1. RabbitMQ是什么？

RabbitMQ是一个消息队列中间件，实现了AMQP消息队列协议，

1. 什么是消息队列

是一个存放消息的中间件，

1. 什么时候使用消息队列

* 生产者-消费者模式是消息队列比较常用的模式，消费者和生产者的处理能力可能不一样，可以使用消息队列的生产者消费者模式平衡两者的能力
* 有多个下游服务需要用到当前服务数据时候，可以使用消息队列解耦，方便下游服务新增。

比如说订单，需要添加个短信通知，通过消息队列可以很好的解耦业务，直接添加一个队列绑定就能实现，不需要更改原有服务来多添加一个逻辑。

* 可以通过消息队列进行广播
* 可以通过消息队列实现发布订阅功能，实现跨进程的发布订阅

1. RabbitMQ特点?
2. AMQP是什么?

是一个进程间传递异步消息(消息队列)的网络协议，是应用层协议的一个开放标准，统一标准可以降低企业和系统的集成开销。Rabbitmq就是该协议的一个实现，

1. AMQP协议3层？

1. AMQP模型的几大组件？

* Broker：代表着一个中间件应用，负责接收消息生产者的消息，然后将消息发送至消息接受者或者其他的broker。
* Virtual host：这是对broker的虚拟化分，主要用于对consumer、producer和他们依赖的AMQP相关结构进行隔离。通常是处于安全因素的考虑。
* Connection：代表着producer、consumer和broker之间的物理网络（TCP），connection只有在客户端断开连接或者网络问题的时候会断开。
* Channel：代表着producer、consumer和broker之间的逻辑连接，一个Connection可以包含多个Channel。Channel使得基同一连接的不同进程之间与broker之间的交互相互隔离，不干扰。而不需要重新建立连接，channel在发生协议错误的时候会被关闭。(.net中使用IModel model = Connection.CreateModel();每个订阅者一个channel)
* Exchang：这是所有被发送的消息首先到达的目的地，Exchange负责根据路由规则将消息路由到不同的目的地。路由规则包括下面几种：direct（point-to-point）、topic（publish-subscribe）和fanout（multicast）
* Queue：这是消息到达的最终目的地，到达queue的消息是已经准备好被消费的消息，一个消息可以被exchange copy发送至多个queue。
* Binding：这是queue和exchange之间的虚拟连接，使得消息从哪个exchange路由到Queue。routing key可以通过binding和exchange routing规则关联。

1. 怎么理解生产者Producer、消费者Consumer?

生产者：消息的供给者

消费者：消息的接收者

1. 为什么需要消息队列？

* 平衡生产者和消费者的能力，有时候数据的生成能力过大，数据的处理能力过小，那么就可以通过消息队列来平衡这两种情况，类似.net的请求队列，可以限制流量，或者是使用生产者消费者模式来用多个消费者处理消息。
* 服务解耦:：比如说订单服务，当需要添加一个短信服务的时候需要在订单服务中修改逻辑，通过消息队列可以很好的解耦业务，直接添加一个队列绑定就能实现，不需要更改原有服务来多添加一个逻辑。
* 异步处理：一下不需要及时返回的服务，可以通过消息队列跨进程投递给其他服务处理，提高当前服务的性能。

1. Broker服务节点？
2. 交换器4种类型？

每个交换机都会和队列用routing key进行绑定，交换器和队列的绑定是可以多对多的。

在发送数据的时候是直接指定routing key而不是队列。然后交换器根据routing key来匹配队列。

而不同交换器匹配规则不一样

model.QueueBind(QueueName,ExchangeName,RoutingKey);

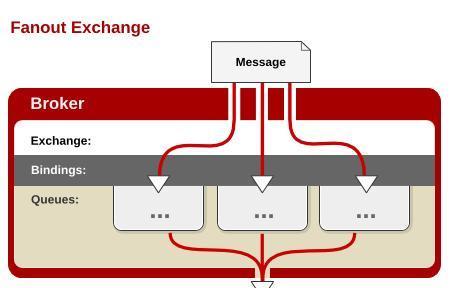
model.BasicPublish(ExchangeName, RoutingKey, prop, message);

* Direct

直接匹配routing key，交换器使用发送过来的数据的routing key直接匹配绑定的routing key，找到绑定和对应的队列

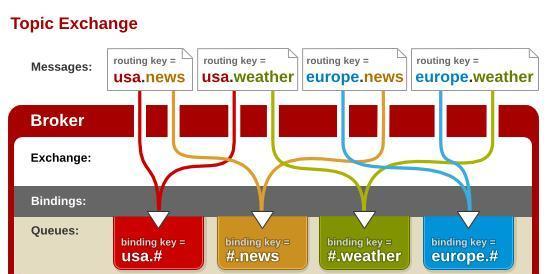
* Fanout

不处理routing key，交换器直接发送到所有绑定的队列上。



* Topic

将路由键和某模式进行匹配。此时队列需要绑定要一个模式上。符号“#”匹配一个或多个词，符号“”匹配不多不少一个词。因此“abc.#”能够匹配到“abc.def.ghi”，但是“abc.” 只会匹配到“abc.def”。



* Headers

不处理路由键。而是根据发送的消息内容中的headers属性进行匹配。在绑定Queue与Exchange时指定一组键值对；当消息发送到RabbitMQ时会取到该消息的headers与Exchange绑定时指定的键值对进行匹配；如果完全匹配则消息会路由到该队列，否则不会路由到该队列。headers属性是一个键值对，可以是Hashtable，键值对的值可以是任何类型。而fanout，direct，topic 的路由键都需要要字符串形式的。

匹配规则x-match有下列两种类型：

x-match = all ：表示所有的键值对都匹配才能接受到消息

x-match = any ：表示只要有键值对匹配就能接受到消息

1. 说说Broker服务节点、Queue队列、Exchange交换器？、

* Broker可以看做RabbitMQ的服务节点。一般请下一个Broker可以看做一个RabbitMQ服务器。
* Queue:RabbitMQ的内部对象，用于存储消息。多个消费者可以订阅同一队列，这时队列中的消息会被平摊（轮询）给多个消费者进行处理。
* Exchange:生产者将消息发送到交换器，由交换器将消息路由到一个或者多个队列中。当路由不到时，或返回给生产者或直接丢弃。

1. 消息队列有什么优缺点

* 优点

能处理很多场景的问题

* 缺点

增加系统复杂度，降低系统稳定性，需要额外去维护消息队列系统，保证消息队列不影响整个系统的稳定性。还需要保证消息的可靠传递，包括发布消息和消费消息。

1. 如何保证消息的可靠性？

* 生产者到RabbitMQ：事务机制和Confirm机制
* RabbitMQ自身：持久化、集群、普通模式、镜像模式。
* RabbitMQ到消费者：basicAck机制、死信队列、消息补偿机制(补偿是需要程序逻辑来实现，rabbitmq本身并没有该功能。)。

1. RoutingKey路由键？

1. Binding绑定？

交换机和队列的一个关联关系，bingding会规定一个routingkey，数据到达交换机时也 携带一个routingkey，根据不同的规则来匹配这两个routingkey确认把数据添加到哪个 队列中。

1. 生产者消息运转？

* Producer先连接到Broker,建立连接Connection,开启一个信道(Channel)。
* Producer声明一个交换器并设置好相关属性。
* Producer声明一个队列并设置好相关属性。
* Producer通过路由键将交换器和队列绑定起来。
* Producer发送消息到Broker,其中包含路由键、交换器等信息。
* 相应的交换器根据接收到的路由键查找匹配的队列。
* 如果找到，将消息存入对应的队列，如果没有找到，会根据生产者的配置丢弃或者退回给生产者。
* 关闭信道。
* 管理连接。

1. 消费者接收消息过程？

* Producer先连接到Broker,建立连接Connection,开启一个信道(Channel)。
* 向Broker请求消费响应的队列中消息，可能会设置响应的回调函数。
* 等待Broker回应并投递相应队列中的消息，接收消息。
* 消费者确认收到的消息,ack。
* RabbitMq从队列中删除已经确定的消息。
* 关闭信道。
* 关闭连接。

1. 交换器无法根据自身类型和路由键找到符合条件队列时，有哪些处理？
   * mandatory ：true 返回消息给生产者。
   * mandatory: false 直接丢弃。
2. 什么是死信队列？

死信队列是一个普通队列，用来存储死信消息的

首先需要定义一个死信队列(dlx.queue)和交换机(dlx.exchange)

在业务队列中添加一个x-dead-letter-exchange指向死信队列的交换机

x-dead-letter-routing-key指定传送到死信队列交互机时的routingkey

x-message-ttl指定队列中消息的存活时间，也可以直接在消息中指定。

Map<String, Object> args = new HashMap<String, Object>();

args.put("x-dead-letter-exchange", "dlx.exchange");

args.put("x-dead-letter-routing-key", DEAD\_LETTER\_QUEUEA\_ROUTING\_KEY);

args.put("x-message-ttl", 6000);

channel.queueDeclare(queueName, true, false, false, args);

在消息发布的时候,给properties添加上过期时间，当消息过期了还没有被消费的时候，就会把推送到x-dead-letter-exchange指定的死信队列交换机中。

(处理订单超时，生成订单时给队列添加一条信息，时间到后会移动到死信队列，然后 消费者会获取订单id查询数据库中订单是否已经支付，如果没有就删除订单。)

1. 导致的死信的有哪些原因？
   * 消息被拒（Basic.Reject /Basic.Nack) 且 requeue = false。
   * 消息TTL过期。
   * 队列满了，无法再添加。
2. 何为延迟队列？

存储对应的延迟消息，指当消息被发送以后，并不想让消费者立刻拿到消息，而是等待 特定时间后，消费者才能拿到这个消息进行消费。

Rabbitmq可以使用死信队列来实现延迟队列功能。

1. 什么是优先级队列？

* 优先级高的队列会先被消费。
* 可以通过x-max-priority参数来实现。
* 当消费速度大于生产速度且Broker没有堆积的情况下，优先级显得没有意义。

1. 熟悉RabbitMQ的事务机制吗？

channel.txSelect 用于将当前的信道设置成事务模式。

channel . txCommit 用于提交事务 。

channel . txRollback 用于事务回滚,如果在事务提交执行之前由于 RabbitMQ 异常崩溃或者其他原因抛出异常,通过txRollback来回滚。

try {

model.TxSelect();

for(var message in messages)

{

model.BasicPublish(ExchangeName, RoutingKey, prop, message);

}

model.TxCommit();

}

catch {

model.TxRollback();

throw;

}

1. 熟悉发送确认机制吗？

Confirm 需要研究一下

1. 消费者获取消息的方式？

* 拉取：

model.BasicGet(QueueName, false);

* 推送

model.BasicQos(0, subscriber.PrefetchCount, false);

var consumer = new EventingBasicConsumer(model);

consumer.Received += async (ch, ea) =>{

model.BasicAck(ea.DeliveryTag, false);

};

string consumerTag = model.BasicConsume(subscriber.QueueName, false, consumer);

1. 消费者某些原因无法处理当前接受的消息如何来拒绝？

* BasicReject(ulong deliveryTag, bool requeue);

deliveryTag 为消息tag，requeue为是否重新投递，如果为true，会重新投递到队列中，否则会丢弃或者放到死信队列中。

* BasicNack(ulong deliveryTag, bool multiple, bool requeue);

deliveryTag 为消息tag，multiple指是否批量，requeue为是否重新投递，其他两个和BasicReject一样，multiple为true时会进行批量操作，对deliveryTag及其之前未消费确认的消息进行处理。

* BasicRecover(bool requeue);

消息重入队列，requeue=true，发送给新的consumer，false发送给相同的consumer。

注释里不建议使用该方法，

//

// 摘要:

// /// Indicates that a consumer has recovered. /// Deprecated. Should not be used.

// ///

[AmqpMethodDoNotImplement(null)]

void BasicRecover(bool requeue);

1. 消息传输保证层级？
2. vhost是什么?

每一个RabbitMQ服务器都能创建虚拟的消息服务器，也叫虚拟主机(virtual host)，简 称vhost。默认为“/”。Vhost可以进行资源隔离，每个vhost是互不相通的。

1. 说说集群中的节点类型？
2. 熟悉队列结构吗？
3. RabbitMQ中消息可能有的几种状态?
4. 在何种场景下使用了消息中间件？
5. 生产者如何将消息可靠投递到MQ？
6. 如何保证RabbitMQ消息队列的高可用?