Documentation Language: Swift

Hummingbird / RouterMiddleware

Protocol

RouterMiddleware

Version of `MiddlewareProtocol` whose Input is `Request` and output is `Response`.

```
protocol RouterMiddleware<Context> : Middleware
Protocol where Self.Input == Request, Self.Output == Response
```

Overview

All middleware has to conform to the protocol `RouterMiddleware`. This requires one function `handle(_:context:next)` to be implemented. At some point in this function unless you want to shortcut the router and return your own response you should call `next(request, context)` to continue down the middleware stack and return the result, or a result processed by your middleware.

The following is a simple logging middleware that outputs every URI being sent to the server

```
public struct LogRequestsMiddleware<Context: RequestContext>: RouterMiddlewa
    public func handle(_ request: Request, context: Context, next: (Request,
        // log request URI
        context.logger.log(level: .debug, String(describing:request.uri.path
        // pass request onto next middleware or the router and return respon
        return try await next(request, context)
    }
}
```

Relationships

Inherits From

`MiddlewareProtocol`, `Swift.Sendable`

Inherited By

`AuthenticatorMiddleware`

Conforming Types

`BasicAuthenticator`
`CORSMiddleware`
`ClosureAuthenticator`
`ContextTransform`
`FileMiddleware`
`IsAuthenticatedMiddleware`
`LogRequestsMiddleware`
`MetricsMiddleware`
`RequestDecompressionMiddleware`
`ResponseCompressionMiddleware`
`Route`
`RouteGroup`
`SessionAuthenticator`
`SessionMiddleware`
`ThrowingContextTransform`
`TracingMiddleware`
`WebSocketUpgradeMiddleware`

See Also

Middleware

`protocol MiddlewareProtocol`

Middleware protocol with generic input, context and output types

`enum MiddlewareFixedTypeBuilder`

Middleware stack result builder

`class MiddlewareGroup`

Group of middleware that can be used to create a responder chain. Each middleware calls the next one

`struct CORSMiddleware`

Middleware implementing Cross-Origin Resource Sharing (CORS) headers.

`struct LogRequestsMiddleware`

Middleware outputting to log for every call to server.

`struct MetricsMiddleware`

Middleware recording metrics for each request

`struct TracingMiddleware`

Middleware creating Distributed Tracing spans for each request.