

**Vic Shóstak**

Posted on Sep 13, 2021 • Updated on Nov 8, 2021

📖 Go Fiber by Examples: Working with middlewares and boilerplates

#go #beginners #tutorial #showdev

Go Fiber by Examples (4 Part Series)

- 1 📖 Go Fiber by Examples: How can the Fiber Web Framework be ...
- 2 📖 Go Fiber by Examples: Delving into built-in functions
- 3 📖 Go Fiber by Examples: Testing the application
- 4 📖 **Go Fiber by Examples: Working with middlewares and boilerpl...**

Introduction

Hey, DEV friends! 🙌

So, we've already got a good understanding of the key features and the inner workings of the Fiber web framework. Now, it's the turn of additional tools and packages that can greatly improve our productivity as Go programmers.

Plan for the Chapter 4

In this fourth article (or chapter), we will review the topics of the [Fiber](#) security & logging middlewares and useful boilerplates.

Yes, these are the main topics 🙌



Table of contents

- [Working with Security middlewares](#)
 - [Helmet middleware](#)
 - [CSRF middleware](#)
 - [Limiter middleware](#)
- [Explore Logging middleware](#)
- [Useful Fiber Boilerplates](#)
- [Summary](#)

Working with Security middlewares

Security middlewares in the Fiber web framework perform the task of protecting your application from various types of hacker attacks. This is **critical** for projects that work in production with real users.



Note: However, even if you don't plan to put your project into production now, knowing about such middleware is still a useful skill.

[↑ Table of contents](#)

Helmet middleware

Helmet middleware helps to secure our Fiber application by setting various HTTP headers:

- [XSS Protection](#)
- [Content-Type No Sniff](#)
- [X-Frame Options](#)
- [HSTS](#) Max Age
- [CSP Report Only](#)
- Exclude Subdomains & Preload Enabled
- Content Security & Referrer Policies

```
// ./go/security_middlewares.go
```

```
import "github.com/gofiber/helmet/v2"

// ...

// Use middlewares for each route
app.Use(
    helmet.New(), // add Helmet middleware
)
```

[↑ Table of contents](#)

CSRF middleware

CSRF middleware for Fiber that provides [Cross-Site request forgery](#) protection by passing a CSRF token via cookies.

This cookie value will be used to compare against the client CSRF token in the POST requests. When the CSRF token is invalid, this middleware will delete the `csrf_` cookie and return the `fiber.ErrForbidden` error.

```
// ./go/security_middlewares.go

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

// ...

// Use middlewares for each route
app.Use(
    csrf.New(), // add CSRF middleware
)
```

We can retrieve the CSRF token with `c.Locals(key)`, where `key` is the option name in the custom middleware configuration.

The CSRF middleware custom config may look like this:

```
// Set config for CSRF middleware
csrfConfig := csrf.Config{
    KeyLookup:      "header:X-Csrf-Token", // string in the form of '<source>:<key>'
    CookieName:     "my_csrf_",            // name of the session cookie
    CookieSameSite: "Strict",              // indicates if CSRF cookie is request
    Expiration:     3 * time.Hour,         // expiration is the duration before (
    KeyGenerator:   utils.UUID,            // creates a new CSRF token
}

// Use middlewares for each route
```

```
app.Use(  
  csrf.New(csrfConfig), // add CSRF middleware with config  
)
```

[↑Table of contents](#)

Limiter middleware

Limiter middleware for Fiber used to limit repeated requests to public APIs or endpoints such as password reset etc. Moreover, useful for API clients, web crawling, or other tasks that need to be throttled.

```
// ./go/security_middlewares.go  
  
import "github.com/gofiber/fiber/v2/middleware/limiter"  
  
// ...  
  
// Use middlewares for each route  
app.Use(  
  limiter.New(), // add Limiter middleware  
)
```

Most of the time, you will probably be using this middleware along with your configuration. It's easy to add a config like this:

```
// Set config for Limiter middleware  
limiterConfig := limiter.Config{  
  Next: func(c *fiber.Ctx) bool {  
    return c.IP() == "127.0.0.1" // limit will apply to this IP  
  },  
  Max:      20, // max count of connections  
  Expiration: 30 * time.Second, // expiration time of the limit  
  Storage:    myCustomStorage{}, // used to store the state of the middleware  
  KeyGenerator: func(c *fiber.Ctx) string {  
    return c.Get("x-forwarded-for") // allows you to generate custom keys  
  },  
  LimitReached: func(c *fiber.Ctx) error {  
    return c.SendFile("./too-fast-page.html") // called when a request hits the  
  },  
}  
  
// Use middlewares for each route  
app.Use(  
  limiter.New(limiterConfig), // add Limiter middleware with config
```

>

[↑ Table of contents](#)

Explore Logging middleware

Like any other framework, Fiber also has its built-in middleware for logging HTTP request/response details and displaying results in the console.

Let's look at an example of what this might look like:

```
// ./go/logger_middlewares.go

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

// ...

// Use middlewares for each route
app.Use(
    logger.New(), // add Logger middleware
)
```

And the console output looks like this:

```
08:17:42 | 404 | 85ms | 127.0.0.1 | GET | /v1/user/123
08:18:07 | 204 | 145ms | 127.0.0.1 | POST | /v1/webhook/postmark
08:19:53 | 201 | 138ms | 127.0.0.1 | PUT | /v1/article/create
```

Yes, Logger middleware connects in the same way as the middleware reviewed earlier. Furthermore, we can save all logs to a file, not console output, like this:

```
// Define file to logs
file, err := os.OpenFile("./my_logs.log", os.O_RDWR|os.O_CREATE|os.O_APPEND, 0666)
if err != nil {
    log.Fatalf("error opening file: %v", err)
}
defer file.Close()

// Set config for logger
loggerConfig := logger.Config{
    Output: file, // add file to save output
}

// Use middlewares for each route
app.Use(
```

```
logger.New(loggerConfig), // add Logger middleware with config
)
```

[↑ Table of contents](#)

Useful Fiber Boilerplates

Fiber has already gathered a friendly community of programmers from all over the world. Every day, they share new and interesting packages and templates, which make starting a new project easier for us.

Boilerplate projects not only allow you to create a complete application structure with all the settings, but also a better understanding of the principle of code organization in the ecosystem of the web framework on a real example.


Here we will only look at two of the most popular examples from the large number of such projects used by Fiber community and authors. But we can always find and use others, or even create our own and offer them to the community!

[↑ Table of contents](#)

The official boilerplate application template

This template was specially created by the authors of Fiber for a quick enter to the framework, without additional third-party packages. The application is specially designed to run in the Docker container.

 [gofiber](#) / [boilerplate](#)

 Boilerplate for  Fiber

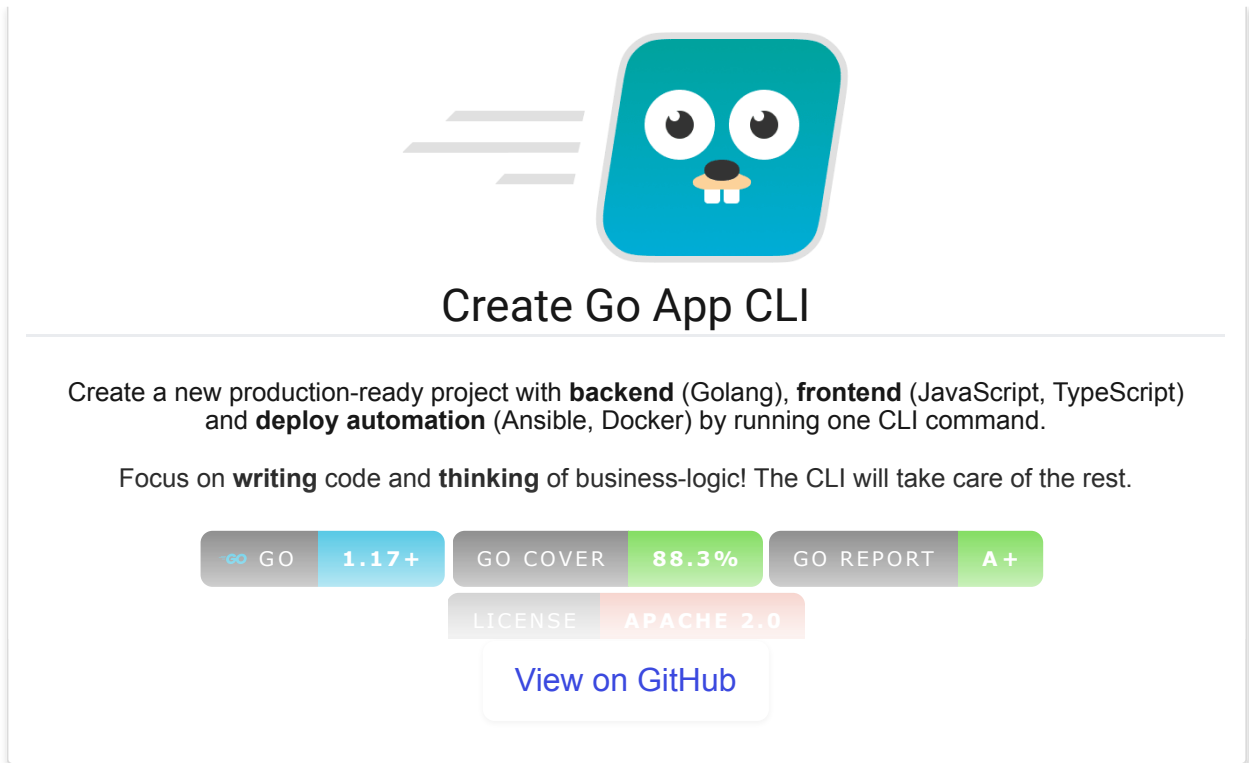
[↑ Table of contents](#)

The Create Go App project

When talking about boilerplate packages, I can't help but mention a project that has already helped many developers (myself included) to create new Go projects in a matter of minutes.

 [create-go-app](#) / [cli](#)

🌟 Create a new production-ready project with backend, frontend and deploy automation by running one CLI command!



The project is a handy interactive CLI with which you can easily create a full-fledged web application in just a couple of clicks:

- Out of the box, the project has its own fully configured Fiber REST API application template with automatic [Swagger](#) documentation and authorization of requests via [JWT token](#).
- The background part will be generated with [Vite.js](#), and you are free to choose absolutely any startup template for React, Preact, Vue, Svelte, web components, vanilla JavaScript or TypeScript and so on.
- Specifically configured roles and playbooks for the [Ansible](#) to deploy the application in isolated [Docker containers](#) on a remote server.

[↑ Table of contents](#)

Summary

Wow, here's a summary of the chapter you passed! We learned how easy it is to make our Fiber application secure by adding some built-in middlewares.

Then there was a detailed breakdown of how the logging system works, which will help us more than once in future articles in this series.

Next time, we'll learn even more about utility middlewares, external Fiber middlewares and the third-party packages for this wonderful web framework.

Stay tuned, don't switch! 😊

[↑ Table of contents](#)

Photos and videos by

- Kolya Korzh <https://unsplash.com/photos/UOq-FqdlTpw>





P.S.

If you want more articles like this on this blog, then post a comment below and subscribe to me. Thanks! 🙌

And, of course, you can support me by donating at [LiberaPay](#). *Each donation will be used to write new articles and develop non-profit open-source projects for the community.*



Go Fiber by Examples (4 Part Series)

- 1  Go Fiber by Examples: How can the Fiber Web Framework be ...
- 2  Go Fiber by Examples: Delving into built-in functions
- 3  Go Fiber by Examples: Testing the application
- 4  **Go Fiber by Examples: Working with middlewares and boilerpl...**

Top comments (6)



nigel447 • Jun 17 '22



thanks for this, writing documentation is a pain at best, writing good documentation is an art form, this is high quality work and very much appreciated. much respect



Xiaowei Liu • Jan 12 '22



Great article, thank you for your time.



Ayдын Yakar • Oct 23 '21



Thanks for effort your time for articles 🙏👍



Vic Shóstak  • Oct 24 '21





No problem! Have a nice read ;)



Comment deleted



Vic Shóstak  • Jan 12 '22



I'm the person who wrote the Fiber documentation in the first place. So, I have every right to do so, check out the copyright on the Fiber website or the GitHub page, just for fun.

Hello, my name is Vic Shostak, I'm the one who created this framework. Whats up, Aaron Kofi Gayi? May I now have your permission to reprint my own lines? :)

Especially since this is a chapter from my unreleased book (which I wrote about in the first article of this series), which I kindly decided to put out for FREE for everyone on this blogging service.

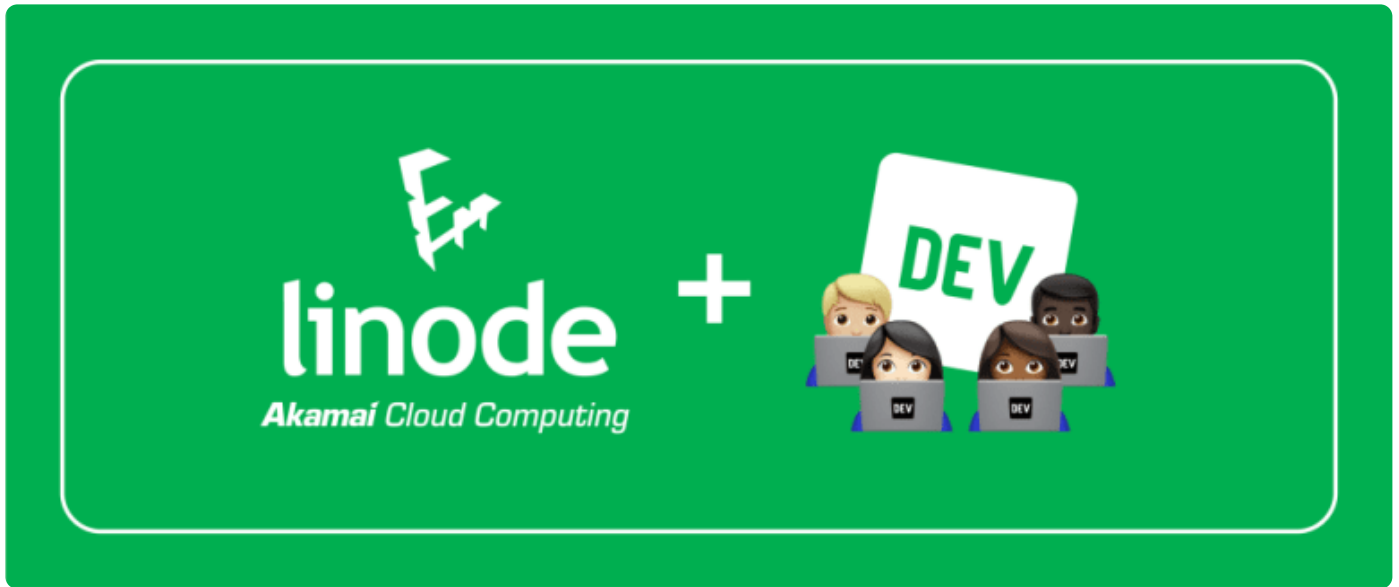
Question for you, Aaron: Why don't you write your own article that doesn't "reprint documentation"?

I just have never understood people who only criticize other people's work and offer nothing in return. This is called "toxicity" and only leads to degradation of our friendly community on Dev.to.

Please think about it. See you soon!

[Code of Conduct](#) • [Report abuse](#)

Build Anything...



Use any Linode offering to create something for the **DEV x Linode Hackathon 2022**. A variety of prizes are up for grabs, including \$1,000 USD. 👁️

→ [Join the Hackathon](#) <-



Vic Shóstak

Hey! 🙌 I'm a Software Engineer with over 13 years of practical experience (Go, Python, TypeScript), UX Consultant, UI Designer, Open Source Supporter

LOCATION



WORK

Software Engineer (Go, Python, Kotlin, TypeScript), UX Consultant

JOINED

Jan 5, 2020

More from [Vic Shóstak](#)

Kotlin short recipes: Generating a chessboard for a console game

[#kotlin](#) [#gamedev](#) [#showdev](#) [#tutorial](#)



Structure of a single-page Vue 3 (TypeScript) app using JWT authorization requests to the backend

[#vue](#) [#typescript](#) [#tutorial](#) [#webdev](#)



You don't need SVG! Creating animated loaders for content with Tailwind CSS

#tutorial #css #html #tailwindcss
