https://github.com/GroundSpeed/BreakingGround

UNIT TESTING FOR THE EFFICIENT XCODE EXPERIENCE

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

Don Miller | GroundSpeed™
rapid web + mobile software

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

GroundSpeed™
rapid web + mobile software

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

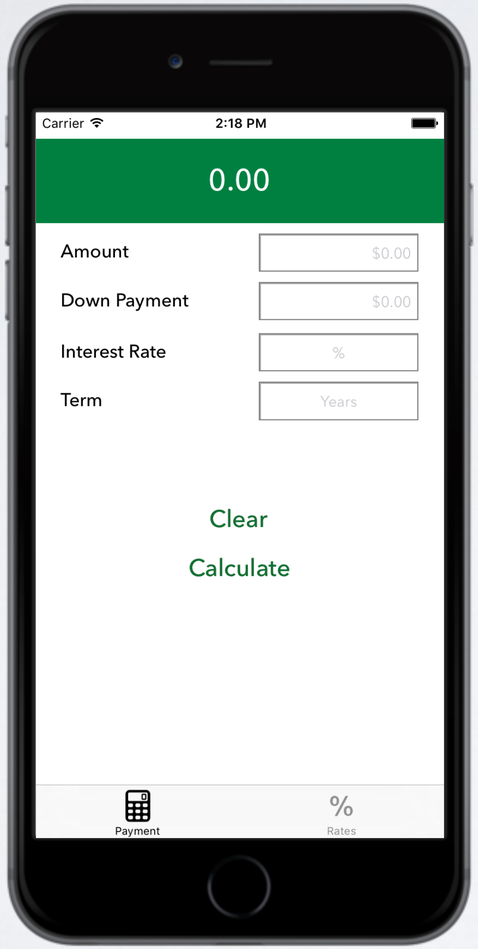
GroundSpeed™
rapid web + mobile software

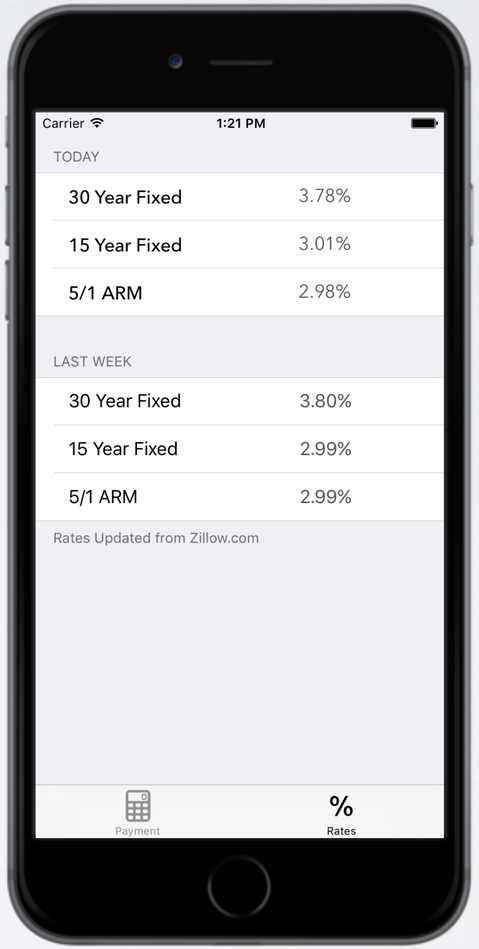
WHY UNIT TEST?

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

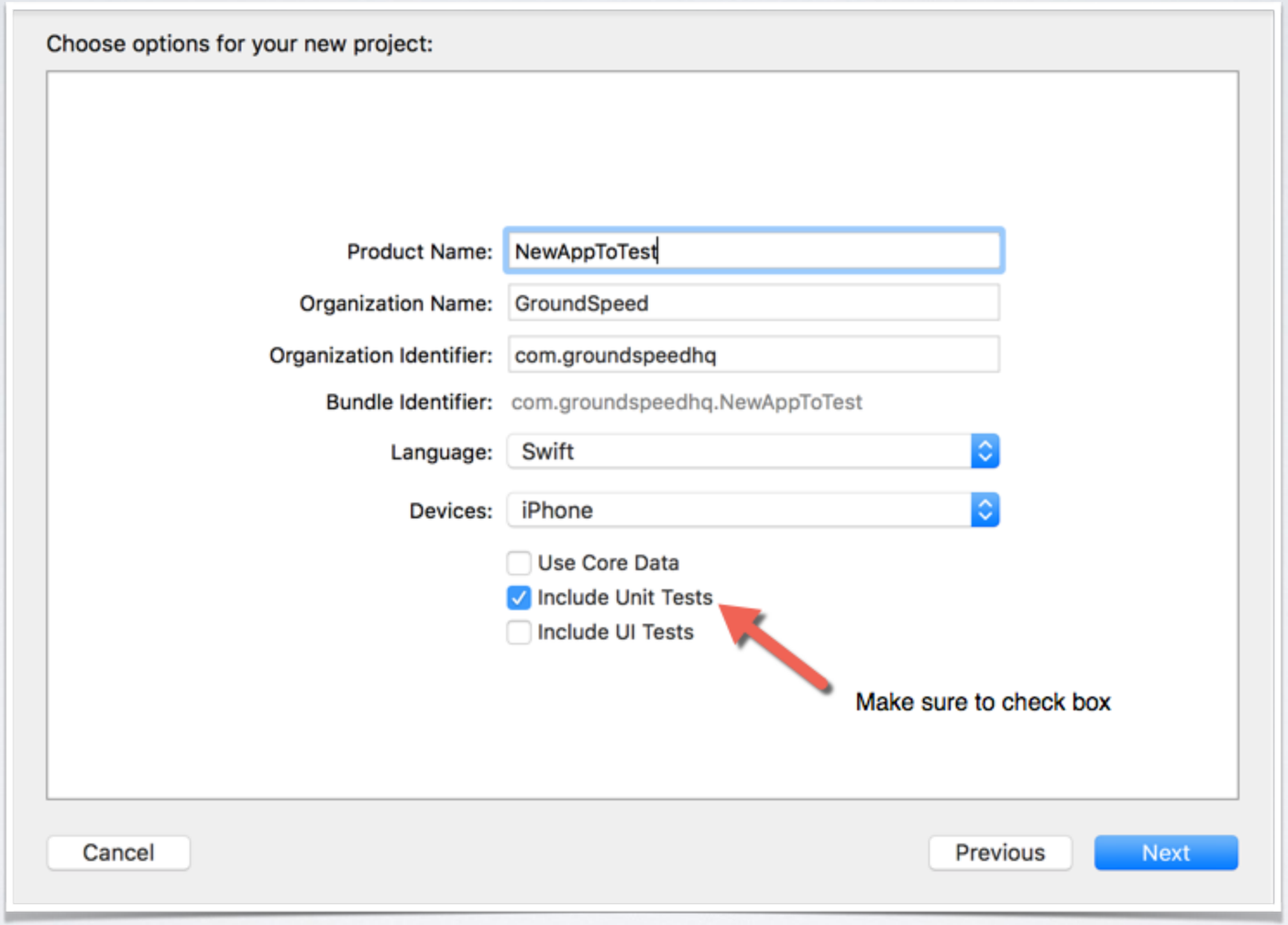
GroundSpeed™
rapid web + mobile software

APP WE WILL BE TESTING



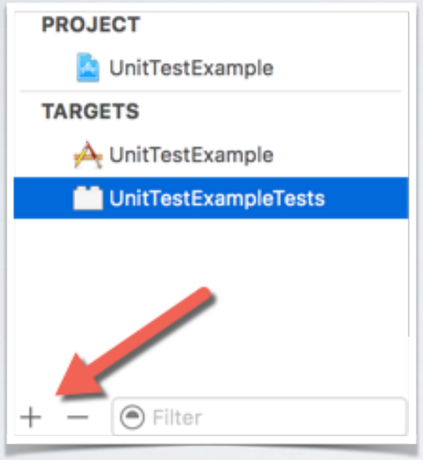


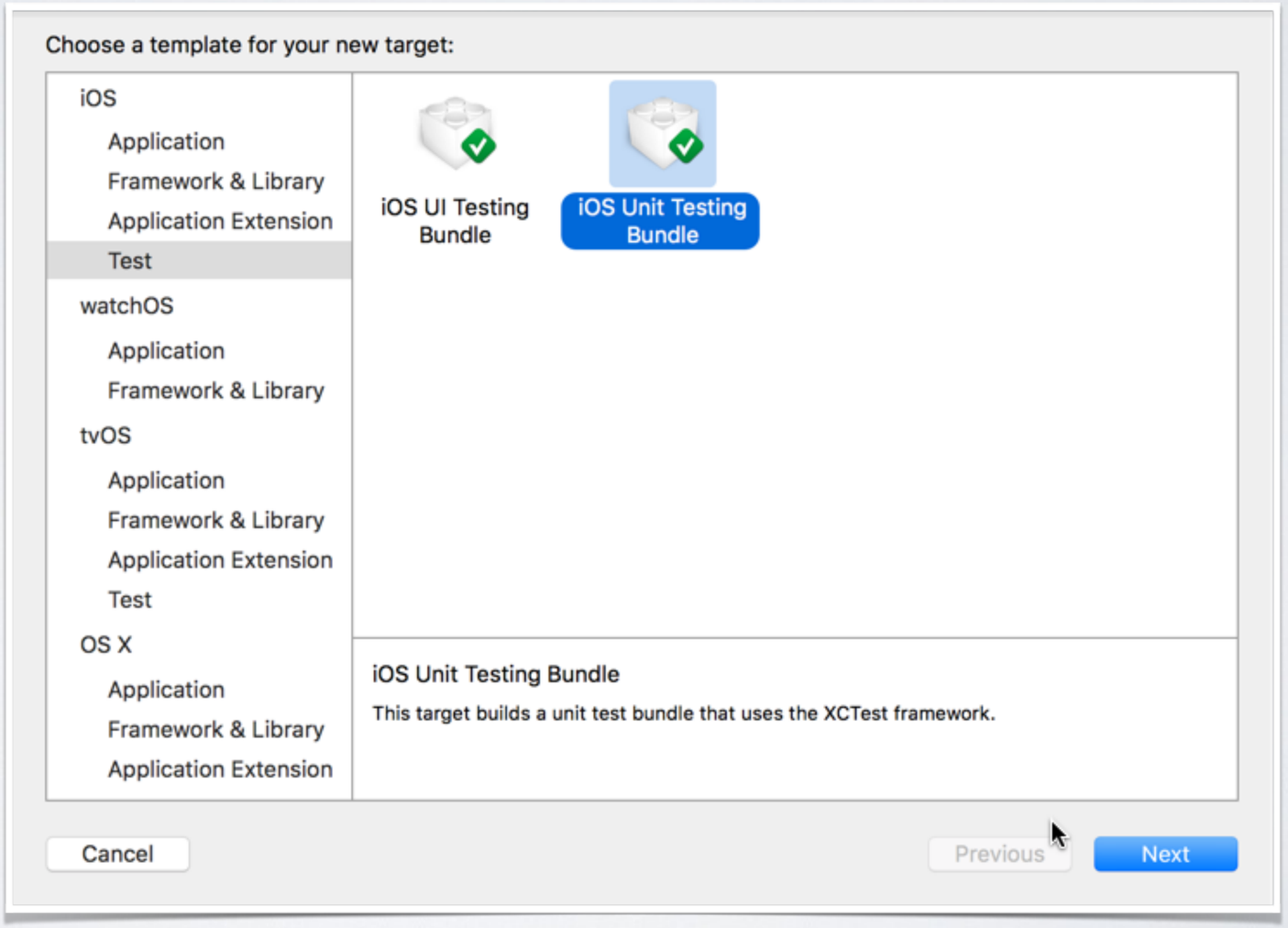
HOW TO ADD UNIT TESTS?



GroundSpeed™
rapid web + mobile software

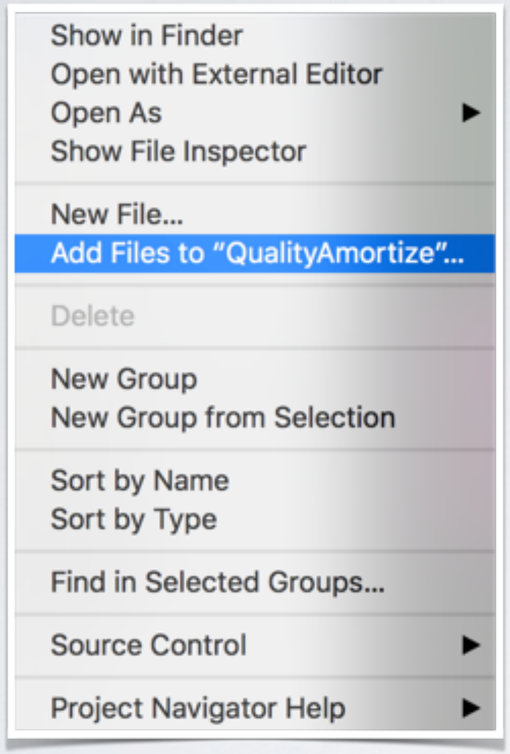
HOW TO ADD TESTS TO AN EXISTING PROJECT

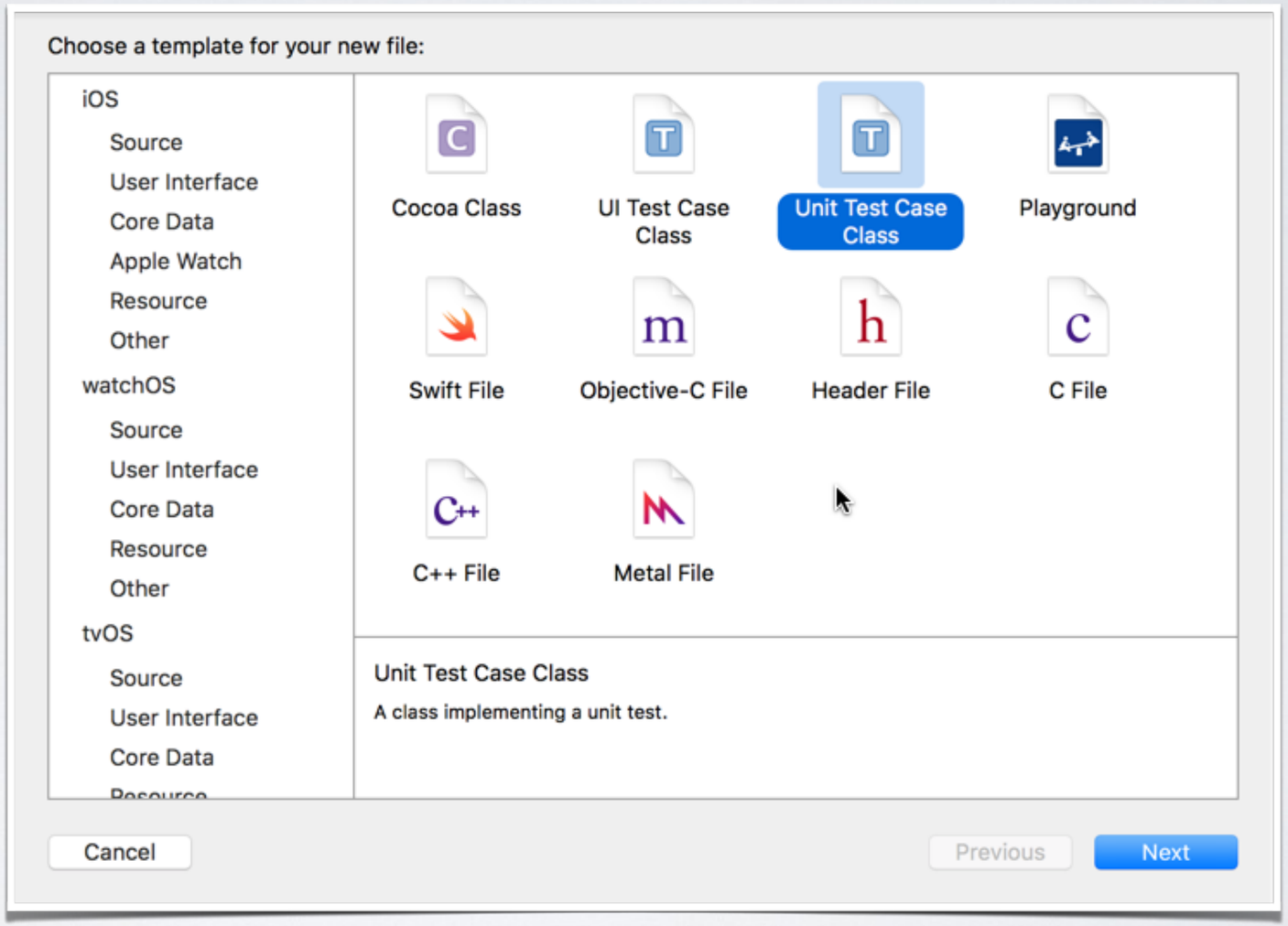




GroundSpeed™
rapid web + mobile software

ADDING NEW TEST CLASSES





GroundSpeed™
rapid web + mobile software

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.dateStyle = .LongStyle
    dateFormatter.timeStyle = .ShortStyle

    let date = NSDate()

    measureBlock() {
        let string = dateFormatter.stringFromDate(date)
    }
}
```

GroundSpeed™
rapid web + mobile software

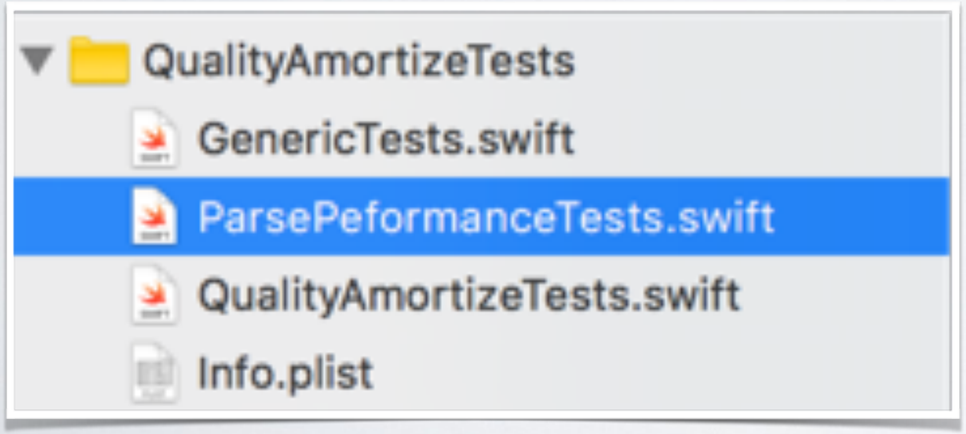
XCTESTEXPECTATION

```
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()
    }
}
```

```swift
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




```
import XCTest
@testable import QualityAmortize
```

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          XCTAssert(result.text == "531.45", "\(result.text) should equal 531.45")
34          XCTAssert(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          XCTAssert(result.text == "You must enter all required fields.", "\(result.text) should read You must enter all
                required fields.")
47          XCTAssert(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,
                futureValue: 0.0, type: 0.0)
62          result = round(100*result)/100  //Round to two decimal places
63          XCTAssert(result == 531.45, "\(result) should equal 531.45")
64
65      }
```

GroundSpeed™
rapid web + mobile software

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      }
```

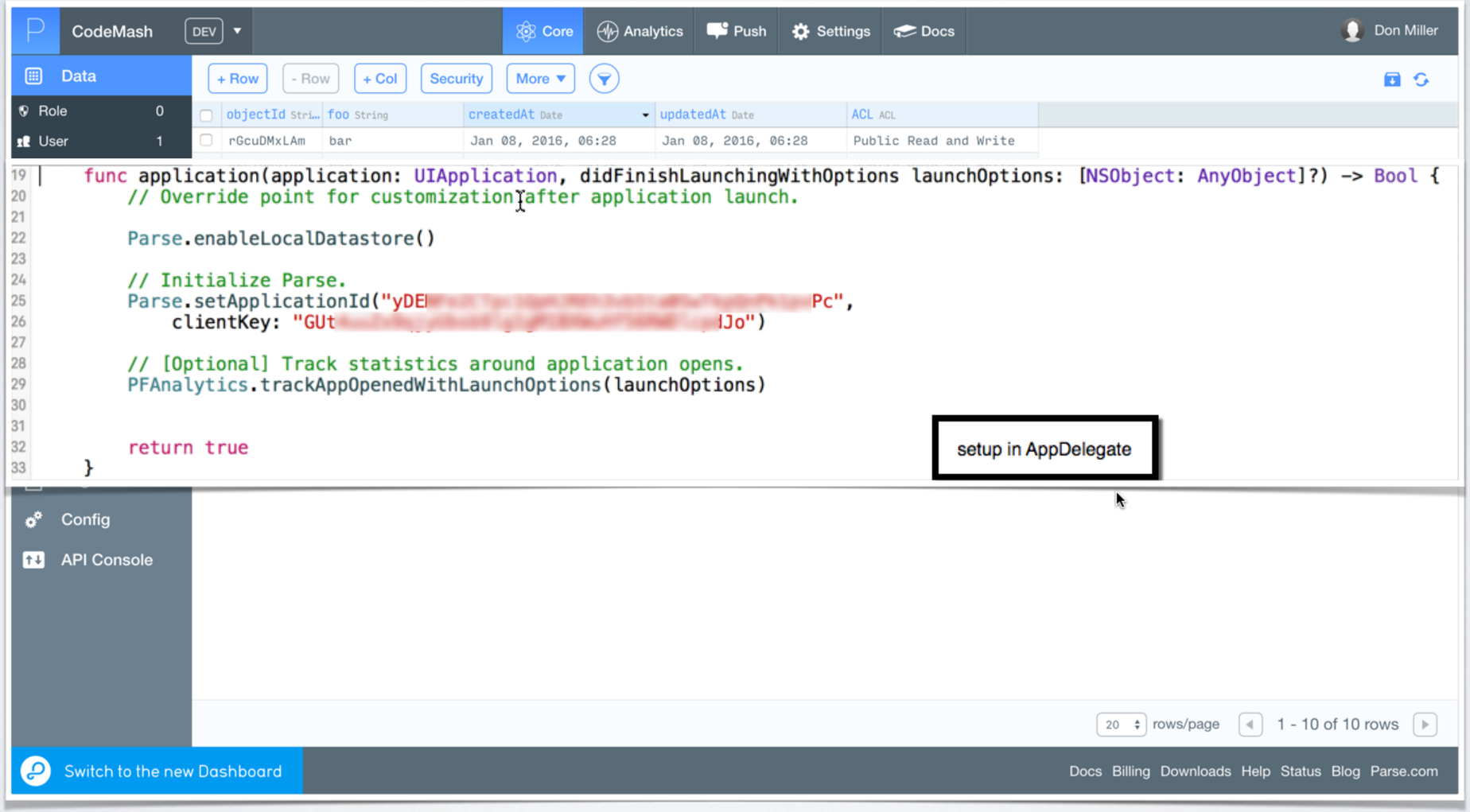
GroundSpeed™
rapid web + mobile software

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     }
134 }
```

GroundSpeed™
rapid web + mobile software

WORKING WITH PARSE



CREATE TEST TO CHECK FOR PARSE CORRECT SETUP

```
9  import XCTest
10 import Parse
11 import Bolts
12
13 class ParsePeformanceTests: 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     }
```

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

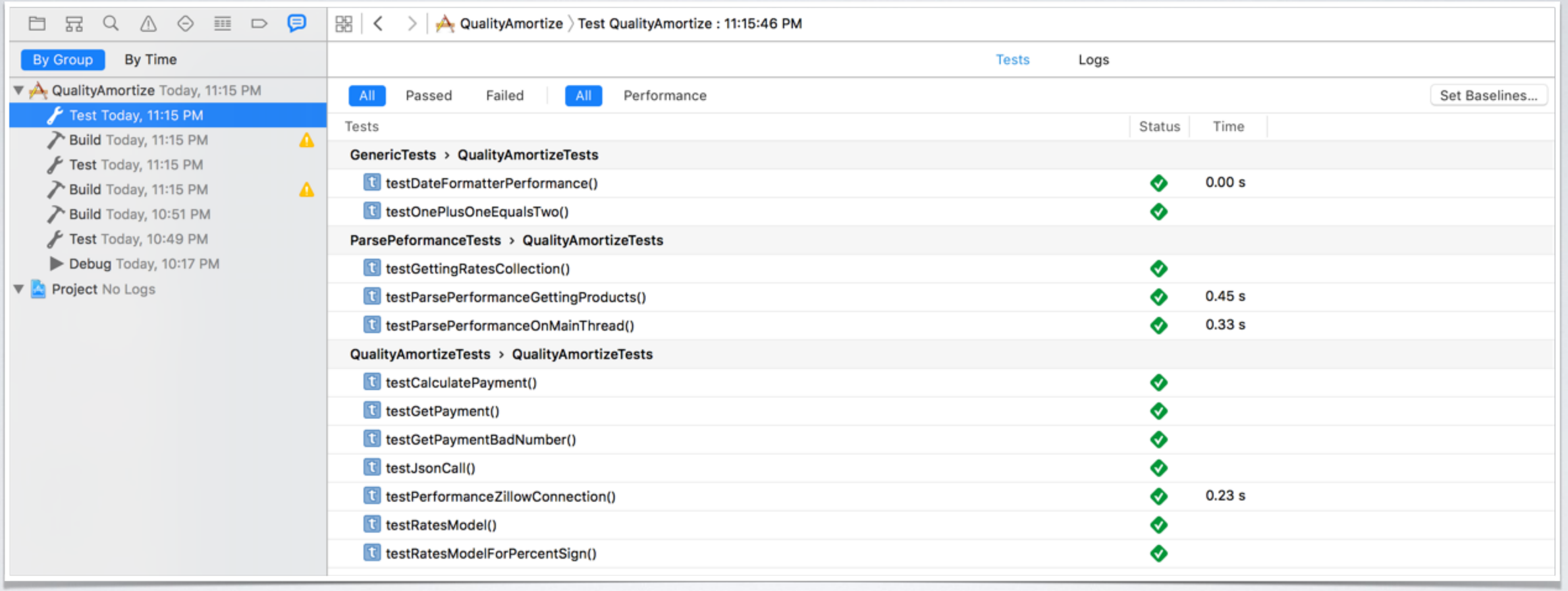
```swift
class Products {

    var objectId : String?
    var productId : Int?
    var photo : String?
    var photoFile : PFFile = PFFile()
    var photoImage : UIImage?
    var title : String?
    var flyer : String?
    var productDesc : String?
    var itemCode : String?
    var packagingOptions : String?
```

```swift
    func testGettingRatesCollection() {
        let arrayProducts : Array<Products> = Products().getAllProductsFromParse()

        XCTAssert(arrayProducts.count > 0, "\(arrayProducts.count) should be greater than zero")
    }
```

```swift
        var arrayProducts : Array<Products> = []

        do { let objects = try query.findObjects()
            for object in objects {
                self.objectId = object["objectId"] as? String
                self.productId = object["Id"] as? Int
                self.photoImage = object["PhotoImage"] as? UIImage
                self.title = object["Title"] as? String
                self.productDesc = object["Description"] as? String

                arrayProducts.append(self)
            }
        }
        catch let e as NSError { print("Parse load error: \(e)") }

        return arrayProducts
    }
```

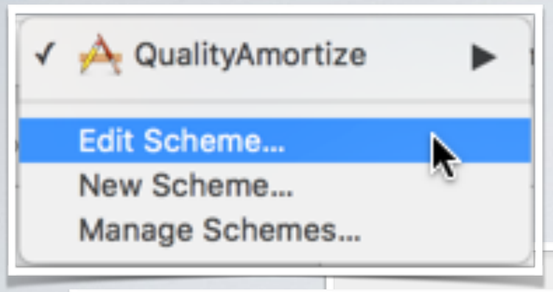
GroundSpeed™
rapid web + mobile software

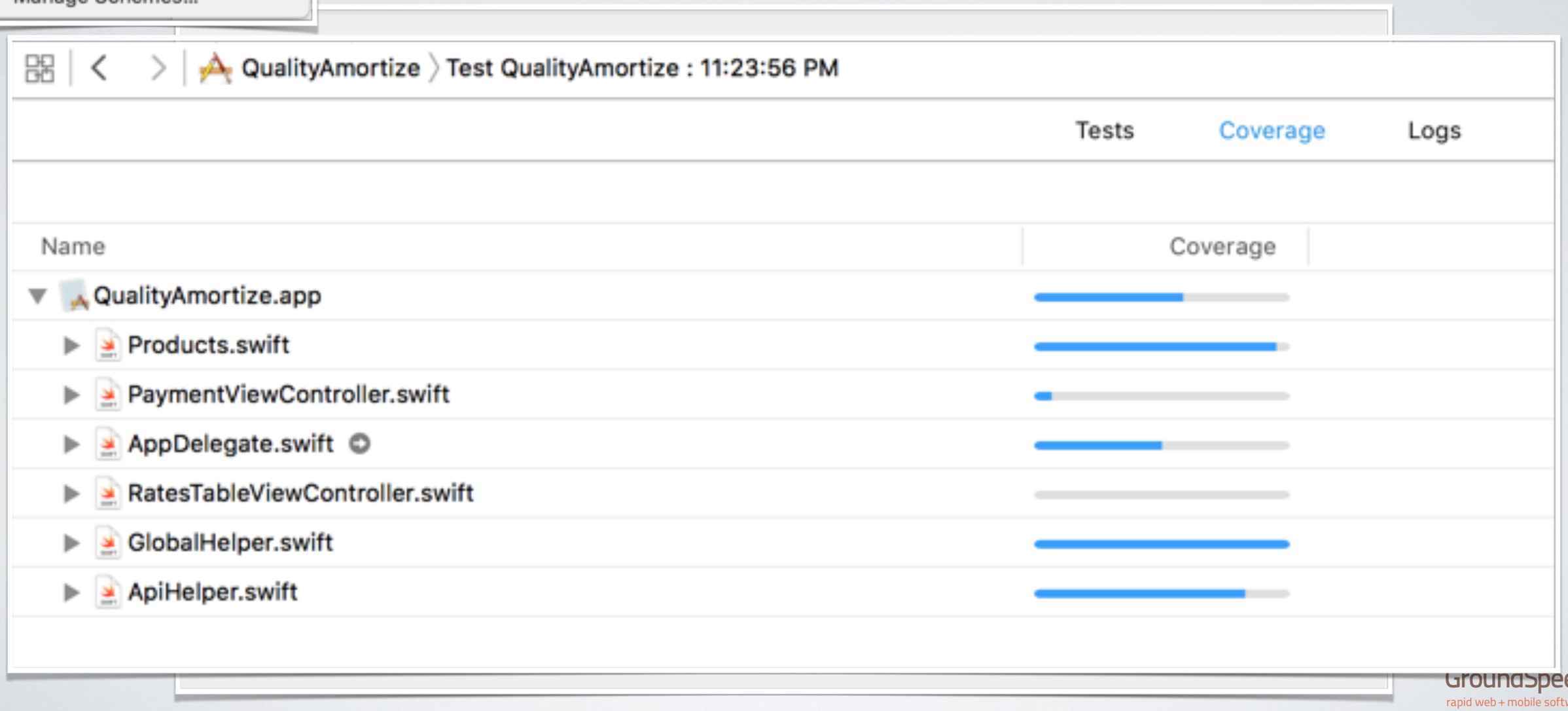
CODE COVERAGE



GroundSpeed™
rapid web + mobile software

CODE COVERAGE





QUESTIONS? THANK YOU!

Don Miller

don@GroundSpeedHQ.com

@donmiller

http://github.com/donmiller

http://github.com/groundspeed

GroundSpeed™

rapid web + mobile software