

Server-Side Swift

HTTP API's and web pages



Christopher G. Prince
Principal iOS Developer, roster technologies,
and SyncServerII open-source developer
chris@SpasticMuffin.biz

Outline

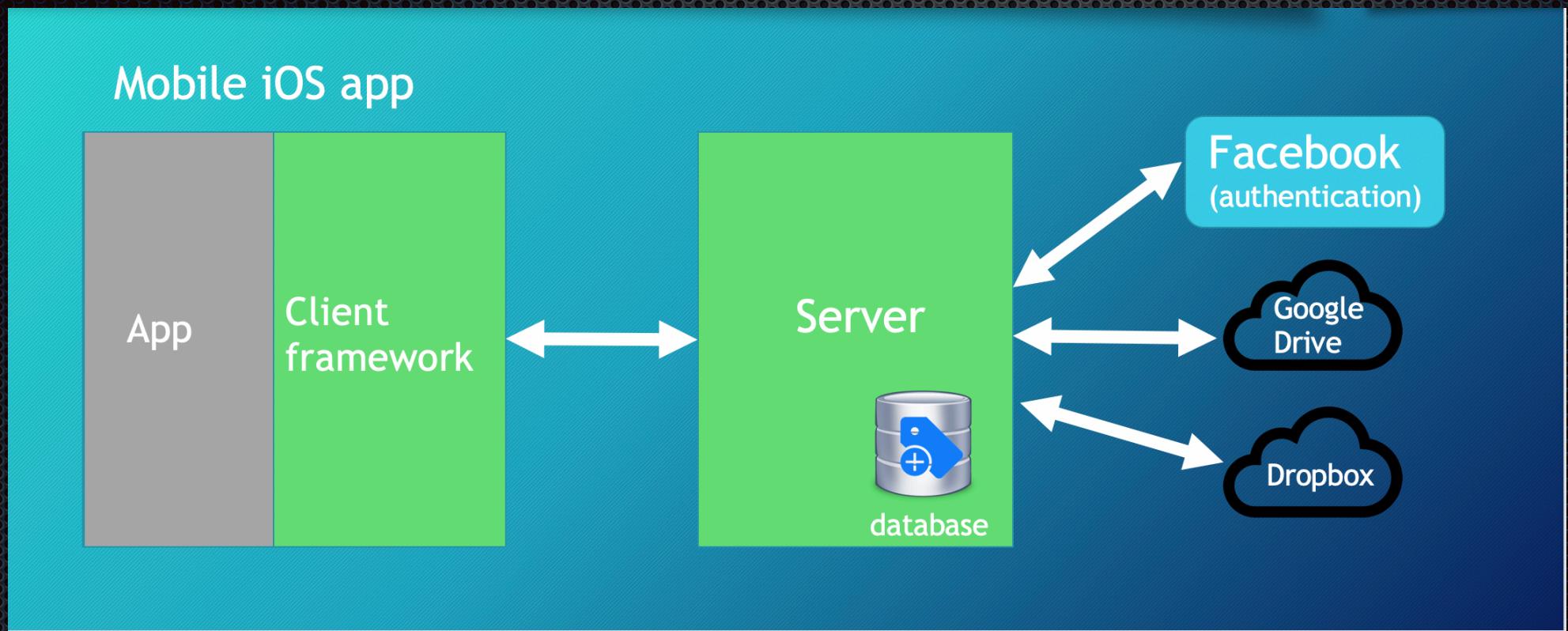
- SyncServerII and SharedImages— open source project
- Basics: Swift package manager and Xcode
- Make your own Swift server for API endpoints
- Make your own Swift server for web pages

SyncServerII and SharedImages

- SyncServerII
 - User-cloud data sync and safe-sharing
 - Uses IBM Kitura framework
- SharedImages
 - Example app using SyncServerII
 - Planned for Apple app store release late this year, or early next year

<https://github.com/crspybits/SyncServerII>

<https://github.com/crspybits/SharedImages>



Architecture: SharedImages + Server

Basics

- Create a Swift package
- Build & run that package at command line
- Build & run that package in Xcode

Create a Swift package

- ❖ swift package init
 - ❖ Executable or library packages
1. Make a directory
 2. Create the package
 3. Build & run

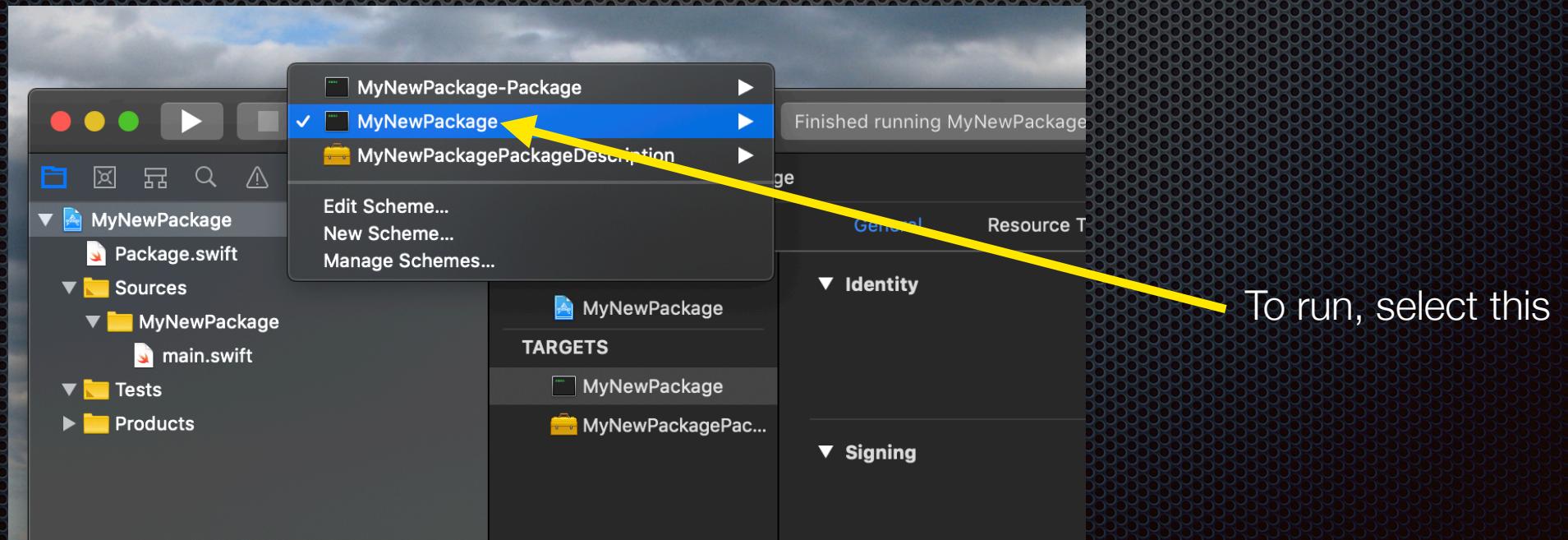
```
MyNewPackage — bash — 72x13
MacBook-Pro-2:Desktop chris$ mkdir MyNewPackage
MacBook-Pro-2:Desktop chris$ cd MyNewPackage/
MacBook-Pro-2:MyNewPackage chris$ swift package init --type executable
Creating executable package: MyNewPackage
Creating Package.swift
```



```
MyNewPackage — bash — 87x8
MacBook-Pro-2:MyNewPackage chris$ swift build
Compile Swift Module 'MyNewPackage' (1 sources)
Linking ./build/x86_64-apple-macosx10.10/debug/MyNewPackage
MacBook-Pro-2:MyNewPackage chris$ ./build/x86_64-apple-macosx10.10/debug/MyNewPackage
Hello, world!
MacBook-Pro-2:MyNewPackage chris$
```

Using Xcode

- `swift package generate-xcodeproj`
 - Note that every time you regenerate it, you lose settings in Xcode

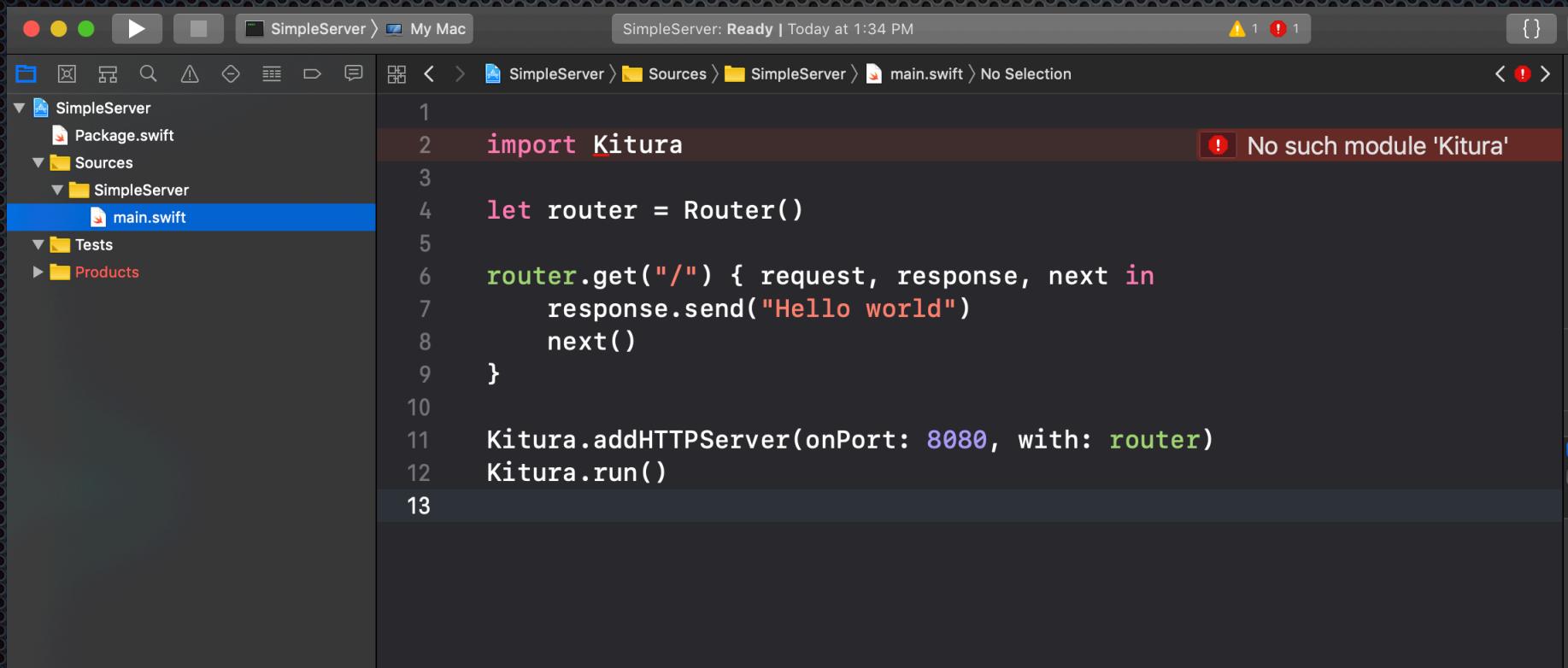


IBM Kitura

- <https://github.com/IBM-Swift/Kitura>
- “Embedded” web server
 - Something like Node.js (not like Apache/PHP)
- Multi-threaded web server
- Other frameworks available too

Let's make an endpoint

- e.g., `http://localhost:8080>HelloWorld`
- Drop in sample code from <https://www.kitura.io>



```
1
2 import Kitura
3
4 let router = Router()
5
6 router.get("/") { request, response, next in
7     response.send("Hello world")
8     next()
9 }
10
11 Kitura.addHTTPServer(onPort: 8080, with: router)
12 Kitura.run()
13
```

(<https://github.com/crspybits/SimpleServer.git>)

Dependencies

- Changes needed for Package.swift
 - A package dependency
 - A target dependency
- Secret sauce: regenerate Xcode project
 - `swift package generate-xcodeproj`
- You may also need to remove the .build directory and try again

Let's serve a web page

- Going to use Kitura's "Stencil" templates
 - [https://github.com/IBM-Swift/Kitura-
StencilTemplateEngine](https://github.com/IBM-Swift/Kitura-StencilTemplateEngine)
 - [http://masteringswift.blogspot.com/2017/02/getting-
started-with-kitura-stencil.html](http://masteringswift.blogspot.com/2017/02/getting-started-with-kitura-stencil.html)
- Same basic server as before, but with an additional Swift package, and a new endpoint

[\(https://github.com/crsypybits/SimpleWeb.git\)](https://github.com/crsypybits/SimpleWeb.git)

Serve pages from Views directory

```
<ServerRepositoryName>
├── Package.swift
├── Sources
│   └── Application
│       └── Application.swift
└── Views
    └── Example.stencil
```

(From <https://github.com/IBM-Swift/Kitura-StencilTemplateEngine>)

See also

- <https://www.kitura.io>
 - <https://www.kitura.io/guides/gettingstarted.html>
 - <https://github.com/IBM/swift-kitura-helloworld>
 - <https://developer.ibm.com/swift/2017/10/30/codable-routing/>
- <https://swift.org/package-manager/>
 - <https://github.com/apple/swift-package-manager/tree/master/Documentation>
- <https://www.ralfebert.de/ios-examples/xcode/ios-dependency-management-with-swift-package-manager/>