打开控制台，我们使用命令行来创建功能测试代码，测试类的基类这次我们选择PantherTestCase。我们需要把功能测试代码添加到FunctionalTest目录中，在类名前我们添加这个目录名称，类名叫做CommentTest。

打开CommentTest类，这里有个自动生成的测试方法，测试类第11行，这次它使用了createPantherClient()方法来生成了一个Client对象 然后Client对象向项目的首页发送了一个GET请求。

我们修改一下方法的名称，叫作testReplyComment()。回到博客首页，在测试代码中，我们将让浏览器点击第一篇文章的Read More链接进入文章详情页。

在详情页，我们在文章下方的评论框中添加一篇评论，然后在评论列表中找到这篇评论，然后在评论的后方点击回复按钮来添加一篇评论。最后测试添加的评论是否也在评论列表中进行了显示。

回到代码，删除第14行代码，第12行$client已经访问了博客的首页，然后我们抓取页面的Read More链接，$link等于selectLink()，链接名称是Read More加个箭头。复制一下，找到链接，Client对象访问链接，，click()方法返回的结果仍然是个Crawler对象，叫做$pageDetailCrawler。

在文章详情页，我们找到Submit按钮所属于的表单。回到代码，selectButton()，button名叫做Submit，找到form()，返回对象是个$form。然后添加表单数据提交一下，我们直接复制之前那个文章详情页的代码。第24行，我们直接通过浏览器响应的内容来判断表单是否已经成功提交了，我们还可以使用其他的断言方法。

assertSelectorTextContains()它会查找页面的DOM元素，然后看页面的DOM元素中是否包含了我们想要的内容。第一个参数就是DOM元素的标签，我们回到浏览器查看代码，我们期望在评论列表中会出现我们提交的评论，检查一下评论列表的代码。在media-body类的div元素中，我们应该能找到我们刚刚提交的表单数据。

复制一下media-body，selector我们输入.media-body，我们查找作者，我们复制24行代码，删除第26行代码。同样的，我们在.media-body中查找我们刚刚提交的评论信息。

Panther它使用了一个真正的浏览器，我们可以在测试代码中让Client对象直接调用JS代码来点击回复按钮。输入$client->excuteScript() 参数这里我们输入JS代码，document.querySelector()。

找到按钮之后，然后执行click()方法，按钮点击后会发送一个Ajax请求，我们等待Ajax请求的响应，$client->waitFor()。

回到浏览器，当我们点击回复按钮时，会在页面中生成一个表单，我们检查一下表单，我们等待这个表单元素显示。我们waitFor(div.reply-comment-card)。

然后我们断言一下，断言表单中包含回复评论这四个字。$this->assertSelectorTextContains()输入div.replay-command-card 包含回复评论四个字。回到浏览器，我们可以使用详情页的Crawler来过滤出回复评论这个表单div。

回到代码，$pageDetailCrawler->filter('div.reply-comment-card')，它的返回值也是个Crawler，我们叫做$replyCommentDivCrawler，再使用$replyCommentDivCrawler获取Submit按钮，然后提交表单数据我们复制这里代码。修改一下叫做$replyform。

作者的名称我们叫做Teebblog2，邮箱Teebblog2，回复的内容我们修改为测试回复评论。我们再次下断言，我们断言评论列表中包含Teebblog2和测试回复评论。我们还可以使用Client对象来截下屏，截屏的图片我们保存为screen.png。

打开控制台，我们来执行测试代码。我们只执行回复评论的测试方法，测试出错了，我们来解决一下错误问题，首先是第23行，它提示我们HttpFoundation Response对象是不可用的。

当使用WebDriver时，也就是说在使用真正的浏览器时，我们不能断言浏览器的响应对象，注释掉23行代码再次进行测试。第32行又出错了，没有找到对应的元素，我们查看一下32行代码，32行代码是在页面中过滤出回复评论表单的div。现在详情页的爬取器(Crawler)对象，它是我们刚刚打开详情页时的对象。在我们点击回复按钮时，详情页的内容已经进行了更改，waitFor()方法会返回一个新的页面爬取器(Crawler)对象，包含页面上最新的内容。我们使用新的爬取器(Crawler)对象来过滤表单的div元素，再次进行测试，现在测试就通过了。后面的警告我们暂时忽略。

```
#tests/FunctionalTest/CommentTest.php

class CommentTest extends PantherTestCase
{
    public function testReplyComment(): void
    {
        $client = static::createPantherClient();
        $crawler = $client->request('GET', '/');

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

        $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->assertSelectorTextContains('.media-body', 'Teebblog');
        $this->assertSelectorTextContains('.media-body', '你好，世界！');

        $client->executeScript('document.querySelector(".js-reply-comment-
btn").click()');
        $newPageDetailCrawler = $client->waitFor('div.reply-comment-
card');

        $this->assertSelectorTextContains('div.reply-comment-card', '回复评
论');

        $replyCommentDivCrawler = $newPageDetailCrawler-
>filter('div.reply-comment-card');
        $replyform = $replyCommentDivCrawler->selectButton('Submit')-
>form();
        $replyform['comment[author]'] = 'Teebblog2';
        $replyform['comment[email]'] = 'Teebblog2@example.com';
        $replyform['comment[message]'] = '测试回复评论';
        $client->submit($replyform);

        $this->assertSelectorTextContains('.media-body', 'Teebblog2');
        $this->assertSelectorTextContains('.media-body', '测试回复评论');
```

```
            $client->takeScreenshot('screen.png');

        }
    }
```

查看项目根目录，测试代码自动生成了一个screen.png文件，我们看到测试代码提交的评论，当然它显示的不够完全。

我们打开浏览器查看一下Panther的README文件。再往下拉，我们可以设置一下Panther的一些环境变量。我们使用PANTHER_NO_HEADLESS环境变量，它会显示一个浏览器的测试窗口。

回到项目，打开控制台粘贴PANTHER_NO_HEADLESS，我们设置为1，后面添加测试命令行，回车。我们看到刚刚命令行自动打开了一个浏览器窗口，然后按照测试代码的步骤一步一步执行了测试。

这就是使用Panther进行的功能测试，现在我们回到博客的首页，我们首页列表下方有个上一页和下一页的分页器按钮。

我们需要对文章进行一下分页，然后在文章的详情页下，我们还需要对评论进行一下分页。在下节课，我们将使用Fixtures生成一些假的数据，使用假数据来完成分页功能。