

PHPUnit in Action

The Easy Way

何广宇

PHPUnit in Action

The Easy Way

何广宇

這本書的網址是 <http://leanpub.com/phpunit-in-action>

此版本發布於 2020-07-22



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

© 2020 何广宇

Contents

前言	1
关于作者	2
1. PHPUnit Hello World	4
1.1 安装 LAMP 环境	4
1.2 安装 PHPUnit	6
1.3 Hello World	7
1.4 Hello World 重构	11
1.5 Hello World Test	12
1.6 phpunit.xml	14
1.7 本章小结	15
2. 继续 Hello World	16
2.1 测试异常	16
2.2 @dataProvider	18
2.3 @group	20
2.4 Code Coverage	22
2.5 关于 Code Coverage 的进一步思考	25
2.6 PHPUnit 手册	26
2.7 本章小结	26
3. 实战项目 OurBlog 第一阶段介绍	27
3.1 用户注册	27
3.2 用户登录	27
3.3 后台首页	27
3.4 添加文章	27
3.5 编辑文章	27
3.6 删除文章	27
3.7 前台首页	28
3.8 文章详情页面	28
3.9 用户退出	28
3.10 本章小结	28
4. OurBlog 目录结构及数据库	29

4.1 OurBlog 目录结构	29
4.2 OurBlog 数据库	29
4.3 本章小结	29
5. OurBlog 用户注册	30
5.1 注册表单	30
5.2 提交表单	30
5.3 autoload	30
5.4 测试提交表单逻辑	30
5.5 reg 准备	30
5.6 reg	31
5.7 测试要连着数据库测吗?	31
5.8 DbUnit	31
5.9 phpunit-no-namespace	31
5.10 reg tests 准备	31
5.11 reg tests	31
5.12 再次测试提交注册表单	31
5.13 本章小结	32
6. OurBlog 用户登录	33
6.1 登录页面逻辑	33
6.2 测试登录逻辑	33
6.3 auth	33
6.4 auth tests 准备	33
6.5 auth tests	33
6.6 本章小结	33
7. OurBlog 后台首页	34
7.1 check login	34
7.2 header footer	34
7.3 后台首页	34
7.4 本章小结	34
8. OurBlog 添加文章	35
8.1 添加表单	35
8.2 测试添加表单	35
8.3 add	35
8.4 add tests	35
8.5 本章小结	35
9. OurBlog 编辑文章	36
9.1 编辑表单	36
9.2 edit	36
9.3 edit tests	36

9.4 本章小结	36
10. OurBlog 删除文章	37
10.1 delete	37
10.2 delete tests	37
10.3 本章小结	37
11. OurBlog 前台首页	38
11.1 header footer	38
11.2 文章列表	38
11.3 本章小结	38
12. OurBlog 文章详情页面	39
13. OurBlog 用户退出	40
14. 实战项目 OurBlog 第二阶段介绍	41
14.1 用户注册发送激活邮件	41
14.2 增加打标签功能	41
14.3 增加文件上传功能	41
14.4 增加外部文章功能	41
14.5 本章小结	41
15. OurBlog 用户注册发送激活邮件	42
15.1 邮件发送方式?	42
15.2 怎么判断用户是否已激活?	42
15.3 生成随机 token	42
15.4 实现发送激活邮件逻辑	42
15.5 写测试	42
15.6 注册成功提示	42
15.7 发送邮件 cronjob	43
15.8 实现激活逻辑	43
15.9 写测试	43
15.10 激活成功提示	43
15.11 修改登录验证逻辑	43
15.12 写测试	43
15.13 run-all-tests.sh	43
15.14 本章小结	44
16. OurBlog 增加打标签功能	45
16.1 新增数据库表	45
16.2 添加文章打标签	45
16.3 写测试	45
16.4 编辑文章打标签	45
16.5 写测试	45

CONTENTS

16.6 删除文章时也删除标签	45
16.7 写测试	46
16.8 本章小结	46
17. OurBlog 增加文件上传功能	47
17.1 文件上传页面	47
17.2 文件上传逻辑	47
17.3 写测试	47
17.4 本章小结	47
18. OurBlog 增加外部文章功能	48
18.1 数据库	48
18.2 添加文章	48
18.3 同步字段修改	48
18.4 写测试	48
18.5 编辑文章	48
18.6 写测试	48
18.7 前台文章详情页面	49
18.8 本章小结	49
19. 让测试跑的更快	50
19.1 DbUnit	50
19.2 SSD	50
19.3 MySQL in memory	50
19.4 使用 MySQL in memory	50
19.5 本章小结	50
20. 真实项目分享1: OurATS	51
21. 真实项目分享2: BobParser	52
22. 结束语	53
附录: Bob Test Theroy	54

前言

本书将向读者介绍一种简单易行、实用高效的自动化测试方法。

自动化测试原本应该很容易，但是现在有了太多了概念名词和所谓的最佳实践，导致测试看起来很难。

如果你之前从未接触过敏捷开发、TDD（测试驱动开发）、单元测试、自动化测试等概念，那么本书会是一个很好的开始；如果你已经读了一些博客文章、专栏介绍，甚至书籍，仍然觉得无从下手，那么本书会是一个很好的示例，让你看到概念、理论是如何落地并融入代码的；如果你已经在实际项目中写过 `TestCase`，那么本书会给你提供一个全新的视角，看后会说：“原来这样写测试真的可以！”。

本书假设读者已经熟悉 PHP 的基本语法和应用，本书不是 PHP 语言的入门书籍。但本书不要求读者有任何 PHPUnit 相关使用经验，本书会从零开始介绍 PHPUnit 的安装使用，并以一个实际项目介绍如何应用 PHPUnit 实现自动化测试。

本书以 Ubuntu 18.04 LTS、Apache 2.4、MySQL 5.7、PHP 7.2、PHPUnit 8.5 为开发环境，如果读者熟悉 LAMP 环境的安装和配置，那么无论是 Windows 还是 Mac OS 应该都不是问题，否则，建议读者安装一个 Ubuntu 18.04 LTS 的虚拟机，这样会更容易一些。

关于作者

本书作者从2010年开始加入云招科技（北京）有限公司，现名云招四海网络（北京）有限公司从事 PHP 开发至今。云招是一家专注于为客户提供基于云计算的在线招聘管理解决方案（OurATS – Our Applicant Tracking System）的服务提供商，10余年来云招专注于开发 OurATS 招聘管理系统，现今该系统已服务于上千家企业。

作者从2012年开始探索 TDD（测试驱动开发）在 OurATS 项目中的应用。期间阅读了大量书籍和文章，还参加了知名公司的敏捷开发分享交流会，发现关于敏捷开发、TDD 的方法论较多，真正如何在实际项目中编写代码，将 TDD 持续进行下去的实战经验很少。最后探索出了一套简单易行、实用高效的自动化测试方法，并将这套方法成功应用于 OurATS 招聘管理系统，使得开发团队得以轻松应对多变的招聘需求，并保持系统稳定地迭代升级。

至今(2020.06),云招开发团队已为 OurATS 增加了 4300万行代码,同时也删除了 3400万行代码,现存近 900万行代码,分布在 34038 个代码文件里,运行环境也从 Ubuntu 12.04 LTS、Apache 2.2、MySQL 5.5、PHP 5.3、PHPUnit 4.8升级到了 Ubuntu 18.04 LTS、Apache 2.4、MySQL 5.7、PHP 7.2、PHPUnit 8.5,这一切很大程度上得益于 PHPUnit 的恰当运用。

Generator:

[GitStats](#) (version 2015.10.03), git version 2.17.1, gnuplot 5.2 patchlevel 2

Report Period:

2010-11-30 11:09:58 to 2020-06-24 12:23:41

Age:

3495 days, 2603 active days (74.48%)

Total Files:

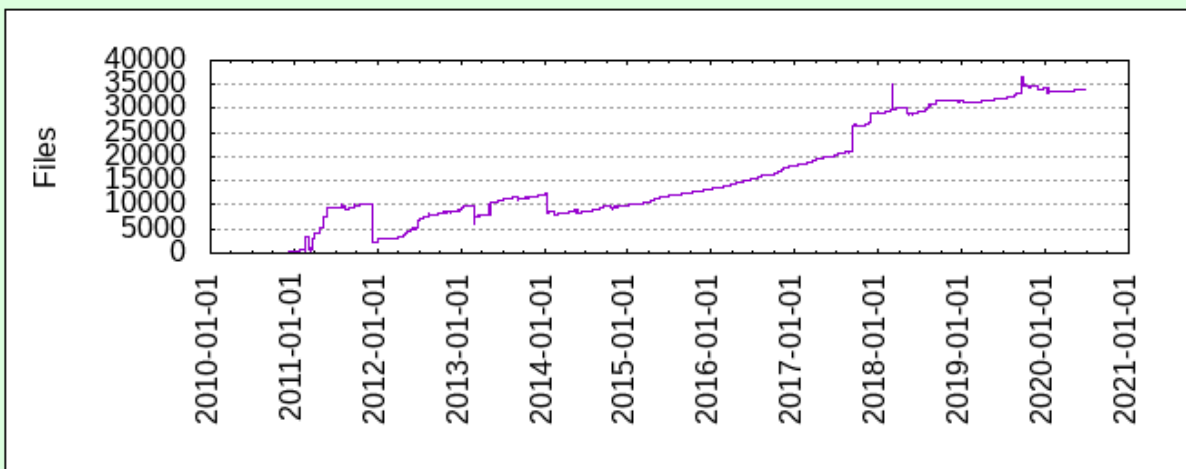
34038

Total Lines of Code:

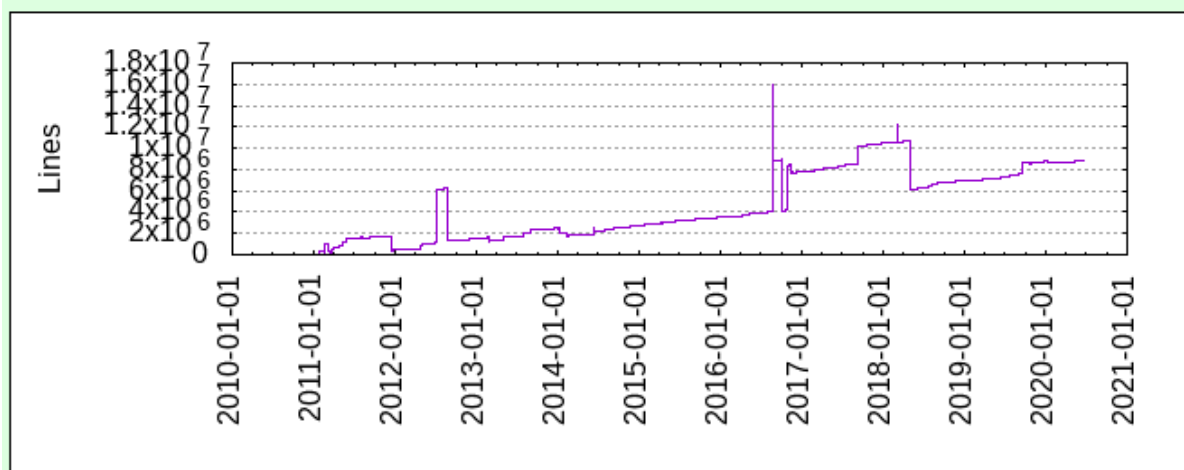
8896670 (43193953 added, 34297283 removed)

Total files:
34038
Total lines:
8896670
Average file size:
102216.37 bytes

File count by date



Lines of Code



1. PHPUnit Hello World

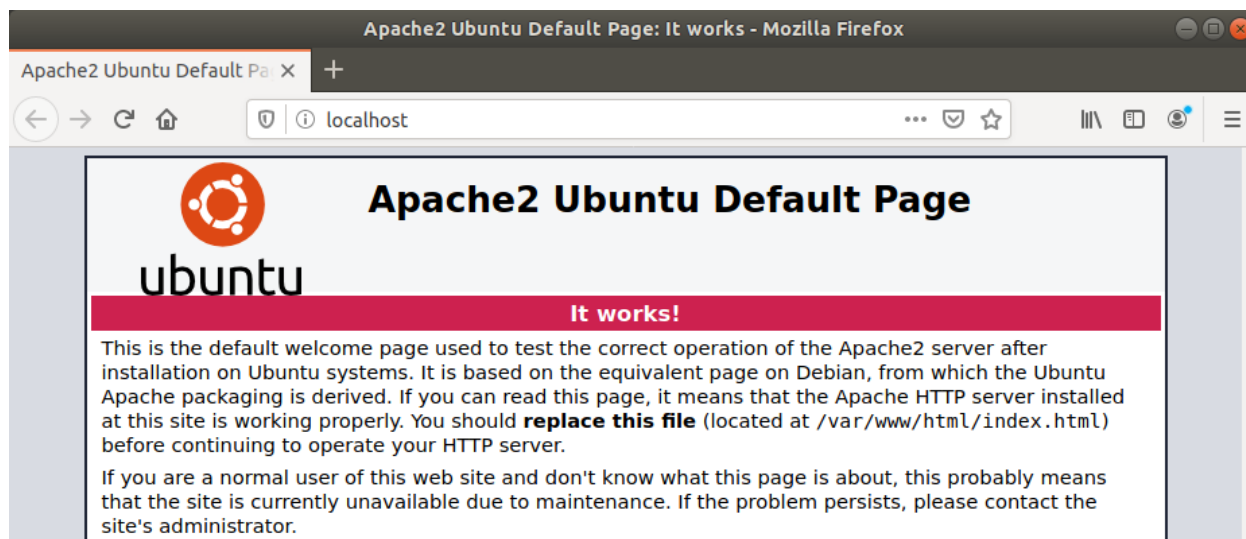
1.1 安装 LAMP 环境

在 Ubuntu 上安装 LAMP 环境非常简单，两条命令即可。

- 1 bob@Bob-VirtualBox:~\$ sudo apt install taskel
- 2 bob@Bob-VirtualBox:~\$ sudo taskel install lamp-server

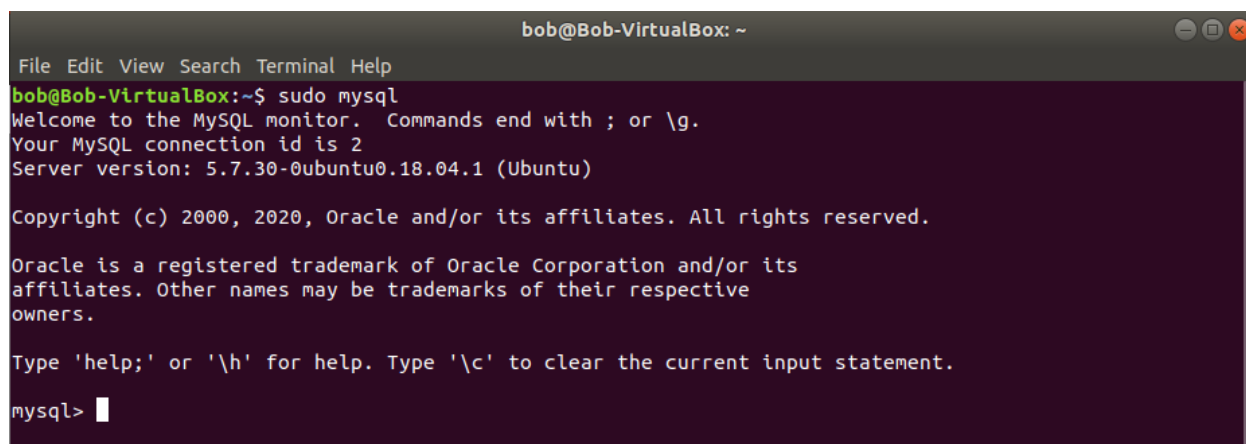
验证 Apache

浏览器打开 <http://localhost>



验证 MySQL

- 1 bob@Bob-VirtualBox:~\$ sudo mysql

A terminal window titled 'bob@Bob-VirtualBox: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The user runs 'sudo mysql', which prompts for a password. After login, it displays the MySQL welcome message, connection ID (2), and server version (5.7.30-0ubuntu0.18.04.1). It also shows copyright information and instructions on how to use help and clear the input. The prompt 'mysql>' is shown at the bottom.

```
bob@Bob-VirtualBox:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.30-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

验证 PHP

为了方便后续开发，我们将 Apache 默认目录 `/var/www` 的权限修从 `root` 修改为当前用户。

- 1 bob@Bob-VirtualBox:~\$ ls -ld /var/www/
- 2 drwxr-xr-x 3 root root 4096 Jun 29 15:56 /var/www/
- 3 bob@Bob-VirtualBox:~\$ sudo chown -R \$USER:\$USER /var/www/
- 4 bob@Bob-VirtualBox:~\$ ls -ld /var/www/
- 5 drwxr-xr-x 3 hgy hgy 4096 Jun 29 15:56 /var/www/

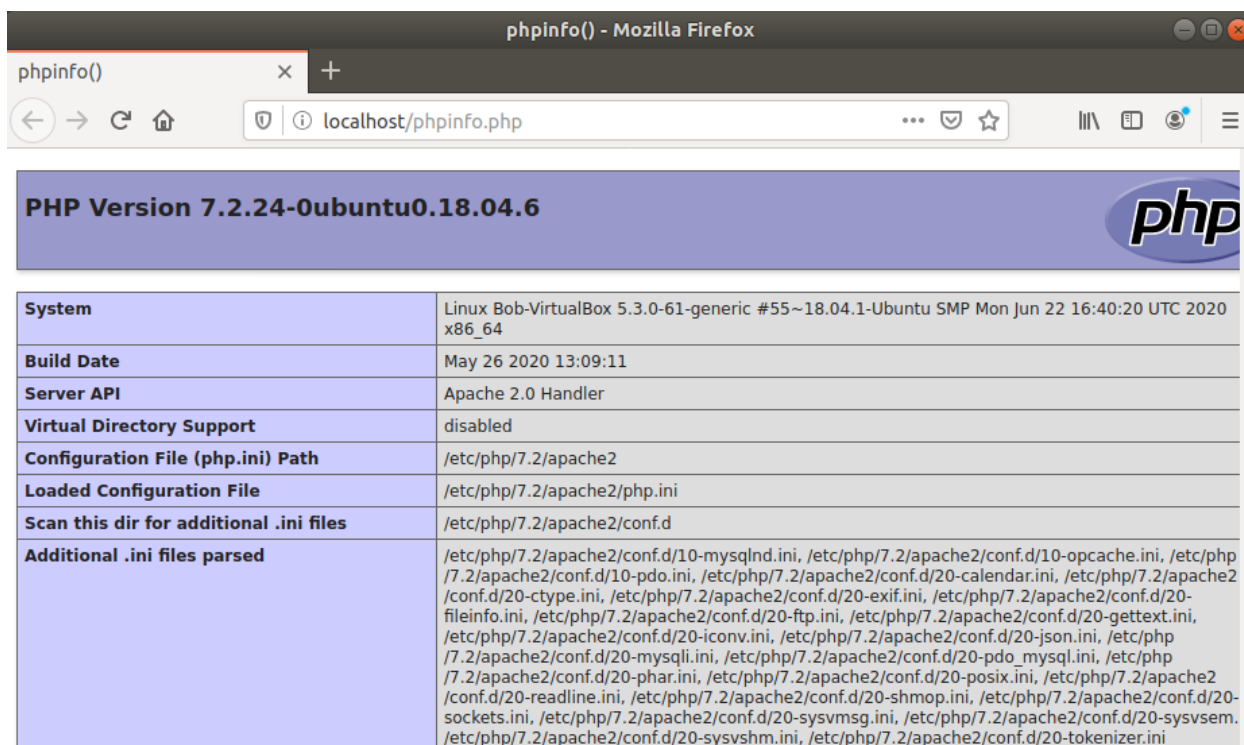
Apache 默认的 VirtualHost 配置把 DocumentRoot 指向了 `/var/www/html`，我们把它修改成 `/var/www`，这样我们可以在 `/var/www` 目录下建不同的目录，把不同项目的代码区分开。

- 1 bob@Bob-VirtualBox:~\$ sudo replace /var/www/html /var/www -- /etc/apache2/sites-available/000-default.conf
- 2
- 3 bob@Bob-VirtualBox:~\$ sudo systemctl reload apache2.service

我们使用 `phpinfo()` 函数来验证 PHP 是否安装好了。

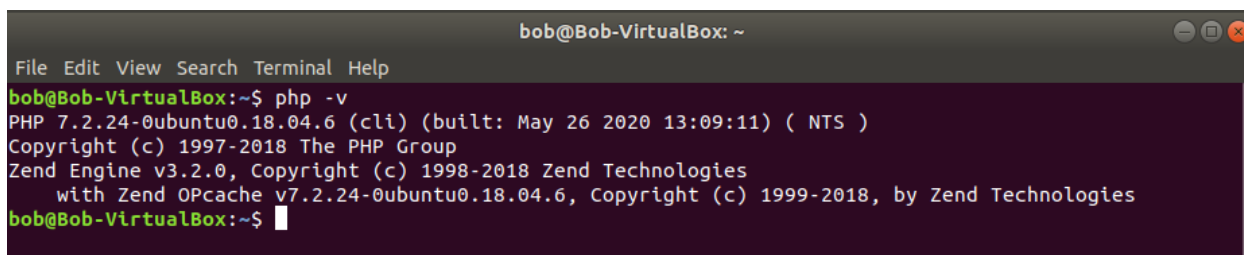
- 1 bob@Bob-VirtualBox:~\$ echo '<?php phpinfo();' > /var/www/phpinfo.php

浏览器打开 `http://localhost/phpinfo.php`



验证 PHP CLI

- 1 bob@Bob-VirtualBox:~\$ php -v



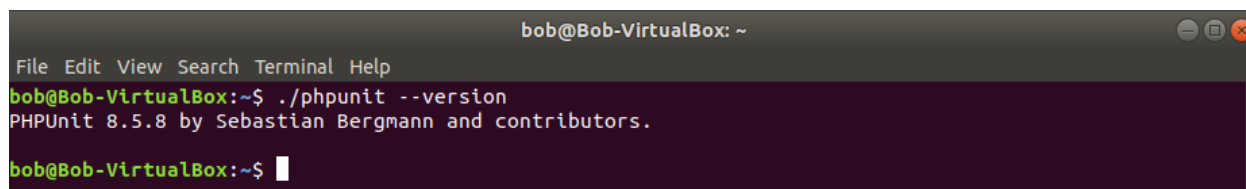
1.2 安装 PHPUnit

PHPUnit 的安装也非常简单，同样两条命令就安装好了。

- 1 bob@Bob-VirtualBox:~\$ wget -O phpunit https://phar.phpunit.de/phpunit-8.phar
- 2 bob@Bob-VirtualBox:~\$ chmod +x phpunit

验证 phpunit

- 1 `./phpunit --version`



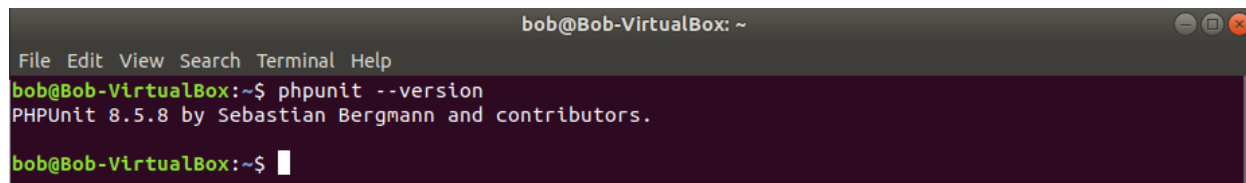
```
bob@Bob-VirtualBox: ~  
File Edit View Search Terminal Help  
bob@Bob-VirtualBox:~$ ./phpunit --version  
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.  
bob@Bob-VirtualBox:~$
```

现在使用 `phpunit` 必须输入相对路径或绝对路径，这样有点麻烦，因为我们会很多次地执行 `phpunit`。我们将其加入到当前用户的 `~/bin` 目录里，这样直接输入 `phpunit` 就可以了。

- 1 `bob@Bob-VirtualBox:~$ mkdir bin`
- 2 `bob@Bob-VirtualBox:~$ mv phpunit bin/`

退出（Logout）当前用户，再登录进来，或者直接重启一下系统。现在直接输入 `phpunit` 就可以了。

- 1 `bob@Bob-VirtualBox:~$ phpunit --version`



```
bob@Bob-VirtualBox: ~  
File Edit View Search Terminal Help  
bob@Bob-VirtualBox:~$ phpunit --version  
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.  
bob@Bob-VirtualBox:~$
```

1.3 Hello World

现在，让我们写一个简单的 Hello World 程序。

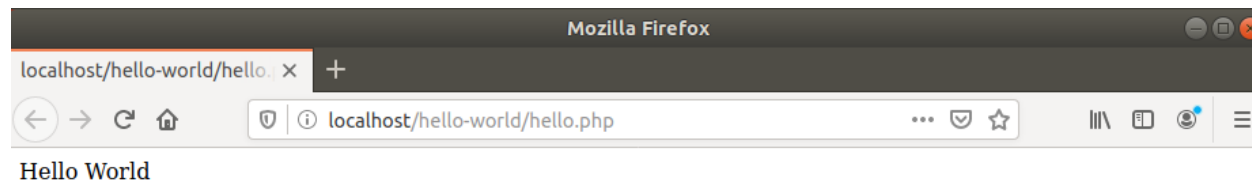
我们在 `/var/www` 目录下新建一个名为 `hello-world` 的子目录。

- 1 `bob@Bob-VirtualBox:~$ mkdir /var/www/hello-world`

然后用你喜欢的编辑器编写如下代码，保存到 `/var/www/hello-world/hello.php`。

- 1 `<?php`
- 2
- 3 `echo 'Hello World';`

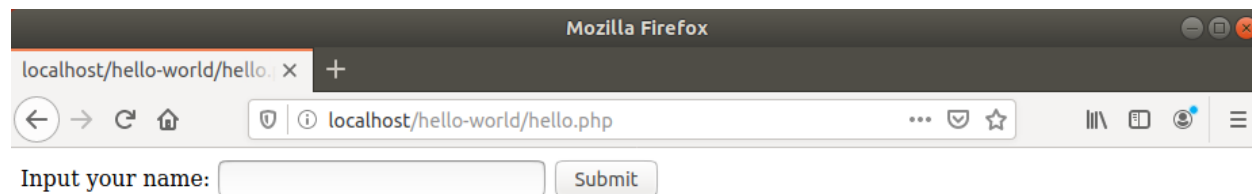
浏览器打开 <http://localhost/hello-world/hello.php>



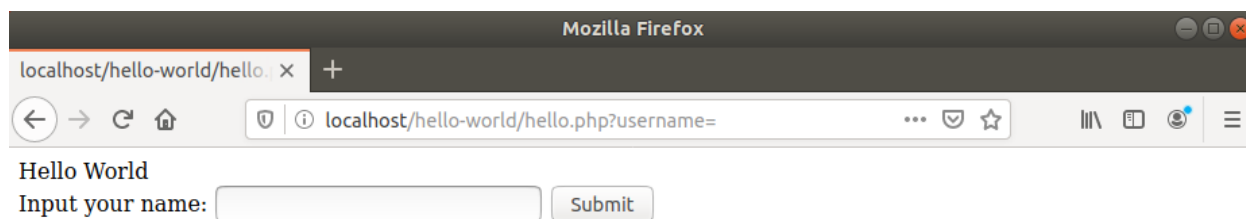
我们让这个 hello world 程序稍微复杂一点点。

```
1 <html>
2 <head>
3 <meta charset="utf-8">
4 </head>
5 <body>
6
7 <?php
8
9 if ($_GET) {
10     echo 'Hello ', isset($_GET['username']) && $_GET['username'] ? $_GET['username']\
11     : 'World';
12 }
13
14 ?>
15
16 <form>
17     Input your name: <input type="text" name="username">
18     <input type="submit" value="Submit">
19 </form>
20
21 </body>
22 </html>
```

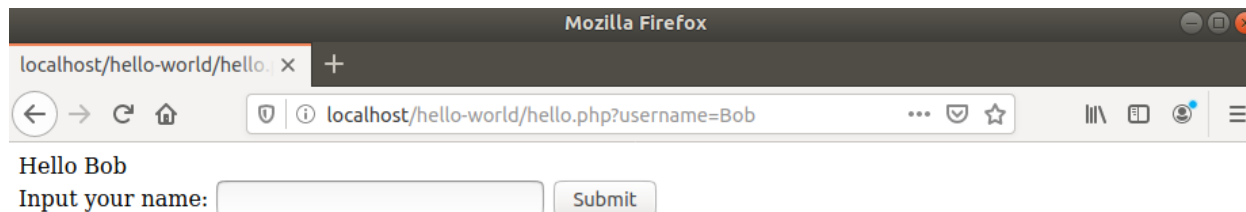
再次打开 <http://localhost/hello-world/hello.php>



直接点击 “Submit” 按钮



输入你的名字再次点击“Submit”按钮



OK，我们的 Hello World 程序运行地很好。

那么，怎么使用 PHPUnit 对其进行自动化测试呢？

作者确实看到过有人使用一些工具类库，通过写代码启动了一个浏览器，访问要测试的网址，然后判断打开的页面里是否有期望的内容。比如：

```
1 from selenium import webdriver
2
3 browser = webdriver.Firefox()
4 browser.get('http://localhost')
5
6 assert 'Hello' in browser.title
```

作者也看到一些 MVC 开发框架提供的测试方法，比如 Zend Framework 1 的 Zend_Test 类，在测试代码里直接驱动 MVC 框架运行，模拟浏览器的请求过程，最终获取到页面输出，然后同样判断页面里是否有期望的内容。

```
1 class UserControllerTest extends Zend_Test_PHPUnit_ControllerTestCase
2 {
3     public function setUp()
4     {
5         $this->bootstrap = array($this, 'appBootstrap');
6         parent::setUp();
7     }
8
9     public function appBootstrap()
10    {
```

```
11     $this->frontController
12     ->registerPlugin(new Bugapp_Plugin_Initialize('development'));
13 }
14
15 public function testValidLoginShouldGoToProfilePage()
16 {
17     $this->request->setMethod('POST')
18     ->setPost(array(
19         'username' => 'foobar',
20         'password' => 'foobar'
21     ));
22     $this->dispatch('/user/login');
23     $this->assertRedirectTo('/user/view');
24
25     $this->resetRequest()
26     ->resetResponse();
27
28     $this->request->setMethod('GET')
29     ->setPost(array());
30     $this->dispatch('/user/view');
31     $this->assertRoute('default');
32     $this->assertModule('default');
33     $this->assertController('user');
34     $this->assertAction('view');
35     $this->assertNotRedirect();
36     $this->assertQuery('dl');
37     $this->assertQueryContentContains('h2', 'User: foobar');
38 }
39 }
```

这些方法给人的第一印象是 沉重。我们不打算这样做。

作者在2012年刚开始探索 TDD 时，就使用过 Zend_Test，并且不只是写个 Hello World，而是真实地应用在了 OurATS 招聘管理系统的一个功能上。

这种测试方法看起来简单，实战时能把开发者累死（心累）。

主要在三方面：

1. 开发者要了解 MVC 框架运行的细节，这样才能把浏览器请求转换为驱动 MVC 框架运行的代码。
2. 开发者需要查看最终生成的 html 代码，然后才能精确定位期望的内容在哪里。并且如果页面改版，虽然呈现的主要内容还在，但显示的位置、样式都已经发生了变化，测试代码就跑不过了，要重写。

3. 测试写起来很繁琐。因为期望的内容往往不是一个点，而每个点除了本身的内容外，还需要考虑如何定位到这个点。

我们的做法是，通过重新组织代码，提炼出核心逻辑，让其易于测试。然后只测试这个核心逻辑，而核心逻辑之外的代码，并不通过 PHPUnit 测试。

这种做法看似不严谨，因为我们没有测试完整的功能，但实际上却是测试得以落地的关键。具体为什么，我们先不展开分析，本书将通过实战项目让读者领会这一点。

1.4 Hello World 重构

hello.php 核心逻辑是：

接收用户的输入，如果用户输入了 username，就返回 Hello username，否则，返回 Hello World。

新建一个文件 /var/www/hello-world/func.php

```
1 <?php
2
3 function sayHello(array $data)
4 {
5     if (isset($data['username']) && $data['username']) {
6         return 'Hello ' . $data['username'];
7     }
8
9     return 'Hello World';
10 }
```

在 hello.php 中调用 sayHello

```
1 <html>
2 <head>
3 <meta charset="utf-8">
4 </head>
5 <body>
6
7 <?php
8
9 if ($_GET) {
10     include __DIR__ . '/func.php';
11     echo sayHello($_GET);
12 }
```

```
13
14 ?>
15
16 <form>
17     Input your name: <input type="text" name="username">
18     <input type="submit" value="Submit">
19 </form>
20
21 </body>
22 </html>
```

再次访问 <http://localhost/hello-world/hello.php>，直接点击“Submit”，输入名字再次点击“Submit”，和重构前运行的一样，完全没问题。

1.5 Hello World Test

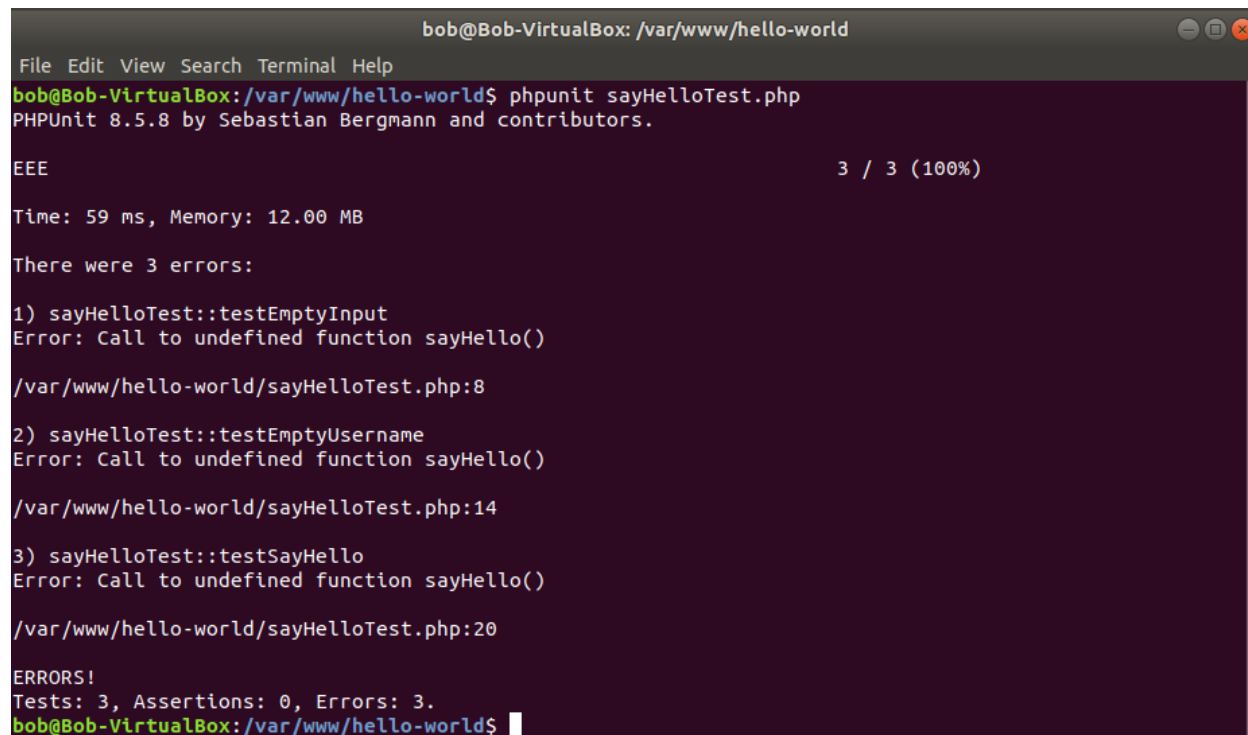
现在，我们可以为这个 Hello World 程序写测试了。

新建一个文件 `/var/www/hello-world/sayHelloTest.php`

```
1 <?php
2
3 class sayHelloTest extends PHPUnit\Framework\TestCase
4 {
5     public function testEmptyInput()
6     {
7         $data = array();
8         $this->assertEquals('Hello World', sayHello($data));
9     }
10
11     public function testEmptyUsername()
12     {
13         $data = array('username' => '');
14         $this->assertEquals('Hello World', sayHello($data));
15     }
16
17     public function testSayHello()
18     {
19         $data = array('username' => 'Bob');
20         $this->assertEquals('Hello Bob', sayHello($data));
21     }
22 }
```

然后执行这个测试

- 1 bob@Bob-VirtualBox:~\$ `cd /var/www/hello-world/`
- 2 bob@Bob-VirtualBox:/var/www/hello-world\$ `phpunit sayHelloTest.php`



```
bob@Bob-VirtualBox: /var/www/hello-world
File Edit View Search Terminal Help
bob@Bob-VirtualBox:/var/www/hello-world$ phpunit sayHelloTest.php
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.

EEE                                                                    3 / 3 (100%)

Time: 59 ms, Memory: 12.00 MB

There were 3 errors:

1) sayHelloTest::testEmptyInput
Error: Call to undefined function sayHello()

/var/www/hello-world/sayHelloTest.php:8

2) sayHelloTest::testEmptyUsername
Error: Call to undefined function sayHello()

/var/www/hello-world/sayHelloTest.php:14

3) sayHelloTest::testSayHello
Error: Call to undefined function sayHello()

/var/www/hello-world/sayHelloTest.php:20

ERRORS!
Tests: 3, Assertions: 0, Errors: 3.
bob@Bob-VirtualBox:/var/www/hello-world$
```

报错了 Error: Call to undefined function sayHello() 。
这是因为我们的测试代码里没有 `include 'func.php';` 。
我们可以使用 `phpunit` 的参数 `--bootstrap` 来解决这个问题。

- 1 `--bootstrap <file>` A PHP script that is included before the tests run

`/var/www/hello-world/bootstrap.php`

- 1 `<?php`
- 2
- 3 `include __DIR__ . '/func.php';`

加上 `--bootstrap` 参数再次执行

- 1 bob@Bob-VirtualBox:/var/www/hello-world\$ `phpunit --bootstrap=bootstrap.php sayHelloTest.php`
- 2

```
bob@Bob-VirtualBox: /var/www/hello-world
File Edit View Search Terminal Help
bob@Bob-VirtualBox:/var/www/hello-world$ phpunit --bootstrap=bootstrap.php sayHelloTest.php
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.

... 3 / 3 (100%)

Time: 61 ms, Memory: 10.00 MB

OK (3 tests, 3 assertions)
bob@Bob-VirtualBox:/var/www/hello-world$
```

1.6 phpunit.xml

在上一小节, 我们要运行 `sayHelloTest.php`, 需要将其作为命令行参数传递给 `phpunit`, 并且还要加上参数 `--bootstrap=bootstrap.php`, 如果要运行多次的话, 就太不方便了。 `phpunit` 可以通过配置文件 `phpunit.xml` 使得我们只需要在当前目录执行 `phpunit` 就可以运行该目录下的所有测试。

新建文件 `/var/www/hello-world/phpunit.xml`

```
1 <phpunit bootstrap="bootstrap.php" stopOnFailure="true" cacheResult="false">
2   <testsuites>
3     <testsuite name="All">
4       <directory>.</directory>
5     </testsuite>
6   </testsuites>
7 </phpunit>
```

`stopOnFailure` 的作用是如果遇到失败的测试 `phpunit` 是继续执行下去, 还是立即停止执行, 我们将其设为 `true`, 这样方便我们修正错误。

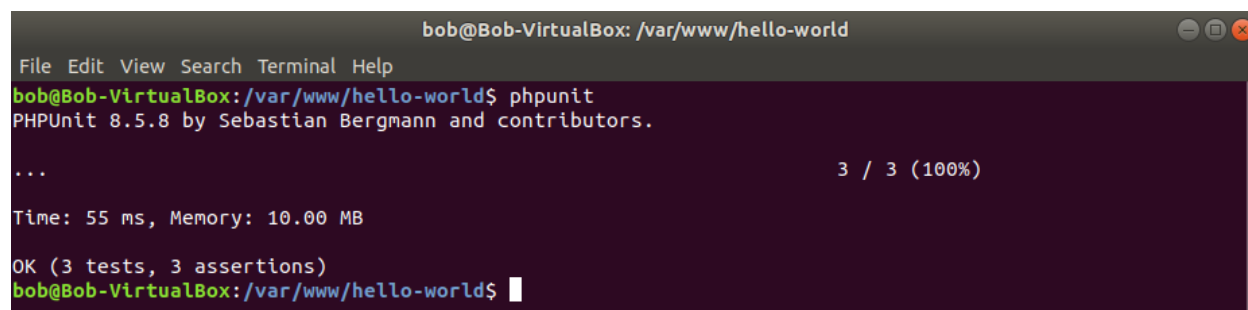
`cacheResult` 的作用 `phpunit` 的使用文档里说明的有点模糊, 默认值为 `true`, 导致运行 `phpunit` 后, 会在当前目录生成一个隐藏文件 `.phpunit.result.cache`, 我们先把它设为 `false`, 如果后边有用到和 `cache` 相关的功能, 再把它打开。

`testsuite` 是把多个测试组织在一块的一个办法, 对于大多数项目, 我们可以直接把所有测试都归到一个 `testsuite` 里去, 这里我们把当前目录下的所有测试 (实际上只有一个 `sayHelloTest.php`) 归到一个名为 `All` 的 `testsuite` 里。

`phpunit` 依赖 `php-xml` 扩展来读取 `phpunit.xml`, Ubuntu 上执行下边的命令来安装 `php-xml` 扩展。

```
1 bob@Bob-VirtualBox:/var/www/hello-world$ sudo apt install php-xml
```

我们再次运行下测试, 这次只用执行 `phpunit` 就好了, `phpunit` 会在当前目录下寻找 `*Test.php` 文件, 然后自动执行里边的测试方法。

A screenshot of a terminal window titled 'bob@Bob-VirtualBox: /var/www/hello-world'. The terminal shows the command 'phpunit' being executed. The output indicates that PHPUnit 8.5.8 by Sebastian Bergmann and contributors was used. It shows '3 / 3 (100%)' tests passed. The execution time was 55 ms and memory usage was 10.00 MB. The final status is 'OK (3 tests, 3 assertions)'. The prompt 'bob@Bob-VirtualBox: /var/www/hello-world\$' is visible at the bottom.

```
bob@Bob-VirtualBox: /var/www/hello-world
File Edit View Search Terminal Help
bob@Bob-VirtualBox: /var/www/hello-world$ phpunit
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.

... 3 / 3 (100%)

Time: 55 ms, Memory: 10.00 MB

OK (3 tests, 3 assertions)
bob@Bob-VirtualBox: /var/www/hello-world$
```

1.7 本章小结

本章我们准备好了基础的开发测试环境，并通过一个 Hello World 程序引出了作者的一个基础测试理论。

作者有一个英文名叫 Bob，我们暂且把这个理论称作 **Bob Test Theory 1**：

通过重新组织代码，提炼出核心逻辑，让其易于测试。然后只测试这个核心逻辑，而核心逻辑之外的代码，并不通过 PHPUnit 测试。

然后我们写了3个测试用例来测试 `sayHello(array $data)`，并通过使用 `phpunit.xml` 简化了测试的执行。

本章代码

<https://github.com/heguangyu5/PHPUnit-in-Action-Code/tree/chapter01/hello-world>¹

¹<https://github.com/heguangyu5/PHPUnit-in-Action-Code/tree/chapter01/hello-world>

2. 继续 Hello World

上一章的 Hello World 程序，我们只验证了用户输入的 username 不为空，在真实的项目里，这显然是不够的。

本章让我们通过完善 username 的验证来继续学习 PHPUnit。

2.1 测试异常

假设我们要求 username 的长度要在 2 ~ 30 个字符之间，并且只能包含英文大小写字母、数字、空格，其它的字符均不允许出现。

/var/www/ourblog/hello-world/func.php

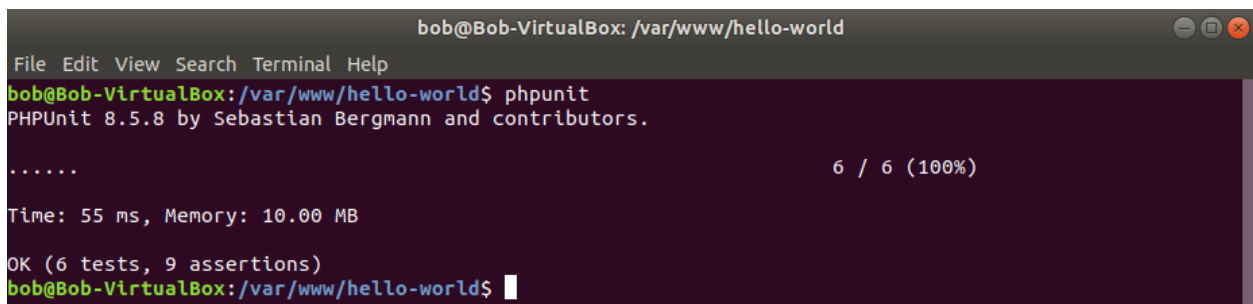
```
1 <?php
2
3 function sayHello(array $data)
4 {
5     if (isset($data['username']) && $data['username']) {
6         if (!preg_match('/^[a-zA-z0-9 ]{2,30}$/', $data['username'])) {
7             throw new InvalidArgumentException("invalid username, should 2 ~ 30 char\
8 actors and only contains a-z, A-Z, 0-9 and space.");
9         }
10        return 'Hello ' . $data['username'];
11    }
12
13    return 'Hello World';
14 }
```

PHPUnit 提供了 expectException() 方法来测试异常。

新建文件 /var/www/hello-world/ExceptionTest.php

```
1 <?php
2
3 class ExceptionTest extends PHPUnit\Framework\TestCase
4 {
5     public function testTooShortUsername()
6     {
7         $this->expectException(InvalidArgumentException::class);
8         $this->expectExceptionMessage('invalid username, should 2 ~ 30 characters an\
9 d only contains a-z, A-Z, 0-9 and space.');
```

运行 phpunit



```

bob@Bob-VirtualBox: /var/www/hello-world
File Edit View Search Terminal Help
bob@Bob-VirtualBox: /var/www/hello-world$ phpunit
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.

.....                                     6 / 6 (100%)

Time: 55 ms, Memory: 10.00 MB

OK (6 tests, 9 assertions)
bob@Bob-VirtualBox: /var/www/hello-world$

```

上一章我们写了3个测试用例，每个测试用例里有1个 `assert`，这次的这个 `ExceptionTest` 里我们又写了3个测试用例，每个测试用例里有2个 `expect`，加起来一共 6 tests, 9 assertions。

读者也许意识到了，我们新写的这3个测试用例除了输入参数不一样，其它的代码都是一样的。那么有没有办法简化测试代码呢？

2.2 @dataProvider

PHPUnit 提供了 `@dataProvider` 注释来简化上一小节我们遇到的情况。

新建文件 `/var/www/hello-world/DataProviderTest.php`

```

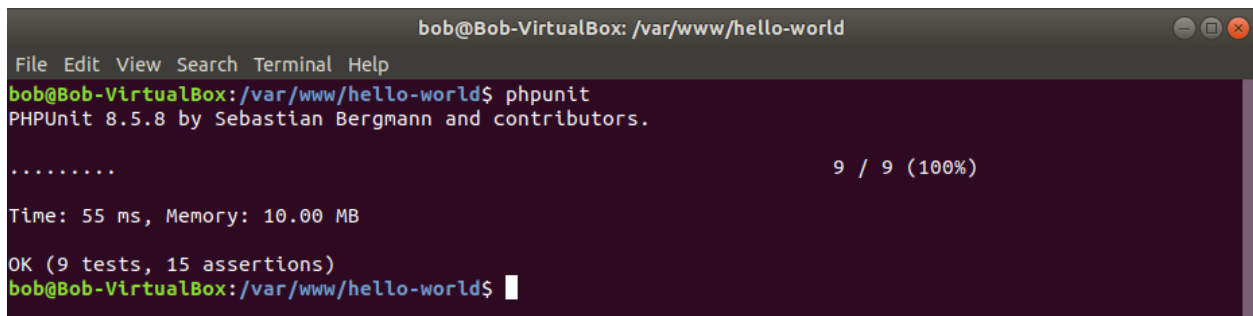
1  <?php
2
3  class DataProviderTest extends PHPUnit\Framework\TestCase
4  {
5      /**
6       * @dataProvider invalidUsernameProvider
7       */
8      public function testInvalidUsername($invalidUsername)
9      {
10         $this->expectException(InvalidArgumentException::class);
11         $this->expectExceptionMessage('invalid username, should 2 ~ 30 characters and
12         only contains a-z, A-Z, 0-9 and space.');
```

```

22     array(str_pad('a', 31, 'A')),
23     array('Bob!')
24 );
25 }
26 }

```

运行 phpunit



The screenshot shows a terminal window titled 'bob@Bob-VirtualBox: /var/www/hello-world'. The terminal output displays the execution of 'phpunit' in the directory '/var/www/hello-world'. It reports 'PHPUnit 8.5.8 by Sebastian Bergmann and contributors.' followed by a progress bar '.....' and '9 / 9 (100%)'. Below this, it shows 'Time: 55 ms, Memory: 10.00 MB' and 'OK (9 tests, 15 assertions)'. The prompt 'bob@Bob-VirtualBox: /var/www/hello-world\$' is visible at the bottom.

现在我们有 9 tests, 15 assertions。

回过头来再看下 sayHelloTest，读者应该意识到那3个测试用例也可以使用 @dataProvider 简化。

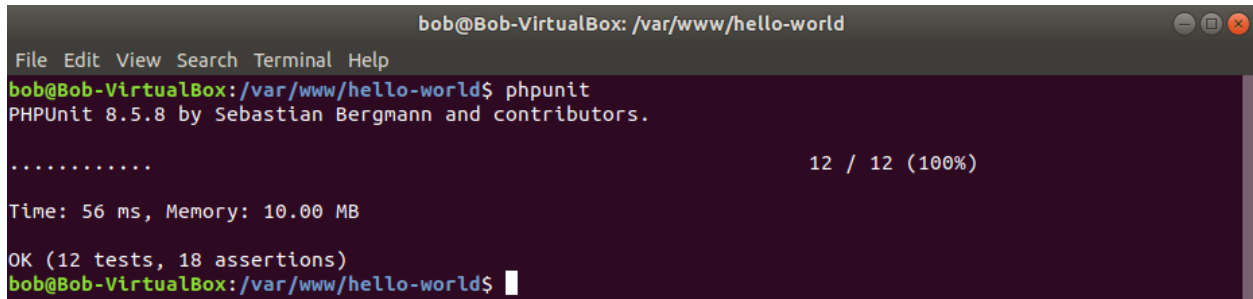
新建文件 /var/www/hello-world/sayHelloWithDataProviderTest.php

```

1  <?php
2
3  class sayHelloWithDataProviderTest extends PHPUnit\Framework\TestCase
4  {
5      /**
6       * @dataProvider validUsernameProvider
7       */
8      public function testEmptyInput($data, $ret)
9      {
10         $this->assertEquals($ret, sayHello($data));
11     }
12
13     public function validUsernameProvider()
14     {
15         return array(
16             array(array(), 'Hello World'),
17             array(array('username' => ''), 'Hello World'),
18             array(array('username' => 'Bob'), 'Hello Bob')
19         );
20     }
21 }

```

运行 phpunit



```
bob@Bob-VirtualBox: /var/www/hello-world
File Edit View Search Terminal Help
bob@Bob-VirtualBox:/var/www/hello-world$ phpunit
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.

.....                                     12 / 12 (100%)

Time: 56 ms, Memory: 10.00 MB

OK (12 tests, 18 assertions)
bob@Bob-VirtualBox:/var/www/hello-world$
```

现在我们有 12 tests, 18 assertions。

2.3 @group

到上一小节结尾，我们一共写了 12 个测试用例，每次执行 phpunit 时，这 12 个测试用例都会跑一遍。在一个稍大点的项目里，会有成百上千的测试用例，全部运行下来会花点时间，也许几十秒，也许几分钟。

有没有办法只运行当前正在开发的功能块的测试，而不运行其它的呢？

PHPUnit 提供了 @group 注释来达到这一目的。

作者特别提示：

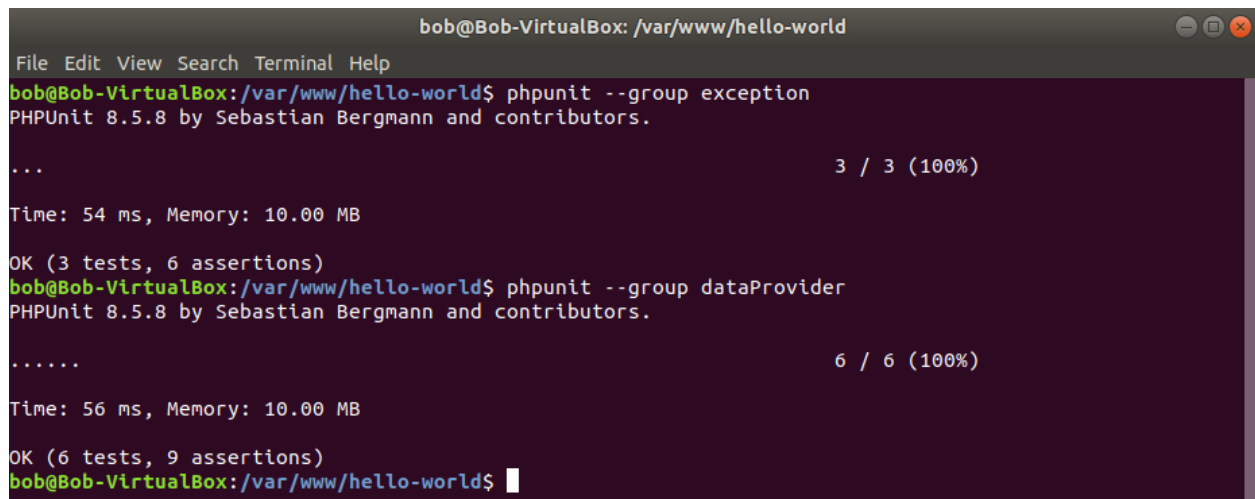
如果所有测试跑的足够快，能在几秒内结束，就没必要使用 @group 来单独跑某几个测试。毕竟自动化测试的一个很重要的目的就是回归测试，确保新的修改不会影响到原来的功能。

不管所有测试跑的快慢，即使开发者可以在开发过程中使用 @group，但在将新的修改提交到代码库之前，也应该完整的跑一次测试，确保新的修改没有影响到已有的功能。

我们给 ExceptionTest 加上 @group exception 注释，给 DataProviderTest 的 sayHelloWithDataProviderTest 加上 @group dataProvider 注释。

```
1 <?php
2 /**
3  * @group exception
4  */
5 class ExceptionTest extends PHPUnit\Framework\TestCase
6 {
7     // ...
8 }
9 <?php
10 /**
11  * @group dataProvider
12  */
13 class DataProviderTest extends PHPUnit\Framework\TestCase
14 {
15     // ...
16 }
17 <?php
18 /**
19  * @group dataProvider
20  */
21 class sayHelloWithDataProviderTest extends PHPUnit\Framework\TestCase
22 {
23     // ...
24 }
```

然后分别运行它们。



```
bob@Bob-VirtualBox: /var/www/hello-world
File Edit View Search Terminal Help
bob@Bob-VirtualBox:/var/www/hello-world$ phpunit --group exception
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.

...                                     3 / 3 (100%)

Time: 54 ms, Memory: 10.00 MB

OK (3 tests, 6 assertions)
bob@Bob-VirtualBox:/var/www/hello-world$ phpunit --group dataProvider
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.

.....                                6 / 6 (100%)

Time: 56 ms, Memory: 10.00 MB

OK (6 tests, 9 assertions)
bob@Bob-VirtualBox:/var/www/hello-world$
```

2.4 Code Coverage

PHPUnit 结合 php-xdebug 扩展可以提供代码测试覆盖率报告。

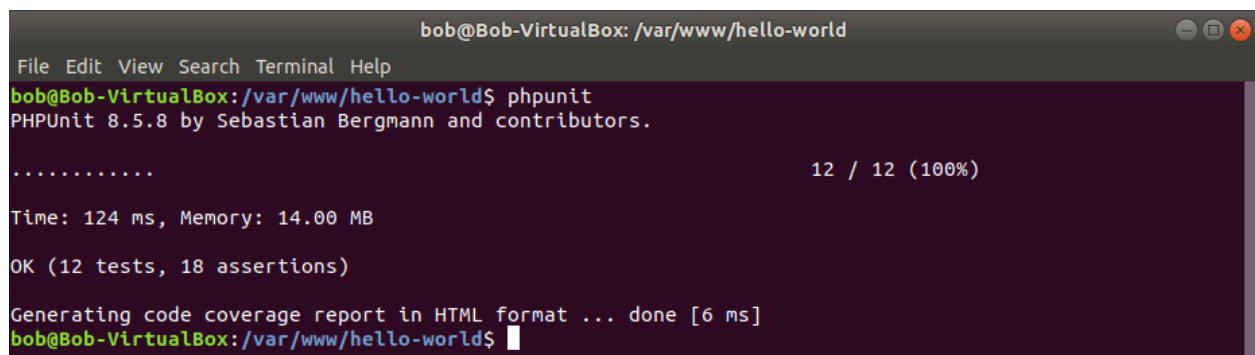
首先，我们要安装 php-xdebug 。

```
1 bob@Bob-VirtualBox:~$ sudo apt install php-xdebug
```

然后修改 phpunit.xml 告诉 phpunit 我们要生成代码测试覆盖率报告。

```
1 <phpunit bootstrap="bootstrap.php" stopOnFailure="true" cacheResult="false">
2   <testsuites>
3     <testsuite name="All">
4       <directory>./</directory>
5     </testsuite>
6   </testsuites>
7   <filter>
8     <whitelist>
9       <file>func.php</file>
10    </whitelist>
11  </filter>
12  <logging>
13    <log type="coverage-html" target="report"/>
14  </logging>
15 </phpunit>
```

运行 phpunit

A terminal window titled 'bob@Bob-VirtualBox: /var/www/hello-world' showing the output of the 'phpunit' command. The output includes the PHPUnit version (8.5.8), the number of tests and assertions (12 tests, 18 assertions), the execution time (124 ms), memory usage (14.00 MB), and the generation of a code coverage report in HTML format. The terminal background is dark purple with light green text.

```
bob@Bob-VirtualBox: /var/www/hello-world
File Edit View Search Terminal Help
bob@Bob-VirtualBox:/var/www/hello-world$ phpunit
PHPUnit 8.5.8 by Sebastian Bergmann and contributors.

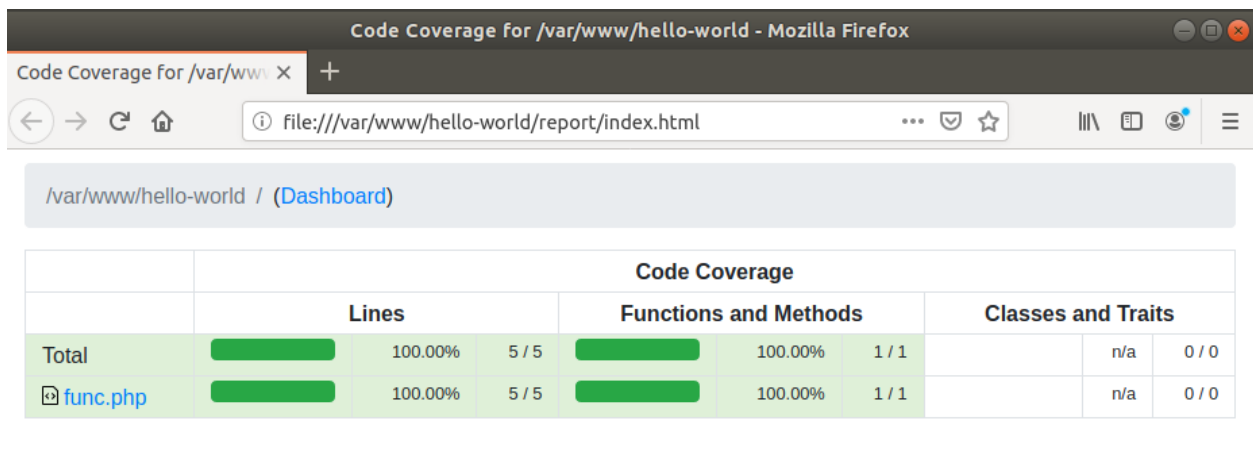
.....                                     12 / 12 (100%)

Time: 124 ms, Memory: 14.00 MB

OK (12 tests, 18 assertions)

Generating code coverage report in HTML format ... done [6 ms]
bob@Bob-VirtualBox:/var/www/hello-world$
```

浏览器打开 `/var/www/hello-world/report/index.html`



Legend

Low: 0% to 50% **Medium:** 50% to 90% **High:** 90% to 100%

Generated by [php-code-coverage 7.0.10](#) using [PHP 7.2.24-0ubuntu0.18.04.6](#) with [Xdebug 2.6.0](#) and [PHPUnit 8.5.8](#) at Thu Jul 2 17:11:12 CST 2020.

Code Coverage for /var/www/hello-world/func.php - Mozilla Firefox

Code Coverage for /var/www/ X +

file:///var/www/hello-world/report/func.php.html

/var/www/hello-world / func.php

Code Coverage										
Classes and Traits				Functions and Methods				Lines		
Total		n/a	0 / 0		100.00%	1 / 1	CRAP		100.00%	5 / 5
sayHello					100.00%	1 / 1	4		100.00%	5 / 5

```

1  <?php
2
3  function sayHello(array $data)
4  {
5      if (isset($data['username']) && $data['username']) {
6          if (!preg_match('/^[a-zA-z0-9 ]{2,30}$/', $data['username'])) {
7              throw new InvalidArgumentException('invalid username, should 2 ~ 30 characters');
8          }
9          return 'Hello ' . $data['username'];
10     }
11
12     return 'Hello World';
13 }

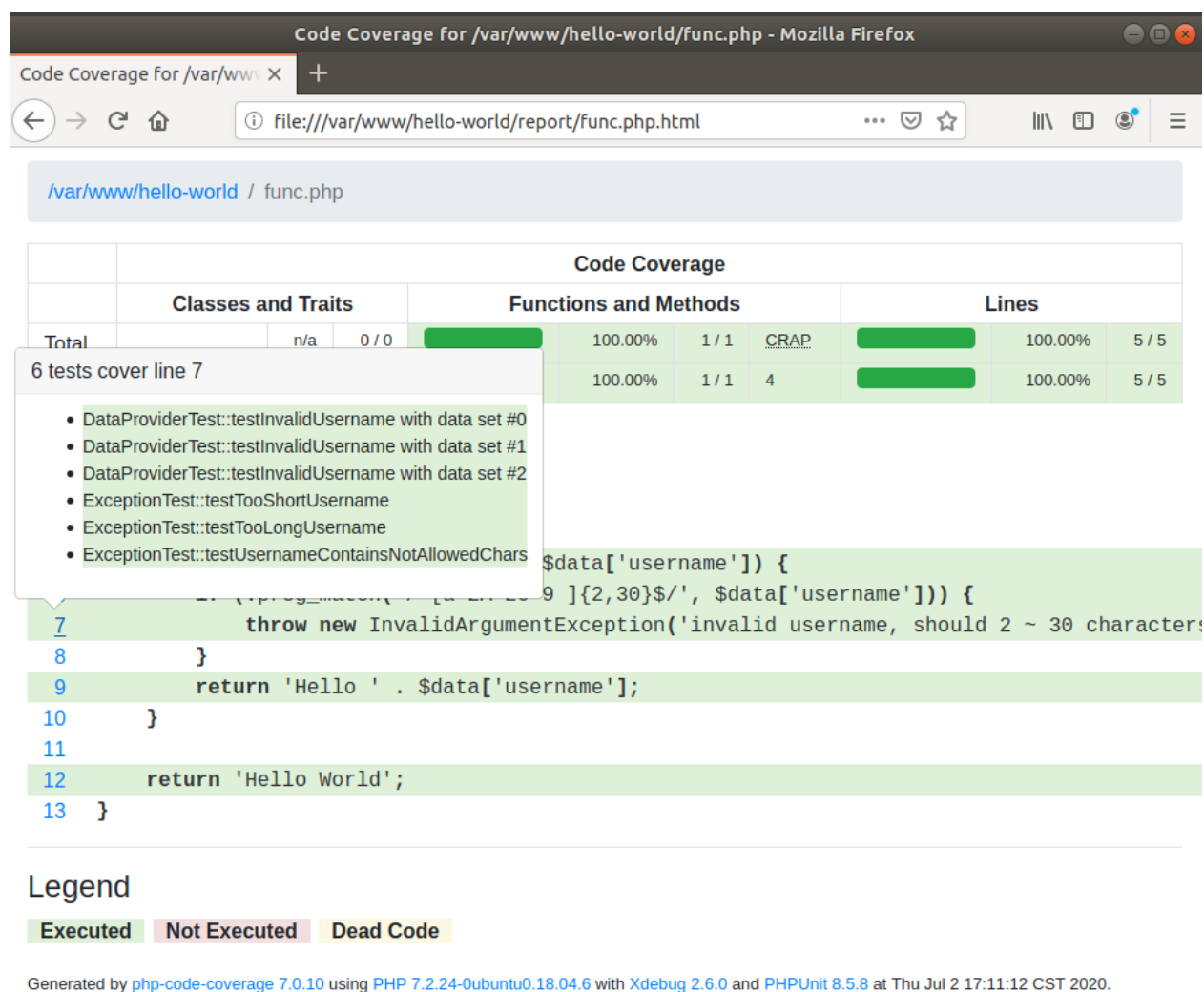
```

Legend

Executed Not Executed Dead Code

Generated by [php-code-coverage 7.0.10](#) using [PHP 7.2.24-0ubuntu0.18.04.6](#) with [Xdebug 2.6.0](#) and [PHPUnit 8.5.8](#) at Thu Jul 2 17:11:12 CST 2020.

鼠标放到代码上，还能看到这行代码都被哪几个测试用例覆盖到了。



2.5 关于 Code Coverage 的进一步思考

现在让我们仔细看一下验证 username 的正则表达式：

```
/^a-zA-z0-9 $/
```

再来回顾下我们写的测试用例，会发现：

1. 非法字符我们仅测试了一个“!”号
2. 有效字符空格我们没有测试

可是我们的测试覆盖率已经 100% 了！

甚至我们将非法字符的测试去掉，再去掉一个长度的测试后，覆盖率依旧是 100% ！

这说明了什么？

Bob Test Theory 2

测试覆盖率是一个参考，是开发者的一个辅助工具，开发人员可以在开发过程中，利用测试覆盖率检查疏漏，补全测试。把测试覆盖率作为一个指标强行要求开发者达到高覆盖率，除了引起开发者的反感外，恐怕没别的好处。

PHPUnit 也提供了几种方法让一段代码不包含在覆盖率统计范围内，所以开发者想要达到高覆盖率是很容易的。那些妄图在覆盖率上打开发者主意的，就省省吧。

2.6 PHPUnit 手册

本书并不打算翻译或者精简重述 [PHPUnit 手册](#)²，那叫”嚼剩饭“。

我们通过这个简单的 Hello World 程序已经介绍了大部分常用的 PHPUnit 功能。还有两个重要的功能 Fixtures 和 Mock 通过这个 Hello World 无法展示了，我们将其留到实战项目里。

PHPUnit 提供了很多方便的 assertion，读者请查阅 [PHPUnit 手册: Assertions](#)³ 大概过一遍，留个印象，在需要时再仔细翻看其用法。

如果读者有耐心，也可以翻阅下手册中的其它章节。PHPUnit 提供的功能很丰富，但在实际项目中，往往用不了那么多。

Bob Test Theory 3

我们一定不要为了测试而测试，不要为了 TDD 而 TDD。

写测试、TDD 都是手段，目的是要解放开发者，让开发人员有更多的时间去思考，去写出更高质量、更易维护的代码。

2.7 本章小结

本章我们结束了 PHPUnit Hello World 程序的测试，介绍了如何测试异常，如何使用 @dataProvider 简化测试，如何使用 @group 选择性地执行测试，并提出了两个新的测试理论：

1. 测试覆盖率应该是开发者的辅助工具，而不是管理人员用来考核的指标。
2. 自动化测试是为了解放开发者，因此要奔着易学易写的方向去，不要复杂化。

本章代码

<https://github.com/heguangyu5/PHPUnit-in-Action-Code/tree/chapter02/hello-world>⁴

²<https://phpunit.readthedocs.io/en/8.5/>

³<https://phpunit.readthedocs.io/en/8.5/assertions.html>

⁴<https://github.com/heguangyu5/PHPUnit-in-Action-Code/tree/chapter02/hello-world>

3. 实战项目 OurBlog 第一阶段介绍

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

3.1 用户注册

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

3.2 用户登录

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

3.3 后台首页

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

3.4 添加文章

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

3.5 编辑文章

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

3.6 删除文章

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

3.7 前台首页

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

3.8 文章详情页面

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

3.9 用户退出

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

3.10 本章小结

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

4. OurBlog 目录结构及数据库

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

4.1 OurBlog 目录结构

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

4.2 OurBlog 数据库

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

4.3 本章小结

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

5. OurBlog 用户注册

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

5.1 注册表单

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

5.2 提交表单

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

5.3 autoload

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

5.4 测试提交表单逻辑

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

5.5 reg 准备

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

5.6 reg

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

5.7 测试要连着数据库测吗？

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

5.8 DbUnit

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

5.9 phpunit-no-namespace

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

5.10 reg tests 准备

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

5.11 reg tests

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

5.12 再次测试提交注册表单

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

5.13 本章小结

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

6. OurBlog 用户登录

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

6.1 登录页面逻辑

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

6.2 测试登录逻辑

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

6.3 auth

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

6.4 auth tests 准备

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

6.5 auth tests

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

6.6 本章小结

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

7. OurBlog 后台首页

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

7.1 check login

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

7.2 header footer

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

7.3 后台首页

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

7.4 本章小结

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

8. OurBlog 添加文章

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

8.1 添加表单

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

8.2 测试添加表单

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

8.3 add

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

8.4 add tests

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

8.5 本章小结

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

9. OurBlog 编辑文章

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

9.1 编辑表单

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

9.2 edit

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

9.3 edit tests

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

9.4 本章小结

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

10. OurBlog 删除文章

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

10.1 delete

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

10.2 delete tests

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

10.3 本章小结

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

11. OurBlog 前台首页

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

11.1 header footer

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

11.2 文章列表

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

11.3 本章小结

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

12. OurBlog 文章详情页面

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

13. OurBlog 用戶退出

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

14. 实战项目 OurBlog 第二阶段介绍

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

14.1 用户注册发送激活邮件

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

14.2 增加打标签功能

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

14.3 增加文件上传功能

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

14.4 增加外部文章功能

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

14.5 本章小结

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

15. OurBlog 用户注册发送激活邮件

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

15.1 邮件发送方式？

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

15.2 怎么判断用户是否已激活？

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

15.3 生成随机 token

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

15.4 实现发送激活邮件逻辑

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

15.5 写测试

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

15.6 注册成功提示

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

15.7 发送邮件 cronjob

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

15.8 实现激活逻辑

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

15.9 写测试

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

15.10 激活成功提示

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

15.11 修改登录验证逻辑

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

15.12 写测试

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

15.13 run-all-tests.sh

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

15.14 本章小结

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

16. OurBlog 增加打标签功能

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

16.1 新增数据库表

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

16.2 添加文章打标签

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

16.3 写测试

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

16.4 编辑文章打标签

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

16.5 写测试

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

16.6 删除文章时也删除标签

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

16.7 写测试

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

16.8 本章小结

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

17. OurBlog 增加文件上传功能

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

17.1 文件上传页面

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

17.2 文件上传逻辑

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

17.3 写测试

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

17.4 本章小结

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

18. OurBlog 增加外部文章功能

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

18.1 数据库

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

18.2 添加文章

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

18.3 同步字段修改

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

18.4 写测试

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

18.5 编辑文章

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

18.6 写测试

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

18.7 前台文章详情页面

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

18.8 本章小结

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

19. 让测试跑的更快

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

19.1 DbUnit

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

19.2 SSD

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

19.3 MySQL in memory

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

19.4 使用 MySQL in memory

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

19.5 本章小结

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

20. 真实项目分享1: OurATS

该内容没有包含在样本书中。您可以在 Leanpub 上购买这本书，网址为 <http://leanpub.com/phpunit-in-action>

21. 真实项目分享2: BobParser

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

22. 结束语

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>

附录：Bob Test Theroy

該內容沒有包含在樣本書中。您可以在 Leanpub 上購買這本書，網址為 <http://leanpub.com/phpunit-in-action>