**gin-gonic:-**

**Installation :-** <https://github.com/gin-gonic/gin>

**Learning** :- <https://www.youtube.com/watch?v=qR0WnWL2o1Q&list=PL3eAkoh7fypr8zrkiygiY1e9osoqjoV9w&ab_channel=PragmaticReviews>

<https://semaphoreci.com/community/tutorials/building-go-web-applications-and-microservices-using-gin>

**Context in gin-gonic:-**

**Learning**:-

<https://pkg.go.dev/context#WithCancel>

<https://tutorialedge.net/golang/go-context-tutorial/>

**auth golang:-**

<https://levelup.gitconnected.com/building-micro-services-in-go-using-keycloak-for-authorisation-e00a29b80a43>

Code that I try:

package controller

import (

"fmt"

//"log"

"example.com/m/entity"

"example.com/m/service"

"github.com/gin-gonic/gin"

)

type VideoController interface {

FindAll() []entity.Video

Save(ctx \*gin.Context) entity.Video

}

type controller struct {

service service.VideoService

}

func New(service service.VideoService) VideoController {

return &controller{

service: service,

}

}

func (c \*controller) FindAll() []entity.Video {

//return c.service.FindAll()

return service.New().FindAll()

}

func (c \*controller) Save(ctx \*gin.Context) entity.Video {

var video entity.Video

ctx.BindJSON(&video)

fmt.Println(video)

//c.service.Save(video)

service.New().Save(video)

return video

}

package entity

type Video struct {

Title string `json: "title"`

Description string `json: "description"`

URL string `json: "url"`

}

package service

import "example.com/m/entity"

type VideoService interface {

Save(entity.Video) entity.Video

FindAll() []entity.Video

}

type videoService struct {

videos []entity.Video

}

func New() VideoService {

return &videoService{}

// videos: make([]entity.Video, 1),

}

func (service videoService) Save(video entity.Video) entity.Video {

service.videos = append(service.videos, video)

return video

}

func (service videoService) FindAll() []entity.Video {

return service.videos

}

package main

import (

"log"

"example.com/m/controller"

"example.com/m/service"

"github.com/gin-gonic/gin"

)

var (

videoservice service.VideoService = service.New()

control controller.VideoController = controller.New(videoservice)

)

func main() {

r := gin.Default()

r.GET("/videos", func(ctx \*gin.Context) {

ctx.JSON(200, control.FindAll())

})

r.POST("/videos", func(ctx \*gin.Context) {

ctx.JSON(200, control.Save(ctx))

})

log.Println("Hello")

r.Run()

}

Day-2

Keycloak APi:- <https://github.com/Nerzal/gocloak>

JWT :- <https://github.com/golang-jwt/jwt>

//server.GET("/", getHandler)

// server.GET("/", func(ctx \*gin.Context) {

// ctx.JSON(http.StatusAccepted, getHandler)

// })

package main

import (

"log"

"github.com/gin-gonic/gin"

"github.com/google/uuid"

)

type user struct {

ID string `json:"id"`

Name string `json:"name"`

Age int `json:"age"`

}

var Users []user //return nil

func main() {

server := gin.Default()

userRoutes := server.Group("/users")

{

userRoutes.GET("/", getHandler) //Read endpoint

userRoutes.POST("/", PostHandler) //Create endpoint

userRoutes.PUT("/:id", PutHandler) //Update endpoint

userRoutes.DELETE("/:id", DelId) //Delete endpoint

}

if err := server.Run(); err != nil {

log.Fatal(err.Error())

}

}

func getHandler(ctx \*gin.Context) {

//return null

ctx.JSON(200, Users)

}

func PostHandler(ctx \*gin.Context) {

var req user

if err := ctx.ShouldBind(&req); err != nil {

ctx.JSON(422, gin.H{

"Error": true,

"message": "invalid body",

})

return

}

req.ID = uuid.New().String()

Users = append(Users, req)

ctx.JSON(200, gin.H{

"error": false,

})

}

func PutHandler(ctx \*gin.Context) {

id := ctx.Param("id")

var req user

if err := ctx.ShouldBind(&req); err != nil {

ctx.JSON(422, gin.H{

"Error": true,

"message": "invalid body",

})

return

}

for i, u := range Users {

if u.ID == id {

Users[i].Name = req.Name

Users[i].Age = req.Age

ctx.JSON(200, gin.H{

"error": false,

})

return

}

ctx.JSON(404, gin.H{

"error": true,

"message": "invalid user id",

})

}

}

func DelId(ctx \*gin.Context) {

id := ctx.Param("id")

for i, u := range Users {

if u.ID == id {

Users = append(Users[:i], Users[i+1:]...)

ctx.JSON(200, gin.H{

"error": false,

})

return

}

}

ctx.JSON(404, gin.H{

"error": true,

"message": "invalid user id",

})

}

I use this link to know about the unit testing in golang <https://www.youtube.com/watch?v=uB_45bSIyik&ab_channel=PragmaticReviews>

<https://www.youtube.com/watch?v=hVFEV-ieeew&ab_channel=justforfunc%3AProgramminginGo>

For mocking:-

<https://www.myhatchpad.com/insight/mocking-techniques-for-go/>

package main

//return the sum of list of integer

func Ints(vs ...int) int {

return ints(vs)

}

func ints(vs []int) int {

if len(vs) == 0 {

return 0

}

return ints(vs[1:]) + vs[0]

}

//you can test the func without including the package by adding main.fun(...interface{})

package main

import (

"testing"

)

// better way to do is using sub test

//it help to run some specific test in for loop

func TestInt(t \*testing.T) {

tt := []struct {

name string

numbers []int

sum int

}{

{"TestSum :one to four", []int{1, 2, 3, 4}, 10},

{"TestSum :one to five", []int{1, 2, 3, 4, 5}, 15},

{"TestSum :of empty array", []int{}, 0},

}

println(tt)

s := Ints(1, 2, 3, 4, 5)

if s != 15 {

t.Errorf("This sum is failed as sum %v is not equal 15 which is expected sum", s)

}

for \_, r := range tt {

t.Run(r.name, func(t \*testing.T) {

if r.sum != Ints(r.numbers...) {

t.Fatalf("%s,test case %v Not a suitable/expected value %v but sum: %v", r.name, r.numbers, r.sum, Ints(r.numbers...))

}

})

}

}

// func foo(){

// //suppose we have thousand of testcases but we have test specific cases then

// // go test-run foo -v

// // go test-run . -v //for all

// }

// func TestInt(t \*testing.T) {

// //t.Errorf("This is fail")// this is fail but execution will continue

// //t.Fatalf("Not worry") //this is fail and stop exec

// //here we are try to add the name of each test cases for better documentation

// tt := []struct {

// name string

// numbers []int

// sum int

// }{

// {"TestSum :one to four", []int{1, 2, 3, 4}, 10},

// {"TestSum :one to five", []int{1, 2, 3, 4, 5}, 1},

// {"TestSum :of empty array", []int{}, 0},

// }

// println(tt)

// s := Ints(1, 2, 3, 4, 5)

// if s != 15 {

// t.Errorf("This sum is failed as sum %v is not equal 15 which is expected sum", s)

// }

// for \_, r := range tt {

// //println(r)

// if r.sum != Ints(r.numbers...) {

// t.Errorf("%s,test case %v Not a suitable/expected value %v but sum: %v", r.name, r.numbers, r.sum, Ints(r.numbers...))

// }

// }

//here we are try to print the multiple error with the help of for loop

// tt := []struct {

// numbers []int

// sum int

// }{

// {[]int{1, 2, 3, 4}, 10},

// {[]int{1, 2, 3, 4, 5}, 1},

// {[]int{}, 0},

// }

// println(tt)

// s := Ints(1, 2, 3, 4, 5)

// if s != 15 {

// t.Errorf("This sum is failed as sum %v is not equal 15 which is expected sum", s)

// }

// for \_, r := range tt {

// //println(r)

// if r.sum != Ints(r.numbers...) {

// t.Errorf("%v Not a suitable/expected value %v but sum: %v",r.numbers,r.sum,Ints(r.numbers...))

// }

// //println(r.numbers,"NIkhil",r.sum)

// }

// s = Ints(1, -1)

// if s != 15 {

// t.Errorf("This sum is failed as sum %v is not equal 15 which is expected sum", s)

// }

// s = Ints()

// if s != 15 {

// t.Errorf("This sum is failed as sum %v is not equal 15 which is expected sum", s)

// }

// s = Ints(1, 2, 3, 4, 5)

// if s != 15 {

// t.Errorf("This sum is failed as sum %v is not equal 15 which is expected sum", s)

// }

// s = Ints(1, 2, 3, 4, 5, 6)

// if s != 15 {

// t.Errorf("This sum is failed as sum %v is not equal 15 which is expected sum", s)

// }

// }

Mocking:

<https://github.com/golang/mock#go-version--116>

<https://www.youtube.com/watch?v=LEnXBueFBzk&ab_channel=hatchpad>

<https://www.youtube.com/watch?v=hVFEV-ieeew&ab_channel=justforfunc%3AProgramminginGo>

package main

import (

"testing"

"github.com/stretchr/testify/assert"

)

func TestAdd(t \*testing.T) {

total := AddNumber(1, 2)

assert.NotNil(t, total, "Total must not be nil")

assert.Equal(t, 3, total, "expecting 3")

}

func TestSub(t \*testing.T) {

total := Sub(1, 2)

assert.NotNil(t, total, "Total must not be nil")

assert.Equal(t, -2, total, "expecting -1")

}

package main

import (

"net/http"

"net/http/httptest"

"testing"

"github.com/gin-gonic/gin"

)

type UserRepositoryMock struct{}

func (r UserRepositoryMock) GetAll() Users {

users := Users{

{Name: "Wilson"},

{Name: "Panda"},

}

return users

}

func (r UserRepositoryMock) Get(id int) User {

users := Users{

{Name: "Wilson"},

{Name: "Panda"},

}

return users[id-1]

}

// TESTING REPOSITORY FUNCTIONS

func TestRepoGetAll(t \*testing.T) {

userRepo := UserRepository{}

amountUsers := len(userRepo.GetAll())

if amountUsers != 2 {

t.Errorf("Esperado %d, recebido %d", 2, amountUsers)

}

}

func TestRepoGet(t \*testing.T) {

expectedUser := struct {

Name string

}{

"Wilson",

}

userRepo := UserRepository{}

user := userRepo.Get(1)

if user.Name != expectedUser.Name {

t.Errorf("Esperado %s, recebido %s", expectedUser.Name, user.Name)

}

}

func TestControllerGetAll(t \*testing.T) {

// Switch to test mode so you don't get such noisy output

gin.SetMode(gin.TestMode)

// Setup your router, just like you did in your main function, and

// register your routes

r := gin.Default()

r.GET("/users", GetUsers)

// Create the mock request you'd like to test. Make sure the second argument

// here is the same as one of the routes you defined in the router setup

// block!

req, err := http.NewRequest(http.MethodGet, "/users", nil)

if err != nil {

t.Fatalf("Couldn't create request: %v\n", err)

}

// Create a response recorder so you can inspect the response

w := httptest.NewRecorder()

// Perform the request

r.ServeHTTP(w, req)

// Check to see if the response was what you expected

if w.Code != http.StatusOK {

t.Fatalf("Expected to get status %d but instead got %d\n", http.StatusOK, w.Code)

}

}

// func TestControllerGetAll(t \*testing.T) {

// gin.SetMode(gin.TestMode)

// w := httptest.NewRecorder()

// c, \_ := gin.CreateTestContext(w)

// c.Params = []gin.Param{{

// Key: "k", Value: "v",

// }}

// GetUser(c)

// if w.Code != 200 {

// b, \_ := ioutil.ReadAll(w.Body)

// t.Error(w.Code, string(b))

// }

// }

/\* HOW TO TEST CONTROLLER?

func TestControllerGetAll(t \*testing.T) {

gin.SetMode(gin.TestMode)

c := &gin.Context{}

c.Status(200)

repo := UserRepositoryMock{}

ctrl := UserController{}

ctrl.GetAll(c, repo)

} \*/