Documentation Language: Swift

Hummingbird / Router

Class

Router

Create rules for routing requests and then create `Responder` that will follow these rules.

```
final class Router<Context> where Context : RequestContext
```

Overview

`Router` requires an implementation of the `on(path:method:use)` functions but because it also conforms to `RouterMethods` it is also possible to call the method specific functions `get`, `put`, head, `post` and `patch`. The route handler closures all return objects conforming to `ResponseGenerator`. This allows us to support routes which return a multitude of types eg

```
router.get("string") { _, _ -> String in
    return "string"
}
router.post("status") { _, _ -> HTTPResponse.Status in
    return .ok
}
router.data("data") { request, context -> ByteBuffer in
    return ByteBuffer(string: "buffer")
}
```

The default `Router` setup in `Application` is the `TrieRouter` . This uses a trie to partition all the routes for faster access. It also supports wildcards and parameter extraction

```
router.get("user/*", use: anyUser)
router.get("user/:id", use: userWithId)
```

Both of these match routes which start with “/user” and the next path segment being anything. The second version extracts the path segment out and adds it to `Request.parameters` with the key “id”.

Topics

Structures

`struct RouteDescription`

Route description

Initializers

`init(context: Context.Type, options: RouterOptions)`

Instance Properties

`let middlewares: MiddlewareGroup<Context>`

`var routes: [RouteDescription]`

List of routes added to router

Instance Methods

`func add(middleware: any MiddlewareProtocol<Request, Response, Context>) -> Self`

Add middleware to Router

`func buildResponder() -> RouterResponder<Context>`

build responder from router

`func on<Responder>(RouterPath, method: HTTPRequest.Method, responder: Responder) -> Self`

Add responder to call when path and method are matched

`func validate() throws`

Validate router

Default Implementations

RouterMethods Implementations

Relationships

Conforms To

`HTTPResponderBuilder`, `RouterMethods`

See Also

Router

`struct RouterGroup`

Used to group together routes under a single path. Additional middleware can be added to the endpoint and each route can add a suffix to the endpoint path

`class RouteCollection`

Collection of routes

`protocol RouterMethods`

Conform to `RouterMethods` to add standard router verb (get, post …) methods

`struct RouterOptions`

Router Options

`protocol HTTPResponder`

Protocol for object that produces a response given a request

`protocol HTTPResponderBuilder`

A type that has a single method to build a HTTPResponder

`struct CallbackResponder`

Responder that calls supplied closure

`struct RouterResponder`

`struct EndpointPath`

Endpoint path storage

`struct RouterPath`

Split router path into components

`struct RequestID`

Generate Unique ID for each request