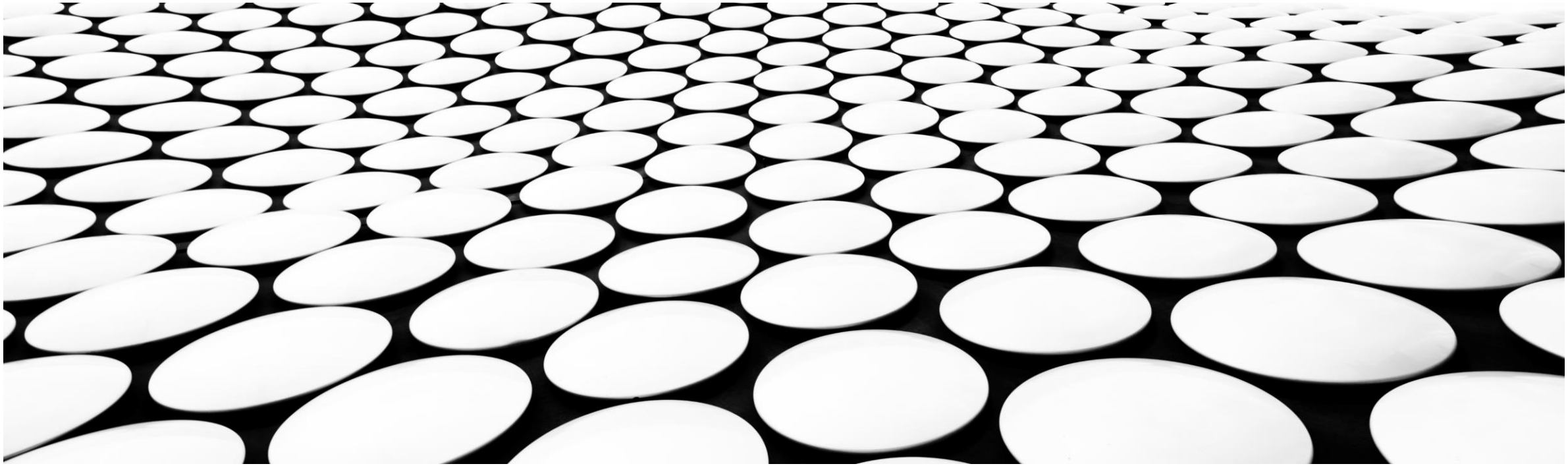


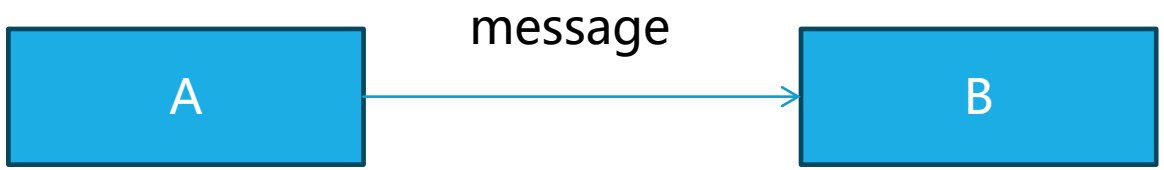
服务端开发-消息中间件(ActiveMQ、RabbitMQ)

陶召胜

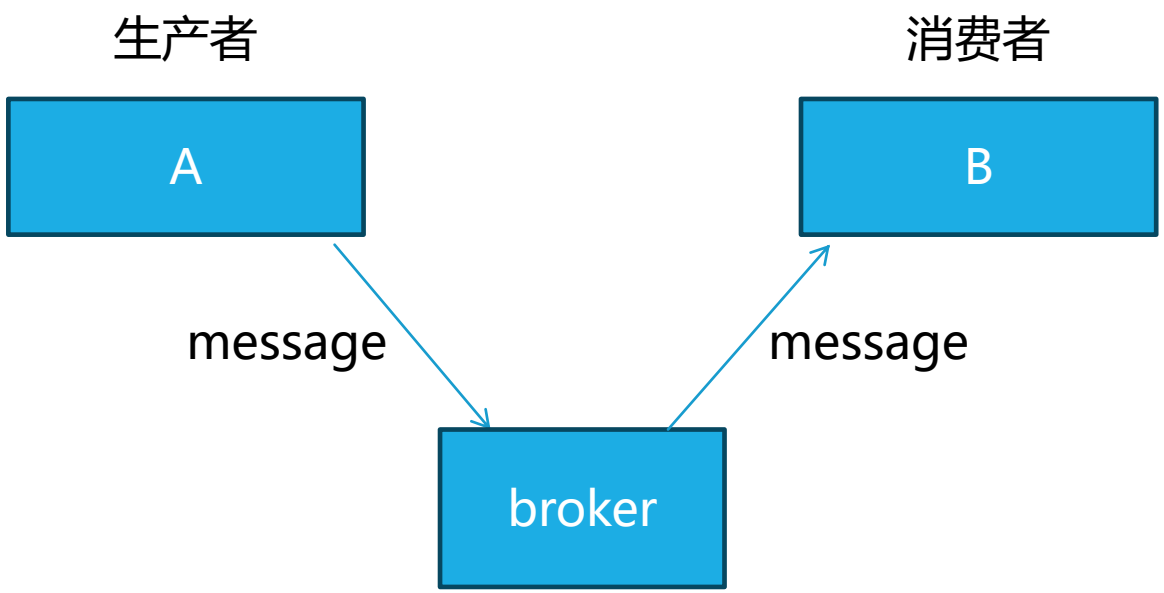
内容

1. ActiveMQ
2. RabbitMQ

同步与异步



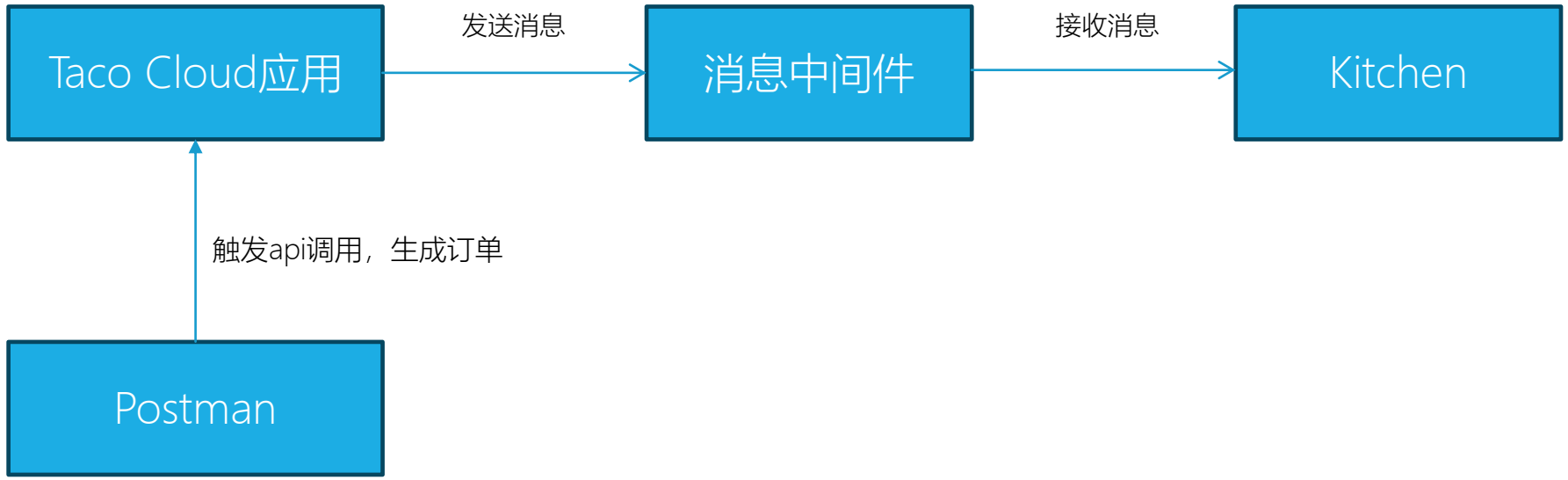
broker



消息中间件

- 消息中间件主要用于组件之间的解耦，消息的发送者无需知道消息使用者的存在，反之亦然
- 常用的消息中间件有：ActiveMQ、RabbitMQ、kafka

本节课例子



JMS

- Java 消息服务（Java Message Servcie）
- JMS是一个Java标准，定义了使用消息代理（message broker）的通用API
- Spring通过基于模板的抽象为JMS功能提供了支持，这个模板就是JmsTemplate

消息代理（broker）

- Apache ActiveMQ
- Apache ActiveMQ Artemis，重新实现的下一代ActiveMQ

ActiveMQ Artemis

- https://activemq.apache.org/components/artemis/
- https://activemq.apache.org/components/artemis/download/
- https://github.com/apache/activemq-artemis/blob/main/docs/user-manual/docker.adoc
- ActiveMQ Artemis 是一个优秀的跨平台、高性能、开源的消息代理系统
- 支持的协议
  - ✓ JMS 协议
  - ✓ AMQP （Advanced Message Queueing Protocol）
  - ✓ MQTT（Message Queuing Telemetry Transport）
- Native 内存模式与 JVM 内存模式
- 分布式架构
- 消息持久化

Docker运行

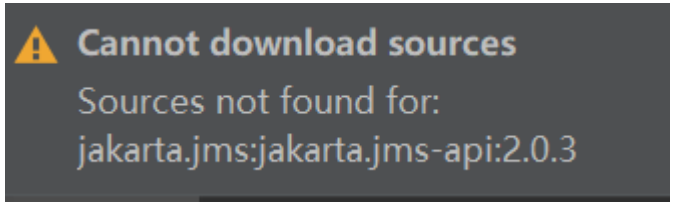
- $ docker run --detach --name mycontainer -p 61616:61616 -p 8161:8161 apache/activemq-artemis:latest-alpine
- docker logs -f mycontainer
- $ docker exec -it mycontainer /var/lib/artemis-instance/bin/artemis shell --user artemis --password artemis
- 管理控制台：http://localhost:8161　artemis/artemis

依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-artemis</artifactId>
</dependency>
```

IntelliJ IDEA不能下载源代码的问题

- mvn dependency:sources
- 翻译插件：Translation
  - ✓ 文档：https://yiiguxing.gitee.io/translation-plugin/#/docs
- AI助手：Tabnine
  - ✓ https://plugins.jetbrains.com/plugin/12798-tabnine-ai-code-completion--chat-in-java-js-ts-python--more



直接使用JMS接口发送与接收消息

- JMS规范：jakarta.jms-api-2.0.3.jar
- artemis客户端：artemis-jms-client-2.17.0.jar

```
ConnectionFactory connectionFactory = new ActiveMQConnectionFactory(BROKER_URL, USERNAME, PASSWORD);
Connection connection = connectionFactory.createConnection();
connection.start();
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
Destination destination = session.createQueue("queue.example");
```

关键概念

- javax.jms.Message：TextMessage、ObjectMessage
- javax.jms.Destination，队列或主题，如：org.apache.activemq.artemis.jms.client.ActiveMQQueue
  - ✓ 3种指定方式：
    - application.yml（default-destination）
    - @Bean（Destination对象）
    - 直接String指定

配置

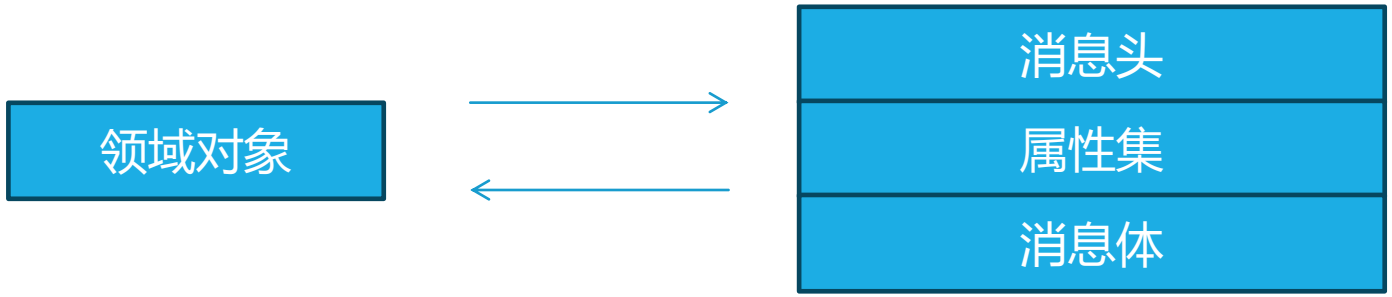
```yaml
spring:
  jms:
    template:
      default-destination: tacocloud.order.queue
  artemis:
    host: localhost
    port: 61616
    user: artemis
    password: artemis
    embedded:
      enabled: false
```

使用JmsTemplate

- JmsTemplate是Spring对JMS集成支持的核心
- 发送的两个方法：send、convertAndSend

消息转换器（MessageConverter）

- SimpleMessageConverter：实现String与TextMessage的相互转换、字节数组与BytesMessage的相互转换、Map与MapMessage的相互转换，以及Serializable对象与ObjectMessage的相互转换
- MappingJackson2MessageConverter：使用Jackson 2 JSON库实现消息与JSON格式的相互转换
- TypeId，目的是告诉对方是什么类型，以便于反序列化
- Message.setStringProperty，随属性集传输
- 消息转换器使用@Configuration定义成Bean



tacocloud.order.queue

org.apache.activemq.artemis:broker="0.0.0.0",component=addresses,address="tacocloud.order.queue",subcomponent=queues,routing-type="anycast",q

Status Connections Sessions Consumers Producers Addresses Queues Attributes Operations

Displaying body as bytes (256 bytes) and text (256 chars)

```
bytes:
7b 22 69 64 22 3a 6e 75 6c 6c 2c 22 70 6c 61 63 65 64 41 74 22 3a 6e 75 6c 6c 2c 22 75 73 65 72 22 3a 7b 22 69 64 22 3a 31 2c 22
22 3a 22 70 61 73 73 77 6f 72 64 22 2c 22 66 75 6c 6c 6e 61 6d 65 22 3a 22 43 72 61 69 67 20 57 61 6c 6c 73 22 2c 22 73 74 72 65
22 3a 22 43 72 6f 73 73 20 52 6f 61 64 73 22 2c 22 73 74 61 74 65 22 3a 22 54 58 22 2c 22 7a 69 70 22 3a 22 37 36 32 32 37 22 2c
2c 22 65 6e 61 62 6c 65 64 22 3a 74 72 75 65 2c 22 61 63 63 6f 75 6e 74 4e 6f 6e 45 78 70 69 72 65 64 22 3a 74 72 75 65 2c 22 63

text:
{"id":null,"placedAt":null,"user":{"id":1,"username":"habuma","password":"password","fullname":"Craig Walls","street":"123 North
1234","enabled":true,"accountNonExpired":true,"c
```

Headers

key ^	value
address	tacocloud.order.queue
durable	true
expiration	0 (never)
largeMessage	false
messageID	14945
persistentSize	1064 (1,064 Bytes)
priority	4
protocol	CORE
redelivered	false
timestamp	1700723325723 (2023-11-23 15:08:45)
type	4 (bytes)
userID	ID:246a365a-89cf-11ee-b758-2e8db1749bf4

Properties

key ^	value
__AMQ_CID	242760f7-89cf-11ee-b758-2e8db1749bf4
__AMQ_ROUTING_TYPE	1 (anycast)
_typeId	order
X_ORDER_SOURCE	WEB

接收消息：拉取模式（pull model），JmsTemplate支持

- 访问：http://localhost:8081/

```
(TacoOrder) JmsTemplate.receiveAndConvert("tacocloud.order.queue");
```

- TacoOrder获取的背后使用了消息转换器（MessageConverter）

接收消息：推送模式（push model），需要定义消息监听器

```
@JmsListener(destination = "tacocloud.order.queue")
 public void receiveOrder(TacoOrder order) {
   ui.displayOrder(order);
 }
```

- TacoOrder获取的背后使用了消息转换器（MessageConverter）

内容

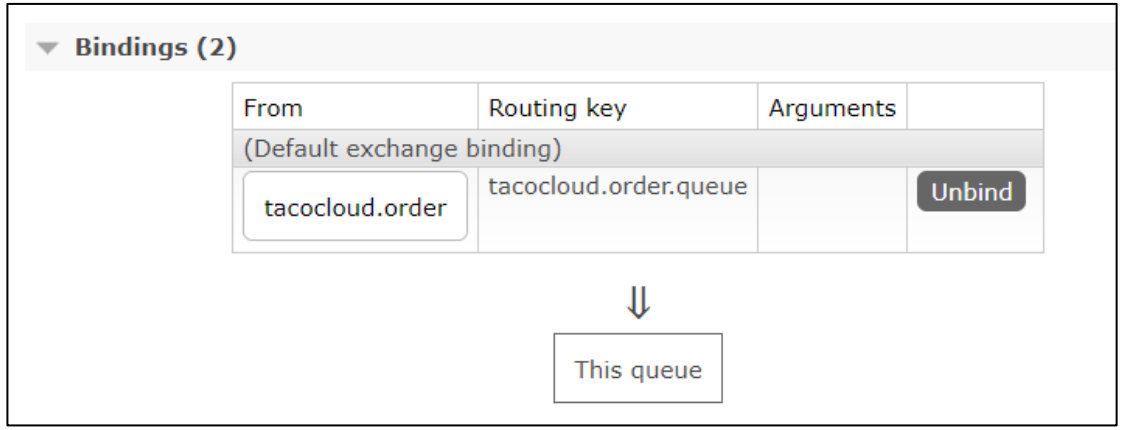
1. ActiveMQ
2. RabbitMQ

RabbitMQ

- AMQP（Advanced Message Queueing Protocol）
- https://www.rabbitmq.com/
- Docker：https://registry.hub.docker.com/_/rabbitmq/
- RabbitMQ基础概念详细介绍：https://www.cnblogs.com/williamjie/p/9481774.html

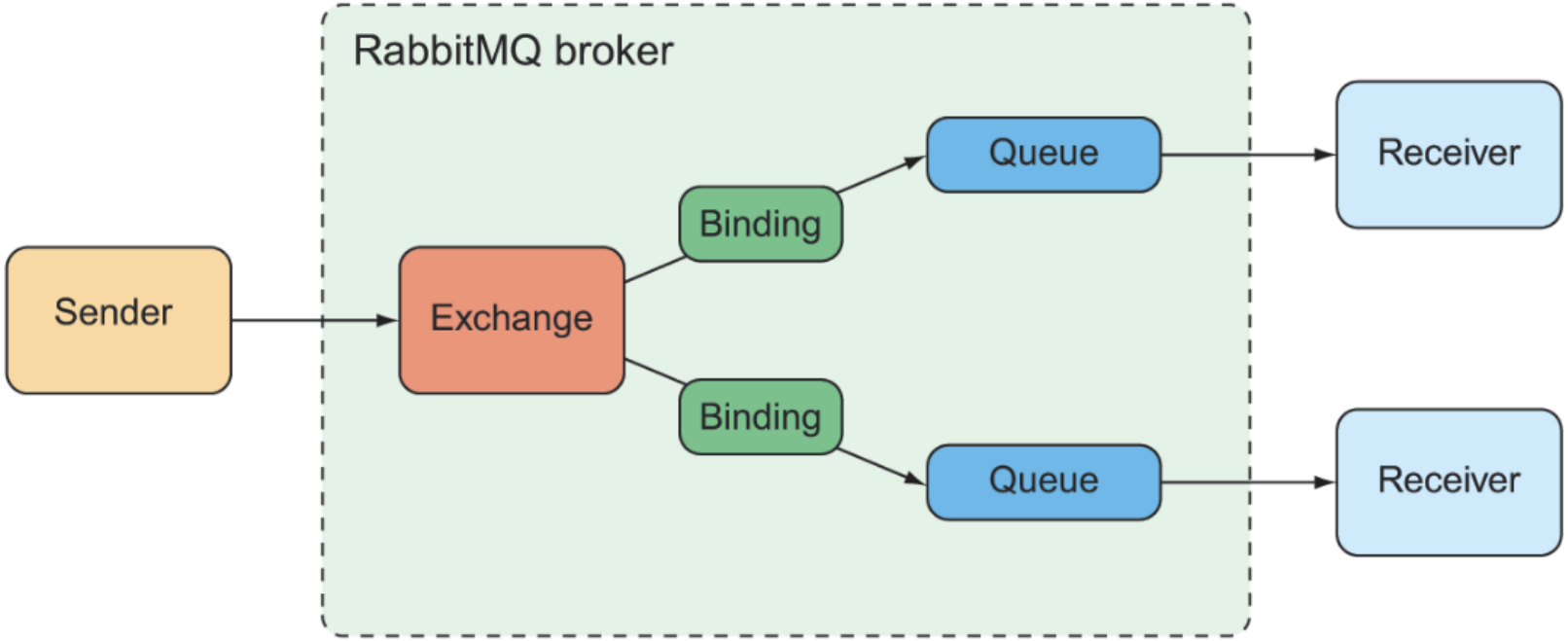
控制台

- http://localhost:15672/，guest/guest
- 要先创建Exchanges：tacocloud.order，以及Queues：tacocloud.order.queue
- 建立bind关系



RabbitMQ概念

- ConnectionFactory、Connection、Channel
- Exchange（交换机）：Default、Direct、Topic、Fanout、Headers、Dead letter
- Queue
- routing key
- Binding key



依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-amqp</artifactId>
</dependency>
```

配置

```yaml
spring:
  rabbitmq:
    host: localhost
    port: 5672
    username: guest
    password: guest
    template:
      exchange: tacocloud.order
```

同样需要消息转换器

- new Jackson2JsonMessageConverter()

Get Message(s)

Message 1

The server reported 0 messages remaining.

Exchange	tacocloud.order
Routing Key	tacocloud.order.queue
Redelivered	○
Properties	priority: 0 delivery_mode: 2 headers: X_ORDER_SOURCE: WEB __TypeId__: tacos.TacoOrder content_encoding: UTF-8 content_type: application/json
Payload 749 bytes Encoding: string	{"id":null,"placedAt":null,"user":{"id":1,"username":"habuma","password":"password","fullname":"Cr

接收消息：拉取模式（pull model），RabbitTemplate支持

- 访问：http://localhost:8081/

```
(TacoOrder) RabbitTemplate.receiveAndConvert("tacocloud.order.queue");
```

接收消息：推送模式（push model），需要定义消息监听器

```
@RabbitListener(queues = "tacocloud.order.queue")
 public void receiveOrder(TacoOrder order) {
   ui.displayOrder(order);
 }
```

谢谢观看！