[PRACTICE] - BUILDING REST API PROJECT-BASED

Rest API

Pada Penutup Pertemuan ini kita akan membuat sebuah project based RestAPI yang sudah bukan hal yang jarang dilakukan yaitu menggunakan konsep RestAPI.

Sebelum memulai tentu kalian harus menginstall semua requirement yang dibutuhkan oleh Golang. Kalian bisa baca dokumentasi untuk menginstall Golang pada halaman di sini.

Di sini akan digunakan **gin-gonic/gin** sebagai router dari Golang, seperti misal pada Node.Js adalah **express** dan untuk database yang digunakan adalah mysql, lalu untuk ORM yang digunakan seperti **Bookshelf** pada JS, adalah **gorm** untuk menginstall dependensi yang digunakan bisa kalian buka terminal kemudian tuliskan command dibawah:

```
go get -u "github.com/gin-gonic/gin"

go get -u "github.com/jinzhu/gorm"

go get -u "github.com/jinzhu/gorm/dialects/mysql"

go get -u "github.com/go-sql-driver/mysql"
```

Membuat Struktur Data #1

Buatlah sebuah folder di dalam folder utama, dalam kasus saya di dalam folder RestFullApi kemudian saya namakan folder tersebut structs. Di dalam folder structs tersebut buat sebuah file berekstensi .go.

File ini berguna untuk mendeklarasikan variable-variable yang dimiliki sebuah data, sebagai contoh yang akan dibuat adalah struktur data person.

structs.go

```
1  package structs
2
3  import "github.com/jinzhu/gorm"
4
5  type Person struct {
6    gorm.Model
7    First_Name string
8    Last_Name string
9  }
```

Perlu diperhatikan bahwa nama sebuah struktur data harus diawali dengan huruf kapital. Kemudian karena kita menggunakan ORM gorm maka harus dimasukkan pada setiap struktur data yang dibuat dan melalui database.

Pada line 1 dilihat bahwa package yang digunakan adalah structs karena ingin dibedakan menurut nama folder yang digunakan. Sebenarnya masih terdapat satu struktur data lagi yang digunakan, namun saya letakkan di dalam folder controller karena penggunaan struktur data tersebut hanya di dalam file controller.

Membuat Struktur Data #2

Struktur data kedua ada pada baris koding di bawah.

gorm.go

```go
package structs

import "github.com/jinzhu/gorm"

type Person struct {
  gorm.Model
  First_Name string
  Last_Name string
}
```

Dinamakan **gorm.go** karena merupakan struktur data yang akan langsung berhubungan dengan ORM database, lalu saya letakkan dalam folder controller yang berada setara dengan folder struct.

Membuat Controller #1

Controller digunakan untuk membuat fungsi yang akan digunakan oleh setiap API yang terbentuk di akhir project ini.

Perlu diketahui bahwa file controller berikut diletakkan di dalam folder controllers bersama dengan file gorm.go.

Untuk controller nya, karena hanya terdapat satu table dalam database, jadi saya menamai sesuai dengan struktur data yang dibuat. Berikut file person.go:

#1

```go
package controllers

import (
    "net/http"

    "../structs"
    "github.com/gin-gonic/gin"
)

// to get one data with {id}
func (idb *InDB) GetPerson(c *gin.Context) {
    var (
        person structs.Person
        result gin.H
    )
    id := c.Param("id")
    err := idb.DB.Where("id = ?", id).First(&person).Error
    if err != nil {
        result = gin.H{
            "result": err.Error(),
            "count":  0,
        }
    } else {
        result = gin.H{
            "result": person,
            "count":  1,
        }
    }

    c.JSON(http.StatusOK, result)
}
```

#2

```go
// get all data in person
func (idb *InDB) GetPersons(c *gin.Context) {
    var (
        persons []structs.Person
        result  gin.H
    )

    idb.DB.Find(&persons)
    if len(persons) <= 0 {
        result = gin.H{
            "result": nil,
            "count":  0,
        }
    } else {
        result = gin.H{
            "result": persons,
            "count":  len(persons),
        }
    }

    c.JSON(http.StatusOK, result)
}
```

Membuat Controller #2

#3

```go
// create new data to the database
func (idb *InDB) CreatePerson(c *gin.Context) {
    var (
        person structs.Person
        result gin.H
    )
    first_name := c.PostForm("first_name")
    last_name := c.PostForm("last_name")
    person.First_Name = first_name
    person.Last_Name = last_name
    idb.DB.Create(&person)
    result = gin.H{
        "result": person,
    }
    c.JSON(http.StatusOK, result)
}
```

#4

```go
// update data with {id} as query
func (idb *InDB) UpdatePerson(c *gin.Context) {
    id := c.Query("id")
    first_name := c.PostForm("first_name")
    last_name := c.PostForm("last_name")
    var (
        person    structs.Person
        newPerson structs.Person
        result    gin.H
    )

    err := idb.DB.First(&person, id).Error
    if err != nil {
        result = gin.H{
            "result": "data not found",
        }
    }
    newPerson.First_Name = first_name
    newPerson.Last_Name = last_name
    err = idb.DB.Model(&person).Updates(newPerson).Error
    if err != nil {
        result = gin.H{
            "result": "update failed",
        }
    } else {
        result = gin.H{
            "result": "successfully updated data",
        }
    }
    c.JSON(http.StatusOK, result)
}
```

Membuat Controller #3

#5

```
105  // delete data with {id}
106  func (idb *InDB) DeletePerson(c *gin.Context) {
107      var (
108          person structs.Person
109          result gin.H
110      )
111      id := c.Param("id")
112      err := idb.DB.First(&person, id).Error
113      if err != nil {
114          result = gin.H{
115              "result": "data not found",
116          }
117      }
118      err = idb.DB.Delete(&person).Error
119      if err != nil {
120          result = gin.H{
121              "result": "delete failed",
122          }
123      } else {
124          result = gin.H{
125              "result": "Data deleted successfully",
126          }
127      }
128
129      c.JSON(http.StatusOK, result)
130  }
```

Hal yang penting untuk diketahui adalah pada baris kode nomor 17, 30, 40, 66, 84 yang merupakan bagian dari gorm untuk melakukan sebuah query pada database.

Membuat koneksi ke database

Untuk bisa mendapatkan data dari database yang dalam hal ini memakai **mysql** maka harus terjadi koneksi antara server dengan database.

Berikut file untuk melakukan koneksi ke database saya taruh pada folder **config** dengan nama file **config.go**. Dalam melakukan koneksi ke database berbeda antar-database. Contoh koneksi ke **mysql**.

```
134  package config
135
136  import (
137      "../structs"
138      "github.com/jinzhu/gorm"
139  )
140
141  // DBInit create connection to database
142  func DBInit() *gorm.DB {
143      db, err := gorm.Open("mysql", "root:@tcp(127.0.0.1:3306)/godb?charset=utf8&parseTime=True&loc=Local")
144      if err != nil {
145          panic("failed to connect to database")
146      }
147
148      db.AutoMigrate(structs.Person{})
149      return db
150  }
```

Membuat main package

Langkah terakhir adalah membuat main file yang digunakan untuk menjalankan server API yang telah kita buat seperti di atas. File berikut diletakkan di luar semua folder namun masih di dalam folder RestFullApi. Gambar disebelah kanan merupakan isi dari file base.go.

Jalankan :

go run base.go dan lihat hasilnya di POSTMAN.

```go
package main

import (
        "./config"
        "./controllers"
        "github.com/gin-gonic/gin"
        _ "github.com/go-sql-driver/mysql"
)

func main() {
        db := config.DBInit()
        inDB := &controllers.InDB{DB: db}

        router := gin.Default()

        router.GET("/person/:id", inDB.GetPerson)
        router.GET("/persons", inDB.GetPersons)
        router.POST("/person", inDB.CreatePerson)
        router.PUT("/person", inDB.UpdatePerson)
        router.DELETE("/person/:id", inDB.DeletePerson)
        router.Run(":3000")
}
```