我们点击详情页底部工具栏上的翻译按钮，在新打开的页面上，这里有翻译地区的显示，现在默认的地区是英文地区，回调地区也是英文地区。回到项目，我们打开config目录下translation.yaml文件。

在framework配置下定义了默认的地区是英文地区，我们需要修改为中文地区。我们可以在services.yaml文件中定义一个参数，设置为中文。在translation.yaml文件中，我们使用单引号百分号来使用这个参数。这样Symfony在进行国际化时，首先会查询中文地区的翻译文件，如果没有找到中文地区的翻译文件，会查询fallbacks下配置的英文地区文件。所有的翻译文件都在项目的translations目录中。

```
#config/services.yaml
parameters:
    # ...
    locale: 'zh_CN'

#config/packages/translation.yaml
framework:
    default_locale: '%locale%'
    translator:
        default_path: '%kernel.project_dir%/translations'
        fallbacks:
```

回到浏览器，我们再次访问文章详情页，点击详情页下面的翻译按钮，现在默认的地区就显示成中文地区了。在下面的消息列表中，Locale已经显示为中文地区，Domain是翻译文件所在的域，对应的就是翻译文件的文件名。

我们回到项目查看一下paginator-bundle的代码，打开vendor目录。在knplabs目录下，打开knp-paginator-bundle目录，在translations目录下，这里就是当前的bundle所有的翻译文件，我们可以复制英文地区的翻译文件，粘贴到translations目录下。

文件名的地区，我们修改为中文地区，OK。我们将上一页按钮翻译为上一页，下一页按钮翻译为下一页，这里搜索关键字，我们可以翻译为搜索。清除一下缓存。symfony console cache:clear，回到浏览器回退，我们再次刷新一下文章详情页。

现在上一页下一按钮就翻译成了中文，我们还可以使用yaml文件来进行翻译，在translations目录下添加一个yaml文件，文件名就是翻译的条目所属于的域，然后是地区，最后是文件后缀。这里我们输入KnpPaginatorBundle。在yaml文件中，我们可以以键值对的方式来对字符串进行翻译。

```
#translations/KnpPaginatorBundle.zh_CN.yaml

label_previous: '上一页'
label_next: '下一页'
filter_searchword: '搜索...'
```

我们修改一下之前翻译文件的后缀，再次清除一下缓存。Symfony把.bak文件也当成了翻译文件，我们删除这个文件，再次清理缓存。回到浏览器刷新，同样的翻译文件也生效了。

在下节课，我们将优化一下表单的提交，每次表单提交后让页面上有一个提示。