Documentation Language: Swift

Hummingbird / ApplicationProtocol

Protocol

ApplicationProtocol

Application protocol bringing together all the components of Hummingbird

```
protocol ApplicationProtocol : Service
```

Overview

`ApplicationProtocol` is a protocol used to define your application. It provides the glue between your router and HTTP server.

Implementing a `ApplicationProtocol` requires two member variables: `responder` and `server`.

```
struct MyApp: ApplicationProtocol {
    /// The responder will return an `Response` given an `Request` and a con
    var responder: some Responder<BasicRequestContext> {
        let router = Router(context: Context.self)
        router.get("hello") { _,_ in "Hello" }
        return router.buildResponder()
    }
    /// Defines your server type. This is the default value so in
    /// effect is unnecessary
    var server: HTTPChannelBuilder<some ChildChannel> { .http1() }
}
let app = MyApp()
try await app.runService()
```

If you don't want to create your own type, Hummingbird provides `Application` a concrete implementation of `ApplicationProtocol`.

Topics

Associated Types

`associatedtype Responder : HTTPResponder`

Responder that generates a response from a requests and context

Required

Instance Properties

`var configuration: ApplicationConfiguration`

Application configuration

Required Default implementation provided.

`var eventLoopGroup: EventLoopGroup`

event loop group used by application

Required Default implementation provided.

`var logger: Logger`

Logger

Required Default implementation provided.

`var processesRunBeforeServerStart: [() async throws -> Void]`

Array of processes run before we kick off the server. These tend to be processes that need other services running but need to be run before the server is setup

Required Default implementation provided.

`var responder: Responder`

Build the responder

Required

`var server: HTTPServerBuilder`

Server channel builder

Required Default implementation provided.

```
var services: [any Service]
```

services attached to the application.

Required Default implementation provided.

Instance Methods

```
func onServerRunning(Channel) async
```

This is called once the server is running and we have an active Channel

Required Default implementation provided.

```
func runService(gracefulShutdownSignals: [UnixSignal]) async throws
```

Helper function that runs application inside a ServiceGroup which will gracefully shutdown on signals SIGINT, SIGTERM

```
func test<Value>(TestingSetup, (any TestClientProtocol) async throws
-> Value) async throws -> Value
```

Test `Application`

Type Aliases

```
typealias Context
```

Context passed with Request to responder

Relationships

Inherits From

`ServiceLifecycle.Service`, `Swift.Sendable`

Conforming Types

`Application`

See Also

Application

`struct Application`

Application type bringing together all the components of Hummingbird

`struct ApplicationConfiguration`

Application configuration

`enum EventLoopGroupProvider`

Where should the application get its EventLoopGroup from