"I TEST YOU NOT!"

A SHORT INTRODUCTION TO UNIT TESTING, CODE COVERAGE AND MOCK OBJECTS.

I'M JENS

@BARKODED AT TWITTER

DO YOU PRACTICE UNIT TESTING? AND DO YOU RUN 'EM ON A REGULAR BASIS?

WHY? WHY NOT?

USEFUL EVEN IF YOU DON'T PRACTICE AN "ACRONYM METHODOLOGY" SUCH AS TDD, BDD, FDD, DDD, MDD, XP OR WHAT EVER YOU CALL IT!

PERSONAL BENEFITS

1. Think before you start typing - plan your attack!
2. Don't repeat yourself - avoid keep doing the same continuously!
3. May lead to a better and cleaner design - single responsibility principle.
4. Avoid regressions
5. Ease refactoring

Testing in Xcode

- Xcode 3 was pain, pain and even more pain (debugging).
- Xcode 4 was less pain and better integration (sadly often broken and debugging issues).
- In Xcode 5 it actually works.
  - Xcode Server is another story!

Test Frameworks for Objective C

1. XCTest
2. OCUnit (Legacy)
3. GHUnit
4. Specta
5. Kiwi
6. Cedar

MATCHERS

1. OCHamcrest
2. Expecta

MOCK FRAMEWORK

1. OCMockito
2. OCMock

Integration Testing Framework

1. UIAutomation
2. KIF
3. Frank
4. Calabash

Assertions

“An assertion is a predicate (a true–false statement) placed in a program to indicate that the developer thinks that the predicate is always true at that place. If an assertion evaluates to false at run-time, an assertion failure results, which typically causes execution to abort.
- Wikipedia”

ASSERTIONS IN XCTEST

```
XCTFail (format…)
XCTAssertNil (a1, format…)
XCTAssertNotNil (a1, format…)
XCTAssert (a1, format…)
XCTAssertTrue (a1, format…)
XCTAssertFalse (a1, format…)
XCTAssertEqual (a1, a2, format…)
XCTAssertEqualObjects (a1, a2, format…)
XCTAssertEqualWithAccuracy (a1, a2, accuracy, format…)
XCTAssertThrows (expression, format…)
XCTAssertThrowsSpecific (expression, specificException, format…)
XCTAssertThrowsSpecificNamed (expression, specificException, exceptionName, format…)
XCTAssertNoThrow (expression, format…)
XCTAssertNoThrowSpecific (expression, specificException, format…)
XCTAssertNoThrowSpecificNamed (expression, specificExcepton, exceptionName, format…)
```

DEMO

Mock Object

“...mock objects are simulated objects that mimic the behavior of real objects in controlled ways...
- Wikipedia”

Code Coverage

“...code coverage is a measure used to describe the degree to which the source code of a program is tested by a particular test suite...
- Wikipedia”

ENABLE COVERAGE

Add the following to your primary target and only for a given configuration (Debug).

```
GCC_INSTRUMENT_PROGRAM_FLOW_ARCS = YES
GCC_GENERATE_TEST_COVERAGE_FILES = YES
```

Consider using ”XcodeCoverage” by Jon Reid to ease the setup.

THANKS!

Resources

- "Mocks Aren't Stubs" by Martin Fowler
- "Using Mock Objects to Stabilize Unit Tests" by Claus Broch
- "Using NSURLProtocol for Injecting Test Data" by Claus Broch
- "XcodeCoverage" by Jon Reid
- Facebook Snapshot Test Case