Soaring Swiftly

Server Side Swift

#trySwiftConf 2016

Caesar Wirth

- iOS Developer for 5 Years
- Works at CyberAgent, Inc.
  - Most recent app released 3 days ago
- Organizes Conferences
  - Example: try! Swift
  - Maybe you've heard of it?

@cjwirth cjwirth cjwirth.com

Goals

1. Write Web Server in Swift
    - MVP - Minimum Viable Pokédex
2. Deploy Server
3. ???
4. Profit

HTTP Requests from the Client Side

```
func fetchPokedex(completion: ([Pokemon]? -> Void)) {
    let url = NSURL(string: "https://api.server.com/pokemon")!
    let request = NSURLRequest(URL: url)
    let session = NSURLSession.sharedSession()

    let task = session.dataTaskWithRequest(request) { data, _, _ in
        let maybePokemon = Pokemon.pokemonFromData(data)
        completion(maybePokemon)
    }

    task.resume()
}
```


What about the other side?

Ruby: Sinatra

```ruby
# Route to show all Posts, ordered like a blog
get '/posts' do
  content_type :json
  @things = Post.all(:order => :created_at.desc)

  @things.to_json
end
```

node.js: Express

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

OH @ #tryswiftconf: I am Ruby and JavaScript developer. I don't like type.

– @ayanonagon, Ayaka Nonaka (March 2, 2016)

I am a Swift developer.
I  *types.*

– Caesar Wirth (literally 2 seconds ago)

What makes an HTTP Request?

```
protocol RequestType {
    var method: Method { get }
    var path: String { get }
    var headers: [Header] { get }
    var body: String? { get }
}

protocol ResponseType {
    var status: Status { get }
    var headers: [Header] { get }
    var body: String? { get }
}
```

Concrete Type to Instantiate

```
struct Response: ResponseType {
    var status: Status
    var headers: [Header] = []
    var body: String?

    init(status: Status, body: String) {
        self.status = status
        self.body = body
    }
}
```

```
typealias Header = (String, String)

enum Method: String {
    case Get = "GET"
    case Post = "POST"
    case Put = "PUT"
    case Delete = "DELETE"
    ...
}

enum Status: Int {
    case OK = 200
    case NotFound = 404
    case ServerError = 500
    ...
}
```

```
typealias ServerType = (RequestType -> ResponseType)
```

```
typealias ServerType = (RequestType -> ResponseType)
```

- Server takes requests and returns responses
- Protocols allow multiple types of responses
  - JSON
  - HTML
  - Errors
- Can run concurrently ...right?


Confession

Hand-Wavy Magic ✨👐✨

```
// Defines RequestType, ResponseType Protocols
// With this, we can abstract away low-level details and app code
import Nest

// Takes care of socket IO, threading, request parsing, etc
import Curassow

// Parsing and Formatting JSON
import Jay
```

- Write

```
// main.swift

import Curassow
// func serve(closure: RequestType -> ResponseType)

serve { _ in
    return Response(status: .OK, body: "Hello World!")
}
```

- Compile

```
$ swift build
```

- Run

```
$ .build/debug/App
[INFO] Listening at http://0.0.0.0:8000 (35238)
[INFO] Booting worker process with pid: 35239

$ curl http://localhost:8000
Hello World!
```

Routing

A single **`ServerType`** won't cover it

We need some way to handle different requests in different places

`GET /users` *and* **`POST /posts`** should not be handled in the same place

```
router.get("/pokemon") { reqest in
    // Do Stuff!
}
```

```
typealias Handler = (RequestType throws -> ResponseType)

class Router {
    private var routes: [String: [Method: Handler]] = [:]

    func route(method: Method, path: String, handler: Handler) { }

    func handle(request: RequestType) -> ResponseType { }
}
```

```
typealias Handler = (RequestType throws -> ResponseType)

class Router {
    private var routes: [String: [Method: Handler]] = [:]

    func route(method: Method, path: String, handler: Handler) {
        if let existing = routes[path] {
            var editing = existing
            editing[method] = handler
            routes[path] = editing
        } else {
            routes[path] = [method: handler]
        }
    }
    ...
}
```

```
class Router {
    ...
    func handle(request: RequestType) -> ResponseType {
        // TODO: Get URL Parameter Parsing
        // eg. /users/:id --> will have a String parameter called "id"
        guard let method = request.HTTPMethod,
            let route = routes[request.path]?[method] else {
                return Error.NotImplemented
        }

        do {
            return try route(request)
        } catch {
            return Error.InternalServerError
        }
    }
}
```

Did you notice?

```
typealias ServerType = (RequestType -> ResponseType)

class Router {
    func handle(request: RequestType) -> ResponseType { }
}
```

`Router().handle` *is a* **`ServerType`**

Now we can do this

```
let router = Router()
let server = router.handle

router.route(.Get, "/pokemon") { _ in
    return Response(.OK, body: "Pikachu")
}

serve(server)
```

Demo

多分動くと思うから、リリースしようぜ！

- Mark Zuckerberg

Deploying to Heroku
What's Needed

1. swiftenv
2. .swift-version file
3. Procfile
4. heroku-buildpack-swift

.swift-version

```
DEVELOPMENT-SNAPSHOT-2016-02-25-a
```

Procfile

```
web: App --workers 1 --bind 0.0.0.0:$PORT
```

Create Heroku Application

```
$ heroku create --buildpack https://github.com/
kylef/heroku-buildpack-swift.git
$ heroku ps:scale web=1
```

Deploy!

```
$ git push heroku master
```

Don't Reinvent the Wheel

- Perfect by PerfectlySoft
  - 🎖 Most Mature Award
  - 🏅 Easiest Setup Award
- Kitura by IBM
  - ⚙️ Modular Modules Award
  - 🍼 Baby Award (first commit 25 days ago)
- Vapor
  - 😜 Most-Like-This-Talk Award


PerfectlySoft Inc @perfectlysoft · 7h
Thanks to #Tokyo #tryswiftconf attendees today for helping us cross 6000 stars on GitHub! github.com/PerfectlySoft/…
#swiftlang
6
10
View summary


PerfectlySoft Inc @perfectlysoft · 7h
今日は、たくさん出席者のみなさまからgithubにスターをいただきましてありがとうございます。これからもよろしくお願いいたします。#tryswiftconf
github.com/PerfectlySoft/…
4
14
View summary

References

- Hello Server Side Swift - Logan Wright
- Kyle Fuller
  - Wrote both `swiftenv` and `heroku-buildpack-swift`
- @zoonref - Made #tryPokemonConf logo

Thanks!

ご清聴ありがとうございました