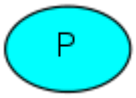
介绍：

RabbitMQ 是一个消息代理，它接收消息并且转发消息。你可以把它想象成一个邮局：你把你想发送的东西装进邮箱，你可以确认邮递员会把你的邮件发送到接收者那里。在这个类比中，RabbitMQ 是一个邮件盒子，有一个邮局，一个邮递员。

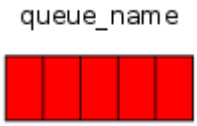
RabbitMQ 和邮局主要区别是它不处理纸张，它仅接收，保存，转发二进制数据（消息）。

RabbitMQ 和消息通常使用以下术语。

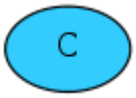
生产者：生产者意味者只生产，发送消息的程序就是生产者。



队列：一个队列就是 RabbitMQ 中邮箱的名字，虽然消息在 RabbitMQ 和你的程序之间流动，但是它只能保存在内部队列中。一个队列大小仅受限于主机内存和磁盘大小，本质上，队列就是一个大的缓冲池。许多生产者发送消息到队列里，许多消费者从队列中消费消息，以下是队列的示意图。



消费者：消费者的含义和接收者相似。一个消息者就是等待接收消息的程序。

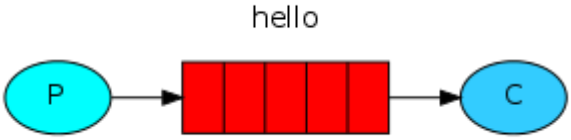


注意生产者，消费者，代理没有一定要在同一台主机内，事实上在大部分场景中它们确实都不在同一台主机。同样，一个程序即可以是消费者也可以是生产者。

Hello World

（使用 Pika Python 客户端）
在这个教学手册部分我们写一个小的 Python 程序，一个生产者发送一个消息，一个消费者消费接收这个消息，然后打印出消息内容“Hello World”。

在图中，P 代表生产者，C 是消费者，盒子就是队列。

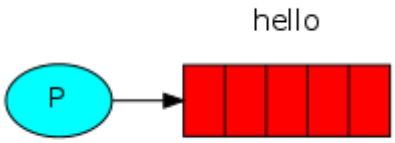


RabbitMQ 库

RabbitMQ 实现了多个协议。这个教学手册使用 AMQP 0-9-1，AMQP 是一个开放的消息协议，许多语言都有各自的 RabbitMQ 客户端。我们使用 Pythone Pika 1.0.0 客户端。使用以下命令安装

Pip install pika --upgrade

发送端



我们第一个程序 send.py 发送一个消息到队列。首先我们需要和 RabbitMQ 服务器建立连接。

```
#!/usr/bin/env python
import pika

connection =
pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()
```

现在我们已经连接本机 RabbitMQ 服务器成功。如果想要连接不同机器上的 RabbitMQ 服务器,localhost 需要改成对应的 ip 或者名字。

在发送消息之前我们必须确认接受消息的队列存在。如果我们发送消息到一个不存在的队列，RabbitMQ 会丢弃消息。使用下面语句创建一个队列。

```
channel.queue_declare(queue='hello')
```

队列也有了，我们可以准备开始发送消息。我们第一条消息仅包含字符串“Hello World”，这条消息将会发送到 hello 这个队列。

在 RabbitMQ 中我们不能直接把消息发送到队列，消息总是要经过交换机。你可以通过教学手册第三章来了解关于交换机的信息。我们需要知道使用空串来使用默认的交换机。这个交换机很特殊-它允许指定确切的队列。队列名字中需要指定 `routing_key` 参数。

```
channel.basic_publish(exchange='',
                      routing_key='hello',
                      body='Hello World!')
print(" [x] Sent 'Hello World!'")
```

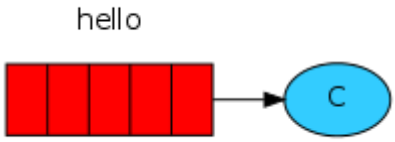
在应用程序退出之前我们需要确认已经刷空网络缓冲区里面的内容，确认我们的消息已经发送到 RabbitMQ，我们可以通过关闭连接来解决这个问题。

```
connection.close()
```

消息没发送成功

如果你是第一次使用 RabbitMQ 并且你没有看到发送的消息，你可能会挠破脑袋也想不到那里出问题了。可能代理没有足够的磁盘空间来保存消息（默认情况下代理需要至少 200MB 的磁盘空间），如果没有剩余空间代理将会拒绝这个消息。检查代理的日志。可以在配置文档里面找到 disk_free_limit 参数来设置这个限制。

接收端



receive.py 将从队列中接收消息，然后打印出消息内容。

同样，我们首先需要连接到 RabbitMQ 服务器，代码和先前的一样，在此不再重复。

下一步，和上面一样，确认队列已经存在，使用 queue_declare 来创建队列（幂等操作）-我们可以运行这个命令多次，但是只会创建一个队列。

```
channel.queue_declare(queue='hello')
```

你可能会好奇为什么我们要再次声明队列。如果我们已经确定队列存在，可以不再次声明队列。但是，比如我们不知道 send.py，receive.py 程序那个先运行。在这种情况下，最好在俩个程序中都创建队列。

列出队列

你可能想知道 RabbitMQ 中有那些队列。你可以使用 rabbitmqctl 工具，使用下面命令查看。

```
sudo rabbitmqctl list_queues
```

如果是 windows 系统，省略 sudo

```
rabbitmqctl.bat list_queues
```

从队列中接收消息比发送消息复杂，需要提交一个回调函数在队列上。当我们接收到一条消息时，回调函数就被 pika 库调用。我们的代码如下。

```
def callback(ch, method, properties, body):
    print(" [x] Received %r" % body)
```

下一步我们需要告诉 RabbitMQ 使用这个函数来接收消息。

```
channel.basic_consume(queue='hello',
                      auto_ack=True,
                      on_message_callback=callback)
```

再次提醒，必须保证我们希望接收消息的队列存在。`auto_ack` 参数将在后面的手册中说明。

最后，我希望一直等待消息。

```
print(' [*] Waiting for messages. To exit press CTRL+C')
channel.start_consuming()
```

所有代码

send.py

```
#!/usr/bin/env python
import pika

connection = pika.BlockingConnection(
    pika.ConnectionParameters(host='localhost'))
channel = connection.channel()

channel.queue_declare(queue='hello')

channel.basic_publish(exchange='', routing_key='hello', body='Hello
World!')
print(" [x] Sent 'Hello World!'")
connection.close()
```

receive.py

```
#!/usr/bin/env python
import pika

connection = pika.BlockingConnection(
    pika.ConnectionParameters(host='localhost'))
channel = connection.channel()

channel.queue_declare(queue='hello')

def callback(ch, method, properties, body):
    print(" [x] Received %r" % body)

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

print(' [*] Waiting for messages. To exit press CTRL+C')
channel.start_consuming()
```

现在我们可以在终端上运行我们的程序，首先，启动消费者，它会一直等待消息

```
python receive.py
# => [*] Waiting for messages. To exit press CTRL+C
# => [x] Received 'Hello World!'
```

然后启动生产者。

```
python send.py
# => [x] Sent 'Hello World!'
```

现在我们就可以通过 RabbitMQ 发送我们第一条消息，和你所看到的一样，receive.py 永远不会退出，它会永远等待消息，可以通过 ctrl+c 来中断它。

现在我们已经学习了怎么发送，接收消息，可以继续下一章来构建一个工作队列。

提醒：

请注意教学文档仅仅只是教学文档，它仅仅用于说明新概念，但是它太简单，不适合在生产环境中使用。例如，为了简洁，省略了大量连接管理，错误处理，连接恢复，并发性，测量等相关内容。请生产环境使用之前请查看剩余的文档，我们建议按照以下的顺序。
生产者确认消息和消费者确认消息，生产者列表，监控
以上代码的位置：GIT
原文的地址：https://www.rabbitmq.com/tutorials/tutorial-one-python.html