

An online API documentation with examples so you can start building web apps with Fiber right away!

Fiber is an Express inspired **web framework** built on top of Fasthttp, the **fastest** HTTP engine for Go. Designed to **ease** things up for **fast** development with **zero memory allocation** and **performance** in mind.

These docs are for Fiber v2, which was released on September 15th, 2020.

Installation

First of all, download and install Go. 1.14 or higher is required.

Installation is done using the go get command:

```
go get github.com/gofiber/fiber/v2
```

Zero Allocation

Some values returned from *fiber.Ctx are not immutable by default.

Because fiber is optimized for **high-performance**, values returned from **fiber.Ctx** are **not** immutable by default and **will** be re-used across requests. As a rule of thumb, you **must** only use context values within the handler, and you **must not** keep any references. As soon as you return from the handler, any values you have obtained from the context will be re-used in future requests and will change below your feet. Here is an example:

```
func handler(c *fiber.Ctx) error {
    // Variable is only valid within this handler
    result := c.Params("foo")

    // ...
}
```

If you need to persist such values outside the handler, make copies of their **underlying buffer** using the copy builtin. Here is an example for persisting a string:

```
func handler(c *fiber.Ctx) error {
    // Variable is only valid within this handler
    result := c.Params("foo")

    // Make a copy
    buffer := make([]byte, len(result))
    copy(buffer, result)
    resultCopy := string(buffer)
    // Variable is now valid forever

// ...
}
```

We created a custom CopyString function that does the above and is available under gofiber/utils.

Alternatively, you can also use the Immutable setting. It will make all values returned from the context immutable, allowing you to persist them anywhere. Of course, this comes at the cost of performance.

For more information, please check #426 and #185.

Hello, World!

Embedded below is essentially the most straightforward **Fiber** app you can create:

```
package main

import "github.com/gofiber/fiber/v2"

func main() {
    app := fiber.New()
```

```
app.Get("/", func(c *fiber.Ctx) error {
          return c.SendString("Hello, World!")
})
app.Listen(":3000")
}
```

```
go run server.go
```

Browse to http://localhost:3000 and you should see Hello, World! on the page.

Basic routing

Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, PUT, POST, etc.).

Each route can have **multiple handler functions** that are executed when the route is matched.

Route definition takes the following structures:

```
// Function signature
app.Method(path string, ...func(*fiber.Ctx) error)
```

- app is an instance of Fiber
- Method is an HTTP request method: GET, PUT, POST, etc.
- path is a virtual path on the server
- func(*fiber.Ctx) error is a callback function containing the Context executed when the route is matched

Simple route

```
// Respond with "Hello, World!" on root path, "/"
app.Get("/", func(c *fiber.Ctx) error {
    return c.SendString("Hello, World!")
})
```

Parameters

```
// GET http://localhost:8080/hello%20world
```

Optional parameter

```
// GET http://localhost:3000/john

app.Get("/:name?", func(c *fiber.Ctx) error {
    if c.Params("name") != "" {
        return c.SendString("Hello " + c.Params("name"))
        // => Hello john
    }
    return c.SendString("Where is john?")
})
```

Wildcards

```
// GET http://localhost:3000/api/user/john

app.Get("/api/*", func(c *fiber.Ctx) error {
    return c.SendString("API path: " + c.Params("*"))
    // => API path: user/john
})
```

Static files

To serve static files such as **images**, **CSS**, and **JavaScript** files, replace your function handler with a file or directory string.

Function signature:

```
app.Static(prefix, root string, config ...Static)
```

Use the following code to serve files in a directory named ./public:

```
app := fiber.New()
```

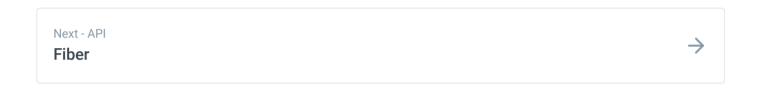
```
app.Static("/", "./public")
app.Listen(":3000")
```

Now, you can load the files that are in the ./public directory:

```
http://localhost:8080/hello.html
http://localhost:8080/js/jquery.js
http://localhost:8080/css/style.css
```

Note

For more information on how to build APIs in Go with Fiber, please check out this excellent article on building an express-style API in Go with Fiber.



Last modified 26d ago