



Server-Side Swift with Vapor

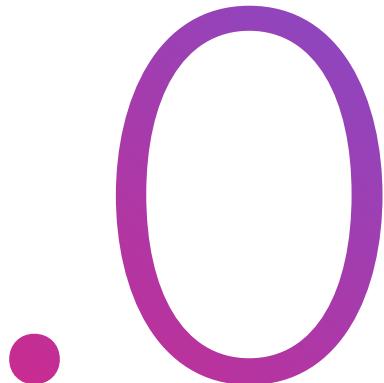
Prepared for CocoaHeads by
Jeff Meador
May 25, 2017



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>



2.0

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
```



3.0

Initialization

Create a Project



Initialization

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

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





Initialization

Web

```
jmeador:web jmeador$ vapor new cocoahead_vapor_web --template=web  
Cloning Template [Done]  
Updating Package Name [Done]  
Initializing git repository [Done]
```

◎ 五言古詩

a web framework for Swift

Project "cocoahead_vapor_web" has been created.
Type `cd cocoahead_vapor_web` to enter the project directory
Enjoy!



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
```



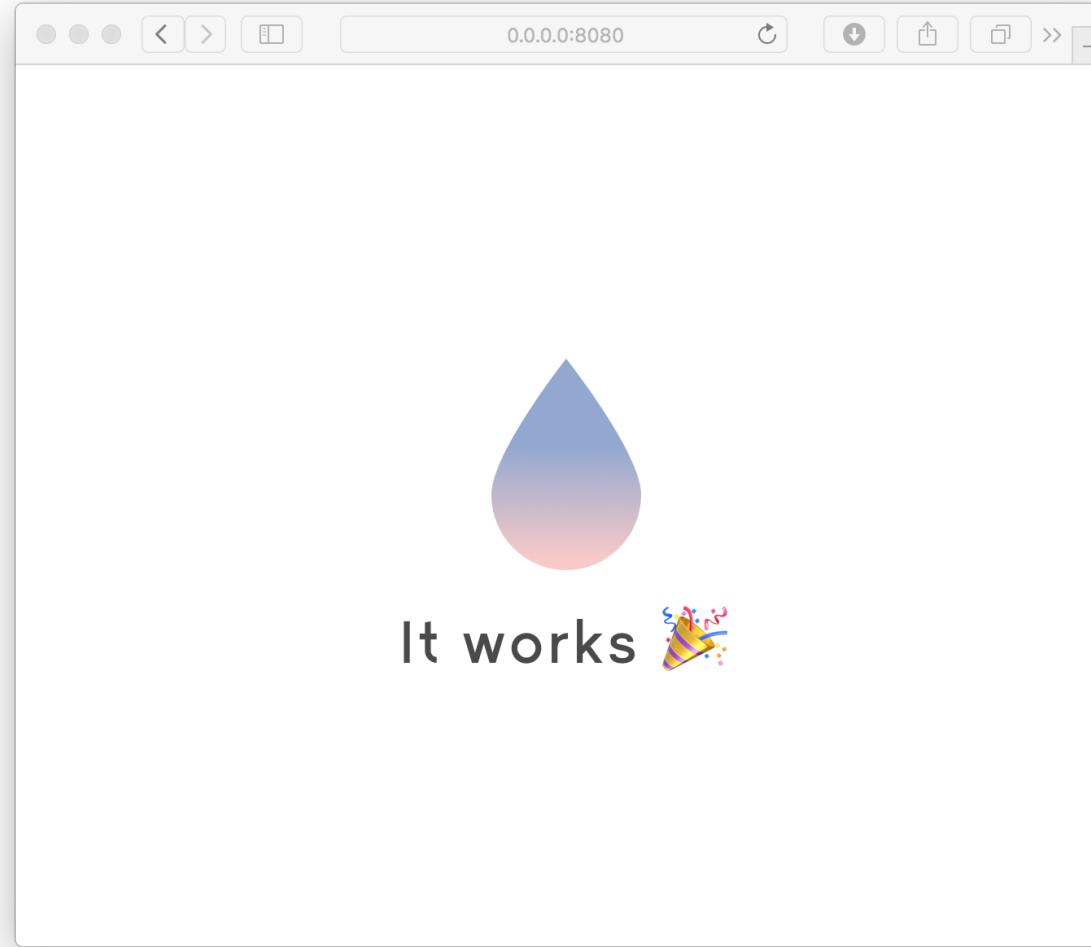
Initialization

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 "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=" 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 "AAAAAAAAAAAAAAAAAAAAA=" 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

```
Request
- Method: GET
- Version: 1.1.0
- URI: http://0.0.0.0:8080/
- Headers:
    Host: 0.0.0.0:8080
    Upgrade-Insecure-Requests: 1
    Connection: keep-alive
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_4)
AppleWebKit/603.1.30 (KHTML, like Gecko) Version/10.1 Safari/603.1.30
    Accept-Language: en-us
    Accept-Encoding: gzip, deflate
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
- Body:
```



4.0

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.
```

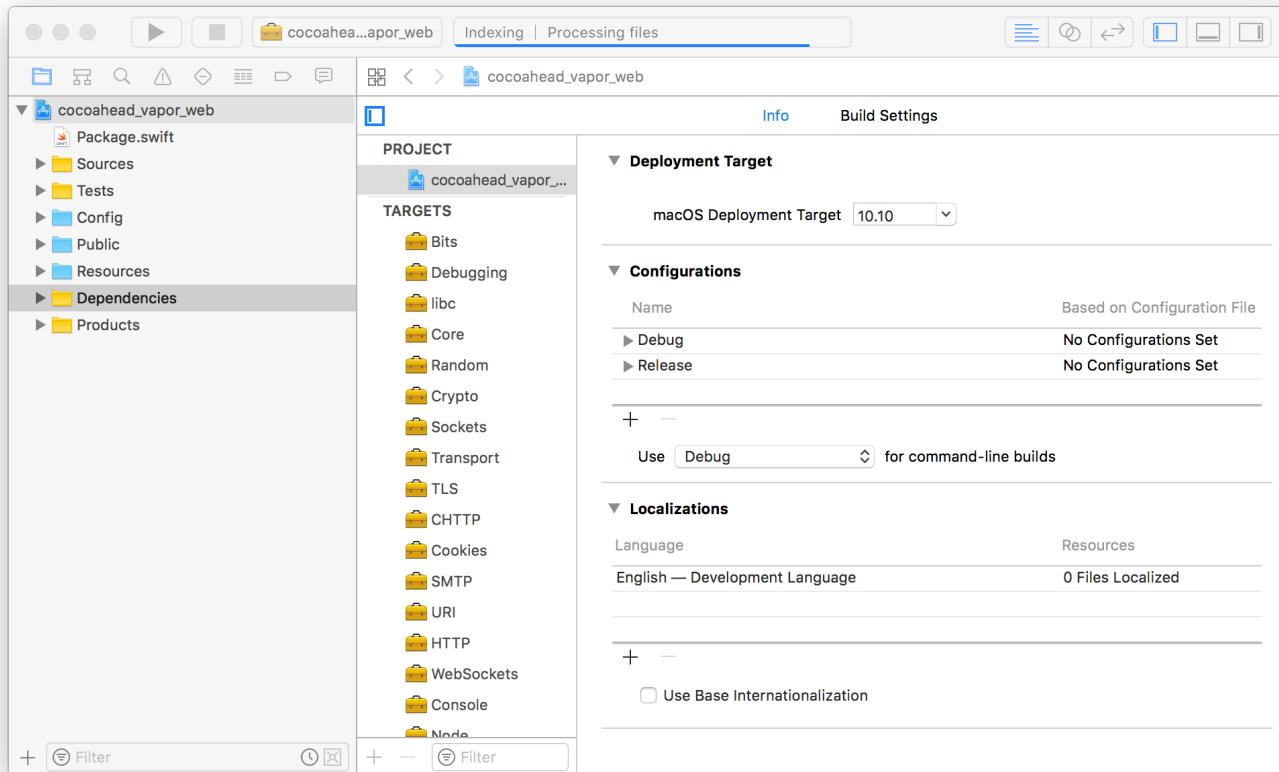
```
Open Xcode project?
```

```
y/n> yes
```

```
Opening Xcode project...
```



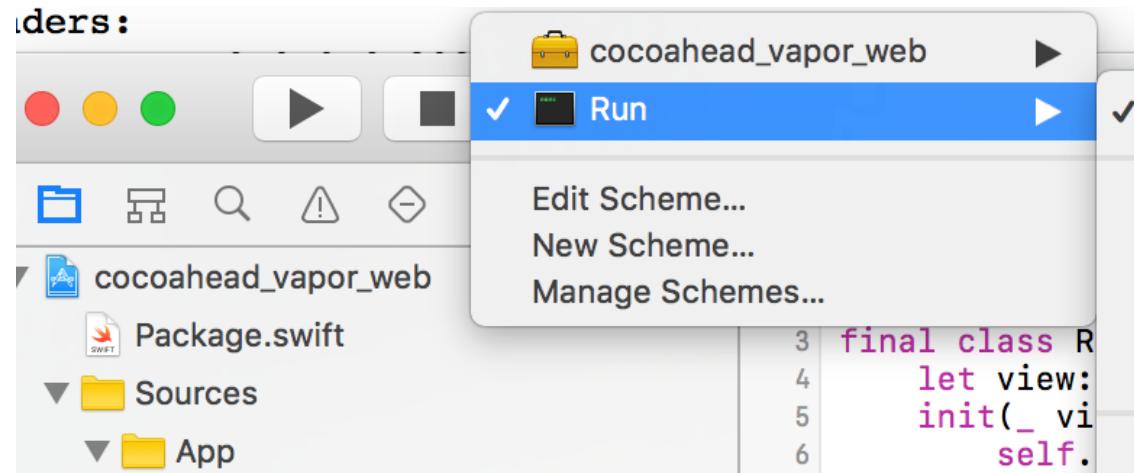
Xcode





Xcode

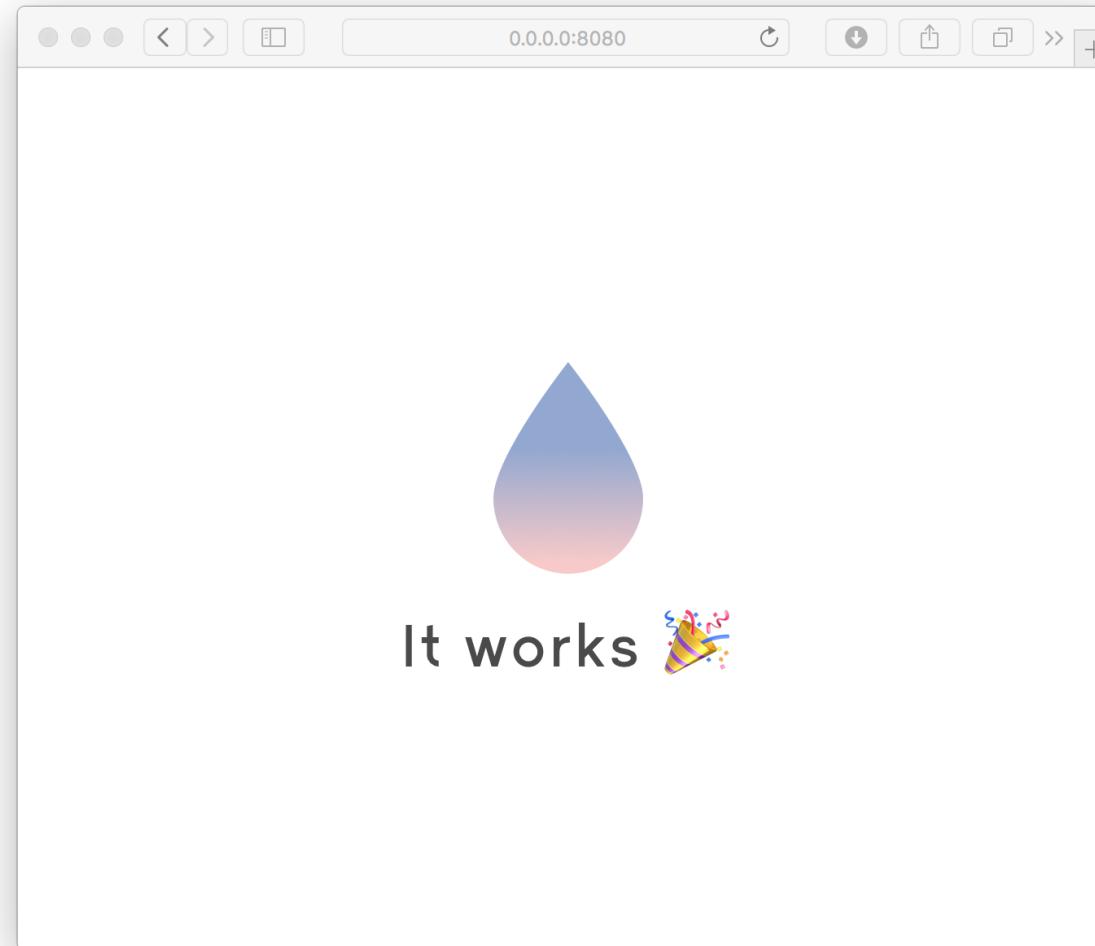
Use Run Scheme





Xcode

Open 0.0.0.0:8080





Xcode

Dig in

The screenshot shows the Xcode interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations, run, stop, and build.
- Search Bar:** "Running Run : Run" in the top right.
- File Navigator:** Shows the project structure under "cocoahead_vapor_web".
- Editor:** Displays the content of `Routes.swift`.

```
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

The screenshot shows the Xcode interface with a project named "cocoahead_vapor_web". The left sidebar displays the project structure:

- Project: 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

The main editor area shows the content of the "welcome.leaf" file:

```
#extend("base")
#export("title") { It works }

#export("content") {
    <div class="welcome">
        
    </div>
}
```

The status bar at the bottom shows the build configuration is set to "My Mac" and the build log says "Running Run : Run".



Xcode

Dig in

Xcode interface showing the project structure and a running instance.

Project Structure:

- 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

Code View:

```
1 #extend("base")
2
3 #export("title") { It works }
4
5 #export("content") {
6     <div class="welcome">
7         🔥💯🔥💯 JEFF WAS HERE🔥💯🔥💯
8         
9     </div>
10 }
```

Output View:

🔥💯🔥💯 JEFF WAS HERE🔥💯🔥💯

It works



Xcode

Dig in

The screenshot shows the Xcode interface with the following details:

- Toolbar:** Standard Xcode toolbar with icons for file operations, run, stop, and build.
- Search Bar:** "Running Run : Run" in the top right.
- File Navigator:** Shows the project structure under "cocoahead_vapor_web".
- Editor:** Displays the content of `Routes.swift`.

```
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")
        }

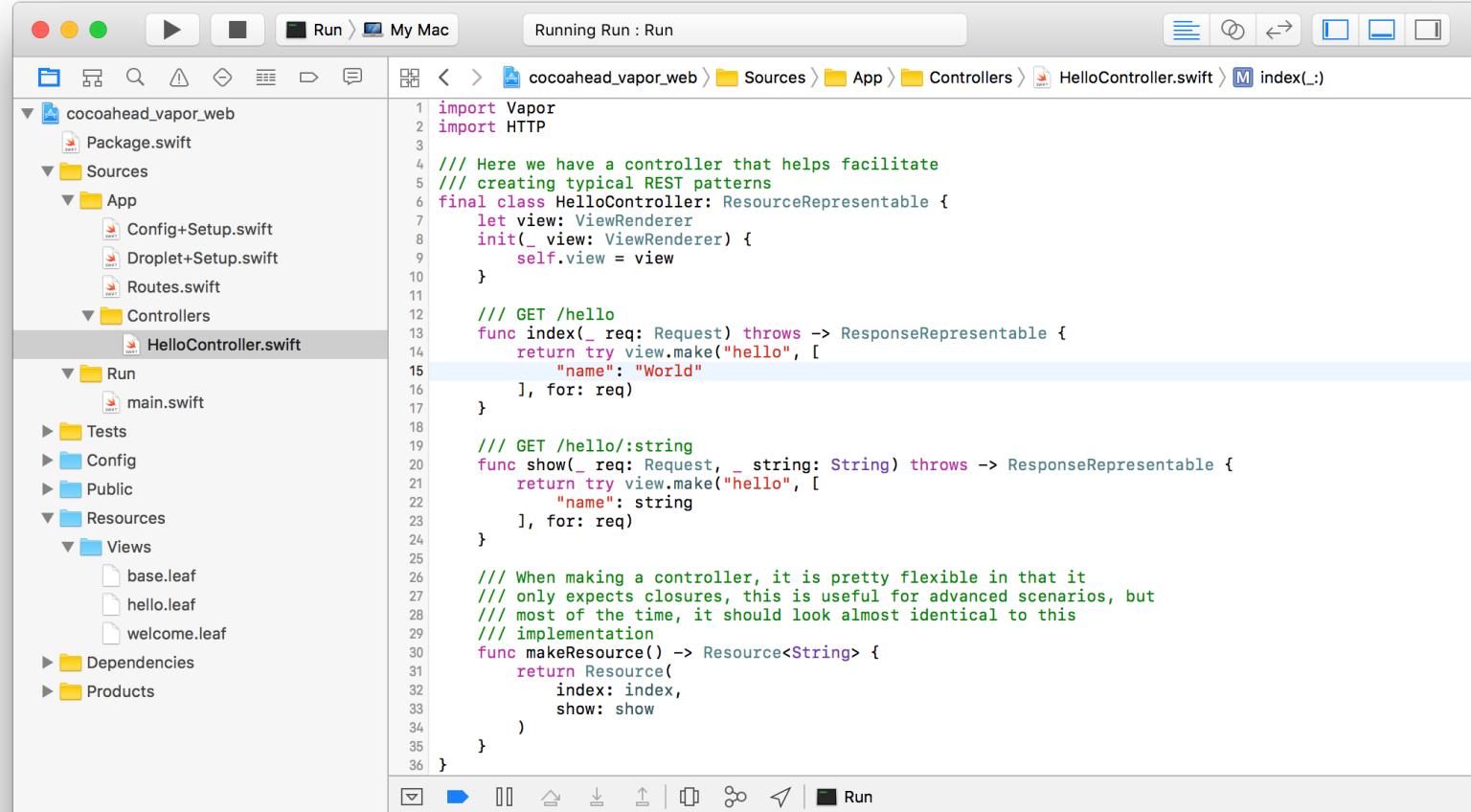
        /// GET /hello/...
        builder.resource("hello", HelloController(view))

        // response to requests to /info domain
        // with a description of the request
        builder.get("info") { req in
            return req.description
        }
    }
}
```

SERVER-SIDE SWIFT

Xcode

Dig in



The screenshot shows the Xcode IDE interface with the following details:

- Project Structure:** The left sidebar displays the project structure of `cocoahead_vapor_web`. It includes a `Package.swift`, a `Sources` folder containing `App` (with `Config+Setup.swift`, `Droplet+Setup.swift`, and `Routes.swift`) and `Controllers` (with `HelloController.swift` selected), a `Run` folder containing `main.swift`, and `Tests`, `Config`, `Public`, `Resources` (containing `Views` with files `base.leaf`, `hello.leaf`, and `welcome.leaf`), `Dependencies`, and `Products`.
- Code Editor:** The main editor pane shows the `HelloController.swift` file. The code defines a `HelloController` class that implements `ResourceRepresentable`. It contains two `func` implementations: `index(_ req: Request) throws -> ResponseRepresentable` and `show(_ req: Request, _ string: String) throws -> ResponseRepresentable`. Both functions return a `try view.make("hello", ["name": "World"], for: req)`. The code also includes a multi-line comment explaining the controller's flexibility.
- Toolbar:** The bottom toolbar contains standard Xcode navigation icons for file operations like Open, Save, Find, and Run.



Xcode

Dig in

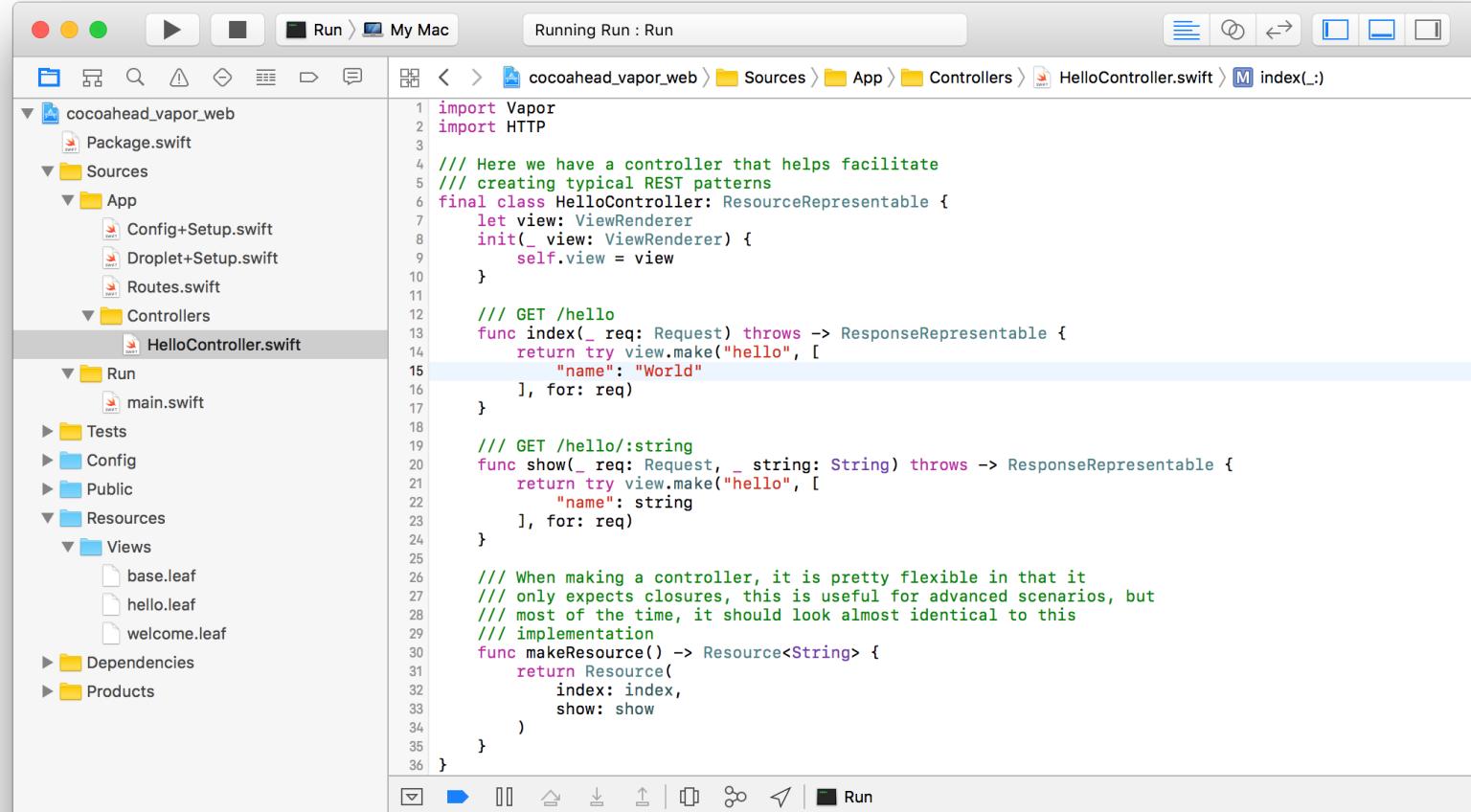
The screenshot shows the Xcode interface with a Vapor project named "cocoahead_vapor_web". The "hello.leaf" file is selected in the "Views" folder under "Resources". The code editor displays the following Leaf template:

```
1 #extend("base")
2
3 #export("title") { Hello, #{name}! }
4
5 #export("content") {
6     <h1>Hello, #{name}!</h1>
7 }
```

SERVER-SIDE SWIFT

Xcode

Dig in



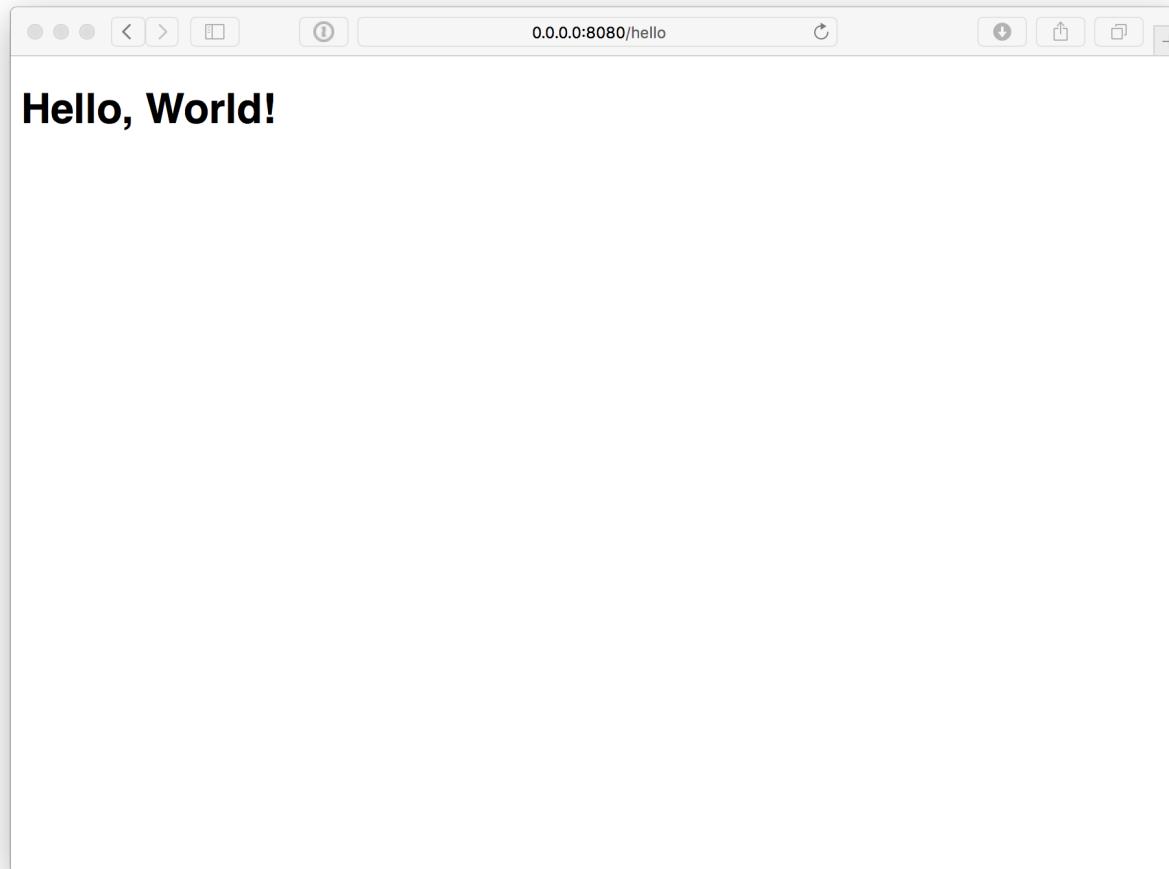
The screenshot shows the Xcode IDE interface with the following details:

- Project Structure:** The left sidebar displays the project structure of `cocoahead_vapor_web`. It includes a `Package.swift`, a `Sources` folder containing `App` (with `Config+Setup.swift`, `Droplet+Setup.swift`, and `Routes.swift`) and `Controllers` (with `HelloController.swift` selected), a `Run` folder containing `main.swift`, and `Tests`, `Config`, `Public`, `Resources` (containing `Views` with files `base.leaf`, `hello.leaf`, and `welcome.leaf`), `Dependencies`, and `Products`.
- Code Editor:** The main editor pane shows the `HelloController.swift` file. The code defines a `HelloController` class that implements `ResourceRepresentable`. It contains two `func` implementations: `index(_ req: Request) throws -> ResponseRepresentable` and `show(_ req: Request, _ string: String) throws -> ResponseRepresentable`. Both functions return a `try view.make("hello", ["name": "World"], for: req)`. The code also includes a multi-line comment explaining the controller's flexibility.
- Toolbar:** The bottom toolbar contains standard Xcode navigation icons for file operations like Open, Save, Find, and Run.



Xcode

Dig in





Xcode

Dig in

The screenshot shows the Xcode interface with a Vapor application project open. The left sidebar displays the project structure:

- cocoahead_vapor_web (Project)
- ↳ 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

The right pane shows the content of `HelloController.swift`:

```
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
        }
    }
}
```

The bottom right corner shows a list of recent API requests:

- 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

0.0.0:8080/hello/CocoaHeads

```

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
        }
    }
}

BasicResponder.respond(to: R...
Protocol witness for Respon...
Router.respond(to: Request) t...
Protocol witness for Respon...
FileMiddleware.respond(to: R...
Protocol witness for Middlewa...
Collection<A where ...>.chai...
BasicResponder.respond(to: ...
Protocol witness for Responde...
DateMiddleware.respond(to: ...
Protocol witness for Middlewa...
Collection<A where ...>.chai...
BasicResponder.respond(to: ...
Protocol witness for Responde...
ErrorMiddleware.respond(to: ...
Protocol witness for Middle...

```

0 HelloController.show(Request, S...throws -> ResponseRepresentable)

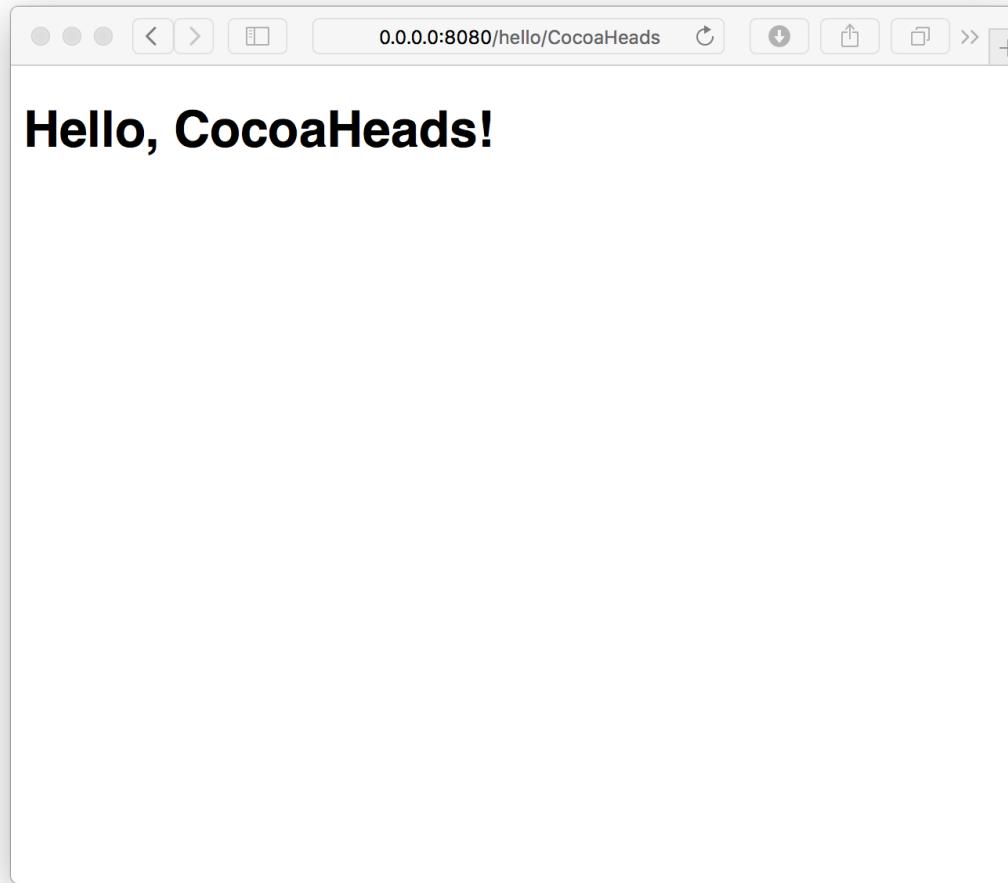
req = (HTTP.Request) 0x00000001032331a0
 string = (String) "CocoaHeads"
 self = (App.HelloController) 0x000000010352a5d0

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

The screenshot shows the Xcode interface with a Vapor project open. The left sidebar displays the project structure:

- 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

The main editor window shows the content of `Routes.swift`:

```
1 import Vapor
2
3 final class Routes: RouteCollection {
4     func build(_ builder: RouteBuilder) throws {
5         builder.get("hello") { req in
6             var json = JSON()
7             try json.set("hello", "world")
8             return json
9         }
10        builder.get("plaintext") { req in
11            return "Hello, world!"
12        }
13        // response to requests to /info domain
14        // with a description of the request
15        builder.get("info") { req in
16            return req.description
17        }
18        builder.get("*") { req in return req.description }
19    }
20    try builder.resource("posts", PostController.self)
21 }
22
23 /// Since Routes doesn't depend on anything
24 /// to be initialized, we can conform it to EmptyInitializable
25 ///
26 /// This will allow it to be passed by type.
27 extension Routes: EmptyInitializable { }
```

The top right shows the URL `0.0.0.8080/hello` and the response `{"hello": "world"}`. The bottom right shows the URL `GET /plaintext`.



```
// 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

Builder Team Library

PASS AUTH AUTH Old Manual IN SYNC

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

The screenshot shows the Postman application window. At the top, there are tabs for 'Runner', 'Import', 'Builder' (which is selected), and 'Team Library'. Below the tabs, there are several status indicators: 'PASS' (red dot), 'AUTH' (green dot), 'AUTH' (green dot), 'Old Manual' (red dot), and 'IN SYNC'. A dropdown menu shows 'No Environment'. The main area is titled 'Old Manual' and contains a 'GET' request to '0.0.0.0:8080/posts'. The 'Authorization' tab is selected, showing 'Type' set to 'No Auth'. Below the request, the response details are shown: 'Status: 200 OK', 'Time: 32 ms', and 'Size: 124 B'. The response body is displayed in a large text area with tabs for 'Pretty', 'Raw', 'Preview', and 'JSON', with 'Pretty' selected. The response body is currently empty, showing only the number '1'.



Xcode

Dig in

The screenshot shows the Xcode interface with the project "cocoahead_vapor_api" open. The left sidebar displays the file structure:

- 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

The code editor shows the content of `Post.swift`:

```
45 static func revert(_ database: Database) throws {
46     try database.delete(self)
47 }
48
49 // MARK: JSON
50
51 // How the model converts from / to JSON.
52 // For example when:
53 //   - Creating a new Post (POST /posts)
54 //   - Fetching a post (GET /posts, GET /posts/:id)
55 //
56 extension Post: JSONConvertible {
    convenience init(json: JSON) throws {
        try self.init(
            content: json.get("content")
        )
    }
}
57
58 func makeJSON() throws -> JSON {
59     var json = JSON()
60     try json.set("id", id)
61     try json.set("content", content)
62     return json
}
63
64 // MARK: HTTP
65
66 // This allows Post models to be returned
67 // directly in route closures
68 extension Post: ResponseRepresentable { }
```

The bottom right pane shows the output of the build process:

```
GET /posts
POST /posts
POST /posts
POST /posts
POST /posts
GET /posts
```



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

```
1 {  
2   "id": 1,  
3   "content": "Hello CocoaHeads"  
4 }
```



Xcode

Dig in

Running Run : Run

My Mac

Postman

Runner Import Builder Team Library IN SYNC

No Environment

Old Manual

POST 0.0.0:8080/posts/ Params Sending... Save Cookies Code

Authorization Headers (9) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary Text

1 { "content" : "Creating Entries" }

Thread 2: breakpoint 2.1

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(addMethod, [St...

8 BasicResponder.respond(to : R...

9 protocol witness for Responde...

10 Router.respond(to : Request)...

11 protocol witness for Responde...

12 FileMiddleware.respond(to : R...

13 protocol witness for Middlewa...

14 Collection<A where ...>.(chai...

15 BasicResponder.respond(to : ...

16 protocol witness for Responde...

17 DateMiddleware.respond(to : ...

18 protocol witness for Middlewa...

19 Collection<A where ...>.(chai...

20 BasicResponder.respond(to : ...

21 protocol witness for Responde...

22 ErrorMiddleware.respond(to : ...

23 protocol witness for Middlew...

24 Collection<A where ...>.(chai...

25 BasicResponder.respond(to : ...

26 protocol witness for Responde...

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)

Cancel Request

Loading...



Xcode

Dig in

My Mac Running Run : Run

Memory 4.5 MB
Energy Impact Zero
Disk Zero KB/s
Network Zero KB/s
Thread 1 Queue: com.apple.hread (serial)
Thread 2 Queue: code... (concurrent)

```

1 import Vapor
2 import HTTP
3
4 /// Here we have a controller that helps facilitate
5 /// RESTful interactions with our Posts table
6 final class PostController: ResourceRepresentable {
7     /// When users call 'GET' on '/posts'
8     /// it should return an index of all available posts
9     func index(request: Request) throws -> ResponseRepresentable {
10         return try Post.all().makeJSON()
11    }
12
13     /// When consumers call 'POST' on '/posts' with valid JSON
14     /// create and save the post
15     func create(request: Request) throws -> ResponseRepresentable {
16         let post = try request.post()
17         try post.save()
18         return post
19    }
20
21     /// When the consumer calls 'GET' on a specific resource, ie:
22     /// '/posts/13rid8B' we should show that specific post
23     func show(request: Request, post: Post) throws -> ResponseRepresentable {
24        return post
25    }
26
27     /// When the consumer calls 'DELETE' on a specific resource, ie:
28     /// '/posts/12jd9' we should remove that resource from the database
29     func delete(request: Request, post: Post) throws -> ResponseRepresentable {
30        try post.delete()
31        return Response(status: .ok)
32    }
33
34     /// When the consumer calls 'DELETE' on the entire table, ie:
35     /// '/posts/' we should remove the entire table
36     func clear(request: Request) throws -> ResponseRepresentable {
37        try Post.makeQuery().delete()
38        return Response(status: .ok)
39    }
40
41    // MARK: - API
42
43    /// POST /posts/
44    func POST(_ req: Request) throws -> ResponseRepresentable {
45        let post = try req.content.decode(Post.self)
46        try post.save()
47        return post
48    }
49
50    // MARK: - Endpoints
51
52    /// GET /posts
53    func index(_ req: Request) throws -> ResponseRepresentable {
54        return try Post.all().makeJSON()
55    }
56
57    /// GET /posts/{id}
58    func show(_ req: Request) throws -> ResponseRepresentable {
59        let id = try req.parameters.get("id")
60        let post = try Post.query(on: req).filter("id", equals: id).first()
61        return post ?? Response(status: .notFound)
62    }
63
64    /// DELETE /posts/{id}
65    func delete(_ req: Request) throws -> ResponseRepresentable {
66        let id = try req.parameters.get("id")
67        let post = try Post.query(on: req).filter("id", equals: id).first()
68        try post?.delete()
69        return post ?? Response(status: .notFound)
70    }
71
72    // MARK: - Configuration
73
74    static var configuration: Configuration {
75        return .init(
76            environment: .development,
77            services: .init(),
78            routes: routes
79        )
80    }
81
82    static var routes: [Route] {
83        return [
84            Get("/", PostController.self, "index"),
85            Get("/posts/{id}", PostController.self, "show"),
86            Post("/posts/", PostController.self, "POST"),
87            Delete("/posts/{id}", PostController.self, "delete")
88        ]
89    }
90
91    static var middlewares: [MiddlewareType] {
92        return [
93            FileMiddleware(.public),
94            ErrorMiddleware(),
95            DateMiddleware(),
96            BasicAuthMiddleware()
97        ]
98    }
99
100   static var Droplet: Droplet {
101       return Droplet.init(
102           config: .init(
103               port: 8080,
104               host: "0.0.0.0"
105           ),
106           middleware: middlewares,
107           routes: routes,
108           environment: .development
109       )
110   }
111 }
```

0 PostController.create(request:...ws -> ResponseRepresentable)

request = (HTTP.Request) 0x00000000101811a60
 self = (App.PostController) 0x00000000101d3f750
 post = (App.Post) 0x00000000101b3f360

(lldb)

Postman

Runner Import Builder Team Library

PASS AUTH AUTH Old Manual No Environment

Old Manual

POST 0.0.0.0:8080/posts/ Sending... Save Cookies Code

Authorization Headers Body Pre-request Script Tests

Body 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 + ⚡ IN SYNC

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 C+ 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

The screenshot shows the Postman application interface. At the top, there are tabs for 'Runner', 'Import', 'Builder' (which is selected), and 'Team Library'. Below the tabs, there's a toolbar with buttons for 'PASS', 'AUTH', 'AUTH', 'Old Manual' (with a red dot), and 'No Environment'. A '+' button is also present. The main workspace is titled 'Old Manual'. It shows a 'DELETE' request to the URL '0.0.0.0:8080/posts/2'. The 'Params' tab is visible, along with 'Send' and 'Save' buttons. Below the URL, there are tabs for 'Authorization', 'Headers (9)', 'Body' (which is selected), 'Pre-request Script', and 'Tests'. The 'Body' tab shows a dropdown for 'Type' set to 'No Auth'. Under the 'Body' tab, there are tabs for 'Body', 'Cookies', 'Headers (2)', and 'Tests'. The status bar at the bottom indicates a 'Status: 200 OK', 'Time: 33 ms', and 'Size: 75 B'. Below these tabs, there are buttons for 'Pretty', 'Raw', 'Preview', 'Text', and 'Save Response'. A preview area shows the number '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

```
1 [ ]
2 {
3   "id": 1,
4   "content": "Hello CocoaHeads"
5 },
6 {
7   "id": 3,
8   "content": "This is Jeff"
9 },
10 {
11   "id": 4,
12   "content": "Creating Entries"
13 }
14 ]
```



5.
.0

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

```
cocoahead_vapor_web — bash — 105x40
[jmeador:cocoahead_vapor_web jmeador$ vapor heroku init
Would you like to provide a custom Heroku app name?
y/n> y
Custom app name:
> cocoaheadsdetroitdemo
Would you like to deploy to other than US region server?
y/n> n
https://cocoaheadsdetroitdemo.herokuapp.com/ | https://git.heroku.com/cocoaheadsdetroitdemo.git

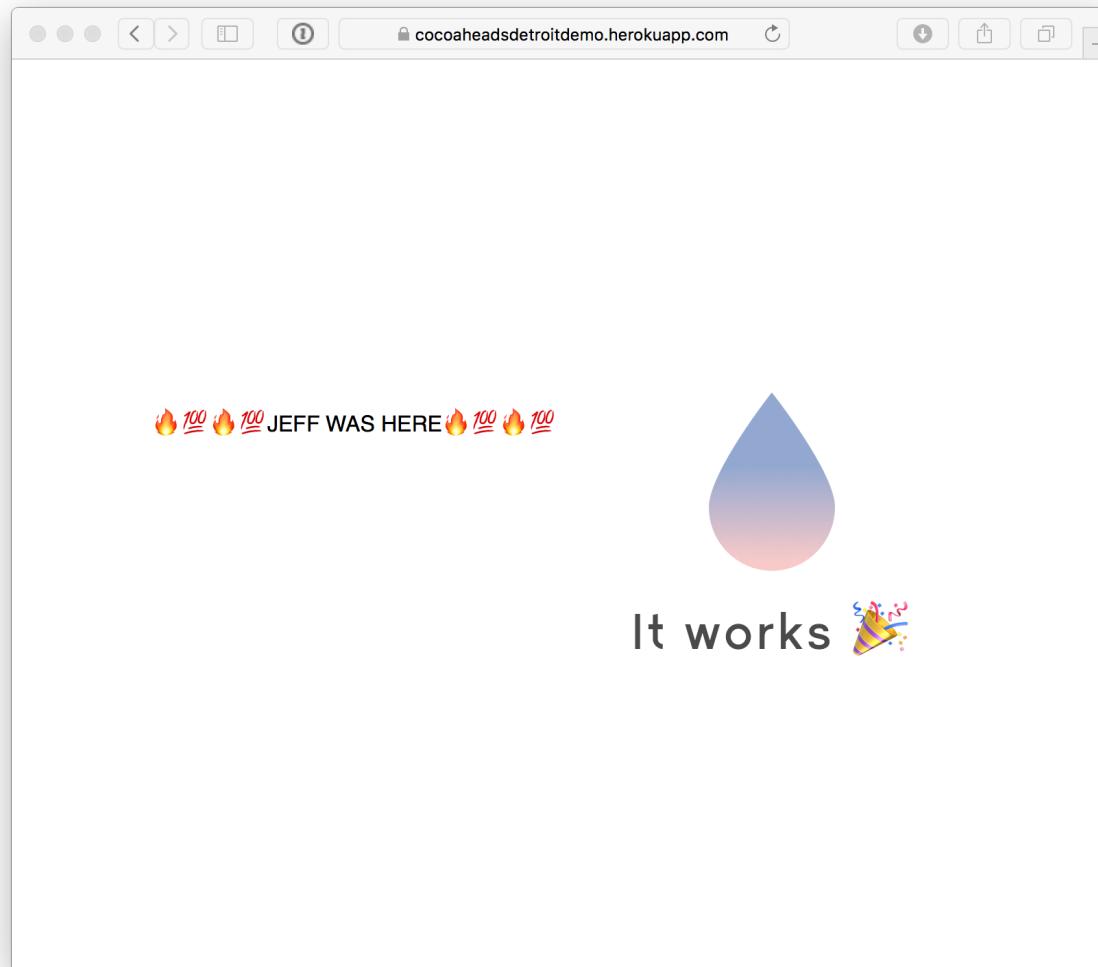
Would you like to provide a custom Heroku buildpack?
y/n> n
Setting buildpack...
Are you using a custom Executable name?
y/n> n
Setting procfile...
Committing procfile...
Would you like to push to Heroku now?
y/n> y
This may take a while...
Building on Heroku ... ~5-10 minutes [Done]
Spinning up dynos [Done]
Visit https://dashboard.heroku.com/apps/
App is live on Heroku, visit
https://cocoaheadsdetroitdemo.herokuapp.com/ | https://git.heroku.com/cocoaheadsdetroitdemo.git

jmeador:cocoahead_vapor_web jmeador$ ]
```



Heroku

Heroku init



Thank You

Vectorform



Invent with us.