Go-Lang HttpRouter

Eko Kurniawan Khannedy

Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 10+ years experiences
- www.programmerzamannow.com
- youtube.com/c/ProgrammerZamanNow

Eko Kurniawan Khannedy

- Telegram : @khannedy
- Facebook : fb.com/ProgrammerZamanNow
- Instagram : instagram.com/programmerzamannow
- Youtube : youtube.com/c/ProgrammerZamanNow
- Telegram Channel : t.me/ProgrammerZamanNow
- Email : echo.khannedy@gmail.com

Sebelum Belajar

- Go-Lang Dasar
- Go-Lang Modules
- Go-Lang Unit Test
- Go-Lang Goroutines
- Go-Lang Embed
- Go-Lang Web

Agenda

- Pengenalan HttpRouter
- Router
- Params
- Serve Files
- Error Handler
- Dan lain-lain

Pengenalan HttpRouter

Pengenalan HttpRouter

- HttpRouter merupakan salah satu OpenSource Library yang populer untuk Http Handler di Go-Lang
- HttpRouter terkenal dengan kecepatannya dan juga sangat minimalis
- Hal ini dikarenakan HttpRouter hanya memiliki fitur untuk routing saja, tidak memiliki fitur apapun selain itu
- https://github.com/julienschmidt/httprouter

Menambah HttpRouter ke Project

go get github.com/julienschmidt/httprouter

go get github.com/stretchr/testify

Kode : Go Module

go.mod

```
module github.com/ProgrammerZamanNow/belajar-golang-httprouter

go 1.16

require (
    github.com/julienschmidt/httprouter v1.3.0 // indirect
    github.com/stretchr/testify v1.7.0 // indirect
)

```

Router

Router

- Inti dari library HttpRouter adalah struct Router
- Router ini merupakan implementasi dari http.Handler, sehingga kita bisa dengan mudah menambahkan ke dalam http.Server
- Untuk membuat Router, kita bisa menggunakan function httprouter.New(), yang akan mengembalikan Router pointer

Kode : Router

```
router := httprouter.New()

server := http.Server{
    Handler: router,
    Addr:    "localhost:3000",
}

server.ListenAndServe()
```

HTTP Method

- Router mirip dengan ServeMux, dimana kita bisa menambahkan route ke dalam Router
- Kelebihan dibandingkan dengan ServeMux adalah, pada Router, kita bisa menentukan HTTP Method yang ingin kita gunakan, misal GET, POST, PUT, dan lain-lain
- Cara menambahkan route ke dalam Router adalah gunakan function yang sama dengan HTTP Method nya, misal router.GET(), router.POST(), dan lain-lain

httprouter.Handle

- Saat kita menggunakan ServeMux, ketika menambah route, kita bisa menambahkan http.Handler
- Berbeda dengan Router, pada Router kita tidak menggunakan http.Handler lagi, melainkan menggunakan type httprouter.Handle
- Perbedaan dengan http.Handler adalah, pada httprouter.Handle, terdapat parameter ke tiga yaitu Params, yang akan kita bahas nanti di chapter tersendiri

```
// Handle is a function that can be registered to a route to handle HTTP
// requests. Like http.HandlerFunc, but has a third parameter for the values of
// wildcards (variables).
type Handle func(http.ResponseWriter, *http.Request, Params)
```

Kode : Route

```
router := httprouter.New()
router.GET("/", func(writer http.ResponseWriter, request *http.Request, _ httprouter.Params) {
    fmt.Fprint(writer, "Hello Get")
})

request := httptest.NewRequest("GET", "http://localhost:3000/", nil)
recorder := httptest.NewRecorder()

router.ServeHTTP(recorder, request)
response := recorder.Result()

bytes, _ := io.ReadAll(response.Body)
assert.Equal(t, "Hello Get", string(bytes))
```

Params

Params

- httprouter.Handle memiliki parameter yang ketiga, yaitu Params. Untuk apa kegunaan Params?
- Params merupakan tempat untuk menyimpan parameter yang dikirim dari client
- Namun Params ini bukan query parameter, melainkan parameter di URL
- Kadang kita butuh membuat URL yang tidak fix, alias bisa berubah-ubah, misal /products/1, /products/2, dan seterusnya
- ServeMux tidak mendukung hal tersebut, namun Router mendukung hal tersebut
- Parameter yang dinamis yang terdapat di URL, secara otomatis dikumpulkan di Params
- Namun, agar Router tahu, kita harus memberi tahu ketika menambahkan Route, dibagian mana kita akan buat URL path nya menjadi dinamis

Kode : Params

```
router := httprouter.New()
router.GET("/products/:id", func(writer http.ResponseWriter, request *http.Request, params httprouter.Params) {
    text := "Product " + params.ByName("id")
    fmt.Fprint(writer, text)
})

request := httptest.NewRequest("GET", "http://localhost:3000/products/1", nil)
recorder := httptest.NewRecorder()

router.ServeHTTP(recorder, request)
response := recorder.Result()

bytes, _ := io.ReadAll(response.Body)
assert.Equal(t, "Product 1", string(bytes))
```

Router Pattern

Router Pattern

- Sekarang kita sudah tahu bahwa dengan menggunakan Router, kita bisa menambah params di URL
- Sekarang pertanyaannya, bagaimana pattern (pola) pembuatan parameter nya?

Named Parameter

- Named parameter adalah pola pembuatan parameter dengan menggunakan nama
- Setiap nama parameter harus diawali dengan : (titik dua), lalu diikuti dengan nama parameter
- Contoh, jika kita memiliki pattern seperti ini :

Pattern	/user/:user
/user/eko	match
/user/you	match
/user/eko/profile	no match
/user/	no match

Catch All Parameter

- Selain named parameter, ada juga yang bernama catch all parameter, yaitu menangkap semua parameter
- Catch all parameter harus diawali dengan * (bintang), lalu diikuti dengan nama parameter
- Catch all parameter harus berada di posisi akhir URL

Pattern	/src/*filepath
/src/	no match
/src/somefile	match
/src/subdir/somefile	match

Kode : Named Parameter

```
router.GET("/products/:id/items/:itemId",
    func(writer http.ResponseWriter, request *http.Request, params httprouter.Params) {
        text := "Product " + params.ByName("id") + " Item " + params.ByName("itemId")
        fmt.Fprint(writer, text)
    })

request := httptest.NewRequest("GET", "http://localhost:3000/products/1/items/2", nil)
recorder := httptest.NewRecorder()

router.ServeHTTP(recorder, request)
response := recorder.Result()

bytes, _ := io.ReadAll(response.Body)
assert.Equal(t, "Product 1 Item 2", string(bytes))
```

Kode : Catch All Parameter

```
router.GET("/images/*image", func(writer http.ResponseWriter, request *http.Request, params httprouter.Params) {
    text := "Image : " + params.ByName("image")
    fmt.Fprint(writer, text)
})

request := httptest.NewRequest("GET", "http://localhost:3000/images/small/profile.png", nil)
recorder := httptest.NewRecorder()

router.ServeHTTP(recorder, request)
response := recorder.Result()

bytes, _ := io.ReadAll(response.Body)
assert.Equal(t, "Image : /small/profile.png", string(bytes))
```

Serve File

Serve File

- Pada materi Go-Lang Web, kita sudah pernah membahas tentang Serve File
- Pada Router pun, mendukung serve static file menggunakan function ServeFiles(Path, FileSystem)
- Dimana pada Path, kita harus menggunakan Catch All Parameter
- Sedangkan pada FileSystem kita bisa melakukan manual load dari folder atau menggunakan golang embed, seperti yang pernah kita bahas di materi Go-Lang Web

Kode : Serve File

```
//go:embed resources
var resources embed.FS

func TestServeFile(t *testing.T) {
    router := httprouter.New()
    directory, _ := fs.Sub(resources, "resources")
    router.ServeFiles("/files/*filepath", http.FS(directory))

    request := httptest.NewRequest("GET", "http://localhost:3000/files/hello.txt", nil)
    recorder := httptest.NewRecorder()
```

Panic Handler

Panic Handler

- Apa yang terjadi jika terjadi panic pada logic Handler yang kita buat?
- Secara otomatis akan terjadi error, dan web akan berhenti mengembalikan response
- Kadang saat terjadi panic, kita ingin melakukan sesuatu, misal memberitahu jika terjadi kesalahan di web, atau bahkan mengirim informasi log kesalahan yang terjadi
- Sebelumnya, seperti yang sudah kita bahas di materi Go-Lang Web, jika kita ingin menangani panic, kita harus membuat Middleware khusus secara manual
- Namun di Router, sudah disediakan untuk menangani panic, caranya dengan menggunakan attribute PanicHandler : func(http.ResponseWriter, *http.Request, interface{})

Kode : Panic Handler

```
router := httprouter.New()
router.PanicHandler = func(writer http.ResponseWriter, request *http.Request, i interface{}) {
    fmt.Fprint(writer, "Panic ", i)
}
router.GET("/", func(writer http.ResponseWriter, request *http.Request, params httprouter.Params) {
    panic("Ups")
})

request := httptest.NewRequest("GET", "http://localhost:3000/", nil)
recorder := httptest.NewRecorder()
```

Not Found Handler

Not Found Handler

- Selain panic handler, Router juga memiliki not found handler
- Not found handler adalah handler yang dieksekusi ketika client mencoba melakukan request URL yang memang tidak terdapat di Router
- Secara default, jika tidak ada route tidak ditemukan, Router akan melanjutkan request ke http.NotFound, namun kita bisa mengubah nya
- Caranya dengan mengubah router.NotFound = http.Handler

Kode : Not Found Handler

```
router.NotFound = http.HandlerFunc(func(writer http.ResponseWriter, request *http.Request) {
    fmt.Fprint(writer, "Gak Ketemu")
})

request := httptest.NewRequest("GET", "http://localhost:3000/", nil)
recorder := httptest.NewRecorder()

router.ServeHTTP(recorder, request)
response := recorder.Result()

bytes, _ := io.ReadAll(response.Body)
assert.Equal(t, "Gak Ketemu", string(bytes))
```

Method Not Allowed Handler

Method Not Allowed Handler

- Saat menggunakan ServeMux, kita tidak bisa menentukan HTTP Method apa yang digunakan untuk Handler
- Namun pada Router, kita bisa menentukan HTTP Method yang ingin kita gunakan, lantas apa yang terjadi jika client tidak mengirim HTTP Method sesuai dengan yang kita tentukan?
- Maka akan terjadi error Method Not Allowed
- Secara default, jika terjadi error seperti ini, maka Router akan memanggil function http.Error
- Jika kita ingin mengubahnya, kita bisa gunakan router.MethodNotAllowed = http.Handler

Kode : Method Not Allowed Handler

```
router := httprouter.New()
router.MethodNotAllowed = http.HandlerFunc(func(writer http.ResponseWriter, request *http.Request) {
    fmt.Fprint(writer, "Gak Boleh")
})
router.POST("/", func(writer http.ResponseWriter, request *http.Request, _ httprouter.Params) {
    fmt.Fprint(writer, "POST")
})

request := httptest.NewRequest("GET", "http://localhost:3000/", nil)
recorder := httptest.NewRecorder()
```

Middleware

Middleware

- HttpRouter hanyalah library untuk http router saja, tidak ada fitur lain selain router
- Dan karena Router merupakan implementasi dari http.Handler, jadi untuk middleware, kita bisa membuat sendiri, seperti yang sudah kita bahas pada course Go-Lang Web

Kode : Log Middleware

```go
type LogMiddleware struct {
    http.Handler
}

func (middleware *LogMiddleware) ServeHTTP(writer http.ResponseWriter, request *http.Request) {
    fmt.Println("Receive request")
    middleware.Handler.ServeHTTP(writer, request)
}
```

Kode : Middleware

```
router.GET("/", func(writer http.ResponseWriter, request *http.Request, _ httprouter.Params) {
    fmt.Fprint(writer, "Middleware")
})
middleware := LogMiddleware{Handler: router}

request := httptest.NewRequest("GET", "http://localhost:3000/", nil)
recorder := httptest.NewRecorder()

middleware.ServeHTTP(recorder, request)
response := recorder.Result()
```

Materi Selanjutnya

Materi Selanjutnya

- Go-Lang RESTful API
- Go-Lang Deployment
- Go-Lang Docker