我们需要使用MessageBus对象将消息发送到传输队列中，我们打开之前的订阅器。修改代码，注释34行到40行代码，我们注释之前的属性和构造方法。新增构造方法，在构造方法中我们注入MessageBus对象，alt加回车初始化属性。

在条件判断中我们来生成一个消息$message，等于new SendEmailMessage()，这里需要传入一个参数$postId，$post->getId()。然后我们使用MessageBus对象将消息发送到传输中，$this->messageBus->dispatch($message);，这样就可以了。

```
#src/EventSubscriber/SendEmailSubscrberSubscriber.php

class SendEmailSubscrberSubscriber implements EventSubscriberInterface
{

    // ...

    /**
     * @var MessageBusInterface
     */
    private MessageBusInterface $messageBus;

    public function __construct(MessageBusInterface $messageBus)
    {
        $this->messageBus = $messageBus;
    }

    public function onAfterEntityPersistedEvent(AfterEntityPersistedEvent
$event)
    {
        $post = $event->getEntityInstance();
        if ($post instanceof Post)
        {
            // ...

            $message = new SendEmailMessage($post->getId());
            $this->messageBus->dispatch($message);
        }
    }
}
```

我们查看messenger.yaml配置文件，在transports下有个同步的传输队列，我们取消行前的注释。然后将我们自定义的消息SendEmailMessage ，配置到同步传输中，这样当订阅器发送消息时，会将消息发送到同步传输中，然后消息处理器Handler，在处理消息时会从同步传输中获取到对应的message进行处理。

```
#config/packages/messenger.yaml

framework:
    messenger:
        # Uncomment this (and the failed transport below) to send failed
messages to this transport for later handling.
```

```
        # failure_transport: failed
        transports:
            # https://symfony.com/doc/current/messenger.html#transport-
configuration
            # async: '%env(MESSENGER_TRANSPORT_DSN)%'
            # failed: 'doctrine://default?queue_name=failed'
            sync: 'sync://'

        routing:
            # Route your messages to the transports
            'App\Message\SendEmailMessage': sync
```

回到浏览器，我们当前已经启动了mailcatcher，我们新增一篇文章，文章的名称叫做Hello word 123，点击添加按钮，因为我们当前使用的是同步的传输，所以当前仍然会等待10秒钟再进行跳转。

当进行跳转之后，我们接收到了第三封邮件。在下节课，我们将学习doctrine队列真正的使用异步发送邮件。