package cmd

import (

"UniApi/internal/controller/user"

"UniApi/internal/service"

"context"

\_ "github.com/gogf/gf/contrib/drivers/mysql/v2"

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/net/ghttp"

"github.com/gogf/gf/v2/os/gcmd"

)

var (

Main = gcmd.Command{

Name: "main",

Usage: "main",

Brief: "start http server",

Func: func(ctx context.Context, parser \*gcmd.Parser) (err error) {

s := g.Server()

s.Use(ghttp.MiddlewareHandlerResponse)

s.Group("/api", func(group \*ghttp.RouterGroup) {

group.Middleware(ghttp.MiddlewareHandlerResponse)

group.Bind(

user.NewV1(),

)

group.GET("/api", func(req \*ghttp.Request) {

req.Response.WriteTpl("/api.html")

})

})

s.BindHandler("/api/db", func(r \*ghttp.Request) {

users, err := service.User().GetList(r.Context())

if err == nil {

r.Response.WriteJson(users)

} else {

r.Response.Writeln("发生错误")

}

return

})

s.Run()

return nil

},

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcCommentDao = \*internal.NmcCommentDao

type nmcCommentDao struct {

internalNmcCommentDao

}

var (

NmcComment = nmcCommentDao{

internal.NewNmcCommentDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcPicsDao = \*internal.NmcPicsDao

type nmcPicsDao struct {

internalNmcPicsDao

}

var (

NmcPics = nmcPicsDao{

internal.NewNmcPicsDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcTopicDao = \*internal.NmcTopicDao

type nmcTopicDao struct {

internalNmcTopicDao

}

var (

NmcTopic = nmcTopicDao{

internal.NewNmcTopicDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcWorkTagDao = \*internal.NmcWorkTagDao

type nmcWorkTagDao struct {

internalNmcWorkTagDao

}

var (

NmcWorkTag = nmcWorkTagDao{

internal.NewNmcWorkTagDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcWorkDao = \*internal.NmcWorkDao

type nmcWorkDao struct {

internalNmcWorkDao

}

var (

NmcWork = nmcWorkDao{

internal.NewNmcWorkDao(),

}

)

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcCommentDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcCommentColumns // columns contains all the column names of Table for convenient usage.

}

type NmcCommentColumns struct {

CommentId string // 评论id

Content string // 评论

CreateTime string // 评论时间

IsDelete string // 是否删除

Uid string // 所属用户id

PostId string // 所属文章id

LikeCount string // 点赞次数

RootCommentId string // 顶级评论id

ToCommentId string // 目标评论id

UpdateTime string // 修改时间

}

var nmcCommentColumns = NmcCommentColumns{

CommentId: "commentId",

Content: "content",

CreateTime: "createTime",

IsDelete: "isDelete",

Uid: "uid",

PostId: "postId",

LikeCount: "likeCount",

RootCommentId: "rootCommentId",

ToCommentId: "toCommentId",

UpdateTime: "updateTime",

}

func NewNmcCommentDao() \*NmcCommentDao {

return &NmcCommentDao{

group: "default",

table: "comment",

columns: nmcCommentColumns,

}

}

func (dao \*NmcCommentDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcCommentDao) Table() string {

return dao.table

}

func (dao \*NmcCommentDao) Columns() NmcCommentColumns {

return dao.columns

}

func (dao \*NmcCommentDao) Group() string {

return dao.group

}

func (dao \*NmcCommentDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcCommentDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcPicsDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcPicsColumns // columns contains all the column names of Table for convenient usage.

}

type NmcPicsColumns struct {

Pid string // 图片自增id

Url string // 图片地址

UploadId string // 上传者

}

var nmcPicsColumns = NmcPicsColumns{

Pid: "pid",

Url: "url",

UploadId: "upload\_id",

}

func NewNmcPicsDao() \*NmcPicsDao {

return &NmcPicsDao{

group: "default",

table: "pics",

columns: nmcPicsColumns,

}

}

func (dao \*NmcPicsDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcPicsDao) Table() string {

return dao.table

}

func (dao \*NmcPicsDao) Columns() NmcPicsColumns {

return dao.columns

}

func (dao \*NmcPicsDao) Group() string {

return dao.group

}

func (dao \*NmcPicsDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcPicsDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcTopicDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcTopicColumns // columns contains all the column names of Table for convenient usage.

}

type NmcTopicColumns struct {

TopicId string // 话题id

TopicCreator string // uid

TopicName string // 话题名字

TopicDc string // 话题描述

TopicUrl string // 话题图片

CreatedTime string // 创建时间

UpdatedTime string // 更新时间

IsDelete string // 是否删除

}

var nmcTopicColumns = NmcTopicColumns{

TopicId: "topic\_id",

TopicCreator: "topic\_creator",

TopicName: "topic\_name",

TopicDc: "topic\_dc",

TopicUrl: "topic\_url",

CreatedTime: "created\_time",

UpdatedTime: "updated\_time",

IsDelete: "isDelete",

}

func NewNmcTopicDao() \*NmcTopicDao {

return &NmcTopicDao{

group: "default",

table: "topic",

columns: nmcTopicColumns,

}

}

func (dao \*NmcTopicDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcTopicDao) Table() string {

return dao.table

}

func (dao \*NmcTopicDao) Columns() NmcTopicColumns {

return dao.columns

}

func (dao \*NmcTopicDao) Group() string {

return dao.group

}

func (dao \*NmcTopicDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcTopicDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcWorkTagDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcWorkTagColumns // columns contains all the column names of Table for convenient usage.

}

type NmcWorkTagColumns struct {

WorkTagId string // 自增id

CreatorId string // 创建者uid

WorkName string // 名字

WorkDc string // 描述

StartTime string // 开始时间

EndTime string // 结束时间

ToRole string // 工作人员

UseId string // 使用人员

Status string // 当前状态0，进行中，1结束

}

var nmcWorkTagColumns = NmcWorkTagColumns{

WorkTagId: "work\_tag\_id",

CreatorId: "creator\_id",

WorkName: "work\_name",

WorkDc: "work\_dc",

StartTime: "start\_time",

EndTime: "end\_time",

ToRole: "to\_role",

UseId: "use\_id",

Status: "status",

}

func NewNmcWorkTagDao() \*NmcWorkTagDao {

return &NmcWorkTagDao{

group: "default",

table: "work\_tag",

columns: nmcWorkTagColumns,

}

}

func (dao \*NmcWorkTagDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcWorkTagDao) Table() string {

return dao.table

}

func (dao \*NmcWorkTagDao) Columns() NmcWorkTagColumns {

return dao.columns

}

func (dao \*NmcWorkTagDao) Group() string {

return dao.group

}

func (dao \*NmcWorkTagDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcWorkTagDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcWorkDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcWorkColumns // columns contains all the column names of Table for convenient usage.

}

type NmcWorkColumns struct {

WorkTagId string // 工作主题

WorkId string // 工作

FileUrl string // 文件地址

WorkStatus string // 0-正在，1-审核中, 2-修改中, 3-结束。

ApplyId string // 此工作的工作人员

WorkWatcher string // 初审，一般是部长

WorkWatcher2 string // 再审核

WorkWatcher3 string // 终审

Judge string // 修改意见

}

var nmcWorkColumns = NmcWorkColumns{

WorkTagId: "work\_tag\_id",

WorkId: "work\_Id",

FileUrl: "file\_url",

WorkStatus: "work\_status",

ApplyId: "apply\_id",

WorkWatcher: "work\_watcher",

WorkWatcher2: "work\_watcher2",

WorkWatcher3: "work\_watcher3",

Judge: "judge",

}

func NewNmcWorkDao() \*NmcWorkDao {

return &NmcWorkDao{

group: "default",

table: "work",

columns: nmcWorkColumns,

}

}

func (dao \*NmcWorkDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcWorkDao) Table() string {

return dao.table

}

func (dao \*NmcWorkDao) Columns() NmcWorkColumns {

return dao.columns

}

func (dao \*NmcWorkDao) Group() string {

return dao.group

}

func (dao \*NmcWorkDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcWorkDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcUserProfileDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcUserProfileColumns // columns contains all the column names of Table for convenient usage.

}

type NmcUserProfileColumns struct {

Id string //

Uid string // user表的id

Name string // 用户名字

Schoolid string // 学号

Spwd string // 教务密码

SchoolSession string // 教务登陆使用

Cnid string // 身份证号

}

var nmcUserProfileColumns = NmcUserProfileColumns{

Id: "id",

Uid: "uid",

Name: "name",

Schoolid: "schoolid",

Spwd: "spwd",

SchoolSession: "school\_session",

Cnid: "cnid",

}

func NewNmcUserProfileDao() \*NmcUserProfileDao {

return &NmcUserProfileDao{

group: "default",

table: "user\_profile",

columns: nmcUserProfileColumns,

}

}

func (dao \*NmcUserProfileDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcUserProfileDao) Table() string {

return dao.table

}

func (dao \*NmcUserProfileDao) Columns() NmcUserProfileColumns {

return dao.columns

}

func (dao \*NmcUserProfileDao) Group() string {

return dao.group

}

func (dao \*NmcUserProfileDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcUserProfileDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcBbuRoleDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcBbuRoleColumns // columns contains all the column names of Table for convenient usage.

}

type NmcBbuRoleColumns struct {

Id string // 自增id

Name string // 角色名

SchoolRole string // 角色id代码

}

var nmcBbuRoleColumns = NmcBbuRoleColumns{

Id: "id",

Name: "name",

SchoolRole: "school\_role",

}

func NewNmcBbuRoleDao() \*NmcBbuRoleDao {

return &NmcBbuRoleDao{

group: "default",

table: "bbu\_role",

columns: nmcBbuRoleColumns,

}

}

func (dao \*NmcBbuRoleDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcBbuRoleDao) Table() string {

return dao.table

}

func (dao \*NmcBbuRoleDao) Columns() NmcBbuRoleColumns {

return dao.columns

}

func (dao \*NmcBbuRoleDao) Group() string {

return dao.group

}

func (dao \*NmcBbuRoleDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcBbuRoleDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcSmsDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcSmsColumns // columns contains all the column names of Table for convenient usage.

}

type NmcSmsColumns struct {

Phone string // 号码验证

Code string // 验证码内容

Time string // 当前时间

}

var nmcSmsColumns = NmcSmsColumns{

Phone: "phone",

Code: "code",

Time: "time",

}

func NewNmcSmsDao() \*NmcSmsDao {

return &NmcSmsDao{

group: "default",

table: "sms",

columns: nmcSmsColumns,

}

}

func (dao \*NmcSmsDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcSmsDao) Table() string {

return dao.table

}

func (dao \*NmcSmsDao) Columns() NmcSmsColumns {

return dao.columns

}

func (dao \*NmcSmsDao) Group() string {

return dao.group

}

func (dao \*NmcSmsDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcSmsDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcNmcPhotoDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcNmcPhotoColumns // columns contains all the column names of Table for convenient usage.

}

type NmcNmcPhotoColumns struct {

Pid string // 图片自增id

Url string // 图片地址

UploadId string // 上传者

UpTime string // 上传时间

DownloadCount string // 下载数

IsVip string // 是否收费,1收费，0不收费，默认1

}

var nmcNmcPhotoColumns = NmcNmcPhotoColumns{

Pid: "pid",

Url: "url",

UploadId: "upload\_id",

UpTime: "up\_time",

DownloadCount: "download\_count",

IsVip: "is\_vip",

}

func NewNmcNmcPhotoDao() \*NmcNmcPhotoDao {

return &NmcNmcPhotoDao{

group: "default",

table: "nmc\_photo",

columns: nmcNmcPhotoColumns,

}

}

func (dao \*NmcNmcPhotoDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcNmcPhotoDao) Table() string {

return dao.table

}

func (dao \*NmcNmcPhotoDao) Columns() NmcNmcPhotoColumns {

return dao.columns

}

func (dao \*NmcNmcPhotoDao) Group() string {

return dao.group

}

func (dao \*NmcNmcPhotoDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcNmcPhotoDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcPostDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcPostColumns // columns contains all the column names of Table for convenient usage.

}

type NmcPostColumns struct {

PostId string // 文章id

Uid string // 文章发布者

Address string // 位置

TopicId string // 话题id

PostType string // 1无，2图片，3视频

Content string // 内容

PostLike string // 点赞数量

CreatedTime string //

IsDelete string // 是否删除0

}

var nmcPostColumns = NmcPostColumns{

PostId: "post\_id",

Uid: "uid",

Address: "address",

TopicId: "topic\_id",

PostType: "post\_type",

Content: "content",

PostLike: "post\_like",

CreatedTime: "created\_time",

IsDelete: "isDelete",

}

func NewNmcPostDao() \*NmcPostDao {

return &NmcPostDao{

group: "default",

table: "post",

columns: nmcPostColumns,

}

}

func (dao \*NmcPostDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcPostDao) Table() string {

return dao.table

}

func (dao \*NmcPostDao) Columns() NmcPostColumns {

return dao.columns

}

func (dao \*NmcPostDao) Group() string {

return dao.group

}

func (dao \*NmcPostDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcPostDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcUserDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcUserColumns // columns contains all the column names of Table for convenient usage.

}

type NmcUserColumns struct {

Uid string // 用户ID

Name string // 用户名

Password string // 密码

Phone string // 手机号码

Sign string // 个性签名

AvatarUrl string // 头像链接地址

Verti string // 认证类型

BgUrl string // 背景图链接地址

Status string // 用户封禁状态，0正常，1封禁

Major string // 专业

SchoolRole string // 学校代码5+组织3+组织内部门3+组织内角色1；aaaaa-bbb-ccc+d。默认普通角色除学校代码外全为0

Accid string // 网易云信id

AccToken string // 网易云信token

CreateTime string // 创建时间

UpdatedTime string // 修改时间

DeletedAt string //

Exp string // 等级

Pt string //

}

var nmcUserColumns = NmcUserColumns{

Uid: "uid",

Name: "name",

Password: "password",

Phone: "phone",

Sign: "sign",

AvatarUrl: "avatar\_url",

Verti: "verti",

BgUrl: "bg\_url",

Status: "status",

Major: "major",

SchoolRole: "school\_role",

Accid: "accid",

AccToken: "acc\_token",

CreateTime: "create\_time",

UpdatedTime: "updated\_time",

DeletedAt: "deleted\_at",

Exp: "exp",

Pt: "pt",

}

func NewNmcUserDao() \*NmcUserDao {

return &NmcUserDao{

group: "default",

table: "user",

columns: nmcUserColumns,

}

}

func (dao \*NmcUserDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcUserDao) Table() string {

return dao.table

}

func (dao \*NmcUserDao) Columns() NmcUserColumns {

return dao.columns

}

func (dao \*NmcUserDao) Group() string {

return dao.group

}

func (dao \*NmcUserDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcUserDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package internal

import (

"context"

"github.com/gogf/gf/v2/database/gdb"

"github.com/gogf/gf/v2/frame/g"

)

type NmcNmcDesignDao struct {

table string // table is the underlying table name of the DAO.

group string // group is the database configuration group name of current DAO.

columns NmcNmcDesignColumns // columns contains all the column names of Table for convenient usage.

}

type NmcNmcDesignColumns struct {

Did string // 设计稿自增id

Url string // 图片地址

UploadId string // 上传者

UpTime string // 上传时间

DownloadCount string // 下载数

IsVip string // 是否收费,1收费，0不收费，默认1

}

var nmcNmcDesignColumns = NmcNmcDesignColumns{

Did: "did",

Url: "url",

UploadId: "upload\_id",

UpTime: "up\_time",

DownloadCount: "download\_count",

IsVip: "is\_vip",

}

func NewNmcNmcDesignDao() \*NmcNmcDesignDao {

return &NmcNmcDesignDao{

group: "default",

table: "nmc\_design",

columns: nmcNmcDesignColumns,

}

}

func (dao \*NmcNmcDesignDao) DB() gdb.DB {

return g.DB(dao.group)

}

func (dao \*NmcNmcDesignDao) Table() string {

return dao.table

}

func (dao \*NmcNmcDesignDao) Columns() NmcNmcDesignColumns {

return dao.columns

}

func (dao \*NmcNmcDesignDao) Group() string {

return dao.group

}

func (dao \*NmcNmcDesignDao) Ctx(ctx context.Context) \*gdb.Model {

return dao.DB().Model(dao.table).Safe().Ctx(ctx)

}

func (dao \*NmcNmcDesignDao) Transaction(ctx context.Context, f func(ctx context.Context, tx gdb.TX) error) (err error) {

return dao.Ctx(ctx).Transaction(ctx, f)

}

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcUserProfileDao = \*internal.NmcUserProfileDao

type nmcUserProfileDao struct {

internalNmcUserProfileDao

}

var (

NmcUserProfile = nmcUserProfileDao{

internal.NewNmcUserProfileDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcBbuRoleDao = \*internal.NmcBbuRoleDao

type nmcBbuRoleDao struct {

internalNmcBbuRoleDao

}

var (

NmcBbuRole = nmcBbuRoleDao{

internal.NewNmcBbuRoleDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcSmsDao = \*internal.NmcSmsDao

type nmcSmsDao struct {

internalNmcSmsDao

}

var (

NmcSms = nmcSmsDao{

internal.NewNmcSmsDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcNmcPhotoDao = \*internal.NmcNmcPhotoDao

type nmcNmcPhotoDao struct {

internalNmcNmcPhotoDao

}

var (

NmcNmcPhoto = nmcNmcPhotoDao{

internal.NewNmcNmcPhotoDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcPostDao = \*internal.NmcPostDao

type nmcPostDao struct {

internalNmcPostDao

}

var (

NmcPost = nmcPostDao{

internal.NewNmcPostDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcUserDao = \*internal.NmcUserDao

type nmcUserDao struct {

internalNmcUserDao

}

var (

NmcUser = nmcUserDao{

internal.NewNmcUserDao(),

}

)

package dao

import (

"UniApi/internal/dao/internal"

)

type internalNmcNmcDesignDao = \*internal.NmcNmcDesignDao

type nmcNmcDesignDao struct {

internalNmcNmcDesignDao

}

var (

NmcNmcDesign = nmcNmcDesignDao{

internal.NewNmcNmcDesignDao(),

}

)

package logic

import (

\_ "UniApi/internal/logic/user"

)

package user

import (

"UniApi/internal/dao"

"UniApi/internal/model/do"

"UniApi/internal/model/entity"

"UniApi/internal/service"

"context"

)

func init() {

service.RegisterUser(&sUser{})

}

type sUser struct{}

func (s sUser) GetList(ctx context.Context) (users []entity.NmcUser, err error) {

err = dao.NmcUser.Ctx(ctx).Scan(&users)

return

}

func (s sUser) Add(ctx context.Context, user do.NmcUser) (err error) {

panic("implement me")

}

func (s sUser) Edit(ctx context.Context, user do.NmcUser) (err error) {

panic("implement me")

}

func (s sUser) Del(ctx context.Context) (err error) {

panic("implement me")

}

package packed

package user

import (

"UniApi/api/user"

)

type ControllerV1 struct{}

func NewV1() user.IUserV1 {

return &ControllerV1{}

}

package user

package user

import (

"UniApi/api/user/v1"

"context"

"github.com/gogf/gf/v2/frame/g"

)

func (c \*ControllerV1) UserList(ctx context.Context, req \*v1.UserListReq) (res \*v1.UserListRes, err error) {

r := g.RequestFromCtx(ctx)

md := g.Model("user")

user, err := md.One()

if err == nil {

g.Log().Debugf(ctx, `receive say: %+v`, user.Json())

}

r.Response.WriteJson(user)

return

}

package user

import (

"UniApi/api/user/v1"

"UniApi/internal/service"

"context"

"github.com/gogf/gf/v2/encoding/gbase64"

"github.com/gogf/gf/v2/errors/gerror"

"github.com/gogf/gf/v2/frame/g"

)

func (c \*ControllerV1) UserRegister(ctx context.Context, req \*v1.UserRegisterReq) (res \*v1.UserRegisterRes, err error) {

r := g.RequestFromCtx(ctx)

md := g.Model("user")

accid := "u" + req.Phone

token := gbase64.EncodeString(req.Phone)

data := g.Map{

"name": req.NickName,

"phone": req.Phone,

"password": req.Password1,

"sign": "因你而新，没你不行",

"accid": accid,

"acc\_token": token,

"school\_role": req.Role,

}

isuser, err := md.Where("phone", req.Phone).One()

if err == nil {

if isuser == nil {

g.Log().Debugf(ctx, `无用户，插入数据`)

result, err := md.Insert(data)

if err == nil {

r.Response.WriteJson(result)

g.Log().Debugf(ctx, `成功: %+v`, result)

service.ImRegister(

accid,

token,

req.NickName,

"因你而新，没你不行",

req.Phone,

)

} else {

r.Response.WriteJson(result)

g.Log().Debugf(ctx, `发生错误: %+v`, result)

}

} else {

err = gerror.New("用户已经存在")

}

}

res = &v1.UserRegisterRes{

Token: accid,

TPsession: token,

}

return

}

package user

import (

"context"

"github.com/gogf/gf/v2/errors/gerror"

"github.com/gogf/gf/v2/frame/g"

"UniApi/api/user/v1"

)

func (c \*ControllerV1) UserAccount(ctx context.Context, req \*v1.UserAccountReq) (res \*v1.UserAccountRes, err error) {

r := g.RequestFromCtx(ctx)

uid := r.GetRouter("uid")

user, err := g.Model("user").Where("uid", uid).One()

if err == nil {

if user == nil {

err = gerror.New("查询不到该用户")

} else {

res = &v1.UserAccountRes{UserInfo: user.Map()}

}

g.Log().Debugf(ctx, `receive say: %+v`, user)

}

return

}

package hello

import (

"context"

"github.com/gogf/gf/v2/frame/g"

"UniApi/api/hello/v1"

)

func (c \*ControllerV1) Hello(ctx context.Context, req \*v1.HelloReq) (res \*v1.HelloRes, err error) {

g.RequestFromCtx(ctx).Response.Writeln("Hello World!")

return

}

package hello

package hello

import (

"UniApi/api/hello"

)

type ControllerV1 struct{}

func NewV1() hello.IHelloV1 {

return &ControllerV1{}

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/os/gtime"

)

type NmcComment struct {

g.Meta `orm:"table:comment, do:true"`

CommentId interface{} // 评论id

Content interface{} // 评论

CreateTime \*gtime.Time // 评论时间

IsDelete interface{} // 是否删除

Uid interface{} // 所属用户id

PostId interface{} // 所属文章id

LikeCount interface{} // 点赞次数

RootCommentId interface{} // 顶级评论id

ToCommentId interface{} // 目标评论id

UpdateTime \*gtime.Time // 修改时间

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

)

type NmcPics struct {

g.Meta `orm:"table:pics, do:true"`

Pid interface{} // 图片自增id

Url interface{} // 图片地址

UploadId interface{} // 上传者

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/os/gtime"

)

type NmcTopic struct {

g.Meta `orm:"table:topic, do:true"`

TopicId interface{} // 话题id

TopicCreator interface{} // uid

TopicName interface{} // 话题名字

TopicDc interface{} // 话题描述

TopicUrl interface{} // 话题图片

CreatedTime \*gtime.Time // 创建时间

UpdatedTime \*gtime.Time // 更新时间

IsDelete interface{} // 是否删除

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/os/gtime"

)

type NmcWorkTag struct {

g.Meta `orm:"table:work\_tag, do:true"`

WorkTagId interface{} // 自增id

CreatorId interface{} // 创建者uid

WorkName interface{} // 名字

WorkDc interface{} // 描述

StartTime \*gtime.Time // 开始时间

EndTime \*gtime.Time // 结束时间

ToRole interface{} // 工作人员

UseId interface{} // 使用人员

Status interface{} // 当前状态0，进行中，1结束

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

)

type NmcWork struct {

g.Meta `orm:"table:work, do:true"`

WorkTagId interface{} // 工作主题

WorkId interface{} // 工作

FileUrl interface{} // 文件地址

WorkStatus interface{} // 0-正在，1-审核中, 2-修改中, 3-结束。

ApplyId interface{} // 此工作的工作人员

WorkWatcher interface{} // 初审，一般是部长

WorkWatcher2 interface{} // 再审核

WorkWatcher3 interface{} // 终审

Judge interface{} // 修改意见

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

)

type NmcUserProfile struct {

g.Meta `orm:"table:user\_profile, do:true"`

Id interface{} //

Uid interface{} // user表的id

Name interface{} // 用户名字

Schoolid interface{} // 学号

Spwd interface{} // 教务密码

SchoolSession interface{} // 教务登陆使用

Cnid interface{} // 身份证号

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

)

type NmcBbuRole struct {

g.Meta `orm:"table:bbu\_role, do:true"`

Id interface{} // 自增id

Name interface{} // 角色名

SchoolRole interface{} // 角色id代码

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/os/gtime"

)

type NmcSms struct {

g.Meta `orm:"table:sms, do:true"`

Phone interface{} // 号码验证

Code interface{} // 验证码内容

Time \*gtime.Time // 当前时间

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/os/gtime"

)

type NmcNmcPhoto struct {

g.Meta `orm:"table:nmc\_photo, do:true"`

Pid interface{} // 图片自增id

Url interface{} // 图片地址

UploadId interface{} // 上传者

UpTime \*gtime.Time // 上传时间

DownloadCount interface{} // 下载数

IsVip interface{} // 是否收费,1收费，0不收费，默认1

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/os/gtime"

)

type NmcPost struct {

g.Meta `orm:"table:post, do:true"`

PostId interface{} // 文章id

Uid interface{} // 文章发布者

Address interface{} // 位置

TopicId interface{} // 话题id

PostType interface{} // 1无，2图片，3视频

Content interface{} // 内容

PostLike interface{} // 点赞数量

CreatedTime \*gtime.Time //

IsDelete interface{} // 是否删除0

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/os/gtime"

)

type NmcUser struct {

g.Meta `orm:"table:user, do:true"`

Uid interface{} // 用户ID

Name interface{} // 用户名

Password interface{} // 密码

Phone interface{} // 手机号码

Sign interface{} // 个性签名

AvatarUrl interface{} // 头像链接地址

Verti interface{} // 认证类型

BgUrl interface{} // 背景图链接地址

Status interface{} // 用户封禁状态，0正常，1封禁

Major interface{} // 专业

SchoolRole interface{} // 学校代码5+组织3+组织内部门3+组织内角色1；aaaaa-bbb-ccc+d。默认普通角色除学校代码外全为0

Accid interface{} // 网易云信id

AccToken interface{} // 网易云信token

CreateTime \*gtime.Time // 创建时间

UpdatedTime \*gtime.Time // 修改时间

DeletedAt \*gtime.Time //

Exp interface{} // 等级

Pt interface{} //

}

package do

import (

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/os/gtime"

)

type NmcNmcDesign struct {

g.Meta `orm:"table:nmc\_design, do:true"`

Did interface{} // 设计稿自增id

Url interface{} // 图片地址

UploadId interface{} // 上传者

UpTime \*gtime.Time // 上传时间

DownloadCount interface{} // 下载数

IsVip interface{} // 是否收费,1收费，0不收费，默认1

}

package entity

import (

"github.com/gogf/gf/v2/os/gtime"

)

type NmcComment struct {

CommentId int64 `json:"commentId" description:"评论id"` // 评论id

Content string `json:"content" description:"评论"` // 评论

CreateTime \*gtime.Time `json:"createTime" description:"评论时间"` // 评论时间

IsDelete bool `json:"isDelete" description:"是否删除"` // 是否删除

Uid int `json:"uid" description:"所属用户id"` // 所属用户id

PostId int64 `json:"postId" description:"所属文章id"` // 所属文章id

LikeCount int `json:"likeCount" description:"点赞次数"` // 点赞次数

RootCommentId int64 `json:"rootCommentId" description:"顶级评论id"` // 顶级评论id

ToCommentId int64 `json:"toCommentId" description:"目标评论id"` // 目标评论id

UpdateTime \*gtime.Time `json:"updateTime" description:"修改时间"` // 修改时间

}

package entity

type NmcPics struct {

Pid int64 `json:"pid" description:"图片自增id"` // 图片自增id

Url string `json:"url" description:"图片地址"` // 图片地址

UploadId int `json:"uploadId" description:"上传者"` // 上传者

}

package entity

import (

"github.com/gogf/gf/v2/os/gtime"

)

type NmcTopic struct {

TopicId int `json:"topicId" description:"话题id"` // 话题id

TopicCreator int `json:"topicCreator" description:"uid"` // uid

TopicName string `json:"topicName" description:"话题名字"` // 话题名字

TopicDc string `json:"topicDc" description:"话题描述"` // 话题描述

TopicUrl string `json:"topicUrl" description:"话题图片"` // 话题图片

CreatedTime \*gtime.Time `json:"createdTime" description:"创建时间"` // 创建时间

UpdatedTime \*gtime.Time `json:"updatedTime" description:"更新时间"` // 更新时间

IsDelete bool `json:"isDelete" description:"是否删除"` // 是否删除

}

package entity

import (

"github.com/gogf/gf/v2/os/gtime"

)

type NmcWorkTag struct {

WorkTagId uint `json:"workTagId" description:"自增id"` // 自增id

CreatorId int `json:"creatorId" description:"创建者uid"` // 创建者uid

WorkName string `json:"workName" description:"名字"` // 名字

WorkDc string `json:"workDc" description:"描述"` // 描述

StartTime \*gtime.Time `json:"startTime" description:"开始时间"` // 开始时间

EndTime \*gtime.Time `json:"endTime" description:"结束时间"` // 结束时间

ToRole int64 `json:"toRole" description:"工作人员"` // 工作人员

UseId int64 `json:"useId" description:"使用人员"` // 使用人员

Status bool `json:"status" description:"当前状态0，进行中，1结束"` // 当前状态0，进行中，1结束

}

package entity

type NmcWork struct {

WorkTagId uint `json:"workTagId" description:"工作主题"` // 工作主题

WorkId int64 `json:"workId" description:"工作"` // 工作

FileUrl string `json:"fileUrl" description:"文件地址"` // 文件地址

WorkStatus uint `json:"workStatus" description:"0-正在，1-审核中, 2-修改中, 3-结束。"` // 0-正在，1-审核中, 2-修改中, 3-结束。

ApplyId int `json:"applyId" description:"此工作的工作人员"` // 此工作的工作人员

WorkWatcher int64 `json:"workWatcher" description:"初审，一般是部长"` // 初审，一般是部长

WorkWatcher2 int64 `json:"workWatcher2" description:"再审核"` // 再审核

WorkWatcher3 int64 `json:"workWatcher3" description:"终审"` // 终审

Judge string `json:"judge" description:"修改意见"` // 修改意见

}

package entity

type NmcUserProfile struct {

Id int `json:"id" description:""` //

Uid int `json:"uid" description:"user表的id"` // user表的id

Name string `json:"name" description:"用户名字"` // 用户名字

Schoolid string `json:"schoolid" description:"学号"` // 学号

Spwd string `json:"spwd" description:"教务密码"` // 教务密码

SchoolSession string `json:"schoolSession" description:"教务登陆使用"` // 教务登陆使用

Cnid string `json:"cnid" description:"身份证号"` // 身份证号

}

package entity

type NmcBbuRole struct {

Id int `json:"id" description:"自增id"` // 自增id

Name string `json:"name" description:"角色名"` // 角色名

SchoolRole int64 `json:"schoolRole" description:"角色id代码"` // 角色id代码

}

package entity

import (

"github.com/gogf/gf/v2/os/gtime"

)

type NmcSms struct {

Phone string `json:"phone" description:"号码验证"` // 号码验证

Code string `json:"code" description:"验证码内容"` // 验证码内容

Time \*gtime.Time `json:"time" description:"当前时间"` // 当前时间

}

package entity

import (

"github.com/gogf/gf/v2/os/gtime"

)

type NmcNmcPhoto struct {

Pid int64 `json:"pid" description:"图片自增id"` // 图片自增id

Url string `json:"url" description:"图片地址"` // 图片地址

UploadId int `json:"uploadId" description:"上传者"` // 上传者

UpTime \*gtime.Time `json:"upTime" description:"上传时间"` // 上传时间

DownloadCount int `json:"downloadCount" description:"下载数"` // 下载数

IsVip int `json:"isVip" description:"是否收费,1收费，0不收费，默认1"` // 是否收费,1收费，0不收费，默认1

}

package entity

import (

"github.com/gogf/gf/v2/os/gtime"

)

type NmcPost struct {

PostId int64 `json:"postId" description:"文章id"` // 文章id

Uid int `json:"uid" description:"文章发布者"` // 文章发布者

Address string `json:"address" description:"位置"` // 位置

TopicId int `json:"topicId" description:"话题id"` // 话题id

PostType uint `json:"postType" description:"1无，2图片，3视频"` // 1无，2图片，3视频

Content string `json:"content" description:"内容"` // 内容

PostLike uint64 `json:"postLike" description:"点赞数量"` // 点赞数量

CreatedTime \*gtime.Time `json:"createdTime" description:""` //

IsDelete bool `json:"isDelete" description:"是否删除0"` // 是否删除0

}

package entity

import (

"github.com/gogf/gf/v2/os/gtime"

)

type NmcUser struct {

Uid int `json:"uid" description:"用户ID"` // 用户ID

Name string `json:"name" description:"用户名"` // 用户名

Password string `json:"password" description:"密码"` // 密码

Phone string `json:"phone" description:"手机号码"` // 手机号码

Sign string `json:"sign" description:"个性签名"` // 个性签名

AvatarUrl string `json:"avatarUrl" description:"头像链接地址"` // 头像链接地址

Verti string `json:"verti" description:"认证类型"` // 认证类型

BgUrl string `json:"bgUrl" description:"背景图链接地址"` // 背景图链接地址

Status bool `json:"status" description:"用户封禁状态，0正常，1封禁"` // 用户封禁状态，0正常，1封禁

Major string `json:"major" description:"专业"` // 专业

SchoolRole string `json:"schoolRole" description:"学校代码5+组织3+组织内部门3+组织内角色1；aaaaa-bbb-ccc+d。默认普通角色除学校代码外全为0"` // 学校代码5+组织3+组织内部门3+组织内角色1；aaaaa-bbb-ccc+d。默认普通角色除学校代码外全为0

Accid string `json:"accid" description:"网易云信id"` // 网易云信id

AccToken string `json:"accToken" description:"网易云信token"` // 网易云信token

CreateTime \*gtime.Time `json:"createTime" description:"创建时间"` // 创建时间

UpdatedTime \*gtime.Time `json:"updatedTime" description:"修改时间"` // 修改时间

DeletedAt \*gtime.Time `json:"deletedAt" description:""` //

Exp uint `json:"exp" description:"等级"` // 等级

Pt uint `json:"pt" description:""` //

}

package entity

import (

"github.com/gogf/gf/v2/os/gtime"

)

type NmcNmcDesign struct {

Did int64 `json:"did" description:"设计稿自增id"` // 设计稿自增id

Url string `json:"url" description:"图片地址"` // 图片地址

UploadId int `json:"uploadId" description:"上传者"` // 上传者

UpTime \*gtime.Time `json:"upTime" description:"上传时间"` // 上传时间

DownloadCount int `json:"downloadCount" description:"下载数"` // 下载数

IsVip bool `json:"isVip" description:"是否收费,1收费，0不收费，默认1"` // 是否收费,1收费，0不收费，默认1

}

package service

import (

"encoding/json"

"fmt"

"github.com/gogf/gf/v2/crypto/gsha1"

"github.com/gogf/gf/v2/frame/g"

"github.com/gogf/gf/v2/os/gctx"

"github.com/gogf/gf/v2/os/gtime"

"github.com/gogf/gf/v2/util/gconv"

"github.com/gogf/gf/v2/util/grand"

"os"

)

type ImService struct {

AppKey string

AppSecret string

}

type ImAccount struct {

AccountId string `json:"account\_id"`

Token string `json:"token"`

UserInformation \*UserInformation `json:"user\_information"`

}

type UserInformation struct {

Name string `json:"name"`

Sign string `json:"sign"`

Mobile string `json:"mobile"`

}

func (h \*ImService) httpDo(i ImAccount) {

AppKey := h.AppKey

CurTime := gconv.String(gtime.Timestamp())

Nonce := grand.Digits(8)

CheckSum := gsha1.Encrypt(h.AppSecret + Nonce + CurTime)

println("AppKey =", AppKey)

println("Nonce =", Nonce)

println("CurTime =", CurTime)

println("CheckSum =", CheckSum)

c := g.Client()

header := map[string]string{

"Content-Type": "application/json; charset=utf-8",

"AppKey": AppKey,

"Nonce": Nonce,

"CurTime": CurTime,

"CheckSum": CheckSum,

}

j, err := json.Marshal(i)

if err != nil {

fmt.Println("转换为json错误")

os.Exit(-1)

}

c.SetHeaderMap(header)

content := c.PostContent(

gctx.New(),

"https://open.yunxinapi.com/im/v2/accounts",

j,

)

fmt.Println(content)

}

func ImRegister(accid, token, name, sign, mobile string) {

iaccount := ImAccount{

AccountId: accid,

Token: token,

}

tempUserInfo := new(UserInformation)

tempUserInfo.Name = name

if sign == "" {

tempUserInfo.Sign = "因为ta太懒了，还没有设置个性签名哦！"

}

tempUserInfo.Sign = sign

tempUserInfo.Mobile = mobile

iaccount.UserInformation = tempUserInfo

im := ImService{AppKey: "05b4c8c3883a6426f3590e5f8fba3762", AppSecret: "e3961aba4f93"}

im.httpDo(iaccount)

}

package service

import (

"UniApi/internal/model/do"

"UniApi/internal/model/entity"

"context"

)

type (

IUser interface {

GetList(ctx context.Context) (users []entity.NmcUser, err error)

Add(ctx context.Context, user do.NmcUser) (err error)

Edit(ctx context.Context, user do.NmcUser) (err error)

Del(ctx context.Context) (err error)

}

)

var (

localUser IUser

)

func User() IUser {

if localUser == nil {

panic("implement not found for interface IUser, forgot register?")

}

return localUser

}

func RegisterUser(i IUser) {

localUser = i

}

package consts

package v1

package v1

import (

"fmt"

unisms "github.com/apistd/uni-go-sdk/sms"

)

func main() {

client := unisms.NewClient("your access key id", "your access key secret") // 若使用简易验签模式仅传入第一个参数即可

message := unisms.BuildMessage()

message.SetTo("your phone number")

message.SetSignature("UniSMS")

message.SetTemplateId("login\_tmpl")

message.SetTemplateData(map[string]string{"code": "6666"}) // 设置自定义参数 (变量短信)

res, err := client.Send(message)

if err != nil {

fmt.Println(err)

return

}

fmt.Println(res)

}

package v1

import "github.com/gogf/gf/v2/frame/g"

type UserAccountReq struct {

g.Meta `path:"user/u{uid}" method:"get"`

}

type UserAccountRes struct {

UserInfo map[string]any

}

package v1

import (

"github.com/gogf/gf/v2/frame/g"

)

type UserListReq struct {

g.Meta `path:"user/list" method:"get"`

}

type UserListRes struct {

}

package v1

import "github.com/gogf/gf/v2/frame/g"

type UserRegisterReq struct {

g.Meta `path:"user/register" method:"get"`

NickName string `dc:"用户名" v:"required|max-length:16#请输入用户名|用户名最大长度为{max}位"`

Phone string `dc:"手机号" v:"required|phone#请输入手机号|请输入正确到手机号"`

Password1 string `dc:"密码" v:"required|length:6,30#请输入密码|密码长度为{min}到{max}位"`

Password2 string `dc:"重复密码" v:"required|same:Password1#请确认密码|两次密码不相符"`

SmsCode string `dc:"验证码" v:"required|size:6#请输入验证码|验证码长度不相符"`

Role string `dc:"用户角色" d:"113050000000" v:"size:12#请输入12位数字码"`

}

type UserRegisterRes struct {

Token string `dc:"acctoken与云信账号关联"`

TPsession string `dc:"鉴权"`

}

package user

import (

"context"

"UniApi/api/user/v1"

)

type IUserV1 interface {

UserAccount(ctx context.Context, req \*v1.UserAccountReq) (res \*v1.UserAccountRes, err error)

UserList(ctx context.Context, req \*v1.UserListReq) (res \*v1.UserListRes, err error)

UserRegister(ctx context.Context, req \*v1.UserRegisterReq) (res \*v1.UserRegisterRes, err error)

}

package v1

package v1

import (

"github.com/gogf/gf/v2/frame/g"

)

type HelloReq struct {

g.Meta `path:"/hello" tags:"Hello" method:"get" summary:"You first hello api"`

}

type HelloRes struct {

g.Meta `mime:"text/html" example:"string"`

}

package hello

import (

"context"

"UniApi/api/hello/v1"

)

type IHelloV1 interface {

Hello(ctx context.Context, req \*v1.HelloReq) (res \*v1.HelloRes, err error)

}

package main

import (

\_ "UniApi/internal/logic"

\_ "UniApi/internal/packed"

"github.com/gogf/gf/v2/os/gctx"

"UniApi/internal/cmd"

)

func main() {

cmd.Main.Run(gctx.GetInitCtx())

}