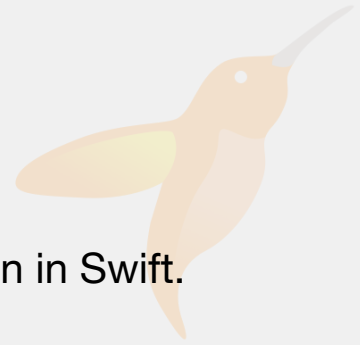


Framework

# Hummingbird

Lightweight, modern, flexible server framework written in Swift.



## Overview

Hummingbird is a lightweight, modern, flexible server framework designed to require the minimum number of dependencies.

It provides a router for directing different endpoints to their handlers, middleware for processing requests before they reach your handlers and processing the responses returned, custom encoding/decoding of requests/responses, TLS and HTTP2.

```
import Hummingbird

// create router and add a single GET /hello route
let router = Router()
router.get("hello") { request, _ -> String in
    return "Hello"
}
// create application using router
let app = Application(
    router: router,
    configuration: .init(address: .hostname("127.0.0.1", port: 8080))
)
// run hummingbird application
try await app.runService()
```

# Topics

## Application

`struct Application`

Application type bringing together all the components of Hummingbird

`protocol ApplicationProtocol`

Application protocol bringing together all the components of Hummingbird

`struct ApplicationConfiguration`

Application configuration

`enum EventLoopGroupProvider`

Where should the application get its EventLoopGroup from

## Router

`class Router`

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

`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

# Request/Response

## `struct Request`

Holds all the values required to process a request

## `typealias Parameters`

Parameters is a special case of FlatDictionary where both the key and value types are Substrings. It is used for parameters extracted from URIs

## `struct MediaType`

Define media type of file

## `struct CacheControl`

Associates cache control values with filename

## `struct Response`

Holds all the required to generate a HTTP Response

`protocol` `ResponseBodyWriter`

HTTP Response Body part writer

`struct` `EditedResponse`

`struct` `Cookie`

Structure holding a single cookie

`struct` `Cookies`

Structure holding an array of cookies

## Request context

`protocol` `RequestContext`

Protocol that all request contexts should conform to. A `RequestContext` is a statically typed metadata container for information that is associated with a [Request](#), and is therefore instantiated alongside the request.

`protocol` `RequestContextSource`

Protocol for source of request contexts

`struct` `ApplicationRequestContextSource`

`RequestContext` source for contexts created by [Application](#).

`struct` `BasicRequestContext`

Implementation of a basic request context that supports everything the Hummingbird library needs

`protocol` `ChildRequestContext`

A `RequestContext` that can be initialized from another `RequestContext`.

`struct` `CoreRequestContextStorage`

Request context values required by Hummingbird itself.

`protocol` `RemoteAddressRequestContext`

Protocol for request context that stores the remote address of connected client.

## Encoding/Decoding

`protocol RequestDecoder`

protocol for decoder deserializing from a Request body

`protocol ResponseEncoder`

protocol for encoders generating a Response

`protocol ResponseEncodable`

Protocol for encodable object that can generate a response. The router will encode the response using the encoder stored in `Application.encoder`.

`protocol ResponseGenerator`

Object that can generate a Response.

`protocol ResponseCodable`

Protocol for codable object that can generate a response

`struct URLEncodedFormDecoder`

The wrapper struct for decoding URL encoded form data to Codable classes

`struct URLEncodedFormEncoder`

The wrapper struct for encoding Codable classes to URL encoded form data

## Errors

`struct HTTPError`

Default HTTP error. Provides an HTTP status and a message

`protocol HTTPResponseError`

An error that is capable of generating an HTTP response

## Middleware

`protocol MiddlewareProtocol`

Middleware protocol with generic input, context and output types

## `enum MiddlewareFixedTypeBuilder`

Middleware stack result builder

## `protocol RouterMiddleware`

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

## `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.

# File management/middleware

## `struct FileMiddleware`

Middleware for serving static files.

## `struct FileIO`

Manages File reading and writing.

## `protocol FileProvider`

Protocol for file provider type used by FileMiddleware

## `protocol FileMiddlewareFileAttributes`

Protocol for all the file attributes required by FileMiddleware

## `struct LocalFileSystem`

Local file system file provider used by FileMiddleware. All file accesses are relative to a root folder

## Storage

`protocol PersistDriver`

Protocol for driver supporting persistent Key/Value pairs across requests

`actor MemoryPersistDriver`

In memory driver for persist system for storing persistent cross request key/value pairs

`struct PersistError`

Errors return by persist framework

## Miscellaneous

`struct Environment`

Access environment variables

`protocol InitializableFromSource`

A type that can be initialized from another type

## Structures

`struct RouterValidationError`

Router validation error

`struct URLEncodedFormError`

Error thrown from parsing URLEncoded forms

---

## See Also

## Related Documentation



## HummingbirdRouter

Alternative result builder based router for Hummingbird.



## HummingbirdTesting

Test framework for Hummingbird.

# Reference Documentation



## HummingbirdCore

Swift NIO based HTTP server.



## HummingbirdAuth

Authentication framework and extensions for Hummingbird.



## HummingbirdCompression

Middleware for decompressing requests and compressing responses



## HummingbirdFluent

Integration with Vapor's Fluent ORM framework.



## HummingbirdLambda

Run Hummingbird inside an AWS Lambda.



## HummingbirdPostgres

Working with Postgres databases.



## HummingbirdRedis

Add Redis support to Hummingbird server with RediStack.



## HummingbirdWebSocket

Adds support for upgrading HTTP connections to WebSocket.



## Jobs

Offload work your server would be doing to another server.

---



## Mustache

Mustache template engine.





## WSCClient

Support for connecting to WebSocket server.