Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский политехнический университет»

Кафедра «Инфокогнитивные технологии»

Образовательная программа «Веб-технологии»

Лабораторная работа №8

по дисциплине

 «Программная инженерия»

на тему: «Написание Unit-тестов (модульное тестирование)»

**Выполнил:**

Студент группы 181-321

                                      Малиновская К.В.

Москва 2020

**PHPUnit**

Для запуска юнит-тестов создаю новую конфигурацию запуска (Run – Edit Configuration), указывая имя и какой файл нужно запустить.

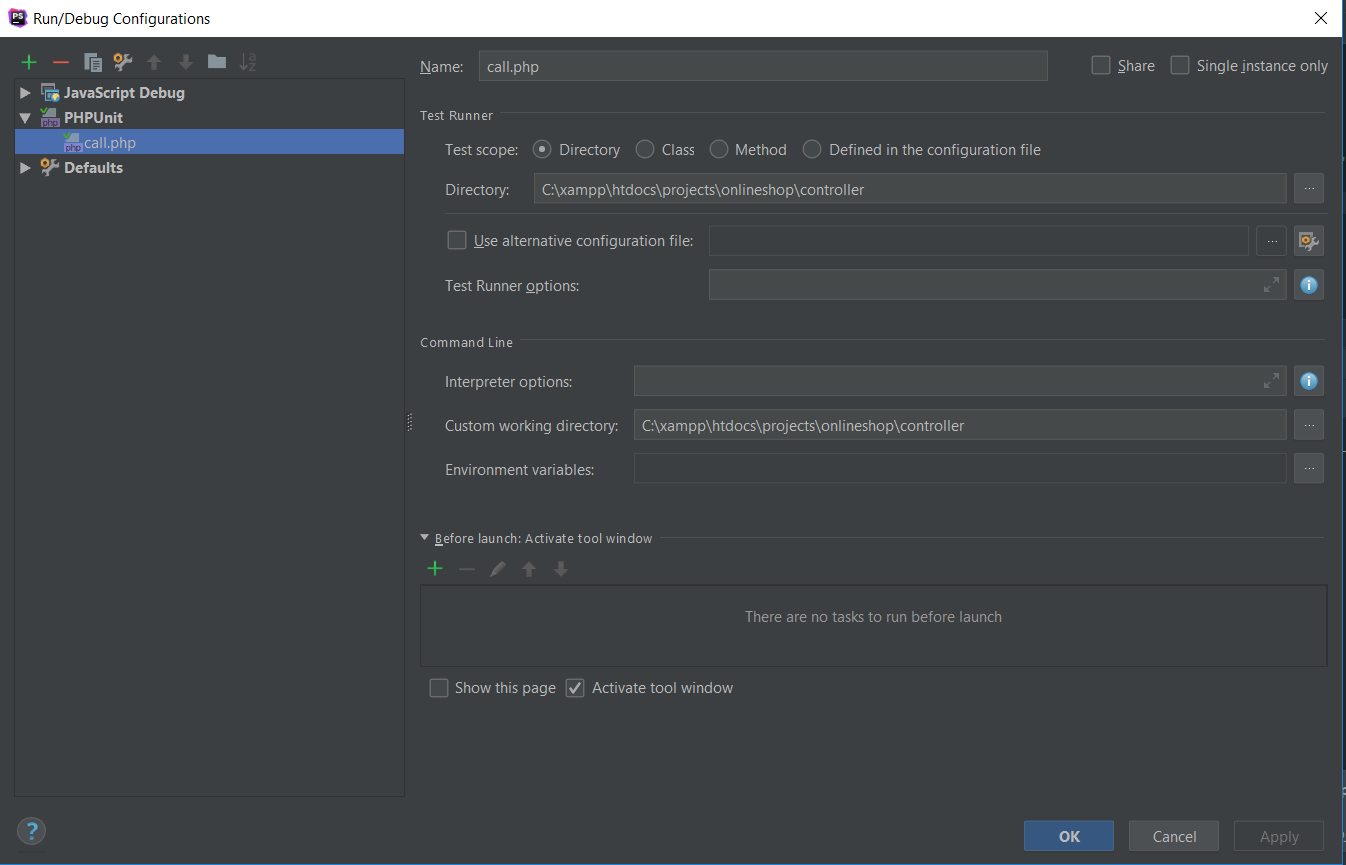


Рисунок 1. Создаю новую конфигурацию

Далее запускаю модульные тест (Run), рисунок 2.

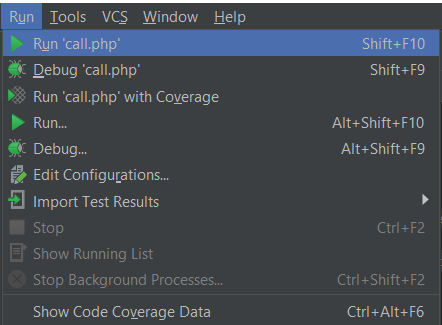
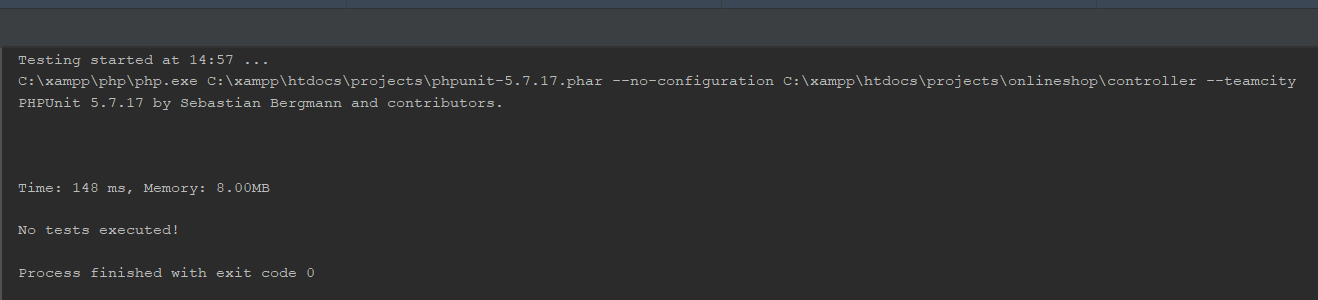


Рисунок 39. Запускаем тест

Результат теста выводится в контекстное меню, рисунок 3.

Рисунок 3. Результат теста

# Примеры модульного тестирования

**Тест 1.** Проверка модели User  
**<?php  
  
namespace** Tests\Unit\Controllers;  
  
**use** App\Http\Controllers\Api\ApiController;  
**use** App\Http\Controllers\Api\V2\UserController;  
**use** App\Http\Requests\Api\V2\Users\UpdatePasswordRequest;  
**use** App\Http\Requests\Api\V2\Users\UserCreationRequest;  
**use** App\Models\User;  
**use** App\Services\UserService;  
**use** PHPUnit\Framework\TestCase;  
**use** ReflectionClass;  
  
**class** UserControllerTest **extends** TestCase  
{  
 */\*\*  
 \** ***@var*** *ReflectionClass  
 \*/* **private $controller**;  
  
 **protected function** setUp(): void  
 {  
 **parent**::*setUp*();  
  
 $this->**controller** = **new** ReflectionClass(UserController::***class***);  
 }  
  
 **public function** test\_user\_controller\_extends\_api\_controller()  
 {  
 $controller = **new** ReflectionClass(UserController::***class***);  
  
 $this->assertSame(  
 ApiController::***class***,  
 $controller->getParentClass()->getName()  
 );  
 }  
  
 **public function** test\_user\_controller\_has\_user\_service\_property()  
 {  
 $this->assertTrue($this->**controller**->hasProperty(**'userService'**));  
 }  
  
 **public function** test\_user\_controller\_has\_service\_in\_constructor()  
 {  
 $parameters = $this->**controller**->getConstructor()->getParameters();  
 $this->assertNotEmpty($parameters);  
  
 $userService = $parameters[0];  
  
 $this->assertSame(UserService::***class***, (string) $userService->getType());  
 }  
  
 **public function** test\_user\_controller\_has\_get\_current\_action()  
 {  
 $this->assertTrue($this->**controller**->hasMethod(**'getCurrent'**));  
 }  
  
 **public function** test\_user\_controller\_has\_store\_action()  
 {  
 $this->assertTrue($this->**controller**->hasMethod(**'store'**));  
 }  
  
 **public function** test\_user\_controller\_has\_update\_action()  
 {  
 $this->assertTrue($this->**controller**->hasMethod(**'update'**));  
 }  
  
 **public function** test\_store\_action\_has\_request()  
 {  
 $method = $this->**controller**->getMethod(**'store'**);  
 $parameters = $method->getParameters();  
  
 $this->assertNotEmpty($parameters);  
  
 $requestParameter = $parameters[0];  
  
 $this->assertSame(UserCreationRequest::***class***, (string) $requestParameter->getType());  
 }  
  
 **public function** test\_update\_action\_has\_user\_and\_request()  
 {  
 $method = $this->**controller**->getMethod(**'update'**);  
 $parameters = $method->getParameters();  
  
 $this->assertNotEmpty($parameters);  
  
 $userParameter = $parameters[0];  
 $requestParameter = $parameters[1];  
  
 $this->assertSame(User::***class***, (string) $userParameter->getType());  
 $this->assertSame(UpdatePasswordRequest::***class***, (string) $requestParameter->getType());  
 }  
}

**Тест 2.** Тест на обработчик запросов, связанных с изменением ролей пользователя (Проверка UserRoleController)

**<?php  
  
namespace** Tests\Unit\Controllers;  
  
**use** App\Http\Controllers\Api\ApiController;  
**use** App\Http\Controllers\Api\V2\UserRoleController;  
**use** App\Models\Role;  
**use** App\Models\User;  
**use** App\Services\UserRoleService;  
**use** ReflectionClass;  
**use** PHPUnit\Framework\TestCase;  
  
**class** UserRoleControllerTest **extends** TestCase  
{  
 */\*\*  
 \** ***@var*** *ReflectionClass  
 \*/* **private $controller**;  
  
 **protected function** setUp(): void  
 {  
 **parent**::*setUp*();  
  
 $this->**controller** = **new** ReflectionClass(UserRoleController::***class***);  
 }  
  
 **public function** test\_user\_role\_controller\_extends\_api\_controller()  
 {  
 $this->assertSame(  
 ApiController::***class***,  
 $this->**controller**->getParentClass()->getName()  
 );  
 }  
  
 **public function** test\_user\_role\_controller\_has\_service\_property()  
 {  
 $this->assertTrue($this->**controller**->hasProperty(**'service'**));  
 }  
  
 */\*\*  
 \* Constroller must has service in construictor  
 \*/* **public function** test\_user\_role\_controller\_has\_service\_in\_constructor()  
 {  
 $parameters = $this->**controller**->getConstructor()->getParameters();  
  
 $this->assertNotEmpty($parameters);  
  
 $serviceParameter = $parameters[0];  
  
 $this->assertSame(UserRoleService::***class***, (string) $serviceParameter->getType());  
 }  
  
 **public function** test\_get\_roles\_action\_exists()  
 {  
 $this->assertTrue($this->**controller**->hasMethod(**'getRoles'**));  
 }  
  
 **public function** test\_attach\_role\_action\_exists()  
 {  
 $this->assertTrue($this->**controller**->hasMethod(**'attachRole'**));  
 }  
  
 **public function** test\_detach\_role\_action\_exists()  
 {  
 $this->assertTrue($this->**controller**->hasMethod(**'detachRole'**));  
 }  
  
 */\*\*  
 \** ***@throws*** *\ReflectionException  
 \*/* **public function** test\_get\_roles\_action\_has\_user\_parameter()  
 {  
 $method = $this->**controller**->getMethod(**'getRoles'**);  
 $parameters = $method->getParameters();  
 $userParameter = $parameters[0];  
  
 $this->assertNotEmpty($parameters);  
 $this->assertSame(User::***class***, (string) $userParameter->getType());  
 }  
  
 */\*\*  
 \** ***@throws*** *\ReflectionException  
 \*/* **public function** test\_attach\_role\_action\_has\_user\_and\_role\_parameters()  
 {  
 $method = $this->**controller**->getMethod(**'attachRole'**);  
 $parameters = $method->getParameters();  
  
 $this->assertCount(2, $parameters);  
  
 $userParameter = $parameters[0];  
 $roleParameter = $parameters[1];  
  
 $this->assertSame(User::***class***, (string) $userParameter->getType());  
 $this->assertSame(Role::***class***, (string) $roleParameter->getType());  
 }  
  
 */\*\*  
 \** ***@throws*** *\ReflectionException  
 \*/* **public function** test\_detach\_role\_action\_has\_user\_and\_role\_parameters()  
 {  
 $method = $this->**controller**->getMethod(**'detachRole'**);  
 $parameters = $method->getParameters();  
  
 $this->assertCount(2, $parameters);  
  
 $userParameter = $parameters[0];  
 $roleParameter = $parameters[1];  
  
 $this->assertSame(User::***class***, (string) $userParameter->getType());  
 $this->assertSame(Role::***class***, (string) $roleParameter->getType());  
 }  
}

**Тест 2.** Проверка модели User

**<?php  
  
namespace** Tests\Unit\Models;  
  
**use** App\Models\User;  
**use** PHPUnit\Framework\TestCase;  
**use** ReflectionClass;  
**use** Spatie\Permission\Traits\HasRoles;  
  
*/\*\*  
 \* Class UserModelTest.  
 \*/***class** UserModelTest **extends** TestCase  
{  
 **public function** test\_user\_model\_has\_guard\_name\_attribute()  
 {  
 $this->assertClassHasAttribute(**'guard\_name'**, User::***class***);  
 }  
  
 **public function** test\_user\_model\_has\_roles\_trait()  
 {  
 $model = **new** ReflectionClass(User::***class***);  
 $traits = $model->getTraits();  
  
 $this->assertTrue(  
 *in\_array*(HasRoles::***class***, *array\_keys*($traits))  
 );  
 }  
}