eZ®
Share your information

# Dutch PHP Conference - Amsterdam, The Netherlands

Derick Rethans - dr@ez.no

http://derickrethans.nl/talks.php

eZ
Share your information

# About Me

- Dutchman living in Norway
- eZ Systems A.S.
- eZ Components project lead
- PHP development
- mcrypt, input_filter, date/time support, unicode
- QA

# About Testing

# Web Application Testing

## Testable Parts

Front-end

- Acceptance Tests and System Tests that run in the browser
- Testing of Web Services with Unit Tests
- Compatibility Testing for Browser/OS/etc. combinations
- Performance Testing
- Security Testing

Back-end

- Functional Testing of business logic with Unit Tests
- Reusable Components, but they often come with their own tests

# Testing Methods

## Unit Testing

Tests small parts of an application or library (units) for correctly working code. Tools: PHPUnit, SimpleTest

## System Testing

The testing of a whole integrated system against the specified requirements. Tools: Selenium

## Non-functional Testing

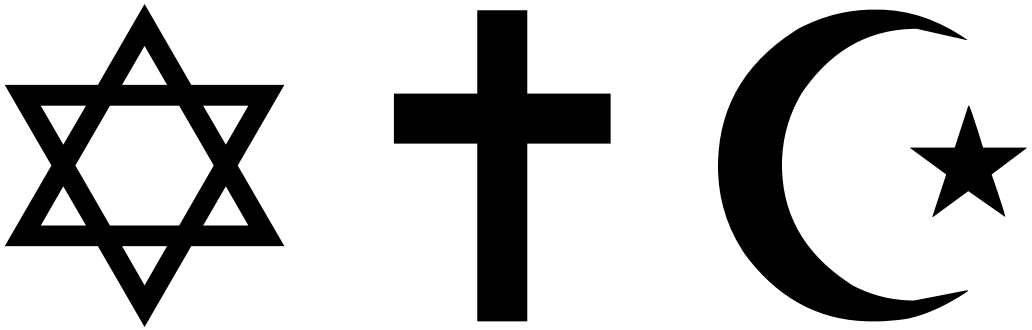
Testing for performance, load, stress, reliability, availability, security. Tools: ab, siege, httperf, chorizo

# Test-Driven Development

# Test-Driven Development

It is not just a method of testing software.

# It is a religion.



# Back In Time

## Traditional Development Phases

### Requirements Specification

Define what the software is supposed to do.

### Design

Define how the software is supposed to be implemented.

### Implementation

The implementation of the software itself.

### Testing

The implemented software is tested.
Sometimes.

# Tests are King

## Tests drive the development

- Tests are written before the code
- There is no code without tests

## Test Suites

- Contain tests that check whether the code does what it is supposed to do
- Also cover things that should fail

# Into the Future

## Test-Driven Development Phases

### Requirements Specification

Define what the software is supposed to do.

### Design

Define how the software is supposed to be implemented.

### Implementation ≡ Testing

The implemented software is tested.
The implementation of the software itself.

## Present the Idea

## Write the Requirements Document

## Design the Component

## Implementation

- Write API stubs with parameter documentation and descriptions
- Write test cases
- Initial implementation
- Initial implementation review
- Updating implementation according to review
- Implementation review

## Pre-release Testing

# Test-Driven Development

## Addressing Defects

### Write Test Case

Write a test case to test the correct behavior of the method, and verify that the test case fails

### Fix the Issue

Fix the implementation

### Verify Fix with Test Case

Make sure that the test cases pass with the new implementation

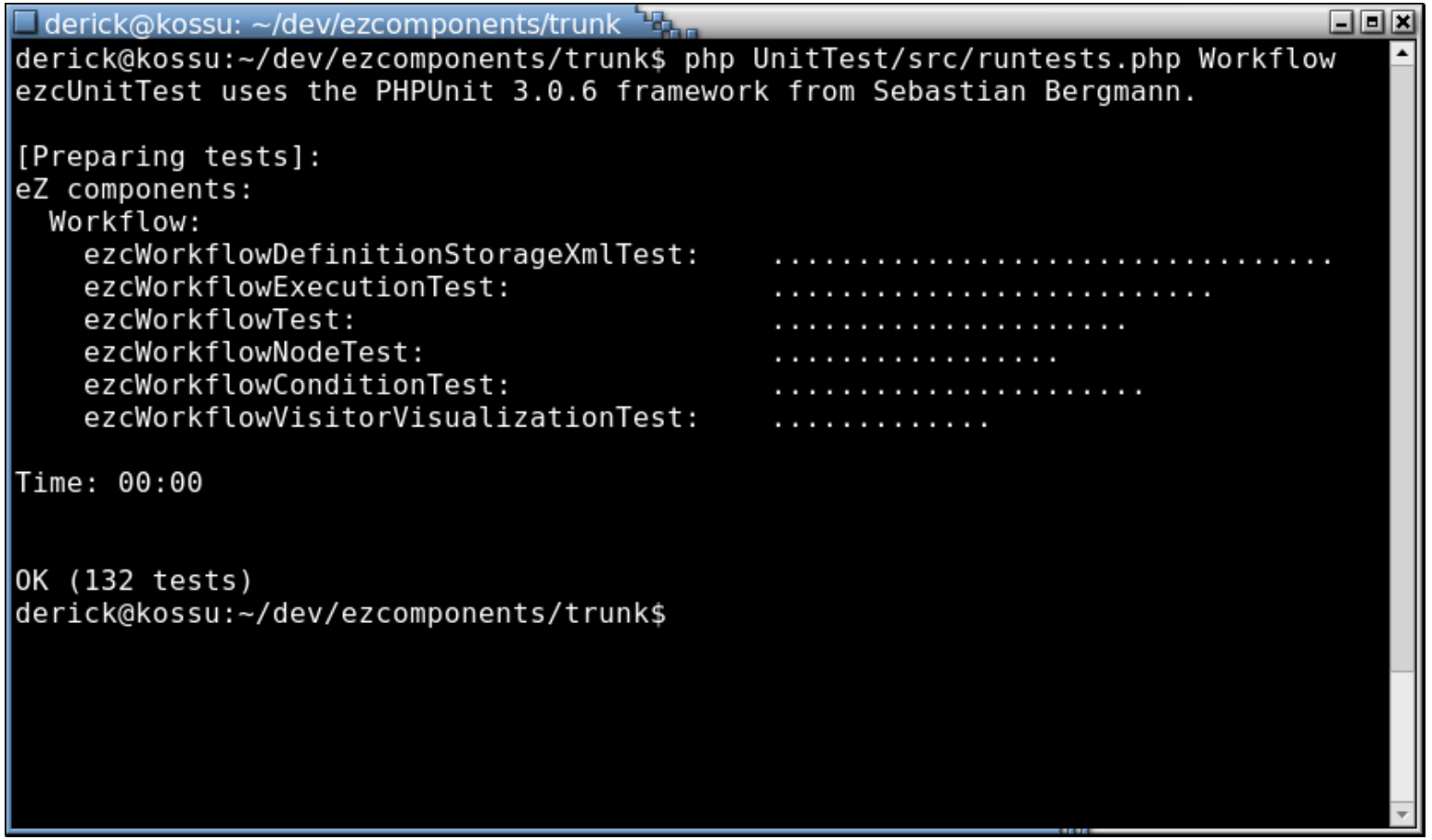
### Check Other Test Cases

Verify that the fix for this defect did not break any other test case

# PHP Unit

# PHPUnit

## Unit Testing Framework for PHP



# PHPUnit

## Code Coverage: No Code Without Test

Current view: /home/derick/dev/ezcomponents/trunk/Graph/src

Date: Thu Jun 14 11:04:17 CEST 2007

Executable lines: 8756

Code covered: 96.04%

Executed lines: 8409

Legend: Low: 0% to 35% | Medium: 35% to 70% | High: 70% to 100%

| | Coverage (show details) | | |
|---|---|---|---|
| axis | | 90.67% | 612 / 675 lines |
| charts | | 99.42% | 518 / 521 lines |
| colors | | 100.00% | 222 / 222 lines |
| data_container | | 100.00% | 41 / 41 lines |
| datasets | | 93.88% | 230 / 245 lines |
| driver | | 96.51% | 1217 / 1261 lines |
| element | | 98.26% | 451 / 459 lines |
| exceptions | | 98.89% | 89 / 90 lines |
| interfaces | | 91.58% | 859 / 938 lines |
| math | | 99.72% | 359 / 360 lines |
| options | | 99.09% | 762 / 769 lines |
| palette | | 100.00% | 8 / 8 lines |
| renderer | | 95.71% | 2811 / 2937 lines |
| structs | | 100.00% | 39 / 39 lines |
| graph.php | | 100.00% | 2 / 2 lines |
| graph_autoload.php | | 100.00% | 103 / 103 lines |
| tools.php | | 100.00% | 86 / 86 lines |

*Generated by: PHPUnit 3.0.6 and Xdebug 2.0.0RC5-dev.*

# PHPUnit

## Code Coverage: No Code Without Test

```
151                 :     /**
152                 :      * Ensure proper timestamp
153                 :      *
154                 :      * Takes a mixed value from datasets, like timestamps, or strings
155                 :      * describing some time and converts it to a timestamp.
156                 :      *
157                 :      * @param mixed $value
158                 :      * @return int
159                 :      */
160                 :     protected static function ensureTimestamp( $value )
161                 :     {
162              19 :         if ( is_numeric( $value ) )
163              19 :         {
164              16 :             $timestamp = (int) $value;
165              16 :         }
166               3 :         elseif ( ( $timestamp = strtotime( $value ) ) === false )
167                 :         {
168               0 :             throw new ezcGraphErrorParsingDateException( $value );
169                 :         }
170                 :
171              19 :         return $timestamp;
172                 :     }
```

< 100% Coverage ≡ Not fully tested code

100% Coverage ≠ Fully tested code

# Un-testable Problems

- Testing singletons more than one time
- System/Operating System dependent tests
- Private methods
- Code that depends on the state of an external resource
- Things that simply should never happen

# Case Studies

# Case Studies

## Microsoft Case Study

- TDD project has twice the code quality
- Writing tests requires 15% more time

## IBM Case Study

- 40% fewer defects
- No impact on the team's productivity

## John Deere / Ericsson Case Study

- TDD produces higher quality code
- Impact of 16% on the team's productivity

# Case Studies
## Developer Comments

- At first I didn't like that I need to write tests for my code, but now after using it for more than 10 months I can't program without it.
- Helps to come up with better APIs.
- It gives confidence that our software is working well at all times. Even after making major changes and/or changing software we are dependent on.
- Productivity increases - you might loose some when you make the initial tests, but you'll get it back later. The code covered by tests is 'insured' against future changes.
- It works well for libraries, not so well for GUI applications.
- Some things cannot be tested like server errors

eZ®
Share your information

# Questions

# Resources

These Slides: http://derickrethans.nl/talks.php

PHP: http://www.php.net