



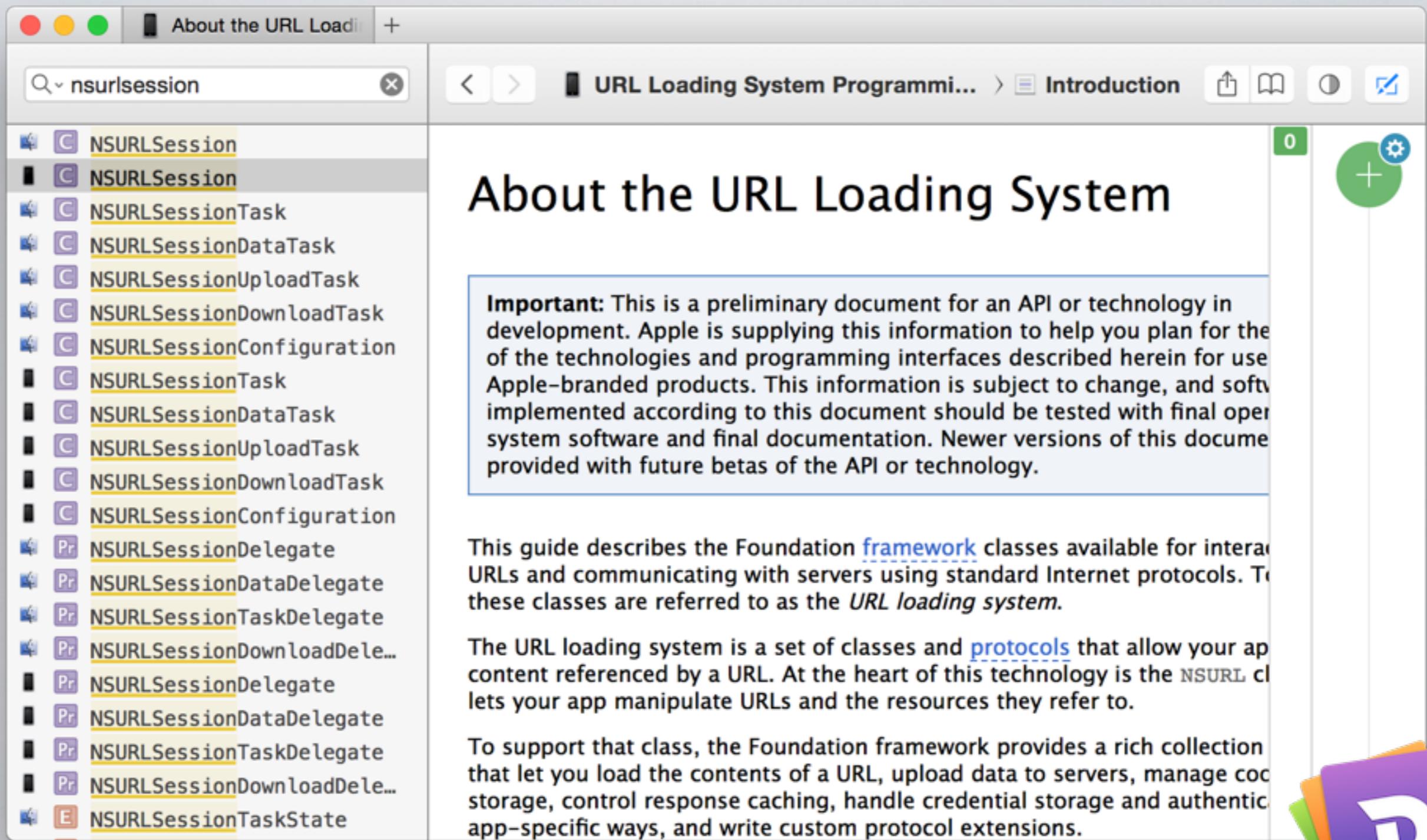
GETTING STARTED WITH XCODE

Nick Chen
Fall 2015
talentspark.io

GOALS

- Refresh Swift knowledge
- Learn how developers use CocoaPods + Xcode
- Pick up Xcode tips and tricks

LET'S LOAD A URL



The screenshot shows a Mac OS X-style window for the Apple Developer Documentation. The title bar says "About the URL Loadi..." and the search bar contains "nsurlsession". The main content area is titled "URL Loading System Programmi..." and has a breadcrumb trail "Introduction". On the right side, there's a green circular button with a plus sign and a gear icon. The left sidebar lists various classes and protocols under the heading "About the URL Loading System".

About the URL Loading System

Important: This is a preliminary document for an API or technology in development. Apple is supplying this information to help you plan for the use of the technologies and programming interfaces described herein for use in Apple-branded products. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document are provided with future betas of the API or technology.

This guide describes the Foundation [framework](#) classes available for interacting with URLs and communicating with servers using standard Internet protocols. Together, these classes are referred to as the *URL loading system*.

The URL loading system is a set of classes and [protocols](#) that allow your app to access content referenced by a URL. At the heart of this technology is the `NSURLSession` class, which lets your app manipulate URLs and the resources they refer to.

To support that class, the Foundation framework provides a rich collection of other classes and protocols that let you load the contents of a URL, upload data to servers, manage cookie storage, control response caching, handle credential storage and authentication in app-specific ways, and write custom protocol extensions.



NSURL

A screenshot of an Xcode playground window titled "MyPlayground.playground". The playground file is named "MyPlayground". The code in the playground is as follows:

```
import UIKit
import XCPlayground

// You need this so that we will wait
// for the results of async tasks
XCPSetExecutionShouldContinueIndefini-
tely(true)

let url = NSURL(string: "https://
api.github.com/users/vazexqi")!
```

The URL "https://api.github.com/users/vazexqi" is partially visible in the code editor, and the rest of the URL "https://api...." is shown in the gutter area.

At the bottom of the playground window, there is a toolbar with a play button icon, a progress bar, and a timer set to "30 sec".

NSURL

NSURL Class Reference +

nsurl

< > NSURLConnection Reference > Creating an NSURL Object > M init(string:)

0 +

init(string:)

Initializes an NSURL object with a provided URL string.

Declaration

SWIFT

```
convenience init?(string URLString: String)
```

Parameters

URLString The URL string with which to initialize the NSURL object. This URL string must conform to URL format as described in RFC 2396, and must not be `nil`. This method parses *URLString* according to RFCs 1738 and 1808.

Return Value

An NSURL object initialized with *URLString*. If the URL string was malformed, returns `nil`.



NSURLSession

A screenshot of an Xcode playground window titled "MyPlayground.playground". The playground contains the following Swift code:

```
let url = NSURL(string: "https://api.github.com/users/vazexqi")!
let task = NSURLSession.sharedSession().dataTaskWithURL(url) {
    data, response, error in
    print(data)
    print(response)
    print(error)
}
```

The URL "https://api.github.com/users/vazexqi" is partially resolved in the gutter, showing "<__NSCF...>" followed by "https://api....". The bottom right corner of the playground shows a timer set to "30 sec".

NSURLSession

NSURLSession Class Reference

The NSURLSession class and related classes provide an API for downloading content via HTTP. This API provides a rich set of delegate methods for supporting authentication and gives your app the ability to perform background downloads when your app is not running or, in iOS, while your app is suspended.

With the NSURLSession API, your app creates a series of sessions, each of which coordinates a group of related data transfer tasks. For example, if you are writing a web browser, your app might create one session per tab or window. Within each session, your app adds a series of tasks, each of which represents a request for a specific URL (and for any follow-on URLs if the original URL returned an HTTP redirect).

The behavior of a session is determined by the configuration object used to create it. Because there are three types of configuration objects, there are similarly three types of sessions: *default sessions* that behave much like NSURLConnection, *ephemeral sessions* that do not cache anything to disk, and *download sessions* that store the results in a file and continue transferring data even when your app is suspended, exits, or crashes.



NSURLSession

A screenshot of an Xcode playground window titled "MyPlayground.playground". The code being typed is:

```
let url = NSURL(string: "https://api.github.com/users/vazexqi")!  
let task =  
    NSURLSession.sharedSession().  
    data  
    data  
    pri  
    pri  
    pri  
}  
}
```

The cursor is positioned at the end of the line "NSURLSession.sharedSession()." A callout box provides the following documentation for the `sharedSession()` method:

Declaration `class func sharedSession() -> NSURLSession`
Description Returns a shared singleton session object.
The shared session uses the currently set global `NSURLCache`, `NSHTTPCookieStorage`, and `NSURLCredentialStorage` objects and is based on the default configuration.
Availability iOS (8.1 and later)
Declared In `Foundation`
Reference [NSURLSession Class Reference](#)

On the right side of the playground, there is a preview area showing the URL "https://api...." and a placeholder text "NSCFL...". At the bottom of the playground window, there is a toolbar with a search field containing "- 30 sec +".

NSURLSession

The screenshot shows an Xcode playground window titled "MyPlayground.playground". The code being run is:

```
let url = NSURL(string: "https://api.github.com/users/vazexqi")!
let task = NSURLSession.sharedSession().dataTaskWithURL(url) {
```

A callout box provides documentation for the `dataTaskWithURL` method:

Declaration: `func dataTaskWithURL(url: NSURL, completionHandler: (NSData?, NSURLResponse?, NSError?) -> Void) -> NSURLSessionDataTask`

Description: Creates an HTTP GET request for the specified URL, then calls a handler upon completion.

The `url` parameter is described as: The http or https URL to be retrieved.

The `completionHandler` parameter is described as:

- The completion handler to call when the load request is complete. If sent to a session created by calling `sessionWithConfiguration:delegate:delegateQueue:` with a non-nil value for the `delegateQueue` parameter, this handler is executed on that delegate queue.
- Unless you have provided a custom delegate, this parameter must not be nil, because there is no other way to retrieve the response data.

Returns: The new session data task.

Availability: iOS (8.1 and later)

TASK

The screenshot shows an Xcode playground window titled "MyPlayground.playground". The code in the playground creates a URLSession task to fetch data from GitHub:

```
let url = NSURL(string: "https://api.github.com/users/vazexqi")!
let task = URLSession.sharedSession().dataTaskWithURL(url) {
    data, response, error in
    print(data)
    print(response)
    print(error)
}

task.resume()
```

The right side of the screen displays the results of the print statements:

- <__NSCFString: 0x109e01000> (length=16)
"Optional("Optional(...))"
- <__NSCFString: 0x109e01000> (length=16)
"Optional("Optional(...))"
- "nil"
- <__NSCFString: 0x109e01000> (length=16)
"Optional("Optional(...))"

At the bottom of the playground window, there is a toolbar with a play button and a timer set to "30 sec".

TASK

NSURLSessionTask Class Reference

NSURLSessionDownloadTask subclasses of NSURLConnection, respectively.

- Data tasks request a resource, returning the server's response as one or more `NSData` objects in memory. They are supported in default, ephemeral, and shared sessions, but are not supported in background sessions.
- Upload tasks are like data tasks, except that they make it easier to provide a request body so you can upload data before retrieving the server's response. Additionally, upload tasks are supported in background sessions.
- Download tasks download a resource directly to a file on disk. Download tasks are supported in any type of session.

After you create a task, you start it by calling its `resume` method. The session then maintains a strong reference to the task until the request finishes or fails; you do not need to maintain a reference to the task unless it is useful to do so for your app's internal bookkeeping purposes.



RECAP

RECAP

NSURL

RECAP

NSURLSession

NSURL

RECAP

NSURLSessionDataTask

NSURLSession

NSURL

RECAP

NSURLSessionDataTask

resume()

NSURLSession

NSURL

RECAP

NSURLSessionUploadTask

NSURLSessionDataTask

resume()

NSURLSession

NSURL

NSURLSessionDownloadT
ask

COCOAPODS

The image shows a screenshot of a web browser displaying the official website for Cocoapods (cocoapods.org). The browser interface includes standard controls like back, forward, and search buttons, along with a lock icon and the URL in the address bar. The main content area features a large section titled "WHAT IS COCOAPODS" with a descriptive paragraph about the tool's purpose. Below this, there are three main navigation links: "INSTALL", "GET STARTED", and "CREATE A POD". Each link is accompanied by a decorative wavy line graphic.

cocoapods.org

WHAT IS COCOAPODS

CocoaPods is the dependency manager for Swift and Objective-C Cocoa projects. It has over ten thousand libraries and can help you scale your projects elegantly.

INSTALL

**GET
STARTED**

**CREATE A
POD**

COCOAPODS

COCOAPODS



My Super Cool Project

COCOAPODS



Framework #1



My Super Cool Project

COCOAPODS



Framework #1



My Super Cool Project



Framework #2

COCOAPODS



Framework #1



Framework #1.a



My Super Cool Project



Framework #2

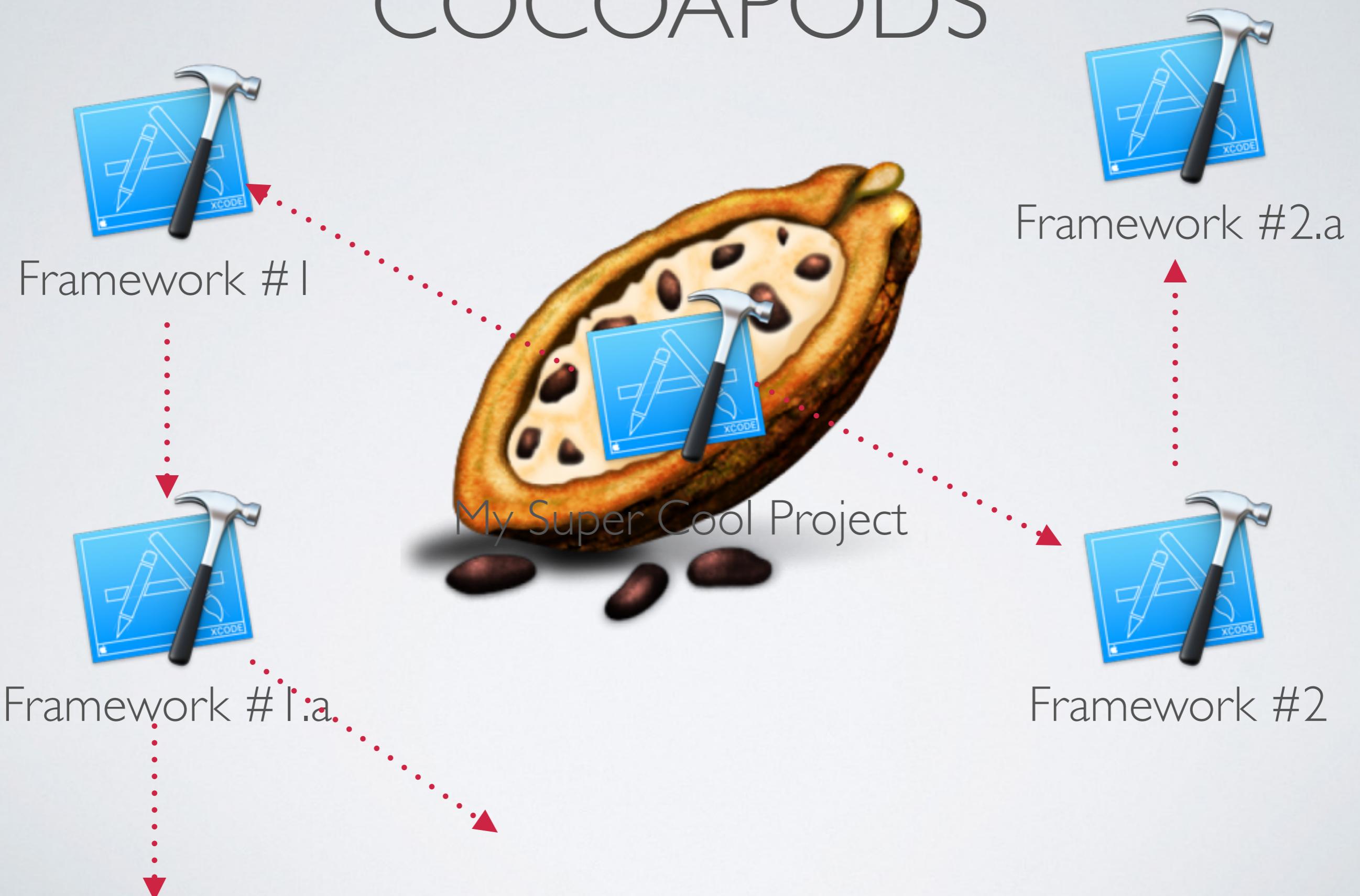
COCOAPODS



COCOAPODS



COCOAPODS



COCOAPODS

```
2. sudo gem install cocoapods (sudo)
vazexqi@daigo:~|=> sudo gem install cocoapods          19:15:53
Fetching: cocoapods-core-0.38.2.gem (100%)
Successfully installed cocoapods-core-0.38.2
Fetching: xcdeproj-0.26.3.gem (100%)
Successfully installed xcdeproj-0.26.3
Fetching: molinillo-0.3.1.gem (100%)
Successfully installed molinillo-0.3.1
Fetching: cocoapods-0.38.2.gem (100%)
Successfully installed cocoapods-0.38.2
Parsing documentation for cocoapods-core-0.38.2
Installing ri documentation for cocoapods-core-0.38.2
Parsing documentation for xcdeproj-0.26.3
Installing ri documentation for xcdeproj-0.26.3
Parsing documentation for molinillo-0.3.1
Installing ri documentation for molinillo-0.3.1
Parsing documentation for cocoapods-0.38.2
Installing ri documentation for cocoapods-0.38.2
```

COCOAPODS

```
2. vazexqi@daigo: ~ (zsh)
vazexqi@daigo:~|⇒ pod --help
19:17:10
Usage:

$ pod COMMAND

CocoaPods, the Cocoa library package manager.

Commands:

+ cache      Manipulate the CocoaPods cache
+ init        Generate a Podfile for the current directory.
+ install     Install project dependencies to Podfile.lock versions
+ ipc         Inter-process communication
+ lib         Develop pods
+ list        List pods
+ outdated   Show outdated project dependencies
+ plugins    Show available CocoaPods plugins
+ repo       Manage spec-repositories
+ search     Searches for pods
+ setup      Setup the CocoaPods environment
+ spec       Manage pod specs
+ trunk      Interact with the CocoaPods API (e.g. publishing new
```

COCOAPODS

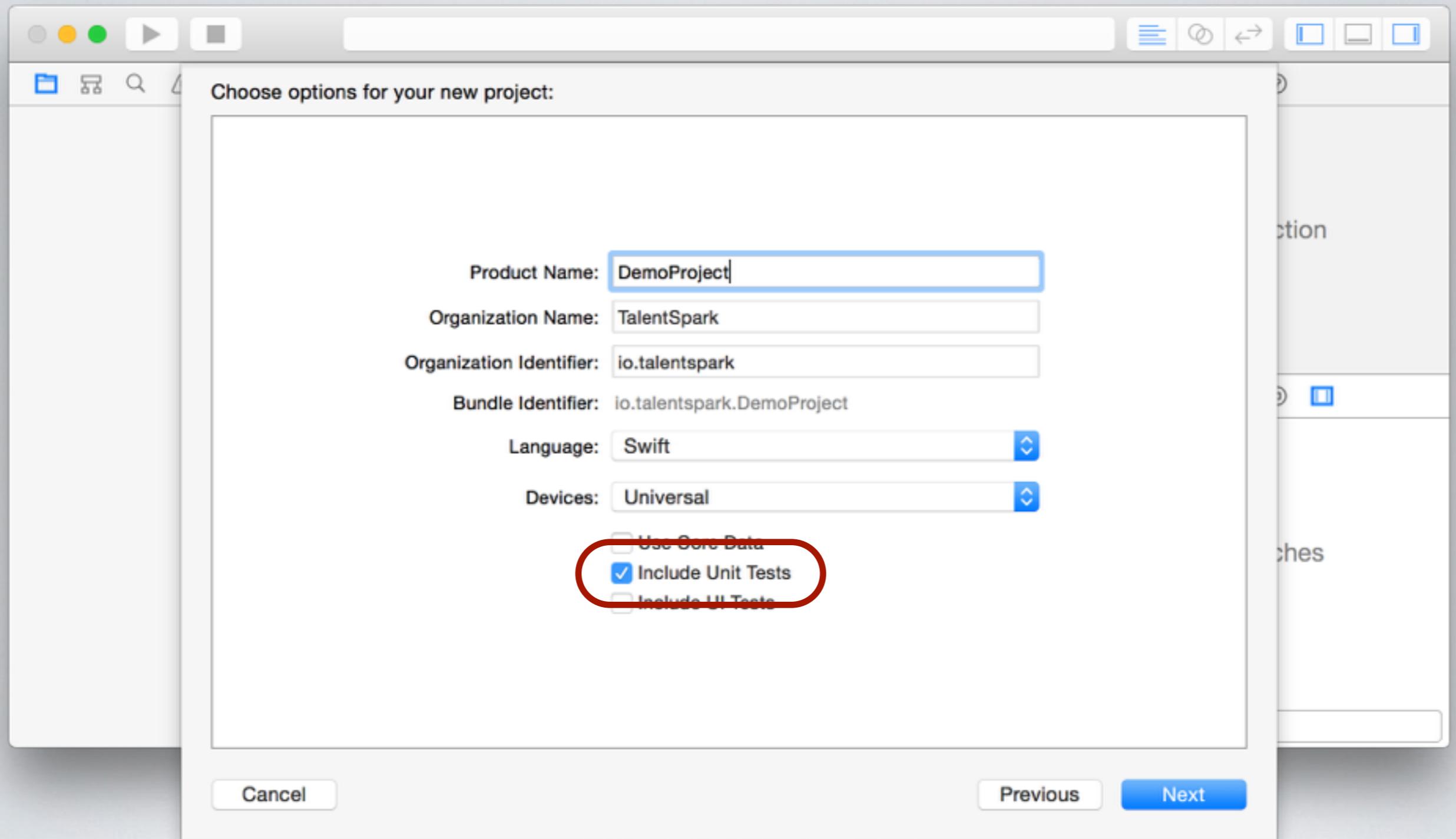
```
2. vazexqi@daigo: ~ (zsh)
vazexqi@daigo:~|=> pod search alamofire          19:19:03

-> Alamofire (1.2.3)
    Elegant HTTP Networking in Swift
    pod 'Alamofire', '~> 1.2.3'
    - Homepage: https://github.com/Alamofire/Alamofire
    - Source:   https://github.com/Alamofire/Alamofire.git
    - Versions: 1.2.3, 1.2.2, 1.2.1, 1.2.0, 1.1.5, 1.1.4, 1.1.3 [master
      repo]

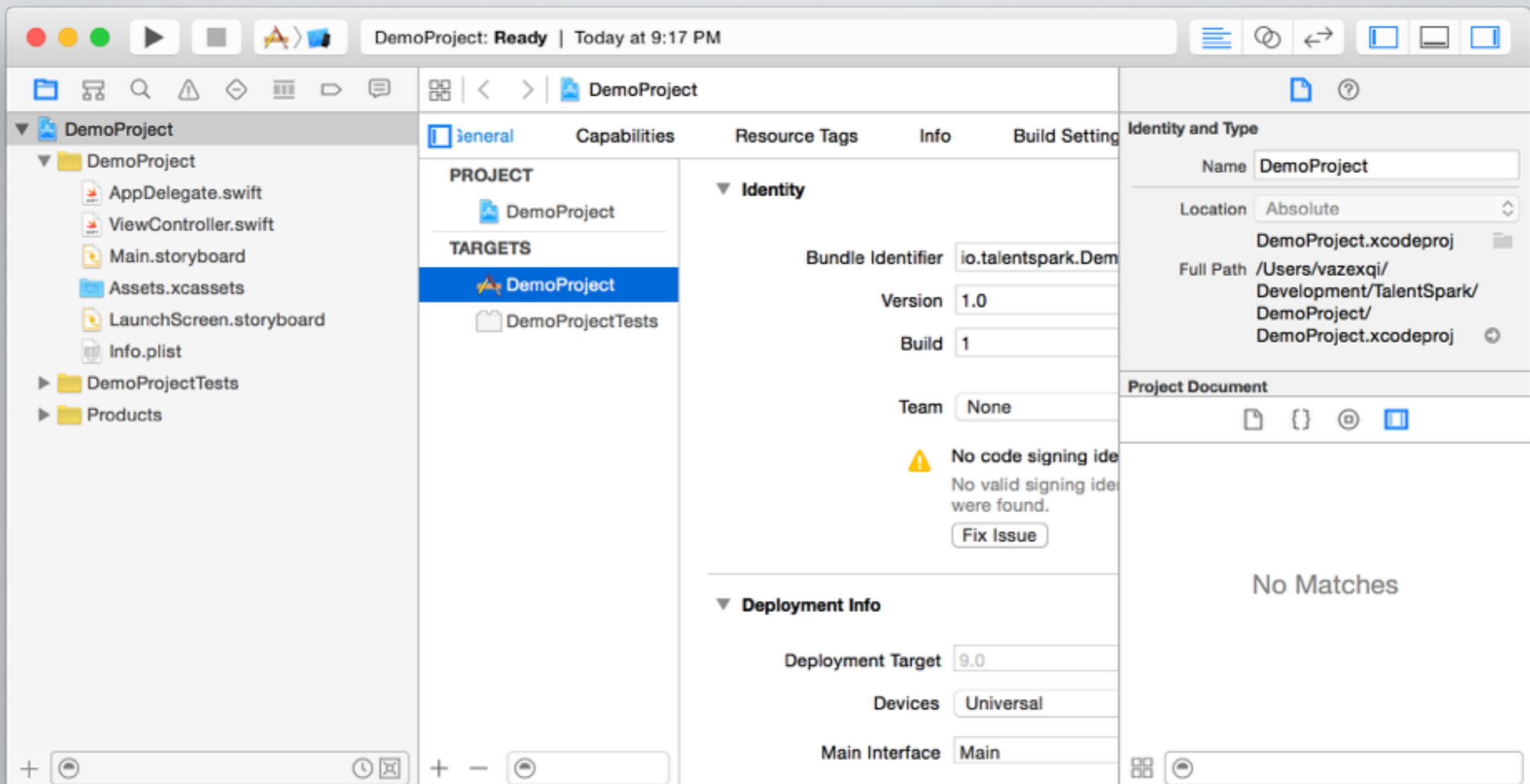
-> AlamofireJsonToObjects (1.0.0)
    An Alamofire extension which converts JSON response data into swift
    objects using EVReflection
    pod 'AlamofireJsonToObjects', '~> 1.0.0'
    - Homepage: https://github.com/evermeer/EVReflection
    - Source:   https://github.com/evermeer/AlamofireJsonToObjects.git
    - Versions: 1.0.0 [master repo]

-> AlamofireOAuth2 (1.0.1)
```

XCODE



XCODE



XCODE

Edit View Find Navigate Editor Product Debug Source Control Window Help

Search documentation

At a Glance

Use the App Store app on your Mac to [download Xcode](#). It's free. After you download Xcode, open it from Launchpad, where you can click the icon for Xcode to launch it.

Documentation and API Reference

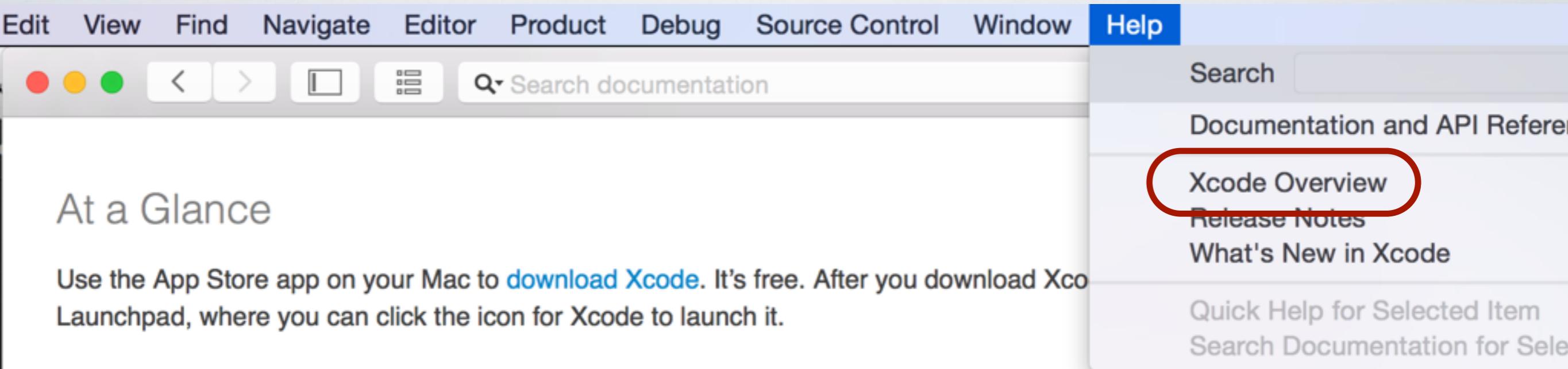
Xcode Overview (highlighted)

Release Notes

What's New in Xcode

Quick Help for Selected Item

Search Documentation for Selected Item



CHEAT SHEET

The screenshot shows a web browser displaying the "XCODE CHEAT SHEET" from www.git-tower.com/blog/xcode-cheat-sheet/. The page features a header with the fournova logo, the title "XCODE CHEAT SHEET" presented by TOWER, and a subtitle "Version control with Git - made easy". To the right is a small illustration of a lighthouse. On the far right, there are social media sharing icons for Twitter, Google+, Facebook, and LinkedIn. The main content is organized into three sections: "SEARCH", "NAVIGATION", and "EDITING", each listing keyboard shortcuts. Below these sections is a "TABS" section. At the bottom, there's a diagram showing a Mac OS X window with tabs, connected by lines to the corresponding keyboard shortcut keys.

SEARCH	
Find in File	⌘ F
Find & Replace in File	⌘ ⌘ F
Find in Project	⌘ ⌘ F
Find & Replace in Project	⌘ ⌘ ⌘ F

NAVIGATION	
Open Quickly	⌘ ⌘ 0
Move Focus to Editor	⌘ J
Next Counterpart	⌘ ⌈ UP
Previous Counterpart	⌘ ⌉ DOWN
Next Recent File	⌘ ⌈ RIGHT
Previous Recent File	⌘ ⌉ LEFT
Definition of Symbol	⌘ click
Go to Line	⌘ L
Fold Section	⌘ ⌘ ⌘ LEFT
Unfold Section	⌘ ⌘ ⌘ RIGHT

EDITING	
Show Assistant Editor	⌘ ⌈ RETURN
Hide Assistant Editor	⌘ RETURN
Toggle Completions	⌃ SPACE
Edit All in Scope	⌘ ⌈ E
Indent Selection	⌘]
Outdent Selection	⌘ [
Fix Indentation	⌃ i
Comment / Uncomment	⌘ /
Move Line Up	⌘ ⌈ [
Move Line Down	⌘ ⌈]

TABS	
New Tab	⌘ T
Previous Tab	⌘ }
Next Tab	⌘ {

TERMINAL

```
2. vazexqi@daigo: ~/Development/TalentSpark/DemoProject (zsh)
vazexqi@daigo:~/Development/TalentSpark/DemoProject|=> ls          21:23:14
DemoProject           DemoProjectTests
DemoProject.xcodeproj  Podfile
vazexqi@daigo:~/Development/TalentSpark/DemoProject|=> █          21:23:15
```

TERMINAL

A screenshot of a terminal window showing a Vim session. The title bar says "2. mvim -vf Podfile (Vim)". The main area contains the following code:

```
1 pod 'Alamofire', :git => 'https://github.com/Alamofire/Alamofire.  
git', :branch => 'swift-2.0'  
2 use_frameworks!
```

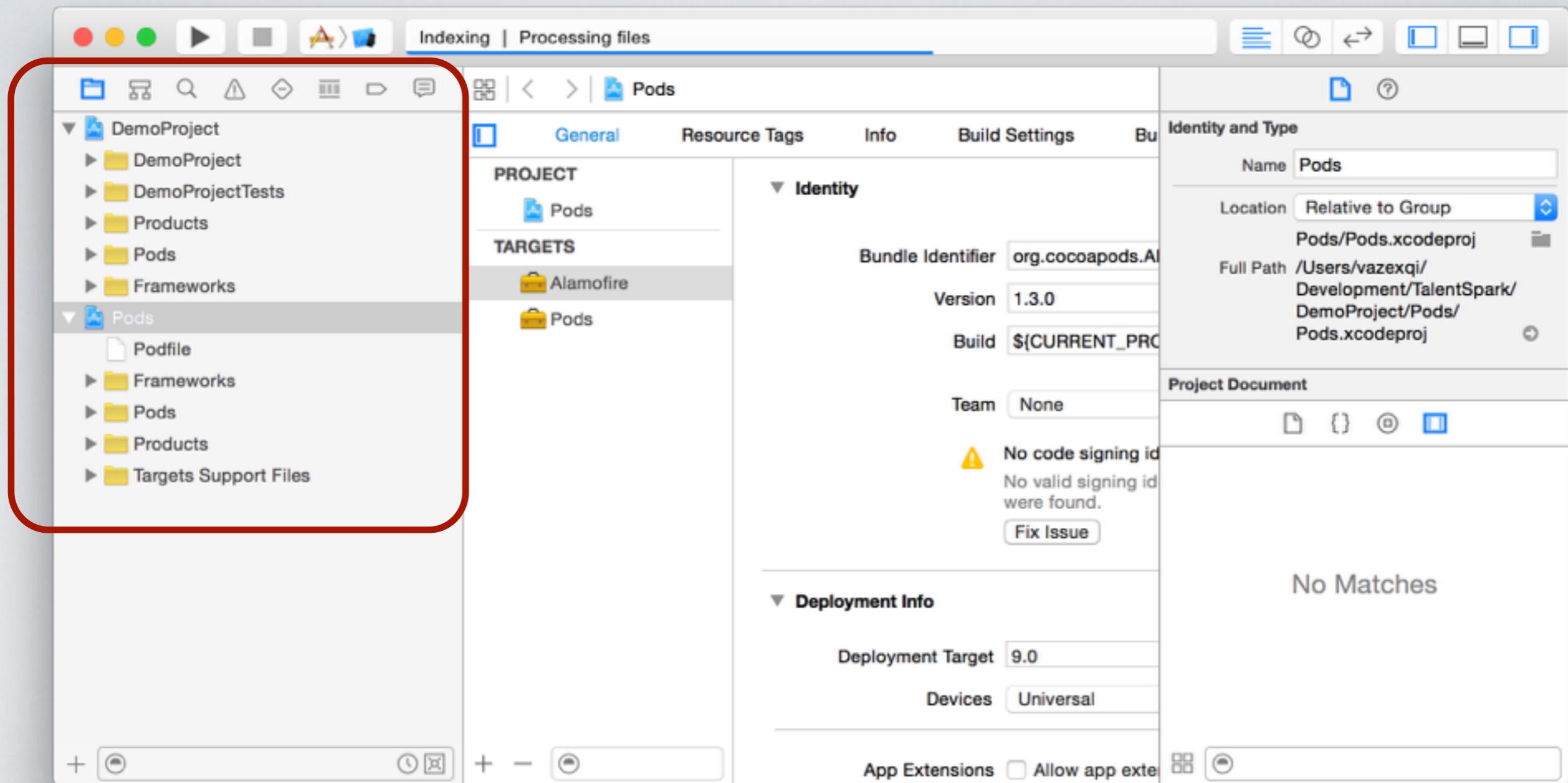
The file has 21 lines, indicated by the "2~" prompt at the bottom left. The status bar at the bottom shows "NORMAL > Podfile +", "unix < utf-8 < ruby < 100% < ^N 2:2".

TERMINAL

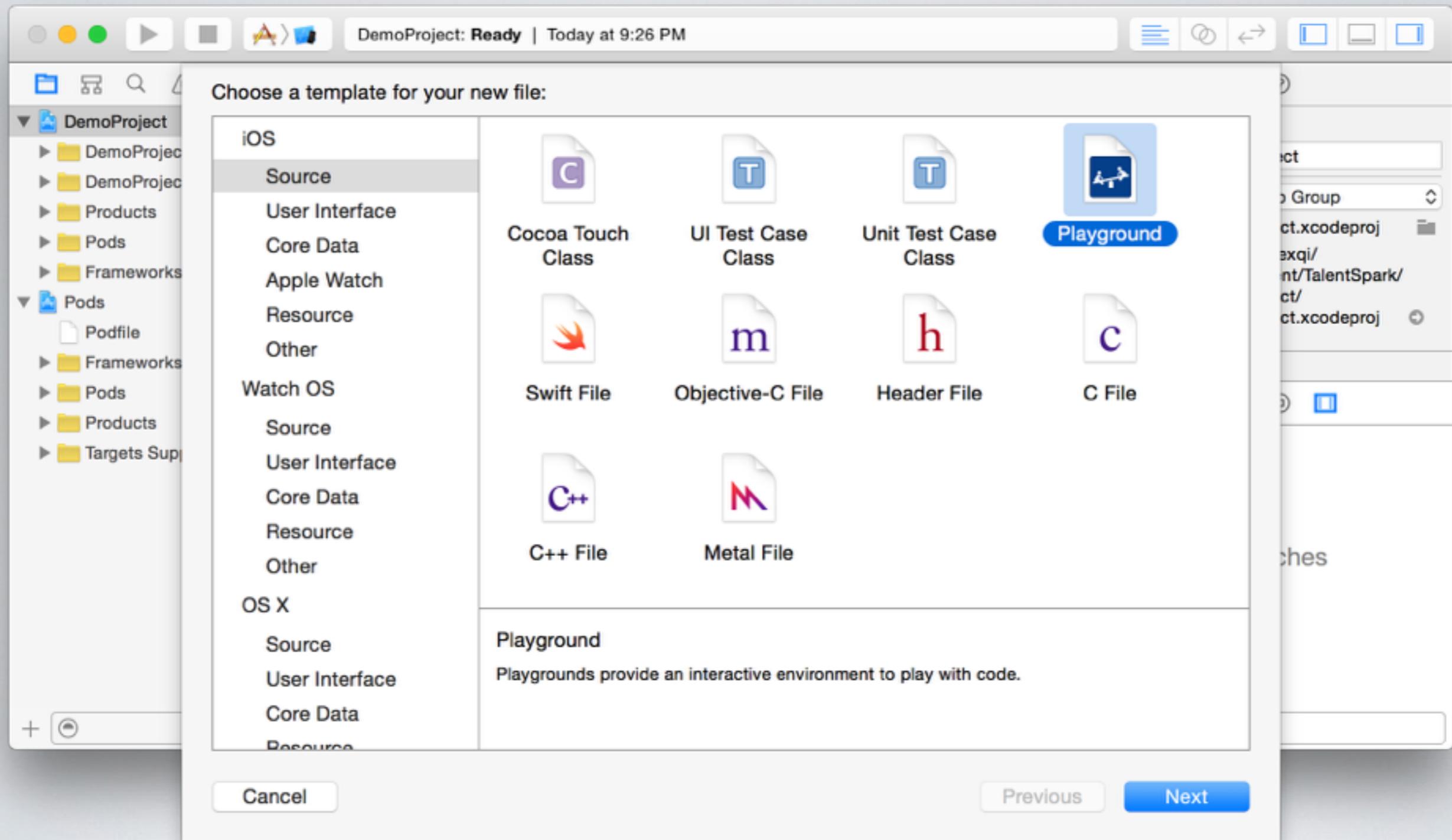
```
2. vazexqi@daigo: ~/Development/TalentSpark/DemoProject (zsh)
vazexqi@daigo:~/Development/TalentSpark/DemoProject|=> ls          21:23:14
DemoProject           DemoProjectTests
DemoProject.xcodeproj Podfile
vazexqi@daigo:~/Development/TalentSpark/DemoProject|=> pod install
Updating local specs repositories
Analyzing dependencies
Pre-downloading: `Alamofire` from `https://github.com/Alamofire/Alamofire.g
it`, branch `swift-2.0`
Downloading dependencies
Installing Alamofire (1.3.0)
Generating Pods project
Integrating client project

[!] Please close any current Xcode sessions and use `DemoProject.xcworkspace` for this project from now on.
Sending stats
vazexqi@daigo:~/Development/TalentSpark/DemoProject|=> █          21:23:45
```

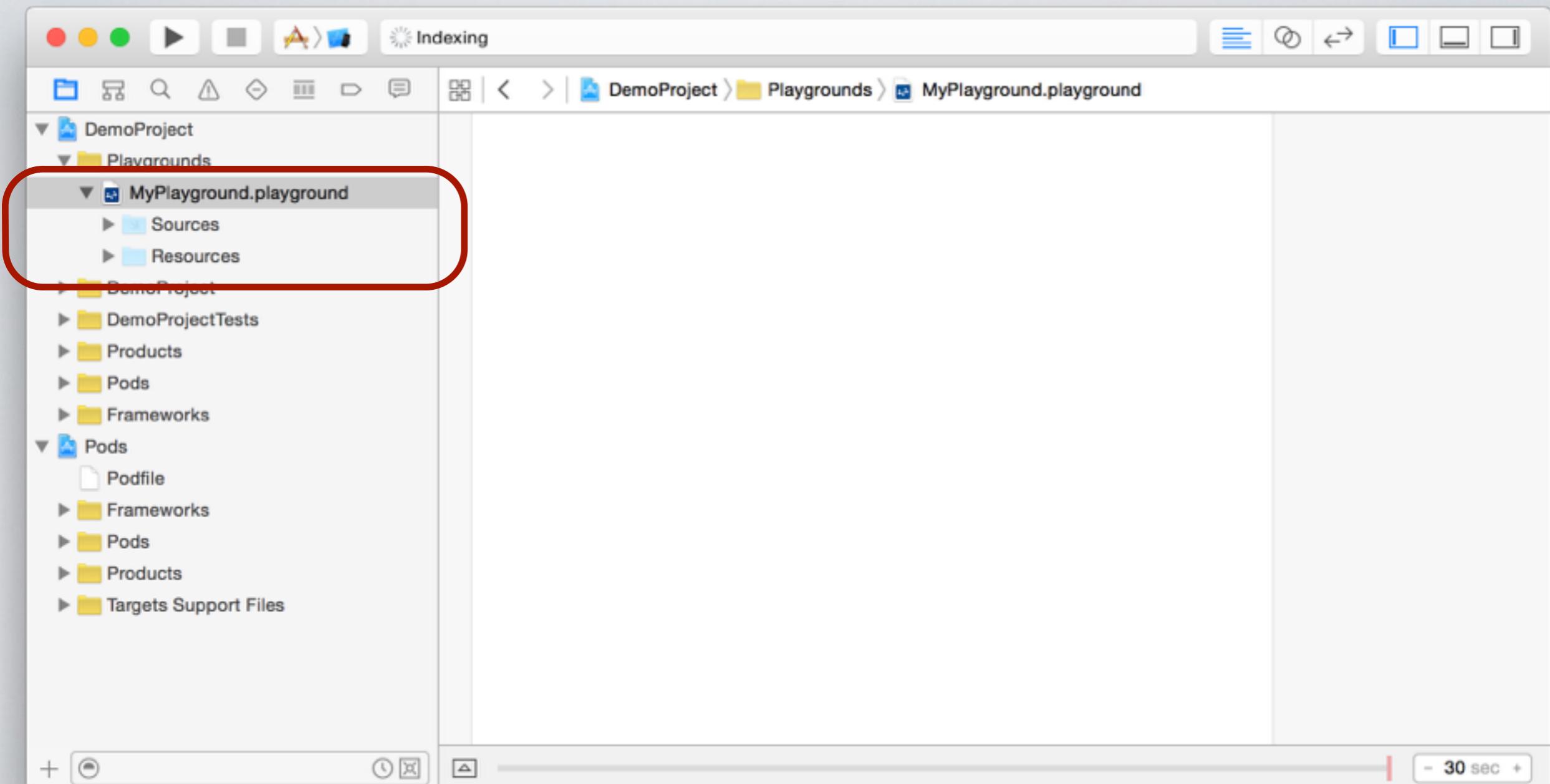
XCODE



XCODE



XCODE



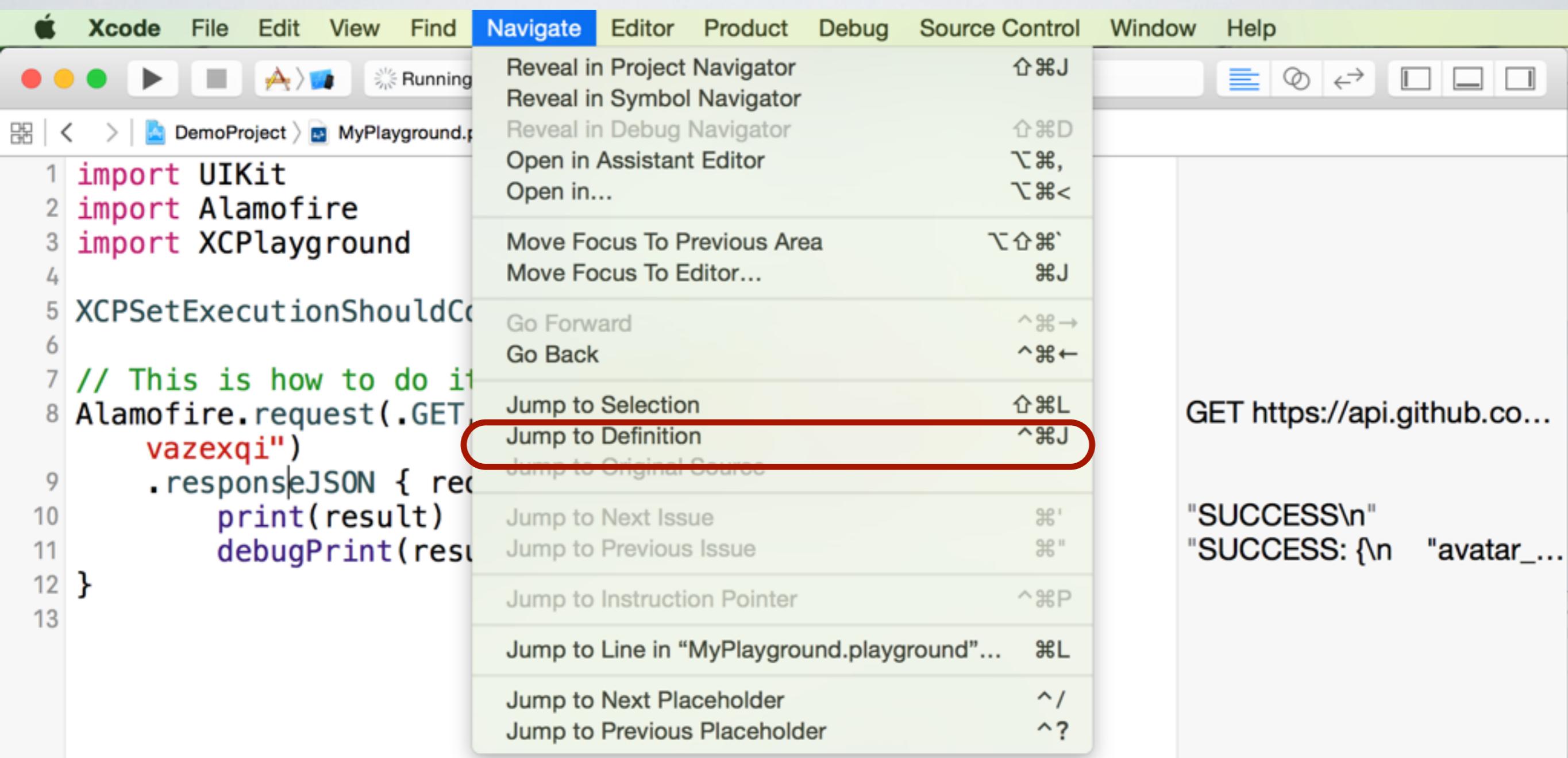
XCODE

The screenshot shows the Xcode interface with a playground window titled "Running MyPlayground". The code in the playground imports UIKit, Alamofire, and XCPlayground, sets execution to continue indefinitely, and performs a GET request to GitHub's user API. The output pane shows the successful response.

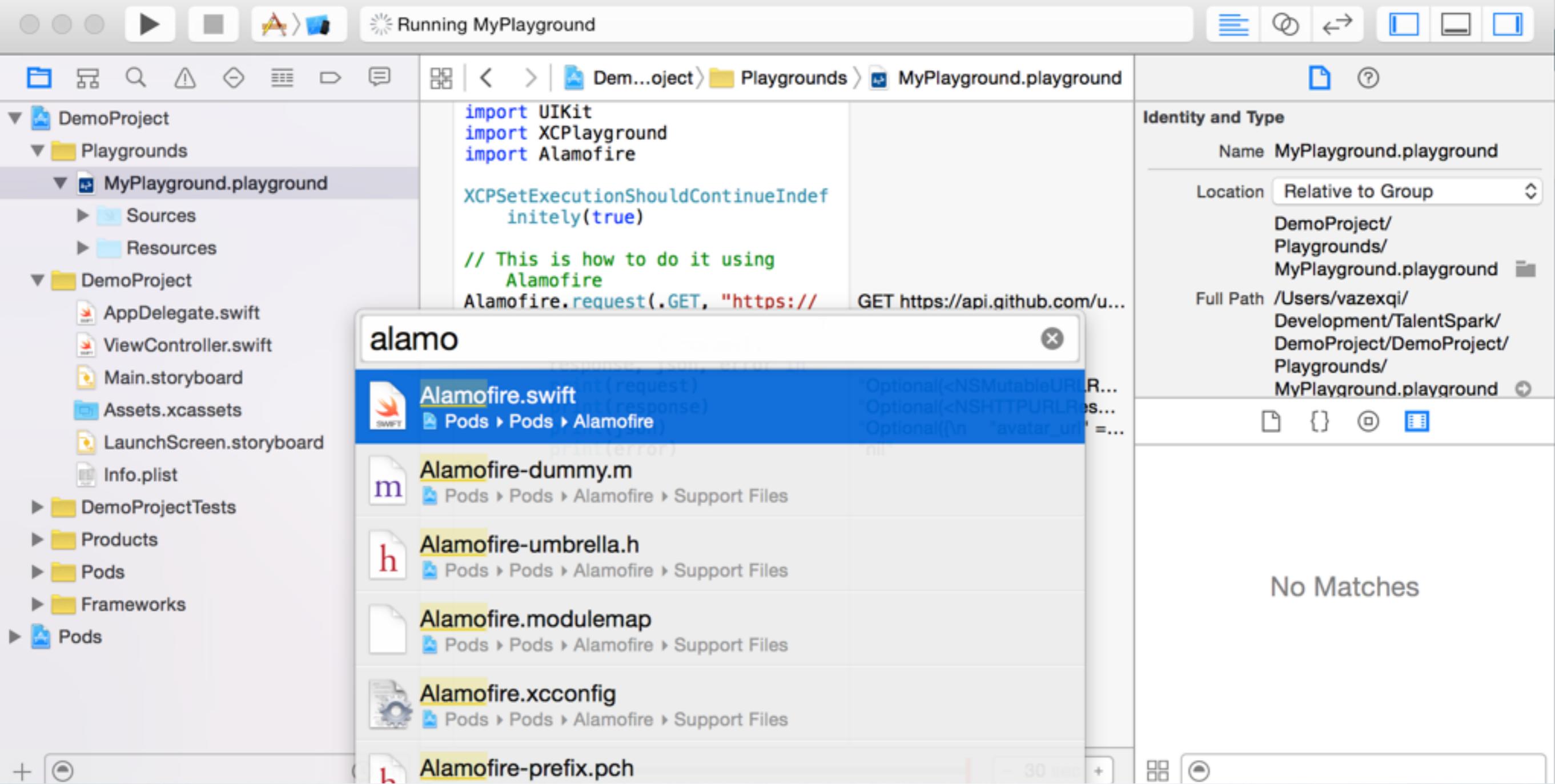
```
1 import UIKit
2 import Alamofire
3 import XCPlayground
4
5 XCPSetExecutionShouldContinueIndefinitely(true)
6
7 // This is how to do it with Alamofire
8 Alamofire.request(.GET, "https://api.github.com/users/vazexqi")
9     .responseJSON { request, response, result in
10         print(result)
11         debugPrint(result)
12 }
13
```

GET https://api.github.co...
"SUCCESS\n"
"SUCCESS: {\n "avatar_...

XCODE



XCODE



ALAMofire



A screenshot of a Mac OS X terminal window. The title bar says "GitHub, Inc.". The search bar contains "responsejson". The main pane shows the following Swift code:

```
Alamofire.request(.GET, "http://httpbin.org/get")
    .responseString { _, _, result in
        print("Success: \(result.isSuccess)")
        print("Response String: \(result.value)")
    }
```

Response JSON Handler

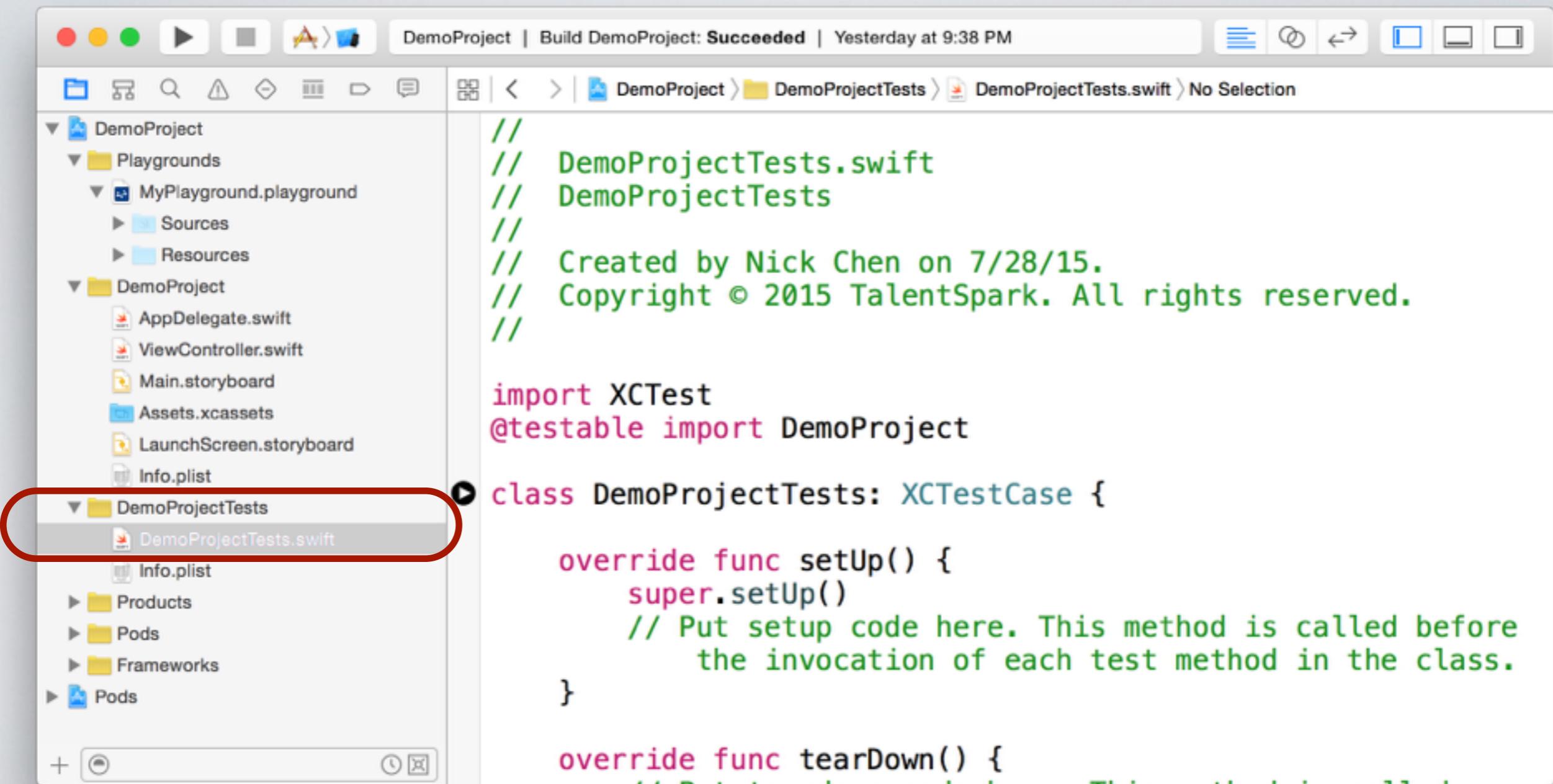
```
Alamofire.request(.GET, "http://httpbin.org/get")
    .responseJSON { _, _, result in
        print(result)
        debugPrint(result)
    }
```

Chained Response Handlers

Response handlers can even be chained:

```
Alamofire.request(.GET, "http://httpbin.org/get")
    .responseString { _, _, result in
        print("Response String: \(result.value)")
```

UNIT TESTS



The screenshot shows the Xcode interface with the following details:

- Toolbar:** Standard Mac OS X window controls.
- Status Bar:** Displays "DemoProject | Build DemoProject: Succeeded | Yesterday at 9:38 PM".
- Navigation Bar:** Shows the current path: DemoProject > DemoProjectTests > DemoProjectTests.swift, with "No Selection" highlighted.
- Project Navigator (Left):** Lists the project structure:
 - DemoProject (selected)
 - Playgrounds
 - MyPlayground.playground
 - Sources
 - Resources
 - DemoProject
 - AppDelegate.swift
 - ViewController.swift
 - Main.storyboard
 - Assets.xcassets
 - LaunchScreen.storyboard
 - Info.plist
 - DemoProjectTests
 - (selected) DemoProjectTests.swift
 - Info.plist
 - Products
 - Pods
 - Frameworks
- Editor Area (Right):** Displays the content of DemoProjectTests.swift:

```
// DemoProjectTests.swift
// DemoProjectTests
//
// Created by Nick Chen on 7/28/15.
// Copyright © 2015 TalentSpark. All rights reserved.

import XCTest
@testable import DemoProject

class DemoProjectTests: XCTestCase {

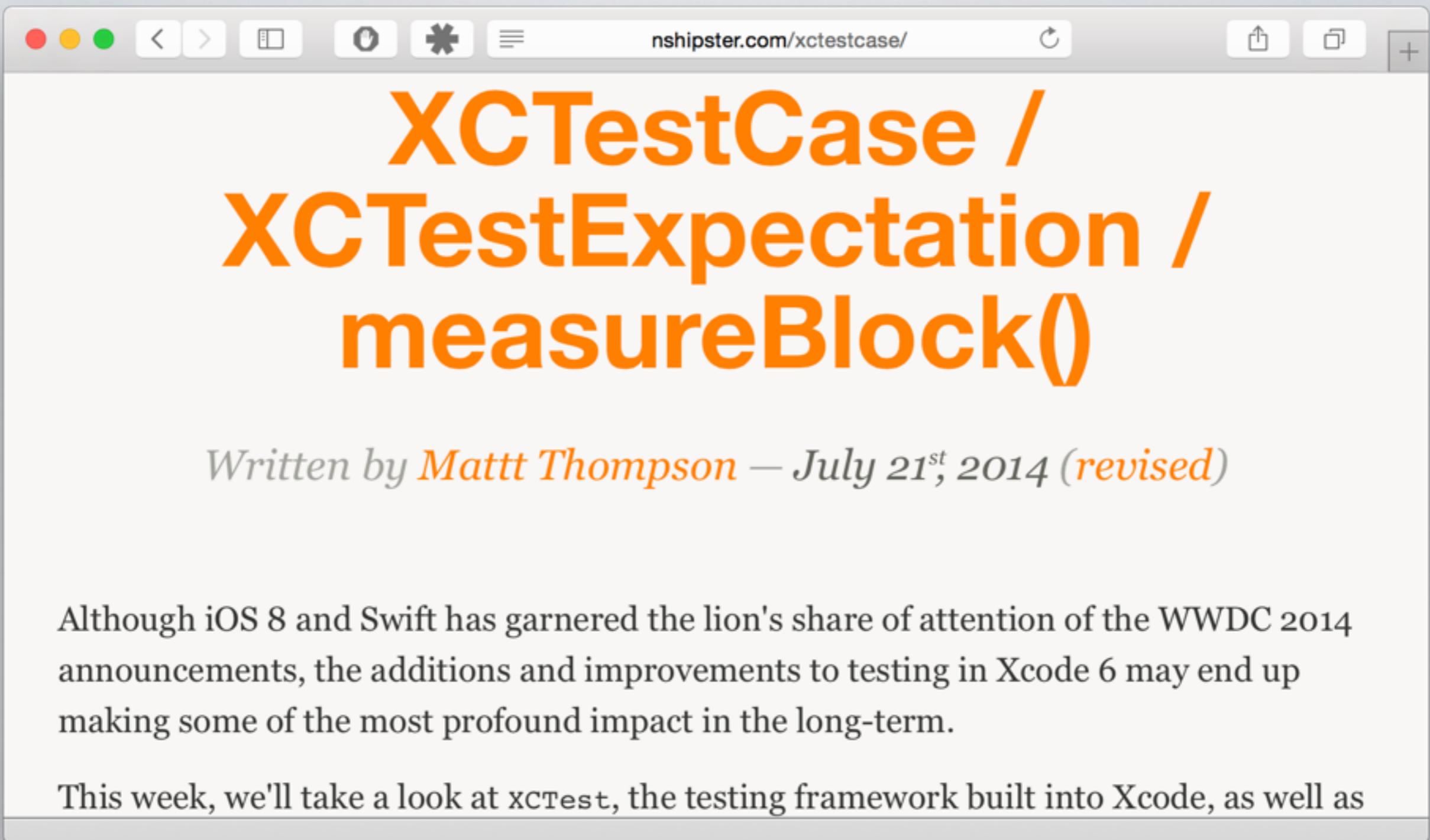
    override func setUp() {
        super.setUp()
        // Put setup code here. This method is called before
        // the invocation of each test method in the class.
    }

    override func tearDown() {
        // Put teardown code here. This method is called after
        // the invocation of each test method in the class.
    }

    func testExample() {
        // This is an example of a functional test case.
        // Use XCTAssert and related functions to verify your tests produce the
        // correct results.
        // Let's write some tests!
    }

    func testPerformanceExample() {
        // This is an example of a performance test case.
        // Use計時器和相关函数来验证你的测试是否产生正确的结果。
        // Let's write some tests!
    }
}
```

UNIT TESTS



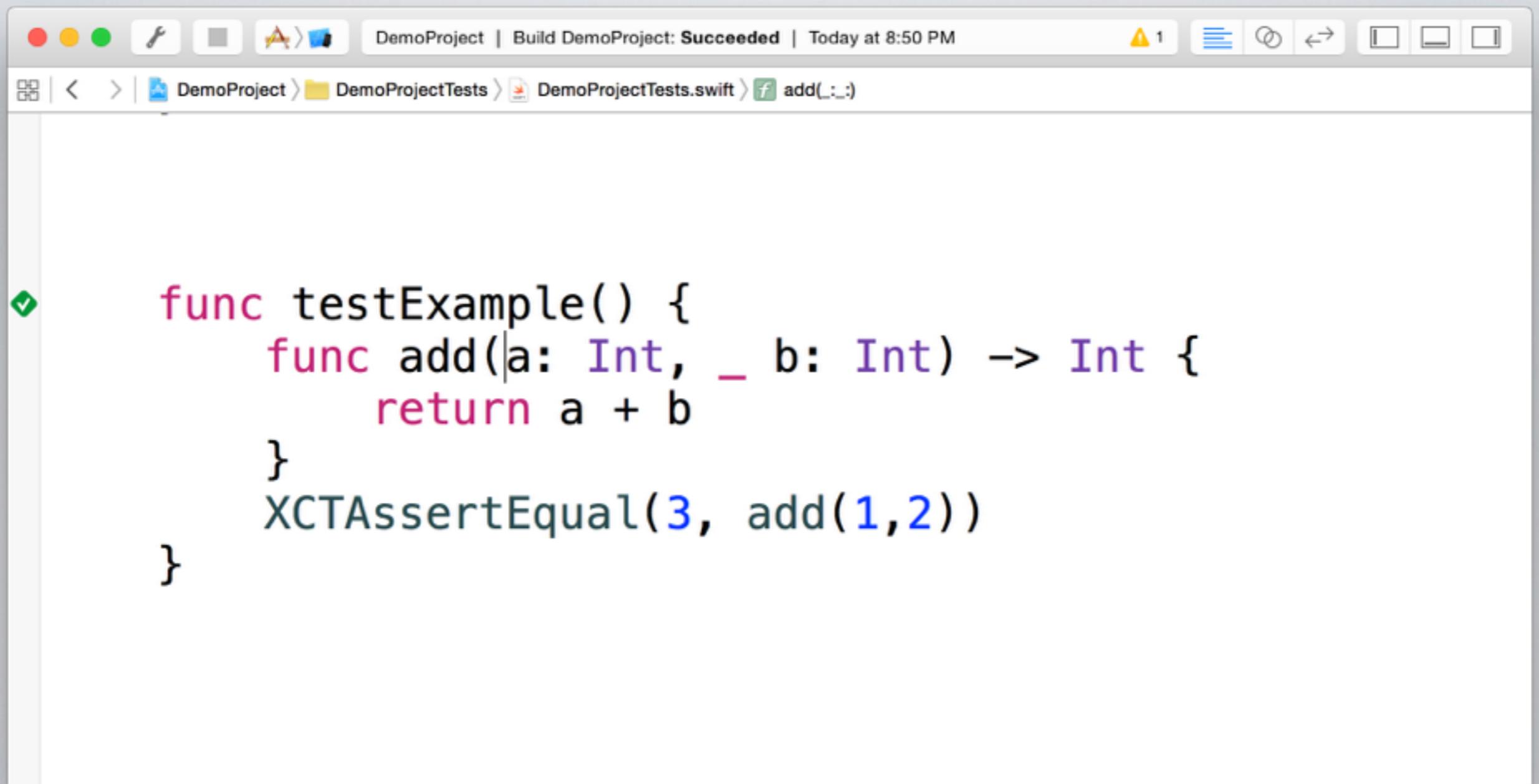
The screenshot shows a web browser window with a light gray header bar. The address bar contains the URL `nshipster.com/xctestcase/`. Below the address bar is a toolbar with standard Mac OS X icons for window control, search, and refresh. The main content area of the browser displays the text "XCCTestCase / XCTestExpectation / measureBlock()" in a large, bold, orange sans-serif font.

Written by Mattt Thompson – July 21st, 2014 (revised)

Although iOS 8 and Swift has garnered the lion's share of attention of the WWDC 2014 announcements, the additions and improvements to testing in Xcode 6 may end up making some of the most profound impact in the long-term.

This week, we'll take a look at `xctest`, the testing framework built into Xcode, as well as

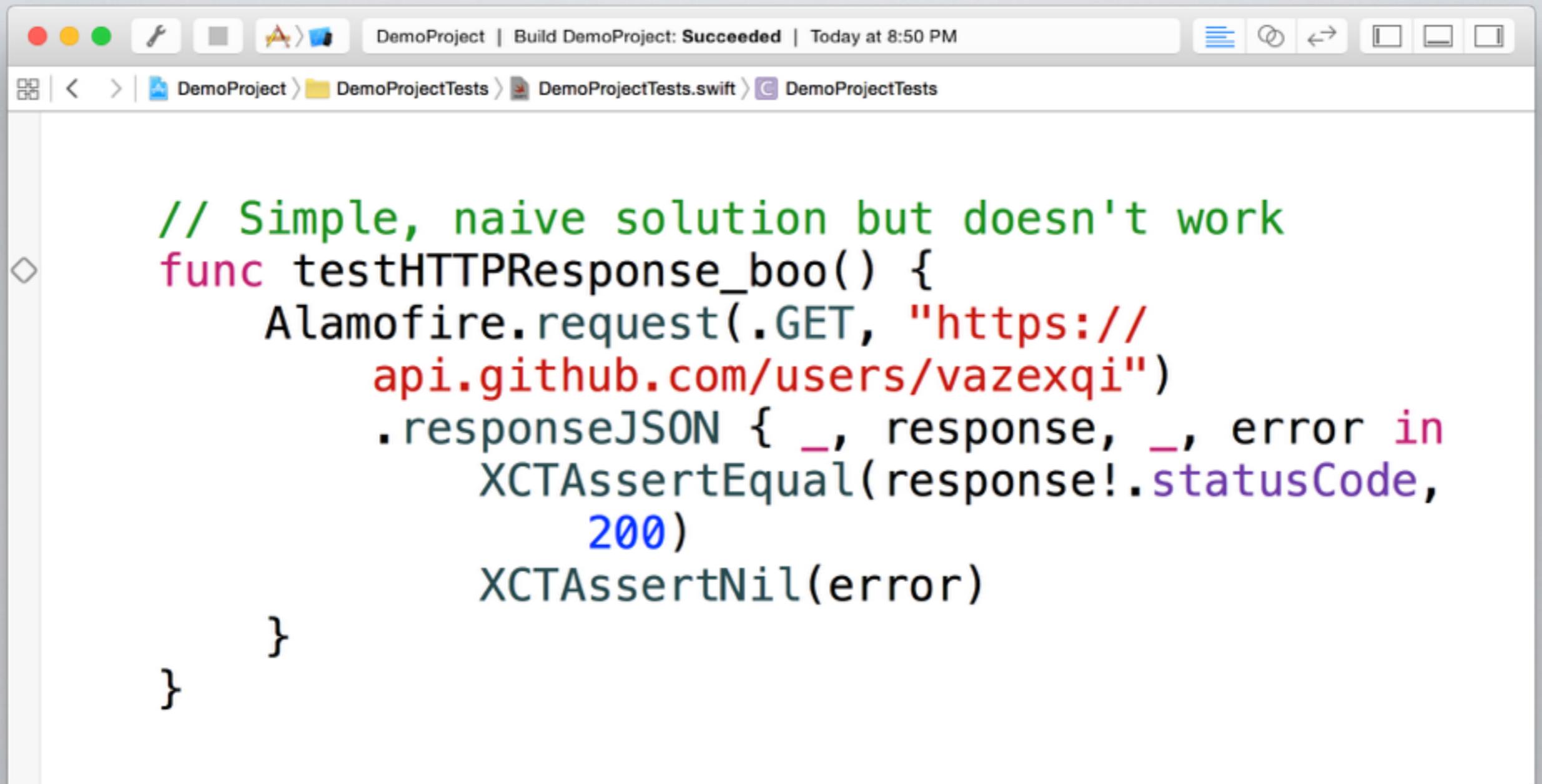
UNIT TESTS



A screenshot of the Xcode IDE interface. The title bar shows "DemoProject | Build DemoProject: Succeeded | Today at 8:50 PM". The navigation bar below the title bar indicates the current file path: "DemoProject > DemoProjectTests > DemoProjectTests.swift > add(_:_:)". The main editor area displays the following Swift code:

```
func testExample() {
    func add(a: Int, _ b: Int) -> Int {
        return a + b
    }
    XCTAssertEqual(3, add(1,2))
}
```

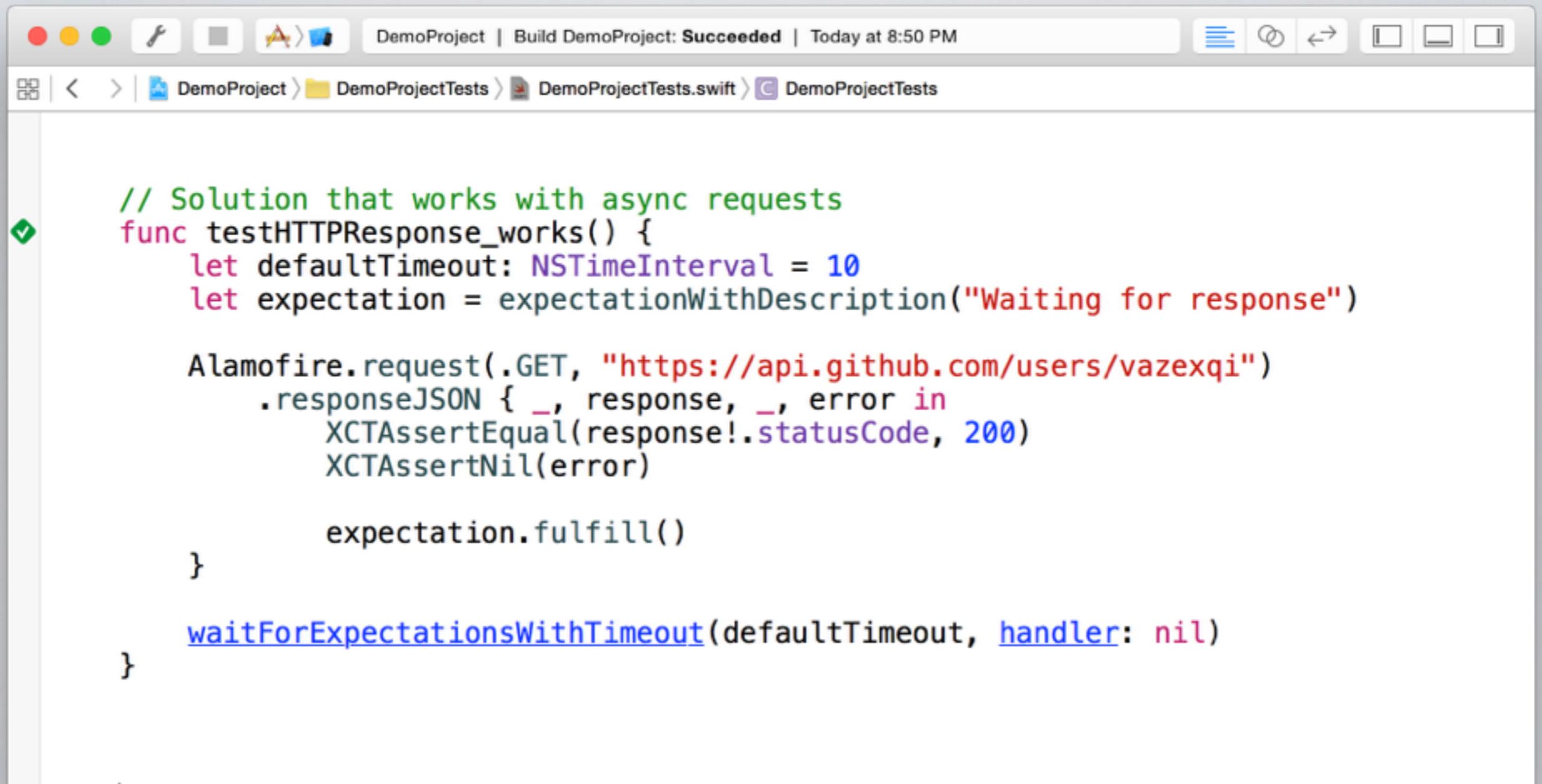
UNIT TESTS



The screenshot shows the Xcode IDE interface. The top bar displays the project name "DemoProject" and a build status of "Build DemoProject: Succeeded | Today at 8:50 PM". The main editor area shows a Swift test file named "DemoProjectTests.swift" under the "DemoProjectTests" target. The code in the file is as follows:

```
// Simple, naive solution but doesn't work
func testHTTPResponse_boo() {
    Alamofire.request(.GET, "https://
        api.github.com/users/vazexqi")
        .responseJSON { _, response, _, error in
            XCTAssertEqual(response!.statusCode,
                           200)
            XCTAssertNil(error)
    }
}
```

UNIT TESTS



The screenshot shows the Xcode interface with a project named "DemoProject". The status bar at the top indicates "Build DemoProject: Succeeded | Today at 8:50 PM". The navigation bar below shows the file structure: "DemoProject > DemoProjectTests > DemoProjectTests.swift". The main editor area contains the following Swift code:

```
// Solution that works with async requests
func testHTTPResponse_works() {
    let defaultTimeout: NSTimeInterval = 10
    let expectation = expectationWithDescription("Waiting for response")

    Alamofire.request(.GET, "https://api.github.com/users/vazexqi")
        .responseJSON { _, response, _, error in
            XCTAssertEqual(response!.statusCode, 200)
            XCTAssertNil(error)

            expectation.fulfill()
    }

    waitForExpectationsWithTimeout(defaultTimeout, handler: nil)
}
```

PROJECT I

- Out today
- Due one week later on Sept 11, 2015
- Push to your GitHub account under Project I and write a note to Piazza.