痴商Server端前六总结

结构

```
server [chimall-server] ~/Documents/data/langwan/chihu
component
  > config viper
  > mysql gorm gen
✓ logic
  > orm
  > password
  > token
✓ ■ service
  > ■ account 有身份
  > server
go.sum
🗸 🥤 main.go
     f main()
  grouter.go
  de config.yaml
IIII External Libraries
Scratches and Consoles
```

```
github.com/gin-gonic/gin v1.9.0
github.com/go-playground/validator/v10 v10.11.2
github.com/langwan/langgo v0.5.53
github.com/spf13/viper v1.15.0
gorm.io/driver/mysql v1.4.3
gorm.io/gen v0.3.21
gorm.io/gorm v1.24.6
github.com/dgrijalva/jwt-go v3.2.0+incompatible
```

```
> account

✓ ■ service

   ∨ h

✓ account

     > 🥤 auth.go

✓ account

        🧃 http.go
                                                          🗸 🥛 Profile.go
     > 🥤 validators.go
                                                               f Profile(*gin.Context)
   > server
                                                       ✓ ■ order
                                                          > 🦉 Get.go
server:
                                                          🗦 🥤 List.go
 host: 127.0.0.1:8080
                                                     > h
jwt:

✓ ■ server

 key: "jwt-chihuo-key"
                                                        > ¡ Login.go
n_sql:
                                                       > 🥞 Logout.go
 dsn: "chimall:password@tcp(localhost:3306)/
                                                        Register.go
✓ logic
                                                         ∨ l h
   ✓ lim orm
                                                            🗡 🥤 auth.go
     ✓ ■ dal
                                                                 f Auth():gin.HandlerFunc
           accounts.gen.go
                                                           Y 🥤 http.go
                               gen 生成的代码
        > 🧃 gen.go
                                                                 f Fail(*gin.Context, error)

✓ ■ model
                                                                 f FailCode(*gin.Context, int, error)
        accounts.gen.go
                                                                 f FailMessage(*gin.Context, string)
           > T Account
                                                                 forbidden(*gin.Context)
             m TableName():string of *Account
                                                                 f Ok(*gin.Context, any)
             TableNameAccount:string
                                                                 RemoveCookie(*gin.Context, string)
                                                                 SetCookie(*gin.Context, string, string)

✓ password

                                                                 Validator(*gin.Context, error)
      > 🧃 password.go
                                                                 • ValidatorError(*gin.Context, string, string)
        password_test.go
                                                            🗡 🧃 validators.go

✓ limit token

                                                                 f init()
     Trans:ut.Translator
        > Token
           f Parse(string):(*Token, error)
```

Sign(*Token):(string, error)

✓ service



```
package main

import (
    "github.com/gin-gonic/gin"
    _ "server/component/config"
    _ "server/component/mysql"
)

func main() {
    g := gin.New()
    Router(g)
    g.Run(":3003")
}
```

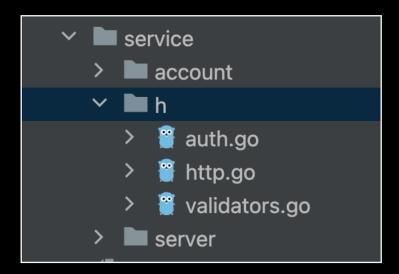
路由

```
func Router(g *gin.Engine) {
 v1 := g.Group("api/v1")
   //登录 注册 忘记密码 注销
   v1.POST("/login", server.Login)
   v1.POST("/register", server.Register)
   v1.POST("/logout", server.Logout)
 account := v1.Group("/account")
 account.Use(h.Auth())
 order := account.Group("/order")
 order.POST("/list", serviceOrder.List)
 order.POST("/get", serviceOrder.Get)
 account.POST("/profile", serviceAccount.Profile)
```

绑定校验

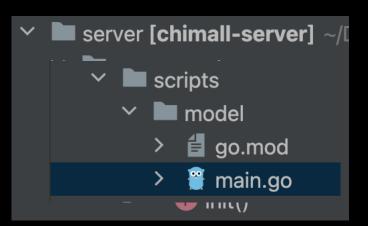
```
type LoginRequest struct {
          string `json:"phone" binding:"required,len=11" label:"手机号"`
 Phone
 Password string `json:"password" binding:"required,len=40" label:"密码"`
type LoginResponse struct {
 Token string `json:"token" `
func Login(c *gin.Context) {
 var request LoginRequest
 if err := c.ShouldBindJSON(&request); err != nil {
   h.Validator(c, err)
   return
```

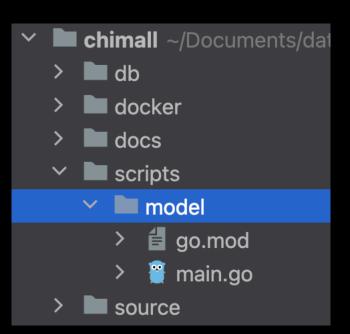
校验库



```
var (
  Trans ut. Translator
func init() {
  if v, ok := binding.Validator.Engine().(*validator.Validate); ok {
    zhHans := localeZhHans.New()
    en := localeEn.New()
    uni := ut.New(zhHans, en)
    Trans, _ = uni.GetTranslator("zhHans")
    err := zh.RegisterDefaultTranslations(v, Trans)
    if err != nil {
      return
    v.RegisterTagNameFunc(func(field reflect.StructField) string {
      return field.Tag.Get("label")
    v.RegisterTranslation("required", Trans, func(ut ut.Translator) error {
      return ut.Add("required", "请输入{0}", true)
    }, func(ut ut.Translator, fe validator.FieldError) string {
       t, err := ut.T(fe.Tag(), fe.Field())
       if err != nil {
         return fe.(error).Error()
      return t
```

ORM查询

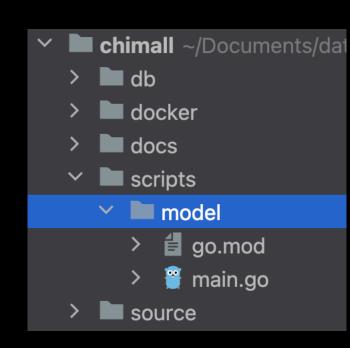




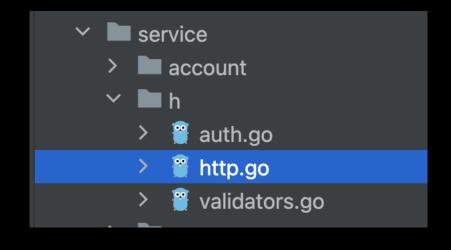
```
acc, err := dal.Account.Where(dal.Account.Phone.Eq(request.Phone)).First()
if err != nil {
   h.Fail(c, err)
   return
}
```

生成ORM代码

```
const dsn = "chimall:password@tcp(localhost:3306)/chimall?charset=utf8mb4&parseTime=True&loc=Loc
func main() {
 db, err := gorm.Open(mysql.Open(dsn))
 if err != nil {
   panic(fmt.Errorf("cannot establish db connection: %w", err))
  // 生成实例
 g := gen.NewGenerator(gen.Config{
   OutPath: "../../source/server/logic/orm/dal",
   ModelPkgPath: "../../source/server/logic/orm/model",
                 gen.WithDefaultQuery | gen.WithoutContext,
   Mode:
 })
  // 设置目标 db
 g.UseDB(db)
 g.ApplyBasic(g.GenerateAllTable()...)
 g.Execute()
```



返回函数



```
func Ok(c *gin.Context, body any)
                                                             200
func Fail(c *gin.Context, err error)
                                                             400
                                                             400
func FailMessage(c *gin.Context, message string)
                                                             400
func FailCode(c *gin.Context, code int, err error)
func Forbidden(c *gin.Context)
                                                             401
                                                             422
func Validator(c *gin.Context, err error)
func ValidatorError(c *gin.Context, tag, message string)
                                                             422
func SetCookie(c *gin.Context, key, val string)
func RemoveCookie(c *gin.Context, key string)
```

密码处理

```
服务器端
 password := logicPassword.Hash(request.Password, salt)
if len(acc.Password) > 0 &&
                                                                                     ✓ logic
                                                                                      > orm
                                                                                      > password
acc.Password == logicPassword.Hash(request.Password, acc.Salt) {
                                                                                         password.go
                                                                                         👸 password_test.go
                                                        服务器端
                                                        func Hash(password, salt string) string {
客户端
                                                          mac := hmac.New(sha1.New, []byte(password))
                                                          mac.Write([]byte(salt))
let hp = hashPassword(values.password);
                                                          hs := mac.Sum(nil)
                                                          return fmt.Sprintf("%x", hs)
import sha1 from "js-sha1";

✓ src

export default function hashPassword(password) {

    ∨ Helper

                                                                        Js axios.js
  const salt = password.substring(0, password.length / 2);
                                                                        Js password.js
  return sha1(sha1(password) + salt);

∨ Layout
```

认证 JWT

```
✓ limit token

                                                   token.go
                                                     > Token
type Token struct {
                                                      Parse(string):(*Token, error)
  Uid
           string `json:"uid"`
                                                      f Sign(*Token):(string, error)
  Nickname string `json:"nickname"`
  jwt.StandardClaims
func Sign(token *Token) (string, error) {
  return jwt.NewWithClaims(jwt.SigningMethodHS256, token).SignedString([]byte(config.GetString("jwt.key")))
func Parse(sign string) (*Token, error) {
  tokenClaims, err := jwt.ParseWithClaims(sign, &Token{}, func(token *jwt.Token) (interface{}, error) {
    return []byte(config.Config.GetString("jwt.key")), nil
  if err != nil {
    return nil, err
  } else {
    if tokenClaims != nil {
       if claims, ok := tokenClaims.Claims.(*Token); ok && tokenClaims.Valid {
         return claims, nil
         else {
         return nil, errors.New("claims error")
      else {
       return nil, errors.New("claims error")
```

✓ logic

> orm

> **p**assword



```
    server [chimall-server] ~/Document
    component
    config
    config.go
    f init()
    Config:*viper.Viper
```

```
var Config *viper.Viper
func init() {
 Config = viper.New()
 wd, err := os.Getwd()
 if err != nil {
   panic(err)
 Config.AddConfigPath(wd)
 Config.SetConfigName("config")
 Config.SetConfigType("yaml")
  if err := Config.ReadInConfig(); err != nil {
   panic(err)
```