Documentation

Hummingbird Document… / Server protocol

API Collection

Server protocol

Support for TLS and HTTP2 upgrades

Overview

By default a Hummingbird application runs with a HTTP/1.1 server. The Hummingbird comes with additional libraries that allow you to change this to use TLS, HTTP2 and WebSockets

Setting server protocol

When you create your `Application` there is a parameter `server` that is used to define the server protocol and its configuration. Below we are creating a server that support HTTP1 with a idle timeout for requests set to one minutes.

```
let app = Application(
    router: router,
    server: .http1(idleTimeout: .seconds(60))
)
```

HTTPS/TLS

HTTPS is pretty much a requirement for a server these days. Many people run Nginx in front of their server to implement HTTPS, but it is also possible to setup HTTPS inside your Hummingbird application.

```
import HummingbirdTLS

let tlsConfiguration = TLSConfiguration.makeServerConfiguration(
    certificateChain: certificateChain,
    privateKey: privateKey
)
let app = Application(
    router: router,
    server: .tls(.http1(), tlsConfiguration: tlsConfiguration)
)
```

HTTPS is the HTTP protocol with an added encryption layer of TLS to protect the traffic. The `tls` function applies the encryption layer using the crytographic keys supplied in the `TLSConfiguration`.

HTTP2

HTTP2 is becoming increasingly common. It allows you to service multiple HTTP requests concurrently over one connection. The HTTP2 protocol does not require you to use TLS but it is in effect only supported over TLS as there aren't any web browsers that support HTTP2 without TLS. Given this the Hummingbird implementation also requires TLS.

```
import HummingbirdHTTP2

let app = Application(
    router: router,
    server: .http2(
        tlsConfiguration: tlsConfiguration,
        configuration: .init(
            idleTimeout: .seconds(60),
            gracefulCloseTimeout: .seconds(15),
            maxAgeTimeout: .seconds(900),
            streamConfiguration: .init(idleTimeout: .seconds(60))
        )
    )
)
```

The HTTP2 upgrade protocol has a fair amount of configuration. It includes a number of different timeouts,

- `idleTimeout`: How long a connection is kept open while idle
- `gracefulCloseTimeout`: The maximum amount of time to wait for the client to respond before all streams are closed after the second GOAWAY is sent
- `maxAgeTimeout`: a maximum amount of time a connection should be open. Then each HTTP2 stream (request) has its own idle timeout as well.

WebSockets

WebSocket upgrades are also implemented via the server protocol parameter.

```
import HummingbirdWebSocket

let app = Application(
    router: router,
    server: .http1WebSocketUpgrade { request, channel, logger in
        // upgrade if request URI is "/ws"
        guard request.uri == "/ws" else { return .dontUpgrade }
        // The upgrade response includes the headers to include in the respo
        // the WebSocket handler
        return .upgrade([:]) { inbound, outbound, context in
            for try await frame in inbound {
                // send "Received" for every frame we receive
                try await outbound.write(.text("Received"))
            }
        }
    }
)
```

In a similar way you add TLS encryption to the HTTP1 connection you can also add TLS to a connection that accepts WebSocket upgrades.

```
let app = Application(
    router: router,
```

```
    server: .tls(
        .http1WebSocketUpgrade { request, channel, logger in
            // upgrade if request URI is "/ws"
            guard request.uri == "/ws" else { return .dontUpgrade }
            // The upgrade response includes the headers to include in the r
            // the WebSocket handler
            return .upgrade([:]) { inbound, outbound, context in
                try await outbound.write(.text("Hello"))
            }
        },
        tlsConfiguration: tlsConfiguration
    )
)
```

To find out more about WebSocket upgrades and handling WebSocket connections read WebSocket Server Upgrade.

Topics

Reference

- HummingbirdHTTP2

  Add HTTP2 support to Hummingbird server.

- HummingbirdTLS

  Add TLS support to Hummingbird server.

- HummingbirdWebSocket

  Adds support for upgrading HTTP connections to WebSocket.

See Also

Hummingbird Server

- Router

  The router directs requests to their handlers based on the contents of their path.

- Request Decoding

  Decoding of Requests with JSON content and other formats.

- Response Encoding

  Writing Responses using JSON and other formats.

- Request Contexts

  Controlling contextual data provided to middleware and route handlers

- Middleware

  Processing requests and responses outside of request handlers.

- Error Handling

  How to build errors for the server to return.

- Logging, Metrics and Tracing

  Considered the three pillars of observability, logging, metrics and tracing provide different ways of viewing how your application is working.

- Result Builder Router

  Building your router using a result builder.

- Service Lifecycle

  Integration with Swift Service Lifecycle

- Testing

  Using the HummingbirdTesting framework to test your application

- Persistent data

  How to persist data between requests to your server.

- Migrating to Hummingbird v2

  Migration guide for converting Hummingbird v1 applications to Hummingbird v2