DFUnit – Writing unit tests

# What to test where

## Unit tests

Test only what the API exposes

Give it good input

Give it bad input

### Each test class

Has many tests

But they must be independent

### Each test

Tests *one* method

Set up, test, tear down, repeat

Touches *one* class (the class you’re testing)

If the method involves another class, then *fake* the other class, otherwise you’re doing an *integration test*

## Integration tests

Test code + database

My code + your code

First unit test your pieces with fakes (stubs/mocks)

Then test them together

Why? With integration tests only you cannot definitely say “The problem is in your code” or “The problem is in the database”

## System tests

Tests that everything works together

Testing the client requires other tools, but hook into the global tests

## Acceptance testing

High level

Written before all others

Based on “User Stories”

Saves you writing code the client doesn’t want

Implies what unit tests are needed

# What to test

Mission critical = test or die

Complex = test or suffer

Non-trivial = test or waste time

Trivial = waste time (don’t test simple getters and setters)

# When to test

The sooner the better

Before writing the code?

Just after?

While writing the code?

Before fixing the bug (to confirm it’s what you think, to know when you’re done fixing, and guarantee you’ll never see the bug again)

# Getting everyone on the same page

Run all tests automatically at given intervals (i.e. at check-in)

Hook the build system (continuous integration server, anyone?) into it

# Naming conventions

Class: For each class, create at least one class with the name ClassNameTests.

Method: For each method, create at least one test with the following name: **MethodName\_StateUnderTest\_ExpectedBehavior**

# Good Unit Tests

* FAST (Many hundreds or thousands per second)
* Isolate (Failure reasons become obsolete)
* Repeatable (run repeatedly in any order, any time)
* Self-validating (no manual evaluation required. Simply pass/fail)
* Timely (written before the code)

Professional – the test code is as important as your production code, and needs to be well maintained, well designed, readable, not duplicated.

# Information stolen from:

[Write Maintainable Unit Tests That Will Save You Time And Tears](http://msdn.microsoft.com/en-us/magazine/cc163665.aspx)

[FIRST properties of unit tests](http://agileinaflash.blogspot.com/2009/02/first.html)

<http://www.masukomi.org/talks/unit_testing_talk_2/index.xul?data=slide_data.txt#page1>