MODUL XX
MIGRATION & GORM

Pertemuan : 18
Waktu : 8 x 60 Menit (Online)

1.1 Tujuan Modul XX

Setelah mahasiswa mempelajari materi ini, diharapkan dapat :

1. Memahami penggunaan migration di golang
2. Memahami penggunaan package gorm di golang

1.2 Landasan Teori

1.2.1 Migration

Migrasi seperti kontrol versi untuk database Anda, memungkinkan tim Anda untuk menentukan dan berbagi definisi skema database aplikasi. Jika Anda pernah harus memberi tahu rekan satu tim untuk secara manual menambahkan kolom ke skema database lokal mereka setelah menarik perubahan Anda dari kontrol sumber, Anda menghadapi masalah yang dipecahkan oleh migrasi database.

Schema Gorm Golang menyediakan dukungan agnostik database untuk membuat dan memanipulasi tabel di semua sistem database yang didukung Laravel. Biasanya, migrasi akan menggunakan fasad ini untuk membuat dan memodifikasi tabel dan kolom database.

1.2.1.1 Membuat migration

1. Install package gorm nya dengan command `go get -u gorm.io/gorm`
2. Create folder dengan command `mkdir script/migration`
3. Create folder lagi dengan command `mkdir entity`
4. Create file di dalam folder entity dengan nama `article_entity.go`
5. Jika sudah buatlah fungsi di file `article_entity.go` dengan fungsi sebagai berikut :

```go
package entity

import (
    "html"
    "strings"

    "github.com/gofrs/uuid"
)

const (
    ArticleTableName = "article"
)

// ArticleModel is a model for entity.Article
type Article struct {
    ID       uuid.UUID `gorm:"type:uuid;primary_key" json:"id"`
    Title    string    `gorm:"type:varchar(200);not_null" json:"title"`
    Content  string    `gorm:"type:text;null" json:"content"`
    Slug     string    `gorm:"type:varchar(200);null" json:"slug"`
    AuthorID uuid.UUID `gorm:"type:uuid;not_null" json:"author_id"`
    Auditable
}

func NewArticle(id uuid.UUID, title, content, slug string, authorID uuid.UUID) *Article {
    return &Article{
        ID:        id,
        Title:     title,
        Content:   content,
        Slug:      slug,
        AuthorID:  authorID,
        Auditable: NewAuditable(),
    }
}
```

6. Create file di dalam folder `script/migration` dengan nama `create_migration_script.go`
7. Jika sudah buatlah fungsi di file `create_migration_script.go` dengan fungsi sebagai berikut :

```go
package main

import (
    "fmt"
    "os"
    "sort"

    "github.com/rs/zerolog"
    "github.com/rs/zerolog/log"
    "gorm.io/driver/postgres"
    "gorm.io/gorm"
```

```
    "codelabs-service/entity"
    "codelabs-service/internal/config"
)

var db *gorm.DB

func main() {
    log.Logger = log.Output(zerolog.ConsoleWriter{Out:
os.Stderr})
    config, err := config.NewConfig(".env")
    checkError(err)

    dsn := fmt.Sprintf("host=%s port=%s user=%s password=%s
dbname=%s sslmode=disable",
        config.Database.Host,
        config.Database.Port,
        config.Database.Username,
        config.Database.Password,
        config.Database.Name)
    db, err = gorm.Open(postgres.Open(dsn),
&gorm.Config{SkipDefaultTransaction: true})
    checkError(err)

    sqlDB, err := db.DB()
    defer sqlDB.Close()

    executePendingMigrations(db)

    // Migrate rest of the models
    log.Info().Msg("AutoMigrate Model [table_name]")

    log.Info().Msg("  ArticleModel [" +
(&entity.Article{}).TableName() + "]")
    db.AutoMigrate(&entity.Article{})

}

func executePendingMigrations(db *gorm.DB) {
    db.AutoMigrate(&MigrationHistoryModel{})
    lastMigration := MigrationHistoryModel{}
```

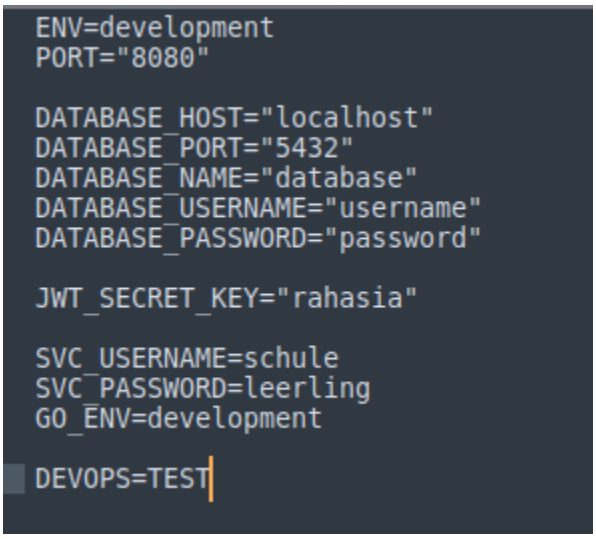
```
    skipMigration := db.Order("migration_id
desc").Limit(1).Find(&lastMigration).RowsAffected > 0

    // skip to last migration
    keys := make([]string, 0, len(migrations))
    for k := range migrations {
        keys = append(keys, k)
    }
    sort.Strings(keys)

    // run all migrations in one transaction
    if len(migrations) == 0 {
        log.Info().Msg("No pending migrations")
    } else {
        db.Transaction(func(tx *gorm.DB) error {
            for _, k := range keys {
                if skipMigration {
                    if k == lastMigration.MigrationID {
                        skipMigration = false
                    }
                } else {
                    log.Info().Msg("  " + k)
                    tx.Transaction(func(subTx *gorm.DB) error {
                        // run migration update
                        checkError(migrations[k](subTx))
                        // insert migration id into history

checkError(subTx.Create(MigrationHistoryModel{MigrationID:
k}).Error)
                        return nil
                    })
                }
            }
            return nil
        })
    }
}

type mFunc func(tx *gorm.DB) error
```

```go
var migrations = make(map[string]mFunc)

// MigrationHistoryModel model migration
type MigrationHistoryModel struct {
    MigrationID string `gorm:"type:text;primaryKey"`
}

// TableName name of migration table
func (model *MigrationHistoryModel) TableName() string {
    return "migration_history"
}

func checkError(err error) {
    if err != nil {
        log.Fatal().Err(err)
        panic(err)
    }
}

func registerMigration(id string, fm mFunc) {
    migrations[id] = fm
}
```

8. Create folder lagi dengan command `mkdir internal/config`
9. Create file di dalam folder `internal/config` dengan nama `config.go`
10. Jika sudah buatlah fungsi di file `article_entity.go` dengan fungsi sebagai berikut :

```go
package config

import (
    "log"

    "github.com/joeshaw/envdecode"
    "github.com/joho/godotenv"
    "github.com/pkg/errors"
)

// Config holds configuration for the project.
type Config struct {
    Port           string `env:"PORT,default=6666"`
    Env            string `env:"ENV,default=development"`
    Database       DatabaseConfig
    JWTConfig      JWTConfig
    InternalConfig InternalConfig
}


// DatabaseConfig holds configuration for database.
type DatabaseConfig struct {
    Host     string `env:"DATABASE_HOST,default=localhost"`
    Port     string `env:"DATABASE_PORT,default=5432"`
    Username string `env:"DATABASE_USERNAME,required"`
    Password string `env:"DATABASE_PASSWORD,required"`
    Name     string `env:"DATABASE_NAME,required"`
}

// JWTConfig holds configuration for JWT secret key
type JWTConfig struct {
    SecretKey string `env:"JWT_SECRET_KEY,required"`
}

// InternalConfig holds configuration for internal communication between microservices.
type InternalConfig struct {
    Username string `env:"SVC_USERNAME,required"`
    Password string `env:"SVC_PASSWORD,required"`
}

// NewConfig creates an instance of Config.
// It needs the path of the env file to be used.
func NewConfig(env string) (*Config, error) {
    var config Config
    if err := godotenv.Load(env); err != nil {
        log.Println(errors.Wrap(err, "[NewConfig] error reading .env file, defaulting to OS environment variables"))
    }

    if err := envdecode.Decode(&config); err != nil {
        return nil, errors.Wrap(err, "[NewConfig] error decoding env")
    }

    return &config, nil
}
```

11. Isi file .env dengan text sebagai berikut :



12. Isi dari .env seperti ini

    POSTGRES_URL="dbname=golang_crud user=postgres password=1234 host=localhost sslmode=disable"

1.3 Praktikum

1.3.1 Latihan Praktikum

Silahkan untuk mengembangkan migration diatas untuk dapat terkoneksi dengan aplikasi kakak - kakak mahasiswa, gunakan framework echo dan database postgres. Untuk referensi bisa ke gitlab ini https://git.gits.id/msib/msib-go.

1.3.2 Tugas Praktikum

untuk tugas buat migration tema bebas. Boleh buku, barang atau apapun dibebaskan. Untuk tugas praktikum ini dikumpulkan yaa.minimal 5 tabel saja