我们先来补充一些知识，打开控制台，输入`symfony console messenger:consume`，我们输入--help查看所有的命令。我们可以在`messenger:consume`命令行后面添加队列的名称。我们要处理async传输中的数据，我们可以使用`symfony console messenger:consume async`，这样就会直接处理async传输中的消息。

按ctrl+c退出命令行，我们可能配置有多个传输，不同的传输具有不同的优先级。我们可以在consume命令行后面，根据优先级的先后添加传输的名称。通过这种方式，async1队列中的消息，将会async2队列中消息优先进行处理。

回到浏览器，我们搜索rabbitmq，查看rabbitmq的文档。在文档中，rabbitmq提供了几种使用模型，messenger组件自动的为我们创建了fanout模型，也就是第3种模型。首先创建一个exchange，然后将不同的消息发送到不同的队列中，再由不同的消费者进程对队列中的数据进行处理。

我们再次搜索messenger文档，我们查看AMQP队列的文档，往下看。我们可以在AMQP传输中添加很多配置，我们可以配置队列的名称，队列绑定的参数，队列绑定的_keys，我们还可以修改exchange的类型。

回到rabbitmq文档，messenger组件默认创建fanout类型的队列，那么我们就可以通过transport配置，来创建Routing模型和Topics模型 (后两种模型是fanout类型模型的进阶)。

如果有需要的话，你可以参考文档对这些选项进行配置，这里需要rabbitmq的知识，我就不再深入讲解了。

回到项目，我们来查看一下`amqp-messenger`的源码，来深入的学习一下。打开vendor目录，打开symfony目录，第一项就是amqb-messenger。我们打开传输目录，在Transport目录中有一个Connection类，Connection类通过我们messenger的配置与AMQP队列进行连接。

我们往下看，这里有个`fromDsn()`方法，首先获取$dsn地址进行解析，然后指定exchange的名称，如果$dsn中没有配置exchange的名称，那么默认的就会使用`messages`作为exchange的名称。

下面是AMQP的一些配置，这里配置了exchange的名称，exchange的名称就是前面的`messages`。通过$amqpOptions这个变量，messenger组件就会自动的创建fanout类型的队列。

AMQP队列我们暂时就讲到这里，在下节课我们来学习最后一个组件，Api-platform组件，用API-platform可以快速的为我们创建统一的API接口。