Scalable Web Service With Golang
sesi 10

Middleware

Middleware

Middleware merupakan sebuah fungsi yang akan terkesekusi sesudah maupun sebelum mencapai sebuah endpoint. Biasa middleware digunakan untuk logging atau untuk mengamankan sebuah endpoint seperti contohnya proses autentikasi dan autorisasi.

Untuk membuat middleware pada bahasa Go, kita akan menggunakan package *net/http* dengan menggunakan multiplexer nya agar kita dapat melakukan kustomisasi.

Gambar dibawah ini merupakan gambaran dari alur sebuah middleware.

```
Router => Middleware Handler => Application Handler
```

Middleware

```go
func middleware1(next http.Handler) http.Handler {

  return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
    next.ServeHTTP(w, r)
  })

}
```

Untuk membuat sebuah middleware, maka kita harus suatu function yang memuaskan interface *http.Handler* yang dimana interface ini memiliki satu buah method dengan berupa ServeHTTP(http.ResponseWriter, *http.Request).

Perhatikan pada gambar diatas. Function *middleware1* merupakan sebuah middleware.

Setiap middleware pada bahasa Go akan menerima satu parameter dengan tipe data *http.Handler* dan harus mereturn tipe data *http.Handler*.

Jika kita perhatikan pada gambar diatas, function *middleware1* mereturn sebuah anonymous function dengan nama *http.HandlerFunc* yang dimana function ini menerima satu argument berupa tipe data function yang mempunyai argument yang sama seperti method *ServeHTTP*.

Middleware

```go
func middleware1(next http.Handler) http.Handler {

  return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
    next.ServeHTTP(w, r)
  })

}
```

Perlu diingat bahwa function *http.Handlerfunc* secara default mengimplementasikan method *ServeHTTP* dan setiap function yang mempunyai skema parameter yang sama dengan method *ServeHTTP* dapat di ubah tipe datanya menjadi interface *http.Handler* dengan cara menjadikannya parameter dari function *http.HandlerFunc*.

Middleware

Perhatikan pada gambar disebelah kanan.

Pada line 11, kita mendeklarasikan sebuah variable bernama *mux* yang akan menjadi *multiplexer* nya.

Kemudian pada line 13, kita membuat variable bernama *endpoint* yang menampung satu-satunya endpoint pada aplikasi kita saat ini yaitu function *greet*. Perhatikan bahwa kita menjadikan function *greet* sebagai argument dari function *http.HandlerFunc* agar function *greet* dapat menjadi sebuah function dengan tipe data *http.Handler*.

Kemudian pada line 15, kita membuat routingnya dengan menggunakan method *Handle* dari pointer struct *ServeMux*. Method *Handle* menerima 2 parameter berupa string dari route path nya, dan handler nya yang bertipe *http.Handler*. Lalu perhatikan bahwa kita menggunakan 2 middleware sebelum kita dapat mencapai function *greet* yang merupakan endpointnya.

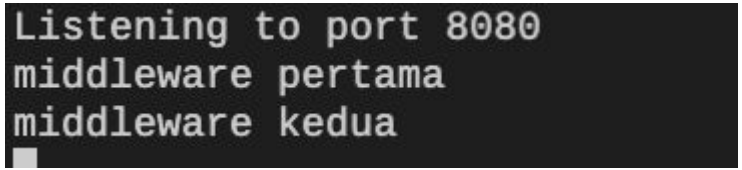
```
9   func main() {
10
11    mux := http.NewServeMux()
12
13    endpoint := http.HandlerFunc(greet)
14
15    mux.Handle("/", middleware1(middleware2(endpoint)))
16
17    fmt.Println("Listening to port 8080")
18
19    err := http.ListenAndServe(":3000", mux)
20
21    log.Fatal(err)
22  }
23
24  func greet(w http.ResponseWriter, r *http.Request) {
25    w.Write([]byte("Hello World!!!"))
26  }
27
28  func middleware1(next http.Handler) http.Handler {
29
30    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
31      fmt.Println("middleware pertama")
32      next.ServeHTTP(w, r)
33    })
34  }
35
36  func middleware2(next http.Handler) http.Handler {
37    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
38      fmt.Println("middleware kedua")
39      next.ServeHTTP(w, r)
40    })
41  }
42
```

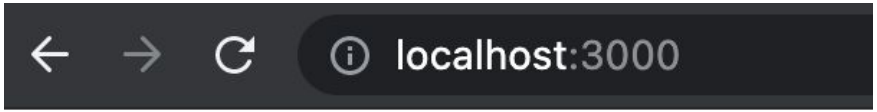
Middleware

Sekarang mari kita jalankan server dari aplikasi kita, dan buka browser dan arahakan kepada url http://localhost:3000/.

Jika sudah, maka kita akan melihat logging yang sudah kita buat pada ke-2 middleware kita seperti pada gambar pertama disebelah kanan.

Dan pada browser kita akan terlihat seperti pada gambar kedua.





Hello World!!!