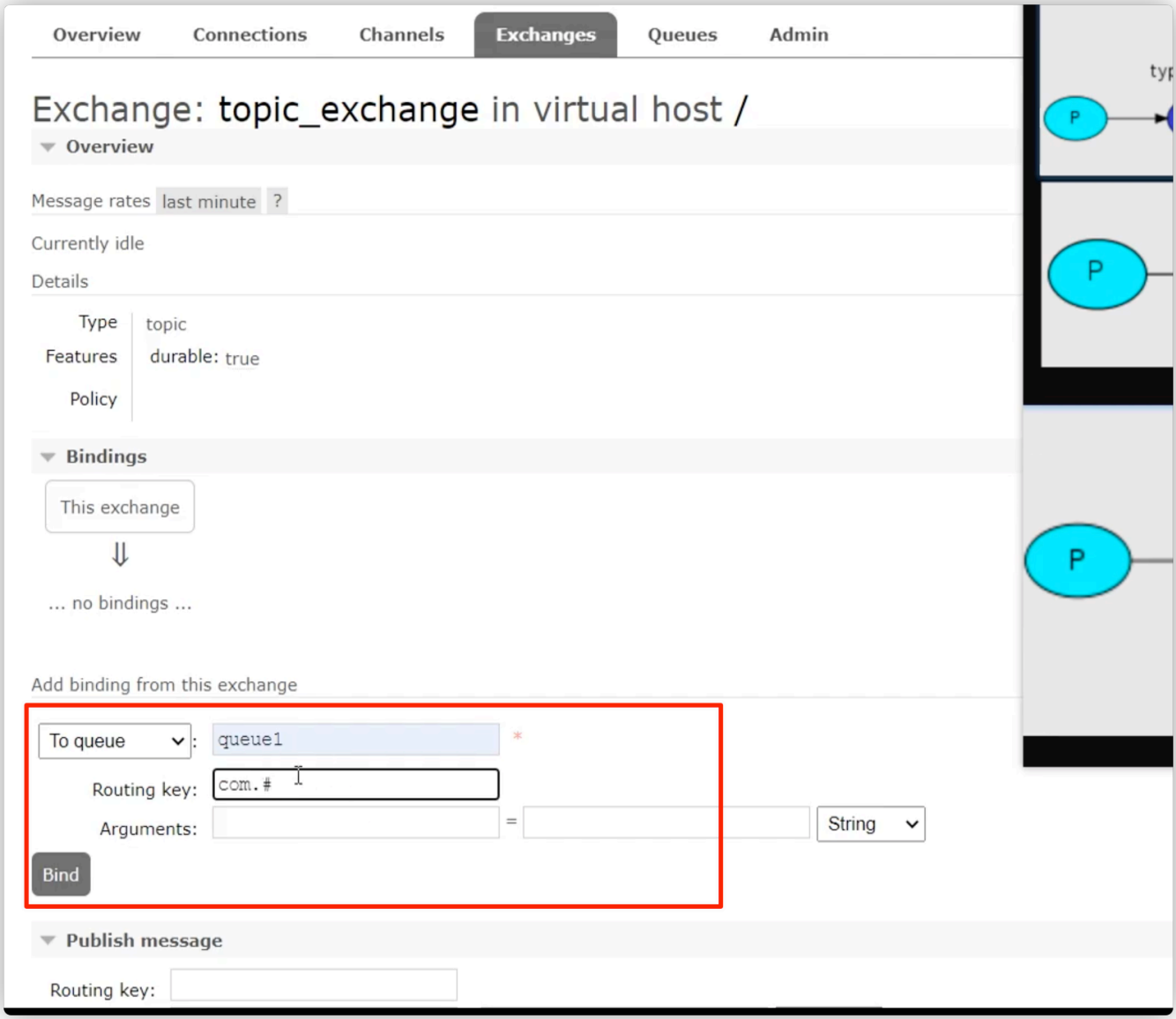
topic模式

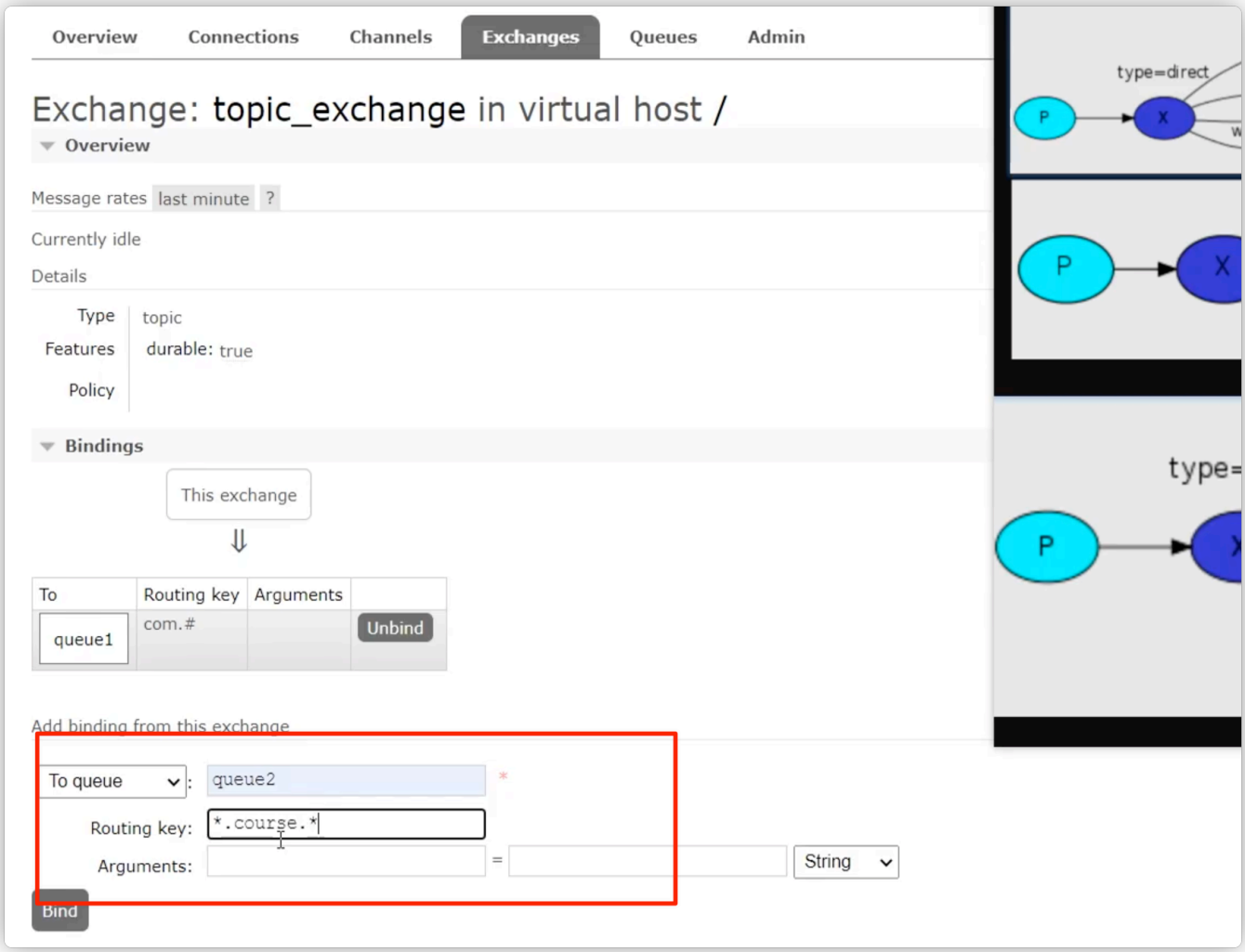


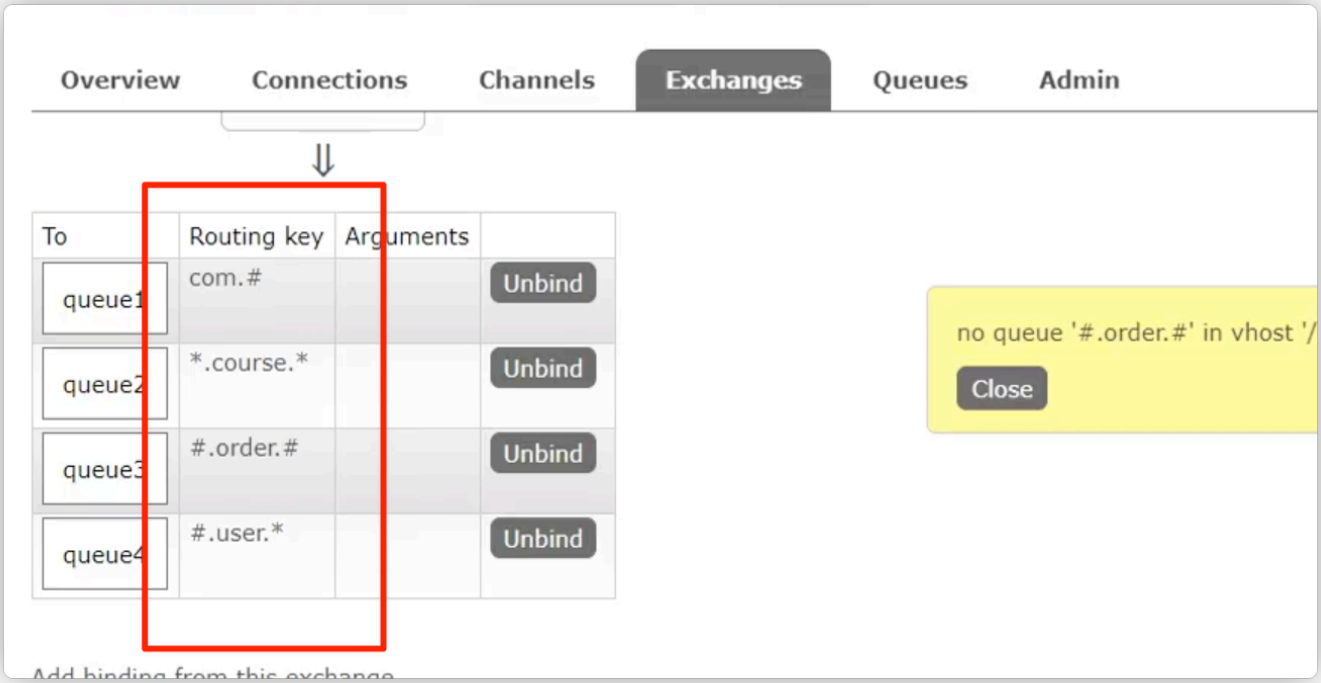
* 代表一个单词
# 代表0到多个单词

界面模拟：

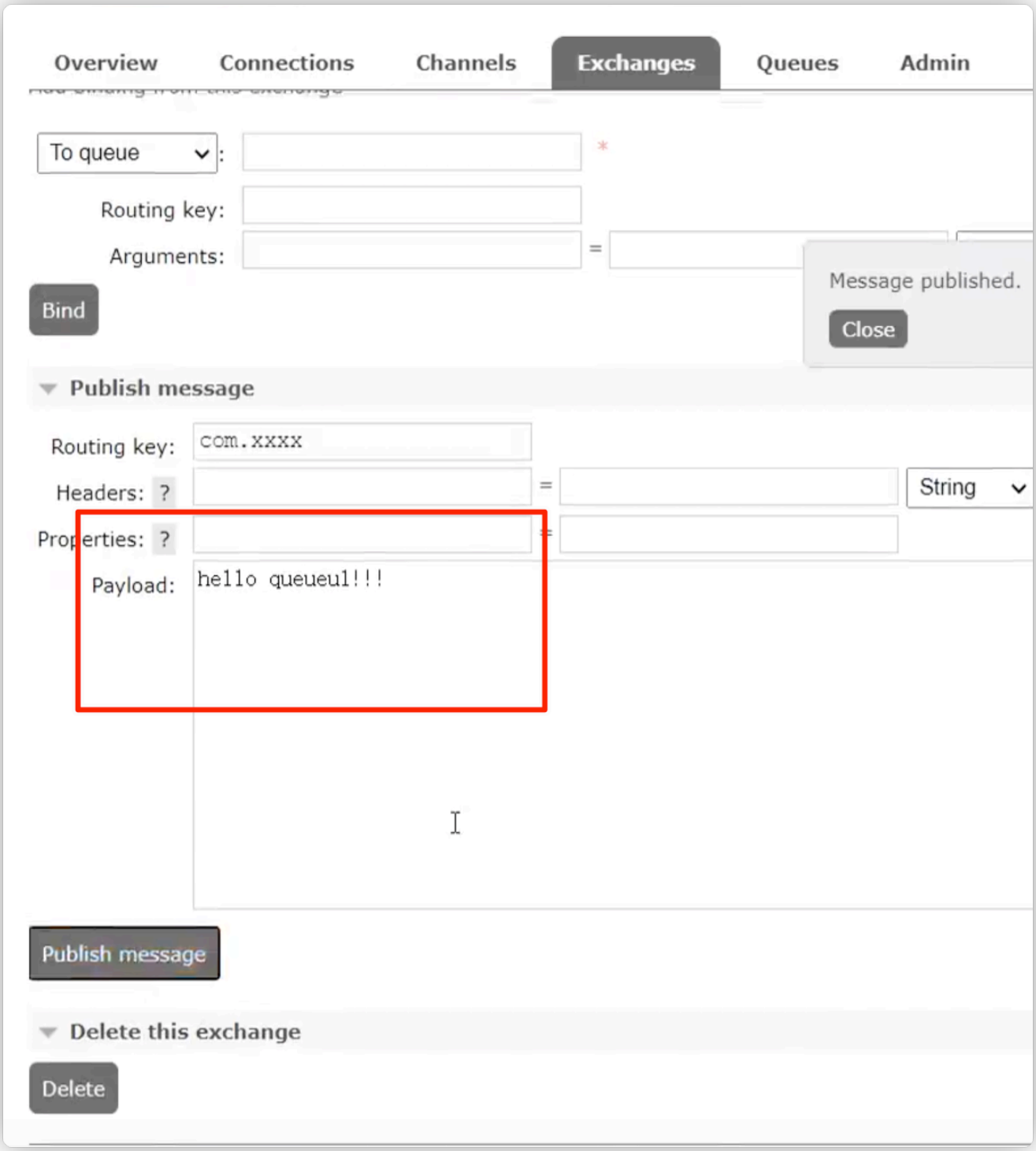

Overview
Connections
Channels
Exchanges
Queues
Admin
/ amq.match headers D
/ amq.rabbitmq.trace topic D I
/ amq.topic topic D
/ direct_exchange direct D 0.00/s 0.00/s
/ fanout-exchange fanout D 0.00/s 0.00/s
/order (AMQP default) direct D
/order amq.direct direct D
/order amq.fanout fanout D
/order amq.headers 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: topic_exchange *
Type: topic
Durability: Durable
Auto delete: ? No
Internal: ? No
Arguments: = String
Add Alternate exchange ?
Add exchange


Overview
Connections
Channels
Exchanges
Queues
Admin
Exchange: topic_exchange in virtual host /
Overview
Message rates last minute ?
Currently idle
Details
Type topic
Features durable: true
Policy
Bindings
This exchange
... no bindings ...
Add binding from this exchange
To queue
queue1
Routing key:
com.#
Arguments:
String
Bind
Publish message
Routing key:
P
P
P


Overview
Connections
Channels
Exchanges
Queues
Admin
Exchange: topic_exchange in virtual host /
Overview
Message rates last minute ?
Currently idle
Details
Type topic
Features durable: true
Policy
Bindings
This exchange
To
Routing key
Arguments
queue1
com.#
Unbind
Add binding from this exchange
To queue
queue2
Routing key:
.course.
Arguments:
String
Bind
type=direct
P
X


Overview
Connections
Channels
Exchanges
Queues
Admin
To
Routing key
Arguments
queue1
com.#
Unbind
queue2
.course.
Unbind
queue3
#.order.#
Unbind
queue4
#.user.*
Unbind
no queue '#.order.#' in vhost '/
Close

发送查看情况



代码

Producer_Topics

```
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_Topics {
    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_topic";
        //5. 创建交换机
        channel.exchangeDeclare(exchangeName, BuiltinExchangeType.TOPIC,true,false,false,null);
        //6. 创建队列
        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
                如果交换机的类型为fanout , routingKey设置为""
         */

        // routing key  系统的名称.日志的级别。
        //=需求:  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.*");
        channel.queueBind(queue2Name,exchangeName,"*.*");

        String body = "日志信息:张三调用了findAll方法...日志级别:info...";
        //8. 发送消息
        channel.basicPublish(exchangeName,"goods.error",null,body.getBytes());

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

    }
}
```

Consumer_Topic1

```
package com.itheima.consumer;

import com.rabbitmq.client.*;

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

public class Consumer_Topic1 {
    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_topic_queue1";
        String queue2Name = "test_topic_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("将日志信息存入数据库.......");
            }
        };
        channel.basicConsume(queue1Name,true,consumer);


        //关闭资源? 不要

    }
}
```

Consumer_Topic2

```
package com.itheima.consumer;

import com.rabbitmq.client.*;
```

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

public class Consumer_Topic2 {
    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_topic_queue1";
        String queue2Name = "test_topic_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("将日志信息打印控制台.......");
```

```
            }
        };
        channel.basicConsume(queue2Name,true,consumer);


        //关闭资源？不要

    }
}
```

总结

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。