Documentation

Hummingbird Document… / Error Handling

Article

Error Handling

How to build errors for the server to return.

Overview

If a middleware or route handler throws an error the server needs to know how to handle this. If the server does not know how to handle the error then the only thing it can return to the client is a status code of 500 (Internal Server Error). This is not overly informative.

HTTPError

Hummingbird uses the Error object `HTTPError` throughout its codebase. The server recognises this and can generate a more informative response for the client from it. The error includes the status code that should be returned and a response message if needed. For example

```
router.get("user") { request, context -> User in
    guard let userId = request.uri.queryParameters.get("id", as: Int.self) e
        throw HTTPError(.badRequest, message: "Invalid user id")
    }
    ...
}
```

The `HTTPError` generated here will be recognised by the server and it will generate a status code 400 (Bad Request) with the body “Invalid user id”.

HTTPResponseError

The server knows how to respond to a `HTTPError` because it conforms to protocol `HTTPResponseError`. You can create your own `Error` object and conform it to `HTTPResponseError` and the server will know how to generate a sensible error from it. The example below is a error class that outputs an error code in the response headers.

```
struct MyError: HTTPResponseError {
    init(_ status: HTTPResponseStatus, errorCode: String) {
        self.status = status
        self.errorCode = errorCode
    }

    let errorCode: String

    // required by HTTPResponseError protocol
    let status: HTTPResponseStatus

    // required by HTTPResponseError protocol
    func response(from request: Request, context: some RequestContext) throw
        .init(
            status: self.status,
            headers: ["error-code": self.errorCode]
        )
    }
}
```

See Also

Related Documentation

`struct HTTPError`

Default HTTP error. Provides an HTTP status and a message

`protocol HTTPResponseError`

An error that is capable of generating an HTTP response

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.

- 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.

- Server protocol

  Support for TLS and HTTP2 upgrades

- 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