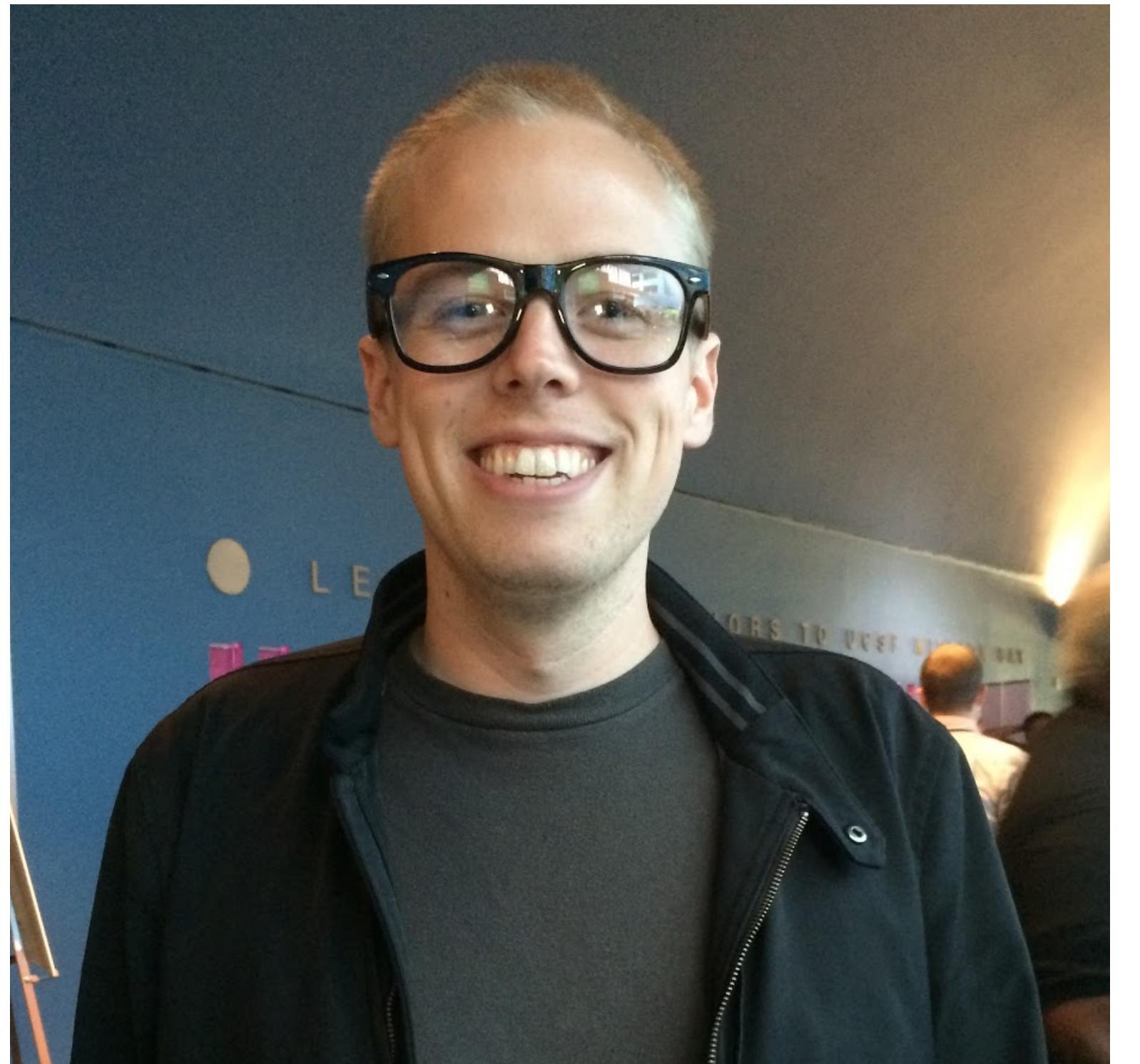


10 Things I learned from building a server side Swift web app

Pithy... I know

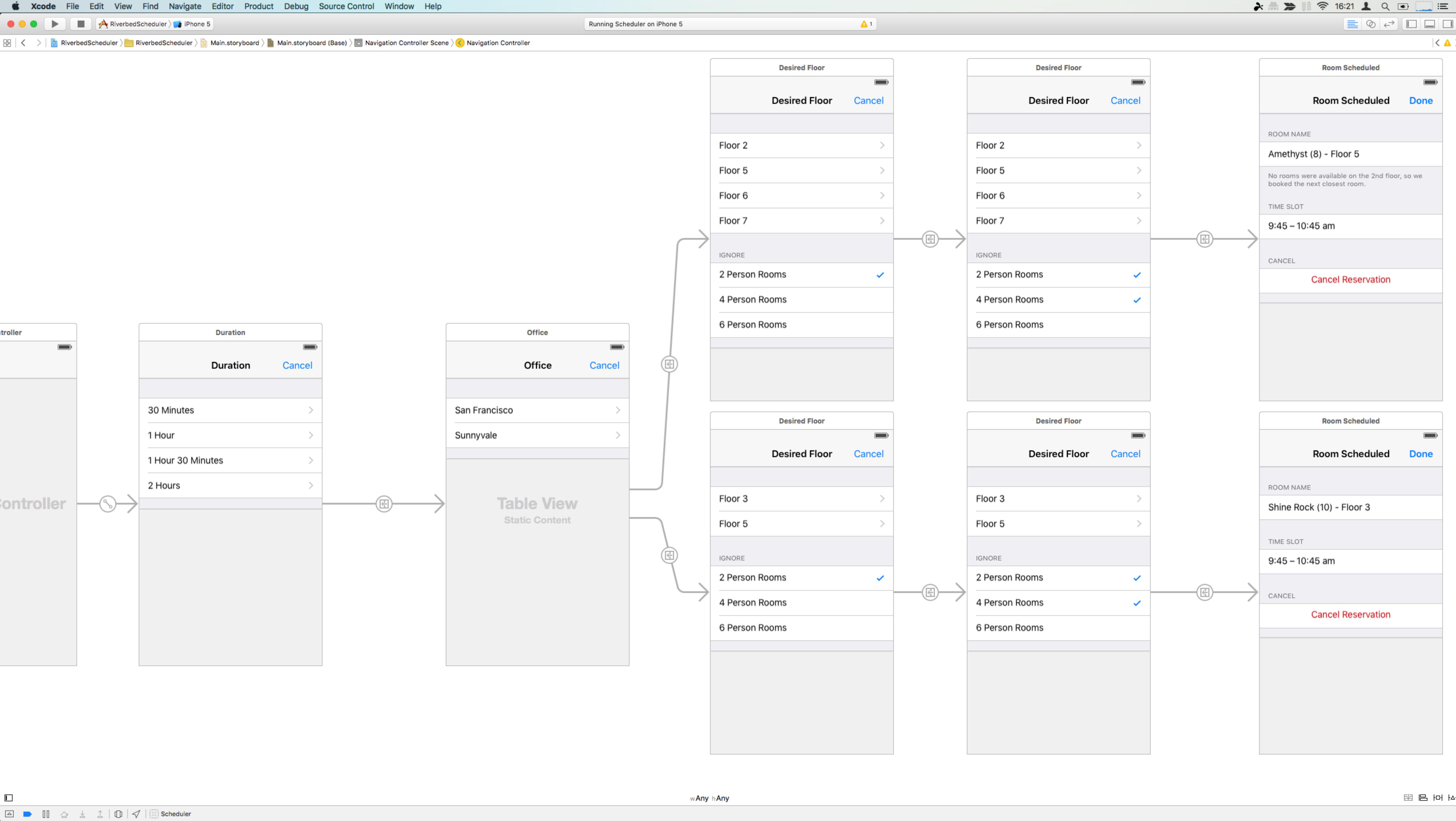
Who Am I?

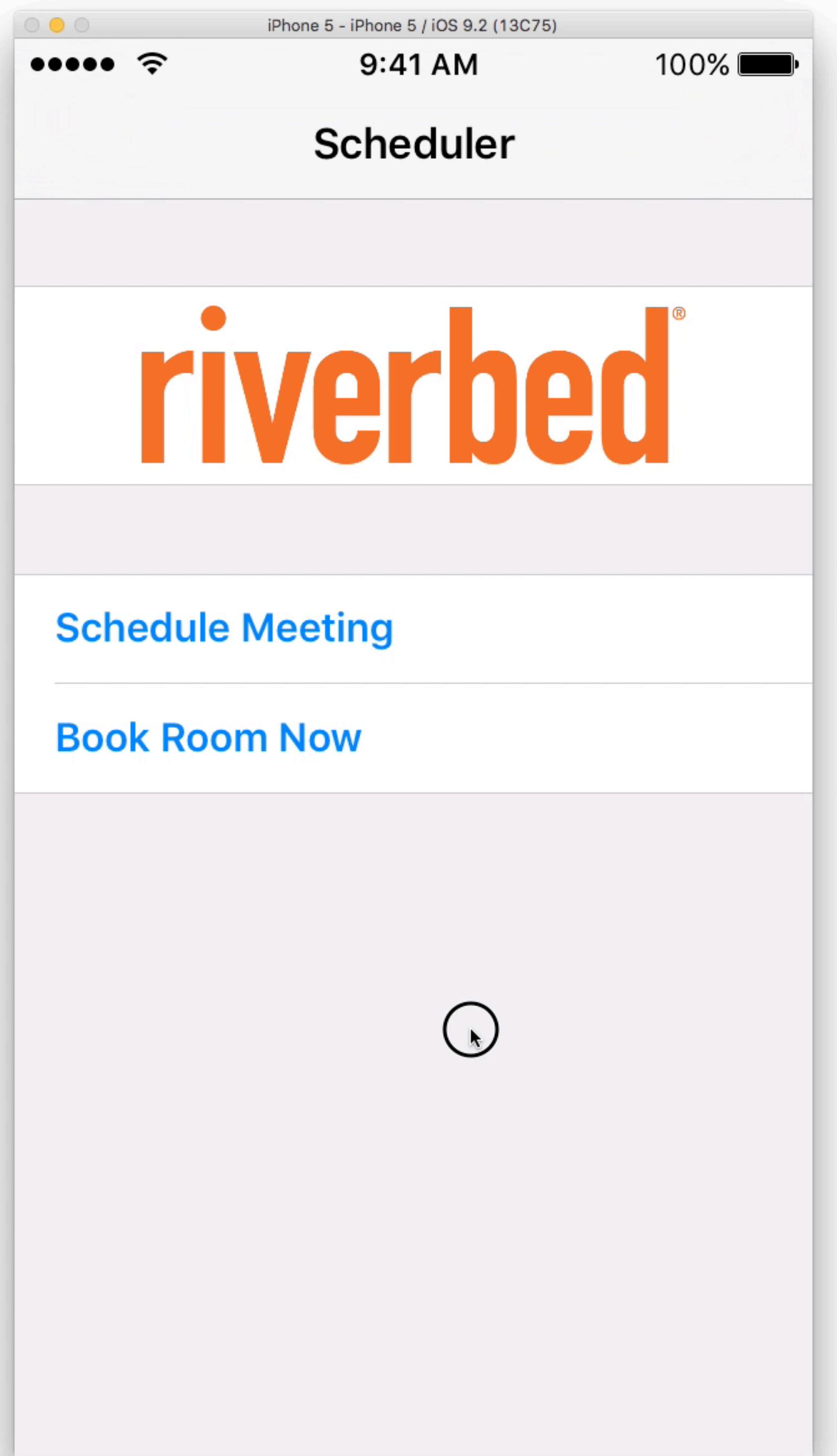
- Jeffrey Bergier
 - UX Designer for Riverbed
 - iOS Developer for Fun
- @jeffburg
- jeffburg.com
- github.com/jeffreybergier/SwiftServerSlugTalk



What is the App?

- A super quick way to reserve a conference room w/o Outlook
- Started as a storyboard prototype





What is the App?

- A super quick way to reserve a conference room w/o Outlook
- Started as a storyboard prototype
- Turned into a Swift playground to test Exchange 'EWS' API

GetUserAvailability request example: Get availability information

The following example of a **GetUserAvailability** operation request shows how to get detailed availability information for two users in the Pacific Time time zone.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
  <soap:Body>
    < GetUserAvailabilityRequest xmlns="http://schemas.microsoft.com/exchange/services/2006/messa
                                 xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
      <t:TimeZone xmlns="http://schemas.microsoft.com/exchange/services/2006/types">
        <Bias>480</Bias>
        <StandardTime>
          <Bias>0</Bias>
          <Time>02:00:00</Time>
          <DayOrder>5</DayOrder>
          <Month>10</Month>
          <DayOfWeek>Sunday</DayOfWeek>
        </StandardTime>
        <DaylightTime>
          <Bias>-60</Bias>
          <Time>02:00:00</Time>
          <DayOrder>1</DayOrder>
          <Month>4</Month>
        </DaylightTime>
      </TimeZone>
    </ GetUserAvailabilityRequest>
  </soap:Body>
</soap:Envelope>
```

GetUserAvailability request example: Get availability information

The following example of a **GetUserAvailability** operation request shows how to get detailed availability information for two users in the Pacific Time time zone.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
    <soap:Body>
        < GetUserAvailabilityRequest xmlns="http://schemas.microsoft.com/exchange/services/2006/messages"
            xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
            <t:TimeZone xmlns="http://schemas.microsoft.com/exchange/services/2006/types">
                <Bias>480</Bias>
                <StandardTime>
                    <Bias>0</Bias>
                    <Time>02:00:00</Time>
                    <DayOrder>5</DayOrder>
                    <Month>10</Month>
                    <DayOfWeek>Sunday</DayOfWeek>
                </StandardTime>
                <DaylightTime>
                    <Bias>-60</Bias>
                    <Time>02:00:00</Time>
                    <DayOrder>1</DayOrder>
                    <Month>4</Month>
                    <DayOfWeek>Sunday</DayOfWeek>
                </DaylightTime>
            </t:TimeZone>
            <MailboxDataArray>
                <t:MailboxData>
                    <t:Email>
                        <t:Address>user1@example.com</t:Address>
                    </t:Email>
                    <t:AttendeeType>Required</t:AttendeeType>
                    <t:ExcludeConflicts>false</t:ExcludeConflicts>
                </t:MailboxData>
                <t:MailboxData>
                    <t:Email>
                        <t:Address>user2@example.com</t:Address>
                    </t:Email>
                    <t:AttendeeType>Required</t:AttendeeType>
                    <t:ExcludeConflicts>false</t:ExcludeConflicts>
                </t:MailboxData>
            </MailboxDataArray>
            <t:FreeBusyViewOptions>
                <t:TimeWindow>
                    <t:StartTime>2006-10-16T00:00:00</t:StartTime>
                    <t:EndTime>2006-10-16T23:59:59</t:EndTime>
                </t:TimeWindow>
                <t:MergedFreeBusyIntervalInMinutes>60</t:MergedFreeBusyIntervalInMinutes>
                <t:RequestedView>DetailedMerged</t:RequestedView>
            </t:FreeBusyViewOptions>
        </ GetUserAvailabilityRequest>
    </soap:Body>
</soap:Envelope>
```

What is the App?

- A super quick way to reserve a conference room w/o Outlook
- Started as a storyboard prototype
- Turned into a Swift playground to test Exchange 'EWS' API
- First written as a Python web app during work Hackathon



[DOWNLOAD](#) [DOCUMENTATION](#) [COMMUNITY](#) [DEVELOPMENT](#)

CherryPy

A Minimalist Python Web Framework

CHERRYPY IS AS EASY AS...

```
import cherrypy

class HelloWorld(object):
    def index(self):
        return "Hello World!"
    index.exposed = True

cherrypy.quickstart(HelloWorld())
```

What is the App?

- A super quick way to reserve a conference room w/o Outlook
- Started as a storyboard prototype
- Turned into a Swift playground to test Exchange 'EWS' API
- First executed as a Python web app during work Hackathon
- Rewritten in Swift and Javascript in my spare time



announcing perfect 2.0

THE NEW **CODE PERFECT** TOOLKIT
FOR SWIFT 3.0

Meet Perfect 2.0 ➤

Demo Time

Why?

- Python works fine
- Any non-Swift solution will have better support





TYPE SAFETY

Why?

- Xcode debugging is awesome
- Xcode autocomplete is awesome
- Swift type safety is awesome
 - I don't know how JS/Ruby/Python people sleep at night
- Safari's Javascript debugging tools are very similar to Xcode

How is it built?

- It's a "modern" web app
- The HTML executes a javascript function that makes a POST request to '/'
- The Swift app responds with a JSON string that represents the UI to display
- The javascript parses the JSON file and converts it into HTMLElements and replaces the Body with them

The Front End

- Bootstrap (because CSS is the worst)
- Javascript (because I wanted to learn it)
- JSON -> UI (because I didn't want my Swift tainted by HTML)
- Cookies store 3 random tokens that decrypt credentials

The Back End

- Router object listens for all POST requests on '/'
- Custom session manager to encrypt / decrypt credentials
- Router parses JSON from request to determine current step and needed information
- Queries Exchange server live to verify login, check for available rooms, and to create new reservation
- Generates new UI JSON to finish the response

Xcode Time

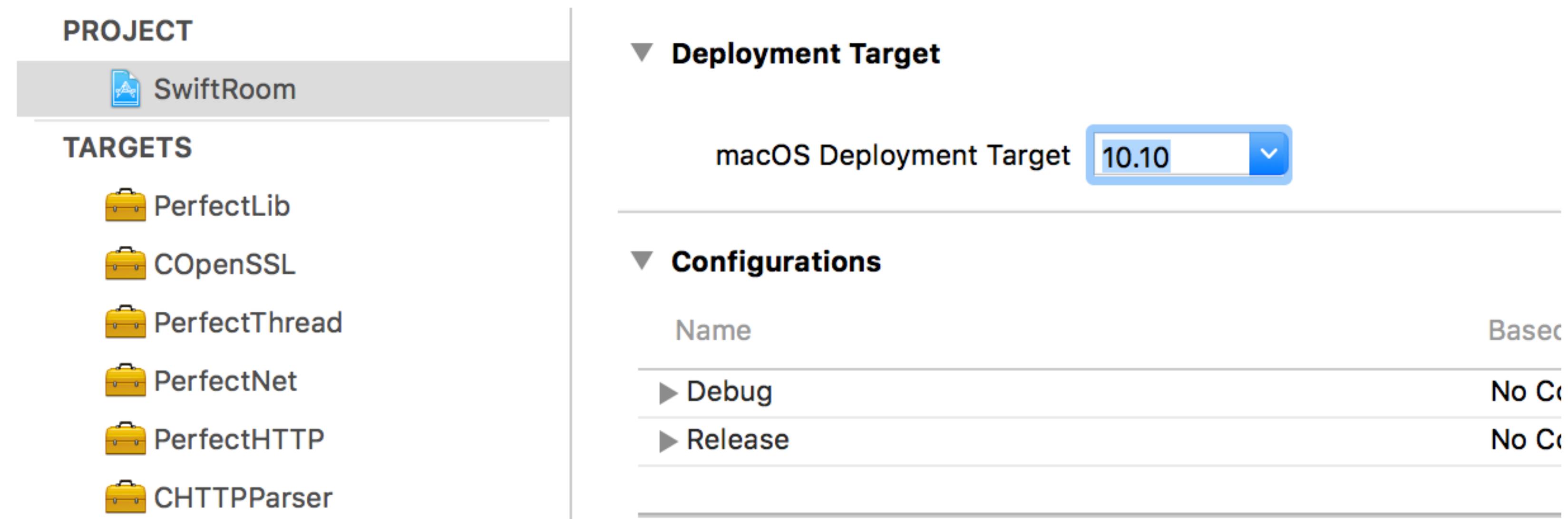
(Well, almost. First we need lesson #1)

1. Forget about the .XcodeProj

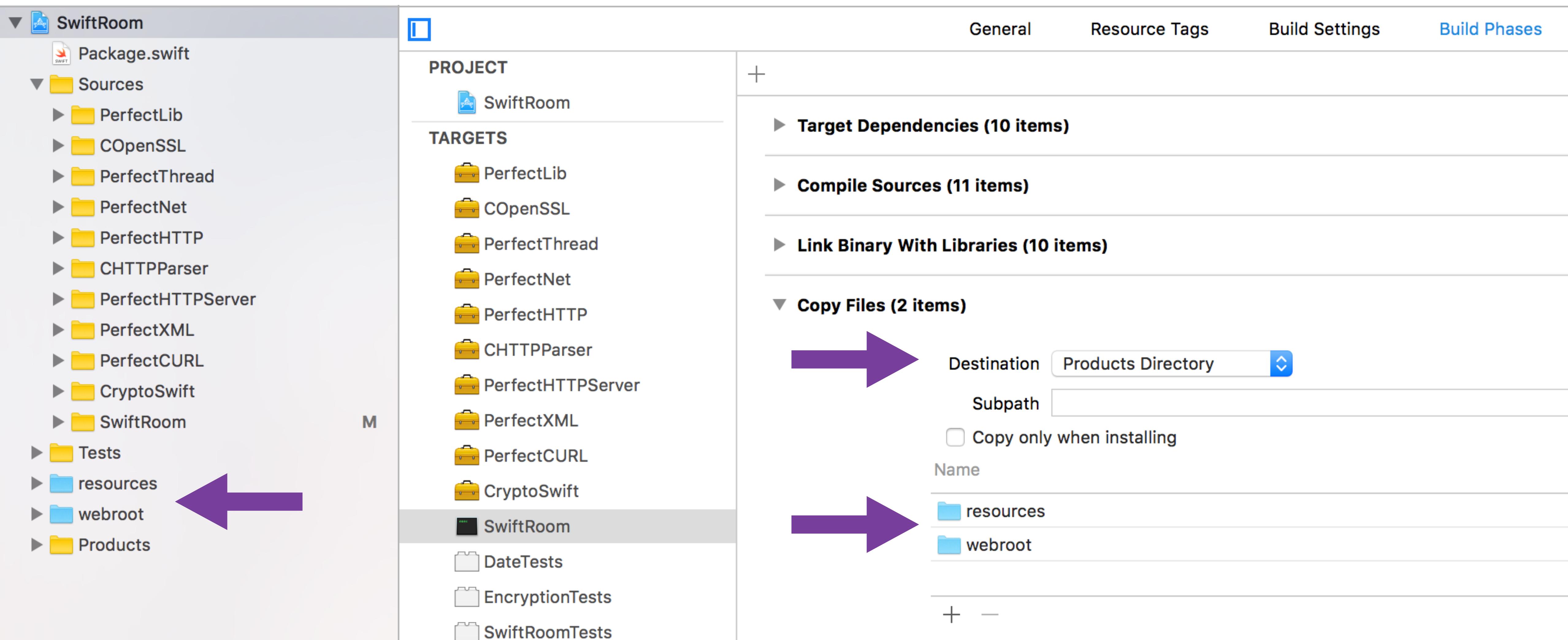
- The Xcode Project file is now something generated for you
- Changes in the Xcode Project are NOT reflected in your project
- Swift Package Manager is the new source of truth
- It revolves around a Package.swift file

XcodeProj Gotcha #1

```
91  
92     init() {  
93         Timer.scheduledTimer(withTimeInterval: 5, repeats: true) { timer in  
94             print("this is cool!")  
95         }  
96     }  
97 }
```



XcodeProj Gotcha #2



SPM Demo

(Init a new project)

(Show SteelReserve package)

(Generate xcode-project)

(Open in Xcode)

Xcode Time

(For Real This Time)

(Show main.swift, router, credentials, curl, xml)

2. Foundation is mostly there

- Foundation exists on Linux ✨
- Simple Foundation types are fine
 - NSDate, NSDateFormatter, NSDateComponents
- Complex ones are not fully implemented*
 - NSURLSession, NSXMLParser

```
extension Date {  
  
    private static func roundedTimeInterval(from date: Date) -> TimeInterval {  
        let dc = Calendar.current.dateComponents([.minute, .second], from: date)  
        let originalMinute = Double(dc.minute ?? 0)  
        let originalSeconds = Double(dc.second ?? 0)  
        let roundTo = 15.0  
        let roundedMinute = round(originalMinute / roundTo) * roundTo  
        let interval = ((roundedMinute - originalMinute) * 60) - originalSeconds  
        return interval  
    }  
  
    mutating func roundMinutes() {  
        let timeInterval = type(of: self).roundedTimeInterval(from: self)  
        self += timeInterval  
    }  
}
```



How to Tell

```
public struct DateComponents : ReferenceConvertible, Hashable, Equatable, _MutableBoxing {
    public typealias ReferenceType = NSDateComponents

    internal var _handle: _MutableHandle<NSDateComponents>

    /// Initialize a `DateComponents`, optionally specifying values for its fields.
    public init(calendar: Calendar? = nil,
                timeZone: TimeZone? = nil,
                era: Int? = nil,
                year: Int? = nil,
                month: Int? = nil,
                day: Int? = nil,
                hour: Int? = nil,
                minute: Int? = nil,
                second: Int? = nil,
                nanosecond: Int? = nil,
                weekday: Int? = nil,
                weekdayOrdinal: Int? = nil,
                quarter: Int? = nil,
                weekOfMonth: Int? = nil,
                weekOfYear: Int? = nil,
                yearForWeekOfYear: Int? = nil) {
        _handle = _MutableHandle(adoptingReference: NSDateComponents())
        if let _calendar = calendar { self.calendar = _calendar }
        if let _timeZone = timeZone { self.timeZone = _timeZone }
        if let _era = era { self.era = _era }
        if let _year = year { self.year = _year }
        if let _month = month { self.month = _month }
        if let _day = day { self.day = _day }
        if let _hour = hour { self.hour = _hour }
        if let _minute = minute { self.minute = _minute }
        if let _second = second { self.second = _second }
        if let _nanosecond = nanosecond { self.nanosecond = _nanosecond }
        if let _weekday = weekday { self.weekday = _weekday }
        if let _weekdayOrdinal = weekdayOrdinal { self.weekdayOrdinal = _weekdayOrdinal }
        if let _quarter = quarter { self.quarter = _quarter }
        if let _weekOfMonth = weekOfMonth { self.weekOfMonth = _weekOfMonth }
        if let _weekOfYear = weekOfYear { self.weekOfYear = _weekOfYear }
        if let _yearForWeekOfYear = yearForWeekOfYear { self.yearForWeekOfYear = _yearForWeekOfYear }
    }
}
```



```
open class URLAuthenticationChallenge : NSObject, NSSecureCoding {

    static public var supportsSecureCoding: Bool {
        return true
    }

    public required init?(coder aDecoder: NSCoder) {
        NSUnimplemented()
    }

    open func encode(with aCoder: NSCoder) {
        NSUnimplemented()
    }

    /*!
     @method initWithProtectionSpace:proposedCredential:previousFailureCount:failureResponse:error:
     @abstract Initialize an authentication challenge
     @param space The NSURLProtectionSpace to use
     @param credential The proposed NSURLCredential for this challenge, or nil
     @param previousFailureCount A count of previous failures attempting access.
     @param response The NSURLResponse for the authentication failure, if applicable, else nil
     @param error The NSError for the authentication failure, if applicable, else nil
     @result An authentication challenge initialized with the specified parameters
    */

    public init(protectionSpace space: URLProtectionSpace, proposedCredential credential: URLCredential?, previousFailureCount: NSUnimplemented())
}
```



But there are subtleties

```
import Foundation

// get date and components
var dc = Calendar.current.dateComponents(
    [.year, .month, .day, .hour, .minute, .second, .calendar, .timeZone],
    from: Date())
)

// get originals and do rounding
let originalMinute = Double(dc.minute ?? 0)
let roundTo = 15.0
let roundedMinute = Int(round(originalMinute / roundTo) * roundTo)

// modify components
dc.minute = roundedMinute
dc.second = 0

// generate new date
let roundedDate = dc.date! // crashes on linux
// fatal error: copy(with:) is not yet implemented: file Foundation/NSCalendar.swift, line 1434
```

3. Test on Linux Often

- At least before every commit, but probably more often
- I've been using Veertu
 - Free, sandboxed Mac App Store app
 - Autodownloads and installs Linux
 - Option to run headless 
- There are Docker based solutions

Veertu Demo

4. JSON is Way Easier

```
import PerfectLib

let data: [String : Any] = [
    "date" : "2016-01-01T12-12-00",
    "name" : "Billy",
    "age" : 22,
    "emails" : [
        "something@something.com",
        "somethingelse@something.com"
    ]
]

let json = try data.jsonEncodedString()
```

4. JSON is Way Easier

- And it needs to be (its a web app)
- No need for NSJSONSerialization any more
- Every String has .jsonDecode()
- Collections and most built-in types have .jsonEncodedString()

5. Random is Hard



```
#if os(Linux)
import Glibc
#else
import Darwin
#endif

for i in 1 ... 5 {
    #if os(Linux)
    let randomNumber = random()
    #else
    let randomNumber = Int(arc4random_uniform(UInt32.max))
    #endif
    print("Round \(i): \(randomNumber)")
}
```

steelreserve@steelreserve-swift ~/RandomDontWork> swift build



5. Random is Hard

- The easiest way is to open /dev/random and read bytes off it
- TurnstileCrypto
 - Relatively light library that has an easy to use random class
- CryptoSwift
 - Massive crypto framework that also generates random numbers this way.

6. Avoid #if os(Linux)

```
func roundingMinutes() -> Date {
    var dc = Calendar.current.dateComponents([.year, .month, .day, .hour, .minute, .calendar, .timeZone], from: self)
    let originalMinute = dc.minute ?? 0
    let roundedMinute = (((originalMinute - 8) / 15) * 15)
    dc.minute = roundedMinute
    dc.second = 0

#if os(Linux)
    // DateComponents copy(with: zone) is NSUnimplemented on Linux
    // Lets build a date with strings :
    let df = Date.linuxDateRoundingFormatter
    let year = dc.year ?? 0
    let month = dc.month ?? 1
    let day = dc.day ?? 1
    let hour = dc.hour ?? 0
    let minute = dc.minute ?? 0
    let second = dc.second ?? 0
    let timeZone = dc.timeZone ?? TimeZone(secondsFromGMT: 0)!
    let dateString = "\(year.leftPaddedString(totalCharacters: 4))-\(month.leftPaddedString())-\(day.leftPaddedString())T\(hou
    let roundedDate = df.date(from: dateString)!)
    return roundedDate
#else
    let roundedDate = dc.date! // crashes on Linux
    return roundedDate
#endif
```

6. Avoid #if os(Linux)

- You lose all help from Xcode
 - No autocomplete, syntax highlighting, compiler errors
- Won't run into compile errors or bugs until running on Linux
- Sometimes its needed, but make sure you test it heavily with Linux specific tests.

7. Perfect is the new Foundation

 **PerfectlySoft Inc.**
Server-side Swift
 Newmarket, Ontario, Ca...  <http://perfect.org>  info@perfect.org

 **Repositories**  People 9

 Filters  Find a repository...

PerfectDocs

Reference and documentation for Perfect Server Side Swift

Updated 2 hours ago

HTML ★ 92 ⚡ 27

Perfect-HTTP

Base HTTP Support for Perfect Server Side Swift

Updated a day ago

Swift ★ 3 ⚡ 10

Perfect-MySQL

A stand-alone Swift wrapper around the MySQL client library, enabling access to MySQL servers.

Swift ★ 23 ⚡ 13

Perfect-HTTPServer

HTTP 1.1 Server for Perfect Server Side Swift

Updated 2 days ago

Swift ★ 6 ⚡ 5

Perfect-Turnstile-PostgreSQL

This project integrates Stormpath's Turnstile authentication system into a single package with Perfect, and a PostgreSQL ORM.

Updated 3 days ago

Swift ★ 0 ⚡ 0

Perfect-Turnstile-SQLite

This project integrates Stormpath's Turnstile authentication system into a single package with Perfect, and a SQLite ORM.

Updated 3 days ago

Swift ★ 2 ⚡ 0

Perfect

Server-side Swift. The Perfect library, application server, connectors and example apps. (For mobile back-end development, website and web app development, and more...)

Updated 3 days ago

Swift ★ 8,950 ⚡ 615

7. Perfect is the new Foundation

- If Foundation is letting you down, browse the repositories for Perfect (or whatever framework you chose)
- I spent a lot of time debugging NSXMLParser when there was a Perfect-XML library that was easier and better.
- Same goes with NSURLSession. Perfect has Perfect-CURL that is a light wrapper around LibCURL.
 - Its not pretty, but it works.

8. Threading is a little “different”

```
71  
92     init() {  
93         Timer.scheduledTimer(withTimeInterval: 5, repeats: true) { timer in  
94             print("This is so awesome!")  
95         }.fire()  
96     }  
97 }
```



This is so awesome!

[INFO] Starting HTTP server on 127.0.0.1:8181 with document root ./webroot

8. Threading is a little “different”

```
92     private func printAwesome() {  
93         print("This is awesome!")  
94         let queue = DispatchQueue.global(qos: .background)
```

```
steelreserve@steelreserve-swift ~/$/swiftroom> swift build
```

```
Compile Swift Module 'SwiftRoom' (11 sources)
```

```
/home/steelreserve/SwiftServer/swiftroom/Sources/CookieCredentialsManager.swift:94:21:
```

```
error: use of unresolved identifier 'DispatchQueue'
```

```
    let queue = DispatchQueue.global(qos: .background)  
    ^~~~~~ me()
```

```
<unknown>:0: error: build had 1 command failures
```

```
error: exit(1): /usr/local/swift/swift-3.0.1-PREVIEW-3-ubuntu15.10/usr/bin/swift-build-  
tool -f /home/steelreserve/SwiftServer/swiftroom/.build/debug.yaml
```

```
This is awesome!  
[INFO] Starting HTTP server on 127.0.0.1:8181 with document root ./webroot  
This is awesome!  
This is awesome!  
This is awesome!  
Program ended with exit code: 9
```

8. Threading is a little “different”

```
92     private func printAwesome() {
93         print("this is awesome")
94         Threading.sleep(seconds: 5)
95         self.printAwesome()
96     }
97
98     init() {
99         Threading.getQueue(name: "SteelReserve::CredentialCheckingThread", type: .concurrent).dispatch {
100             self.printAwesome()
101         }
102     }
103 }
```



```
this is awesome
[INFO] Starting HTTP server on 127.0.0.1:8181 with document root ./webroot
this is awesome
```

8. Threading is a little “different”

- No GCD on Linux
 - NSTimer doesn't appear to work
 - No Dispatch_After
- Import PerfectThread library to make Queues
 - Call sleep on them if you want to simulate a timer
 - Don't sleep a Queue you did not create! 

9. Force Unwrapping is Bad

```
private let allRooms: [RoomJSON] = {
    let jsonFile = File("./resources/webmail.riverbed.com.json")
    let jsonFileString = try! jsonFile.readString()
    let json = try! jsonFileString.jsonDecode()
    let array = json as! [Any]
    let rooms = RoomJSON.array(from: array)
    return rooms
}()
```

9. Force Unwrapping is Bad

- In Cocoa/Touch development, force unwrapping nil crashes one person's app
 - In a Swift server, ALL your users rely on this app not crashing
 - This changes the calculus quite a bit
- I made the decision that only things that could crash on Server startup are allowed to crash
 - Everything else must fail "gracefully"

10. Its a Ton of Fun!

Thank You!

Q&A

@jeffburg

github.com/jeffreybergier/SwiftServerSlugTalk