PRAKTIKUM MCS BAB 7

Servo Controller

PENDAHULUAN

Pada praktikum MCS bab 7 akan belajar bagaimana caranya membangun RESFTUL API yang digunakan untuk menggerakan servo. Untuk menggerakan servo dengan restful api memiliki cara kerja microcontroller yang selalu memantau database. Apabila terdapat perubahan di database maka microcontroller akan menghasilkan output untuk menggerakkan servo.

PRAKTIKUM BAB 7

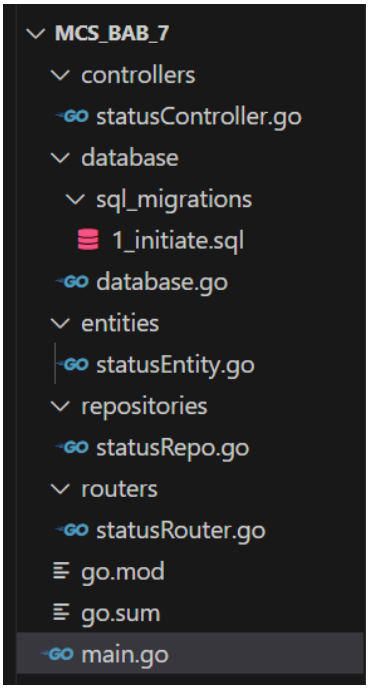
Pada praktikum kali ini database memiliki table yang bernama status dan atributnya yaitu id dengan tipe integer primary key dan srv_status dengan tipe data integer. Field id akan berisi angka 1 dan tidak ada data id lainnya. srv_status akan diisi angka 0 atau 1, angka 0 dan 1 akan dibuat perkondisian utnuk membuat servo bergerak. Buat folder project misalnya "mcs_bab_7" kemudian buka command prompt dan ketik command berikut :

```
go mod init mcs_bab_7
```

Setelah muncul file go.mod masukkan juga command lainnya yaitu :

```
go get -u "github.com/gin-gonic/gin"
go get -u "github.com/lib/pq"
go get -u "github.com/rubenv/sql-migrate"
go get -u "github.com/joho/godotenv"
```

Buat root project dengan folder dan file sebagai berikut :



Root project

Buka file statusEntity.go dan isi dengan code berikut :

```
package entities

type Status struct {
    Id        int `json:"id"`
    SrvStatus int `json:"srv_status"`
}
```

Code dari file statusEntity.go merupakan model yang disesuaikan dengan table database.

Buka file 1_initiate.sql dan isi dengan code berikut :

```
-- +migrate Up
-- +migrate StatementBegin

CREATE TABLE status(
    id INTEGER PRIMARY KEY,
    srv_status INTEGER
);

-- +migrate StatementEnd
```

Pada dasarnya di prakitkum ini table status hanya membutuhkan 1 row sehingga nanti id dibuat primary key dan menjadi patokan perubahan field srv_status. Field srv_status akan berisi 0 dan 1 dimana 0 dan 1 nantinya akan memiliki perkondisiannya masing-masing.

Buka file database.go untuk membuat migrasi databasenya dengan mengisi code berikut :

```
package database

import (
    "database/sql"
    "embed"
    "fmt"

    migrate "github.com/rubenv/sql-migrate"
)

//go:embed sql_migrations/*.sql
var dbMigrations embed.FS
var DbConnection *sql.DB

func DBMigrate(dbParam *sql.DB) {
    migrations := &migrate.EmbedFileSystemMigrationSource{
        FileSystem: dbMigrations,
        Root:       "sql_migrations",
    }

    n, errs := migrate.Exec(dbParam, "postgres", migrations, migrate.Up)

    if errs != nil {
        panic(errs)
    }

    DbConnection = dbParam

    fmt.Println("Migration success applied", n, migrations)
}
```

Buka file statusRepo.go untuk interaksi dengan database menggunakan query sql dengan code berikut :

```
package repositories

import (
    "database/sql"
    "mcs_bab_7/entities"
)

func InitProj(db *sql.DB) (err error) {
    sql := "INSERT INTO status(id, srv_status) values(1, 0)"
    _, err = db.Query(sql)
    return err
```

```
}

func GetStatus(db *sql.DB) (result []entities.Status, err error) {
    sql := "SELECT * FROM status"
    rows, err := db.Query(sql)

    if err != nil {
        return
    }

    defer rows.Close()

    for rows.Next() {
        var data entities.Status
        err = rows.Scan(&data.Id, &data.SrvStatus)
        if err != nil {
            return
        }
        result = append(result, data)
    }
    return
}

func UpdateStatus(db *sql.DB, status entities.Status) (err error) {
    sql := "UPDATE status SET srv_status = $1 WHERE id = 1"
    _, err = db.Exec(sql, status.SrvStatus)
    return
}
```

Fungsi InitProj() digunakan untuk mengisi id. Fungsi InitProj() hanya bisa dilakukan satu kali saja karena query yang dimasukkan adalah mengisi id dengan nilai 1, sedangkan id adalah primary key sehingga apabila dijalankan kembali maka akan terjadi kesalahan. Fungsi GetStatus() digunakan untuk membaca data dari database. Fungsi UpdateStatus() digunakan untuk melakukan perubahan pada field srv_status yang memiliki patokan field id adalah 1.

Buka file statusController.go mengontrol menentukan apa yang akan dilakukan oleh sistem dengan code sebagai berikut :

```
package controllers

import (
    "mcs_bab_7/database"
    "mcs_bab_7/entities"
    "mcs_bab_7/repositories"
    "net/http"
```

```
    "strconv"

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

func InitProj(c *gin.Context) {
    err := repositories.InitProj(database.DbConnection)

    if err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    }

    c.JSON(http.StatusOK, gin.H{})
}

func GetStatus(c *gin.Context) {
    var result gin.H
    status, err := repositories.GetStatus(database.DbConnection)

    if err != nil {
        result = gin.H{
            "result": err.Error(),
        }
    } else {
        result = gin.H{
            "result": status,
        }
    }

    c.JSON(http.StatusOK, result)
}

func UpdateStatus(c *gin.Context) {
    var status entities.Status
    srv_status, _ := strconv.Atoi(c.Param("srv_status"))
    status.SrvStatus = srv_status
    err := repositories.UpdateStatus(database.DbConnection, status)

    if err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
        return
    }

    c.JSON(http.StatusOK, gin.H{"srvStatus": status.SrvStatus})
}
```

Fungsi InitPtoj() digunakan untuk menginisialisasi field id agar memiliki nilai 1 dengan memanggil fungsi InitProj() yang berada di package repositories. Fungsi GetStatus() untuk membaca table status dari database dengan memanggil fungsi GetStatus() yang berada di package repositories. Fungsi UpdateStatus() untuk mengubah field srv_status dengan mengambil nilai dari parameter srv_status.

Buka file statusRouter.go untuk menentukan url yang digunakan sesuai kebutuhan dengan code secagai berikut :

```
package routers

import (
    "mcs_bab_7/controllers"

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

func StartServer() *gin.Engine {
    router := gin.Default()
    router.POST("/servo/init-proj", controllers.InitProj)
    router.GET("/servo/status", controllers.GetStatus)
    router.PUT("/servo/update/:srv_status", controllers.UpdateStatus)
    return router
}
```

Route atau jalur ditentukan dengan menggunakan fungsi-fungsi yang ada di package controllers.

Buat database di PostgreSQL dengan membuka aplikasi pgAdmin 4

Database bab 7

Buka file main.go dan isi dengan code berikut :

```
package main

import (
    "database/sql"
    "fmt"
    "log"
    "mcs_bab_7/database"
    "mcs_bab_7/routers"
```

```
    _ "github.com/lib/pq"
)

const (
    host     = "localhost"
    port     = 5432
    user     = "postgres"
    password = ""
    dbName   = "praktikum_mcs_bab7"
)

var (
    DB  *sql.DB
    err error
)

func main() {
    var PORT = ":8080"

    psqlInfo := fmt.Sprintf(
        `host=%s port=%d user=%s password=%s dbname=%s sslmode=disable`,
        host, port, user, password, dbName,
    )

    DB, err = sql.Open("postgres", psqlInfo)

    if err != nil {
        log.Fatal("Error open DB", psqlInfo)
    }

    database.DBMigrate(DB)

    defer DB.Close()

    routers.StartServer().Run(PORT)
    fmt.Println("DB Success Connected")
}
```

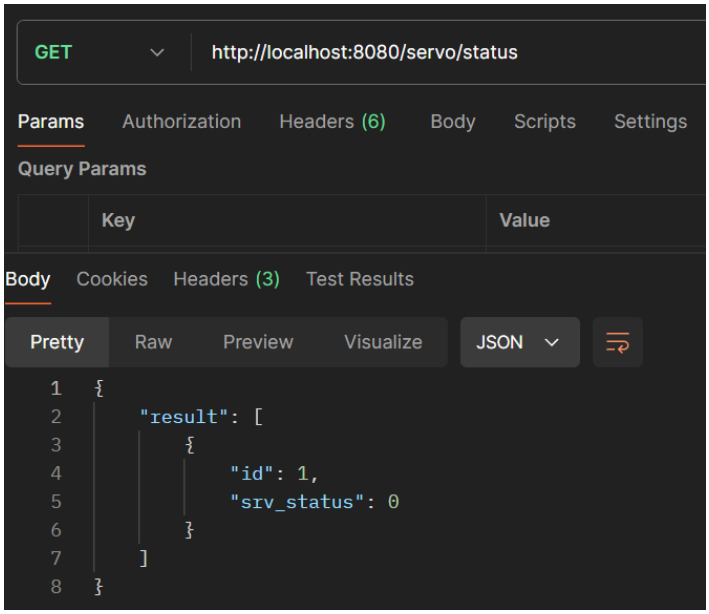
Kita migrasikan konfigurasi database ke PostgreSQL dab menyesyuaikan dengan host, port, user, password, dbname. Serta menjalankan router dengan port 8080.

Jalanklan aplikasi back end yang sudah dibuat dan lihat table status di ogAdmin 4

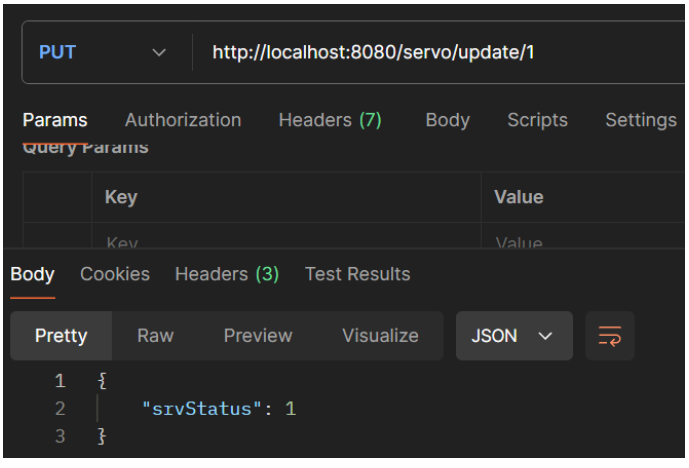
```
PS D:\mcs_bab_7> go run "d:\mcs_bab_7\main.go"
Migration success applied 0 &{{0xd42a40} sql_migrations}
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] POST   /servo/init-proj          --> mcs_bab_7/controllers.InitProj (3 handlers)
[GIN-debug] GET    /servo/status             --> mcs_bab_7/controllers.GetStatus (3 handlers)
[GIN-debug] PUT    /servo/update/:srv_status --> mcs_bab_7/controllers.UpdateStatus (3 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :8080
```

Run back end

Buka aplikasi Postman dan jalankan url sesuai rute yang sudah diuat

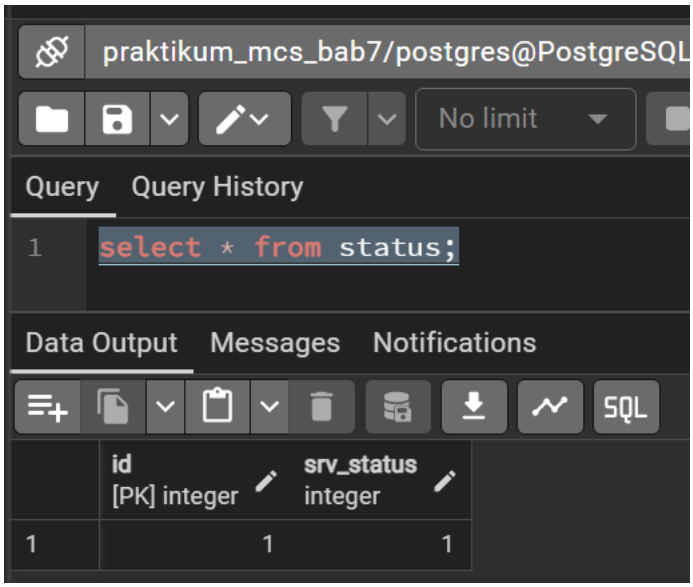


Metode get



Metode put

Cek di aplikasi pgAdmin 4 untuk melihat hasilnya



Cek di pgadmin 4

LAPORAN PENDAHULUAN (LP)

1, Berikan penjelasan apa itu primary key!

2. Berikan penjelasan apa itu Get Post, Put dan Delete!

3. Berikan contoh query update pada sql!

4. Sebutkan macam-macam http status code mulai dari 100, 200 300 dan 400 beserta penjelasannya!

LAPORAN AKHIR (LA)

1. Berikan kesimpulan pada praktikum bab 7!