Server-Side Swift with Vapor

Prepared for CocoaHeads by
Jeff Meador
May 25, 2017

Overview

01 Vapor

02 Installation

03 Initialization

04 Xcode

05 Heroku

What is Vapor?

I thought we were talking about Swift…

Vapor

Vapor is the most used web framework for Swift. It provides a beautifully expressive and easy to use foundation for your next website or API.

https://github.com/vapor/vapor

https://vapor.codes

Installation

This section will be short

Installation

REQUIREMENTS

- MacOS (you can install on Ubuntu as well, different instructions)
- Xcode 8.3 with Swift 3.1 (from iTunes)
- Homebrew (https://brew.sh)
- Add Homebrew Tap (in Terminal)

```
brew tap vapor/homebrew-tap
brew update
```

- Install (in Terminal)

```
brew install vapor
```

Initialization

Create a Project

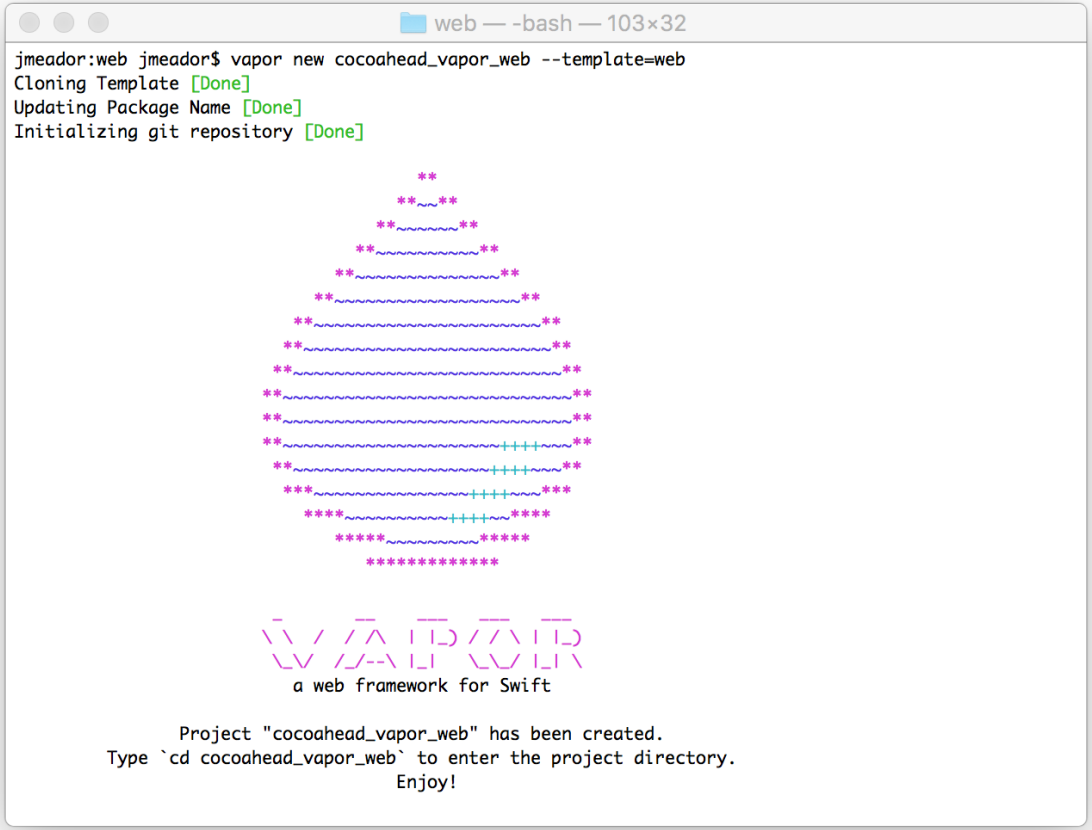
Initialization

- Two Templates for New Projects
  - Web
  - API
- Create Project In Terminal

```
vapor new <name> [--template]
```

Initialization

Web



Initialization

Web

Familiar with web frameworks??

```
├── Config
|   ├── app.json
|   ├── crypto.json
|   ├── droplet.json
|   └── server.json
├── Package.pins
├── Package.swift
├── Public
|   ├── images
|   |   └── it-works.png
|   └── styles
|       └── app.css
├── README.md
├── Resources
|   └── Views
|       ├── base.leaf
|       ├── hello.leaf
|       └── welcome.leaf
├── Sources
|   ├── App
|   |   ├── Config+Setup.swift
|   |   ├── Controllers
|   |   |   └── HelloController.swift
|   |   ├── Droplet+Setup.swift
|   |   ├── Models
|   |   └── Routes.swift
|   └── Run
|       └── main.swift
├── Tests
|   ├── AppTests
|   |   ├── PostControllerTests.swift
|   |   ├── RouteTests.swift
|   |   └── Utilities.swift
|   └── LinuxMain.swift
├── circle.yml
└── license
```

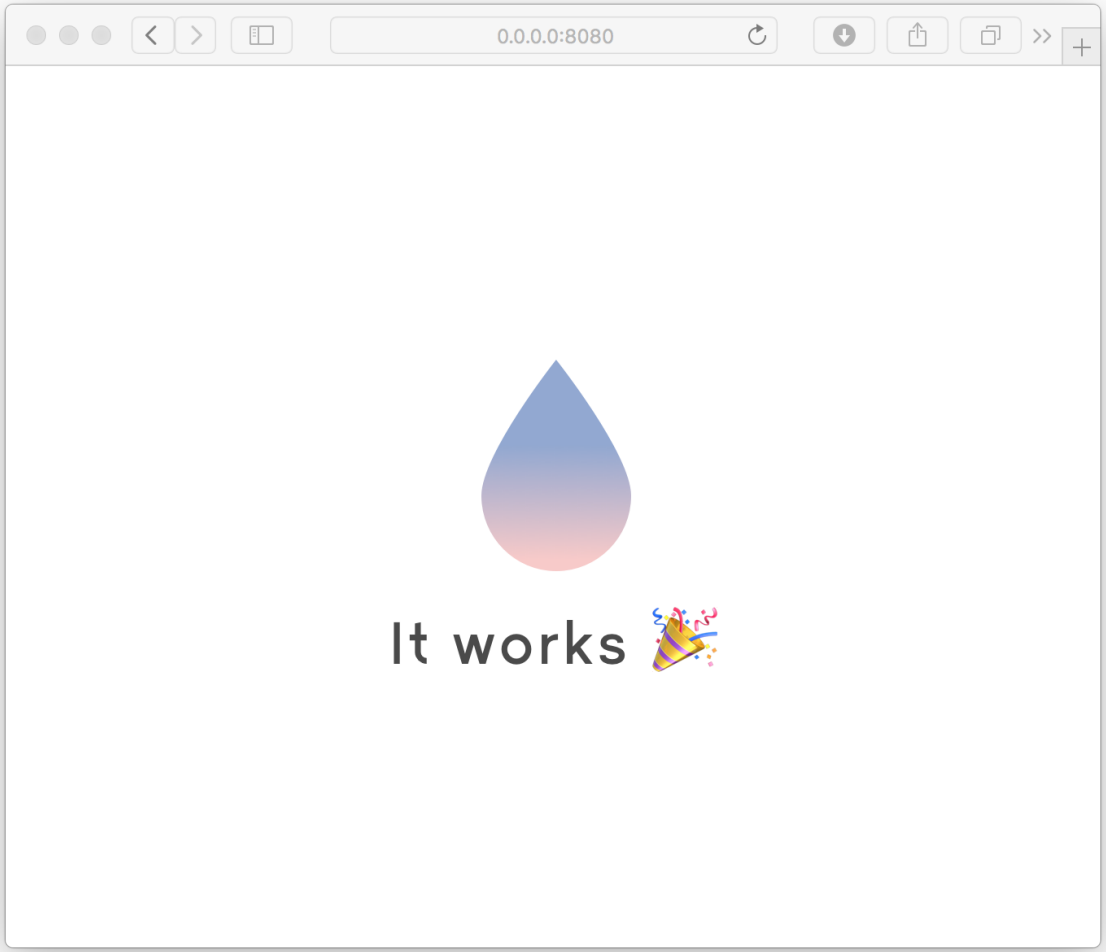
Initialization

Web

```
jmeador:cocoahead_vapor_web jmeador$ vapor build

No .build folder, fetch may take a while...
Fetching Dependencies [Done]
Building Project [Done]

jmeador:cocoahead_vapor_web jmeador$ vapor run

Running cocoahead_vapor_web ...
The current hash key "0000000000000000" is not secure.
Update hash.key in Config/crypto.json before using in production.
Use `openssl rand -base64 <length>` to generate a random string.
The current cipher key "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=" is not secure.
Update cipher.key in Config/crypto.json before using in production.
Use `openssl rand -base64 32` to generate a random string.
No command supplied, defaulting to serve...
Starting server on 0.0.0.0:8080
GET /
GET /styles/app.css
GET /images/it-works.png
GET /favicon.ico
```

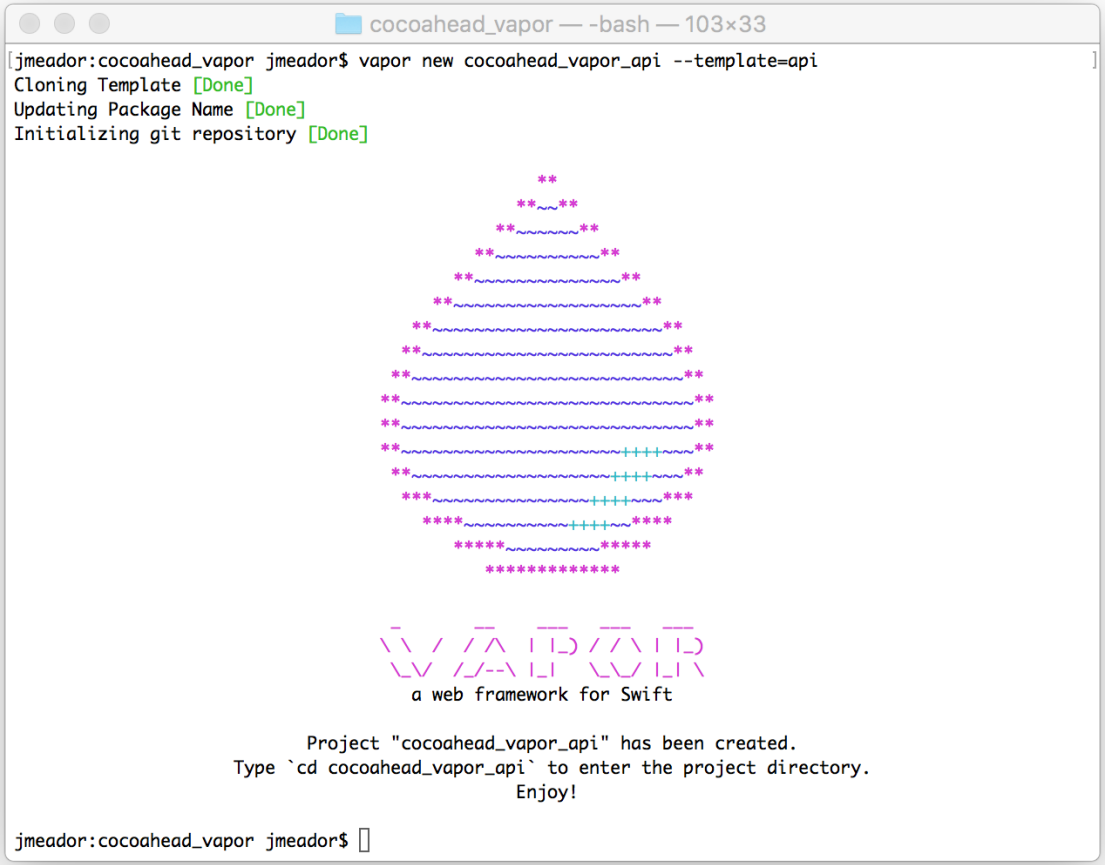
Initialization

Web



Initialization

API



Initialization

API

Familiar with web frameworks??

```
├── Config
|   ├── app.json
|   ├── crypto.json
|   ├── droplet.json
|   ├── fluent.json
|   └── server.json
├── Package.pins
├── Package.swift
├── Public
├── README.md
├── Sources
|   ├── App
|   |   ├── Config+Setup.swift
|   |   ├── Controllers
|   |   |   └── PostController.swift
|   |   ├── Droplet+Setup.swift
|   |   ├── Models
|   |   |   └── Post.swift
|   |   └── Routes.swift
|   └── Run
|       └── main.swift
├── Tests
|   ├── AppTests
|   |   ├── PostControllerTests.swift
|   |   ├── RouteTests.swift
|   |   └── Utilities.swift
|   └── LinuxMain.swift
├── circle.yml
└── license
```

Initialization

Web (recap)

Familiar with web frameworks??

```
├── Config
|   ├── app.json
|   ├── crypto.json
|   ├── droplet.json
|   └── server.json
├── Package.pins
├── Package.swift
├── Public
|   ├── images
|   |   └── it-works.png
|   └── styles
|       └── app.css
├── README.md
├── Resources
|   └── Views
|       ├── base.leaf
|       ├── hello.leaf
|       └── welcome.leaf
├── Sources
|   ├── App
|   |   ├── Config+Setup.swift
|   |   ├── Controllers
|   |   |   └── HelloController.swift
|   |   ├── Droplet+Setup.swift
|   |   ├── Models
|   |   └── Routes.swift
|   └── Run
|       └── main.swift
├── Tests
|   ├── AppTests
|   |   ├── PostControllerTests.swift
|   |   ├── RouteTests.swift
|   |   └── Utilities.swift
|   └── LinuxMain.swift
├── circle.yml
└── license
```

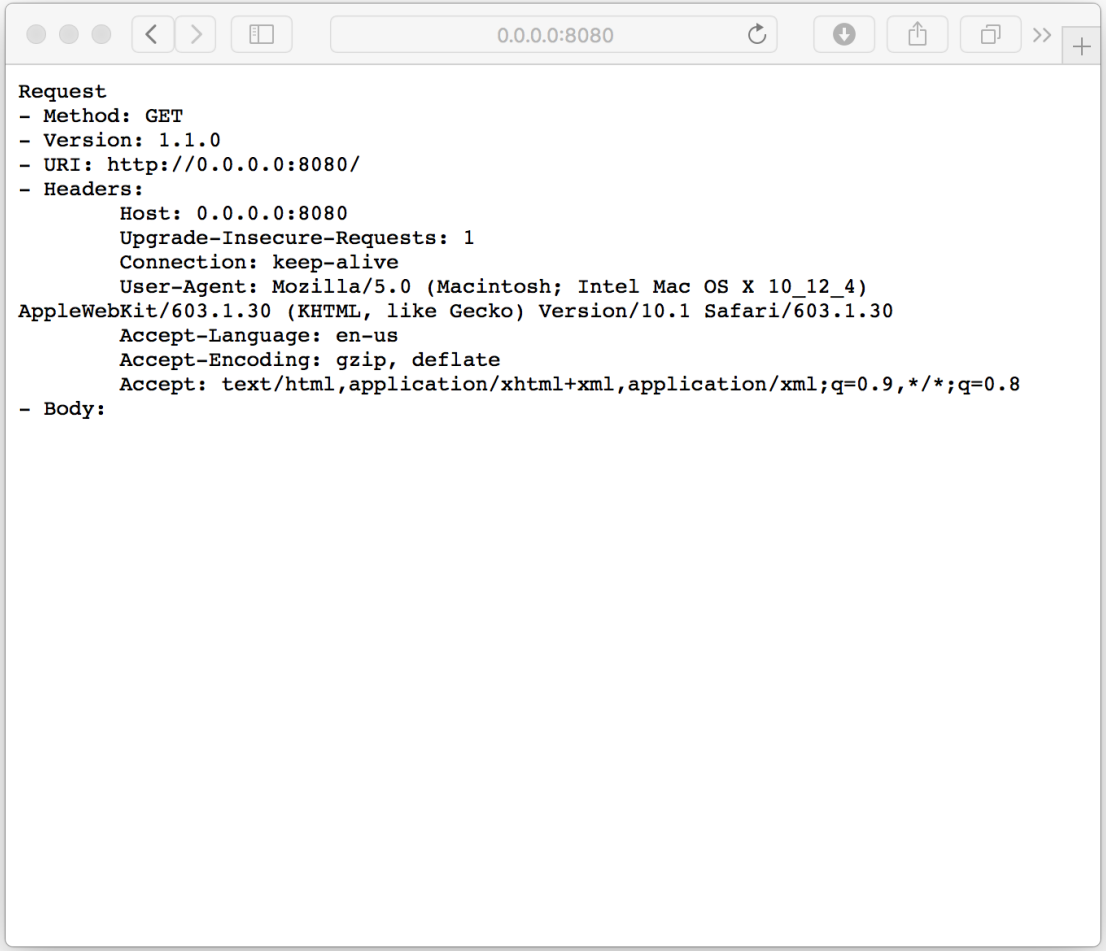
Initialization

API

```
jmeador:cocoahead_vapor_api jmeador$ vapor build

No .build folder, fetch may take a while...
Fetching Dependencies [Done]
Building Project [Done]

jmeador:cocoahead_vapor_api jmeador$ vapor run

Running cocoahead_vapor_api ...
The current hash key "0000000000000000" is not secure.
Update hash.key in Config/crypto.json before using in production.
Use `openssl rand -base64 <length>` to generate a random string.
The current cipher key "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=" is not secure.
Update cipher.key in Config/crypto.json before using in production.
Use `openssl rand -base64 32` to generate a random string.
Database prepared
No command supplied, defaulting to serve...
Starting server on 0.0.0.0:8080
```

Initialization

API



Xcode

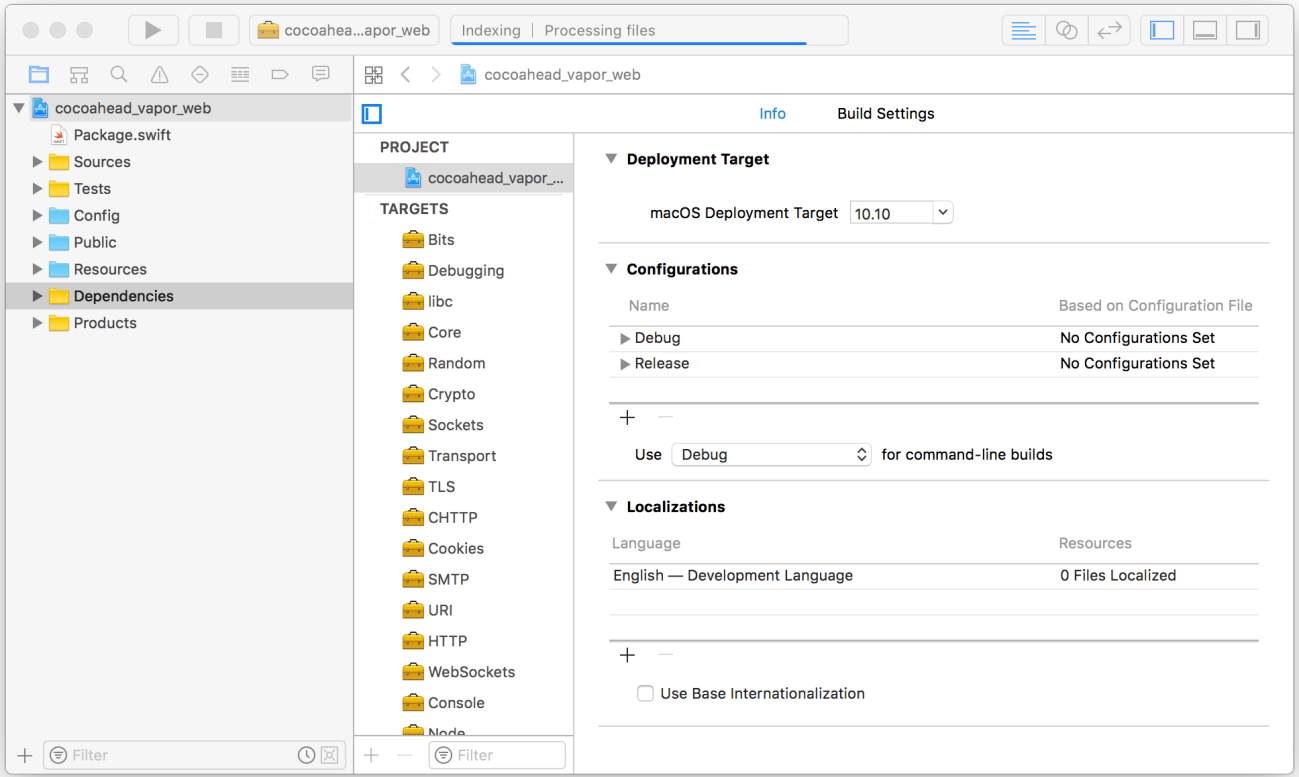
You're welcome to use Atom, but Xcode is here for you if you need him.

Xcode

Running Vapor Within Xcode

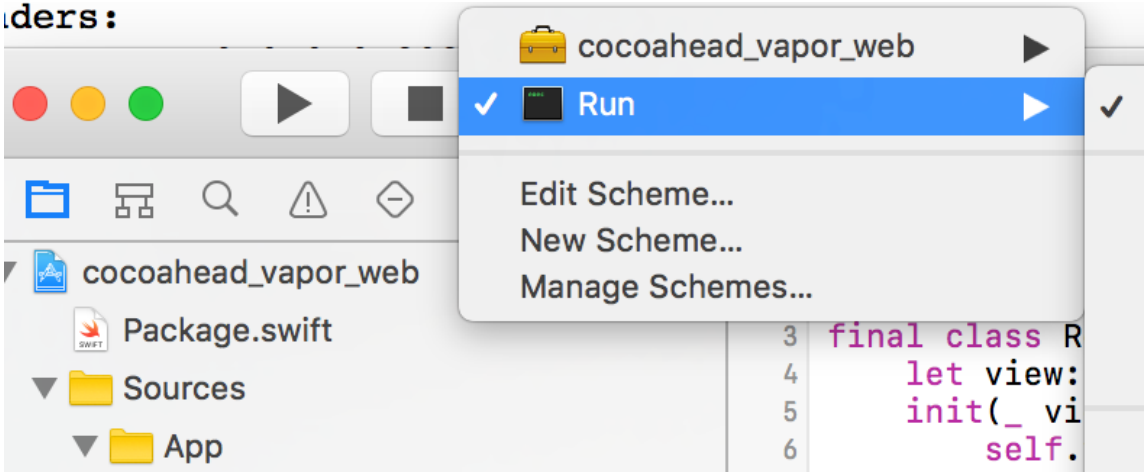
```
jmeador:cocoahead_vapor_web jmeador$ vapor xcode -y

Generating Xcode Project [Done]
Select the `Run` scheme to run.
Open Xcode project?
y/n> yes
Opening Xcode project...
```

Xcode


cocoahea...apor_web
Indexing | Processing files
cocoahead_vapor_web
cocoahead_vapor_web
Package.swift
Sources
Tests
Config
Public
Resources
Dependencies
Products
Info
Build Settings
PROJECT
cocoahead_vapor_...
TARGETS
Bits
Debugging
libc
Core
Random
Crypto
Sockets
Transport
TLS
CHTTP
Cookies
SMTP
URI
HTTP
WebSockets
Console
Deployment Target
macOS Deployment Target
10.10
Configurations
Name
Based on Configuration File
Debug
No Configurations Set
Release
No Configurations Set
Use
Debug
for command-line builds
Localizations
Language
Resources
English — Development Language
0 Files Localized
Use Base Internationalization
Filter
Filter

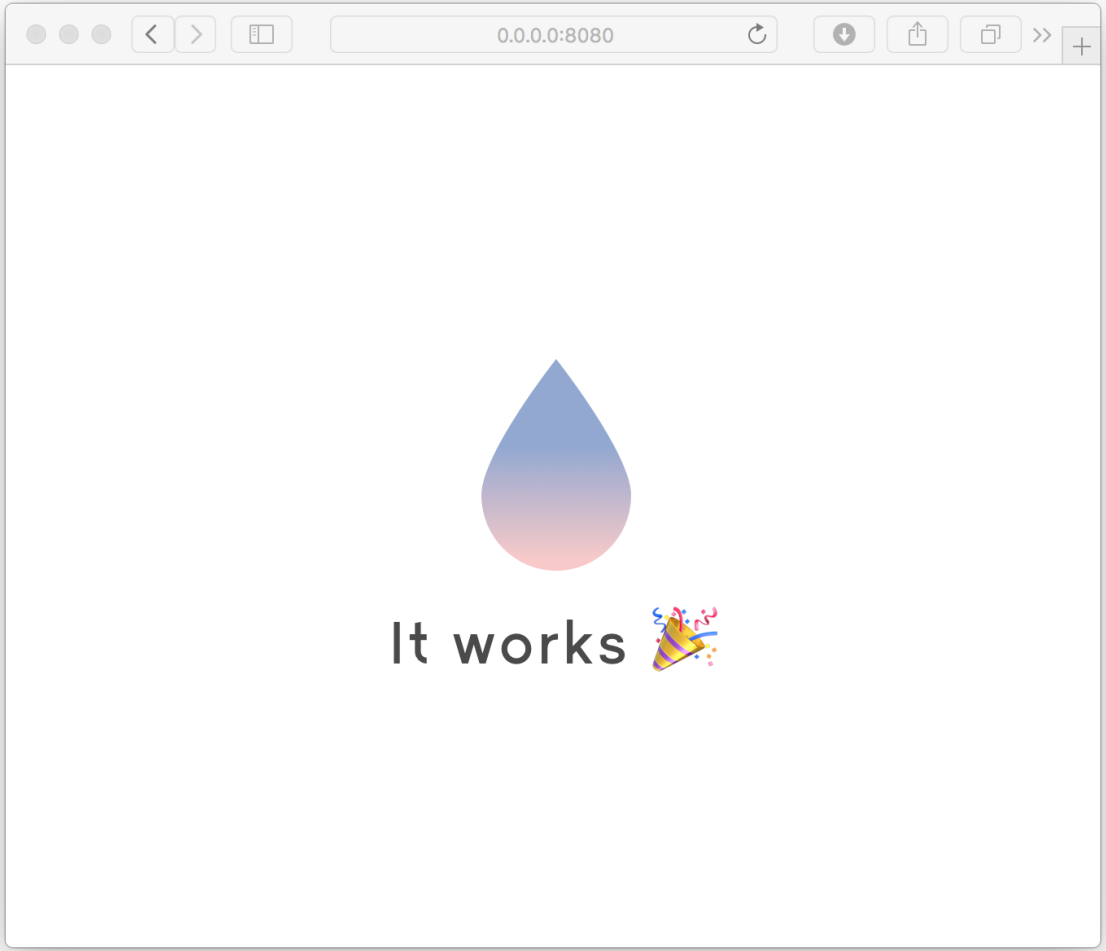
Xcode

Use Run Scheme



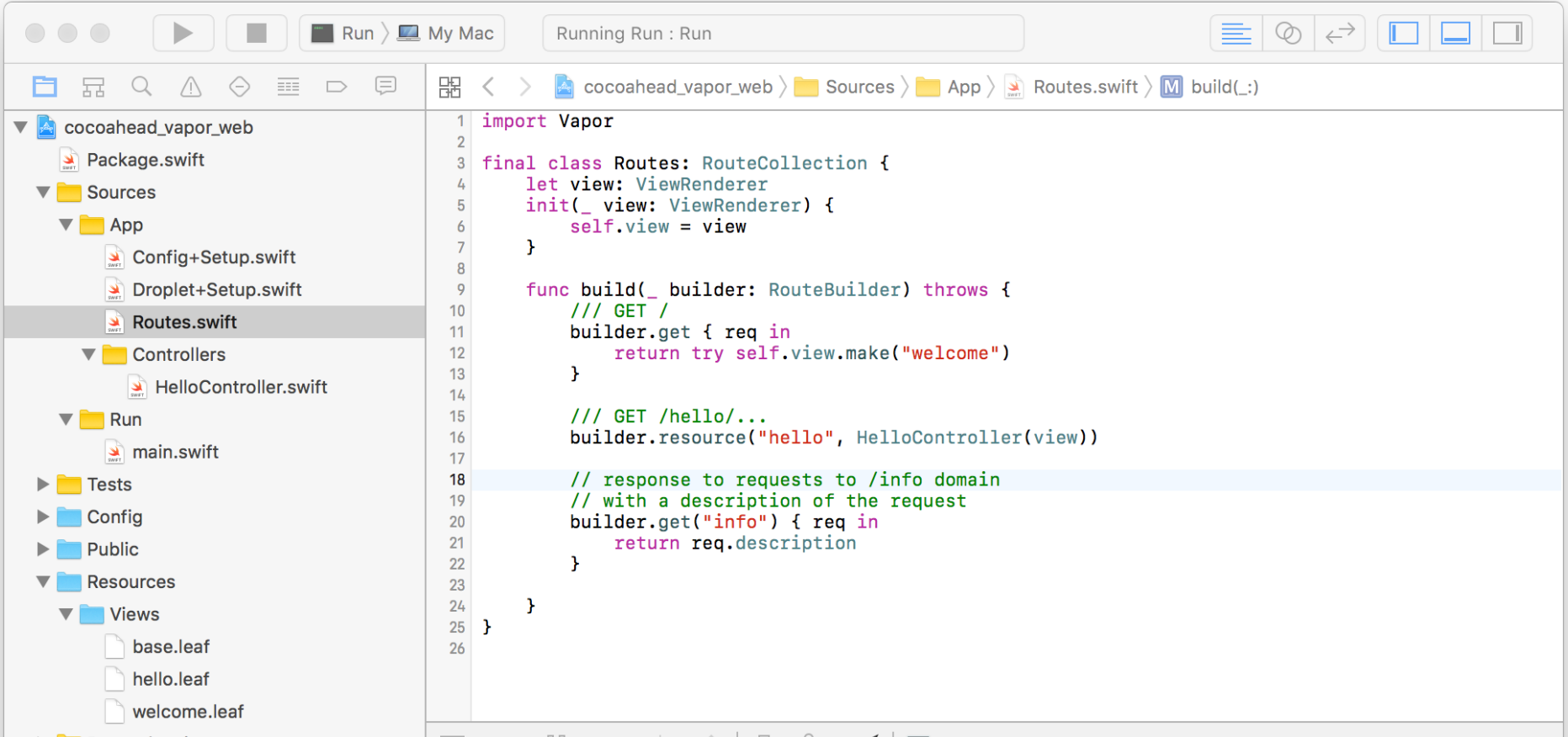
Xcode

Open 0.0.0.0:8080



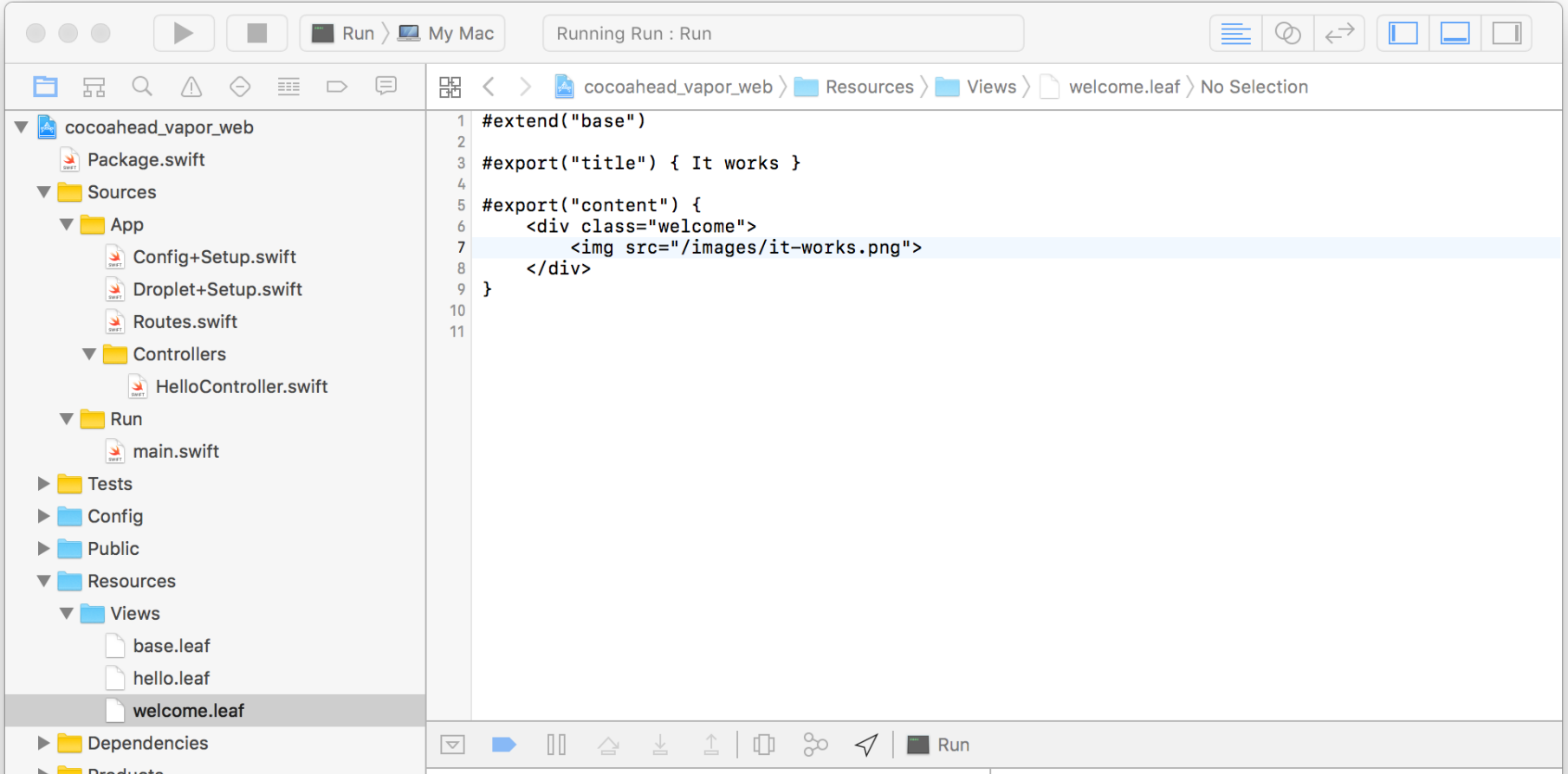
Xcode

Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_web
Sources
App
Routes.swift
build(_:)
cocoahead_vapor_web
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
HelloController.swift
Run
main.swift
Tests
Config
Public
Resources
Views
base.leaf
hello.leaf
welcome.leaf
import Vapor
final class Routes: RouteCollection {
let view: ViewRenderer
init(_ view: ViewRenderer) {
self.view = view
}
func build(_ builder: RouteBuilder) throws {
/// GET /
builder.get { req in
return try self.view.make("welcome")
}
/// GET /hello/...
builder.resource("hello", HelloController(view))
// response to requests to /info domain
// with a description of the request
builder.get("info") { req in
return req.description
}
}
}

Xcode

Dig in

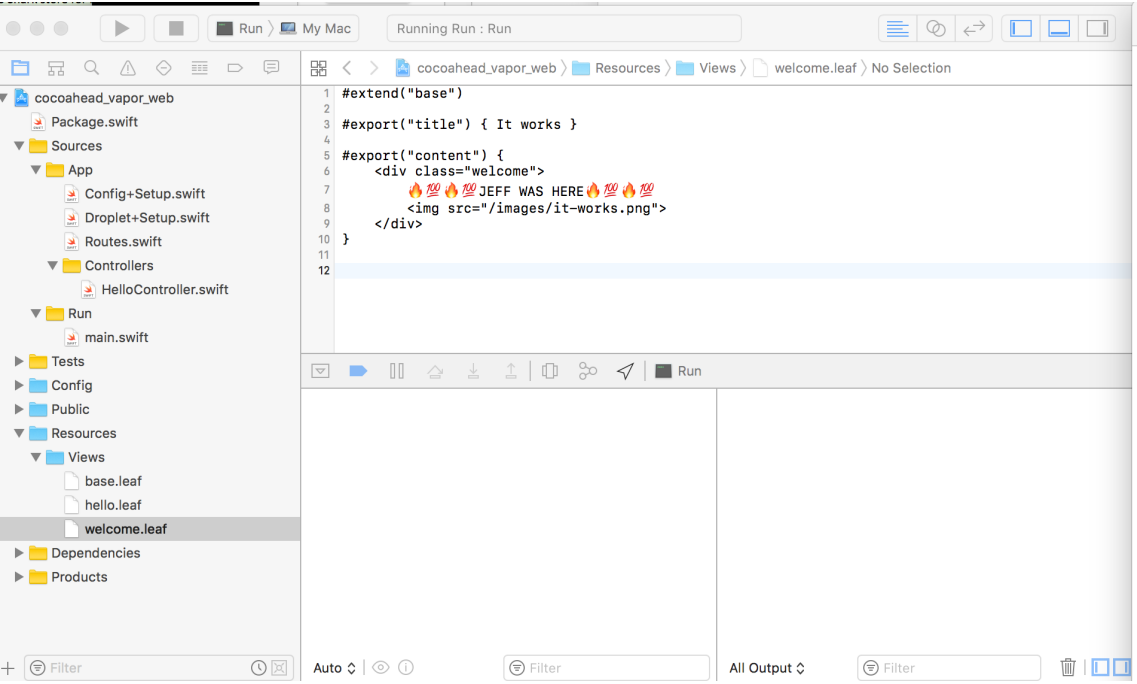

Run
My Mac
Running Run : Run
cocoahead_vapor_web
Resources
Views
welcome.leaf
No Selection
cocoahead_vapor_web
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
HelloController.swift
Run
main.swift
Tests
Config
Public
Resources
Views
base.leaf
hello.leaf
welcome.leaf
Dependencies
#extend("base")
#export("title") { It works }
#export("content") {

<img src="/images/it-works.png">

}
Run

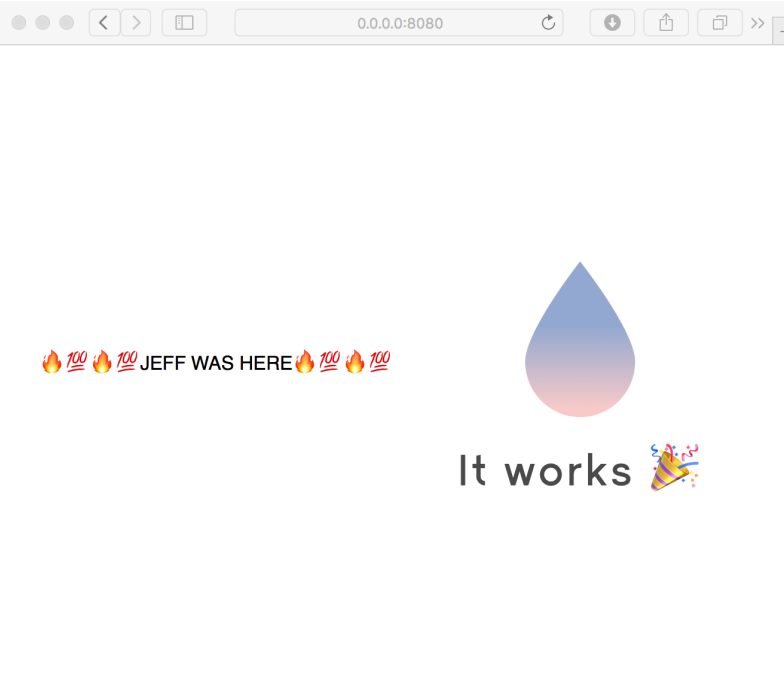
Xcode

Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_web
Resources
Views
welcome.leaf
No Selection
cocoahead_vapor_web
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
HelloController.swift
Run
main.swift
Tests
Config
Public
Resources
Views
base.leaf
hello.leaf
welcome.leaf
Dependencies
Products
#extend("base")
#export("title") { It works }
#export("content") {

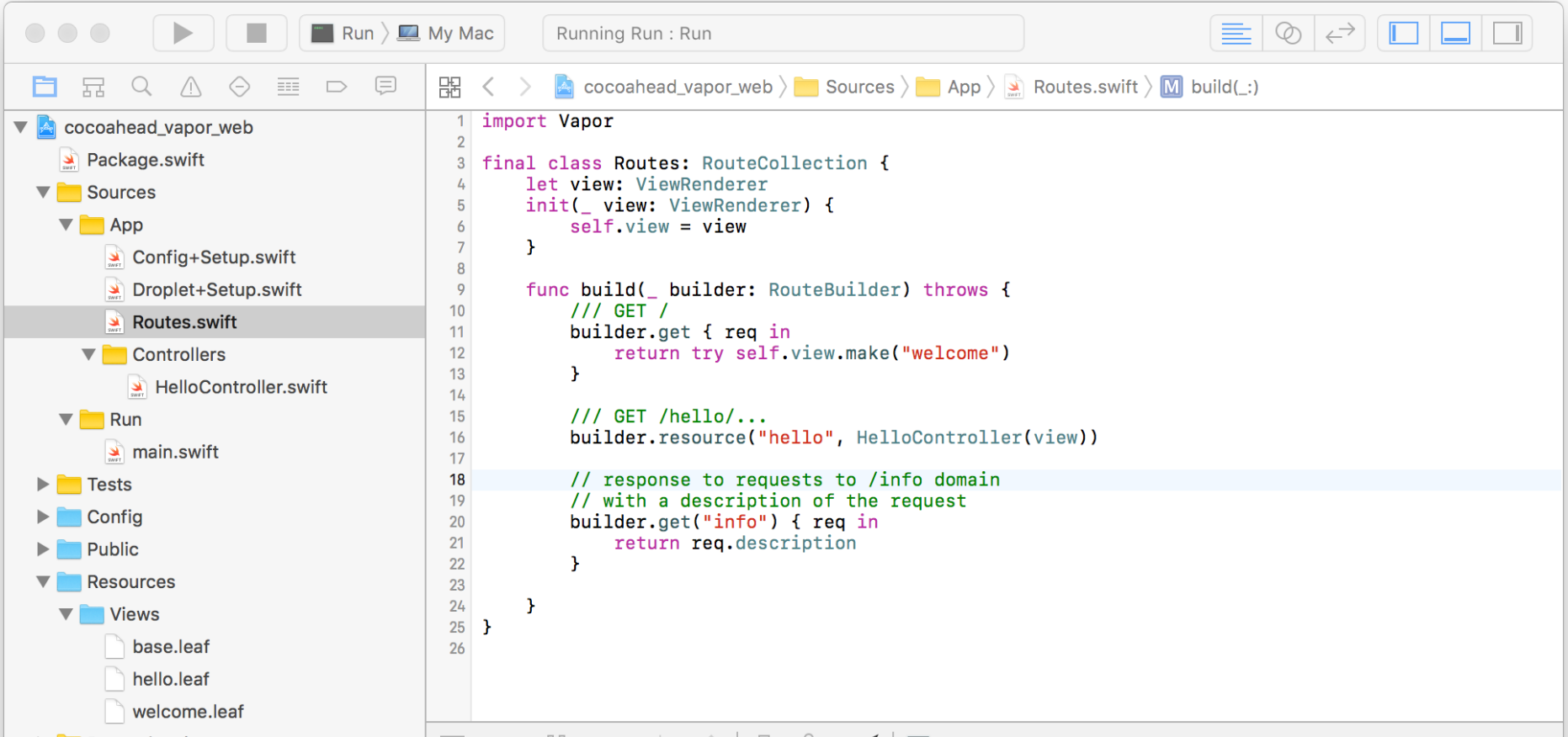
🔥💯🔥💯JEFF WAS HERE🔥💯🔥💯
<img src="/images/it-works.png">

}
Run
Auto
All Output
Filter


0.0.0.0:8080
🔥💯🔥💯JEFF WAS HERE🔥💯🔥💯
It works

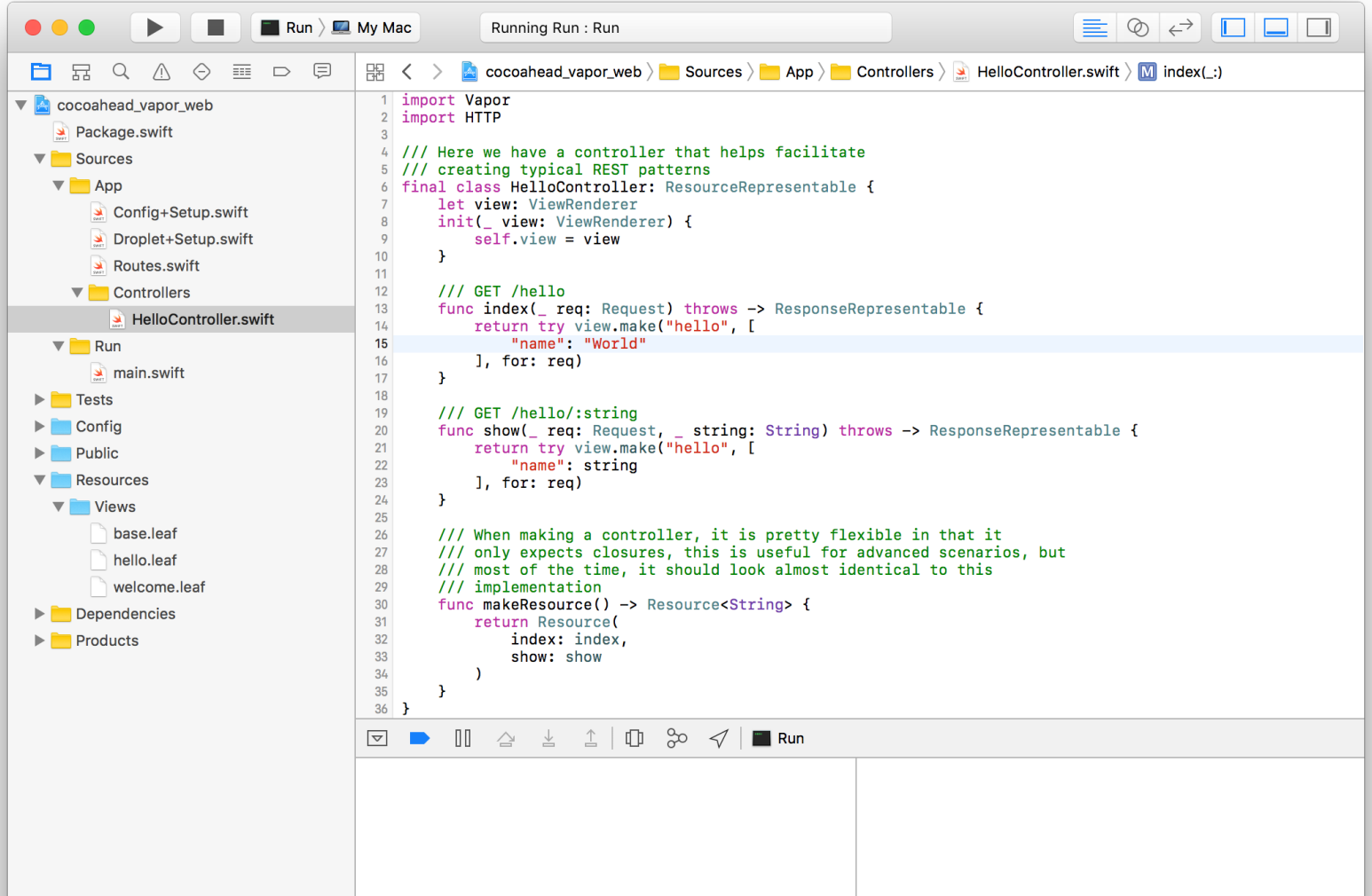
Xcode

Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_web
Sources
App
Routes.swift
build(_:)
cocoahead_vapor_web
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
HelloController.swift
Run
main.swift
Tests
Config
Public
Resources
Views
base.leaf
hello.leaf
welcome.leaf
import Vapor
final class Routes: RouteCollection {
let view: ViewRenderer
init(_ view: ViewRenderer) {
self.view = view
}
func build(_ builder: RouteBuilder) throws {
/// GET /
builder.get { req in
return try self.view.make("welcome")
}
/// GET /hello/...
builder.resource("hello", HelloController(view))
// response to requests to /info domain
// with a description of the request
builder.get("info") { req in
return req.description
}
}
}

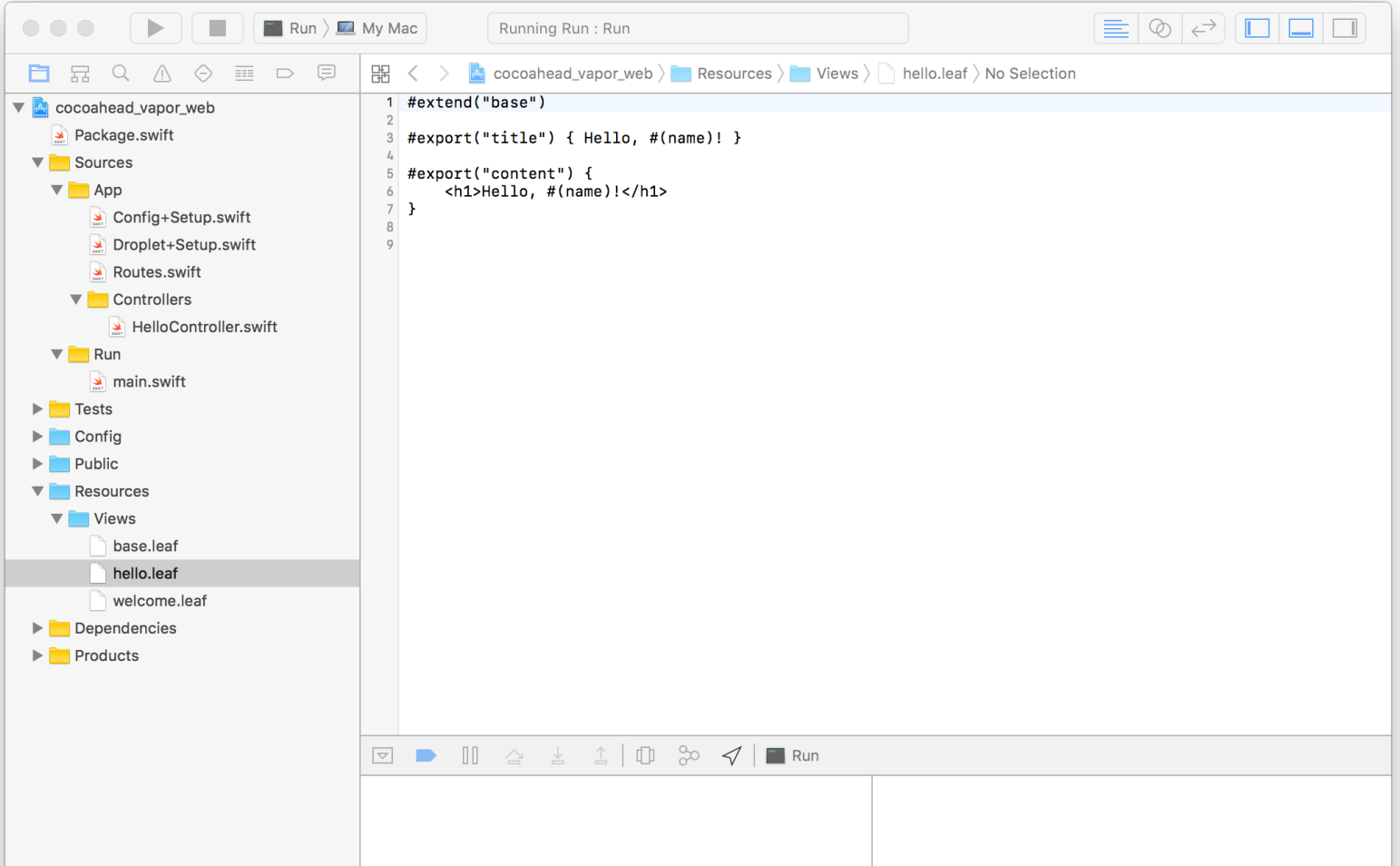
Xcode

Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_web
Sources
App
Controllers
HelloController.swift
index(_:)
cocoahead_vapor_web
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
HelloController.swift
Run
main.swift
Tests
Config
Public
Resources
Views
base.leaf
hello.leaf
welcome.leaf
Dependencies
Products
import Vapor
import HTTP
/// Here we have a controller that helps facilitate
/// creating typical REST patterns
final class HelloController: ResourceRepresentable {
let view: ViewRenderer
init(_ view: ViewRenderer) {
self.view = view
}
/// GET /hello
func index(_ req: Request) throws -> ResponseRepresentable {
return try view.make("hello", [
"name": "World"
], for: req)
}
/// GET /hello/:string
func show(_ req: Request, _ string: String) throws -> ResponseRepresentable {
return try view.make("hello", [
"name": string
], for: req)
}
/// When making a controller, it is pretty flexible in that it
/// only expects closures, this is useful for advanced scenarios, but
/// most of the time, it should look almost identical to this
/// implementation
func makeResource() -> Resource<String> {
return Resource(
index: index,
show: show
)
}
}
Run

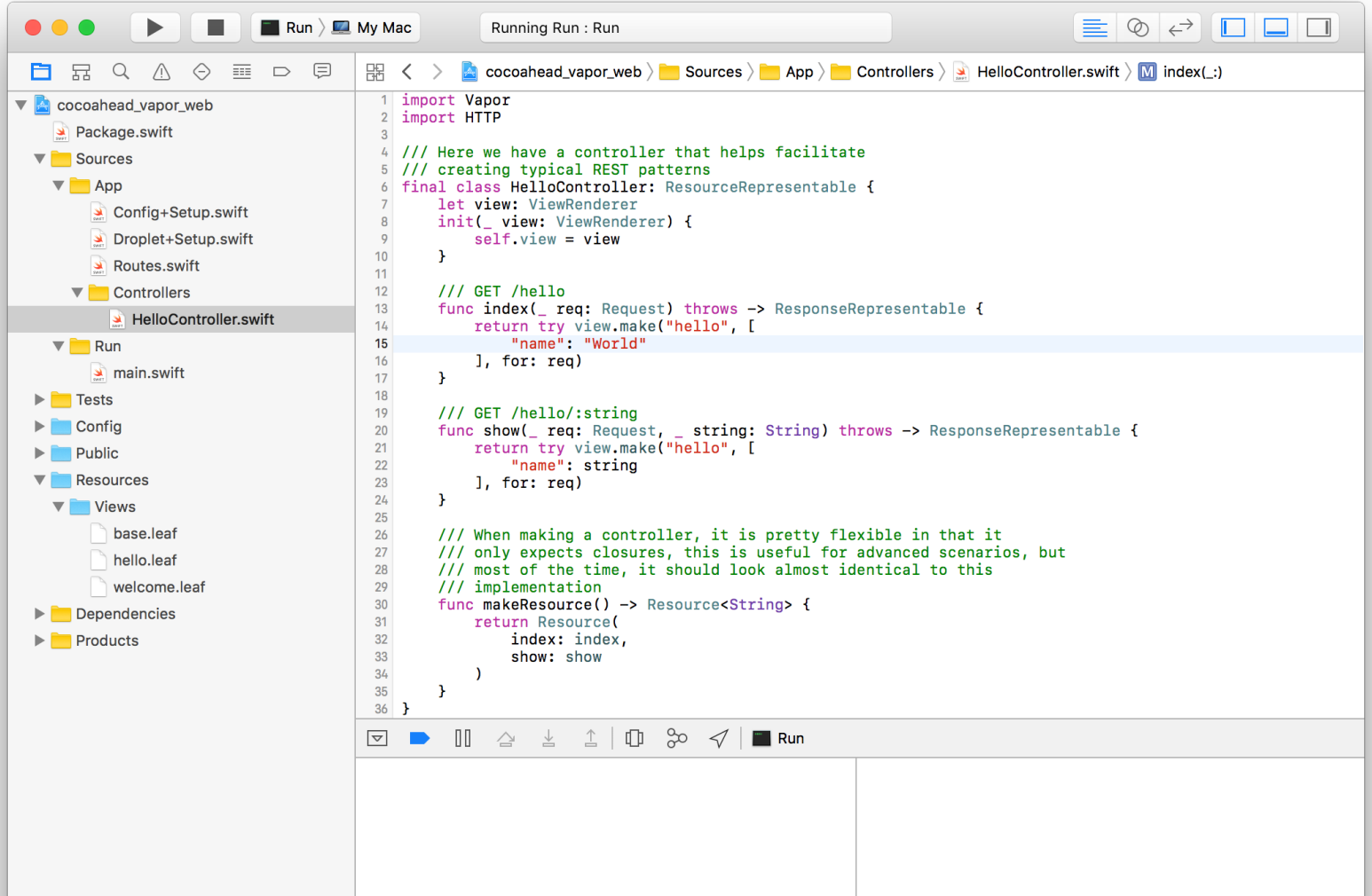
Xcode

Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_web
Resources
Views
hello.leaf
No Selection
cocoahead_vapor_web
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
HelloController.swift
Run
main.swift
Tests
Config
Public
Resources
Views
base.leaf
hello.leaf
welcome.leaf
Dependencies
Products
#extend("base")
#export("title") { Hello, #(name)! }
#export("content") {
<h1>Hello, #(name)!</h1>
}
Run

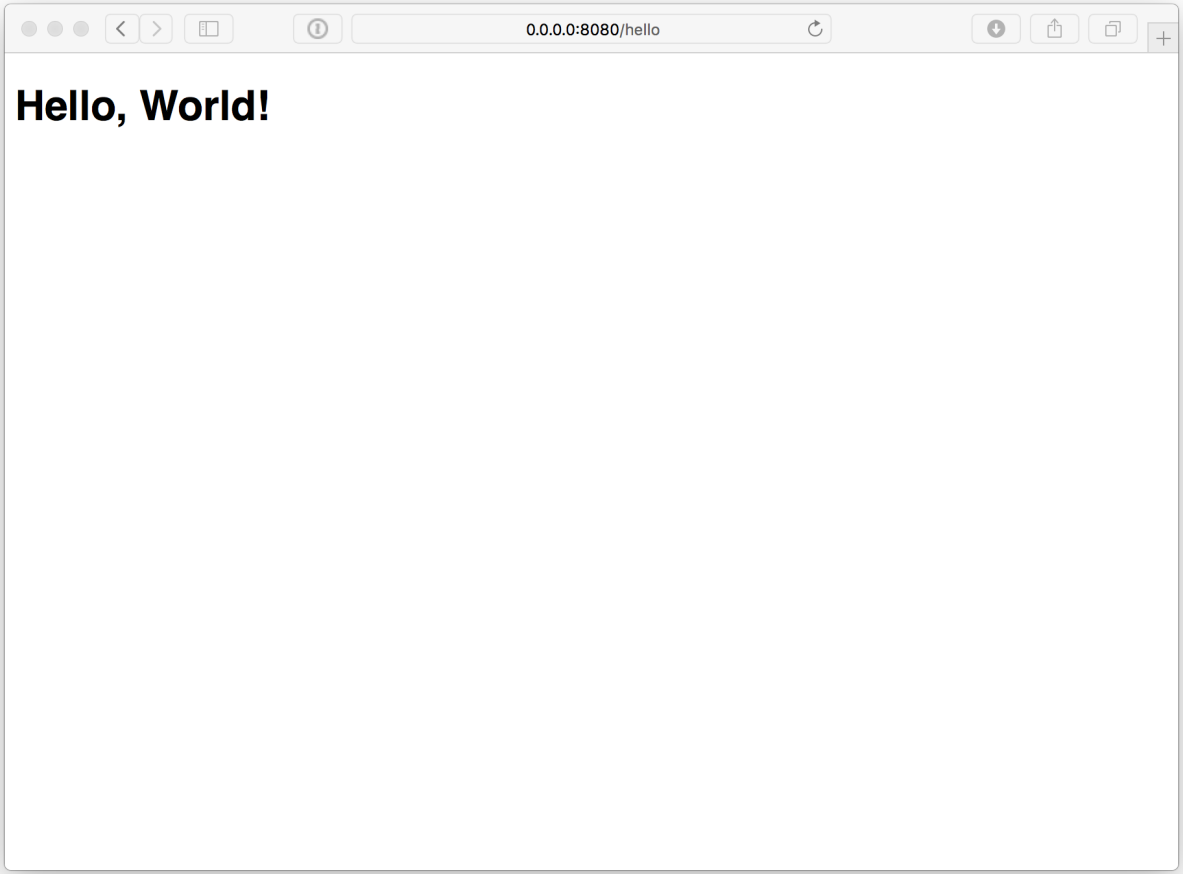
Xcode

Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_web
Sources
App
Controllers
HelloController.swift
index(_:)
cocoahead_vapor_web
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
HelloController.swift
Run
main.swift
Tests
Config
Public
Resources
Views
base.leaf
hello.leaf
welcome.leaf
Dependencies
Products
import Vapor
import HTTP
/// Here we have a controller that helps facilitate
/// creating typical REST patterns
final class HelloController: ResourceRepresentable {
let view: ViewRenderer
init(_ view: ViewRenderer) {
self.view = view
}
/// GET /hello
func index(_ req: Request) throws -> ResponseRepresentable {
return try view.make("hello", [
"name": "World"
], for: req)
}
/// GET /hello/:string
func show(_ req: Request, _ string: String) throws -> ResponseRepresentable {
return try view.make("hello", [
"name": string
], for: req)
}
/// When making a controller, it is pretty flexible in that it
/// only expects closures, this is useful for advanced scenarios, but
/// most of the time, it should look almost identical to this
/// implementation
func makeResource() -> Resource<String> {
return Resource(
index: index,
show: show
)
}
}
Run

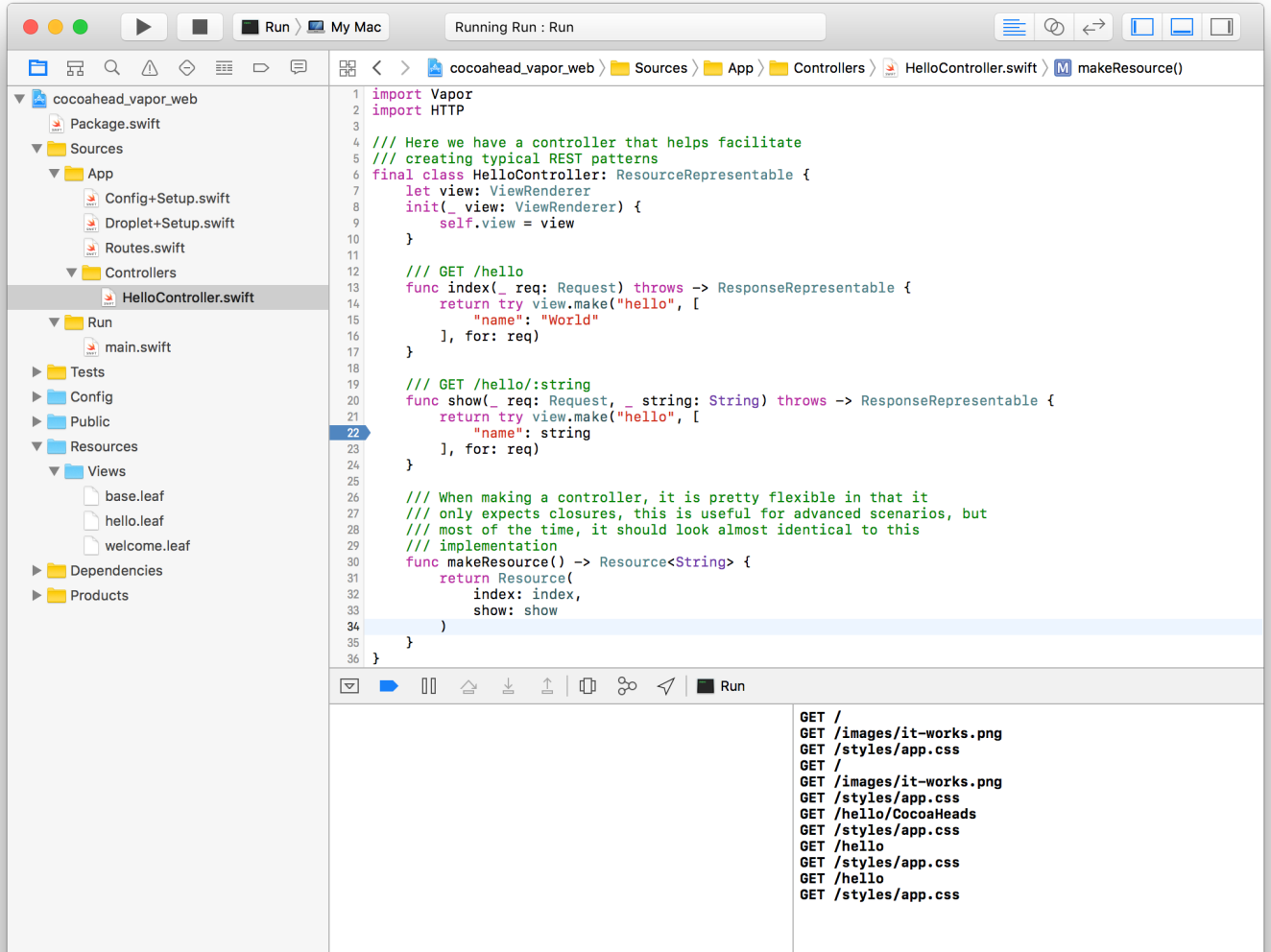
Xcode

Dig in



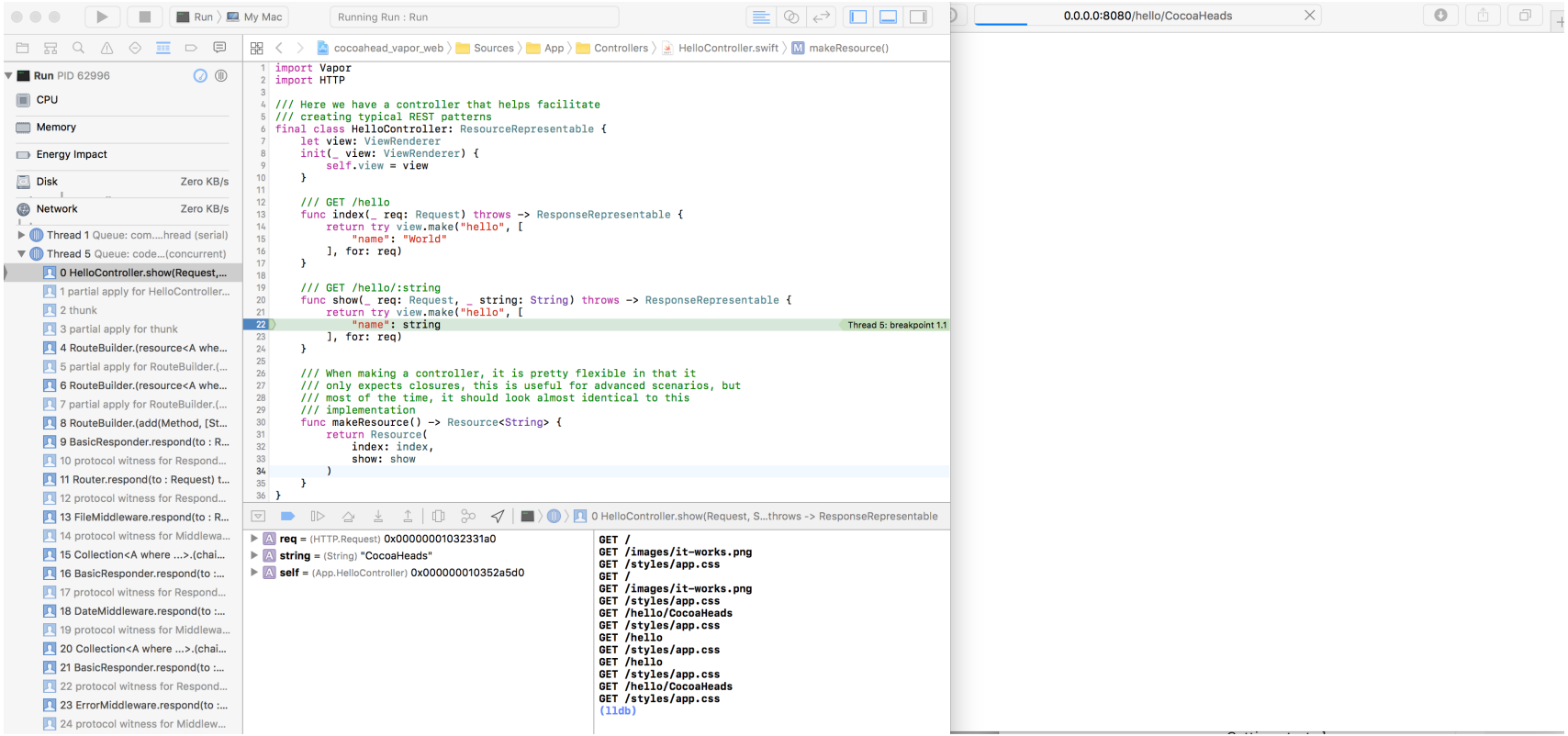
Xcode

Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_web
Sources
App
Controllers
HelloController.swift
makeResource()
cocoahead_vapor_web
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
HelloController.swift
Run
main.swift
Tests
Config
Public
Resources
Views
base.leaf
hello.leaf
welcome.leaf
Dependencies
Products
import Vapor
import HTTP
/// Here we have a controller that helps facilitate
/// creating typical REST patterns
final class HelloController: ResourceRepresentable {
let view: ViewRenderer
init(_ view: ViewRenderer) {
self.view = view
}
/// GET /hello
func index(_ req: Request) throws -> ResponseRepresentable {
return try view.make("hello", [
"name": "World"
], for: req)
}
/// GET /hello/:string
func show(_ req: Request, _ string: String) throws -> ResponseRepresentable {
return try view.make("hello", [
"name": string
], for: req)
}
/// When making a controller, it is pretty flexible in that it
/// only expects closures, this is useful for advanced scenarios, but
/// most of the time, it should look almost identical to this
/// implementation
func makeResource() -> Resource<String> {
return Resource(
index: index,
show: show
)
}
}
Run
GET /
GET /images/it-works.png
GET /styles/app.css
GET /
GET /images/it-works.png
GET /styles/app.css
GET /hello/CocoaHeads
GET /styles/app.css
GET /hello
GET /styles/app.css
GET /hello
GET /styles/app.css

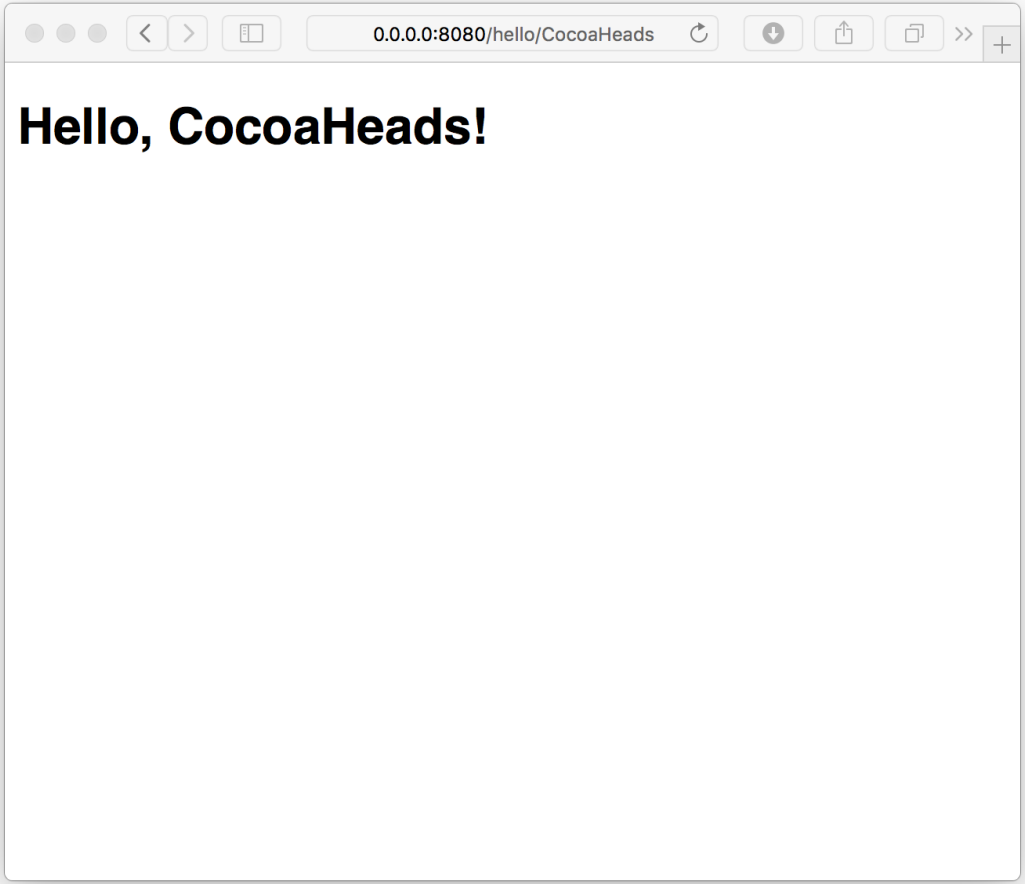
Xcode

Dig in


Run
My Mac
Running Run : Run
0.0.0.0:8080/hello/CocoaHeads
cocoahead_vapor_web
Sources
App
Controllers
HelloController.swift
makeResource()
Run PID 62996
CPU
Memory
Energy Impact
Disk
Zero KB/s
Network
Zero KB/s
Thread 1 Queue: com....hread (serial)
Thread 5 Queue: code...(concurrent)
0 HelloController.show(Request,...
1 partial apply for HelloController...
2 thunk
3 partial apply for thunk
4 RouteBuilder.(resource<A whe...
5 partial apply for RouteBuilder.(...
6 RouteBuilder.(resource<A whe...
7 partial apply for RouteBuilder.(...
8 RouteBuilder.(add(Method, [St...
9 BasicResponder.respond(to : R...
10 protocol witness for Respond...
11 Router.respond(to : Request) t...
12 protocol witness for Respond...
13 FileMiddleware.respond(to : R...
14 protocol witness for Middlewa...
15 Collection<A where ...>.(chai...
16 BasicResponder.respond(to :...
17 protocol witness for Respond...
18 DateMiddleware.respond(to :...
19 protocol witness for Middlewa...
20 Collection<A where ...>.(chai...
21 BasicResponder.respond(to :...
22 protocol witness for Respond...
23 ErrorMiddleware.respond(to :...
24 protocol witness for Middlew...
import Vapor
import HTTP
/// Here we have a controller that helps facilitate
/// creating typical REST patterns
final class HelloController: ResourceRepresentable {
let view: ViewRenderer
init(_ view: ViewRenderer) {
self.view = view
}
/// GET /hello
func index(_ req: Request) throws -> ResponseRepresentable {
return try view.make("hello", [
"name": "World"
], for: req)
}
/// GET /hello/:string
func show(_ req: Request, _ string: String) throws -> ResponseRepresentable {
return try view.make("hello", [
"name": string
], for: req)
}
/// When making a controller, it is pretty flexible in that it
/// only expects closures, this is useful for advanced scenarios, but
/// most of the time, it should look almost identical to this
/// implementation
func makeResource() -> Resource<String> {
return Resource(
index: index,
show: show
)
}
}
Thread 5: breakpoint 1.1
0 HelloController.show(Request, S...throws -> ResponseRepresentable
req = (HTTP.Request) 0x00000001032331a0
string = (String) "CocoaHeads"
self = (App.HelloController) 0x000000010352a5d0
GET /
GET /images/it-works.png
GET /styles/app.css
GET /
GET /images/it-works.png
GET /styles/app.css
GET /hello/CocoaHeads
GET /styles/app.css
GET /hello
GET /styles/app.css
GET /hello
GET /styles/app.css
GET /hello/CocoaHeads
GET /styles/app.css
(lldb)

Xcode

Dig in



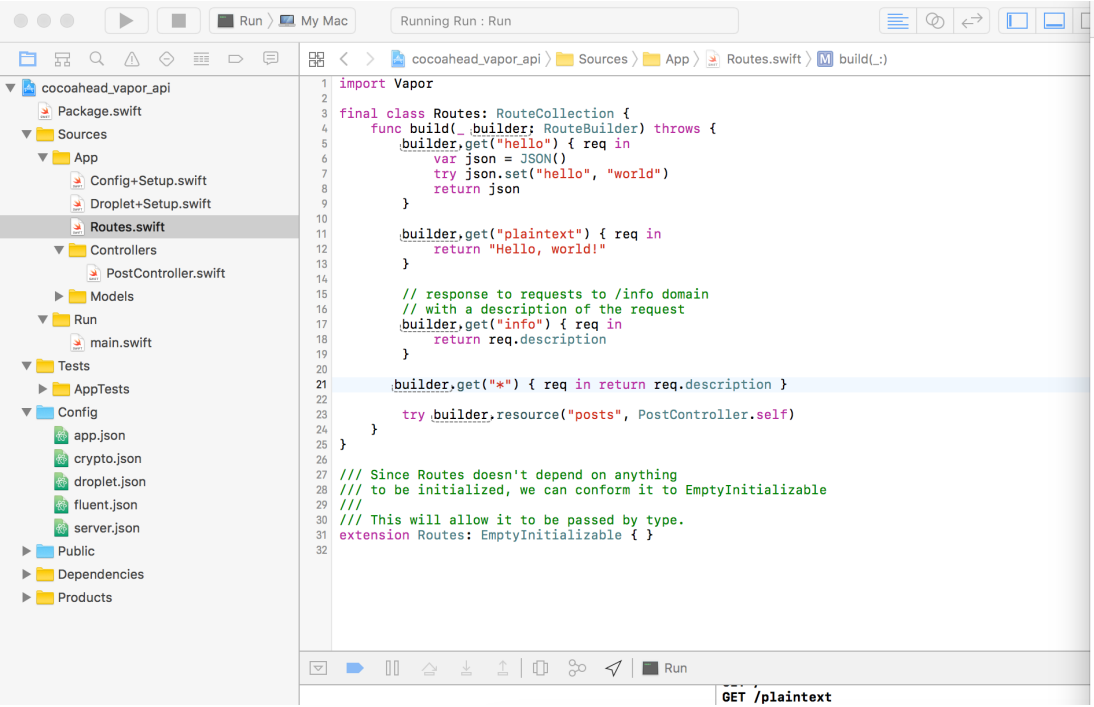
Xcode

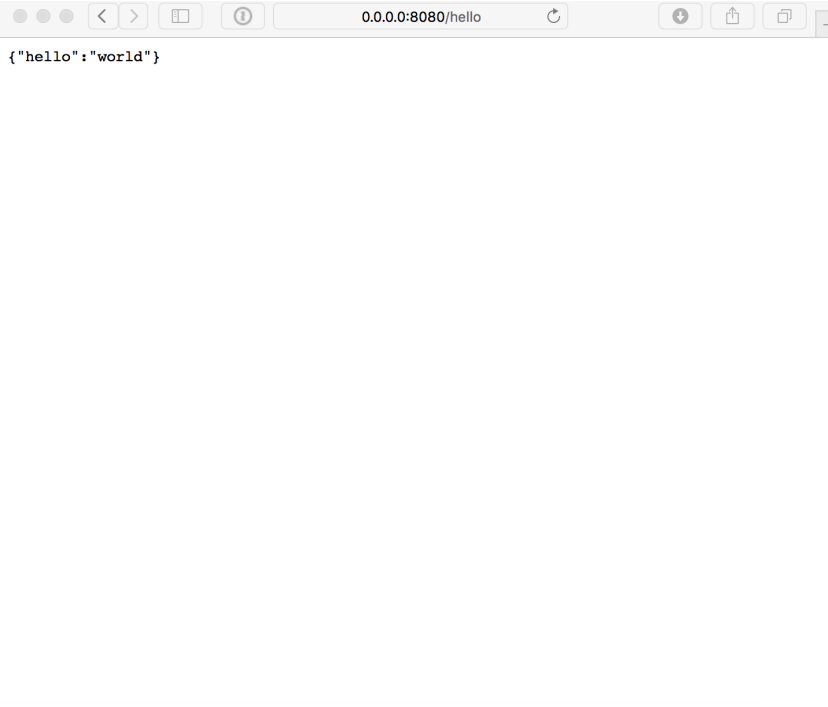
Now with the API Project

```
jmeador:cocoahead_vapor_api jmeador$ vapor xcode -y

Generating Xcode Project [Done]
Select the `Run` scheme to run.
Open Xcode project?
y/n> yes
Opening Xcode project...
```

Xcode

Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_api
Sources
App
Routes.swift
build(_:)
cocoahead_vapor_api
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
PostController.swift
Models
Run
main.swift
Tests
AppTests
Config
app.json
crypto.json
droplet.json
fluent.json
server.json
Public
Dependencies
Products
import Vapor
final class Routes: RouteCollection {
func build(_ builder: RouteBuilder) throws {
builder.get("hello") { req in
var json = JSON()
try json.set("hello", "world")
return json
}
builder.get("plaintext") { req in
return "Hello, world!"
}
// response to requests to /info domain
// with a description of the request
builder.get("info") { req in
return req.description
}
builder.get("*") { req in return req.description }
try builder.resource("posts", PostController.self)
}
}
/// Since Routes doesn't depend on anything
/// to be initialized, we can conform it to EmptyInitializable
///
/// This will allow it to be passed by type.
extension Routes: EmptyInitializable { }
Run
GET /plaintext


0.0.0.0:8080/hello
{"hello":"world"}

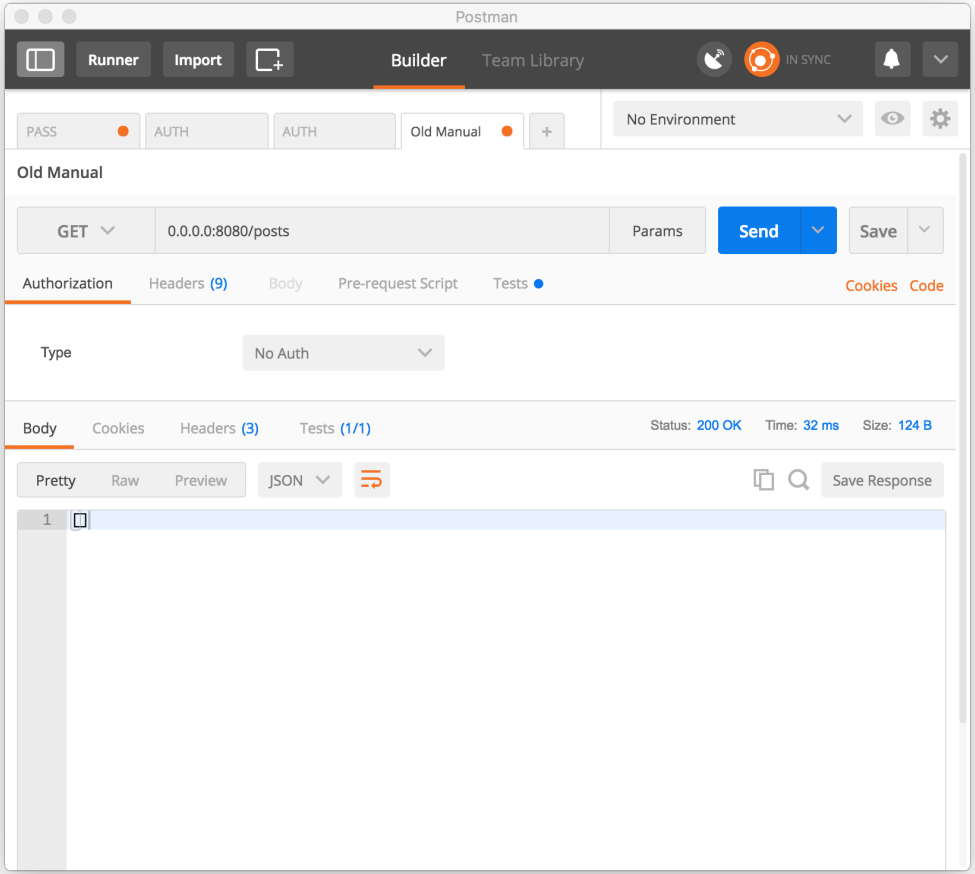
Xcode

Dig in

```
// Here we have a controller that helps facilitate
/// RESTful interactions with our Posts table
final class PostController: ResourceRepresentable {
    /// When users call 'GET' on '/posts'
    /// it should return an index of all available posts
    func index(request: Request) throws -> ResponseRepresentable {
        return try Post.all().makeJSON()
    }

    /// When consumers call 'POST' on '/posts' with valid JSON
    /// create and save the post
    func create(request: Request) throws -> ResponseRepresentable {
        let post = try request.post()
        try post.save()
        return post
    }

    /// When the consumer calls 'GET' on a specific resource, ie:
    /// '/posts/13rd88' we should show that specific post
    func show(request: Request, post: Post) throws -> ResponseRepresentable {
        return post
    }

    /// When the consumer calls 'DELETE' on a specific resource, ie:
    /// 'posts/l2jd9' we should remove that resource from the database
    func delete(request: Request, post: Post) throws -> ResponseRepresentable {
        try post.delete()
        return Response(status: .ok)
    }

    /// When the consumer calls 'DELETE' on the entire table, ie:
    /// '/posts' we should remove the entire table
    func clear(request: Request) throws -> ResponseRepresentable {
        try Post.makeQuery().delete()
        return Response(status: .ok)
    }

    /// When the user calls 'PATCH' on a specific resource, we should
    /// update that resource to the new values
    func update(request: Request, post: Post) throws -> ResponseRepresentable {
        let new = try request.post()
        post.content = new.content
        try post.save()
        return post
    }

    /// When a user calls 'PUT' on a specific resource, we should
    /// delete the current value and completely replace it with the
    /// new parameters
    func replace(request: Request, post: Post) throws -> ResponseRepresentable {
        try post.delete()
        return try create(request: request)
    }
```

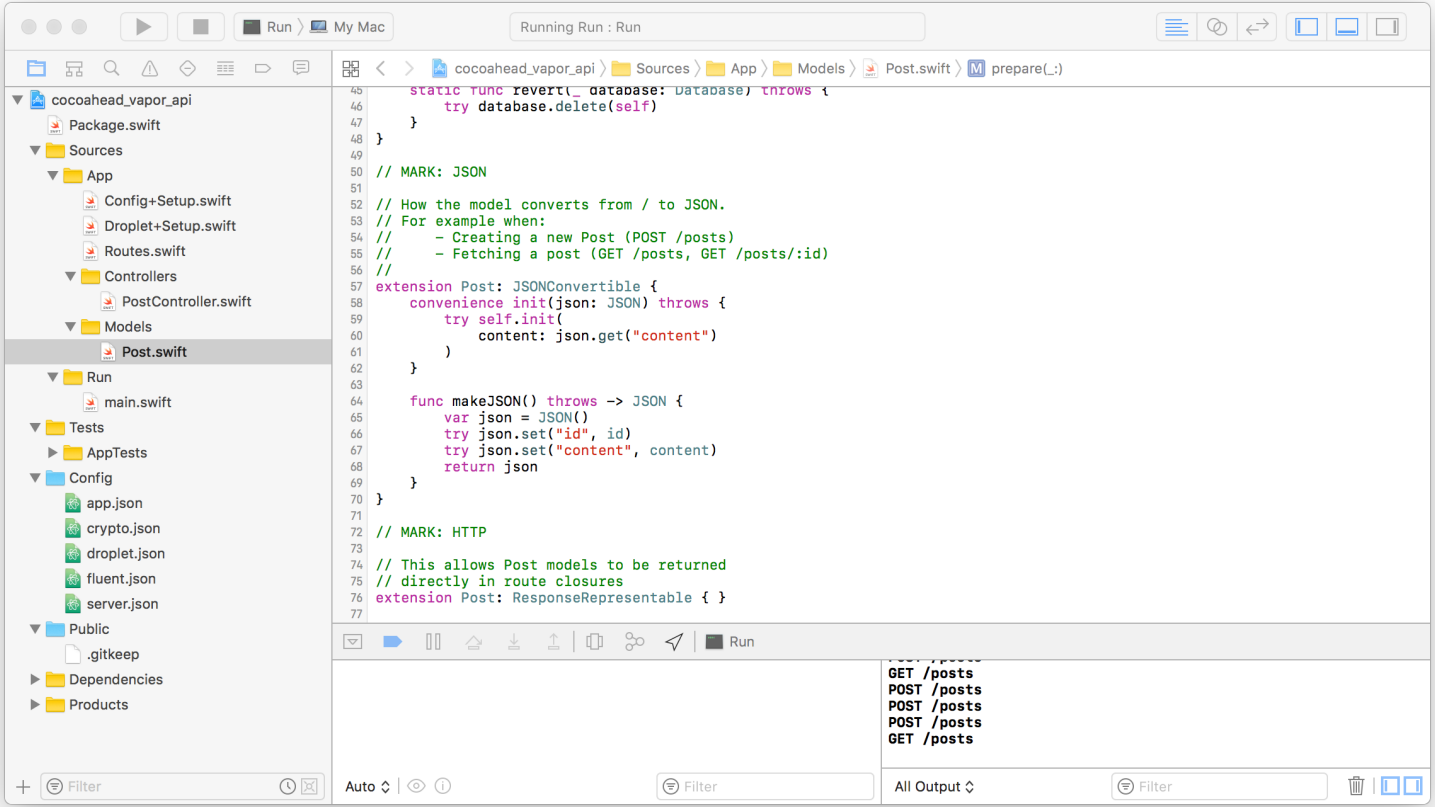
Xcode

Dig in


Postman
Runner
Import
Builder
Team Library
IN SYNC
PASS
AUTH
AUTH
Old Manual
No Environment
Old Manual
GET
0.0.0.0:8080/posts
Params
Send
Save
Authorization
Headers (9)
Body
Pre-request Script
Tests
Cookies
Code
Type
No Auth
Body
Cookies
Headers (3)
Tests (1/1)
Status: 200 OK
Time: 32 ms
Size: 124 B
Pretty
Raw
Preview
JSON
Save Response
1

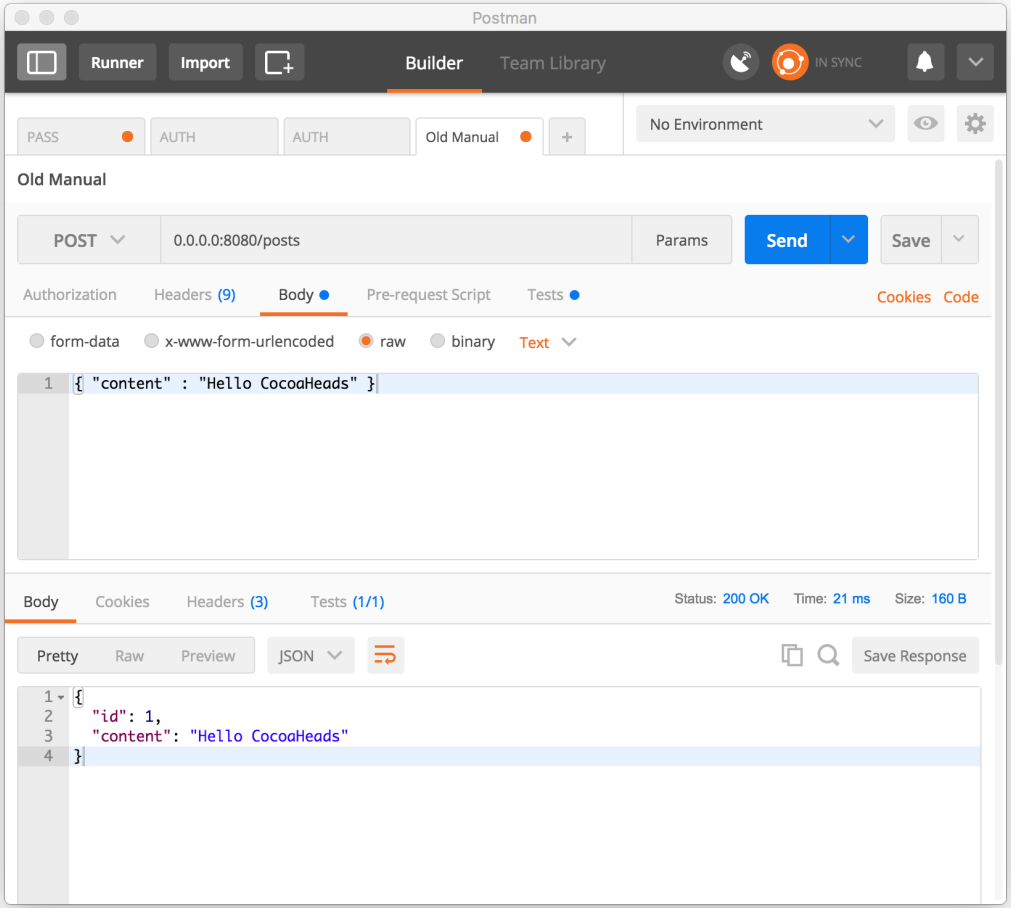
Xcode

Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_api
Sources
App
Models
Post.swift
prepare(_:)
cocoahead_vapor_api
Package.swift
Sources
App
Config+Setup.swift
Droplet+Setup.swift
Routes.swift
Controllers
PostController.swift
Models
Post.swift
Run
main.swift
Tests
AppTests
Config
app.json
crypto.json
droplet.json
fluent.json
server.json
Public
.gitkeep
Dependencies
Products
static func revert(_ database: Database) throws {
try database.delete(self)
}
}
// MARK: JSON
// How the model converts from / to JSON.
// For example when:
// - Creating a new Post (POST /posts)
// - Fetching a post (GET /posts, GET /posts/:id)
//
extension Post: JSONConvertible {
convenience init(json: JSON) throws {
try self.init(
content: json.get("content")
)
}
func makeJSON() throws -> JSON {
var json = JSON()
try json.set("id", id)
try json.set("content", content)
return json
}
}
// MARK: HTTP
// This allows Post models to be returned
// directly in route closures
extension Post: ResponseRepresentable { }
Run
GET /posts
POST /posts
POST /posts
POST /posts
GET /posts
Auto
All Output
Filter

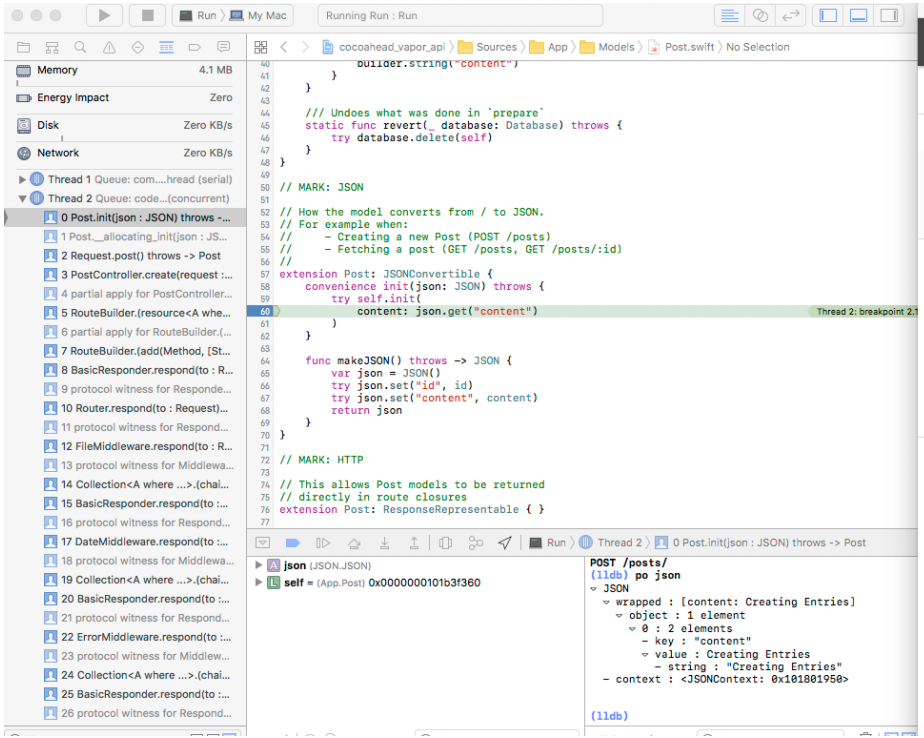
Xcode

Dig in


Postman
Runner
Import
Builder
Team Library
IN SYNC
PASS
AUTH
AUTH
Old Manual
No Environment
Old Manual
POST
0.0.0.0:8080/posts
Params
Send
Save
Authorization
Headers (9)
Body
Pre-request Script
Tests
Cookies
Code
form-data
x-www-form-urlencoded
raw
binary
Text
1 { "content" : "Hello CocoaHeads" }
Body
Cookies
Headers (3)
Tests (1/1)
Status: 200 OK
Time: 21 ms
Size: 160 B
Pretty
Raw
Preview
JSON
Save Response
1 {
2 "id": 1,
3 "content": "Hello CocoaHeads"
4 }

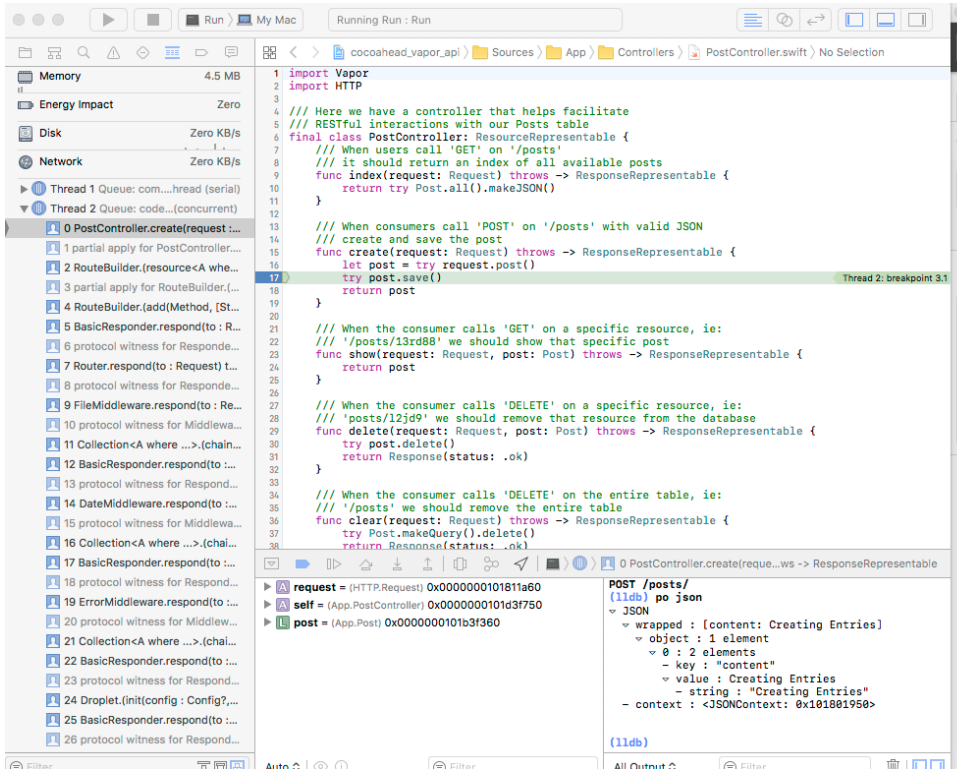
Xcode

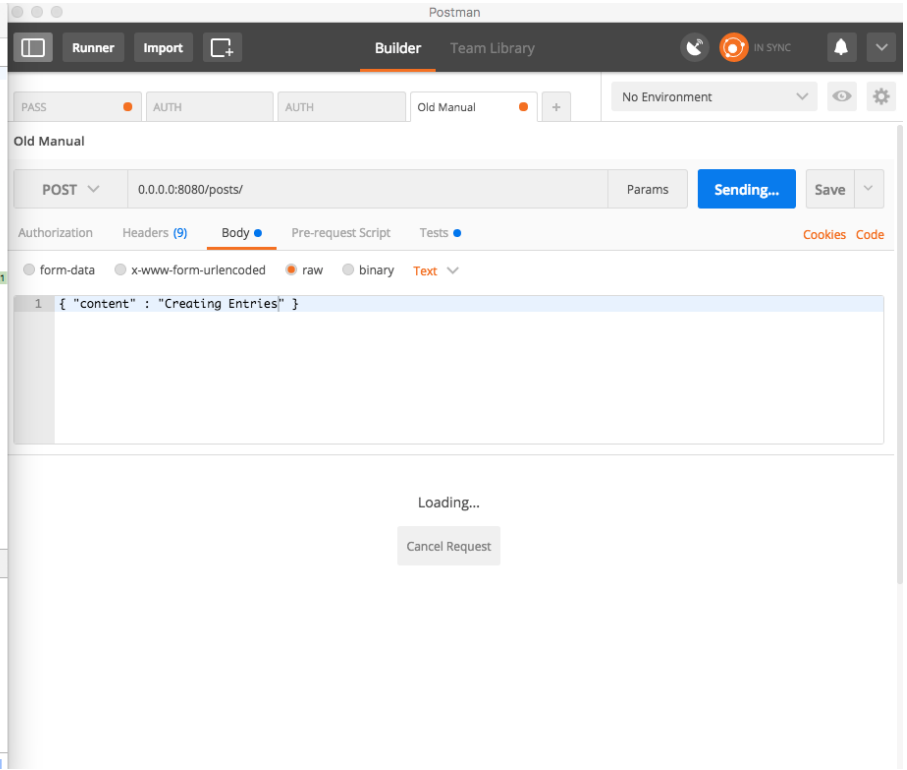
Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_api
Sources
App
Models
Post.swift
No Selection
Memory 4.1 MB
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s
Thread 1 Queue: com....hread (serial)
Thread 2 Queue: code...(concurrent)
0 Post.init(json : JSON) throws -...
1 Post.__allocating_init(json : JS...
2 Request.post() throws -> Post
3 PostController.create(request :...
4 partial apply for PostController...
5 RouteBuilder.(resource<A whe...
6 partial apply for RouteBuilder.(...
7 RouteBuilder.(add(Method, [St...
8 BasicResponder.respond(to : R...
9 protocol witness for Responde...
10 Router.respond(to : Request)...
11 protocol witness for Respond...
12 FileMiddleware.respond(to : R...
13 protocol witness for Middlewa...
14 Collection<A where ...>.(chai...
15 BasicResponder.respond(to :...
16 protocol witness for Respond...
17 DateMiddleware.respond(to :...
18 protocol witness for Middlewa...
19 Collection<A where ...>.(chai...
20 BasicResponder.respond(to :...
21 protocol witness for Respond...
22 ErrorMiddleware.respond(to :...
23 protocol witness for Middlew...
24 Collection<A where ...>.(chai...
25 BasicResponder.respond(to :...
26 protocol witness for Respond...
builder.string("content")
}
}
/// Undoes what was done in `prepare`
static func revert(_ database: Database) throws {
try database.delete(self)
}
}
// MARK: JSON
// How the model converts from / to JSON.
// For example when:
// - Creating a new Post (POST /posts)
// - Fetching a post (GET /posts, GET /posts/:id)
//
extension Post: JSONConvertible {
convenience init(json: JSON) throws {
try self.init(
content: json.get("content")
)
}
func makeJSON() throws -> JSON {
var json = JSON()
try json.set("id", id)
try json.set("content", content)
return json
}
}
// MARK: HTTP
// This allows Post models to be returned
// directly in route closures
extension Post: ResponseRepresentable { }
Thread 2: breakpoint 2.1
Run
Thread 2
0 Post.init(json : JSON) throws -> Post
json (JSON.JSON)
self = (App.Post) 0x0000000101b3f360
POST /posts/
(lldb) po json
▿ JSON
▿ wrapped : [content: Creating Entries]
▿ object : 1 element
▿ 0 : 2 elements
- key : "content"
▿ value : Creating Entries
- string : "Creating Entries"
- context : <JSONContext: 0x101801950>
(lldb)
Postman
Runner
Import
Builder
Team Library
IN SYNC
PASS
AUTH
AUTH
Old Manual
No Environment
Old Manual
POST
0.0.0.0:8080/posts/
Params
Sending...
Save
Authorization
Headers (9)
Body
Pre-request Script
Tests
Cookies
Code
form-data
x-www-form-urlencoded
raw
binary
Text
1 { "content" : "Creating Entries" }
Loading...
Cancel Request

Xcode

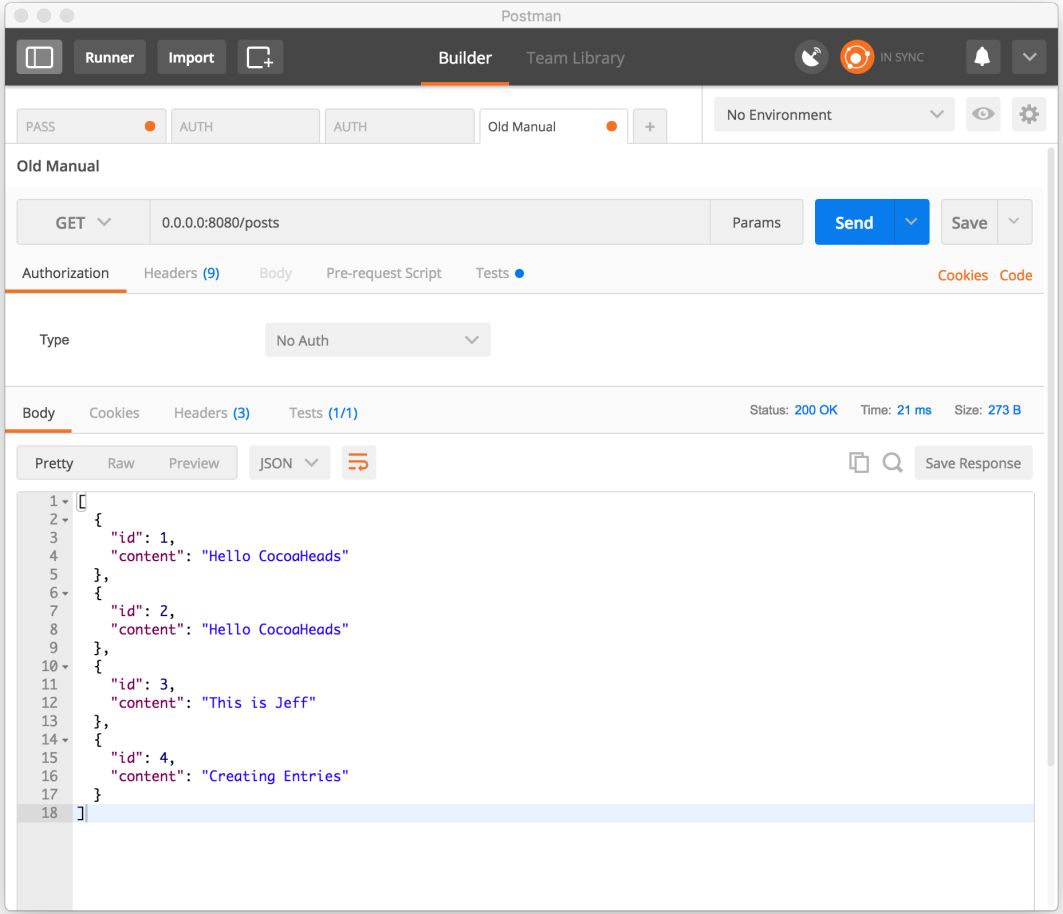
Dig in


Run
My Mac
Running Run : Run
cocoahead_vapor_api
Sources
App
Controllers
PostController.swift
No Selection
Memory 4.5 MB
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s
Thread 1 Queue: com....hread (serial)
Thread 2 Queue: code...(concurrent)
0 PostController.create(request :...
1 partial apply for PostController....
2 RouteBuilder.(resource<A whe...
3 partial apply for RouteBuilder.(...
4 RouteBuilder.(add(Method, [St...
5 BasicResponder.respond(to : R...
6 protocol witness for Responde...
7 Router.respond(to : Request) t...
8 protocol witness for Responde...
9 FileMiddleware.respond(to : Re...
10 protocol witness for Middlewa...
11 Collection<A where ...>.(chain...
12 BasicResponder.respond(to :...
13 protocol witness for Respond...
14 DateMiddleware.respond(to :...
15 protocol witness for Middlewa...
16 Collection<A where ...>.(chai...
17 BasicResponder.respond(to :...
18 protocol witness for Respond...
19 ErrorMiddleware.respond(to :...
20 protocol witness for Middlew...
21 Collection<A where ...>.(chai...
22 BasicResponder.respond(to :...
23 protocol witness for Respond...
24 Droplet.(init(config : Config?,...
25 BasicResponder.respond(to :...
26 protocol witness for Respond...
import Vapor
import HTTP
/// Here we have a controller that helps facilitate
/// RESTful interactions with our Posts table
final class PostController: ResourceRepresentable {
/// When users call 'GET' on '/posts'
/// it should return an index of all available posts
func index(request: Request) throws -> ResponseRepresentable {
return try Post.all().makeJSON()
}
/// When consumers call 'POST' on '/posts' with valid JSON
/// create and save the post
func create(request: Request) throws -> ResponseRepresentable {
let post = try request.post()
try post.save()
Thread 2: breakpoint 3.1
return post
}
/// When the consumer calls 'GET' on a specific resource, ie:
/// '/posts/13rd88' we should show that specific post
func show(request: Request, post: Post) throws -> ResponseRepresentable {
return post
}
/// When the consumer calls 'DELETE' on a specific resource, ie:
/// 'posts/l2jd9' we should remove that resource from the database
func delete(request: Request, post: Post) throws -> ResponseRepresentable {
try post.delete()
return Response(status: .ok)
}
/// When the consumer calls 'DELETE' on the entire table, ie:
/// '/posts' we should remove the entire table
func clear(request: Request) throws -> ResponseRepresentable {
try Post.makeQuery().delete()
return Response(status: .ok)
0 PostController.create(reque...ws -> ResponseRepresentable
request = (HTTP.Request) 0x0000000101811a60
self = (App.PostController) 0x0000000101d3f750
post = (App.Post) 0x0000000101b3f360
POST /posts/
(lldb) po json
JSON
wrapped : [content: Creating Entries]
object : 1 element
0 : 2 elements
key : "content"
value : Creating Entries
string : "Creating Entries"
context : <JSONContext: 0x101801950>
(lldb)


Postman
Runner
Import
Builder
Team Library
IN SYNC
PASS
AUTH
AUTH
Old Manual
No Environment
Old Manual
POST
0.0.0.0:8080/posts/
Params
Sending...
Save
Authorization
Headers (9)
Body
Pre-request Script
Tests
Cookies
Code
form-data
x-www-form-urlencoded
raw
binary
Text
{ "content" : "Creating Entries" }
Loading...
Cancel Request

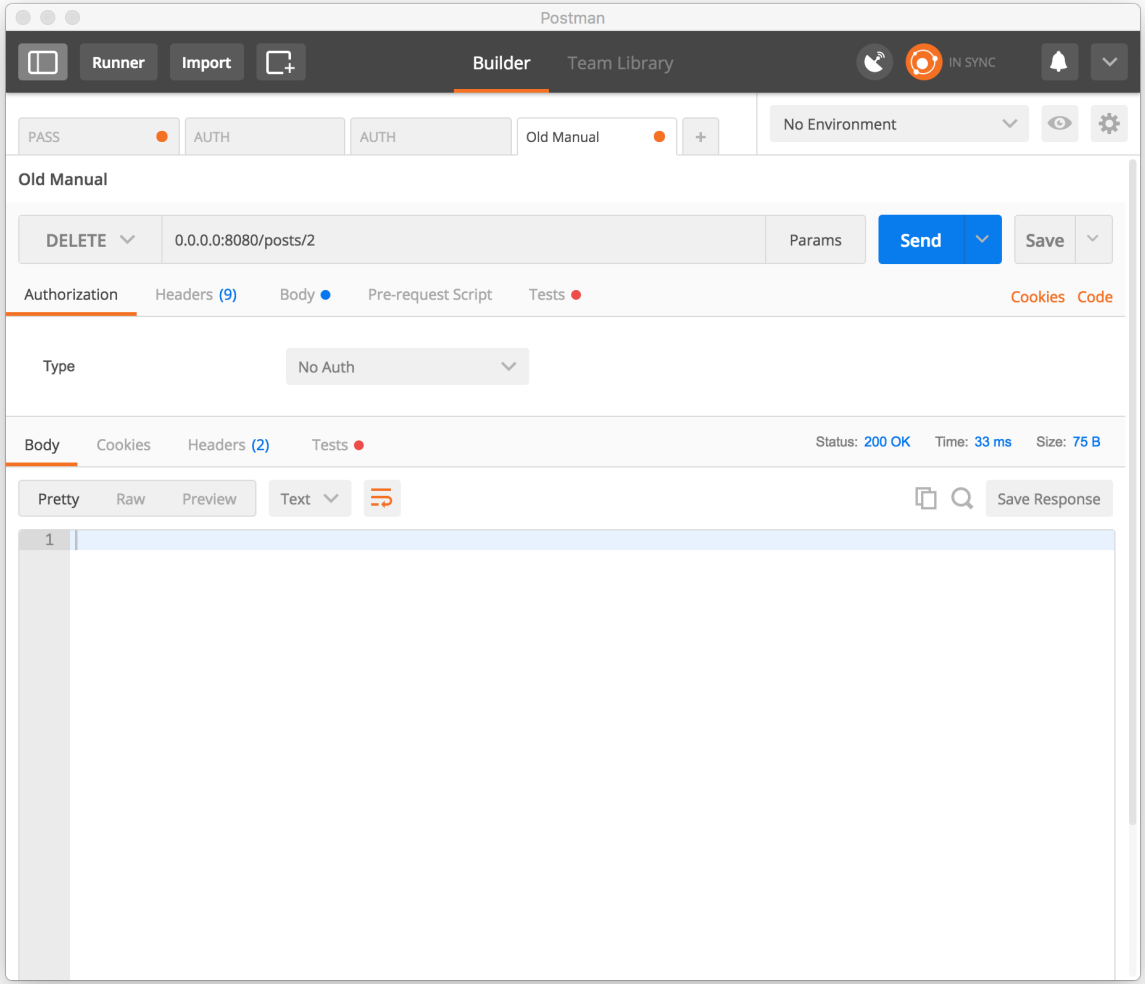
Xcode

Dig in


Postman
Runner
Import
Builder
Team Library
IN SYNC
PASS
AUTH
AUTH
Old Manual
No Environment
Old Manual
GET
0.0.0.0:8080/posts
Params
Send
Save
Authorization
Headers (9)
Body
Pre-request Script
Tests
Cookies
Code
Type
No Auth
Body
Cookies
Headers (3)
Tests (1/1)
Status: 200 OK
Time: 21 ms
Size: 273 B
Pretty
Raw
Preview
JSON
Save Response
[
{
"id": 1,
"content": "Hello CocoaHeads"
},
{
"id": 2,
"content": "Hello CocoaHeads"
},
{
"id": 3,
"content": "This is Jeff"
},
{
"id": 4,
"content": "Creating Entries"
}
]

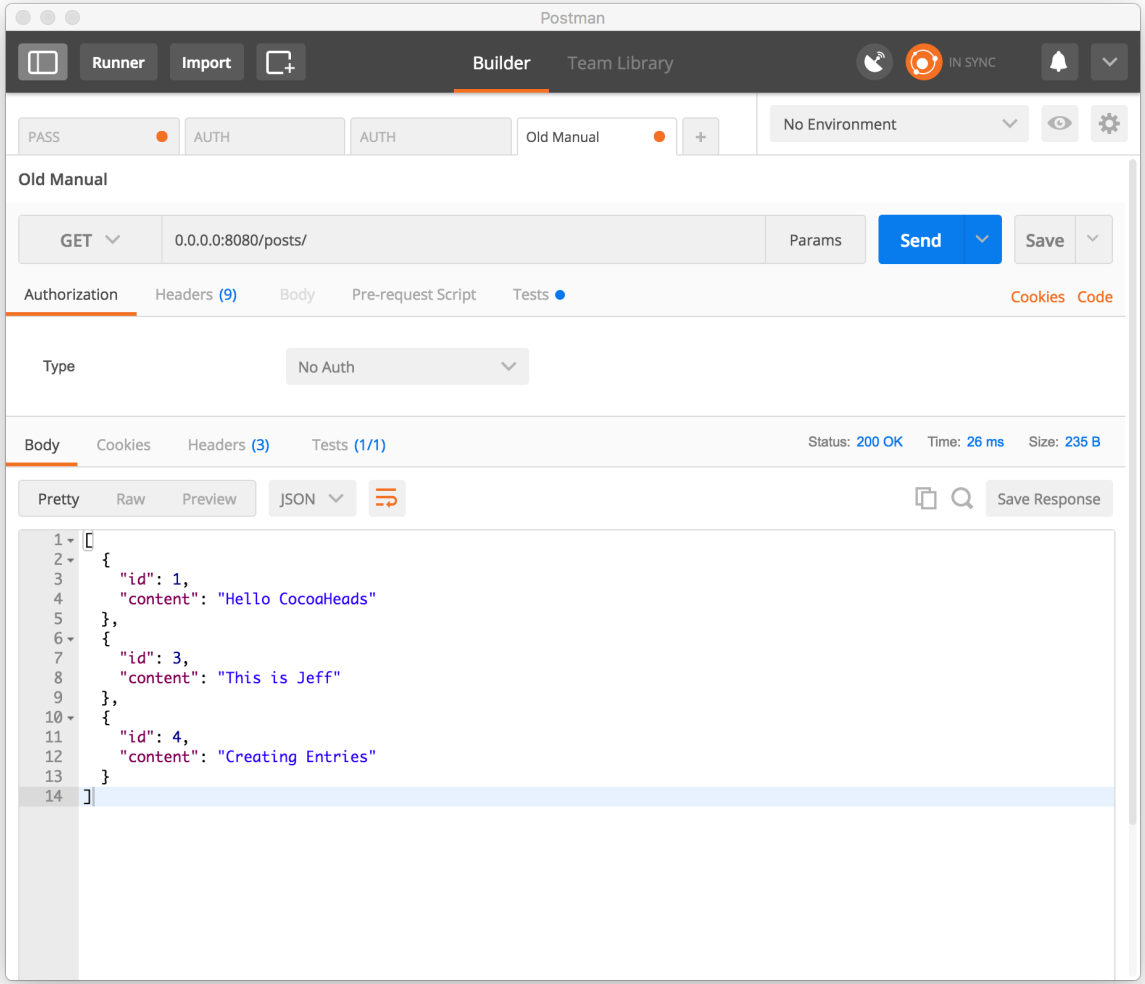
Xcode

Dig in


Postman
Runner
Import
Builder
Team Library
IN SYNC
PASS
AUTH
AUTH
Old Manual
No Environment
Old Manual
DELETE
0.0.0.0:8080/posts/2
Params
Send
Save
Authorization
Headers (9)
Body
Pre-request Script
Tests
Cookies
Code
Type
No Auth
Body
Cookies
Headers (2)
Tests
Status: 200 OK
Time: 33 ms
Size: 75 B
Pretty
Raw
Preview
Text
Save Response
1

Xcode

Dig in


Postman
Runner
Import
Builder
Team Library
IN SYNC
PASS
AUTH
AUTH
Old Manual
No Environment
Old Manual
GET
0.0.0.0:8080/posts/
Params
Send
Save
Authorization
Headers (9)
Body
Pre-request Script
Tests
Cookies
Code
Type
No Auth
Body
Cookies
Headers (3)
Tests (1/1)
Status: 200 OK
Time: 26 ms
Size: 235 B
Pretty
Raw
Preview
JSON
Save Response
[
{
"id": 1,
"content": "Hello CocoaHeads"
},
{
"id": 3,
"content": "This is Jeff"
},
{
"id": 4,
"content": "Creating Entries"
}
]

Heroku

Ok a local setup isn't useful

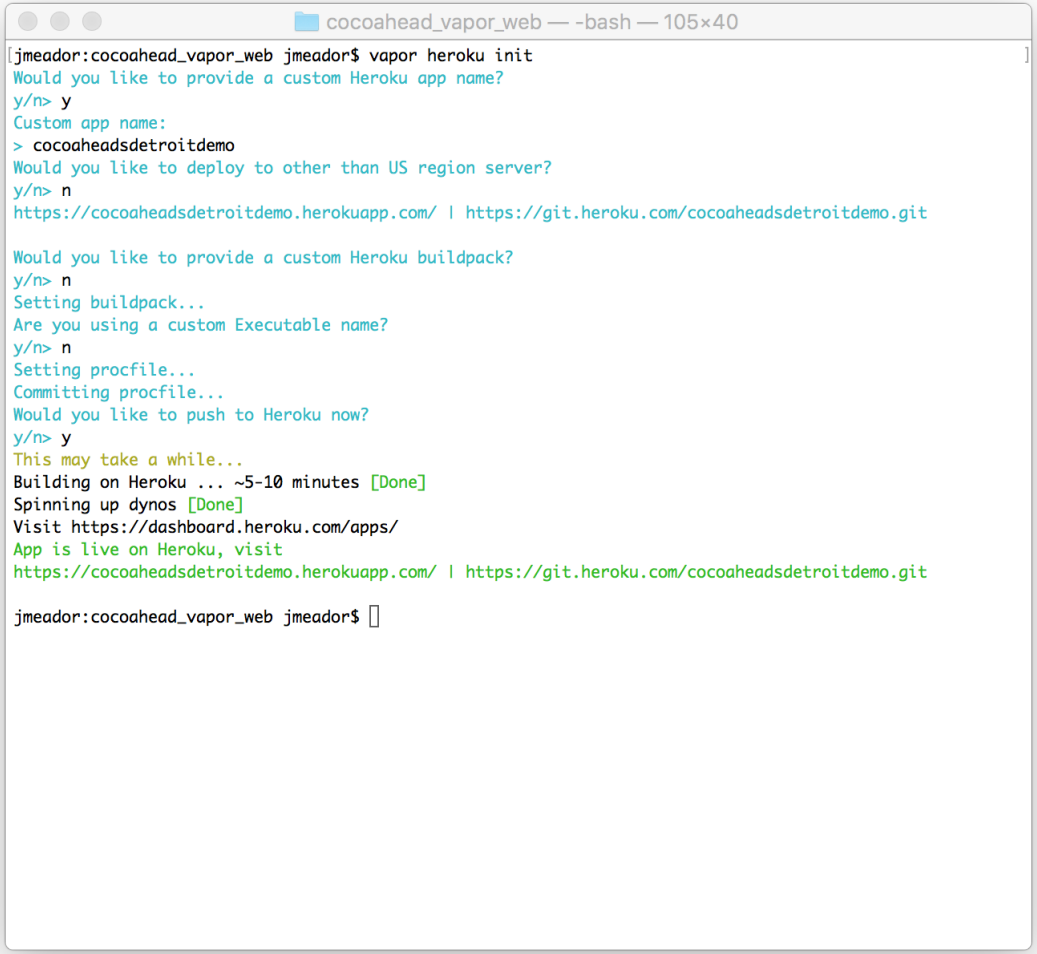
Heroku

We this in a Git repo….wait a second

```
.               Config          Sources
..              Package.pins    Tests
.DS_Store       Package.swift   circle.yml
.build          Public          cocoahead_vapor_web.xcodeproj
.git            README.md       license
.gitignore      Resources
```

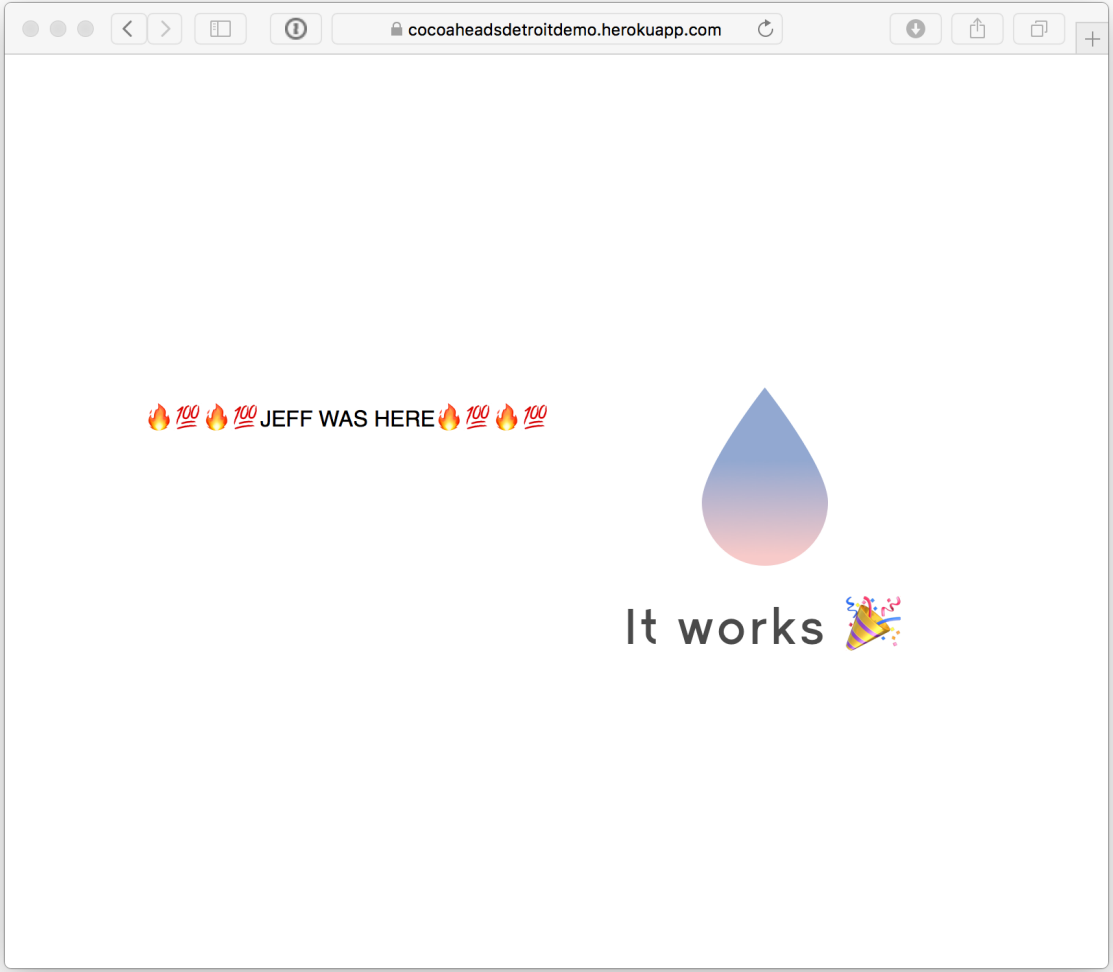
Heroku

Heroku init



Heroku

Heroku init



Thank You

Vectorform  Invent with us.