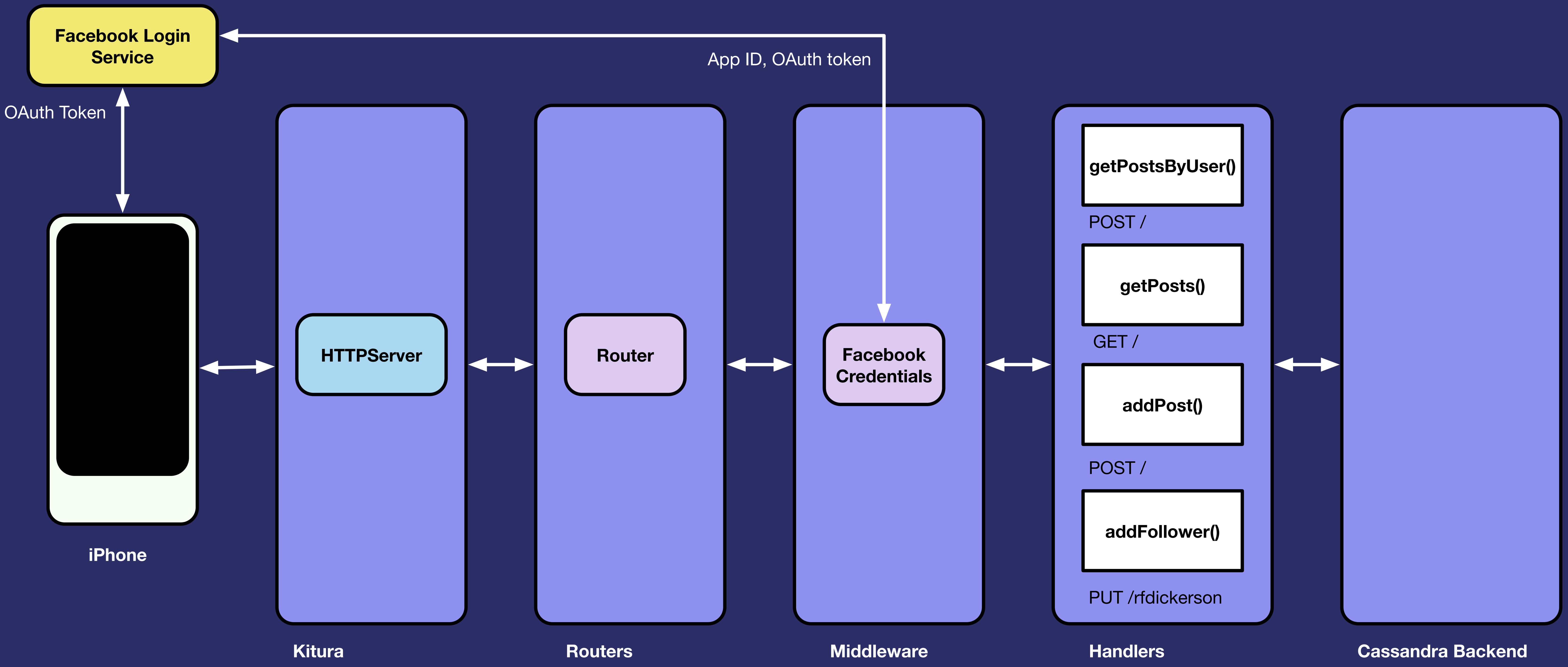
Blitter

Building a Social networking backend in Swift 3


Facebook Login Service
App ID, OAuth token
OAuth Token
iPhone
HTTPServer
Kitura
Router
Routers
Facebook Credentials
Middleware
getPostsByUser()
POST /
getPosts()
GET /
addPost()
POST /
addFollower()
PUT /rfdickerson
Handlers
Cassandra Backend

Steps

→ Set up up project and dependencies

→ Set up routes

→ Add Facebook authentication

→ Set up the model and database

→ Handle the requests

Steps

→ **Set up up project and dependencies**

→ Set up routes

→ Add Facebook authentication

→ Set up the model and database

→ Handle the requests

Create the boilerplate

```
$ ~/> mkdir Blitter && cd Blitter
$ ~/Blitter/> swift package init
```

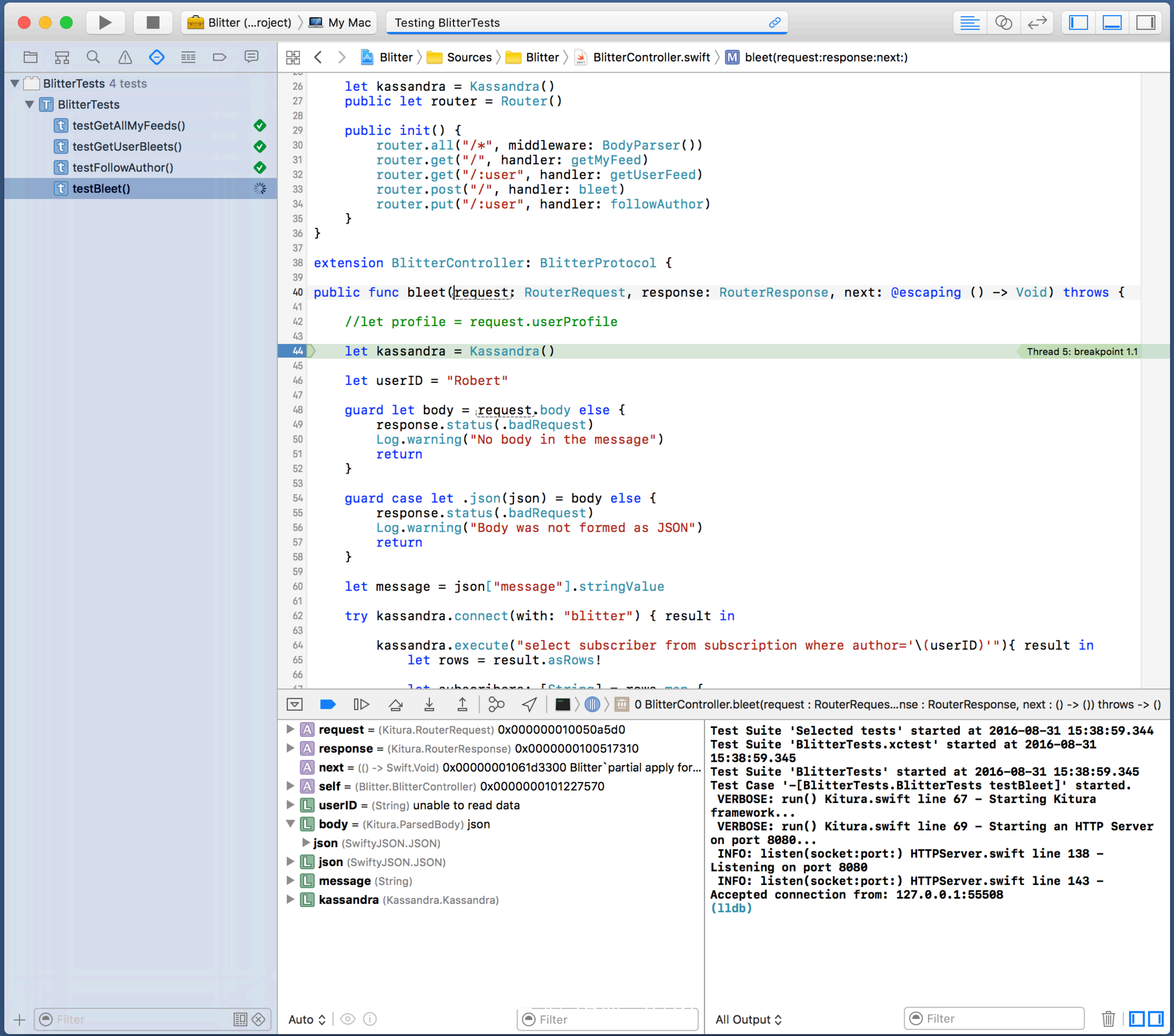
Create the boilerplate

```
$ ~/Blitter/> swift package init
```

```
Creating library package: Blitter
Creating Package.swift
Creating .gitignore
Creating Sources/
Creating Sources/Blitter.swift
Creating Tests/
Creating Tests/LinuxMain.swift
Creating Tests/BlitterTests/
Creating Tests/BlitterTests/BlitterTests.swift
```

If you want to develop in XCode

```
$ ~/Blitter/> swift package generate-xcodeproj
$ ~/Blitter/> open Blitter.xcodeproj
```



```swift
    let kassandra = Kassandra()
    public let router = Router()

    public init() {
        router.all("/*", middleware: BodyParser())
        router.get("/", handler: getMyFeed)
        router.get("/:user", handler: getUserFeed)
        router.post("/", handler: bleet)
        router.put("/:user", handler: followAuthor)
    }
}

extension BlitterController: BlitterProtocol {

public func bleet(request: RouterRequest, response: RouterResponse, next: @escaping () -> Void) throws {

    //let profile = request.userProfile

    let kassandra = Kassandra()

    let userID = "Robert"

    guard let body = request.body else {
        response.status(.badRequest)
        Log.warning("No body in the message")
        return
    }

    guard case let .json(json) = body else {
        response.status(.badRequest)
        Log.warning("Body was not formed as JSON")
        return
    }

    let message = json["message"].stringValue

    try kassandra.connect(with: "blitter") { result in

        kassandra.execute("select subscriber from subscription where author='\(userID)'"){ result in
            let rows = result.asRows!
```

```
request = (Kitura.RouterRequest) 0x000000010050a5d0
response = (Kitura.RouterResponse) 0x0000000100517310
next = (() -> Swift.Void) 0x00000001061d3300 Blitter`partial apply for...
self = (Blitter.BlitterController) 0x0000000101227570
userID = (String) unable to read data
body = (Kitura.ParsedBody) json
  json (SwiftyJSON.JSON)
json (SwiftyJSON.JSON)
message (String)
kassandra (Kassandra.Kassandra)
```

```
Test Suite 'Selected tests' started at 2016-08-31 15:38:59.344
Test Suite 'BlitterTests.xctest' started at 2016-08-31
15:38:59.345
Test Suite 'BlitterTests' started at 2016-08-31 15:38:59.345
Test Case '-[BlitterTests.BlitterTests testBleet]' started.
 VERBOSE: run() Kitura.swift line 67 - Starting Kitura
framework...
 VERBOSE: run() Kitura.swift line 69 - Starting an HTTP Server
on port 8080...
  INFO: listen(socket:port:) HTTPServer.swift line 138 -
Listening on port 8080
  INFO: listen(socket:port:) HTTPServer.swift line 143 -
Accepted connection from: 127.0.0.1:55508
(lldb)
```

Add dependencies

```
// Package.swift
import PackageDescription

let package = Package(
    name: "TwitterClone",
    dependencies: [
        .Package(url: "https://github.com/IBM-Swift/Kassandra",      majorVersion: 0,  minor: 1),
        .Package(url: "https://github.com/IBM-Swift/Kitura.git",     majorVersion: 0,  minor: 27),
        .Package(url: "https://github.com/IBM-Swift/SwiftyJSON.git", majorVersion: 0,  minor: 13)
        ]
)
```

Importing packages

```swift
// main.swift
import Foundation
import Dispatch

import Kitura
import Kassandra
import SwiftyJSON
```

Steps

→ Set up up project and dependencies

→ **Set up routes**

→ Add Facebook authentication

→ Set up the model and database

→ Handle the requests

Basic Routing

```
router.get("/") { request, response, next throws in

  // Get my Feed here

}

router.get("/:user") { request, response, next throws in

  // Get user bleets here
  let user = request.parameters["user"]

}

router.post("/") { request, response, next throws in

   // Add a Bleet here.

}
```

Steps

- → Set up up project and dependencies
- → Set up routes
- → **Add Facebook authentication**
- → Set up the model and database
- → Handle the requests

Adding Credentials middleware:

```
import Credentials
import CredentialsFacebook

let credentials = Credentials()
let facebookCredentials = CredentialsFacebook()

credentials.register(fbCredentials)
```

Using the Credentials middlware

```
router.post("/", middleware: credentials)

router.post("/") { request, response, next in
    /// ...
    let profile  = request.userProfile
    let userId   = profile.id                // "robert.dickerson"
    let userName = profile.displayName       // "Robert F. Dickerson"
    /// ...
}
```

Steps

→ Set up up project and dependencies

→ Set up routes

→ Add Facebook authentication

→ **Set up the model and database**

→ Handle the requests

Bleet Model

```swift
struct Bleet {

  let id:             UUID
  let user:           String
  let message:        String
  let postDate:       Date

}

extension Bleet : Model {
  static let tableName = "Bleet"

  // other mapping goes here
}
```

Get the list of Bleets

```swift
func getBleets(oncomplete: ([Bleet]?, Error?) -> Void) {
  try kassandra.connect(with: "blitter") { _ in
    Post.fetch(limit: 50) { bleets, error in

       guard let bleets = bleets else {
          oncomplete( nil, error )
       }

       oncomplete( bleets.flatMap() { Bleet.init() }, nil )
    }

  }
}
```

Save the Bleet

```
let bleet = Bleet(id        : UUID(),
                  user      : userId,
                  body      : "I love Swift!",
                  timestamp : Date()
                  )

try kassandra.connect(with: "blitter") { _ in
    bleet.save()
}
```

Steps

→ Set up up project and dependencies

→ Set up routes

→ Add Facebook authentication

→ Set up the model and database

→ **Handle the requests**

Get back Blitter feed

```
getBleets { bleets, error in

    guard let bleets = bleets else {
        response.status(.badRequest).send()
        response.next()
        return
    }

    response.status(.OK)
        .send(json: JSON(bleets.toDictionary()))
        response.next()
    }
}
```

Save a post

```
router.post("/") { request, response, next throws in

    guard let httpBody = request.body else { /* ... */ }
    guard case let .json(json) = body else { /* ... */ }
    guard let message = json["message"].stringValue { /* ... */ }

    let bleet = Bleet(id: UUID(), message, Date(), userId)
    saveBleet(bleet)
        .onSuccess {
            response.status(.OK).send().end()
        }
}
```

See it in action