Test Driven Development in the iOS World Part 2

Doug Sjoquist
http://www.sunetos.com
@dwsjoquist

What Is It?

Using automated software tests to drive the design and development of an application iteratively by writing your tests **BEFORE** you write your code

Why?

It greatly enhances my ability to do professional level work

TDD Process

"Just Enough" Design

Development Iterations

Review & Refactoring

Iterations

Keep each iteration short

Shorter cycles

==

Quicker feedback

Thinking in TDD fashion

Just In Time Design

Error Handling

- Special return codes
- Exceptions with try / catch
- NSError

```
// TestGameBoard.m

- (void) testMoveOutsideRange {
    NSError *error = nil;
    [gameBoard_ movePlayer:@"playerA"
                       row:-1
                       col:-1
                     error:&error];
    GHAssertNotNil(error,
        @"Expected an error");

    GHAssertEquals([error code],
        (NSInteger) kError_invalidBoardPosition,
        @"Expected kError_invalidBoardPosition");
}
```

{cc014,cc015}

More GameBoard Tests

- cc016, c017 - testMoveOutsideRange
- cc018, cc019 - testInvalidMove
- cc020, cc021 - testOnlyTwoPlayersAllowed
- cc022, cc023 - testAlternatePlayers

Tests for Game End StateConditions in GameBoard

```
//  PreconfiguredGameBoard.h
//  TDD_TicTacToe
//
//  Created by Douglas Sjoquist on 8/13/11.
//  Copyright 2011 Ivy Gulch, LLC. All rights reserved.
//

#import <Foundation/Foundation.h>
#import "GameBoard.h"

@interface PreconfiguredGameBoard : GameBoard {
}

- (void) setInternalBoard:(NSString*[3][3]) newBoard;
- (void) displayBoard;
```

{cc024}

```
- (void) setInternalBoard:(NSString*[3][3]) newBoard {
    for (int row=0; row<3; row++) {
        for (int col=0; col<3; col++) {
            board_[row][col] = newBoard[row][col];
        }
    }
}

- (void) displayBoard {
    NSLog(@"board");
    for (int row=0; row<2; row++) {
        NSLog(@"%@, %@, %@",
              board_[row][0],
              board_[row][1],
              board_[row][2]);
    }
}

@end
```

{cc025}

```
@implementation TestGameBoardEndStates
- (void) setUp {
    [super setUp];
    gameBoard_ = [[PreconfiguredGameBoard alloc] init];
}

- (void) tearDown {
    [gameBoard_ release];
    [super tearDown];
}
```

```
- (void) testFirstRowIsWinner {
    [gameBoard_ setInternalBoard:(NSString *[3][3]) {
        {@"X", @"X", @"X"},
        {nil, nil, nil},
        {nil, nil, nil}
    }];
    GHAssertEqualStrings([gameBoard_ winner], @"X", @"X
should win this game");
}
```

{cc026}

```
// GameBoard.m

- (NSString *) winner {
    NSString *currentValue = board_[0][0];
    if (currentValue
        && [currentValue isEqualToString:board_[0][1]]
        && [currentValue isEqualToString:board_[0][2]]){

        return currentValue;
    }

    return nil;
}
```

{cc027}

More GameBoard Tests

- cc028, cc029 - testSecondRowIsWinner
- cc030, cc031 - testFirstCollsWinner
- cc032 cc033 - testSecondCollsWinner

Moving through the TDD cycle quickly becomes addicting and encouraging

Time for some refactoring

```
// TestGameBoardEndStates.m

...

- (void) testFirstRowIsWinner {
    [gameBoard_ setInternalBoard:(NSString *[3][3]) {
        {@"X", @"X", @"X"},
        {nil, nil, nil},
        {nil, nil, nil}
    }];
    GHAssertEqualStrings([gameBoard_ winner],
                         @"X", @"X should win this game");

}

...
```

{cc034}

Remaining End State Tests

- cc035, cc036 - testDiagonalFrom00IsWinner
- cc037, cc038 - testDiagonalFrom02IsWinner
- cc039, cc040 - testDrawOnFullBoardWithNoWinner

GameBoard Behavior Is Covered By Tests, What's Next?

Tests to add:

GameViewController:
- test that a request to clear the view does indeed clear the view
- test that a request to display X or O in a specific location does so
- test that we can display a message
- test that we can disable user input
- test that we can enable user input
- test that user input generates a message to the delegate method when appropriate

Things to consider:
...

{cc042}

Develop Tests for GameViewController

Hidden assumptions

```
- (void) testOutletsLinked {
    GHAssertNotNil(gameViewController.messageLabel,
          @"messageLabel should be assigned");
    GHAssertNotNil(gameViewController.grid_0_0,
          @"grid_0_0 should be assigned");
...
    GHAssertNotNil(gameViewController.grid_2_2,
          @"grid_2_2 should be assigned");
}
```

{cc044}


Carrier
2:25 PM
Tests
testOutletsLinked
Re-run
TestGameViewController/testOutletsLinked ✘ 0.01s
Name: GHTestFailureException
File: /Users/dwsjoquist/ivygulch/presentations/CocoaConf/tddPart2/TDD_TicTacToe/TDD_TicTacToeTests/TestGameViewController.m
Line: 26
Reason: '((gameViewController_.messageLabel) != nil)' should be FALSE. messageLabel should be assigned
#0 0xe1f58c __exceptionPreprocess() (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation)
#1 0xf73313 objc_exception_throw() (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/usr/lib/libobjc.A.dylib)
#2 0xe1f4e1 +[NSException raise] (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation)
#3 0x00a114 +[TestGameViewController testOutletsLinked] (/Users/dwsjoquist/Library/Application Support/iPhone Simulator/4.3.2/Applications/8D13C02E-

Verify where and WHY your test failed

"Understanding the View Management Cycle section of Apple's View Controller Programming Guide for iOS"

```
// TestGameViewController.m

- (void) setUp {
    [super setUp];

    gameViewController_ = [[GameViewController alloc]
                           init];

// trigger loading of nib before every test
    gameViewController_.view;
}
```

{cc046}

More Tests for GameViewController

- cc047, cc048 - testClearView
- cc049, cc050, cc051 - testPlaceMarkInValidLocations
- cc052, cc053 - testPlaceMarkInInvalidLocations

GameViewController

Test that we can
display a message

Why?

- The cost of doing so is very low
- It makes our class more testable
- It hides implementation details
- It makes our interface cleaner

```
// TestGameViewController.m

- (void) testSetMessage {
    [gameViewController_ setMessage:@"New Message"];
    GHAssertEqualStrings(
          gameViewController_.messageLabel.text,
          @"New Message",
          @"messageLabel.text should be 'New Message'");
}
```

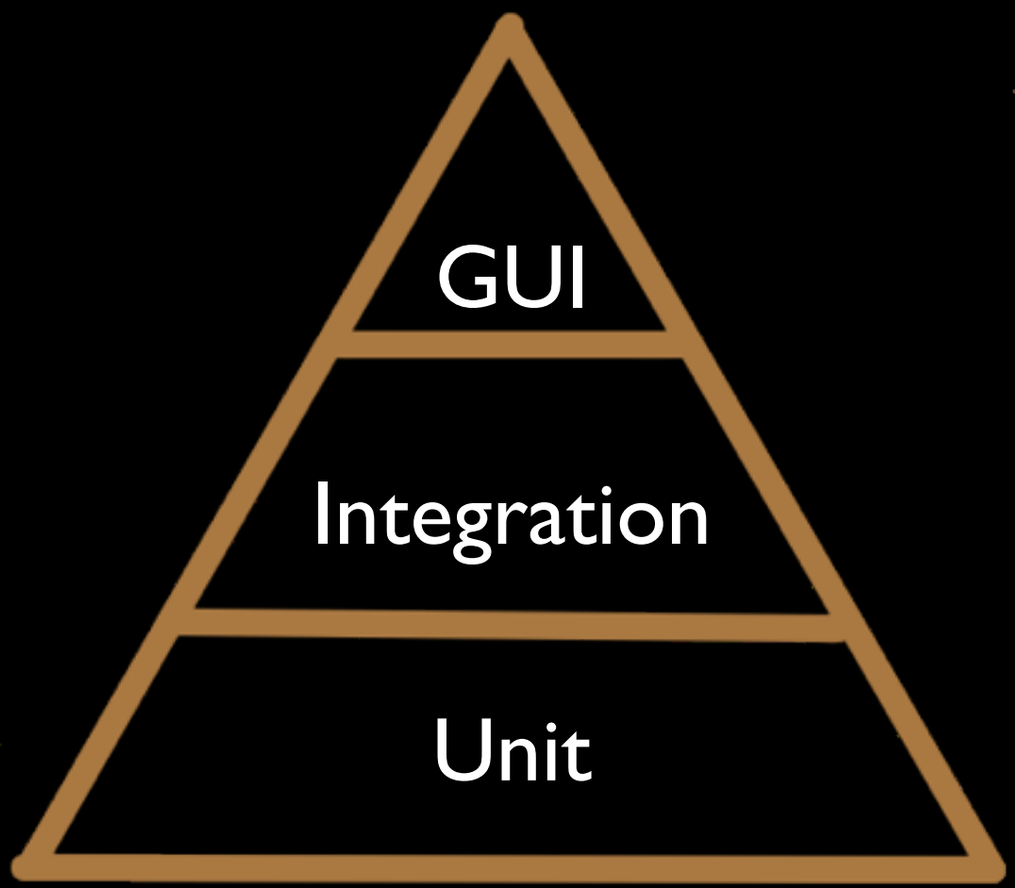
{cc054,cc055}

Testing components with dependencies

Using Delegate Pattern and OCMock to Test User Input

Continuum of Testing Types

Frequency
Ease of use
Immediacy



Brittleness
Complexity
Duration
Intrusiveness

Adding a protocol and delegate to GameViewController

```
// GameViewController.h

@protocol GameViewUserInput
- (void) userSelectedRow:(int) row col:(int) col;
@end

@interface GameViewController : UIViewController {
    id< GameViewUserInput> delegate_;
...

@property (nonatomic, assign)
        id<GameViewUserInput> delegate;
```

{cc056}

Using mock objects as stand-ins

We want to avoid using other “real” classes in our tests

Why avoid using other classes?

- It requires the other class to exist and and be at least partially functional
- It means working on multiple classes at the same time
- It increases complexity of our test setup
- It increases the brittleness of our tests

Benefits of using substitutes like OCMock?

- Reduces how much code we have to write
- Keeps us from worrying about whether the "real" class is behaving correctly,
- Focuses us on responsibilities of our class
- Reduces interdependencies of our tests

GameViewController

Test that each grid button is linked to an action

```
// TestGameViewController.h

- (void) testActionsLinked {
   NSArray *actions = [gameViewController.grid_0_0
         actionsForTarget:gameViewController
          forControlEvent:UIControlEventTouchUpInside];

   GHAssertEquals([actions count], (NSUInteger) 1,
       @"should have one action for 'Touch Up Inside'
for grid_0_0");
}
```

{cc057,cc058}

```
// TestGameViewController.h

- (void) checkActionsFor:(UIButton *) button {
    NSArray *actions = [button
          actionsForTarget:gameViewController
           forControlEvent:UIControlEventTouchUpInside];

    GHAssertEquals([actions count], (NSUInteger) 1,
        @"should have one action for 'Touch Up Inside'
for grid_0_0");
}

- (void) testActionsLinked {
     [self checkActionsFor:gameViewController.grid_0_0];
...
     [self checkActionsFor:gameViewController.grid_2_2];
}
```

{cc059,cc060}

GameViewController

Test that user input generates a message to the delegate method when appropriate

```
- (void) testGameViewUserInputDelegateMethodCalled {
    id mock = [OCMockObject
       mockForProtocol:@protocol(GameViewUserInput)];

// tell the mock to expect a method to be called
    [[mock expect] userSelectedRow:0 col:0];

// set the delegate on gameViewController_ so we
// can know if it actually calls the delegate's method
    gameViewController_.delegate = mock;

// trigger action for button grid_0_0 (row 0, col 0)
    [gameViewController_
        gridButtonAction:gameViewController_.grid_0_0];

// ask the mock if all of our expectations were met
    [mock verify];
}
```

{cc061}


Carrier
1:28 PM
Tests
testGameViewUs...
Re-run
TestGameViewController/testGameViewUserInputDelegateMethodCalled ✘ 0.01s
Name: NSInternalInconsistencyException
File: Unknown
Line: Unknown
Reason: OCMockObject[GameViewUserInput]: expected method was not invoked: userSelectedRow:0 col:0
#0 0xe1f58c __exceptionPreprocess() (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation)
#1 0xf73313 objc_exception_throw() (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/usr/lib/libobjc.A.dylib)
#2 0xdd7ef8 +[NSException raise:format:arguments:] (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation)
#3 0xdd7e6a +[NSException raise:format:] (/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator4.3.sdk/System/Library/Frameworks/CoreFoundation.framework/CoreFoundation)
#4 0x01927c +[OCMockObject verify]

```
// GameViewController.m

- (void) handleUserSelectedRow:(int) row col:(int) col {
    [self.delegate userSelectedRow:row col:col];
}

- (IBAction) gridButtonAction:(id) sender {
    if ([sender isEqual:self.grid_0_0]) {
        [self handleUserSelectedRow:0 col:0];
    } else if ([sender isEqual:self.grid_0_1]) {
        [self handleUserSelectedRow:0 col:1];
...
    } else if ([sender isEqual:self.grid_2_2]) {
        [self handleUserSelectedRow:2 col:2];
    }
}
```

{cc062}

```
- (void) checkUserInputForButton:(UIButton *) button
                       expectRow:(int) row
                             col:(int) col {
... same as old simpler method
}

- (void) testGameViewUserInputDelegateMethodCalled {
    [self
   checkUserInputForButton:gameViewController.grid_0_0
                 expectRow:0
                       col:0];
    [self
   checkUserInputForButton:gameViewController.grid_2_2
                 expectRow:2
                       col:2];
}
```

{cc063}

Lot’s more to do

TDD in iOS

Coming soon: TDD/iOS tutorial series on

http://www.raywenderlich.com

Doug Sjoquist
http://www.sunetos.com
@dwsjoquist