

Unit Testing

Sebastian Vieira
Vikram Kriplaney

Unit Testing Overview

- Test framework: OCUUnit by sente.ch
 - Added to the Developer Tools in 2005
- Logic Tests
- Application Tests

Logic Tests

- Traditional Unit Tests
 - They run on a clean-room environment
- Run from command line or build time
- They can not test application components
 - ... like User Interface elements
- Good for libraries, frameworks and model components of the application

Application Tests

- They run within a running Application
- Good for Controller and View tests
 - Test that outlets and actions are in place
 - Hardware testing
- Run as a separate step (not as part of the build)
- They run only on the device!
 - Google Toolbox or manually invoking SenTestingKit after the application starts are workarounds for this

OCCMock

- Objective-C implementation of Mock Objects
 - Creates mock objects on the fly
 - Expectation and Verification
 - `[[mock expect] someMethod:someArgument]`
 - `[mock verify]`
 - Stubs
 - Tells the mock object a method should return a value
 - `[[[mock stub] andReturn:aValue] someMethod:someArgument]`
- Argument constraints
- Protocol Mocks, Partial Mocks, Observer Mocks
- Per Instance and Per Method Swizzling

Google Toolbox

- **Install Google Toolbox GTMXCodePlugin**
 - A preference to clean up end of line white space from text files.
 - A "Create Unit Test Executable" menu item to make it easier to debug unit tests.
 - Some quick links under the "Help" menu to some useful documentation.
 - Some menu items for working with Coverstory

Setting Up Logic Tests

- Create a new Unit Test bundle target
- Create new tests from Unit Test template
 - Added only to this newly created bundle
- Drag the Logic Test target in the App Target
 - When the App Builds the Tests run
 - If tests fail the build fails
- Tests are run by octest (/Developer/tools)

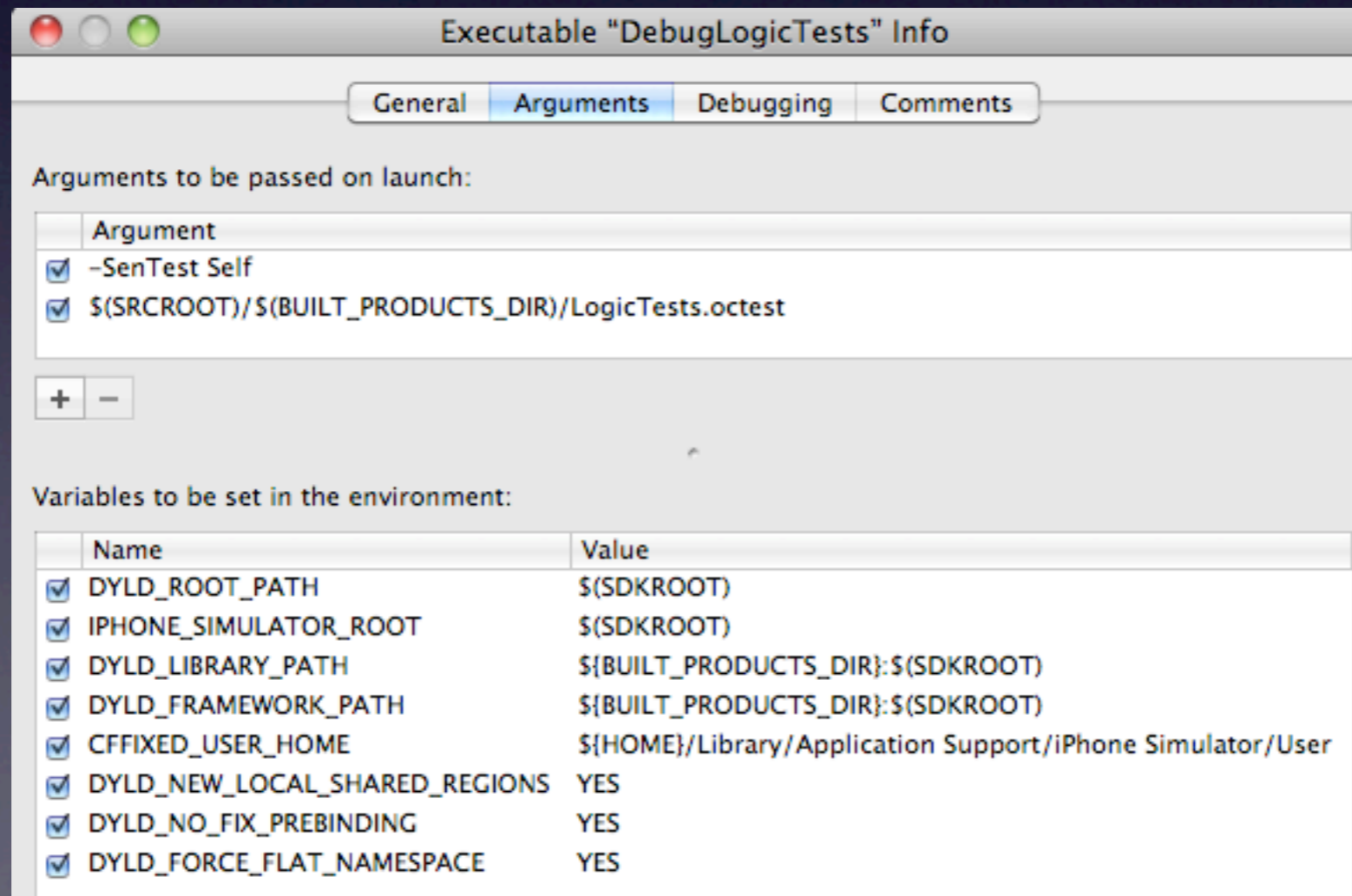
Debugging Logic Tests

- Create a duplicate of the target with tests in it
- Create another main (maintest.c)

```
int main(int argc, char *argv[]) {  
  
    SenTestSuite *suite= [SenTestSuite testSuiteForBundlePath:nil];  
  
    BOOL hasFailed = ![[suite run] hasSucceeded];  
  
}
```

Debugging Logic Tests

- Create new custom executable
 - Running Developer/usr/bin/otest (relative path to SDK)
- This executable will run our test bundle



Setting Up Application Testing

- Create a Unit Test Bundle target
 - Add APPLICATION_TESTS Preprocessor Macro to the Unit Test Target's Build Settings: GCC 4.2 - Preprocessing→Preprocessor Macros
- Create an Application as a copy of the main target
 - Drag the Unit Test bundle target to the new App
 - Drag the testing bundle (on Products) to the new App's Copy Bundle Resources phase
 - You need to keep the duplicate in Sync!
- Tests can introspect UI elements
- They can be easily debugged!

Useful Links

- **Unit Testing Guide - Introduction**
<http://developer.apple.com/mac/library/documentation/DeveloperTools/Conceptual/UnitTesting/index.html>
- **Google Toolbox**
<http://code.google.com/p/google-toolbox-for-mac/wiki/GTMXcodePlugin>
- **A sample iPhone Application with complete Unit Tests**
<http://cocoawithlove.com/2009/12/sample-iphone-application-with-complete.html>

Continuous Integration

Sebastian Vieira
Vikram Kriplaney

Continuous Integration Overview

- Needed to do builds and to run tests
- At local we use Bamboo from Atlassian
- This time we will use Hudson

Hudson

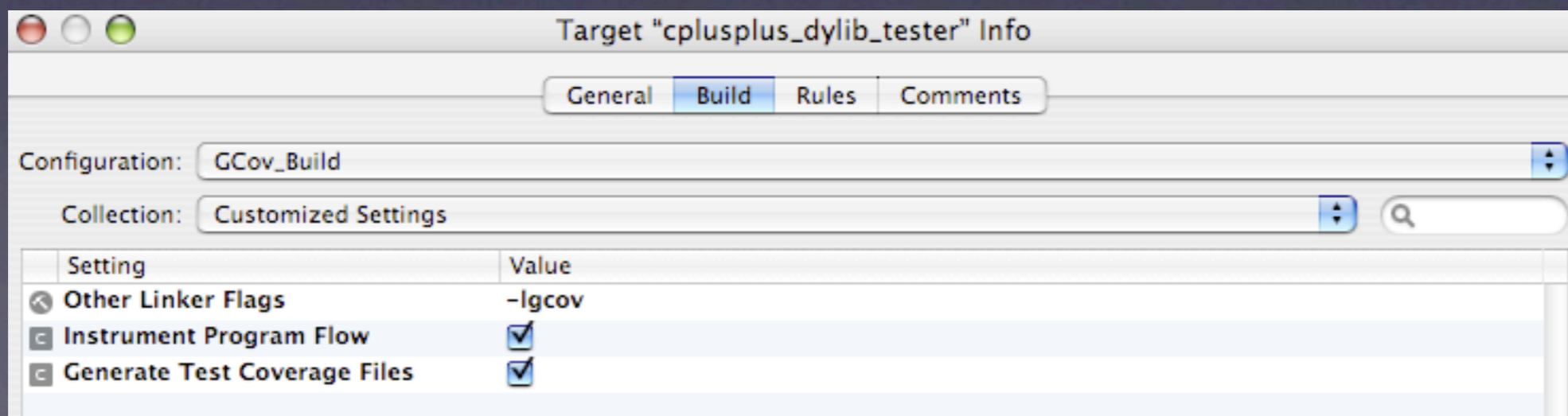
- CI Tool written in Java which runs as a servlet container
- Easy installation and configuration
- A wealth of plugins available
- Released under MIT license, Hudson is Free

Hudson

- We manage to do:
 - Automated Builds
 - Unit Testing
 - Profiling
 - Code Coverage
 - Static Analysis

Code Coverage Setup

- Make a new build Configuration
- To configure the target to work with GCOV code coverage data
 - Check “Generate Test Coverage Files”
 - Check “Instrument Program Flow”
 - Add “-lgcov” to “Other Linker Flags”



Static Analysis 'scan-build'

- Command line utility - also integrated in Xcode
- It runs the static analyzer over the code
- Results are published as html files

Setup

- Download the latest build (hudson-ci.org)
- Run it Standalone or with Tomcat, Glashfish...
- Enable Plugins:
 - Cobertura (for code coverage)
 - DocLinks (to publish results)
 - Chuck Norris Plugin



Setup new Job

- Create a new Job
- Check This build is parameterized:
 - Name: TARGET - Choices: LocalSearch3
- Set up subversion repository
- Poll SCM
 - Every 15 mins: */15 * * * *
- Setup Build
 - Execute build script with xcodebuild

Code Coverage Setup

- Post Build Actions
- Publish Cobertura Coverage Report
 - xml pattern as `**/html/coverage.xml`
 - Check consider stable builds
- Publish Documents:
 - Point to our symbolic link `trunk/html/static`

Run your build!!!

The screenshot shows the Hudson web interface for a build. The top navigation bar includes the 'Hudson' logo, a search box, and a help icon. The breadcrumb trail is 'Hudson > LocalSearch > #11'. On the right, there is an 'ENABLE AUTO REFRESH' link and a 'Delete this build' button. The main content area displays 'Build #11 (Aug 8, 2010)' with a blue sphere icon and a timestamp of '4:29:26 PM'. Below this, it shows 'Revision: 70943' with a 'No changes.' note, and 'Started by user anonymous' with a diamond icon. A Chuck Norris meme is featured, stating 'Chuck Norris can read all encrypted data, because nothing can hide from Chuck Norris.' The left sidebar contains links for 'Back to Project', 'Status', 'Changes', 'Console Output [raw]', 'Parameters', 'Tag this build', 'Coverage Report', and 'Previous Build'. At the bottom right, there is a large image of Chuck Norris giving a thumbs up, with a red banner below it that says 'Chuck Norris'.

Hudson

Hudson > LocalSearch > #11

ENABLE AUTO REFRESH

Delete this build

Started 23 min ago
Took 31 sec

add description

Build #11 (Aug 8, 2010)
4:29:26 PM

Revision: 70943
No changes.

Started by user [anonymous](#)

Chuck Norris can read all encrypted data, because nothing can hide from Chuck Norris.

Chuck Norris

Useful Links

- Mac Continuous Integration - August 2009
http://osx-ci.blogspot.com/2009_08_01_archive.html
- Unit Test and Code Coverage with Xcode
<http://www.supermegaultragroovy.com/blog/2005/11/03/unit-testing-and-code-coverage-with-xcode/>
- Using GCOV from Xcode
<http://developer.apple.com/mac/library/qa/qa2007/qa1514.html>
- Clang Static Analyzer
<http://clang-analyzer.llvm.org/scan-build.html>