发布订阅模式(pub/sub)

Publish/Subscribe(发布/订阅模式)

产生背景

RabbitMQ中轮询模式和公平分发模式解决的都是消费快慢的问题，并不能解决一个消息对多个消费者的问题。例如：业务场景触发通知，需要同时发送短信和邮件通知，则可以用两个队列接收这个消息，然后短信模块，邮件模块各自独立消费。这种场景就符合 发布/订阅模式(即：一个生产者对多个消费者)

模型解读

1. 一个生产者，对多个消费者
2. 每个消费者拥有自己对队列
3. 每个队列都要绑定到交换机
4. 生产者直接把消息发给交换机，由交换机路由到多个多列，然后被多个消费者消费

我们可以使用 fanout 交换机来实现发布/订阅模型，fanout不处理路由键

Python实现发布/订阅

发布者

```
import pika, sys

credentials = pika.PlainCredentials("admin", "admin")
connection = pika.BlockingConnection(
    pika.ConnectionParameters(
        host="10.211.55.5",
        port=5672,
        virtual_host="admin",
        credentials=credentials
    )
)
channel = connection.channel()

message = ' '.join(sys.argv[1:]) or "Hello dcs!"

# 声明交换机
channel.exchange_declare(exchange='logs', exchange_type='fanout')

# 使用由空字符串表示的默认交换机即可
# 默认交换机比较特别，它允许我们指定消息究竟需要投递到哪个具体的队列中，队列名字需要在routing_key参数中指定
```

```
channel.basic_publish(exchange='logs', routing_key='', body=message)
print(" 发送消息 %s" % message)
# 安全关闭连接
connection.close()
```

订阅者

```
import pika

credentials = pika.PlainCredentials("admin", "admin")
connection = pika.BlockingConnection(
    pika.ConnectionParameters(
        host="10.211.55.5",
        port=5672,
        virtual_host="admin",
        credentials=credentials
    )
)
# 建立通道
channel = connection.channel()
# 申明交换机
channel.exchange_declare(exchange='logs', exchange_type='fanout')
# 定义随机队列，传空，即随机生成，exclusive=True 表示消费者断开后，即删除队列
result = channel.queue_declare(queue='', exclusive=True)
queue_name = result.method.queue
# 绑定队列和交换机
channel.queue_bind(exchange='logs', queue=queue_name)


def callback(ch, method, properties, body):
    print(" 获取消息内容 %r" % (body,))


channel.basic_consume(on_message_callback=callback, queue=queue_name,
auto_ack=True)

channel.start_consuming()
```

解释说明：

1. 可以启动多个消费者，每启动一个消费者则新建一个随机队列，绑定到该交换机上
2. 往交换机里写消息，不指定队列，不指定路由键，消息则会转发到所有队列，各自消费
3. 消费者进程关闭后，队列则删除

```
```