Swift Server Setup

Perfect Server Setup

1. docker pull ubuntu:14.04
2. apt-get update
3. apt-get install clang-3.4 libicu-dev openssl libssl-dev uuid-dev libicu-dev libedit git vim
4. check the ubuntu version
5. https://swift.org/builds/swift-3.0.1-release/ubuntu1404/swift-3.0.1-RELEASE/swift-3.0.1-RELEASE-ubuntu14.04.tar.gz
6. tar xzf swift-3.0.1-RELEASE-ubuntu14.04.tar
7. swift —version
8. git clone https://github.com/PerfectlySoft/PerfectTemplate.git
9. cd PerfectTemplate
10. swift build
11. .build/debug/

Basic Hello world Project

```
import PerfectLib
import PerfectHTTP
import PerfectHTTPServer

// Create HTTP server.
let server = HTTPServer()

// Register your own routes and handlers
var routes = Routes()
routes.add(method: .get, uri: "/", handler: {
        request, response in
        response.setHeader(.contentType, value: "text/html")
        response.appendBody(string: "<html><title>Hello, world!</title><body>Hello, world!</
body></html>")
        response.completed()
    }
```

Swift Server Setup

```
)

// Add the routes to the server.
server.addRoutes(routes)

// Set a listen port of 8181
server.serverPort = 8181

do {
    // Launch the HTTP server.
    try server.start()
} catch PerfectError.networkError(let err, let msg) {
    print("Network error thrown: \(err) \(msg)")
}
```

Build and run the project again with:

```
swift build
.build/debug/MyAwesomeProject
```

The server is now running and it's waiting for connections! 🎉 Access http://localhost:8181/ to see the greeting. Hit "control-c" to terminate the server.

Refer to the examples from below link

https://github.com/PerfectExamples/Perfect-JSON-API.git

Kitura Server Setup

1. docker pull ubuntu:14.04
2. apt-get update
3. apt-get install clang libicu-dev openssl libssl-dev uuid-dev libicu-dev libedit git vim
4. check the ubuntu version

Swift Server Setup

5. https://swift.org/builds/swift-3.0.1-release/ubuntu1404/swift-3.0.1-RELEASE/swift-3.0.1-RELEASE-ubuntu14.04.tar.gz
6. tar xzf swift-3.0.1-RELEASE-ubuntu14.04.tar
7. swift —version
8. $ mkdir myFirstProject
9. $ cd myFirstProject
10. $ swift package init --type executable
11. In Package.swift, add Kitura as a dependency for your project i.e .Package(url: "https://github.com/IBM-Swift/Kitura.git", majorVersion: 1, minor: 1)

Basic Hello world Project

In Sources/main.swift, add the following code.

```
import Kitura

// Create a new router
let router = Router()

// Handle HTTP GET requests to /
router.get("/") {
    request, response, next in
    response.send("Hello, World!")
    next()
}

// Add an HTTP server and connect it to the router
Kitura.addHTTPServer(onPort: 8090, with: router)

// Start the Kitura runloop (this call never returns)
Kitura.run()
```

Swift Server Setup

12. swift build
13. .build/debug/myFirstProject

Vapor Server Setup

1. curl -sL swift.vapor.sh/ubuntu | bash
2. sudo apt-get update
3. sudo apt-get install clang libicu-dev binutils git
4. wget https://swift.org/builds/swift-3.0-release/ubuntu1404/swift-3.0-RELEASE/swift-3.0-RELEASE-ubuntu14.04.tar.gz (if OS is Ubuntu 14.04)
5. wget https://swift.org/builds/swift-3.0-release/ubuntu1510/swift-3.0-RELEASE/swift-3.0-RELEASE-ubuntu15.10.tar.gz (id OS is Ubuntu 15.04)
6. curl -sL check.vapor.sh | bash (Double check the installation was successful by running:)
7. curl -sL toolbox.vapor.sh | bash (this installs the tool box)
8. vapor —help
9. vapor self update

Basic Hello world Project

1. vapor new Hello (creating a new project called Hello, World)
2. let drop = Droplet() (in the main.swift file)
3. Right after the creation of drop, add the following code snippet.

```
drop.get("hello") { request in
return "Hello, world!"
}
```

4. drop.run() (At the bottom of the main file, make sure to serve your Droplet.)
5. vapor build (this command compiles the code)
6. vapor run serve (this command boots up the server)
7. you should see a message Server starting.... You can now visit http://localhost:8080/hello in your browser