搜索symfony translation，我们查看Symfony文档。我们往下看，首先这里提示我们需要安装translation组件。我们的管理端已经使用了翻译的功能，这个包我们在安装其他包时已经安装过了。然后是配置文件，这里我们也配置过了。

继续往下看，这里有段示例代码在controller方法中。我们通过依赖注入的方式注入了$translator对象，使用$translator对象trans()方法对文本进行翻译。

回到项目，在PostController的show()方法中，我们也可以通过依赖注入的方式注入$translator 对象，然后我们使用$translator对象对消息内容进行翻译。

回到浏览器，我们打开博客首页，查看第一篇文章。在评论框中我们提交一篇评论，点击提交。评论表单提交后仍然显示了一条flash消息，我们查看一下底部的翻译按钮。在列表中我们找到了flash消息的翻译项，虽然它显示的中文，但是这条消息并没有翻译。我们需要在messages翻译文件中对这条消息进行翻译。

通常我们需要翻译的Message ID使用英文。回到项目，这里我们修改一下，修改为comment_submit_message，打开messages翻译文件，添加翻译项comment_submit_message。这里翻译为您的评论已提交。

回到浏览器，刷新文章详情页，重新提交表单，现在消息的内容已经修改了。我们查看一下底部的翻译按钮，打开Defined选项卡，这里就是我们定义的消息ID然后这是翻译的结果。

回到项目，我们在PostController类中使用addFlash()方法添加了flash消息。我们打开CommentController，在CommentController的方法中也使用了addFlash()方法，添加了flash消息，flash消息的消息内容也需要进行翻译。

我们可以对addFlash()方法进行一下封装，CommentController继承于AbstractController，PostController类也继承于AbstractController。我们可以创建一个BaseController类，让BaseController类继承于AbstractController。在BaseController类中对addFlash()消息进行一下封装。

在Controller目录中新增一个PHP类，BaseController，点击OK。我们让BaseController继承AbstractController，在BaseController中我们定义一个方法public function addFlashMessages()。

在方法体中我们使用$this->addFlash()方法来添加flash消息，addFlash()方法需要传递两个参数，分别是$type和$message。这两个参数我们可以通过方法的参数进行传入，我们还需要对$message消息进行翻译，就需要使用$translator对象。

我们可以通过方法来传入$translator对象，这是一个办法，我们还可以使用另外一个办法。在AbstractController类中有个$container容器，我们可以通过容器来获取$translator对象。

回到BaseController，$this->container->get('translator'), $translator对象的服务别名就是translator。它的返回值我们定义为$translator，我们使用$translator对象的trans()方法对消息进行翻译。

trans()方法有四个参数，第一个参数就是消息的ID，第二个参数是消息中的参数，我们可以在消息中定义参数。第三个参数是当前消息所在的域，第四个参数是当前的区域。我们将这四个参数都进行一下传入，后面三个参数，我们通过方法的参数进行传入。拷贝，粘贴。单词拼错了，我们为后面三个参数添加默认值。

回到PostController，我们修改Controller类的继承类，这里继承于BaseController。在addFlash()方法这里我们进行一下修改，使用$this->addFlashMessages()，消息类型我们传入success，消息体我们传入comment_submit_message。

回到浏览器，后退一步，再次刷新文章详情页。它提示出错了，没有找到translator服务类。

回到项目，打开BaseController，我们的服务类应该在容器中的，但是没有找到这个服务类。我们查看一下容器，刷新，我们当前的容器并不是一个完整的容器。它是一个ServiceLocator对象。

在前面课程我们讲解集成测试时讲解过ServiceLocator，我们查看一下当前ServiceLocator对象中的服务类，我们展开serviceMap属性，这里定义了一些服务类id。

当我们通过容器的get的方法来获取这些服务类时，会调用对应的get方法来获取对应的服务类对象。当前没有translator对象的获取方法，所以没有找到$translator对象。

回到项目，我们查看AbstractController类。查看类中的方法，这里有个GetSubscribedService()方法。我们可以覆盖这个方法，将translator服务类对象添加到Controller的容器中。

回到BaseController，我们可以覆盖一下方法，我们对方法进行一下修改。父类的GetSubscribedService()方法返回的是一个数组，我们可以在数组中添加一项。使用array_marge()，添加的服务类ID是translator，它指向'?'.TranslatorInterface::class。

```
#src/Controller/BaseController.php

class BaseController extends AbstractController
{
    public function addFlashMessages(string $type, $message, $parameters =
[], $domain = null, $locale = null)
    {
//        dd($this->container);
        $translator = $this->container->get('translator');
        $this->addFlash($type, $translator->trans($message, $parameters,
$domain, $locale));
    }

    public static function getSubscribedServices()
    {
        return array_merge(parent::getSubscribedServices(), [
            'translator' => '?'.TranslatorInterface::class
        ]);
    }
}
```

注释14行代码，回到浏览器再次刷新，起作用了。我们的评论消息进行了翻译。回到项目，我们对CommentController类也进行一下修改，BaseController，注释69行代码。使用$this->addFlashMessages()，消息类型success，消息内容是common_submit_message。

打开翻译文件，我们可以在翻译文件中定义一些变量，我们使用双百分号对变量进行包裹。当评论提交后，这里需要替换为提交者的姓名。回到PostController，我们需要为addFlashMessages()方法添加第三个参数，是个数组，数组的键就是我们参数的名称%name%。

它的值就是当前的评论对象的作者getAuthor()，我们需要调整一下flash消息的代码位置，调整到flush()方法后面。回到浏览器再次刷新，提交表单，现在评论的作者就进行了显示，这就是通过服务类的方式进行的翻译。

我们博客的前端页面使用了很多硬编码，并没有进行翻译。在下节课我们来修改模板代码，增加翻译功能。