打开底部控制台，我们输入`symfony console make:test`，来创建测试代码。测试类的基类我们选择`WebTestCase`。测试类名，我们想让功能测试的代码保存到`FunctionalTest`目录中。在测试类名前添加这个目录名称，类名叫做`PostDetailTest`。

我们打开`PostDetailTest`类，修改测试方法，方法名称叫做`testCommentSubmit()`。在测试代码中会创建一个客户端，然后客户端会向项目的首页发送一个GET请求。点击后会返回一个`Crawler`对象。

回到浏览器，我们搜索`WebTestCase`，查看一下文档。搜索`symfony WebTestCase`。打开Symfony的测试文档，在最底部。我们找到`DomCrawler`这个组件。点开，`DomCrawler`组件可以对html页面上的DOM元素进行过滤和抓取。

回到测试代码，`Crawler`对象抓取到某些元素后，$client提供一些方法，可以执行DOM元素的事件。比如说点击链接，点击按钮或者表单提交等等。最后通过DOM事件的响应来查看页面的显示结果是不是符合我们的预期。

回到浏览器，打开博客首页，首先我们让`Crawler`对象抓取到`Read More`这个链接，然后点击`Read More`链接，进入文章详情页之后，我们找到`Submit`按钮所在的表单。然后模拟一下表单的提交，再判断表单提交后评论列表中能不能找到我们提交的数据。

回到代码，删除第15行，首先我们验证博客的首页响应的是否已经成功了，然后我们找到第一个`Read More`链接，检查。`Read More`文本是`Read More加上箭头儿`，复制一下。回到代码，我们创建一个变量$link，它等于`$crawler->selectLink()`，链接的名称就是Read More加上箭头，然后再获取到它的链接。

在使用$client点击链接`click($link)`，最后click()方法，它的返回值仍然是个`Crawler`对象，我们叫做$pageDetailCrawler。我们再次添加一个断言，我们断言一下详情页是否已经响应成功了。在详情页我们找到Submit按钮所在的表单，添加一个$form变量，它等于`$pageDetailCrawler->selectButton('Submit')`。查找Submit按钮，然后找到Submit所在的表单，然后在表单中添加一些数据。我们回到首页，点开详情页，检查表单行元素，我们复制表单行的名称，等于Teebblog，然后是表单行的邮件地址。

查看代码，输入Teebblog，然后是表单行的正文`comment[message]`，我们输入中文`你好，世界！`。最后，使用$client浏览器对象提交这个表单，首先我们断言提交后页面响应是否已经成功了。然后在响应后的页面，我们来查找响应后的页面中是否包含作者的名称和作者提交的内容，我们使用$this。 使用`$client->getResponse()`。

getContent()获取到响应的内容，command+d复制一行，我们查找评论的消息，然后执行这个测试。

```
#tests/FunctionalTest/PostDetailTest.php

class PostDetailTest extends WebTestCase
{
    public function testCommentSubmit(): void
    {
        $client = static::createClient();
        $crawler = $client->request('GET', '/');

        $this->assertResponseIsSuccessful();

        $link = $crawler->selectLink('Read More →')->link();
        $pageDetailCrawler = $client->click($link);
```

```
        $this->assertResponseIsSuccessful();

        $form = $pageDetailCrawler->selectButton('Submit')->form();
        $form['comment[author]'] = 'Teebblog';
        $form['comment[email]'] = 'Teebblog@example.com';
        $form['comment[message]'] = '你好，世界！';
        $client->submit($form);

        $this->assertResponseIsSuccessful();
        $this->assertStringContainsString('Teebblog', $client-
>getResponse()->getContent());
        $this->assertStringContainsString('你好，世界！', $client-
>getResponse()->getContent());
    }
}
```

打开控制台，我们新建一个控制台，使用`--filter`选项，我只测试`testCommentSubmit()`方法，测试通过了。我们来查看一下测试数据库中的内容，打开数据库客户端，在comment表中，我们找到了刚刚使用测试代码插入的数据。

这就是使用功能测试对表单页面的提交，在下节课我们将解决一下评论的嵌套回复功能。