PHPUnit

https://phpunit.de/manual/6.3/zh_cn/index.html

安装phpunit

- 下载phar
  - wget https://phar.phpunit.de/phpunit-6.2.phar
- 添加可执行权限
  - chmod +x phpunit-6.2.phar
- 放入bin目录可全局执行
  - cp phpunit-6.2.phar /usr/local/bin/phpunit
- 查看phpunit版本
  - phpunit —version

编写规范

- 针对类的测试写在classTest中
- classTest继承自\Base\Test\TestCase(phpunit默认的是继承PHPUnit\Framework\TestCase)
- 测试都是命名为test*的方法，也可以在方法的文档注释中使用@test标注将其标记为测试方法

```php
class DemoTest extends \Base\Test\TestCase {

    public function setUp() {
        parent::setUp();
    }

    /**
     * @test
     * @dataProvider additionProvider
     */
    public function testAction($id, $name) {
        $params   = array(
            'id'   => $id,
            'name' => $name,
        );

        $response = $this->execRequest(\S\Http::METHOD_GET, 'api_demo', $params);

        $expect = $name;

        $this->assertEquals($expect, $response['data']);
    }

    public function additionProvider() {
        return array(
            array(3, 'xin'),
        );
    }
}
```

布尔，NULL类型

https://phpunit.de/manual/6.3/zh_cn/appendixes.assertions.html

- assertTrue(断言为真)
- assertFalse(断言为假)
- assertNull(断言为NULL　)
- assertNotNull(断言非NULL)
- assertInternalType（断言是指定类型）
- assertNotInternalType（断言不是指定类型）

```php
/* * * * * * * *
 *    布尔型    *
 * * * * * * * */
public function testBool() {
    $this->assertTrue(true);//断言是true类型
    $this->assertFalse(false);//断言是false类型
    $this->assertInternalType('bool', true);//断言是bool类型
}

/* * * * * * * *
 *    NULL 型   *
 * * * * * * * */
public function testNull() {
    $this->assertNull(null);//断言是null
    $this->assertNotNull(array());//断言不是null
}
```

数字类型

https://phpunit.de/manual/6.3/zh_cn/appendixes.assertions.html

- assertEquals(断言等于)
- assertNotEquals(断言不等于)
- assertGreaterThan(断言大于)
- assertGreaterThanOrEqual(断言大于等于)
- assertLessThan(断言小于)
- assertLessThanOrEqual(断言小于等于)
- assertInternalType（断言是指定类型）
- assertNotInternalType（断言不是指定类型）
- assertSame(断言类型和格式都相等)
- assertNotSame（断言类型或格式不相等）

```php
/* * * * * * * *
 *             *
 *   数字类型   *
 *             *
 * * * * * * * */
public function testNumber() {
    $a = 1;
    $b = 2;

    $this->assertEquals(1, $a);//断言相等
    $this->assertNotEquals($a, $b);//断言不相等
    $this->assertGreaterThan($a, $b);//断言$b大于$a
    $this->assertGreaterThanOrEqual($a, $b);//断言$b大于等于$a
    $this->assertLessThan($b, $a);//断言$a小于$b
    $this->assertLessThanOrEqual($b, $a);//断言$a小于等于$b
    $this->assertInternalType('int', 1);//断言是int类型
    $this->assertSame(12, 12);//断言类型和值都相等
    $this->assertNotSame(12, '12');//断言类型和值有一个不相等
}
```

字符串类型

https://phpunit.de/manual/6.3/zh_cn/appendixes.assertions.html

- assertEquals(断言等于)
- assertNotEquals(断言不等于)
- assertContains(断言包含)
- assertNotContains(断言不包含)
- assertContainsOnly(断言只包含某种类型)
- assertNotContainsOnly(断言不只包含某种类型)
- assertInternalType（断言是指定类型）
- assertNotInternalType（断言不是指定类型）
- assertJsonStringEqualsJsonString（断言json字符串等于json字符串）
- assertSame(断言类型和格式都相等)
- assertNotSame（断言类型或格式不相等）
- assertStringEndsWith（断言字符串以什么结尾）
- assertStringStartsWith(断言字符串以什么开头)

```php
/********
 *        *
 *  字符串类型 *
 *        *
 ********/
public function testString() {
    $this->assertEquals('aa', 'aa');//断言两个字符串相等
    $this->assertNotEquals('aa', 'bb');//断言两个字符创不相等
    $this->assertContains('c', 'abc');//断言字符串包含在另外一个字符串中
    $this->assertNotContains('d', 'abc');//断言字符串不包含在另外一个字符串中
    $this->assertContainsOnly('string', array('3'));//断言只包含string类型
    $this->assertNotContainsOnly('string', array('aaa', 11));//断言不仅仅包含string类型
    $this->assertInternalType('string', '11');//断言是string类型
    $this->assertJsonStringEqualsJsonString("[1]", "[1]");//断言json字符串相等
    $this->assertSame('12', '12');//断言类型和值都相等
    $this->assertNotSame(12, '12');//断言类型和值有一个不相等
    $this->assertStringEndsWith('1', '121');//断言字符串: 121 是以1开头
}
```

数组类型

https://phpunit.de/manual/6.3/zh_cn/appendixes.assertions.html

- assertEquals(断言等于)
- assertNotEquals(断言不等于)
- assertArrayHasKey(断言有键)
- assertArrayNotHasKey(断言没有键)
- assertContains(断言包含)
- assertNotContains(断言不包含)
- assertContainsOnly(断言只包含)
- assertNotContainsOnly(断言不只包含)
- assertArraySubset(断言数组包含)
- assertCount(断言数组元素个数)

- assertEmpty（断言为空）
- assertNotEmpty（断言非空）
- assertInternalType（断言是指定类型）
- assertNotInternalType（断言不是指定类型）
- assertSame(断言类型和格式都相等)
- assertNotSame（断言类型或格式不相等）

```php
/* * * * * * * *
 *             *
 *    数组类型  *
 *             *
 * * * * * * * */
public function testArray() {
    $data = array(
        'age'=>12,
        'name'=>'测试'
    );

    $this->assertEquals(array(1), array('1'));//断言两个数组相等
    $this->assertNotEquals(array(1), array(1, 2));//断言两个数组不相等
    $this->assertArrayHasKey('name', $data);//断言data中存在key为name
    $this->assertArrayNotHasKey('gender', $data);//断言数组data中不存在key为gender
    $this->assertContains(12, $data);//断言data中包含12
    $this->assertNotContains(11, $data);//断言data中不包含11
    $this->assertContainsOnly('array', array(array(1)));//断言包含数组
    $this->assertNotContainsOnly('array', array(1, array(1)));//断言不仅仅包含数组(实际上不包含数组也可以)
    $this->assertArraySubset(array(1, 2), array(1, 2, 3));//断言数组中是否包含另外一个数组，类似于in_array
    $this->assertCount(1, array(1));//断言数组元素个数
    $this->assertEmpty(array());//断言数组为空
    $this->assertNotEmpty(array(1));//断言数组不为空
    $this->assertInternalType('array', array(1));//断言类型是数组
    $this->assertSame(array(1), array(1));//断言类型或值相等
    $this->assertNotSame(array(1), array('1'));//断言类型或值不相等
}
```

对象类型

https://phpunit.de/manual/6.3/zh_cn/appendixes.assertions.html

- assertAttributeContains(断言属性包含)
- assertAttributeContainsOnly(断言属性只包含)
- assertAttributeEquals(断言属性等于)
- assertAttributeGreaterThan(断言属性大于)
- assertAttributeGreaterThanOrEqual(断言属性大于等于)
- assertAttributeLessThan(断言属性小于)
- assertAttributeLessThanOrEqual(断言属性小于等于)

- assertAttributeNotContains(断言不包含)
- assertAttributeNotContainsOnly(断言属性不只包含)
- assertAttributeNotEquals(断言属性不等于)
- assertAttributeNotSame(断言属性不相同)
- assertAttributeSame(断言属性相同)
- assertSame(断言类型和值都相同)
- assertNotSame(断言类型或值不相同)
- assertObjectHasAttribute(断言对象有某属性)
- assertObjectNotHasAttribute(断言对象没有某属性)

class类型

https://phpunit.de/manual/6.3/zh_cn/appendixes.assertions.html

- assertClassHasAttribute(断言类有某属性)
- assertClassHasStaticAttribute(断言类有某静态属性)
- assertClassNotHasAttribute(断言类没有某属性)
- assertClassNotHasStaticAttribute(断言类没有某静态属性)
- assertContainsOnlyInstancesOf(断言并非只包含某个实例)
- assertInstanceOf(断言是对象的实例)
- assertNotInstanceOf（断言不是对象的实例）

文件相关

https://phpunit.de/manual/6.3/zh_cn/appendixes.assertions.html

- assertFileEquals(断言文件内容等于)
- assertFileExists(断言文件存在)
- assertFileNotEquals(断言文件内容不等于)
- assertFileNotExists(断言文件不存在)
- assertDirectoryExists（断言目录存在）
- assertDirectoryIsReadable（断言目录可读）
- assertDirectoryIsWritable（断言目录可写）
- assertFileIsReadable(断言文件可读)
- assertFileIsWritable（断言文件可写）
- assertIsReadable（断言指定文件或目录可读）
- assertIsWritable（断言指定文件或目录不可读）
- assertJsonFileEqualsJsonFile（断言json文件相等）
- assertJsonStringEqualsJsonFile（断言json文件和json值相等）

正则

https://phpunit.de/manual/6.3/zh_cn/appendixes.assertions.html

- assertRegExp（断言匹配正则表达式）
- assertStringMatchesFormat（断言符合格式）
- assertStringMatchesFormatFile（断言符合格式文件）

XML相关

https://phpunit.de/manual/6.3/zh_cn/appendixes.assertions.html

- assertXmlFileEqualsXmlFile(断言XML文件内容相等)
- assertXmlFileNotEqualsXmlFile(断言XML文件内容不相等)
- assertXmlStringEqualsXmlFile(断言XML字符串等于XML文件内容)
- assertXmlStringEqualsXmlString(断言XML字符串相等)
- assertXmlStringNotEqualsXmlFile(断言XML字符串不等于XML文件内容)
- assertXmlStringNotEqualsXmlString(断言XML字符串不相等)

测试关键字

- 在方法的文档注释块中用 @test 标注来将其标记为测试方法：@test
- 依赖关系：@depends
- 数据供给器：@dataProvider
- 异常测试：@expectedException
- 指明方法应当在测试用例类中的每个测试方法运行完成之后调用：@after
- 指明此静态方法应该于测试类中的所有测试方法都运行完成之后调用，用于清理共享基境：@afterClass
- 标注用于指明此方法应当在测试用例类中的每个测试方法开始运行之前调用：@before
- 用于指明此静态方法应该于测试类中的所有测试方法都运行完成之后调用，用于建立共享基境：@beforeClass
- 全局变量的备份与还原操作可以对某个测试用例类中的所有测试彻底禁用：@backupGlobals
- 在每个测试之前备份所有已声明的类的静态属性的值，并在测试完成之后全部恢复。它可以用在测试用例类或测试方法级别：@backupStaticAttributes

依赖关系,数据供给

对测试方法之间的显式依赖关系进行声明，使用@depends关键字来标注，如示例中：testPop依赖testPush，testPush依赖testEmpty

注意：最好按照依赖关系，顺序编写测试方法

多重依赖，如示例中：testConsumer依赖testProducerFirst，testProducerSecond两个方法

```php
class DependsTest extends \Base\Test\TestCase {

    /* * * * * * * *
     *             *
     *   单个依赖   *
     *             *
     * * * * * * * */
    public function testEmpty() {
        $stack = [];
        $this->assertEmpty($stack);

        return $stack;
    }

    /**
     * @depends testEmpty
     */
    public function testPush(array $stack) {
        array_push($stack, 'foo');
        $this->assertEquals('foo', $stack[count($stack)-1]);
        $this->assertNotEmpty($stack);

        return $stack;
    }

    /**
     * @depends testPush
     */
    public function testPop(array $stack) {
        $this->assertEquals('foo', array_pop($stack));
        $this->assertEmpty($stack);
    }

    /* * * * * * * *
     *             *
     *   多重依赖   *
     *             *
     * * * * * * * */
    public function testProducerFirst() {
        $this->assertTrue(true);
        return 'first';
    }

    public function testProducerSecond() {
        $this->assertTrue(true);
        return 'second';
    }

    /**
     * @depends testProducerFirst
     * @depends testProducerSecond
     */
    public function testConsumer() {
        $this->assertEquals(
            ['first', 'second'],
            func_get_args()
        );
    }
}
```

数据供给：如下面示例，additionProvider的数据提供给了testAdd方法，使用了dataProvide关键字进行标注

```php
class ProviderTest extends \Base\Test\TestCase {
    /**
     * @dataProvider additionProvider
     */
    public function testAdd($a, $b, $expected) {
        $this->assertEquals($expected, $a + $b);
    }

    public function additionProvider() {
        return [
            [0, 0, 0],
            [0, 1, 1],
            [1, 0, 1],
            [1, 1, 2]
        ];
    }
}
```

如果测试同时从 @dataProvider 方法和一个或多个 @depends 测试接收数据，那么来自于数据供给器的参数将先于来自所依赖的测试的

对异常进行测试

- 断言抛出异常的类
  - expectException()
  - @ expectedException
- 断言抛出异常的code
  - expectExceptionCode()
  - @expectedExceptionCode
- 断言抛出异常的message
  - expectExceptionMessage()
  - @expectedExceptionMessage
- 断言抛出的异常符合正则表达式
  - expectExceptionMessageRegExp
  - @expectedExceptionMessageRegExp
- PHP报通知错的类：PHPUnit\Framework\Error\Notice
- PHP报警告错误的类：PHPUnit\Framework\Error\Warning

```php
<?php

class ExceptionTest extends \Base\Test\TestCase {
    /**
     * @expectedException Exception
     * @expectedExceptionCode 4000012
     * @expectedExceptionMessage 参数错误
     */
    public function testException() {
        throw  new \Exception('参数错误', 4000012);
    }

    public function test1Exception() {
        $this->expectException('\Exception');
        $this->expectExceptionCode(4000012);
        $this->expectExceptionMessage('参数错误');
        throw new \Exception('参数错误', 4000012);
    }
}
```

钩子

- 基镜
  - 最费时的部分之一是编写代码来将整个场景设置成某个已知的状态，并在测试结束后将其复原到初始状态。这个已知的状态称为测试的基境，提供基镜的方法包含如下四中：setUp，tearDown，setUpBeforeClass，tearDownAfterClass
- setUp
  - test测试方法调用前执行，用于初始化测试数据，每个test方法执行前都会运行一次
- tearDown
  - test测试方法执行结束后运行，用于清理测试公共数据，每个test方法执行完成都会运行一次
- setUpBeforeClass
  - 所有test测试方法调用之前执行，用于初始化全局测试数据，整个test文件执行之前运行一次
- tearDownAfterClass
  - 所有test测试方法执行结束后运行，用于销毁全局测试数据，整个test文件执行之后运行一次

钩子例子

```php
<?php

class HookTest extends \PHPUnit\Framework\TestCase {

    public function testAction() {
        echo 'test action run!' . PHP_EOL;
    }

    public function test2Action() {
        echo 'test1 action run!'. PHP_EOL;
    }

    public function setUp() {
        echo 'setUp run!'. PHP_EOL;
    }

    public function tearDown() {
        echo 'tearDown run!'. PHP_EOL;
    }

    public static function setUpBeforeClass() {
        echo 'setUpBeforeClass run!'. PHP_EOL;
    }

    public static function tearDownAfterClass() {
        echo 'tearDownAfterClass run!'. PHP_EOL;
    }
}
```

```
[root@bf690b918b7d test]# phpunit controllers/HookTest.php
PHPUnit 6.1.4 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.1.7
Configuration: /data1/htdocs/demo/test/phpunit.xml

setUpBeforeClass run!
RsetUp run!
test action run!
tearDown run!
R
test1 action run!
tearDown run!
tearDownAfterClass run!


Time: 84 ms, Memory: 8.00MB
```

Stubs，mock

- 将对象替换为（可选地）返回配置好的返回值的测试替身的实践方法称为上桩(stubbing)

- 将对象替换为能验证预期行为（例如断言某个方法必会被调用）的测试替身的实践方法称为模仿(mocking)

```php
<?php

use \PHPUnit\Framework\TestCase;

class SomeClass {

    public function doSomething() {

    }
}
```

```php
<?php

require_once "SomeClass.php";

class StubTest extends \PHPUnit\Framework\TestCase {

    public function testStub() {
        // 为 SomeClass 类创建桩件。
        $stub = $this->createMock(SomeClass::class);

        // 配置桩件。
        $stub->expects($this->any())->method('doSomething')->willReturn('foo');

        // 现在调用 $stub->doSomething() 将返回 'foo'。
        $this->assertEquals('foo', $stub->doSomething());
    }

    public function testMock() {
        $stub = $this->getMockBuilder('SomeClass')->disableOriginalConstructor()->getMock();

        //建立预期doSomething被执行0次或多次，返回foo值
        $stub->expects($this->any())->method('doSomething')->will($this->returnValue('foo'));

        $this->assertEquals('foo', $stub->doSomething());
    }
}
```

桩件用到的函数

- createMock()　在生成测试替身时所使用的默认值不符合你的要求
- getMockBuilder($type)　用流畅式接口定制测试替身的生成过程
- setMethods(array $methods)　可以在仿件生成器对象上调用，来指定哪些方法将被替换为可配置的测试替身
- setConstructorArgs(array $args)　可用于向原版类的构造函数（默认情况下不会被替换为伪实现）提供参数数组
- setMockClassName($name)　可用于指定生成的测试替身类的类名
- disableOriginalConstructor()　参数可用于禁用对原版类的构造方法的调用
- disableOriginalClone()　可用于禁用对原版类的克隆方法的调用
- disableAutoload()　可用于在测试替身类的生成期间禁用 __autoload()

- disableArgumentCloning
- disallowMockingUnknownTypes
- getMock
- method() 设置预期的方法
- willReturn 设置预期返回的值
- with 方法可以携带任何数量的参数
- withConsecutive() 方法可以接受任意多个数组作为参数
- callback() 约束用来进行更加复杂的参数校验

匹配器

- any() 返回一个匹配器，当被评定的方法执行0次或更多次（即任意次数）时匹配成功
- never() 返回一个匹配器，当被评定的方法从未执行时匹配成功
- atLeastOnce() 返回一个匹配器，当被评定的方法执行至少一次时匹配成功
- once() 返回一个匹配器，当被评定的方法执行恰好一次时匹配成功
- exactly(int $count) 返回一个匹配器，当被评定的方法执行恰好 $count 次时匹配成功
- at(int $index) 返回一个匹配器，当被评定的方法是第 $index 个执行的方法时匹配成功