github.com/braddle/php-uk-2023

TDD Workshop

PHP UK Conference - 14/02/2023

Mark Bradley
Senior Consulting Software Engineer

@braddle

Armakuni

About me

Software Engieneering Consultant for Armakuni

Cyclist, Photographer & Board Gamer

Test Driven Development

youtube.com/c/TestingAllTheThings

Twitter

twitter.com/braddle

What will we cover?

- Describe the concept & benefits of Test Driven Development (TDD)
- Describe the **red**, **green**, **refactor** cycle
- Describe the different Types of Tests
- Practice TDD using ping-pong pair programming style
- Identify different types of Refactoring
- Identify different types of test doubles

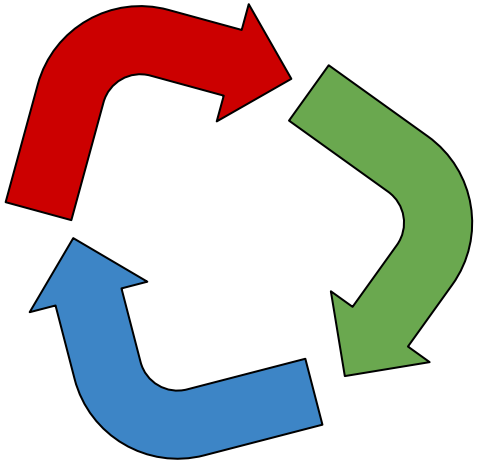
What?

What is TDD?

Writing tests before you write new code

Using tests to design how an application should work

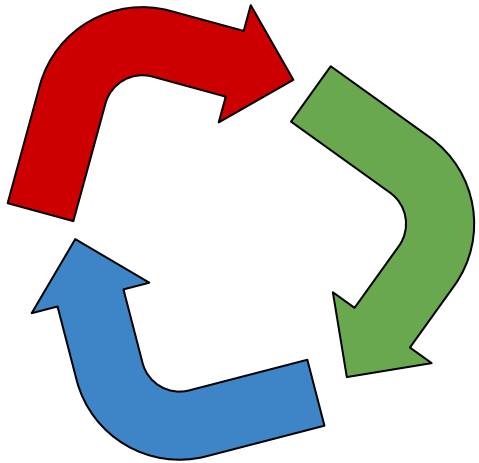
Not just unit tests



Red

Write a failing Test & see it fail

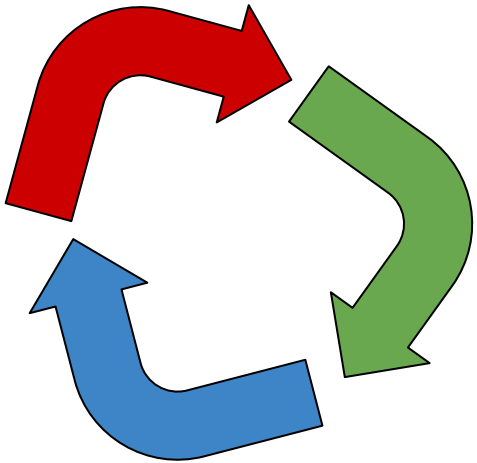
- Make sure you are testing something
- Ensure you see an error/failure you are expecting



Green

Make the failing test pass

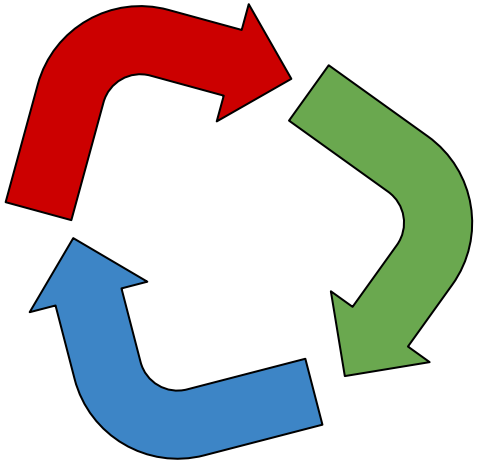
- Write just enough code to fix the error you see
- Don’t think about the whole solution



Refactor

Clean up the code

- Design the code to be easily understandable and maintainable
- always keeping the tests passing
- one refactoring technique at a time



Demo 1

Stack

First In Last Out

Arrange - Act - Assert

```php
public function testIsGreaterThan()
{
    // Arrange
    $five = new Integer(5);
    $four = new Integer(4);

    // Act
    $isGreaterThan = $five->isGreaterThan($four);
    $notGreaterThan = $four->isGreaterThan($five);

    // Assert
    $this->assertTrue($isGreaterThan);
    $this->assertFalse($notGreaterThan);
}
```

```php
public function testIsGreaterThan()
{
    // Arrange
    $five = new Integer(5);
    $four = new Integer(4);

    // Act & Assert
    $this->assertTrue($five->isGreaterThan($four));
    $this->assertFalse($four->isGreaterThan($five));
}
```

Why?

Why use TDD

Fewer defects in your code

Only implement what is required

Increased code quality

Test First vs Test After

Boring

Tests are hard to write

Easy to start skipping test

_

Fun!

Easy to write

You ~~don't~~ can't skip tests

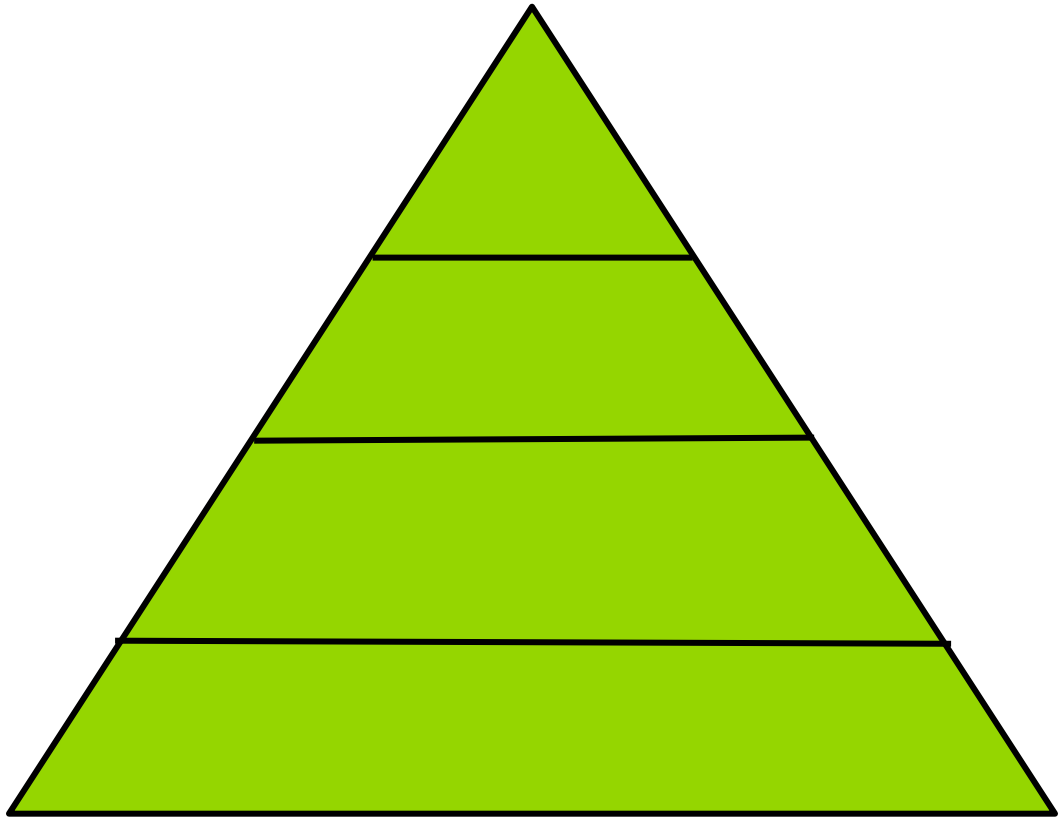
Lab 1

Queue

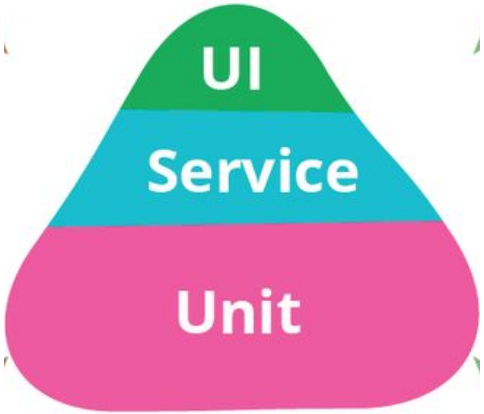
First In First Out (FIFO)

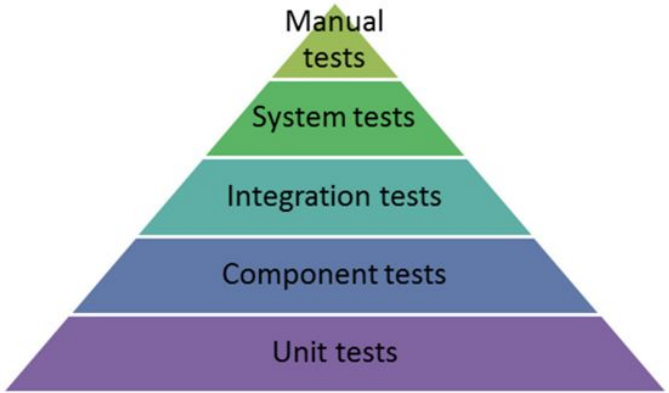
Types of Tests

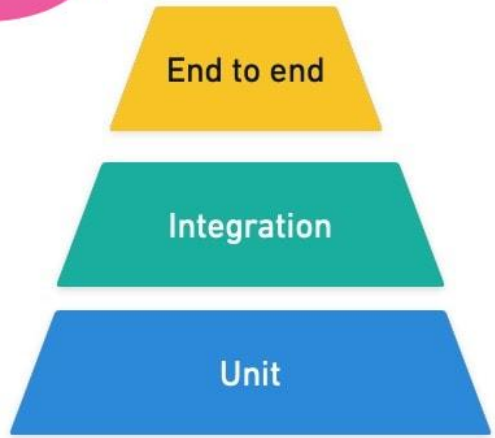
What is this?

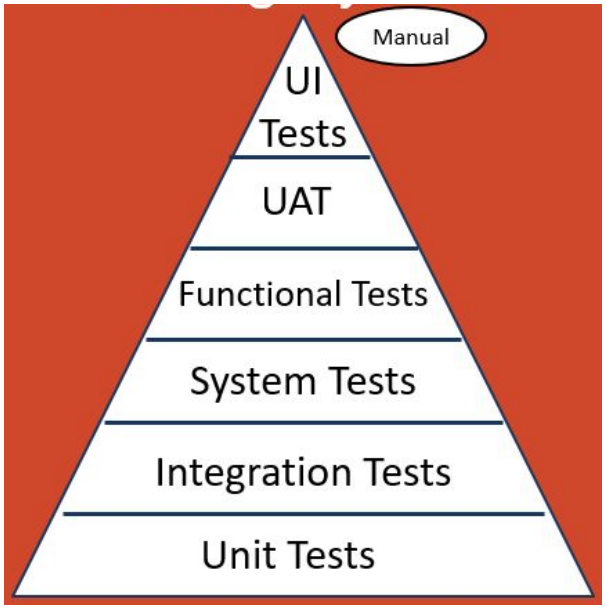


Testing Pyramid



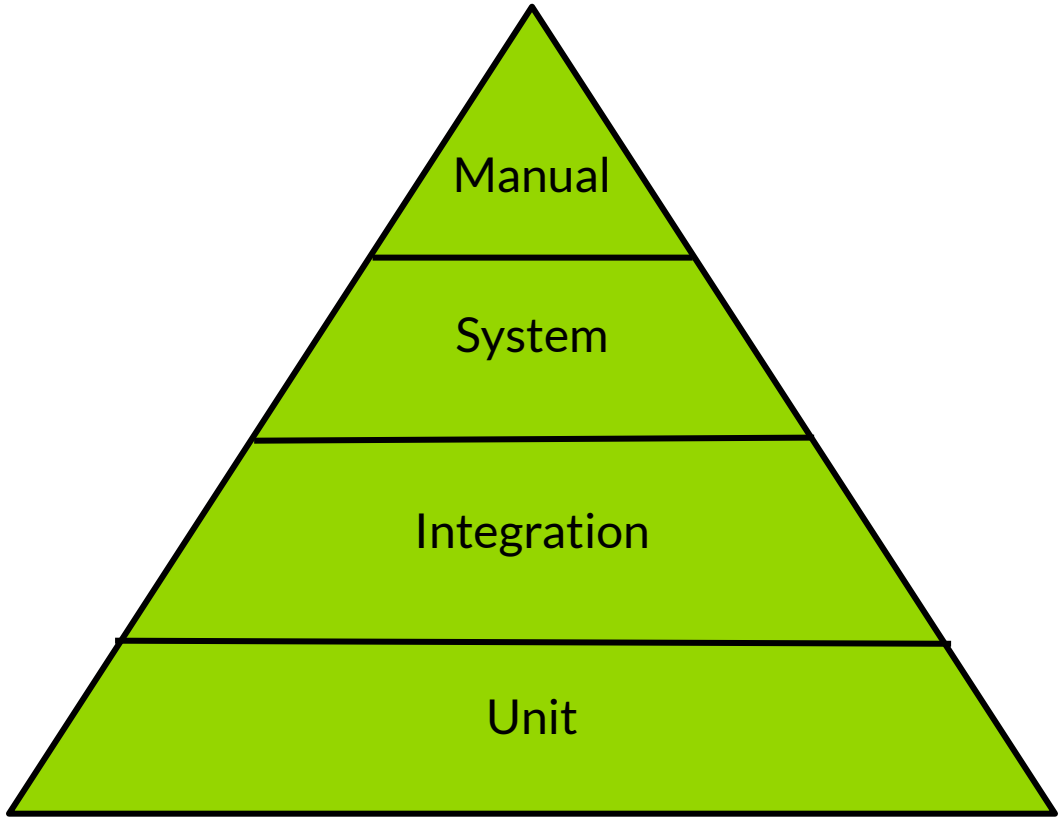






@braddle

Testing Pyramid



Unit Tests

Targeted

Isolated

Repeatable & predictable

Fast

100% Code Coverage

Integration Tests

Tests a small number of units or component together

Ensure the different units or component work together as expected

Mocks external dependencies (Database, Email, ….)

Could be behavioural Tests

10% Code Coverage

System (End to End) Tests

Flows through you application, usually a few core journeys

Uses all really services (Database, Email, …)

May interact with many different parts of you app in a single test

May require some seeding of external services

5% Code Coverage

Manual Tests

Still has its place

Exploratory Testing

Penetration Testing

Load Testing

Dependencies

Task 1

Discover different Test Doubles?

https://martinfowler.com/bliki/TestDouble.html

https://learn.microsoft.com/en-us/archive/msdn-magazine/2007/september/unit-testing-exploring-the-continuum-of-test-doubles

Test Doubles

Imitating the functionality of dependencies

Allowing for isolation of unit and integration tests

Do not necessarily require a mocking framework

Implement the interface of a dependency

Stubs

Implements the interface

Returns specific values

Spies

Implements the interface

Returns specific values

Tracks call count and calling values

Mocks

Implement the interface

Returns specific values

Tracks call count and calling values

Setup by the test

Fakes

Implement the interface

Will contain some “Real” business logic

Demo 2

Driving Licence Number Generator

Test Doubles

Lab 2

Driving Licence Number Generator

Mockery

Refactoring

Refactoring

Refactoring is a systematic process of improving code without creating new functionality that can transform a mess into clean code and simple design

Task 2

Discover a New Refactoring Technique

https://refactoring.guru/refactoring/catalog

Demo 3

Leap Year Calculator

Lab III

Roman Numeral Converter

Arabic -> Roman Numerals

Resources

- 30 Days of TDD by James Bender (@jamesbender)
  - http://www.telerik.com/blogs/30-days-tdd-day-one-what-is-tdd
- Code Coverage: Testing Private Functions (Me)
  - http://mark-bradley.net/2017/03/11/code-coverage-testing-private-functions/
- Testing All The Things
  - youtube.com/c/TestingAllTheThings
-

Katas & Dojos

- codingdojo.org/kata
- Kata-log.rocks

- meetup.com/London-Code-Dojo/
- meetup.com/london-software-craftsmanship/

Feedback

Twitter: @braddle

Email: braddle@gmail.com

Retrospective

Thank you

@braddle