

QUALITY IS AS QUALITY DOES

Don Miller
Founder of GroundSpeed™
Co-founder of Seed Coworking
@donmiller



OBJECTIVES

- What is Unit Testing?
- Why Unit Testing?
- Xcode - Setting Up Unit Tests
- Xcode - Creating Performance Tests When Using A 3rd Party API
- Xcode - Testing parse.com Connection
- Xcode - Code Coverage



WHAT IS UNIT TESTING?

- Test small pieces of your code
- Xcode provides a separate testing environment that allows you to bolt on tests to your application
- Unit testing is a great way to test methods that you are planning to create
 - You can start by calling the method as you have planned
 - Once test fails, start writing the method to make it pass

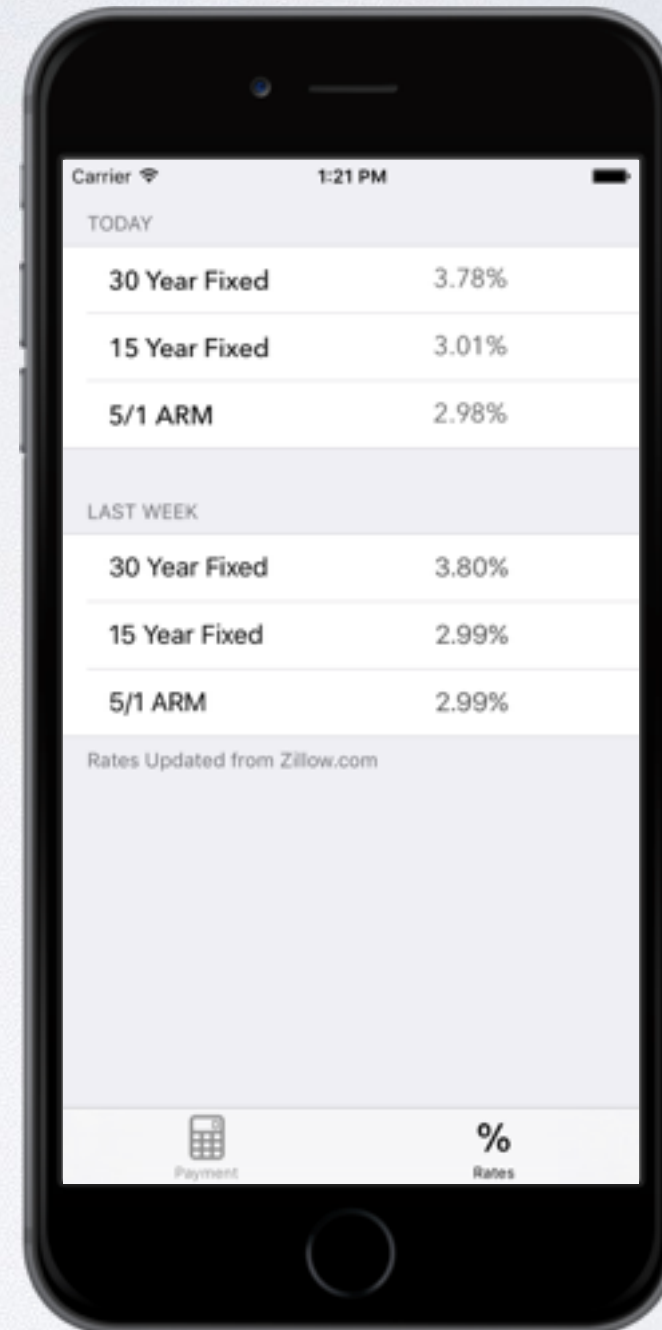
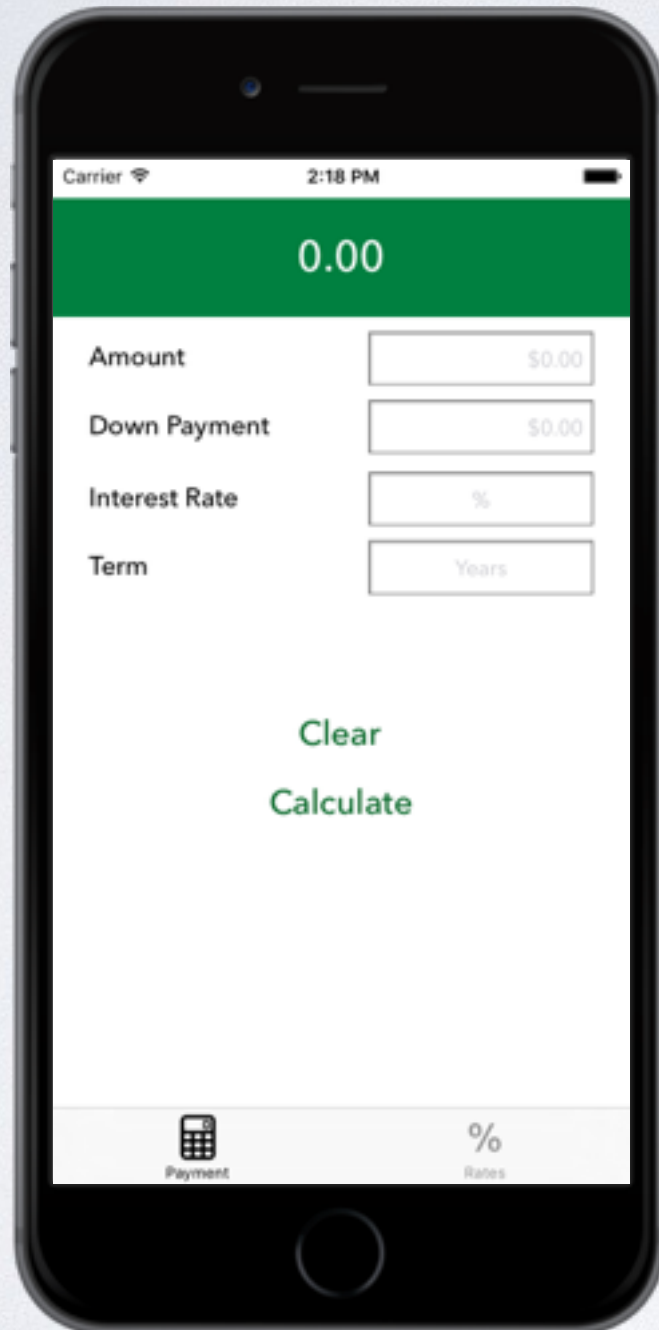


WHY UNIT TEST?

- Verify your code does what you expect
- Makes refactoring much less painful
- Creates a discipline of creating smaller more concise methods



APP WE WILL BE TESTING



HOW TO ADD UNIT TESTS?

Choose options for your new project:

Product Name:

Organization Name:

Organization Identifier:

Bundle Identifier:

Language:

Devices:

☐ Use Core Data

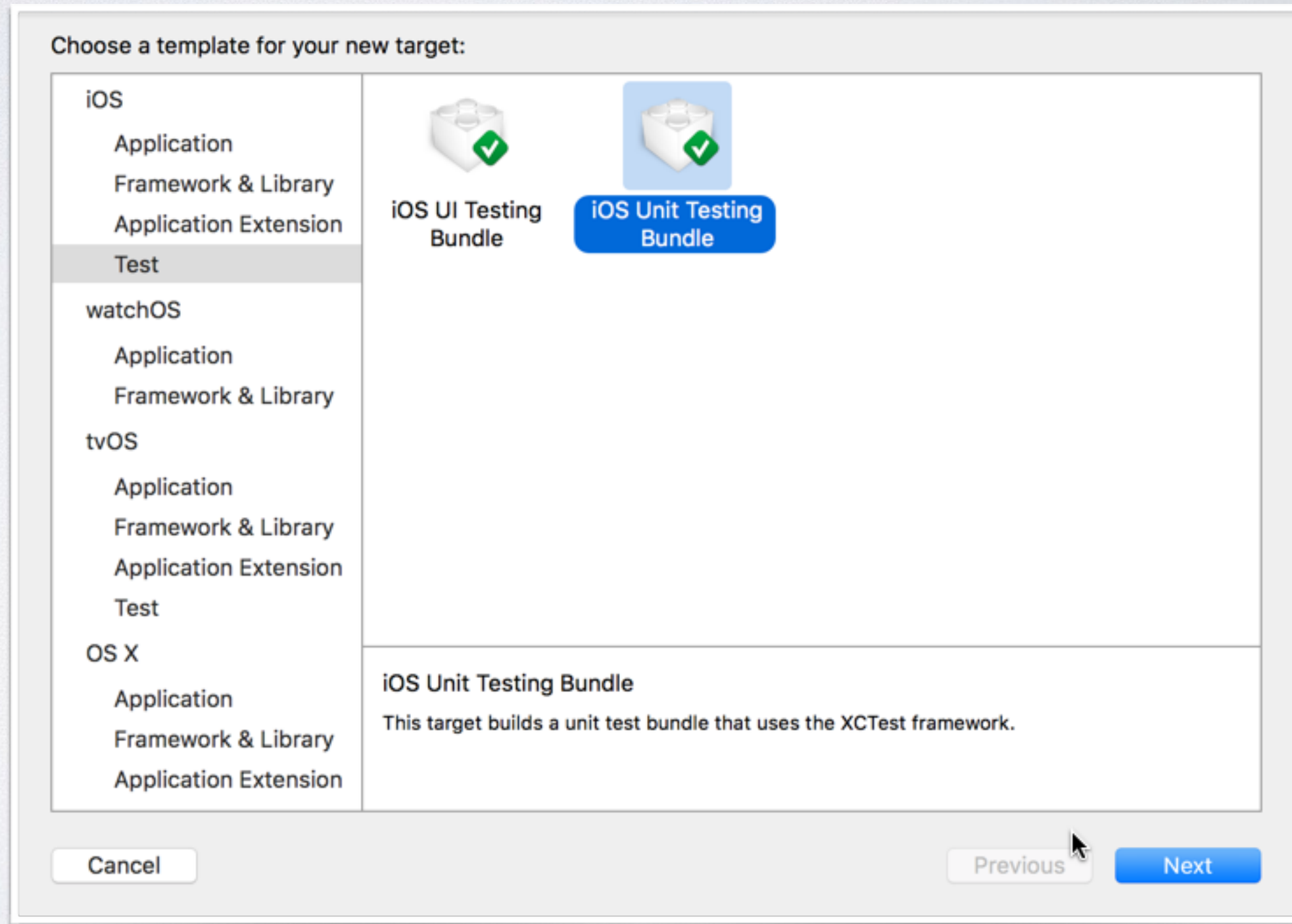
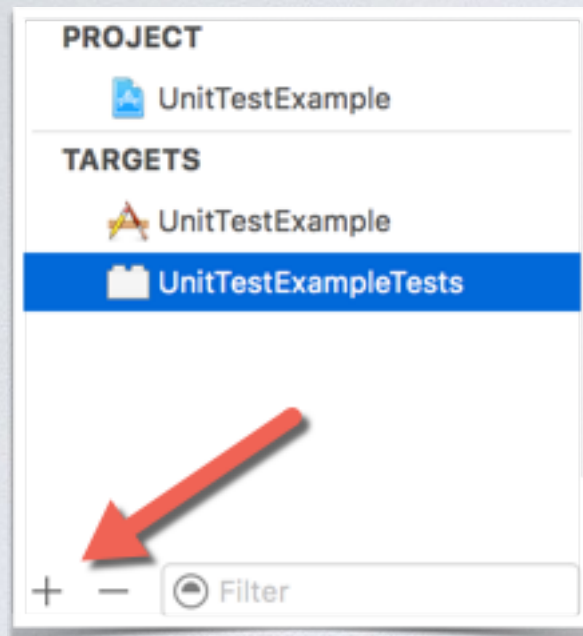
☒ Include Unit Tests

☐ Include UI Tests

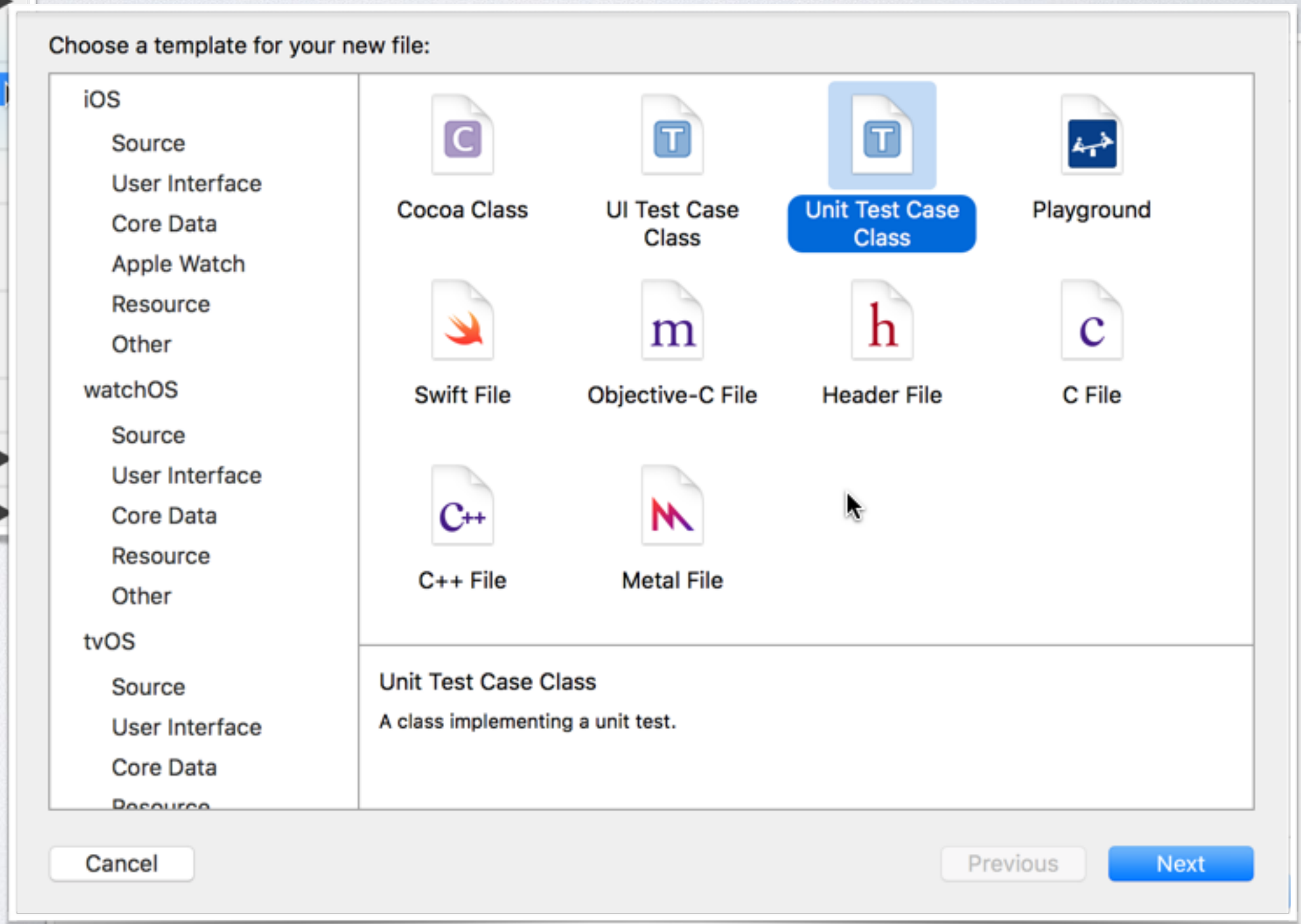
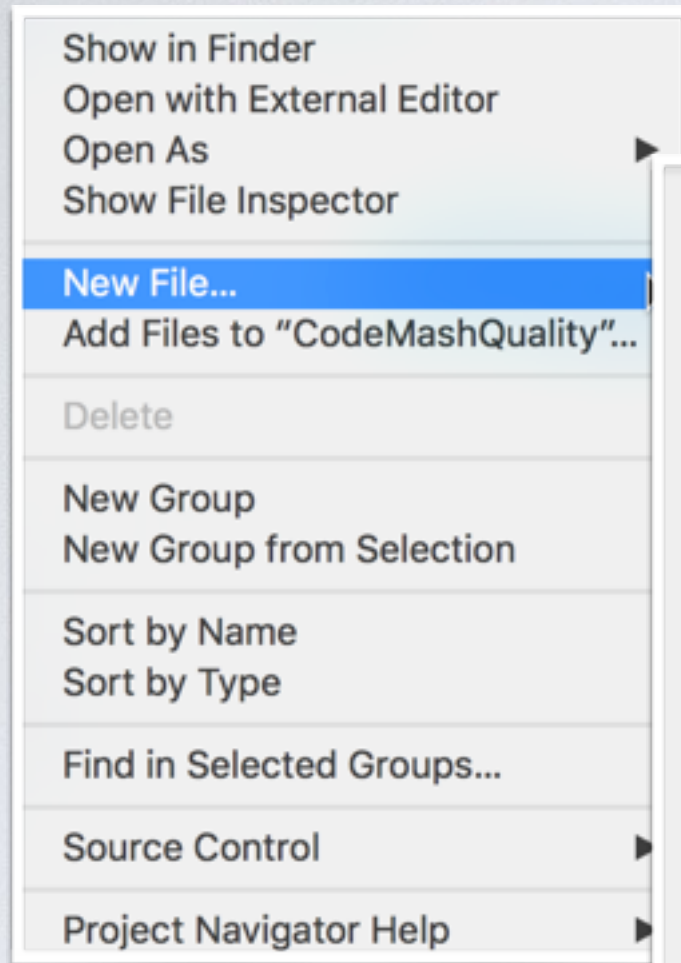
Make sure to check box

Cancel Previous Next

HOW TO ADD TESTS TO AN EXISTING PROJECT



ADDING NEW TEST CLASSES



UNIT TEST CONVENTIONS

- All Unit Test methods should begin with “test”
- Unit Test methods cannot have parameters
- Unit Test methods are subclasses of XCTestCase
- There are no restrictions on the amount of test classes or tests
- To run code prior to each test, override the setup method
- To run code after each test, override the teardown method



FUNDAMENTAL TEST

- All of the XCTest assertions come down to a single, base assertion

```
XCTAssert(expression, format...)
```

- If the expression evaluates to true, the test passes. Otherwise, the test fails, printing the formatted message.



BOOLEAN TESTS

- For Bool values, or simple boolean expressions, use XCTAssertTrue & XCTAssertFalse

```
[XCTAssertTrue(expression, format...)  
XCTAssertFalse(expression, format...)]
```


EQUALITY TESTS

- When testing whether two values are equal, use `XCTAssert[Not]Equal` for scalar values and `XCTAssert[Not]EqualObjects` for objects

```
XCTAssertEqual(expression1, expression2, format...)
XCTAssertNotEqual(expression1, expression2, format...)
```

```
func testOnePlusOneEqualsTwo() {
    XCTAssertEqual(1 + 1, 2, "one plus one should equal two")
}
```



NIL TESTS

- Use `XCTAssert[Not]Nil` to assert the existence (or non-existence) of a given value

```
XCTAssertNil(expression, format...)
XCTAssertNotNil(expression, format...)
```



UNCONDITIONAL FAILURE

- The XCTFail assertion will always fail

```
XCTFail(format...)
```



PERFORMANCE TESTING

```
func testDateFormatterPerformance() {  
    let dateFormatter = NSDateFormatter()  
    dateFormatter.dateFormat = .LongStyle  
    dateFormatter.timeStyle = .ShortStyle  
  
    let date = NSDate()  
  
    measureBlock() {  
        let string = dateFormatter.stringFromDate(date)  
    }  
}
```



XCTESTEXPECTATION

```
func testPerformanceZillowConnection() {
    self.measureBlock {
        self.callAsynchronousZillowConnection()
    }
}

func callAsynchronousZillowConnection() {
    let URL = NSURL(string: GlobalHelper().kPostEndpoint)!
    let expectation = expectationWithDescription("GET \(URL)")

    let session = NSURLSession.sharedSession()
    let task = session.dataTaskWithURL(URL) { data, response, error in
        XCTAssertNotNil(data, "data should not be nil")
        XCTAssertNil(error, "error should be nil")

        if let HTTPResponse = response as? NSHTTPURLResponse,
            responseURL = HTTPResponse.URL,
            MIMETYPE = HTTPResponse.MIMETYPE
        {
            XCTAssertEqual(responseURL.absoluteString, URL.absoluteString, "HTTP response URL should be equal to original URL")
            XCTAssertEqual(HTTPResponse.statusCode, 200, "HTTP response status code should be 200")
            XCTAssertEqual(MIMETYPE, "application/json", "HTTP response content type should be application/json")
        } else {
            XCTFail("Response was not NSHTTPURLResponse")
        }
    }

    expectation.fulfill()
    task.resume()
    waitForExpectationsWithTimeout(task.originalRequest!.timeoutInterval) { error in
        if let error = error {
            print("Error: \(error.localizedDescription)")
        }
        task.cancel()
    }
}
```



```

import UIKit

class GlobalHelper: NSObject {

    static let kApiKey = "X1-ZWz1f3t3bvl4i3_1brbp"
    let kPostEndpoint = "https://www.zillow.com/webservice/GetRateSummary.htm?zws-id=\(kApiKey)&output=json"

    func getMonthlyPayment(amount: Float, downPayment: Float, term: Float, interestRate: Float, lblPayment: UILabel) -> UILabel {
        // A = payment Amount per period
        // P = initial Principal (loan amount)
        // r = interest rate per period
        // n = total number of payments or periods

        let principal : Float = amount - downPayment
        let payments = term*12
        let rate = interestRate/12/100
        let amount = calculatPMTWithRatePerPeriod(rate, numberOfPayments: payments, loanAmount: principal, futureValue: 0, type: 0)

        if (isnan(amount) || isinf(amount))
        {
            lblPayment.font = UIFont.boldSystemFontOfSize(18)
            lblPayment.textColor = UIColor.redColor()
            lblPayment.text = "You must enter all required fields.";
        }
        else
        {
            lblPayment.font = UIFont(name: "Avenir Next", size: 28)
            lblPayment.textColor = UIColor.whiteColor()
            lblPayment.text = String(format: "%.02f", amount)
        }

        return lblPayment
    }

    func calculatPMTWithRatePerPeriod (ratePerPeriod: Float, numberOfPayments: Float, loanAmount: Float, futureValue: Float, type: Float) -> Float {
        var q : Float
        q = pow(1 + ratePerPeriod, numberOfPayments)
        let returnValue = (ratePerPeriod * (futureValue + (q * loanAmount))) / ((-1 + q) * (1 + ratePerPeriod * (type)))
        return returnValue
    }
}

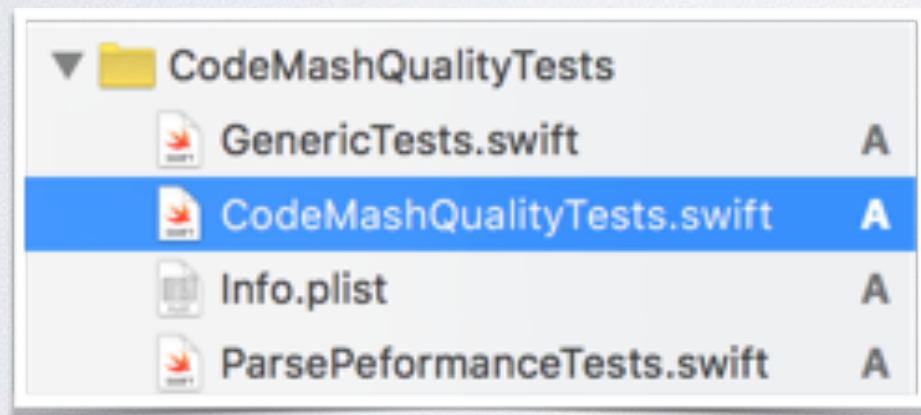
extension String {
    func toPercent() -> String {
        return String.localizedStringWithFormat("%.2f%%", Float(self)!)
    }
}

```



@TESTABLE

- @testable solves the problem of previous Swift versions by giving you access to everything that is internal
- Access levels were different in Objective-C and this problem was not noticed until Swift 1.0



```
9 import XCTest
10 @testable import CodeMashQuality
```



A red arrow points from the right side of the code block to the '@testable import' statement on line 10.



TEST OUR GETPAYMENT() METHOD

```
24 func testGetPayment() {
25     let amount: Float = 100000.00
26     let downPayment: Float = 1000.00
27     let interestRate: Float = 5.0
28     let term: Float = 30
29     let label : UILabel = UILabel()
30
31     let result = GlobalHelper().getMonthlyPayment(amount, downPayment: downPayment, term: term, interestRate:
        interestRate, lblPayment: label)
32
33     XCTAssertEqual(result.text == "531.45", "\(result.text) should equal 531.45")
34     XCTAssertEqual(result.textColor == UIColor.whiteColor(), "\(result.textColor) should be white")
35 }
36
37 func testGetPaymentBadNumber() {
38     let amount: Float = 0
39     let downPayment: Float = 0
40     let interestRate: Float = 0
41     let term: Float = 0
42     let label : UILabel = UILabel()
43
44     let result = GlobalHelper().getMonthlyPayment(amount, downPayment: downPayment, term: term, interestRate:
        interestRate, lblPayment: label)
45
46     XCTAssertEqual(result.text == "You must enter all required fields.", "\(result.text) should read You must enter all
        required fields.")
47     XCTAssertEqual(result.textColor == UIColor.redColor(), "\(result.textColor) should be red")
48 }
```



TEST CALCULATEPAYMENT() METHOD

```
51 func testCalculatePayment() {  
52     let amount: Float = 100000.00  
53     let downPayment: Float = 1000.00  
54     let interestRate: Float = 5.0  
55     let term: Float = 30  
56  
57     let principal : Float = amount - downPayment  
58     let payments = term*12  
59     let rate = interestRate/12/100  
60  
61     var result = GlobalHelper().calculatPMTWithRatePerPeriod(rate, numberOfPayments: payments, loanAmount: principal,  
62         futureValue: 0.0, type: 0.0)  
63     result = round(100*result)/100 //Round to two decimal places  
64     XCTAssert(result == 531.45, "\ (result) should equal 531.45")  
65 }
```



TEST THE ZILLOW API JSON CALL

```
67 func testJsonCall() {  
68     let zillowRates : Dictionary<String, String> = ApiHelper().getRatesFromZillow()  
69  
70     XCTAssert(zillowRates.count == 6, "\(zillowRates.count) should include 6 records")  
71 }
```



TEST THE RATES MODEL

```
73 func testRatesModel() {  
74     let rates = Rates()  
75  
76     rates.thirtyYearFixed = "2.00"  
77     rates.fifteenYearFixed = "1.00"  
78     rates.fiveOneARM = "1.50"  
79  
80     XCTAssertNotNil(rates.thirtyYearFixed)  
81     XCTAssertNotNil(rates.fifteenYearFixed)  
82     XCTAssertNotNil(rates.fiveOneARM)  
83 }  
84  
85 func testRatesModelForPercentSign() {  
86     let rates = Rates()  
87  
88     rates.thirtyYearFixed = "2.00".toPercent()  
89     rates.fifteenYearFixed = "1.00".toPercent()  
90     rates.fiveOneARM = "1.50".toPercent()  
91  
92     XCTAssertTrue(rates.thirtyYearFixed.characters.last! == "%", "\ (rates.thirtyYearFixed) should end with a percent")  
93     XCTAssertTrue(rates.fifteenYearFixed.characters.last! == "%", "\ (rates.thirtyYearFixed) should end with a percent")  
94     XCTAssertTrue(rates.fiveOneARM.characters.last! == "%", "\ (rates.thirtyYearFixed) should end with a percent")  
95 }
```



CALL ZILLOW TO VERIFY A 200 JSON RESPONSE

```
97 func testPerformanceZillowConnection() {
98     self.measureBlock {
99         self.callAsynchronousZillowConnection()
100     }
101 }
102
103 func callAsynchronousZillowConnection() {
104
105     let URL = NSURL(string: GlobalHelper().kPostEndpoint)!
106     let expectation = expectationWithDescription("GET \(URL)")
107
108     let session = NSURLSession.sharedSession()
109     let task = session.dataTaskWithURL(URL) { data, response, error in
110         XCTAssertNotNil(data, "data should not be nil")
111         XCTAssertNil(error, "error should be nil")
112
113         if let HTTPResponse = response as? NSHTTPURLResponse,
114             responseURL = HTTPResponse.URL,
115             MIMETYPE = HTTPResponse.MIMETYPE
116         {
117             XCTAssertEqual(responseURL.absoluteString, URL.absoluteString, "HTTP response URL should be equal to original URL")
118             XCTAssertEqual(HTTPResponse.statusCode, 200, "HTTP response status code should be 200")
119             XCTAssertEqual(MIMETYPE, "application/json", "HTTP response content type should be application/json")
120         } else {
121             XCTFail("Response was not NSHTTPURLResponse")
122         }
123
124         expectation.fulfill()
125     }
126     task.resume()
127     waitForExpectationsWithTimeout(task.originalRequest!.timeoutInterval) { error in
128         if let error = error {
129             print("Error: \(error.localizedDescription)")
130         }
131         task.cancel()
132     }
133 }
```



WORKING WITH PARSE

The screenshot displays the Parse dashboard interface. At the top, a navigation bar includes links for CodeMash, DEV, Core, Analytics, Push, Settings, and Docs, along with a user profile for Don Miller. Below this, a 'Data' section shows a table with columns for Role, User, objectId, foo, createdAt, updatedAt, and ACL. The table contains one row of data. To the right of the table is a code editor with Swift code for application setup. A box labeled 'setup in AppDelegate' points to the code. The bottom of the dashboard features a sidebar with 'Config' and 'API Console' options, and a footer with a 'Switch to the new Dashboard' button and links for Docs, Billing, Downloads, Help, Status, Blog, and Parse.com.

CodeMash DEV Core Analytics Push Settings Docs Don Miller

Data + Row - Row + Col Security More

Role	0	objectId String	foo String	createdAt Date	updatedAt Date	ACL ACL
User	1	rGcuDMxLAm	bar	Jan 08, 2016, 06:28	Jan 08, 2016, 06:28	Public Read and Write

```
19 func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
20     // Override point for customization after application launch.
21
22     Parse.enableLocalDatastore()
23
24     // Initialize Parse.
25     Parse.setApplicationId("yDEI", clientKey: "Gut", Pc", IJo")
26
27     // [Optional] Track statistics around application opens.
28     PFAnalytics.trackAppOpenedWithLaunchOptions(launchOptions)
29
30
31
32     return true
33 }
```

setup in AppDelegate

Config API Console

20 rows/page 1 - 10 of 10 rows

Switch to the new Dashboard Docs Billing Downloads Help Status Blog Parse.com



CREATE TEST TO CHECK FOR PARSE CORRECT SETUP

```
9 import XCTest
10 import Parse
11 import Bolts
12
13 class ParsePerformanceTests: XCTestCase {
14
15     func testParsePerformanceOnMainThread() {
16         // This is an example of a performance test case.
17         self.measureBlock {
18             self.checkParseIsSetup()
19         }
20     }
21
22     func checkParseIsSetup() {
23         let testObject = PFObject(className: "TestObject")
24         testObject["foo"] = "bar"
25
26         do { try testObject.save()
27             print("Object has been saved.")
28         }
29         catch let e as NSError { print("Parse save error: \(e)") }
30     }
31 }
```



TEST PARSE PERFORMANCE

```
44 func testParsePerformanceGettingProducts() {  
45     // This is an example of a performance test case.  
46     self.measureBlock {  
47         self.checkParsePullingData()  
48     }  
49 }  
50  
51 func checkParsePullingData() {  
52     let query = PFQuery(className:"Products")  
53  
54     do { let objects = try query.findObjects()  
55         for object in objects {  
56             print(object.objectId)  
57         }  
58     }  
59     catch let e as NSError { print("Parse load error: \(e)") }  
60 }
```



TEST LOADING OF PRODUCTS OBJECT FROM PARSE

```
12 class Products {
13
14     var objectId : String?
15     var productId : Int?
16     var photo : String?
17     var photoFile : PFFile = PFFile()
18     var photoImage : UIImage?
19     var title : String?
20     var flyer : String?
21     var productDesc : String?
22     var itemCode : String?
23     var packagingOptions : String?
24     var fromParse : String?
25 }
26
62 func testGettingRatesCollection() {
63     let arrayProducts : Array<Products> = Products().getAllProductsFromParse()
64
65     XCTAssert(arrayProducts.count > 0, "\(arrayProducts.count) should be greater than zero")
66 }
67
30     var masterCase : Int?
31
32     func getAllProductsFromParse() -> Array<Products> {
33         let query = PFQuery(className:"Products")
34         var arrayProducts : Array<Products> = []
35
36         do { let objects = try query.findObjects()
37             for object in objects {
38                 self.objectId = object["objectId"] as? String
39                 self.productId = object["Id"] as? Int
40                 self.photoImage = object["PhotoImage"] as? UIImage
41                 self.title = object["Title"] as? String
42                 self.productDesc = object["Description"] as? String
43
44                 arrayProducts.append(self)
45             }
46         }
47         catch let e as NSError { print("Parse load error: \(e)") }
48
49         return arrayProducts
50     }
```

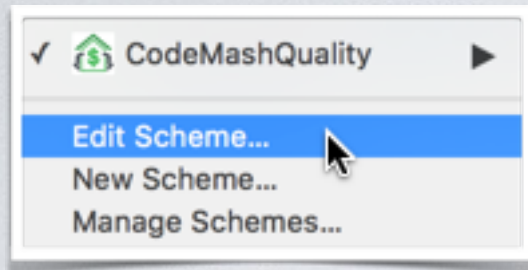


CODE COVERAGE

By GroupBy Time		TestsLogs	
<div>CodeMashQuality Today, 1:55 AM</div> <div>Test Today, 2:02 AM</div> <div>Build Today, 2:02 AM</div> <div>Test Today, 2:01 AM</div> <div>Build Today, 2:01 AM</div> <div>Test Today, 1:55 AM</div> <div>Build Today, 1:55 AM</div> <div>Debug Today, 12:58 AM</div> <div>Project No Logs</div>	AllPassedFailedAllPerformance		
	Tests	Status	Time
	CodeMashQualityTests > CodeMashQualityTests		
	testCalculatePayment()	✓	
	testGetPayment()	✓	
	testGetPaymentBadNumber()	✓	
	testJsonCall()	✓	
	testPerformanceZillowConnection()	✓	0.07 s
	testRatesModel()	✓	
	testRatesModelForPercentSign()	✓	
	GenericTests > CodeMashQualityTests		
	testDateFormatterPerformance()	✓	0.00 s
	testOnePlusOneEqualsTwo()	✓	
	ParsePerformanceTests > CodeMashQualityTests		
	testGettingRatesCollection()	✓	
	testParsePerformanceGettingProducts()	✓	0.08 s
	testParsePerformanceOnMainThread()	✓	0.04 s



CODE COVERAGE



		Tests	Coverage	Logs
Name		Coverage		
▼	CodeMashQuality.app	<div><div></div></div>		
▶	GlobalHelper.swift	<div><div></div></div>		
▶	ApiHelper.swift	<div><div></div></div>		
▶	Products.swift	<div><div></div></div>		
▶	AppDelegate.swift	<div><div></div></div>		
▶	PaymentViewController.swift	<div><div></div></div>		
▶	RatesTableViewController.swift	<div><div></div></div>		

+ -

Filter

Duplicate Scheme

Manage Schemes...

☐ Shared

Close

QUESTIONS? THANK YOU!

Don Miller

don@GroundSpeedHQ.com

@donmiller

http://github.com/donmiller

http://github.com/groundspeak

