MATERI

Hari 13 - JSON data & Web Server

JSON data

JSON atau *Javascript Object Notation* adalah notasi standar yang umum digunakan untuk komunikasi data dalam web. JSON merupakan subset dari *javascript*.

Go menyediakan package `encoding/json` yang berisikan banyak fungsi untuk kebutuhan operasi json.

Di bab ini, kita akan belajar cara untuk konverstri string yang berbentuk json menjadi object Go, dan sebaliknya.

Decode JSON Ke Variabel Object Struct

Di Go, data json dituliskan sebagai `string`. Dengan menggunakan `json.Unmarshal`, json string bisa dikonversi menjadi bentuk object, entah itu dalam bentuk `map[string]interface{}` ataupun object struct.

Program berikut ini adalah contoh cara decoding json ke bentuk object. Pertama import package yang dibutuhkan, lalu siapkan struct `User`.

```
package main

import "encoding/json"
import "fmt"

type User struct {
  FullName string `json:"Name"`
  Age int
}
```

Struct `user` ini nantinya digunakan untuk membuat variabel baru penampung hasil decode json string. Proses decode sendiri dilakukan lewat fungsi `json.Unmarshal()`, dengan json string tersebut dimasukan ke statement fungsi tersebut.

Silakan tulis kode berikut.

```
func main() {
  var jsonString = `{"Name": "john doe", "Age":
27}`

  var jsonData = []byte(jsonString)

  var data User

  var err = json.Unmarshal(jsonData, &data)
  if err != nil {
    fmt.Println(err.Error())
    return
  }

  fmt.Println("user :", data.FullName)
  fmt.Println("age  :", data.Age)
}
```

Fungsi unmarshal hanya menerima data json dalam bentuk `[]byte`, maka dari itu data json string pada kode di atas di-casting terlebih dahulu ke tipe `[]byte` sebelum dipergunakan pada fungsi unmarshal.

Juga, perlu diperhatikan, argument ke-2 fungsi unmarshal harus diisi dengan **pointer** dari object yang nantinya akan menampung hasilnya.

```
C:\Workspaces\Materi-Golang\JSON>go run main.go
user : john doe
age  : 27
```

Jika kita perhatikan lagi, pada struct `User`, salah satu property-nya yaitu `FullName` memiliki **tag** `json:"Name"`. Tag tersebut digunakan untuk mapping informasi json ke property yang bersangkutan.

Data json yang akan diparsing memiliki 2 property yaitu `Name` dan `Age`. Kebetulan penulisan `Age` pada data json dan pada struktur struct adalah sama, berbeda dengan `Name` yang tidak ada pada struct.

Decode JSON Ke `map[string]interface{}` & `interface{}`

Tak hanya ke object cetakan struct, target decoding data json juga bisa berupa variabel bertipe `map[string]interface{}`.

```
var data1 map[string]interface{}
json.Unmarshal(jsonData, &data1)

fmt.Println("user :", data1["Name"])
fmt.Println("age  :", data1["Age"])
```

Variabel bertipe `interface{}` juga bisa digunakan untuk menampung hasil decode. Dengan catatan pada pengaksesan nilai property, harus dilakukan casting terlebih dahulu ke `map[string]interface{}`.

```
var data2 interface{}
json.Unmarshal(jsonData, &data2)

var decodedData = data2.(map[string]interface{})
fmt.Println("user :", decodedData["Name"])
fmt.Println("age  :", decodedData["Age"])
```

Decode Array JSON Ke Array Object

Decode data dari array json ke slice/array object masih sama, siapkan saja variabel penampung hasil decode dengan tipe slice struct. Contohnya bisa dilihat pada kode berikut.

```
var jsonString = `[
    {"Name": "john doe", "Age": 27},
    {"Name": "doe john", "Age": 32}
]`

var data []User

var err = json.Unmarshal([]byte(jsonString),
&data)
if err != nil {
    fmt.Println(err.Error())
    return
}

fmt.Println("user 1:", data[0].FullName)
fmt.Println("user 2:", data[1].FullName)
```

Encode object Ke JSON String

Setelah sebelumnya dijelaskan beberapa cara decode data dari json string ke object, sekarang kita akan belajar cara **encode** data object ke bentuk json string.

Fungsi `json.Marshal` digunakan untuk encoding data ke json string. Sumber data bisa berupa variabel object cetakan struct, `map[string]interface{}`, atau slice.

Pada contoh berikut, data slice struct dikonversi ke dalam bentuk json string. Hasil konversi berupa `[]byte`, casting terlebih dahulu ke tipe `string` agar bisa ditampilkan bentuk json string-nya.

```
var object = []User{{"john doe", 27}, {"doe
john", 32}}
var jsonData, err = json.Marshal(object)
if err != nil {
    fmt.Println(err.Error())
    return
}

var jsonString = string(jsonData)
fmt.Println(jsonString)
```

Output:

```
C:\Workspaces\Materi-Golang\JSON>go run main.go
[{"Name":"john doe","Age":27},{"Name":"doe john","Age":32}]
```

Go menyediakan package `net/http`, berisi berbagai macam fitur untuk keperluan pembuatan aplikasi berbasis web. Termasuk di dalamnya web server, routing, templating, dan lainnya.

Go memiliki web server sendiri, dan web server tersebut berada di dalam Go, tidak seperti bahasa lain yang servernya terpisah dan perlu diinstal sendiri (seperti PHP yang memerlukan Apache, .NET yang memerlukan IIS).

Di bab ini kita akan belajar cara pembuatan aplikasi web sederhana dan pemanfaatan template untuk mendesain view.

Membuat Aplikasi Web Sederhana

Package `net/http` memiliki banyak sekali fungsi yang bisa dimanfaatkan. Di bagian ini kita akan mempelajari beberapa fungsi penting seperti *routing* dan *start server*.

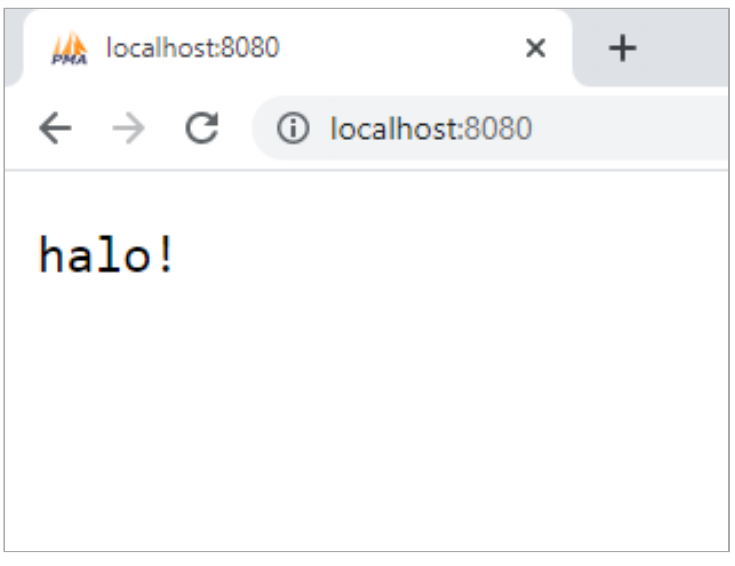
Program di bawah ini merupakan contoh sederhana untuk memunculkan text di web ketika url tertentu diakses.

```
package main

import "fmt"
import "net/http"

func index(w http.ResponseWriter, r
*http.Request) {
  fmt.Fprintln(w, "apa kabar!")
}

func main() {
  http.HandleFunc("/", func(w
http.ResponseWriter, r *http.Request) {
    fmt.Fprintln(w, "halo!")
  })

  http.HandleFunc("/index", index)

  fmt.Println("starting web server at
http://localhost:8080/")

  http.ListenAndServe(":8080", nil)
}
```

Jalankan program tersebut.

```
C:\Workspaces\Materi-Golang\WebServer>go run main.go
starting web server at http://localhost:8080/
```

Jika muncul dialog **Do you want the application "main" to accept incoming network connections?** atau sejenis, pilih allow. Setelah itu, buka url http://localhost/ dan http://localhost/index lewat browser.



Fungsi `http.HandleFunc()` digunakan untuk routing aplikasi web. Maksud dari routing adalah penentuan aksi ketika url tertentu diakses oleh user.

Pada kode di atas 2 rute didaftarkan, yaitu `/` dan `/index`. Aksi dari rute `/` adalah menampilkan text `"halo"` di halaman website. Sedangkan `/index` menampilkan text `"apa kabar!"`.

Fungsi `http.HandleFunc()` memiliki 2 buah parameter yang harus diisi. Parameter pertama adalah rute yang diinginkan. Parameter kedua adalah *callback* atau aksi ketika rute tersebut diakses. Callback tersebut bertipe fungsi `func(w http.ResponseWriter, r *http.Request)`.

aplikasi adalah 1 buah server berbeda.

Pada contoh di atas, server dijalankan pada port `8080`.

Perlu diingat, setiap ada perubahan pada file `.go`, `go run` harus dipanggil lagi.

Untuk menghentikan web server, tekan **CTRL+C** pada terminal atau CMD, dimana pengeksekusian aplikasi berlangsung.

Referensi Tulisan:

- https://dasarpemrogramangolang.novalagung.com/A-web-server.html
- https://dasarpemrogramangolang.novalagung.com/A-json.html

Rating - Feedback

Berikan Rating pada posting ini:

0

Berikan kritik dan saran..

Submit