Digitalent Kominfo - Pengenalan Golang

Modul hands-on ini memfasilitasi pembentukan kompetensi untuk mampu memahami konsep dasar bahasa pemrograman Golang. Target dari modul ini adalah peserta mampu memahami dan membuat program sederhana menggunakan bahasa Golang. Kode lengkap dari modul pelatihan ini dapat dilihat melalui link berikut https://github.com/FadhlanHawali/Digitalent-Kominfo_Introduction-Golang .

BAGIAN I - Menginstall Go Lang

Untuk melakukan penginstallan Go Lang, langkah lengkap nya dapat dilihat pada dokumentasi official pada website Golang berikut sesuai dengan sistem operasinya https://golang.org/doc/install.

BAGIAN II

Pada kegiatan Lab Mandiri ini, akan dibagi menjadi 3 bagian dalam tahap pengenalan bahasa pemrograman Golang.

1. Memahami konsep package, variable, dan functions
   Untuk melihat kode lengkap dari bagian ini pengguna dapat melihat pada folder Basic_Package_Variable_Function dari repo github yang sudah tersedia.
   a. main.go
      Pada main.go ini disimulasikan kita mengimport package dari file package_1 untuk melakukan operasi penjumlahan dimana operasi tersebut terletak ada function Add pada package 1. Pada main.go ini juga

```go
package main

import (
    "fmt"
    "github.com/FadhlanHawali/Digitalent-Kominfo_Introduction-Golang/Basic_Package_Variable_Functions/package_1"
)

func main () {

    var x,y int

    nama1,nama2 := "fadhlan","hawali"

    x = 10
    y = 5

    result1 := package_1.Add(x,y)
    fmt.Println(fmt.Sprintf("Hasil penjumlahannya adalah %d",result1))

    fmt.Println(package_1.CustomPrint(nama1))
    fmt.Println(package_1.CustomPrint(nama2))

    result2,flag2:= package_1.AddDoubleReturn(x,y)

    fmt.Println(fmt.Sprintf("Hasil penjumlahannya adalah %d , %t",result2,flag2))
}
```

b. package_1/package1.go

```go
package package_1

import "fmt"

func PrintPackage1(){
    fmt.Println("Print dari package_1")
}

func CustomPrint (str string) string{
    return fmt.Sprintf("Halo, nama saya %s",str)
}

//Tidak Bisa Di Import karena tidak Kapital
func add(x int, y int) int {
    return x + y
}

//Bisa Di Import Karena Kapital
func Add(x int, y int) int {
    return x + y
}

func AddDoubleReturn(x int, y int) (int,bool) {
    return x + y, true
}
```

c. package_2/package2.go

```go
package package_2

import (
    "fmt"
    "github.com/FadhlanHawali/Digitalent-Kominfo_Introduction-Golang/Basic_Package_Variable_Functions/package_1"
)

func PrintPackage2(){
    fmt.Println("Print dari package_2")
}

func ImportDariPackage1(){
    package_1.PrintPackage1()
}
```

2. Memahami konsep flow control

```go
package main

import (
    "fmt"
    "runtime"
)

func main (){
    arr :=  []string{"satu","dua","tiga","empat"}
    perulangan(arr)
    switchstatement()
}
```

a. If Statement

```go
func ifstatement (x,y,max int) bool{

    //Uncomment kode ini untuk melihat tipe 1
    if x < y {
        return true
    }else {
        return false
    }

    //Uncomment kode ini untuk melihat tipe 2
    if v:=x*y;v < max{
        return true
    }else if v>max {
        return false
    }else {
        return true
    }

    //Uncomment kode ini untuk melihat tipe 3
    if ok := x<y;ok{
        return ok
    }else {
        return false
    }

}
```

b. For Statement

```go
func perulangan(arr []string){
    //Basic general For
    for i:= 0; i< len(arr);i++{
        fmt.Println(arr[i])
    }

    //For dengan prinsip Range
    for i,v:= range arr{
        fmt.Println(fmt.Sprintf("Index ke %d  : %s",i,v))
    }

    //Simplified For (While)
    i:=0
    for i<len(arr) {
        fmt.Println(fmt.Sprintf("Index ke %d  : %s",i,arr[i]))
        i++
    }
}
```

c. Switch Case Statement

```go
func perulangan(arr []string){
    //Basic general For
    for i:= 0; i< len(arr);i++{
        fmt.Println(arr[i])
    }

    //For dengan prinsip Range
    for i,v:= range arr{
        fmt.Println(fmt.Sprintf("Index ke %d  : %s",i,v))
    }

    //Simplified For (While)
    i:=0
    for i<len(arr) {
        fmt.Println(fmt.Sprintf("Index ke %d  : %s",i,arr[i]))
        i++
    }
}
```

3. Memahami konsep tipe data
    a. Array / Slice

```go
func array (){
    //array
    primes := [6]int{2, 3, 5, 7, 11, 13}
    fmt.Println(primes)
}

func slices (){

    names := []string{
        "nama_1",
        "nama_2",
        "nama_3",
        "nama_4",
    }
    fmt.Println(names)

    a := names[0:2]
    b := names[1:3]
    fmt.Println(a, b)

    b[0] = "--"
    fmt.Println(a, b)
    fmt.Println(names)
}
```

b. Map

```go
type Geo struct {
    Lat float64
    Long float64
}

func mapExample (){
    mapGeo := make(map[string]interface{})

    mapGeo["Yogyakarata"] = Geo{
        Lat:  7.7956,
        Long: 110.3695,
    }

    mapGeo["Jakarta"] = Geo{
        Lat:  6.2088,
        Long: 106.8456,
    }


    //print map key and value
    for i,v := range mapGeo{
        fmt.Println(fmt.Sprintf("Koordinat %s  : %v",i,v))
    }
}
```

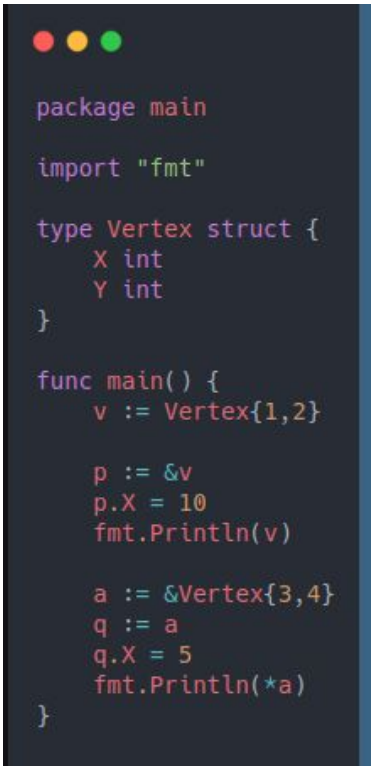
c. Method

```go
package main


import (
    "fmt"
)

type Sisi struct {
    Panjang int
    Lebar int
}

func (s Sisi) Luas() int {
    return s.Panjang * s.Lebar
}

func(s *Sisi) Scale(i int){
    s.Lebar = s.Lebar*i
    s.Panjang = s.Panjang*i
}

func main() {
    s := Sisi{3, 4}
    s.Scale(3)
    fmt.Println(s.Luas())
}
```

d. Pointer

```go
package main

import "fmt"

func main(){
    i, j := 42, 2701

    p := &i         // point to i
    fmt.Println(*p) // read i through the pointer
    *p = 21         // set i through the pointer
    fmt.Println(i)  // see the new value of i

    p = &j         // point to j
    *p = *p / 37   // divide j through the pointer
    fmt.Println(j) // see the new value of j
}
```

e. Struct

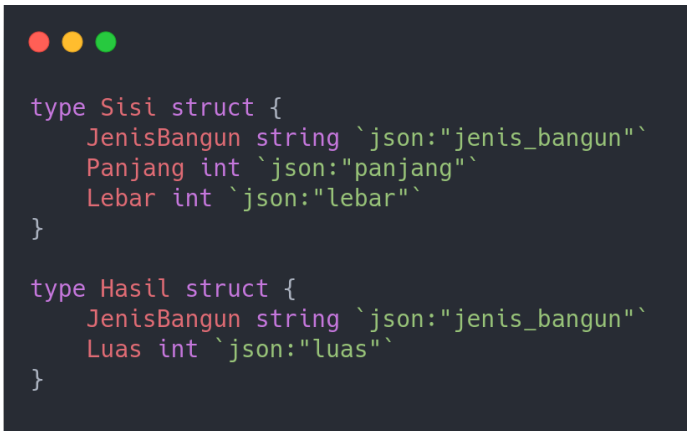


```go
package main

import "fmt"

type Vertex struct {
    X int
    Y int
}

func main() {
    v := Vertex{1,2}

    p := &v
    p.X = 10
    fmt.Println(v)

    a := &Vertex{3,4}
    q := a
    q.X = 5
    fmt.Println(*a)
}
```

BAGIAN III

Pada kegiatan Lab Mandiri ini, peserta akan mencoba untuk mengimplementasi seluruh konsep pemrograman GoLang yang sudah dijelaskan pada bagian sebelumnya. Keluaran dari bagian ini adalah peserta mampu membuat sebuah REST API sederhana dengan mengimplementasikan konsep-konsep dasar pemrograman GoLang, sebagai contoh pada bagian ini adalah menghitung luas dari sebuah bangunan. Kode lengkap ini dapat dilihat pada Repo yang telah diberikan sebelumnya pada folder Simple_Rest_API

1. Deklarasi Entitas Data



```go
type Sisi struct {
    JenisBangun string `json:"jenis_bangun"`
    Panjang int `json:"panjang"`
    Lebar int `json:"lebar"`
}

type Hasil struct {
    JenisBangun string `json:"jenis_bangun"`
    Luas int `json:"luas"`
}
```

2. Membuat Fungsi Main

```go
func main(){
    router := mux.NewRouter()
    router.HandleFunc("/api/hitung-luas",Luas)
    log.Fatal(http.ListenAndServe(":8080", router))
}
```

3. Membuat Method dan Function RumusLuas

```go
func (s *Sisi) RumusLuas () int {
    return s.Panjang * s.Lebar
}
```

4. Membuat Handler Luas

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

    var hasilHitung []Hasil
    var sisi []Sisi
    if r.Method != "POST"{
        WrapAPIError(w,r,http.StatusText(http.StatusMethodNotAllowed),http.StatusMethodNotAllowed)
        return
    }

    body, err := ioutil.ReadAll(r.Body)
    defer r.Body.Close()
    if err != nil{
        WrapAPIError(w,r,"can't read body",http.StatusBadRequest)
        return
    }

    err = json.Unmarshal(body, &sisi)
    if err != nil {
        WrapAPIError(w,r,"error unmarshal : "+err.Error(),http.StatusInternalServerError)
        return
    }

    for _,v := range sisi{
        hasilHitung = append(hasilHitung,Hasil{
            JenisBangun: v.JenisBangun,
            Luas:        v.RumusLuas(),
        })
    }

    WrapAPIData(w,r,hasilHitung,http.StatusOK,"success")
}
```

5. Membuat Wrapper Untuk Output Data

```go
func WrapAPIError(w http.ResponseWriter, r *http.Request, message string, code int) {
    w.Header().Set("Content-Type", "application/json")
    w.WriteHeader(code)
    result, err := json.Marshal(map[string]interface{}{
        "code":          code,
        "error_type":    http.StatusText(code),
        "error_details": message,
    })
    if err == nil {
        w.Write(result)
    } else {
        log.Println(fmt.Sprintf("can't wrap API error : %s", err))
    }
}

func WrapAPISuccess(w http.ResponseWriter, r *http.Request, message string, code int) {
    w.Header().Set("Content-Type", "application/json")
    w.WriteHeader(code)
    result, err := json.Marshal(map[string]interface{}{
        "code":   code,
        "status": message,
    })
    if err == nil {
        log.Println(message)
        w.Write(result)
    } else {
        log.Println(fmt.Sprintf("can't wrap API success : %s", err))
    }
}

func WrapAPIData(w http.ResponseWriter, r *http.Request, data interface{}, code int, message
string) {
    w.Header().Set("Content-Type", "application/json")
    w.WriteHeader(code)
    result, err := json.Marshal(map[string]interface{}{
        "code":   code,
        "status": message,
        "data":   data,
    })
    if err == nil {
        log.Println(message)
        w.Write(result)
    } else {
        log.Println(fmt.Sprintf("can't wrap API data : %s", err))
    }
}
```