Modernus PHP programų testavimas

Žilvinas Kuusas

Kas yra automatinis testavimas?

Automatinis testas – programos bei įrankiai, skirti kitų programų ar modulių darbui patikrinti.

Kodėl testuoti?

- Su PHP kuriami ne tik "interneto puslapiai", bet ir sudėtingos verslo sistemos
- Prevencinė priemonė klaidų atsiradimui
- Kliento nuostoliai
- Testais padengtą kodą saugiau pertvarkyti
- Padeda išvengti stresinių situacijų prieš paleidimą

Ką testuoti?

- Kritines vietas
- Dažnai keičiamą kodą
- Vietas, kur aptinkamos klaidos
- Viską

Kada testuoti?

- Prieš rašant kodą (TDD, BDD)
- Parašius kodą
- Aptikus klaidą

Kaip testuoti?

- Nepersistengiant!
- Testai turi būti kuo paprastesni
- Testai turi būti paleidžiami kuo paprasčiau
- Testai turi veikti maksimaliai greitai
- Testai turi būti lengvai suprantami, lengvai taisomi

Testų tipai / kodo kokybės užtikrinimo būdai

- Vienetiniai (unit)
- Integraciniai
- Funkciniai
- Atitikties(?) (acceptance)
- Kodo peržiūra (code review)

Vienetiniai (unit) testai

- Testuoja kodo vienetus
- Užtikrina teisingą kodo veikimą
- Supaprastina kodo perrašymą
- Jeigu neįmanoma testuoti visko - svarbu, kad testavimui skiramas laikas būtų tikslingas

Kas yra vienetas vienetiniuose testuose?

Kodo dalis atsakinga už mikro funkcionalumą, kurį galima paleisti nepriklausomai nuo programos sukuriamos būsenos, turintis apibrėžtą input/output interfeisą. Paprastai tai būna vienas metodas.

Kodo izoliavimas

- Stub
- Skirtas pakeisti metodą kodu, kuris grąžina nurodytą reikšmę
- Mock
- Tas pats stub, tik su papildomais patikrinimais ar nurodyti metodai buvo tikrai iškviesti
- Mock gali priversti jūsų testus griūti, stub - ne

Stub

```
namespace VilniusPHP;

class StubTest extends PHPUnit_Framework_TestCase
{
    public function testStub()
    {
        $stub = $this->getMock('UserManager');
        $stub->expects($this->any())
             ->method('getUser')
             ->will($this->returnValue('foo'));

        // Kviečiant $stub->getUser() metodą, jis grąžins 'foo'.
    }
}
```

Mock

```php
namespace VilniusPHP;

class ObserverTest extends PHPUnit_Framework_TestCase
{
    public function testUpdateIsCalledOnce()
    {
        // Sukuriamas netikras UserManager klasės objektas (Mock)
        // nurodom reaguoti tik į update metodą
        $managerMock = $this->getMock('UserManager', array('update'));

        // Nurodom ko tikimės iš update() metodo
        // norim, kad metodas būtų kviečiamas vieną kartą
        // su parametru 'Tutituti'
        $managerMock->expects($this->once())
                    ->method('update')
                    ->with($this->equalTo('Tutituti'));

        // Sukuriam objektą, kuris naudoja UserManager
        // Observer object to it.
        $globalManager = new GlobalManager($managerMock);
```

Jeigu vykdant $globalManager->updateAll() metodą nebus iškviestas UserManager update() metodas - testas nepavyks.

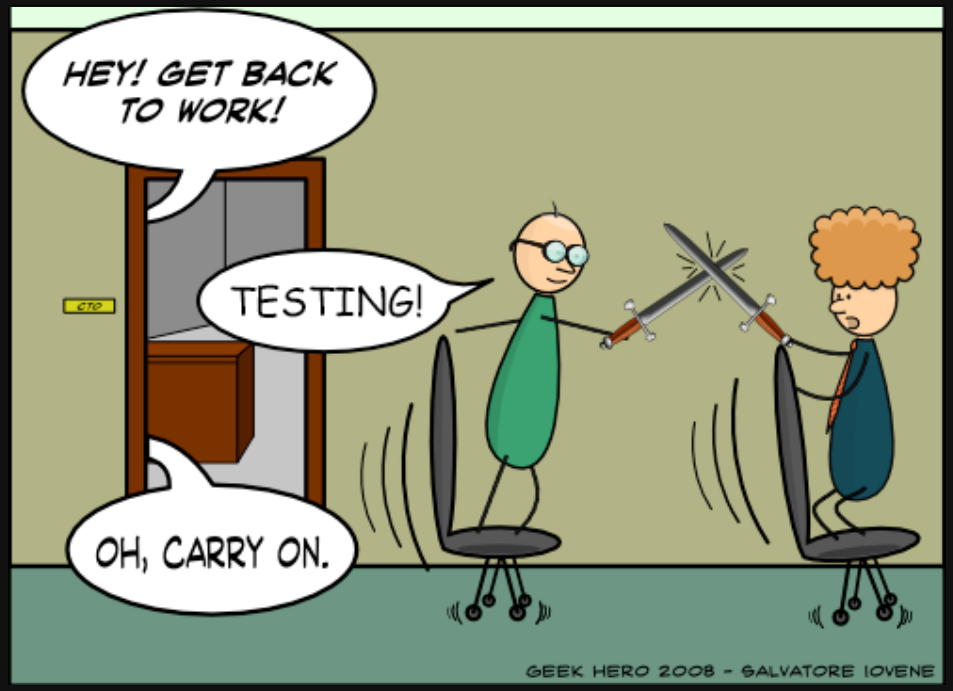
Mock - ką testuojame

```php
namespace VilniusPHP;

class GlobalManager
{
    private $userManager;

    public function __construct(UserManager $userManager)
    {
        $this->userManager = $userManager;
    }

    public function updateAll()
    {
        $this->userManager->update('Tutituti');
    }
}

class UserManager
{
    public function update($arg)
```

Kodo vykdymo laikas

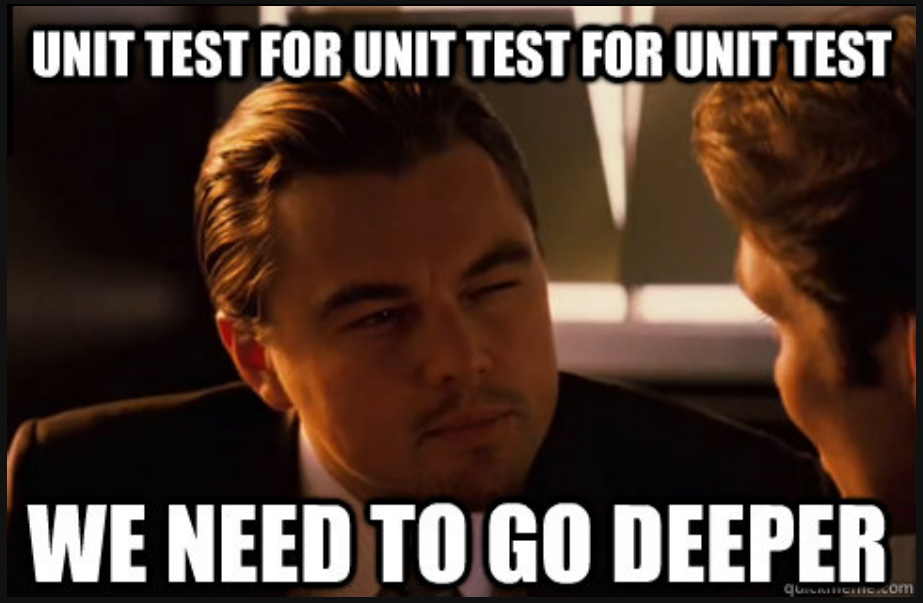
```
$ phpunit -c app

PHPUnit 3.7.28-12-g236f65c by Sebastian Bergmann.

Configuration read from /app/phpunit.xml

Time: 237 ms, Memory: 21.75Mb
```

Vykdymo laikas - labai svarbus faktorius, kad nedingtų noras programuotojui leisti testus visada.



Bendri pastebėjimai

- Testuojant esamą kodą, atsiranda tikimybė, kad testas bus tiesiog kodo reflekcija atkartojanti tas pačias klaidas
- Sudėtinti testai prašosi būt testuojami



- Rašyti 100% kodo padengimo testus - beprasmiška
- Netestuokit tik tam, kad testuoti - iškelkite testams konkrečius tikslus
- Metodo testas turi simuliuoti ne tik sėkmingus atvejus, bet ir klaidas

Įrankiai vienetiniams testams

PHPUnit

De facto standartas.

Integraciniai testai

- Atskirų komponentų integracijos testavimas
- Naudingi, kai nepasitikima komponento Unit testais
- Tingima rašyti Unit testus

Įrankiai integraciniams testams

- PHPUnit

Funkciniai testai

- Funkcionalumo testavimas per vartotojo sąsają
- Testuojama ar nuorodos grąžina tai, ko tikimės
- Norint užtikrinti, kad vartotojo sąsaja grąžina tai, ko tikimės

Įrankiai funkciniam testavimui

- Symfony2 WebTestCase
- PHPUnit + CURL
- bet koks Unit testų karkasas + bet koks PHP web crawleris

Atitikties (acceptance) testai

- Testai tikrinantys ar sistema atitinka funkcinius reikalaivmus
- Užtikrina ne tik vartotojo sąsajos funkcionalumą...
- ... bet ir atlieka papildomus sąsajos tikrinimus
- Ar sąsajos elementai vietose ir funkcionuoja
- Ar teisignai veikia JavaScript
- Ar tenkina programos performance

Paprasti įrankiai, paprastas naudojimas

Atitikties testai skirti testuoti funkcinius reikalavimus, todėl yra kuriami įrankiai, kuriais pasinaudoję žmonės (ne programuotojai) atsakingi už funkcinius reikalavimus, gali patys rašyti automatinius testus.

Acceptance testas parašytas paprasčiausia anglų kalba:

```
I WANT TO SIGN IN
I am on page '/login'
I fill field ['signin[username]', 'davert']
I fill field ['signin[password]', 'qwerty']
I click 'LOGIN'
I see 'Welcome, Davert!'
```

Įrankiai atitikties testavimui

- Selenium IDE
- Codeception
- Mink

Kodo peržiūra (code review)

- Rankinis kodo tikrinimas
- Reikalingas kompetetingas žmogus
- Padeda aptikti architektūrines klaidas

TDD ir BDD

- Test-driven development orientuotas į developerio sprendimus kaip turi programa veikti
- Behavior-driven development orientuotas į vartotojo lūkesčius kaip turi programa elgtis
- TDD paremtas iteratyviu vienetinių testų rašymu
- BDD paremtas iteratyviu acceptance testų rašymu

TDD ir BDD įrankiai

- PHPUnit
- BeHat
- Codeception

Apibendrinimas

- Visos šios testavimo metodikos padeda išvengti netikėtumų keičiant kodą
- Apjungus automatinius testus su tęstine integracija (continuous integration), turėsime aplinką greitai reaguojančią į sugadintą kodą
- Turėdami klaidų prevenciją galim pasiekti continuous delivery
- Nuo continuous delivery nedaug trūksta iki continuous deployment
- Testai leidžia ramiau miegoti

Nuorodos

http://codeception.com/
http://phpunit.de/
http://behat.org/

Ačiū!

Klausimai?

---

twitter: @kuusas
email: zilvinas@estina.com