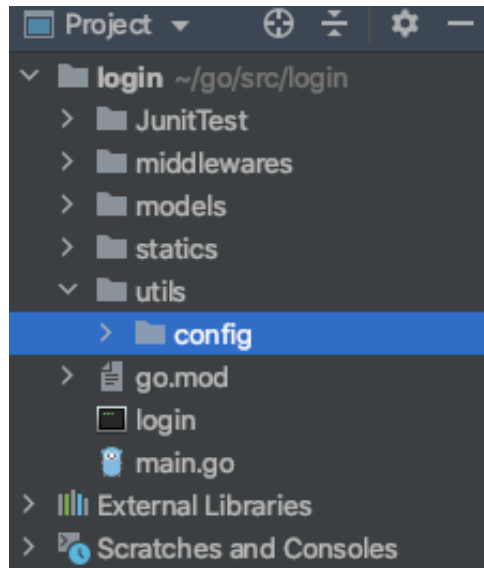


登陆注册后台

简介

(本来想写一个注册手机短信验证码的,但是在腾讯云申请短信签名通过不了, 所以对于注册使用了随机来生产验证码存放着Redis中)

首先呢、讲一下项目的整个目录



1.JunitTest是单元测试的文件夹：里面主要写了对users数据库的操作的一些测试。对于每一个功能点一个测试

2.middlewares是实现中间件的一个文件：里面主要实现了jwt来限制未登陆的用户访问特定的api。

3.models文件夹：主要写了对于User类型，以及Controller层的操作也写在里面了。(其实应该在新建一个Controller文件夹的)

4.statics文件夹：存放了静态网页，对于前端部分没有实现，只是写了个简单的页面测试。

5.utils文件夹：写了对于Postgres、Redis数据库的配置及调用连接。

实现

1.使用了postgres

对数据库的连接操作：

```
type DBServer struct {
    Host string //主机
    Port int  //端口
    DBName string //数据库名
    User string //登陆用户
    Password string //密码
```

```

}

func (db DBServer) ConnectString() string {
    if db.User == "" {
        return fmt.Sprintf("host=%s port=%d dbname=%s sslmode=disable",
            db.Host, db.Port, db.DBName)
    }
    return fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s
sslmode=disable",
        db.Host, db.Port, db.User, db.Password, db.DBName)
}

func (db DBServer) NewGormDB() (DB *gorm.DB ,err error){
    return gorm.Open("postgres",db.ConnectString())
}

func (db DBServer) NewPostgresDB( ) (DB *sqlx.DB, err error) {
    return sqlx.Open("postgres",db.ConnectString())
}

```

2.使用了Redis

连接本地Redis:

```

func NewRedis() (rdb *redis.Client){
    return redis.NewClient((*redis.Options)(&RedisServer{
        Addr:      "localhost:6379",
        Password:  "",
        DB:        0,
    })))
}

```

3.对于api使用了简单的路由分组和中间件:

```

v1 := r.Group( relativePath: "/api")
{
    //更新用户信息:修改密码或用户名
    v1.POST( relativePath: "/user/update", middlewares.TokenMiddleware(), func(c *gin.Context) {...})

    //获取用户自身信息
    v1.POST( relativePath: "/user/get", middlewares.TokenMiddleware(), func(c *gin.Context) {...})

    //用户列表
    v1.POST( relativePath: "/user/list", middlewares.TokenMiddleware(), func(c *gin.Context) {...})

    //增加用户
    v1.POST( relativePath: "/user/add", Userregister())
}

```

4.TokenMiddleware中间件:使用jwt协议来限制未登陆的用户访问特定的api:

```

func TokenMiddleware() gin.HandlerFunc{
    return func(c *gin.Context) {
        tokenStr := c.Request.Header.Get("token")
        if tokenStr == "" {
            c.JSON(http.StatusUnauthorized,gin.H{
                "result": Result{
                    Message: "未登陆",
                    Status: 0,
                },
            })
            //终止后面的HandlerFunc
            c.Abort()
            return
        }
        claims, err := ParseToken(tokenStr)
        if err != nil {
            c.JSON(http.StatusUnauthorized,gin.H{
                "result": Result{
                    Message: "解析token失败",
                    Status: 0,
                },
            })
            c.Abort()
            return
        }
        c.Set("mobilenumber",claims["mobilenumber"])
        c.Set("username",claims["username"])
        c.Next()
    }
}

```

5.定义了一个返回json的类型，对于每一个返回json数据都包含“result”: Result{}

```

//Json返回结果
type Result struct {
    Message string `json:"msg"`
    Status int `json:"status"`
}

```

6.API(不足：对于这个没有使用resultful):

请求方式	地址	说明
POST	http://localhost:8080/api/user/update	更新用户信息
POST	http://localhost:8080/api/user/get	获取用户自身信息
POST	http://localhost:8080/api/user/list	获取用户列表
POST	http://localhost:8080/api/user/add	增加用户(注册)
GET	http://localhost:8080/login	用户登陆界面
POST	http://localhost:8080/login	用户登陆操作
GET	http://localhost:8080/register	用户注册界面
POST	http://localhost:8080/register	用户注册操作

7.由于之前设想的使用腾讯云来实现短信验证注册(但是签名无法通过审核), 使用了随机数来模拟验证码, 随机数存放至Redis中

```
//初始化随机数的资源库
rand.Seed(time.Now().UnixNano())
num := rand.Int31n(10000)
Rdb.Set(mobilenumber, num, 60*time.Second) //存储1分钟

//获取验证码 verify为用户输入验证码 checkverify为随机生成存储至redis中的验证码
verify := c.PostForm("verify") //验证码
checkverify, _ := Rdb.Get(user.MobileNumber).Result()
```

测试

首先需要通过手机号码收到注册验证码, 这里是使用一个请求来模拟获取。

POST http://localhost:8080/r...

POST http://localhost:8080/v...

POST http://localhost:8080/a...

+

⋮

No Environment

⌵

⌵

Untitled Request

BUILD

✎

POST

⌵

http://localhost:8080/verify

Send

⌵

Save

ParamsAuthorizationHeaders (8)Body●Pre-request ScriptTestsSettings

Cooki

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

	KEY	VALUE	DESCRIPTION	⋮
<input checked="" type="checkbox"/>	mobilenumber	13767511111		
	Key	Value	Description	

BodyCookiesHeaders (3)Test Results

🌐Status: 200 OKTime: 61 msSize: 171 B

Save Res

PrettyRawPreviewVisualize

JSON

⌵

≡

```
1 {
2   "result": {
3     "msg": "ok",
4     "status": 1
5   },
6   "verify": 7758
7 }
```

POST http://localhost:8080/r...

POST http://localhost:8080/v...

POST http://localhost:8080/a...

+

⋮

No Environment

⌵

⌵

Untitled Request

BUILD

✎

📄

POST

⌵

http://localhost:8080/register

Send

⌵

Save

⌵

ParamsAuthorizationHeaders (8)Body●Pre-request ScriptTestsSettings

CookiesCor

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

	KEY	VALUE	DESCRIPTION	⋮	Bulk Edit
<input checked="" type="checkbox"/>	mobilenumber	13767511111			
<input checked="" type="checkbox"/>	password	123456			
<input checked="" type="checkbox"/>	checkpassword	123456			
<input checked="" type="checkbox"/>	verify	7758			
<input checked="" type="checkbox"/>	username	注册测试			

BodyCookiesHeaders (3)Test Results

🌐Status: 200 OKTime: 123 msSize: 167 B

Save Response

PrettyRawPreviewVisualize

JSON

⌵

≡

```
1 {
2   "result": {
3     "msg": "注册成功",
4     "status": 1
5   }
6 }
```


POST http://localhost:8080/r... POST http://localhost:8080/v... POST http://localhost:8080/a... + ... No Environment

Untitled Request BUILD

POST http://localhost:8080/api/user/get Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Headers 8 hidden

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Prese
<input checked="" type="checkbox"/>	token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJtb2JpbG...				
	Key	Value	Description			

Body Cookies Headers (3) Test Results Status: 200 OK Time: 26 ms Size: 236 B Save Respor

Pretty Raw Preview Visualize JSON

```
1 {
2   "result": {
3     "msg": "OK",
4     "status": 1
5   },
6   "user": {
7     "MobileNumber": "13767511111",
8     "userName": "注册测试",
9     "password": ""
10  }
```

代码中也还有一些单元测试、测试Controller层的功能