

A Pragmatic Introduction to (PHP) Unit Testing

Vienna PHP, March 2015

Peter Kofler, 'Code Cop'

@codecopkofler

www.code-cop.org

Copyright Peter Kofler, licensed under CC-BY.

Peter Kofler

- Ph.D. (Appl. Math.)
- Professional Software Developer for 15 years
- “fanatic about code quality”
- I help development teams



Training on the Job?



Yes, some but...

- only what is already there
- Trial & Error not popular in production
- no practice - only production
- time pressure



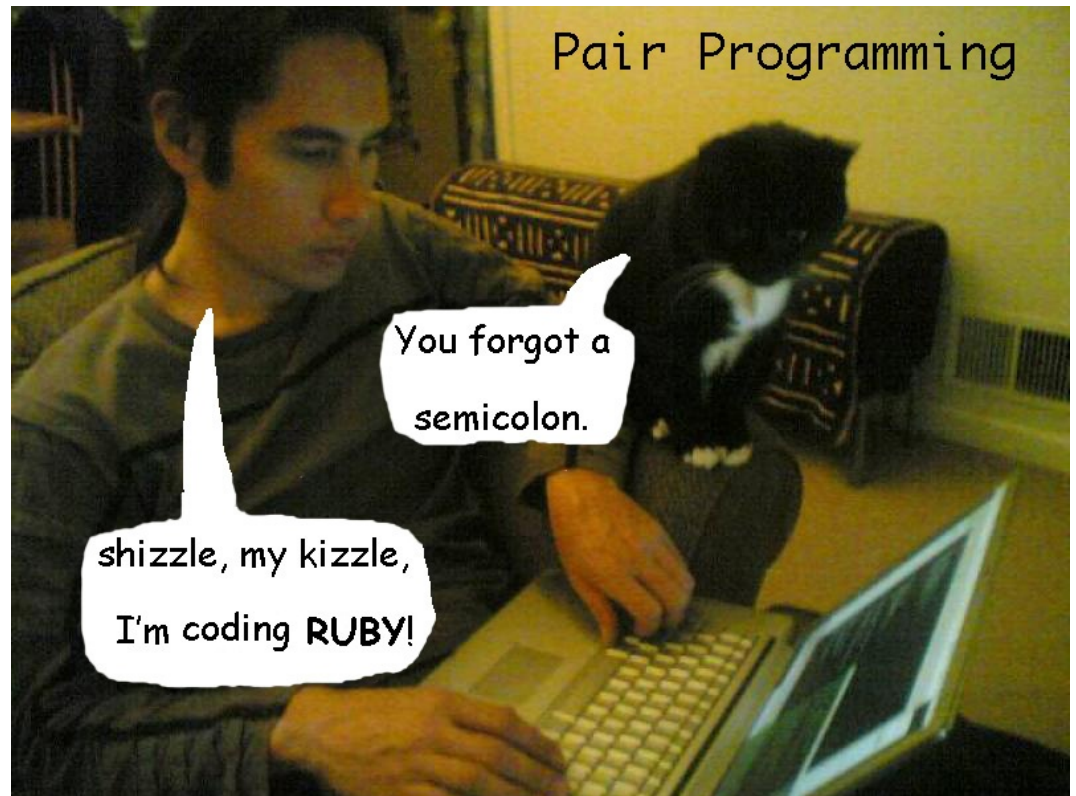
I help development teams with

- Professionalism
- Quality and Productivity
- Continuous Improvement



Mentoring

- Pair Programming
- Programming Workshops
- Deliberate Practice, e.g. Coding Dojos



Developing Quality Software Developers

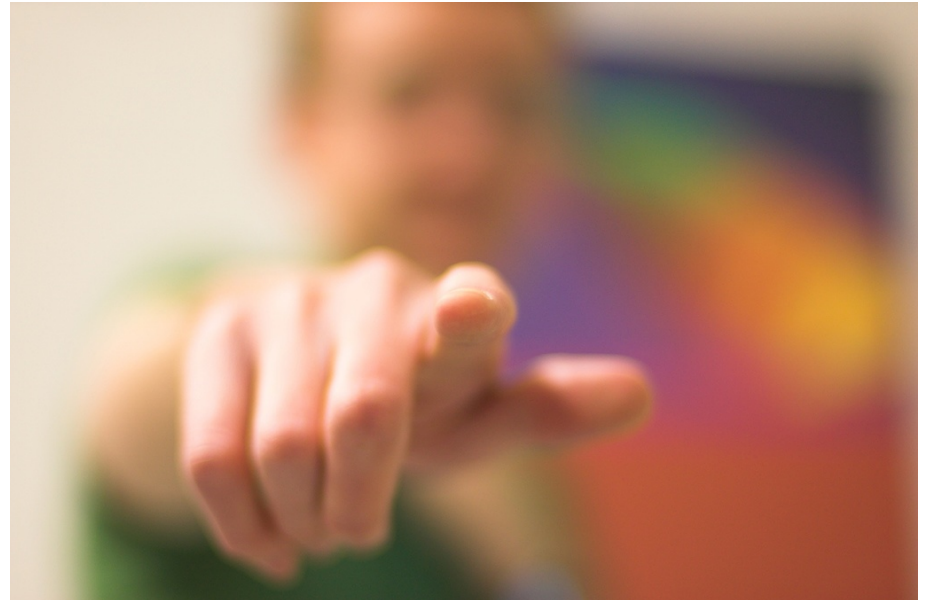
Agenda

- Unit Testing
- PHPUnit
(Coding)
- Break
- PHPUnit
(Coding)
- Retrospective



Quick Poll

- Who has heard of PHPUnit? (xUnit)
- Who has never written a unit test?
- Who has written a few?
- Who writes unit tests every day?
- Who does TDD?



We Make Mistakes

- at least I do... 😊
- number of bugs proportional LoC
 - 7 bugs per 1.000 LoC
 - 1 bug per 10.000 LoC in critical software
- Assume you have lots of bugs.



TESTING
I FIND YOUR LACK OF TESTS DISTURBING.

What is a
Unit Test?

Unit Test (Micro Test)

- code written by a developer
- tests an individual unit
 - isolate each part
- shows that the individual part is correct
- sort of living documentation

Unit of Work

- single logical functional use case
- invoked by some public interface
 - a single method,
 - a whole class or
 - multiple classes
- with one single logical purpose

What is it
made of?

Test Class

- **unit test** tests the functionality of a unit
- for various inputs and outputs
- usually a single test (class/script)

```
class FooTest extends  
    \PHPUnit_Framework_TestCase  
    { ... }
```

contains all test cases for Foo




Test Methods

- a **test case** tests the response of a single method to a particular set of inputs
- multiple test cases for a single method
- test methods should be short, simple
- tests without test methods are pointless

```
(public) function testMethod() {...}  
/** @test */ function anyName() {...}
```

Only one aspect/feature

- tests work best with lots of small tests
- tests only fail because of one reason
- more than 1 assert in test
→ split into multiple tests

`testHighAndLowInput()`   `testHighInput()`
`testLowInput()` 

- 1 test case – 1 method – 1 assertion

Assertions

- no output from your tests!
- check expectations programmatically
- e.g. `$this->assertEquals,`
`$this->assertTrue, ...`
- test method without assert is pointless
- one test method - one assertion
- test runner reports failure on each test
→ **Regression Testing**

Fixtures

- sets up data needed to run tests
- **functions** `setUp()`, `tearDown()` **or**
`/** @before */`, `/** @after */`
- Test data is more likely to be wrong than the tested code!

What should
we test?

The Right BICEP

- **R****ight**: Are the results **r****ight**?
- **B**: All the **b**oundary conditions correct?
- **I**: Can we check **i**nverse relationships?
- **C**: Can we **c**ross-check results using other means?
- **E**: Can we force **e**rror conditions?
- **P**: Are **p**erformance characteristics within bounds?

“I write unit tests for one
reason: so my co-workers
don't f*** up my code.”
(David Angry)

A Unit Test
should ...

Focus on Behaviour

- e.g. requirements are behaviour
- names from (problem) domain
- full sentences
(long, descriptive test method names)
- expected behaviour **should**
- separation per business module

Consist of 3 Simple Steps

- Prepare Input – **A**rrange
- Call Method – **A**ct
- Check Output – **A**ssert
- Use in form of **Given-When-Then**
- No conditional logic (→ more test cases)
- No loops (→ parametrized test)

```
/** @dataProvider <static function> */
```

Be F.I.R.S.T

- Fast
- Isolated (data, sequence)
- Repeatable
- Self-verifying
- Timely

Test Code Quality
must be equal
Prod. Code Quality

Try PHPUnit yourself



Coding Dojo Mindset

- Safe place outside work
- We are here to learn
- Need to slow down
- Focus on doing it right
- Collaborative Game



Assignment

- Find a pair.
- Get <https://bitbucket.org/pkofler/phpunit-koans>
- Run `phpunit` – should see no tests
- Go through the test code
 - assertions commented/ incomplete
 - uncomment the assertions
 - and complete them making tests pass

→ Practice

Closing Circle

- What did you learn today?
- What surprised you today?
- What will you do differently in the future?





Peter Kofler



@codecopkofler

www.code-cop.org

CC Images

- green <http://www.flickr.com/photos/43442082@N00/296362882/>
- Hamster <http://www.flickr.com/photos/zebrapares/4529836138>
- Bruce <http://www.flickr.com/photos/sherpas428/4350620602/>
- pairing <http://www.flickr.com/photos/dav/94735395/>
- agenda <http://www.flickr.com/photos/24293932@N00/2752221871/>
- hands <https://www.flickr.com/photos/ninahiironniemi/497993647/>
- Dojo <http://www.flickr.com/photos/49715404@N00/3267627038/>
- wants you <http://www.flickr.com/photos/shutter/105497713/>