Testing, Testing, iOS

Heath Borders

Senior Software Engineer
Asynchrony Solutions, Inc.
@heathborders
@asynchrony

https://github.com/StlMobileDev/iPhone-Testing

Road Map

- Why test?
- OCUnit
- Google Toolbox for Mac (GTM)
- GHUnit
- UISpec
- UIAutomation
- OCMock
- Final Recommendations

Test Automatically

Unit Test

Integration Test

Interaction Test

Mocking

Evaluation Criteria

- Ease of setup, writing, and running tests
- Error reporting
- Debugging
- Integration with Xcode
- Integration with the command line (interfacing with continuous integration systems)

Check out our example app!

(demo)

OCUnit - Setup

- http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/iphone_development/135-Unit_Testing_Applications/unit_testing_applications.html
- Sorry for the small font, you'll copy-paste it anyway
- Need different targets and platforms for Unit tests and Integration tests

OCUnit - Writing

- Extend SenTestCase
- -(void) testMyTestNameGoesHere { }
- STAssertEquals
- STAssertEqualsWithAccuracy
- STAssertEqualObjects

OCUnit - Running

- Unit tests only run on the iOS simulator
  - Use the 'Test' scheme to run
  - Run individual tests by editing the 'Test' scheme
- Integration tests only run on your iOS device
  - Use the 'Run' scheme to run
  - Can't run individual tests

OCUnit – Error Reporting

- Errors include line numbers and failure messages

- For JUnit-style output:
  - https://github.com/hborders/BPOCUnitXMLReporter

OCUnit – Debugging

- Add a breakpoint and re-run.
- Really, this is nice. It didn't used to be this easy.

OCUnit – Xcode Integration

- Unit test failures show up in the issue navigator and in the editor
- Integration tests have no integration, success/failure only shows in console

OCUnit – Command Line

- For unit tests, run test scheme with xcodebuild
  - xcodebuild -workspace /absolute/path/to/workspace -scheme MyUnitTestScheme -sdk iphonesimulator
- Integration tests only run on an iOS device, so they can't be triggered from the command line

GTM - Setup

- http://code.google.com/p/google-toolbox-for-mac/wiki/iPhoneUnitTesting
- Need a separate Test app target
- Tests essentially run from the command line

GTM - Writing

- Extend GTMTestCase
- -(void) testMyTestNameGoesHere { }
- Assertions are compatible with OCUnit!
- STAssertEqualStrings

GTM - Running

- Tests run in both the simulator and on device
- Can't choose which tests to run

GTM – Error Reporting

- Errors include line numbers and failure messages
- Error report is equivalent to OCUnit, doesn't have hooks for configuring junit-style reports

GTM – Debugging

- Disable the 'Run Script' build phase
  - If you don't, your build will fail and you won't be able to debug!
- Add a breakpoint
- Build and Run your test app.

GTM – Xcode Integration

- Unit test failures show up in the issue navigator and in the editor
- Integration tests have no integration, success/failure only shows in console

GTM – Command Line

- Build your test app using xcodebuild.
  - The 'Run Script' phase runs the tests
- iOS device tests can't be triggered from the command line

GHUnit - Setup

- http://gabriel.github.com/gh-unit/docs/appledoc_include/guide_install_ios_4.html
  - Use GHUnitIOSAppDelegate
  - http://stackoverflow.com/q/6738822/9636
- Need a separate Test app target
- Tests run in a separate app

GHUnit - Writing

- Extend GHTestCase
- -(void) testMyTestNameGoesHere { }

- Assertions are similar to GTM, but are prefixed with GH (GHAssertTrue...)

GHUnit - Running

- Tests run in both the simulator and on device
- Tests run inside test app
  - Can't use your UIApplicationDelegate as

    -[UIApplication sharedDelegate]
- You can choose which tests to run!
- Doesn't properly catch unexpected Exceptions

GHUnit – Error Reporting

- Errors include line numbers, failure messages, and stack traces

- Error report can output in junit format
  - WRITE_JUNIT_XML=YES

GHUnit – Debugging

- Add a breakpoint
- Build and Run your test app.

GHUnit – Xcode Integration

- None

GHUnit – Command Line

- http://gabriel.github.com/gh-unit/docs/appledoc_include/guide_command_line.html
  - A 'Run Script' phase runs the tests when an environment variable is present

- iOS device tests can't be triggered from the command line

UISpec - Setup

- http://code.google.com/p/uispec/wiki/Installation
- Need a duplicate app target to host your tests

UISpec - Writing

- http://code.google.com/p/uispec/wiki/Documentation
- Implement UISpec protocol
- -(void) itMyExampleGoesHere { }
  - UISpec calls test files “specs” and individual tests “examples”

UISpec - UIQuery

- http://code.google.com/p/uispec/source/browse/trunk/headers/UIQuery.h
- Returns UIQuery objects that pass through to UIView
- app.button // the first button
- app.button.all // collection of all the buttons
- [app.button index:2] // 3rd button
- [app.button.with currentTitle:@”foo”] // “foo” button
  - currentTitle: is a dynamic selector, this will lead to many warnings in your target
  - Each argument is anded to the filter
- Interactions: touch, show, flash

UISpec - Expectations

- expectThat() macro
- should:, not:, have:, be:
  - Call with expected values, child UIViews, or selectors holding expected states

UISpec - Running

- Duplicate your main.m
- Run all or some of your specs with
  - +(void)runSpecsAfterDelay:(int)seconds;
  - +(void)runSpec:(NSString *)specName afterDelay:(int)seconds;
  - +(void)runSpec:(NSString *)specName example:(NSString *)exampleName afterDelay:(int)seconds;
- Choose specs at runtime with
  - +(NSDictionary *)specsAndExamples;
  - Returns a dictionary of list of example names keyed by spec name.

UISpec – Error Reporting

- Errors include line numbers, expected and actual values
- Can customize logging with
  - +[UISpec setLog:]

UISpec – Debugging

- Add a breakpoint
- Run your app
- Beware of timeouts! Adjust with:
  - [[UIQuery withApplication] setTimeout:5000] //seconds
  - Give yourself enough time to debug

UISpec – Xcode Integration

- None

UISpec – Command Line

- Build the simulator tests with xcodebuild
- Run the simulator from the command line iphonesim
  - https://github.com/hborders/iphonesim
- iOS device tests can't be triggered from the command line

UIAutomation - Setup

- http://developer.apple.com/library/ios/#documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/Built-InInstruments/Built-InInstruments.html#//apple_ref/doc/uid/TP40004652-CH6-SW76

- The default accessibility options in the simulator don't work properly with UIAutomation
  - http://stackoverflow.com/questions/1388179
  - You don't need to do this when testing on device.
- Build your application
- Open Instruments, choose Automation template
- Click “Choose Target” and select your app inside ~/Library/Application Support/iPhone Simulator/<version>/

UIAutomation - Writing

- http://developer.apple.com/library/ios/#documentation/DeveloperTools/Reference/UIAutomationRef/_index.html
- Tests are written in javascript in a single test file
  - #import “path/relative/to/test.js”
- No assertions
  - Manually log a test starting with UIALogger.logStart
  - Manually log pass/fail with UIALogger.logPass or UIALogger.logFail
- https://github.com/alexvollmer/tuneup_js
- Cool hardware simulations

UIAutomation - UIAccessibility

- http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/iPhoneAccessibility/Making_Application_Accessible/Making_Application_Accessible.html#//apple_ref/doc/uid/TP40008785-CH102-SW5

- Queries for views rely primarily on accessibility labels
  - Mostly query for an exact match:
    - window.buttons()[“Reset”]
  - Can get more advanced with UIAElement.withPredicate

UIAutomation - Running

- Runs only in Instruments
- Can't choose which tests to run

UIAutomation – Error Reporting

- Errors only include data you pass to UIALogger.logFail

UIAutomation – Debugging

- Capture screenshot or arbitrary rectangles
- UIALogger log with different severities
- UIAElement.logElement UIAElement.logElementTree
- Can export results as plist from Instruments

UIAutomation – Xcode Integration

- None
- Can integrate with other Instruments
- Use Interaction tests to drive performance metrics

UIAutomation – Command Line

- None
- However, it should theoretically be possible to write an Automator script to screen-scrape Instruments
  - Exercise to the reader (please post on github)

OCMock

- Easy way to subclass Classes and implement Protocols at runtime
  - Can add custom behavior and returns
- Use niceMocks
- Declare your mock objects as id

Final Recommendations

- Use GHUnit for unit tests
- Use UIAutomation for Interaction tests
  - Other options:
    - https://github.com/square/KIF
      - I just couldn't fit this in. Sorry!
    - https://github.com/moredip/Frank
      - Based on UISpec, but drives it with Cucumber scripts

Chart

	GDB Debuggable	JUnit Output	Simulator Command-line Support	XCode Integration
OCUnit	Yes	Yes	Yes	Issues, Line Highlights
GTM	Yes	No (Open Source)	Yes	Issues, Line Highlights
GHUnit	Yes	Yes	Yes	None
UISpec	Yes	No (Open Source)	Yes	None
UIAutomation	No	No	No	None