蓝山后端第六次作业

```go
package user

import "github.com/dgrijalva/jwt-go"

type MyClaims struct {
    Username string
    Pwd      string
    jwt.StandardClaims
}

```

```go
package main

import "gin_new/bluework/api"

func main() {
    api.InitRouter()
}
```

定义的一些函数

```go
package funcpackage

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

var datebase map[string]string

func Adduse(username, pwd string) {
    datebase = make(map[string]string)
    if datebase[username] != "" {
        fmt.Println("用户已经存在")
    } else {
        datebase[username] = pwd
        Adduserinformation(username, pwd)
    }
}
func Selectuser(username string) bool {
    datebase = make(map[string]string)
    Readuserinformation()
    if datebase[username] == "" {
        fmt.Println("用户不存在")
        return false
    } else {
        return true
    }
}
func Checkpwd(username string) string {
    datebase = make(map[string]string)
    Readuserinformation()
    return datebase[username]
}

// 添加用户信息到文件
func Adduserinformation(username, pwd string) {
    file := "d:/user.txt"
    addfile, _ := os.OpenFile(file, os.O_WRONLY|os.O_CREATE|os.O_APPEND, 0666)
    write := bufio.NewWriter(addfile)
    defer addfile.Close()
    //如果不输入就请求post,则直接退出
    if username == "" || pwd == "" {
        return
    }

    str1 := username + " "
    str2 := pwd + " "
    write.WriteString(str1)
    write.WriteString(str2)
    write.Flush()
}

// 读用户文件里面的内容并添加到"数据库"
func Readuserinformation() {
    datebase = make(map[string]string)
    file := "d:/user.txt"
    readfile, _ := os.ReadFile(file)
    a := strings.Split(string(readfile), " ")
    for i := 0; i < len(a)-1; i++ {
        datebase[a[i]] = a[i+1]
    }

}

// 修改密码
func Changepwd(username, originalpwd, repwd string) bool {
    datebase = make(map[string]string)
    Readuserinformation()
    if datebase[username] != originalpwd {
        return false
    } else {
        Readuserinformation()
        datebase[username] = repwd
        return true
    }

}
```

```go
package middleware
```

```go
import (
    "errors"
    "gin_new/bluework/user"
    "github.com/dgrijalva/jwt-go"
    "github.com/gin-gonic/gin"
    "net/http"
    "strings"
)

var Secret = []byte("李灏宇")

func JWTAuthMiddleware() func(context *gin.Context) {
    return func(context *gin.Context) {
        authHeader := context.Request.Header.Get("Authorization")
        if authHeader == "" {
            context.JSON(http.StatusOK, gin.H{
                "code": 2003,
                "msg":  "请求头中auth为空",
            })
            context.Abort()
            return
        }

        parts := strings.SplitN(authHeader, " ", 2)
        if !(len(parts) == 2 && parts[0] == "Bearer") {
            context.JSON(http.StatusOK, gin.H{
                "code": 2004,
                "msg":  "请求头中auth格式有误",
            })
            context.Abort()
            return
        }

        mc, err := ParseToken(parts[1])
        if err != nil {
            context.JSON(http.StatusOK, gin.H{
                "code": 2005,
                "msg":  "无效的Token",
            })
            context.Abort()
            return
        }

        context.Set("username", mc.Username)
        context.Next()
    }
}

// ParseToken 解析JWT
func ParseToken(tokenString string) (*user.MyClaims, error) {
    // 解析token
    token, err := jwt.ParseWithClaims(tokenString, &user.MyClaims{}, func(token
*jwt.Token) (i interface{}, err error) {
        return Secret, nil
    })
    if err != nil {
        return nil, err
    }
    if claims, ok := token.Claims.(*user.MyClaims); ok && token.Valid { // 校验
token
        return claims, nil
    }
    return nil, errors.New("invalid token")
}
func CORS() gin.HandlerFunc {
    return func(ctx *gin.Context) {
        ctx.Writer.Header().Set("Access-Control-Allow-Origin", "*")
        ctx.Writer.Header().Set("Access-Control-Allow-Credentials", "true")
        ctx.Writer.Header().Set("Access-Control-Allow-Headers", "Content-Type,
Content-Length, Accept-Encoding, X-CSRF-Token, Authorization, accept, origin,
Cache-Control, X-Requested-With, token, x-access-token")
        ctx.Writer.Header().Set("Access-Control-Allow-Methods", "POST, OPTIONS,
GET, PUT, DELETE")

        if ctx.Request.Method == "OPTIONS" {
            ctx.AbortWithStatus(204)
            return
        }
        ctx.Next()
    }
}
```

```
package api

import (
    "gin_new/bluework/funcpackage"
    "gin_new/bluework/middleware"
    "gin_new/bluework/user"
    "gin_new/bluework/utils"
    "github.com/dgrijalva/jwt-go"
    "github.com/gin-gonic/gin"
    "time"
)

func Register(context *gin.Context) {

    Username := context.PostForm("Username")
    Pwd := context.PostForm("Pwd")
    flag := funcpackage.Selectuser(Username)
    if flag {
        context.JSON(500, gin.H{
            "state":   500,
            "message": "user has exists",
        })
        return
    }
    funcpackage.Adduse(Username, Pwd)
    context.JSON(200, gin.H{
        "state":   200,
```

```go
        "message": "adduser successful",
    })

}
func login(context *gin.Context) {
    if err := context.ShouldBind(&user.MyClaims{}); err != nil {

        utils.RespFail(context, "verification failed")
        return
    }
    Username := context.PostForm("Username")
    Pwd := context.PostForm("Pwd")
    flag := funcpackage.Selectuser(Username)
    if flag == false {
        utils.RespFail(context, "user doesn't exists")
        return
    }
    rightpwd := funcpackage.Checkpwd(Username)
    if rightpwd != Pwd {
        utils.RespFail(context, "worry pwd")
        return
    }
    claim := user.MyClaims{
        Username: Username,
        Pwd:      Pwd,
        StandardClaims: jwt.StandardClaims{
            ExpiresAt: time.Now().Add(time.Hour * 2).Unix(),
            Issuer:    "李灏宇",
        },
    }

    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claim)
    tokenString, _ := token.SignedString(middleware.Secret)
    utils.RespSuccess(context, tokenString)

}
func getUsernameFromToken(c *gin.Context) {
    username, _ := c.Get("username")
    utils.RespSuccess(c, username.(string))
}
func Editpwd(context *gin.Context) {
    Username := context.PostForm("Username")
    Pwd := context.PostForm("Pwd")
    //想要修改的密码
    Repwd := context.PostForm("Repwd")
    flag := funcpackage.Changepwd(Username, Pwd, Repwd)
    if !flag {
        utils.RespFail(context, "Incorrect password")
        return
    }
    utils.RespSuccess(context, "modify successfully")

}
```

```go
package api
```

```go
import (
	"gin_new/bluework/middleware"
	"github.com/gin-gonic/gin"
)

func InitRouter() {
	r := gin.Default()
	r.Use(middleware.CORS())

	r.POST("/register", Register)
	r.POST("/login", login)
	r.POST("/modify", Editpwd)
	UserRouter := r.Group("/user")
	{
		UserRouter.Use(middleware.JWTAuthMiddleware())
		UserRouter.GET("/get", getUsernameFromToken)
	}

	r.Run(":8080") // 跑在 8088 端口上
}
```

```go
package utils

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

func RespFail(c *gin.Context, message string) {

	c.JSON(400, gin.H{
		"status":  "fail",
		"message": message,
	})
}
func RespSuccess(c *gin.Context, tokenString string) {

	c.JSON(200, gin.H{
		"status": "success",
		"token":  tokenString,
	})
}
```