

PHP AUTOMATIC UNIT TESTING

By Eddo Rotman

- I wrote the code, it's working
- Writing tests is not my job, the testers should do that
- We don't have time to write tests, we have a deadline
- Writing tests is boring...
- My application is too complex to separate into different components for proper unit testing
- In a nutshell, unit test will not work for our application. You just don't understand the situation here, I can't explain it to you.

- What is a Unit?
 - The smallest possible logic element of the application
 - The smaller the unit is, the more testable it is
- What is a Unit Test?
 - A test which ensures that an individual unit of the application is working properly, according to the software's specifications
 - The smaller the unit is, or low-level, the more elaborate the unit tests should be for it
 - The Unit Tests should check how the unit acts both on a valid input *but also* how it acts on invalid input

Why use Unit Tests?

- Better code
 - You know it actually works
- Better design
 - Unit tests help understand the design and encourage decoupling
- Regression tests
 - You can improve your code and refactor it and still be sure it works
- Safer bottom-up integration
 - If the foundations are strong, the project is strong
- Bug fixing aid
 - Simulate the environment to isolate the problem

Who should write the tests?

- ***The programmer writing the application***
 - It requires detailed knowledge of the internal program design and code
 - May require developing test driver modules or test mock objects
- The tester / programmer
 - Can *add* additional tests to verify buggy behavior
- The product manager / owner
 - Can write specifications which can later become the unit tests (PHPFit)

- Three Unit Testing tools which will be discussed
 - PHPT
 - PHPFit
 - PHPUnit

PHPT

- PHPT is a file-based testing framework for PHP's functionality
- The PHPT files scripts which define the input and expected output of the test
- Each PHPT has at least three parts
 - `--TEST--` - a one line short description of the test
 - `--FILE--` - the code to be run as the test
 - `--EXPECT--` / `--EXPECTREGEX--` / `--EXPECTF--` - the expected result
- The PHPT file may have more parts such as
 - `--SKIPIF--`
 - `--CLEAN--`
 - `--INI--`

PHPT (cont)

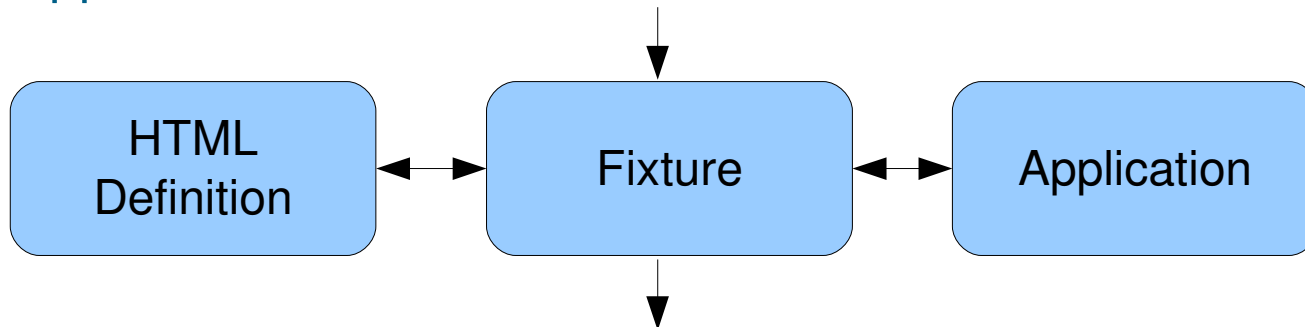
- Example:

```
--TEST--
strtr() function - basic test for strtr()
--FILE--
<?php
$strans = array(  "hello"=>"hi",
                  "hi"=>"hello",
                  "a"=>"A",
                  "world"=>"planet");

var_dump(strtr("# hi all, I said hello world! #", $strans));
?>
--EXPECT--
string(32) "# hello All, I sAid hi planet! #"
```

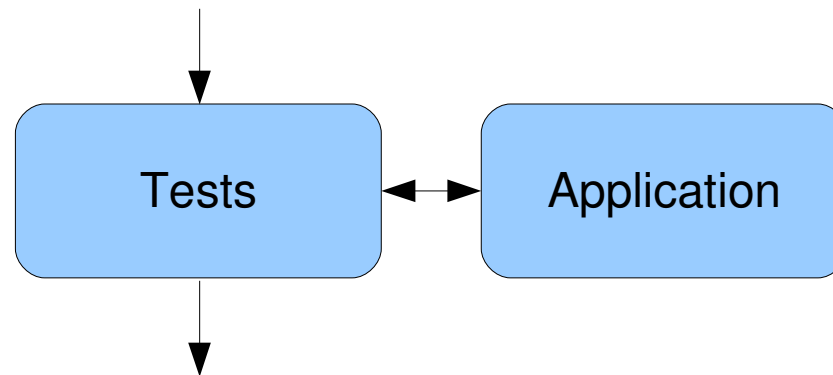
PHPFit

- PHPFit is a port for PHP5 of the FIT acceptance testing framework originally developed for Java
- PHPFit is a three-parts system
 - HTML Definition – can be created by anyone and describes how the application should act; can be edited with any WYSIWYG HTML editing tool
 - Fixture – should be created by a PHP programmer; connects the HTML definition to the actual Application
 - Application



PHPUnit

- PHPUnit is a PHP5 testing framework
- PHPUnit is a dual-layered system
 - Tests
 - Application
- All code written for a PHPUnit test is in PHP5



PHPUnit (cont)

- **Example:**

```
class Test extends PHPUnit_Framework_TestCase {

    /**
     * Prepares the environment before running a test.
     */
    protected function setUp() {
        parent::setUp ();
    }

    /**
     * Cleans up the environment after running a test.
     */
    protected function tearDown() {
        parent::tearDown ();
    }

    /**
     * Constructs the test case.
     */
    public function __construct() {
    }

    /**
     * A Test
     */
    public function testSomeMethodWhichShouldBeTested() {
    }

}
```

Comparison

- PHPT
 - Used almost entirely for testing PHP itself by the PHP community
 - Good for continuous integration automatic testing
- PHPFit
 - Good for projects where there is the option to separate the test writing from the code writing
- PHPUnit
 - Good for clear-cut cases of testing units
 - Can be run in command line mode, browser & with Zend Studio
 - Gives additional information like code-coverage

Limitations of Unit Testing

- Keep in mind that Unit Tests only test the functionality of the units themselves
 - They do not test integration errors
 - They do not test performance
 - They do not test usability
 - They do not test GUI
- Writing tests can be time consuming – but it's worth it!
- Test cases will need maintenance if requirements change

Conclusions

- Start using unit tests, turn it into a habit
 - When you program
 - When you fix bugs
- Choose whichever testing tool is best for your project
 - Keep in mind that *print()*, *print_r()*, *var_dump()* etc. are not testing tools

MUST HAVE A QUOTE



Program testing can be used to show the presence of bugs, but never to show their absence!

Edsger Dijkstra