1. Hello World

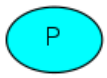
介绍（http://www.rabbitmq.com/tutorials/tutorial-one-java.html）

RabbitMQ是一个消息代理：它接受并转发消息。您可以将其视为邮局，当你将要发布的邮件放在邮箱中时，您可以确信Postman先生最终会将邮件发送给收件人。在这个比喻中，RabbitMQ是一个邮箱，邮局和邮递员。

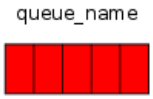
RabbitMQ和邮局之间的主要区别在于它不处理智障，而是接收，存储和转发二进制数据库的消息。

RabbitMQ和消息传递一般使用一些术语。

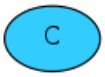
生产意味着什么 不仅仅是发送。发送消息的程序是一个生产者：



队列是居中在RabbitMQ中的邮箱的名称。虽然消息流过RabbitMQ和您的应用程序，但它们只能存储在队列中。甲队列仅由主机的存储区&磁盘限制结束，它本质上是一个大的消息缓冲器。许多生产者可以将消息发送到一个队列，许多消费者可以尝试从一个队列接收数据。这是我们如何代表一个队列：



消费具有与接收相似的含义。一个消费者是一个程序，主要是等待接收信息：



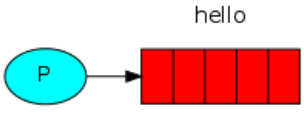
请注意，生产者，消费者和服务中间件不必在同一个主机上

“Hello， World”

在本教程的这一部分，我们将用Java编写两个程序；发送单个消息的生产者，以及接收消息并将其打印出来的消费者。我们将介绍Java API中的一些细节，专注于这个非常简单的事情，只需要开始。这是一个“Hello World”的消息传递。

在下图中，“P”是我们的生产者，“C”是我们的消费者。中间的框是队列 - RabbitMQ代表消费者的消息缓冲区。

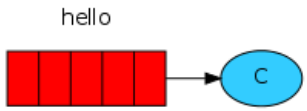
发出



```
public class Send {
    private final static String QUEUE_NAME ="hello";

    public static void main(String[] args) {

    }

    public void sendMessage() throws IOException, TimeoutException {
        ConnectionFactory factroy = new ConnectionFactory();
        factroy.setHost("127.0.0.1"); //本地地址
        Connection connection = factroy.newConnection();
        Channel channel = connection.createChannel();
        channel.queueDeclare(QUEUE_NAME, durable: false, exclusive: false, autoDelete: false, arguments: null);
        String message = "hello, world";
        channel.basicPublish( exchange: "", QUEUE_NAME, props: null, message.getBytes());
        System.out.println("[x] sent'" + message + "'");
        channel.close();
        connection.close();
    }

}
```

接收

接下来是消费者，从RabbitMQ推送消息，所以不同于发布单个消息的发布者，我们将继续运行，以收听消息并打印出来。




```
public class Recv {
  private final static String QUEUE_NAME = "hello";

  public static void main(String[] argv)
      throws java.io.IOException,
             java.lang.InterruptedException {

    ConnectionFactory factory = new ConnectionFactory();
    factory.setHost("localhost");
    Connection connection = factory.newConnection();
    Channel channel = connection.createChannel();

    channel.queueDeclare(QUEUE_NAME, false, false, false, null);
    System.out.println(" [*] Waiting for messages. To exit press CTRL+C");
    ...
    }
}
```

我们即将高速服务器将队列中的消息传递给我们。由于它会异步的推送我们的邮件，所以我们提供一个对象形式的回调，缓冲消息，知道我们准备好使用它们。这是一个DefaultConsumer子类。

```
Consumer consumer = new DefaultConsumer(channel) {
    @Override
    public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties properties, byte[] body) th
        String message = new String(body, charsetName: "UTF-8");
        System.out.println("[x] received '" + message + "'");
    }
};
channel.basicConsume(QUEUE_NAME, autoAck: true, consumer);
```