Written by Ade Putra Prima Suhendri

What is Go?
Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.

Go was designed at Google in 2007 to improve programming productivity in an era of multicore, networked machines and large codebases. The designers wanted to address criticism of other languages in use at Google, but keep their useful characteristics.

How to use Go?
Download and install Go at https://golang.org/doc/install

Check your Go version

$ go version

What we build?
We will build 'POST', 'GET', 'PUT', 'DELETE' Http Requests for simple articles RESTful API.

Method	**URL**	**Description**
POST	http://localhost:10000/article	Create new article
GET	http://localhost:10000/article	List all article
GET	http://localhost:10000/article/{id}	Single article
PUT	http://localhost:10000/article/{id}	Update article
DELETE	http://localhost:10000/article/{id}	Delete article

Go app directory structure will looks like:

go/
├── bin/
├── pkg/
├── src/
└── **main.go**

main.go

```
package main

import (
    "database/sql"
    "encoding/json"
    "fmt"
    "io/ioutil"
    "log"
    "net/http"

    _ "github.com/go-sql-driver/mysql"
    "github.com/gorilla/mux"
)

// init Payload structure Model for response
type Payload struct {
    Status  bool      `json:"status"`
    Message string    `json:"message"`
    Data    []Article `json:"data"`
}

// init Article structure Model
type Article struct {
    ID      string `json:"id"`
    Title   string `json:"title"`
    Content string `json:"content"`
}

// Database Connection function
func connect() (*sql.DB, error) {
    fmt.Println("MySQL Connection")
    db, err := sql.Open("mysql", "user:password@tcp(host:port)/database")
    if err != nil {
        panic(err.Error())
    }
    return db, nil
}
```

```go
// List Article Controller
func listArticle(w http.ResponseWriter, r *http.Request) {
    // set header response
    w.Header().Set("Content-Type", "application/json")

    // call Database Connection
    db, err := connect()
    if err != nil {
        fmt.Println(err.Error())
        return
    }
    defer db.Close()

    // init Article Model
    var articles []Article

    // set Query
    query, err := db.Query("SELECT * FROM article")
    if err != nil {
        panic(err.Error())
    }
    defer query.Close()

    // Loop Query
    for query.Next() {
        var article Article
        err := query.Scan(&article.ID, &article.Title, &article.Content)
        if err != nil {
            panic(err.Error())
        }
        articles = append(articles, article)
    }

    // set Payload
    var payload = Payload{Status: true, Message: "list article", Data: articles}

    // Encode Payload to json
    json.NewEncoder(w).Encode(payload)
}
```

```go
// Detail Article Controller
func detailArticle(w http.ResponseWriter, r *http.Request) {
    // set header response
    w.Header().Set("Content-Type", "application/json")

    // get parameter {id}
    vars := mux.Vars(r)
    id := vars["id"]

    // call Database Connection
    db, err := connect()
    if err != nil {
        fmt.Println(err.Error())
        return
    }
    defer db.Close()

    // init Article & Payload
    var articles []Article
    var payload Payload

    // set Query
    query, err := db.Query("SELECT * FROM article WHERE id = ?", id)
    if err != nil {
        panic(err.Error())
    }
    defer query.Close()

    // Loop Query
    for query.Next() {
        var article Article
        err := query.Scan(&article.ID, &article.Title, &article.Content)
        if err != nil {
            panic(err.Error())
        }
        articles = append(articles, article)
    }
```

```
        // check articles
        if articles == nil {
            payload = Payload{Status: false, Message: "detail article", Data:
articles}
        } else {
            payload = Payload{Status: true, Message: "detail article", Data:
articles}
        }

        // Encode Payload to json
        json.NewEncoder(w).Encode(payload)
    }

    // Create Article Controller
    func createArticle(w http.ResponseWriter, r *http.Request) {
        // set header response
        w.Header().Set("Content-Type", "application/json")

        // call Database Connection
        db, err := connect()
        if err != nil {
            fmt.Println(err.Error())
            return
        }
        defer db.Close()

        // set Payload
        var payload Payload

        // set Query
        query, err := db.Prepare("INSERT INTO article (title, content) VALUES (?, ?
)")
        if err != nil {
            panic(err.Error())
        }
```

```
    // get all Request from Body
    reqBody, err := ioutil.ReadAll(r.Body)
    if err != nil {
        panic(err.Error())
    }
    keyVal := make(map[string]string)
    json.Unmarshal(reqBody, &keyVal)
    title := keyVal["title"]
    content := keyVal["content"]

    // run Query
    _, err = query.Exec(title, content)
    if err != nil {
        panic(err.Error())
    }

    // set payload response
    payload = Payload{Status: true, Message: "create success", Data: nil}

    // Encode Payload to json
    json.NewEncoder(w).Encode(payload)
}

// Update Article Controller
func updateArticle(w http.ResponseWriter, r *http.Request) {
    // set header response
    w.Header().Set("Content-Type", "application/json")

    // get parameter {id}
    vars := mux.Vars(r)
    id := vars["id"]

    // get all Request from Body
    reqBody, err := ioutil.ReadAll(r.Body)
    if err != nil {
        panic(err.Error())
    }
    keyVal := make(map[string]string)
    json.Unmarshal(reqBody, &keyVal)
    title := keyVal["title"]
    content := keyVal["content"]
```

```
        // call Database Connection
        db, err := connect()
        if err != nil {
            fmt.Println(err.Error())
            return
        }
        defer db.Close()

        // set Payload
        var payload Payload

        // set Query
        query, err := db.Prepare("UPDATE article SET title = ?, content = ? WHERE id
= ?")
        if err != nil {
            panic(err.Error())
        }

        // run Query
        _, err = query.Exec(title, content, id)
        if err != nil {
            panic(err.Error())
        }

        // set payload response
        payload = Payload{Status: true, Message: "update success", Data: nil}

        // Encode Payload to json
        json.NewEncoder(w).Encode(payload)
    }

    // Delete Article Controller
    func deleteArticle(w http.ResponseWriter, r *http.Request) {
        // set header response
        w.Header().Set("Content-Type", "application/json")

        // get parameter {id}
        vars := mux.Vars(r)
        id := vars["id"]
```

```
    // call Database Connection
    db, err := connect()
    if err != nil {
        fmt.Println(err.Error())
        return
    }
    defer db.Close()

    // init Payload
    var payload Payload

    // set Query
    query, err := db.Query("DELETE FROM article WHERE id = ?", id)
    if err != nil {
        panic(err.Error())
    }
    defer query.Close()

    // set payload response
    payload = Payload{Status: true, Message: "delete success", Data: nil}

    // Encode Payload to json
    json.NewEncoder(w).Encode(payload)
}

// Routing
func handleRequests() {
    route := mux.NewRouter().StrictSlash(true)
    route.HandleFunc("/article", createArticle).Methods("POST")
    route.HandleFunc("/article", listArticle).Methods("GET")
    route.HandleFunc("/article/{id}", detailArticle).Methods("GET")
    route.HandleFunc("/article/{id}", updateArticle).Methods("PUT")
    route.HandleFunc("/article/{id}", deleteArticle).Methods("DELETE")
    log.Fatal(http.ListenAndServe(":10000", route))
}

// main Application
func main() {
    // set routing
    handleRequests()
}
```