Digitalent Kominfo 2020 - Pelatihan IT Perbankan
Hands-On Pertemuan 12 Pengantar REST API
Penerapan REST API berbasis Bahasa Pemrograman Go Lang

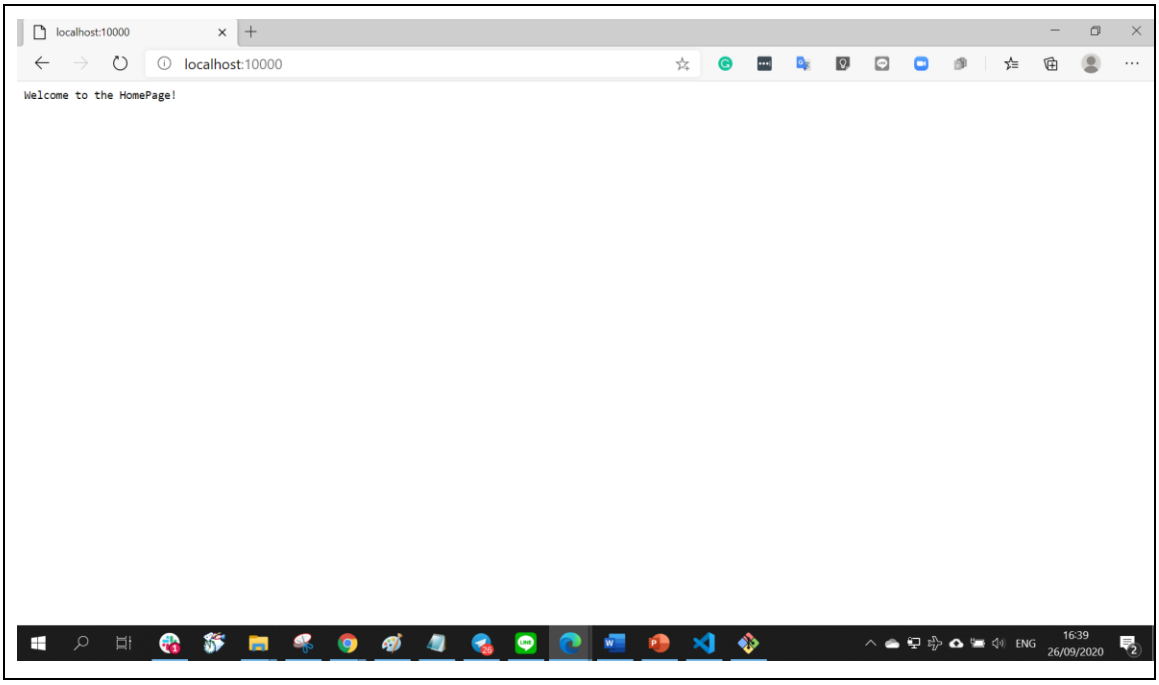
Modul hands-on ini memfasilitasi pembentukan kompetensi untuk mampu memahami konsep dasar REST API dan penerapan menggunakan Golang. Target dari modul ini adalah peserta mampu membuat REST API sederhana dengan Bahasa Pemrograman Go Lang.

1. **Memulai dengan API Dasar**

   Untuk memulai kita harus membuat server yang sangat sederhana yang dapat menangani permintaan HTTP. Untuk melakukan ini, kita akan membuat file baru bernama main.go. Di dalam main.gofile ini kita ingin menentukan 3 fungsi berbeda. Sebuah fungsi homePage yang akan menangani semua permintaan ke URL root, sebuah fungsi handleRequests yang akan cocok dengan jalur URL yang terkena dengan fungsi yang ditentukan dan fungsi main.

```go
package main

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

func homePage(w http.ResponseWriter, r *http.Request){
    fmt.Fprintf(w, "Welcome to the HomePage!")
    fmt.Println("Endpoint Hit: homePage")
}

func handleRequests() {
    http.HandleFunc("/", homePage)
    log.Fatal(http.ListenAndServe(":10000", nil))
}

func main() {
    handleRequests()
}
```

Jika kita menjalankan ini di komputer kita sekarang, kita akan melihat API yang sangat sederhana mulai di port 10000 (jika belum digunakan oleh proses lain). Jika sekarang kita menavigasi ke http://localhost:10000/di browser lokal kita, kita akan melihat hasil “Welcome to the HomePage!” tercetak di layar kita seperti pada gambar berikut ini. Gambar ini menunjukkan kita telah berhasil membuat basis tempat kita akan membangun REST API kita.



2. Struktur Article

Kita akan menciptakan sebuah REST API yang memungkinkan kita untuk CREATE, READ, UPDATE dan DELETE sebuah artikel di website. Saat kita berbicara tentang ‘CRUD’ API, kita mengacu pada API yang dapat menangani semua tugas ini: Membuat, Membaca, Memperbarui, dan Menghapus.

Sebelum kita dapat memulai, kita harus mendefinisikan struktur `Article` kita . Go memiliki konsep struct yang sempurna hanya untuk skenario ini. Mari buat sebuah struct `Article` yang menampilkan Title, Description (desc) dan Content seperti ini:

```go
type Article struct {
    Title string `json:"Title"`
    Desc string `json:"desc"`
    Content string `json:"content"`
}

// let's declare a global Articles array
// that we can then populate in our main function
// to simulate a database
var Articles []Article
```

Struktur kita berisi 3 properti yang kita butuhkan untuk mewakili semua **`Article`** di situs kita. Agar ini berfungsi, kita juga harus mengimpor package "encoding/json" ke dalam daftar import kita.

Sekarang mari perbarui fungsi main kita sehingga variabel **`Article`** kita diisi dengan beberapa data dummy yang bisa kita ambil dan modifikasi nanti.

```go
func main() {
    Articles = []Article{
        Article{Title: "Hello", Desc: "Article Description", Content: "Article Cont
        Article{Title: "Hello 2", Desc: "Article Description", Content: "Article Co
    }
    handleRequests()
}
```

Sekarang mari beralih ke membuat endpoint /articles yang akan mengembalikan semua **`Article`** yang baru saja kita definisikan di sini.

3. Mengambil Semua Article

Di bagian ini kita akan membuat endpoint REST baru yang ketika dipanggil dengan HTTP Request GET, akan mengembalikan semua artikel untuk situs kita.

Kita pertama-tama akan memulai dengan membuat fungsi baru yang disebut returnAllArticles, yang akan melakukan tugas sederhana mengembalikan variabel **`Article`** yang baru diisi, dikodekan dalam format JSON:

```go
func returnAllArticles(w http.ResponseWriter, r *http.Request){
    fmt.Println("Endpoint Hit: returnAllArticles")
    json.NewEncoder(w).Encode(Articles)
}
```

Panggilan untuk json.NewEncoder(w).Encode(article) berfungsi menyandikan larik artikel kita ke dalam string JSON dan kemudian menulis sebagai bagian dari Response.

Sebelum ini akan bekerja, kita juga perlu menambahkan rute baru ke fungsi handleRequests yang akan memetakan panggilan apa pun ke http://localhost:10000/articles yang baru kita definisikan.

```
func handleRequests() {
    http.HandleFunc("/", homePage)
    // add our articles route and map it to our
    // returnAllArticles function like so
    http.HandleFunc("/articles", returnAllArticles)
    log.Fatal(http.ListenAndServe(":10000", nil))
}
```

Sekarang setelah kita melakukan ini, jalankan kode dengan mengetik go run main.go dan kemudian buka http://localhost:10000/articles di browser Anda dan Anda akan melihat representasi JSON dari daftar artikel Anda seperti ini: