



PHP 测试框架 PHPUnit 和 Mockery 可以帮助我们编写测试。本文介绍了如何使用 PHPUnit 和 Mockery 来测试 PHP 代码。

# 1. 使用 Mock(Mock) 测试

PHPUnit 和 Mockery 可以帮助我们编写测试。本文介绍了如何使用 PHPUnit 和 Mockery 来测试 PHP 代码。

本文

介绍了，如何使用 PHPUnit 和 Mockery。

```
namespace App\Services;

interface GreetingServiceInterface
{
    public function greet($name);
}

class GreetingService implements GreetingServiceInterface
{
    public function greet($name)
    {
        return "Hello, $name!";
    }
}
```

本文，介绍了如何使用 PHPUnit。

```
namespace App\Http\Controllers;

use App\Services\GreetingServiceInterface;

class GreetingController extends Controller
{
    protected $greetingService;
```

```

public function __construct(GreetingServiceInterface $greetingService)
{
    $this->greetingService = $greetingService;
}

public function greet($name)
{
    return $this->greetingService->greet($name);
}
}

```

이제 이 코드를 테스트해 보겠습니다.

```

use Tests\TestCase;
use App\Services\GreetingServiceInterface;
use App\Http\Controllers\GreetingController;
use Mockery;

class GreetingControllerTest extends TestCase
{
    public function testGreet()
    {
        // GreetingServiceInterface를 모킹합니다
        $mockService = Mockery::mock(GreetingServiceInterface::class);

        // greet 메서드가 "Hello, John!"을 반환하도록 모킹합니다
        $mockService->shouldReceive('greet')
            ->with('John')
            ->andReturn('Hello, John!');

        // 컨트롤러를 생성합니다
        $controller = new GreetingController($mockService);

        // 테스트 실행
        $response = $controller->greet('John');
        $this->assertEquals('Hello, John!', $response);
    }

    protected function tearDown(): void
    {

```

```
Mockery::close();
parent::tearDown();
}
}
```

## 2. 创建测试用例

首先，我们需要创建一个测试用例类，用于测试我们的服务提供者。

在 `tests` 目录下创建 `GreetingServiceTest` 类。

在 `GreetingServiceTest` 类中，我们需要实现 `TestCase` 接口。

```
namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use App\Services\GreetingServiceInterface;
use App\Services\GreetingService;

class AppServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->bind(GreetingServiceInterface::class, GreetingService::class);
    }

    public function boot()
    {
        //
    }
}
```

在 `GreetingServiceTest` 类中，我们需要实现 `setUp` 方法。

```
use Tests\TestCase;
use App\Services\GreetingServiceInterface;

class GreetingServiceTest extends TestCase
{
```

```

public function testGreet()
{
    // 测试 GreetingService 接口
    $greetingService = $this->app->make(GreetingServiceInterface::class);

    // 测试 greet 方法
    $response = $greetingService->greet('John');
    $this->assertEquals('Hello, John!', $response);
}
}

```



- **Mockery** 是一个强大的 PHP 测试框架，用于创建测试用的模拟对象。它支持多种测试框架，如 PHPUnit、Symfony 等。
- **Mockery** 提供了丰富的 API，用于创建模拟对象、设置期望、验证行为等。它还可以与 PHPUnit 集成，方便在测试用例中使用。

**Q1:** Mock 是什么？它有什么作用？

**Q2:** 如何创建 Mock 对象？

**Q3:** 如何验证 Mock 对象的行为？

You wanna more detailed information?

Revision #1

Created 4 August 2024 00:41:51 by 111

Updated 4 August 2024 00:42:06 by 111