# SpringAMQP实现发布订阅-Direct Exchange

## 前言

上节课已经学习了SpringAMQP实现发布订阅中的第一个交换机：广播交换机Fanout Exchange，这个交换机会把消息发给与之绑定的所有队列！

## Direct Exchange路由交换机

这节课学习发布订阅中的第二个交换机：路由交换机Direct Exchange，这个交换机会将消息根据规则路由到指定的队列，注意了，不是所有的Queue，是指定的Queue！所以这种方式被官方称为路由模式。

那么问题来了，这种Direct Exchange到底把消息路由给那一个队列？这是有一定规则的！

首先每一个Queue都与Exchange设置一个BindingKey，这个BindingKey就像是一个约定的暗号一样，Queue与Exchange只要约定好，将来就按照这个安好进行一个通信！

将来publisher在发送消息时候，就也需要指定一个RoutingKey！

Exchange交换机就会将RoutingKey和BindingKey进行比对，就是对暗号！暗号对上了，即将消息了路由到对应BindingKey的路由！

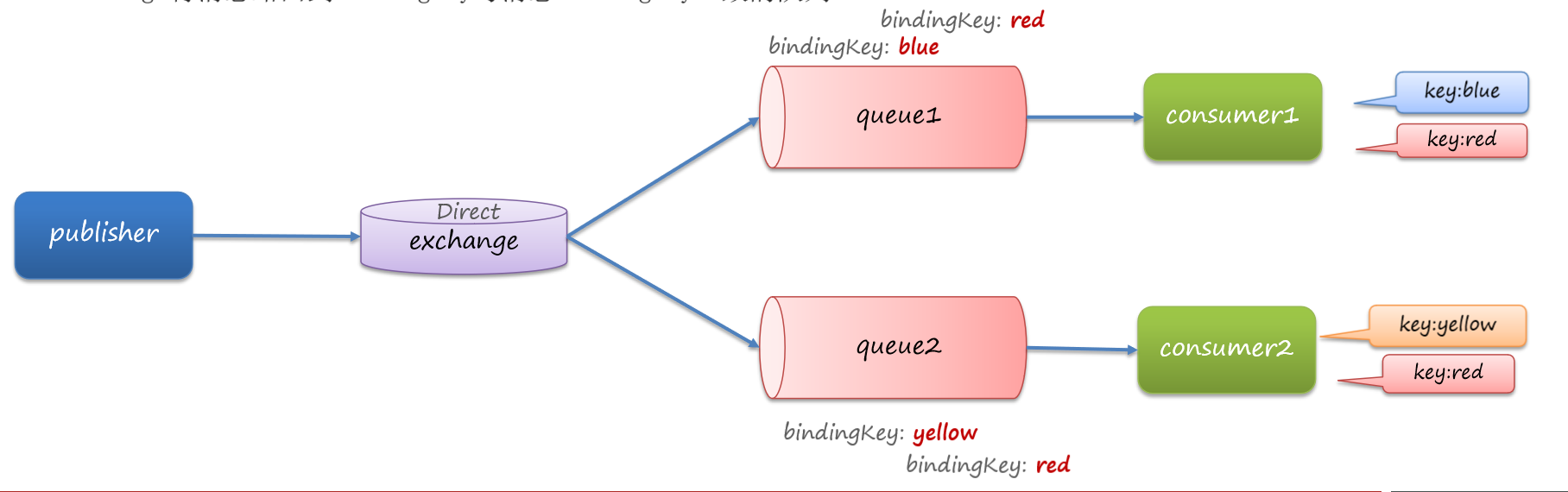
### 总结

Direct Exchange会将接收到的消息根据规则路由到指定的Queue，因此称为路由模式。

1：每一个Queue都与Exchange设置一个BindingKey

2：发布者发送消息时，指定消息的RoutingKey

3：Exchange将消息路由到BindingKey与消息RoutingKey一致的队列



### 问：

既然Queue能与Exchange绑定Key，那么不同队列能不能绑定相同的BindingKey到Exchange呢？

### 答：

是可以的，如上图所示！这时候的效果就和FanoutExchange一样了！即可以认为Direct Exchange可以模拟Fanout Exchange！Direct Exchange比Fanout Excahnge更加灵活！

这种灵活性是由代价的，即你在指定队列和交换机绑定的时候，还需要指定BindingKey，而且发布者发消息的时候，也要指定RoutingKey。

## 案例：利用SpringAMQP演示DirectExchange的使用

### 实现思路如下：

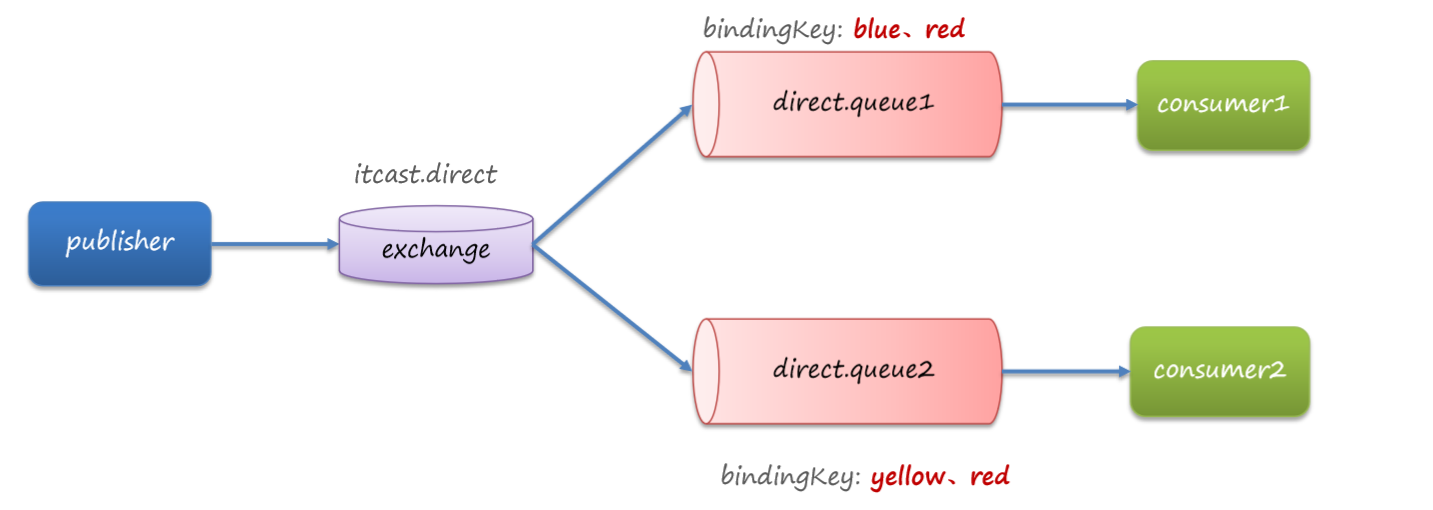
#### 1：利用@RabbitListener声明Exchange、Queue、RoutingKey;

这里不像之前案例，用@Bean去声明Exchange，Queue、Binding，因为用@Bean的方式去声明，其实有一点点复杂，你想一下，声明一个交换机Exchange，声明两个队列Queue，两个绑定关系Binding，现在的DirectExchange在绑定的时候还要指定BingdingKey，这里用@Bean至少有5个以上的Bean要去声明了！事实上用@Bean的方式去实现会更复杂，所以用@Bean的方式就太麻烦了！

学一种同样效果的方式：基于@RabbitListener注解来声明Exchange、Queue、RountingKey，在之前我们写的所有的消费者当中都用了这个@RabbitListener注解，但是仅仅用来声明消费者监听什么队列。也就是说在这个注解上还可以请完成队列、交换机、队列与交换机绑定的声明，你不用额外去创建Bean!

#### 2：在consumer服务中，编写两个消费者方法，分别监听direct.queue1和direct.queue2

#### 3：在publisher中编写测试方法，向hyh. direct发送消息

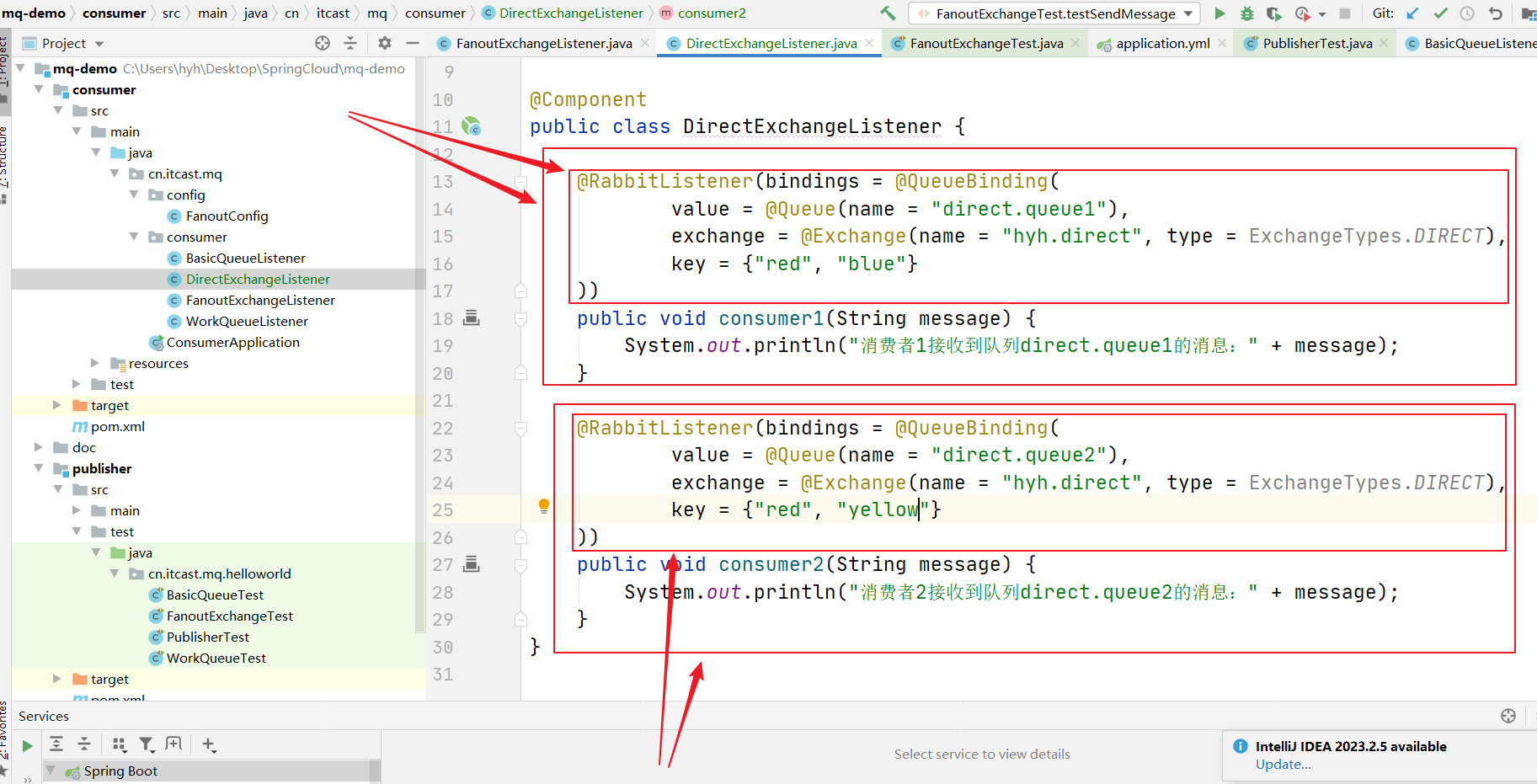


### 代码实现

#### 1：在consumer服务中编写消息监听、消费方法

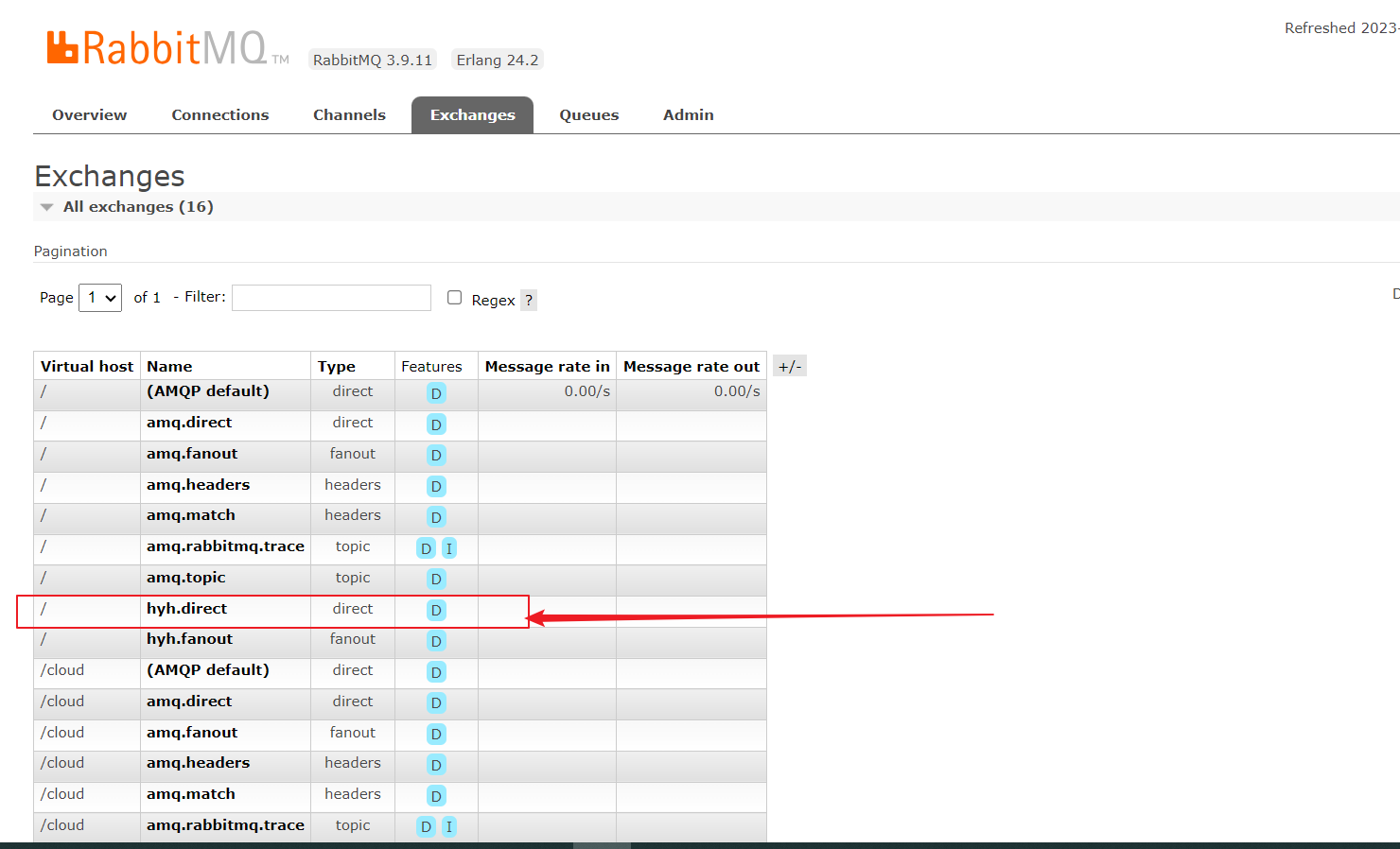
在consumer服务中，编写两个消费者方法，分别监听direct.queue1和direct.queue2，并利用@RabbitListener声明Exchange、Queue、RoutingKey。

可以在下面代码看到，在@RabbblitListener注解中，我们一次性指定/声明好了队列Queue、交换机Exchange、队列和交换机的绑定，以及队列和交换机的BindingKey！

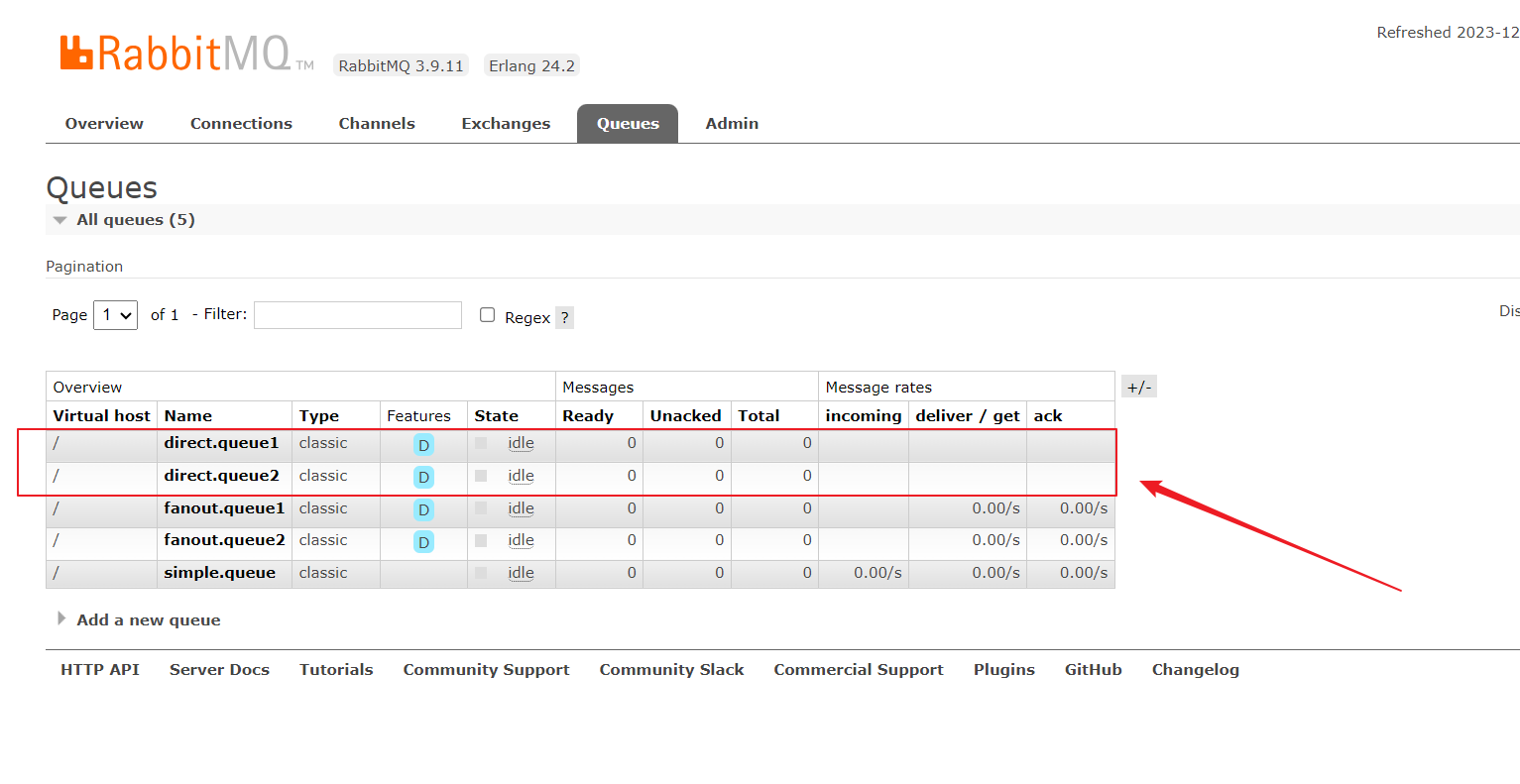


#### 2：运行Consumer，到RabbitMq管理后台查看自动创建的Exchange、Queue、Binding、BindingKey

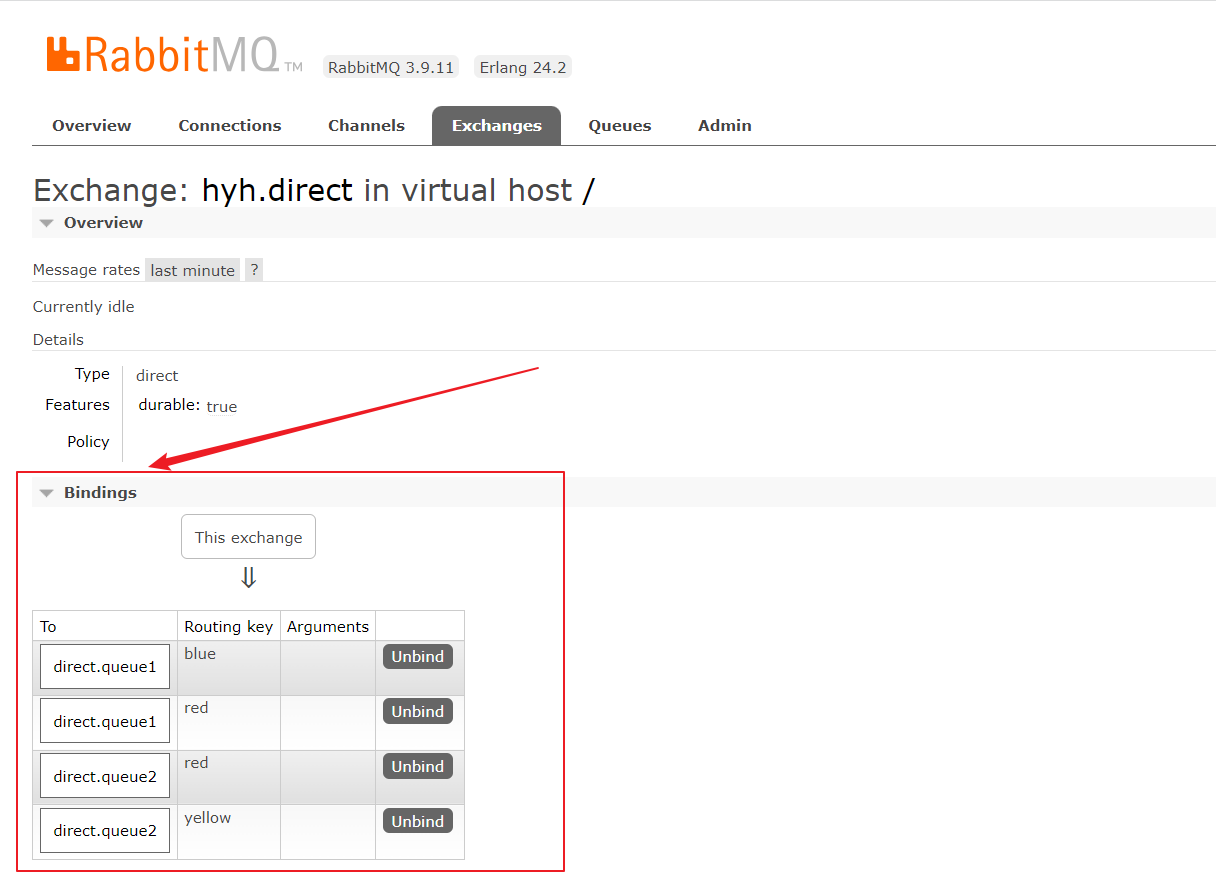
##### Excahnge



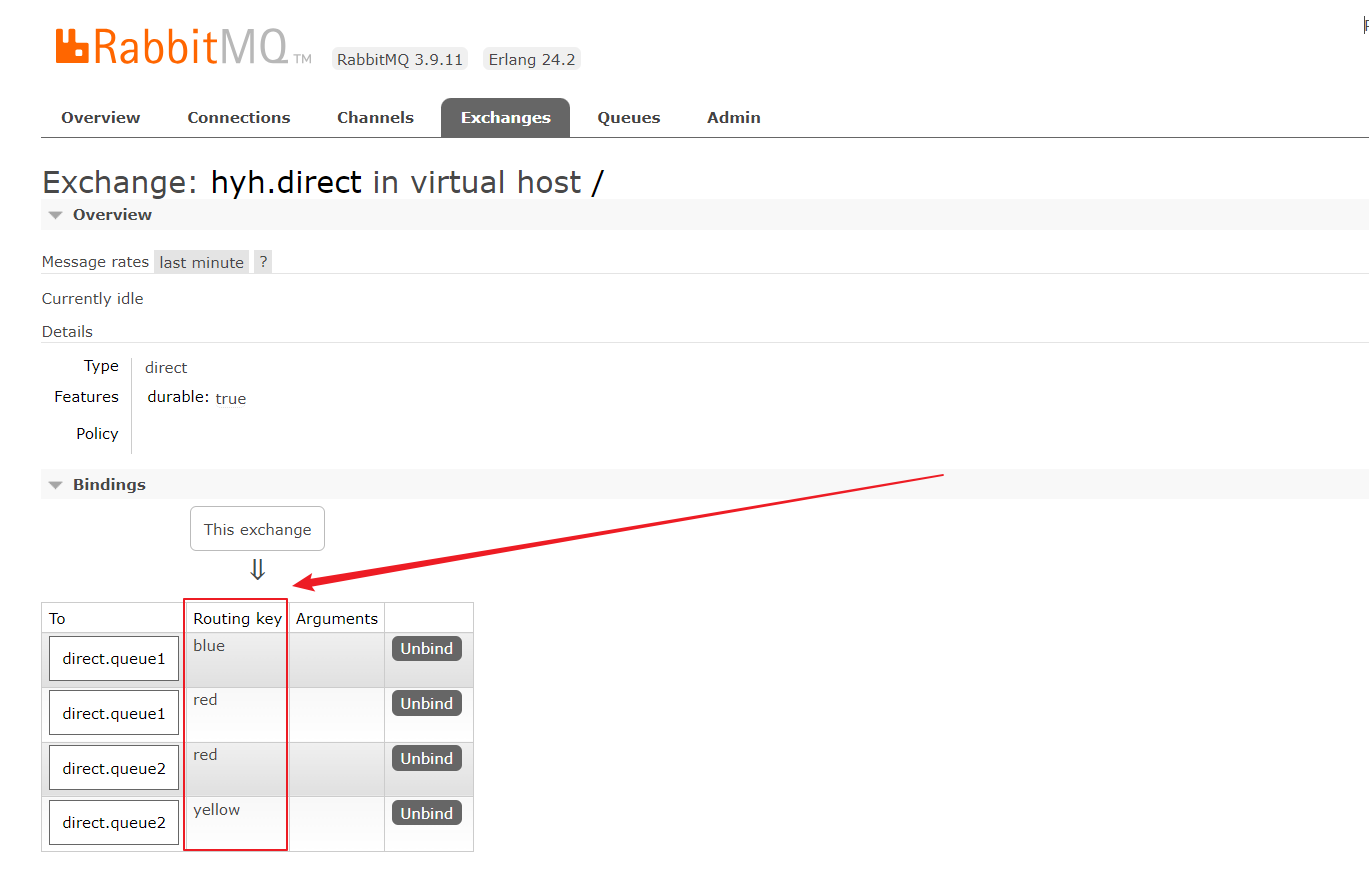
##### Queue



##### Binding



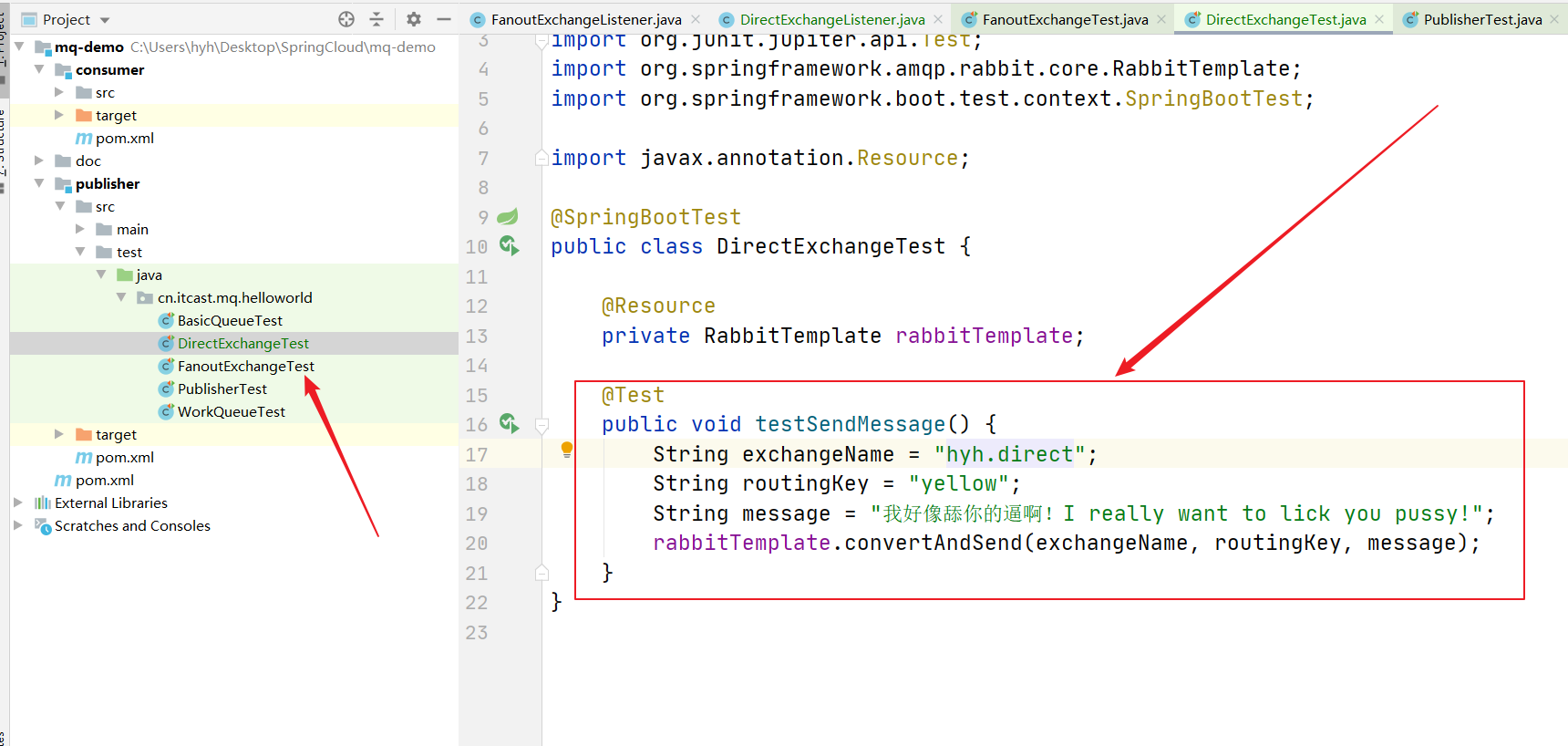
##### BindingKey



#### 3：启动consumer

#### 4：在publisher服务编写测试用例方法发送消息

**编写方法，把消息发送给hyh.direct交换机的，并且让交换机把消息路由给与交换机绑定的，并且BindingKey为yellow的队列**



#### 5：运行测试用例方法，在consumer查看消息消费日志

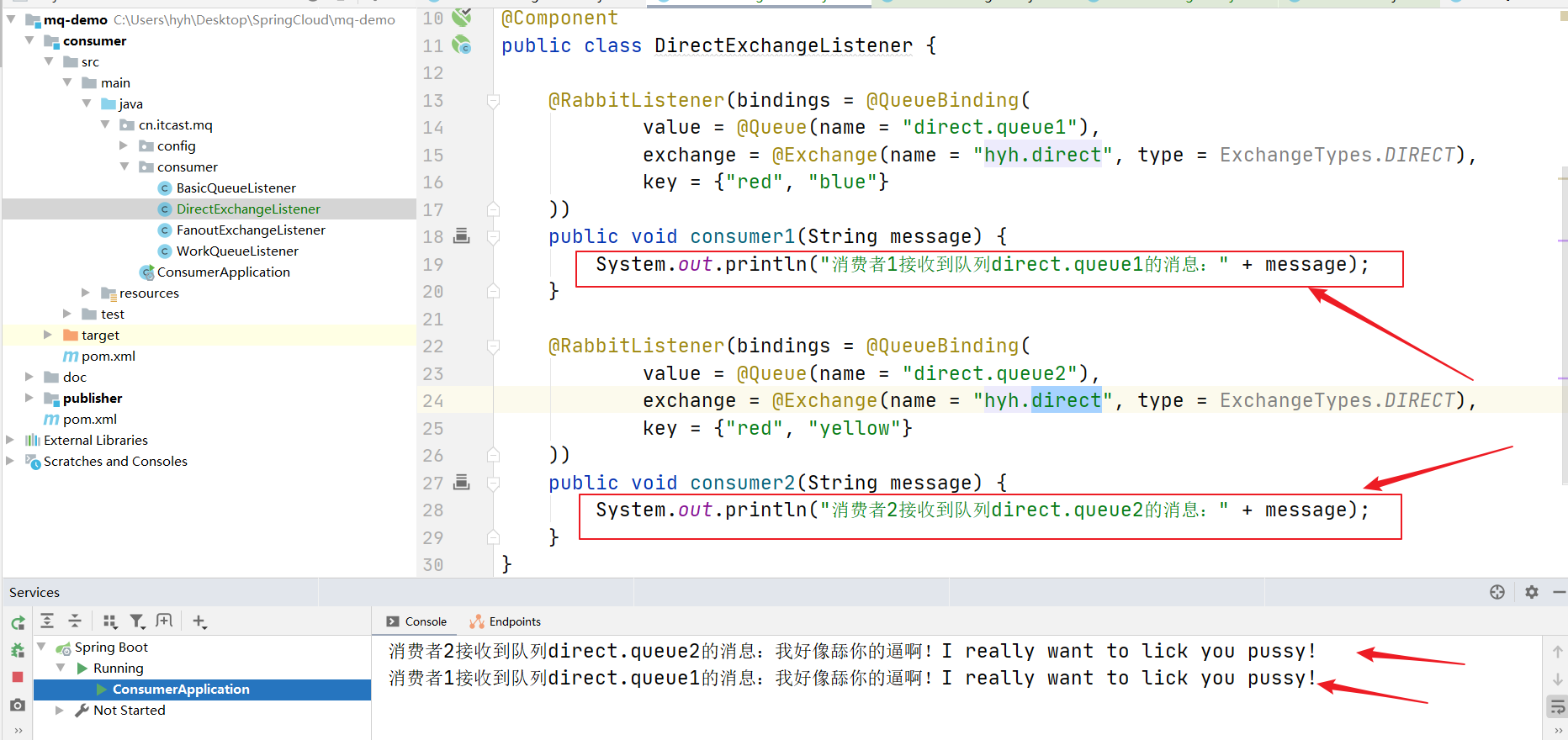
##### 1：RoutingKey为yellow运行测试用例方法

**看，消息确实被hyh.direct交换机的路由给与之绑定的，并且BindingKey为yellow的队列direct.queue2**



##### 2：RoutingKey为red运行测试用例方法

**看，消息确实被hyh.direct交换机的路由给与之绑定的，并且BindingKey为red的队列direct.queue2和direct.queue2**



这就是Direct模式的交换机可以模拟Fanout的交换机，但是Direct类型的交换机更加灵活，Direct交换机可以发给指定的Queue，而Fanout是发送给所有与之绑定的Queue！

## 总结：

### 1：发布订阅模型之：路由交换机（Direct Exchange）

Direct Exchange会将接收到的消息根据规则路由到指定的queue，因此称为路由模式（routes）

1：每一个queue都与Exchange设置一个BindingKey；

2：发布者发送消息时，指定消息的RoutingKey；

3：Exchange将消息路由到BindingKey与消息RoutingKey一致的队列；

4：一个Queue队列与交换机Exchange绑定的时候可以指定多个BindingKey；

5：利用@RabbitListener声明Exchange、Queue、RoutingKey，不用额外在配置类创建Bean声明，运行项目，会发现RabbitMQ管理平添自动创建好对应的交换机、队列、以及交换机和队列的绑定关系，也可以证明，交换机、队列、交换机和队列的绑定关系为啥更适合在消费者出声明！

### 2：Direct交换机和Fanout交换机的差异：

1：Fanout交换机将消息路由给每一个与之绑定的队列；

2：Direct交换机根据RoutingKey判断路由给哪个队列；

3：如果多个队列具有相同的RoutingKey，则与Fanout功能相同；

### 3：基于@RabbitListener注解声明队列和交换机有哪些常见注解？

详情见代码

@QueueBinding

@Exchange

@Queue