# EFL Zürich 2012

## Unit testing in erlang

Muharem Hrnjadovic <mh@star.io>

@al_maisan

git://github.com/al-maisan/efl-zh-2012.git

# What is it?

Testing that a program unit behaves as it is supposed to do

- Used to be(?) hip
- Not a silver bullet
  - if you don't know what you're supposed to do you are doomed anyway
  - be careful with the dogmatically inclined ;-)
    → difference: methodologist vs. terrorist?

# Show of hands

- Who has written unit tests?
- In which language(s)?

- EUnit user's guide

# Why bother?

- Code correctness, documentation
- Team confidence, refactoring, regressions
- TDD (improves focus)?
- Facilitates
  - system integration
  - problem analysis, debugging

# Besides..

- Unit tests help
  - prototype, exercise and debug code
  - focus on what is needed(?)
  - avoid going down a garden path (sometimes)
- Writing code w/o tests feels “unhygienic”

# Really?

- “Testing theatre”, tests: `good xor dead weight`
- Writing *good* tests is expensive
  - weigh cost vs. benefit
  - Test code rots too and needs refactoring
- “Testing like the TSA” (not!)
  - Code-to-test ratio above 1:2 “smells” (OpenQuake: 1:1.14)
  - “wrong if testing is taking more than 1/3 of your time”

# Structure!

- Unit test suites
- setup() and teardown() functions per
  - suite
  - test class/group
  - test
- test frameworks with functions, macros
- test runner tool, runs all/selected tests

# Example 1

- Simple tests
- Macros
- Test generators
- Conditional compilation

# Example 2

- Setup
- Teardown
- The !dreaded!

```
*** test module not found ***
```

→

http://media.tumblr.com/tumblr_ljiv495hvs1qbus6u.jpg

# Example 3

- Separate tests for exported functions

# Example 4

- tests for “impure” functions

# Example 5

- functions that send results as messages

# Mocking with meck

- Functions stubs
- Mock modules
- Partial mocks
- meck's github page
- Slides, erlang factory talk, London 2011