先来修改一下单词，这里少写了一个字母b，我们打开底部控制台，输入`symfony consolemake:test`，我们要编写测试代码来验证我们的首页模板是否按照我们的预期进行了显示。我们来看`WebTestCase`，它用来运行一个类似浏览器的场景，但是并不会执行JS代码，我们现在的页面没有JS代码，所以我们可以使用这个`WebTestCase`作为我们测试类的基类。

测试类的名称，我们叫做首页测试，我们打开命令行生成的测试类。测试类中有一个示例的测试方法，第11行首先它会创建一个$client，然后$client访问项目的根目录，首先它会验证$client的访问是否已经响应成功了，然后第15行它会对页面上的文字进行抓取，然后验证文字中是否包含我们预期响应的文字，这里我们修改一下我们的首页模板上有个h1标签，它的内部是`Teebblog List`。我们复制`Teebblog List`，修改`hello world`为`Teebblog List`。

这行代码的功能就是它将会抓取首页的HTML代码，然后查找h1标签，看看h1标签中的text是不是`Teebblog List`。我们直接执行测试代码，打开控制台，所有的测试代码都通过了，我们修改一下测试代码的方法名称，在测试代码中使用了两个简单的断言方法进行了页面功能的测试。

`WebTestCase`也提供了很多其他的断言方法，比如我们可以断言一下web页面标题是否按照我们的预期进行了显示，比如说我们断言我们的标题为`Teebblog`，再次进行测试，测试依旧通过了。

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

        $this->assertResponseIsSuccessful();
        $this->assertSelectorTextContains('h1', 'Teebblog List');

        $this->assertPageTitleSame('Teebblog');
    }
}
```

现在我们的测试代码是对页面上的功能进行测试，这一部分测试代码我们可以叫做功能测试。

在tests目录中创建一个新的目录，将我们编写的IndexPageTest移动到FunctionalTest目录，然后修改测试类的命名空间。 回到浏览器，刷新页面，现在显示的三篇文章并没有设置封面图像，我们打开管理端，打开文章列表，编辑我们手动添加的文章，让它的状态设置为已发布状态。再次回到首页刷新，我们看到我们手动添加的文章已经在列表中显示了，文章的正文这里HTML标签进行了转义

我们需要让正文以HTML标签的样式进行显示，postImage封面图像它保存的是个文件名，我们想让它显示一张图像，该怎么办呢?

在下一节课，我们也将解决这些问题。