Writing API Services in Go

Nandakumar Edamana

July 14, 2023

Question

What do you mean by API Server Software? Think about a single program that serves an API. How does it differ from a program that accepts two numbers via the command line, prints the sum and quits?

API Server Characteristics

API Server Characteristics

- Daemon

API Server Characteristics

- Daemon
- Serves API over HTTP over TCP

API Server Characteristics

- Daemon
- Serves API over HTTP over TCP
- Sticks to something like REST (quality of the API rather than the program itself)

Levels

- HTTP server in Go using no third-party libraries
- HTTP server in Go using a third-party router (chi)
- Auto-generation of boilerplate using OpenAPI
- Writing multiple services and combining them using an API Gateway

NOTE: OpenAPI is for more than mere code generation.

Packages

- net/http - Built-in library for HTTP client, server and mux; supports middlewares, does not support RegEx path matching.
- chi, gorilla/mux, etc. - Feature-rich HTTP routers

Hello World

```
mkdir 1-hello-world && cd 1-hello-world
go mod init helloapi
# "to add module requirements and sums"
go mod tidy
```

Hello World

```
package main

import (
  "fmt"
  "log"
  "net/http"
)
```

Hello World

```
func main() {
  // Adds to DefaultServeMux
  // Note: you can use anonymous functions
  http.HandleFunc("/", handleIndex)
  http.HandleFunc("/hello", handleHello)

  fmt.Println("Listening␣at␣:8080...")

  // Address and handler;
  // handler = nil means DefaultServeMux
  err := http.ListenAndServe(":8080", nil)
  if err != nil {
    log.Fatal(err)
  }
}
```

Hello World

```
func handleHello(w http.ResponseWriter,
                 r *http.Request) {

  w.Write([]byte("world!\n"))
}

func handleIndex(w http.ResponseWriter,
                 r *http.Request) {

  w.Write([]byte("Welcome!\n"))
}
```

Hello World

```
$ go run .
Listening at :8080...
```

Hello World

```
$ curl localhost:8080/
Welcome!

$ curl localhost:8080/hello
world!
```

Hello World

```
$ go run .
Listening at :8080...
^Csignal: interrupt
```

The Complexity Underneath

See the manpage of bind(2) and add forked or threaded serving.

Questions?

Thank You