



Soal ORM & Code Structure (MVC)

Date March 28, 2023 → April 2, 2023

Assign Empty

Status Empty

Minggu Minggu-7 (puasa)

Tipe Soal Section

▼ Praktikum ORM & Code Structure (MVC)

Overview

Peserta mampu membuat REST API menggunakan Echo dan GORM.

Soal

▼ Soal Prioritas 1 (80)

1. Buat API CRUD User dengan spesifikasi seperti berikut!

- Pada bagian Sample Code artinya sudah disediakan contoh code yang bisa kamu implementasikan.
- Pada bagian Need Solution Code kamu perlu membuat sendiri code-nya!

Route	HTTP Method	Deskripsi	Profil
/users	GET	Mendapatkan semua data user	Sample Code
/users/:id	GET	Mendapatkan data user berdasarkan ID	Need Solution Code
/users	POST	Membuat user baru	Sample Code
/users/:id	PUT	Mengubah data user	Need Solution Code

		berdasarkan ID	
<code>/users/:id</code>	<code>DELETE</code>	Menghapus data user berdasarkan ID	Need

```

package main

import (
    "fmt"
    "net/http"
    "strconv"

    "github.com/jinzhu/gorm"
    _ "github.com/jinzhu/gorm/dialects/mysql"
    "github.com/labstack/echo"
)

var (
    DB *gorm.DB
)

func init() {
    InitDB()
    InitialMigration()
}

type Config struct {
    DB_Username string
    DB_Password string
    DB_Port     string
    DB_Host     string
    DB_Name     string
}

func InitDB() {

    config := Config{
        DB_Username: "root",
        DB_Password: "root123",
        DB_Port:     "3306",
        DB_Host:     "localhost",
        DB_Name:     "crud_go",
    }

    connectionString := fmt.Sprintf("%s:%s@tcp(%s:%s)/%s?charset=utf8&parseTime=1",
        config.DB_Username,
        config.DB_Password,
        config.DB_Host,
        config.DB_Port,
        config.DB_Name)

```

```

        config.DB_Password,
        config.DB_Host,
        config.DB_Port,
        config.DB_Name,
    )

    var err error
    DB, err = gorm.Open("mysql", connectionString)
    if err != nil {
        panic(err)
    }
}

type User struct {
    gorm.Model
    Name      string `json:"name" form:"name"`
    Email     string `json:"email" form:"email"`
    Password  string `json:"password" form:"password"`
}

func InitialMigration() {
    DB.AutoMigrate(&User{})
}

// get all users
func GetUsersController(c echo.Context) error {
    var users []User

    if err := DB.Find(&users).Error; err != nil {
        return echo.NewHTTPError(http.StatusBadRequest, err.Error())
    }
    return c.JSON(http.StatusOK, map[string]interface{}{
        "message": "success get all users",
        "users":   users,
    })
}

// get user by id
func GetUserController(c echo.Context) error {
    // your solution here
}

// create new user
func CreateUserController(c echo.Context) error {
    user := User{}
    c.Bind(&user)

    if err := DB.Save(&user).Error; err != nil {
        return echo.NewHTTPError(http.StatusBadRequest, err.Error())
    }
}

```

```

    }
    return c.JSON(http.StatusOK, map[string]interface{}{
        "message": "success create new user",
        "user":    user,
    })
}

// delete user by id
func DeleteUserController(c echo.Context) error {
    // your solution here
}

// update user by id
func UpdateUserController(c echo.Context) error {
    // your solution here
}

func main() {
    // create a new echo instance
    e := echo.New()
    // Route / to handler function
    e.GET("/users", GetUsersController)
    e.GET("/users/:id", GetUserController)
    e.POST("/users", CreateUserController)
    e.DELETE("/users/:id", DeleteUserController)
    e.PUT("/users/:id", UpdateUserController)

    // start the server, and log if it fails
    e.Logger.Fatal(e.Start(":8000"))
}

```

kode sampel yang bisa digunakan. Untuk ORM yang digunakan adalah [gorm.io](#).

2. Berdasarkan soal nomor 1, yang telah dikerjakan. Implementasikan arsitektur MVC.

▼ Soal Prioritas 2 (20)

- Berdasarkan soal prioritas 1 yang telah dikerjakan, tambahkan fitur CRUD untuk data buku dengan spesifikasi sebagai berikut.
 - Data buku terdiri dari ID, judul, penulis dan penerbit.
 - Menggunakan arsitektur MVC.

Route	HTTP Method	Deskripsi
<code>/books</code>	<code>GET</code>	Mendapatkan se
<code>/books/:id</code>	<code>GET</code>	Mendapatkan d berdasarkan ID
<code>/books</code>	<code>POST</code>	Membuat buku


/books/:id	PUT	Mengubah data ID
/books/:id	DELETE	Menghapus data ID

▼ Soal Eksplorasi (20)

- Berdasarkan soal prioritas 1 yang telah dikerjakan, tambahkan fitur untuk mengelola data blog dengan kriteria sebagai berikut:
 - Data blog terdiri dari ID, ID user, judul dan konten.
 - Terdapat relasi antara entitas user dengan entitas blog.
 - Menggunakan arsitektur MVC
 - Rincian endpoint dapat dilihat pada tabel berikut:

Route	HTTP	Description
/blogs	POST	Membuat data blog
/blogs	GET	Mendapatkan semua data
/blogs/:id	GET	Melihat rincian data blog berdasarkan ID.
/blogs/:id	PUT	Mengubah data blog
/blogs/:id	DELETE	Menghapus data blog.

Note

- simpan project kalian ke dalam github yang telah kalian buat. jangan lupa untuk screen shoot dan membuat review terkait materi yang kalian pelajari sekarang
- Standart penilaian :  Standard penilaian