TDD (with FLOW3)

Karsten Dambekalns <karsten@typo3.org>
What?
Why?
How?
Test first, code later
TDD is about Tests

- Write tests for all your code
  - New features come with a test
  - Bugfixes come with a test
TDD is about Tests

- Write tests for all your code
  - New features come with a test
  - Bugfixes come with a test
- Write tests before your code
  - Adding a feature means adding a test first
  - Fixing a bug means writing a test first
The TDD Mantra

Design
The TDD Mantra

Design

Test fails
The TDD Mantra

Design

Test fails

Implement

Implement

Design
The TDD Mantra

- Design
- Implement
- Test passes
- Test fails

FLOW3
The TDD Mantra

- Test passes
- Test fails
- Design
- Implement
Unit tests

Unit tests are small programs that test your code

They check

- the result of methods against expected values
- whether methods are (not) called as expected

A test should

- be well-named
- check only one assertion
Good tests

Should be...

- automated
- self-contained
- repeatable
- thorough
- small
- talk about the domain
Running unit tests

- Unit tests are usually run with a test runner
- The xUnit family is most widespread
- PHPUnit does the job for PHP
- FLOW3 has a Testing package
  - acts as test runner
  - provides helpful environment
  - web-based and CLI tool
Running unit tests

Unit tests are usually run with a test runner

The xUnit family is most widespread

PHPUnit does the job for PHP

FLOW3 has a Testing package

• acts as test runner
• provides helpful environment
• web-based and CLI tool
Running unit tests

Unit tests are usually run with a test runner. The xUnit family is most widespread. PHPUnit does the job for PHP. FLOW3 has a Testing package that acts as test runner, provides helpful environment, and is a web-based and CLI tool.
Inspiring people to share

Running unit tests

Unit tests are usually run with a test runner. The XUnit family is most widespread. PHPUnit does the job for PHP. FLOW3 has a Testing package:

• acts as test runner
• provides helpful environment
• web-based and CLI tool
Tests aren’t the goal
They make you feel good
They make you feel good

What the tests do for you is the key

The tests make you

- focus on your task
- code exactly what you need
- think from the outside
- a better programmer*
They make you feel good

- What the tests do for you is the key
- The tests make you
  - focus on your task
  - code exactly what you need
  - think from the outside
  - a better programmer*

* decoupling helps with testing, decoupled code is easier to maintain, easier is better – q.e.d.
Use TDD to...

- Feel more comfortable
  - Build confidence in your code
  - Reduce fear of change

- Have a safety net
  - Regression testing built in
  - Helps with refactoring

- Provide (good) documentation through (good) tests
D for Development
D for Design
D for Design

- Writing tests first is likely to improve your code
  - You use the API before you code
  - You make the API fit your needs
- Unit testing ensures decoupling
- You only code what is really needed
- Constant refactoring keeps code clean
TDD in Practice

- Write down your next goal
- Now write a test to check this
- The test will fail, it might even break with a fatal error
- Now write the code you need to write
- If you test again, the test should now pass
- Iterate – until it passes or all features are in
No FLOW3?
No TDD!
Dependencies

- Classes explicitly refer to other classes
- But you want to test a small unit
- You don't want to test
  - The Steamer
  - The Grinder
- You want to test
  - if the Barista uses the right amount of coffee and the requested milk when asked for coffee
class Barista {

/**
 * Makes a drink
 *
 * @param string $drink
 */
 public function make($drink) {
  if ($drink === 'Tea') {
    throw new F3::Demo::Exception::NotAvailable('We don\'t serve no tea, Sir', 1223385110);
  }

  $coffeeGrinder = new Grinder();
  $steamer = new Steamer();

  $coffee = $coffeeGrinder->grind(9, Grinder::FINE);
  $milk = $steamer->getSteamedMilk(Steamer::FAT_LOW);
}
}
Dependency Injection

- This methodology is referred to as the "Hollywood Principle": "Don't call us, we'll call you"
- A class doesn't ask for the instance of another class but gets it injected
- Enforces loose coupling and high cohesion
- Allows you to mock collaborators
- Makes you a better programmer
class Barista {

/**
 * @var F3::Demo::Grinder
 */
protected $grinder;

/**
 * @var F3::Demo::Steamer
 */
protected $steamer;

/**
 * @param Grinder $grinder
 * @param Steamer $steamer
 */
public function __construct(F3::Demo::Grinder $grinder, F3::Demo::Steamer $steamer) {
    $this->grinder = $grinder;
    $this->steamer = $steamer;
}
/**
 * Makes a drink
 *
 * @param string $drink
 */

public function make($drink) {
    if ($drink === 'Tea') {
        throw new F3::Demo::Exception::NotAvailable(
            'We don\'t serve no tea, Sir', 1223385110);
    }

    $coffee = $this->grinder->grind(9, Grinder::FINE);
    $milk = $this->steamer->getSteamedMilk(Steamer::FAT_LOW);
}
Without Dependency Injection you cannot do unit testing
So, what to inject?
Mocks & Stubs

- Sometimes you need to test interaction with external systems, like milk steamers.
- A test should be small, encapsulated, stand-alone.
- Mock objects allow you to use fake objects.
- Stubs can be used to return hard-coded results.
- Using them is actually (somewhat) easy with PHPUnit.
FLOW3 + TDD = FUN

- FLOW3 makes Dependency Injection easy
- With Dependency Injection you can do proper unit testing
- Proper unit tests
  - help produce reliable code
  - make refactoring possible
  - make you feel better
- All this means more fun while coding
Questions!
Literature

Test-Driven Development By Example
Kent Beck, Addison-Wesley

Continuous Integration – Improving Software Quality and Reducing Risk
Paul M. Duvall, Addison-Wesley

xUnit Test Patterns – Refactoring Test Code
Gerard Meszaros, Addison-Wesley
Links

FLOW3
http://flow3.typo3.org/

PHPUnit
http://www.phpunit.de/

TDD in Wikipedia
http://en.wikipedia.org/wiki/Test-driven_development