

# FISCOBCOS中的GoSDK操作---(引入工具包版本)

## FISCOBCOS中的GoSDK操作---(引入工具包版本)

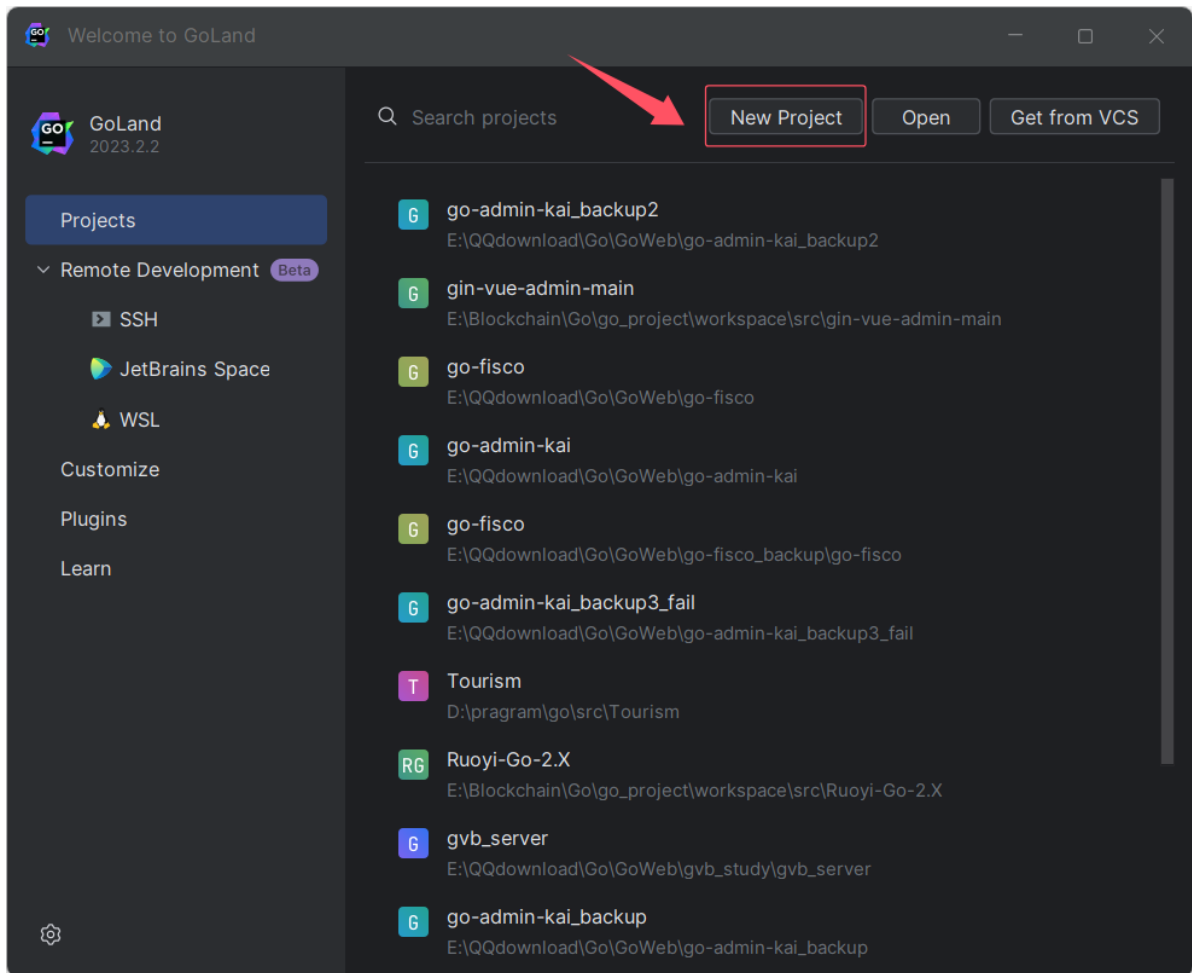
- 一， GoWEB项目创建：
- 二， 项目书写：
  - 1.创建目录：
  - 2.合约配置文件的读取：
  - 3.连接FISCOBCOS的网络配置文件读取：
  - 4.HelloWorld的调用（通过crypto/ecdsa包下的函数生成私钥，发送交易。）
  - 5.Anouncement的调用（通过crypto/ecdsa包下的函数生成私钥，发送交易。）
  - 6.通过WeBASE导出的私钥发送交易
- 三， 小结：

## 文档操作记录

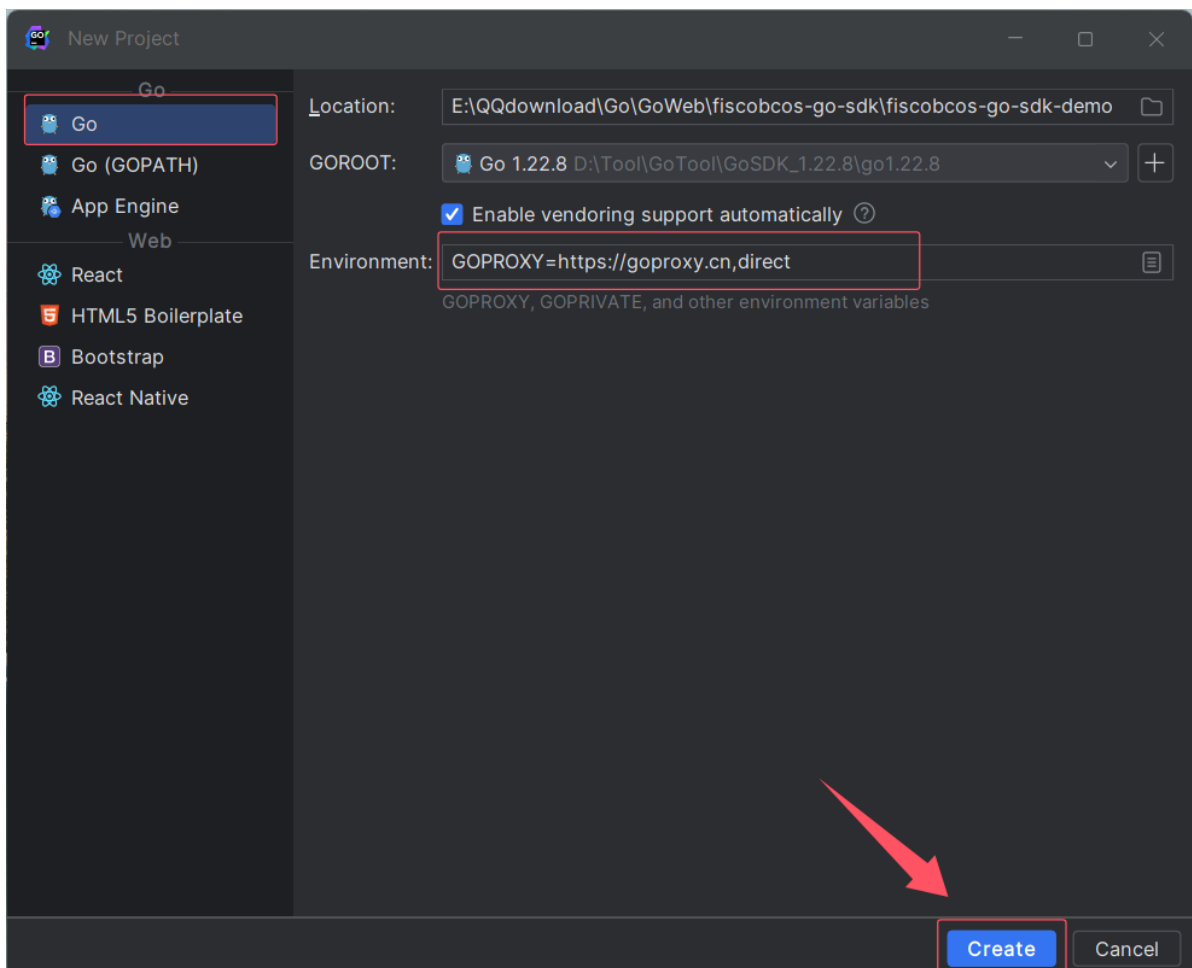
内容	作者	时间	版本号	联系方式
文档创建	陈雅凯，高俭豪，谭俊	2024-11-16	1.0	email: <a href="mailto:2040575063@qq.com">2040575063@qq.com</a>

## 一， GoWEB项目创建：

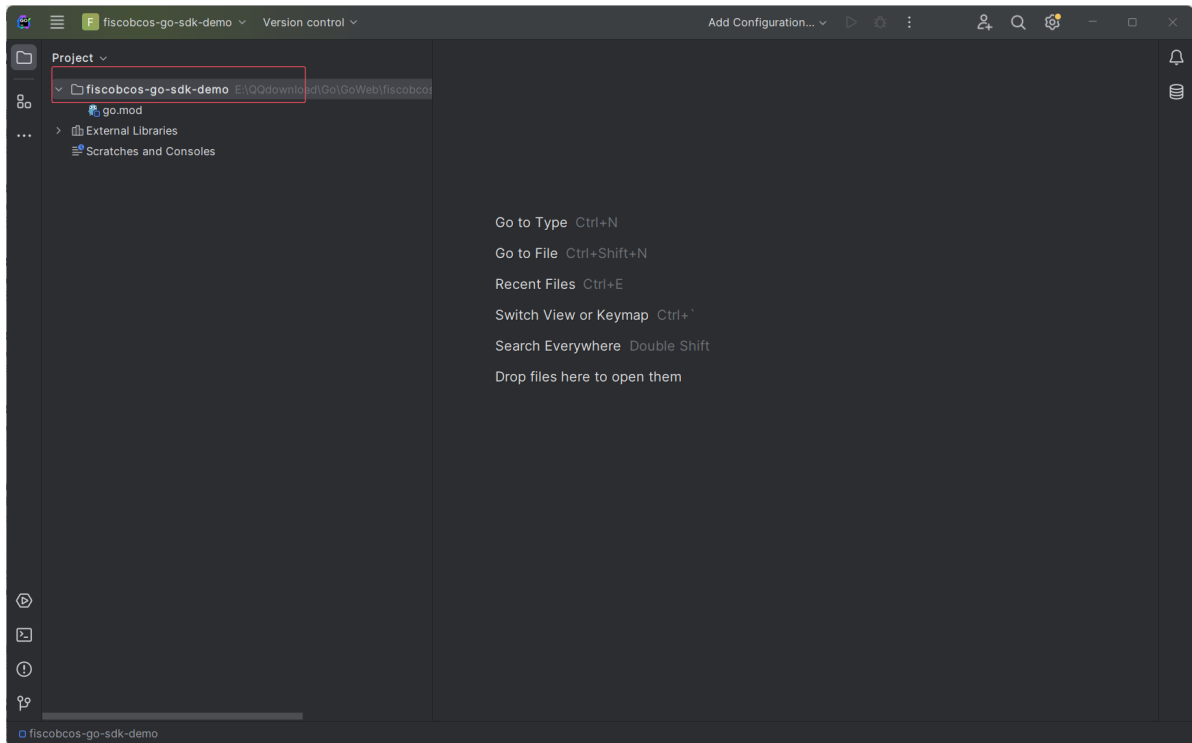
- 1.新建项目：



2.项目路径（随自己心意）：



### 3.生成成功：

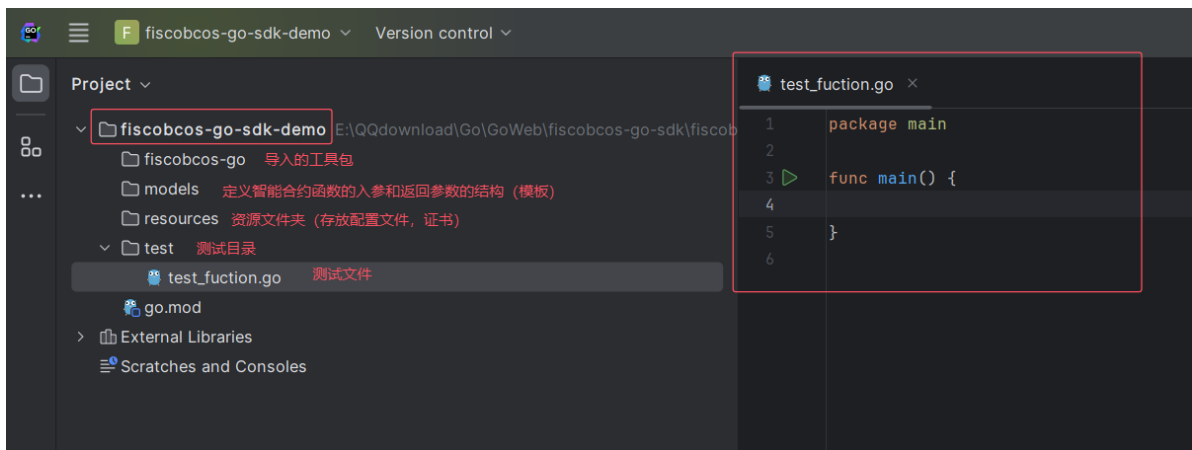


## 二，项目书写：

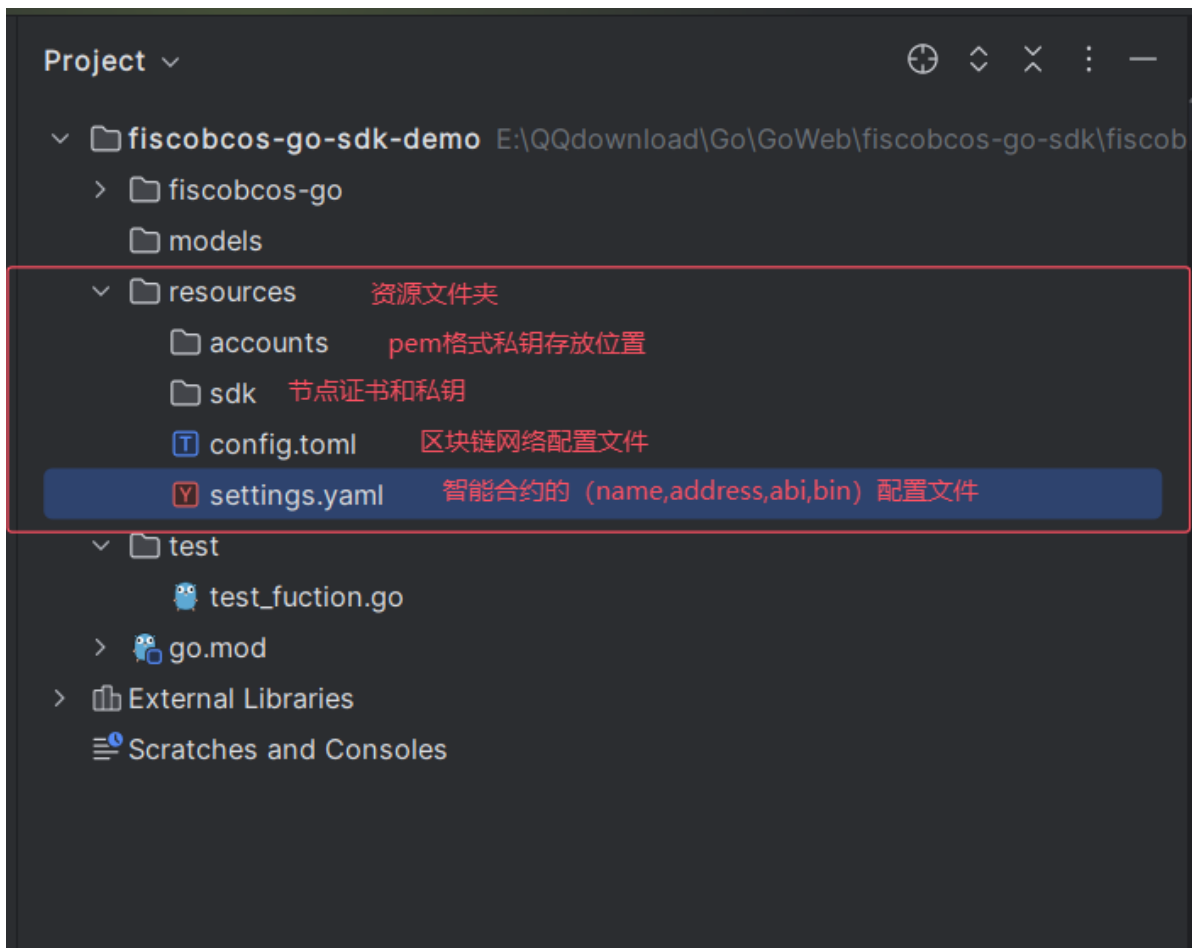
### 1.创建目录：

注意事项：

**fiscobcos-go**在打包好的文件夹中寻找。导入之后会提示项目目录爆红，修改成自己的就好了



resources文件夹配置：



导入工具包fiscobcos-go之后(需要引入的依赖):

```
go get -u github.com/FISCO-BCOS/go-sdk
go get -u gopkg.in/yaml.v3
```



## 2.合约配置文件的读取:

书写配置文件settings.yaml(书写时, 输入自己合约对应的name(合约名称),address,abi,bin。bin可以填写, 也可以不用填写):

```
contracts:
  contract1: (这里的设计只是一个键对应一个结构体数据)
    name:
    address:
    abi:
    bin:
  contract2: (这里的设计只是一个键对应一个结构体数据)
    name:
    address:
    abi:
    bin:
```



```

//SPDX-License-Identifier: MIT
pragma solidity ^0.6.10;
pragma experimental ABIEncoderV2;

contract Anonouncement {

    struct Announcement {
        uint256 id;//宣言ID
        string userID;//用户编号
        string userPK;//用户的公钥
        string nounce;//签名的随机值
        string message;//消息
        string cipher;//消息的密文    --消息
        string attachment;//附件的摘要    --哈希
        uint256 signtime;//存证时间
    }

    mapping(uint256 => Announcement) public announcements;
    uint256 nextId = 1;
    // uint256 nextline;
    // 定义查询条件结构体
    struct QueryConditions {
        // bool byId;
        // uint256 id;
        bool byUserID;
        string userID;
        bool byTimeRange;
        uint256 startTime;
        uint256 endTime;
        // 可以根据需要继续添加其他查询条件字段，比如按消息内容、公钥等查询
    }

    // 添加公告的函数
    // function addAnnouncement(uint256 id, Announcement memory announcement) public
    returns (bool) {
        //     if (announcements[id].id == 0) {
        //         announcements[id] = announcement;
        //         return true;
        //     }
        //     return false;
        // }

    function addAnnouncement(string memory userID,string memory userPK,string
    memory nounce,
        string memory message,string memory cipher,string memory attachment,uint256
    signtime) public returns (bool) {
        Announcement memory announcement =
    Announcement(nextId,userID,userPK,nounce,message,cipher,attachment,signtime);
        announcements[nextId] = announcement;
        nextId++;
        return true;
    }

    // 根据公告ID获取公告信息的函数

```

```

function getAnnouncement(uint256 id) public view returns (Announcement memory)
{
    return announcements[id];
}

// 根据查询条件、每页显示数量和起始位置返回公告列表的函数
function listAnnouncement(bool byUserID,string memory userID,bool
byTimeRange,uint256 startTime,uint256 endTime,uint256 pageSize, uint256
startIndex) public view returns (Announcement[] memory) {
    uint256[] memory ids = new uint256[](120);
    // uint256[] ids;

    uint256 foundCount = 0;
    uint256 totalMatched = 0;
    uint256 nextline = 0;

    // 遍历所有公告
    for (uint256 i = 1; i < nextId; i++) {
        // Announcement storage ann = announcements[i];
        Announcement memory ann = announcements[i];

        bool isMatch = true;

        // 检查是否满足查询条件
        // if (byId && ann.id!= id) {
        //     isMatch = false;
        // }

        if (byUserID && keccak256(abi.encodePacked(ann.userID))!=
keccak256(abi.encodePacked(userID))) {
            isMatch = false;
        }

        if (byTimeRange && (ann.signtime < startTime || ann.signtime >
endTime)) {
            isMatch = false;
        }

        // 如果满足查询条件
        if (isMatch) {
            totalMatched++;

            // 判断是否在当前页范围内
            if (totalMatched > startIndex && foundCount < pageSize) {
                // ids.push(i);
                ids[nextline] = i;
                nextline ++;
                foundCount++;
            }
        }
    }

    // 创建一个数组来存储最终要返回的公告列表
    Announcement[] memory result = new Announcement[] (foundCount);

    // 将满足条件的公告添加到结果数组中

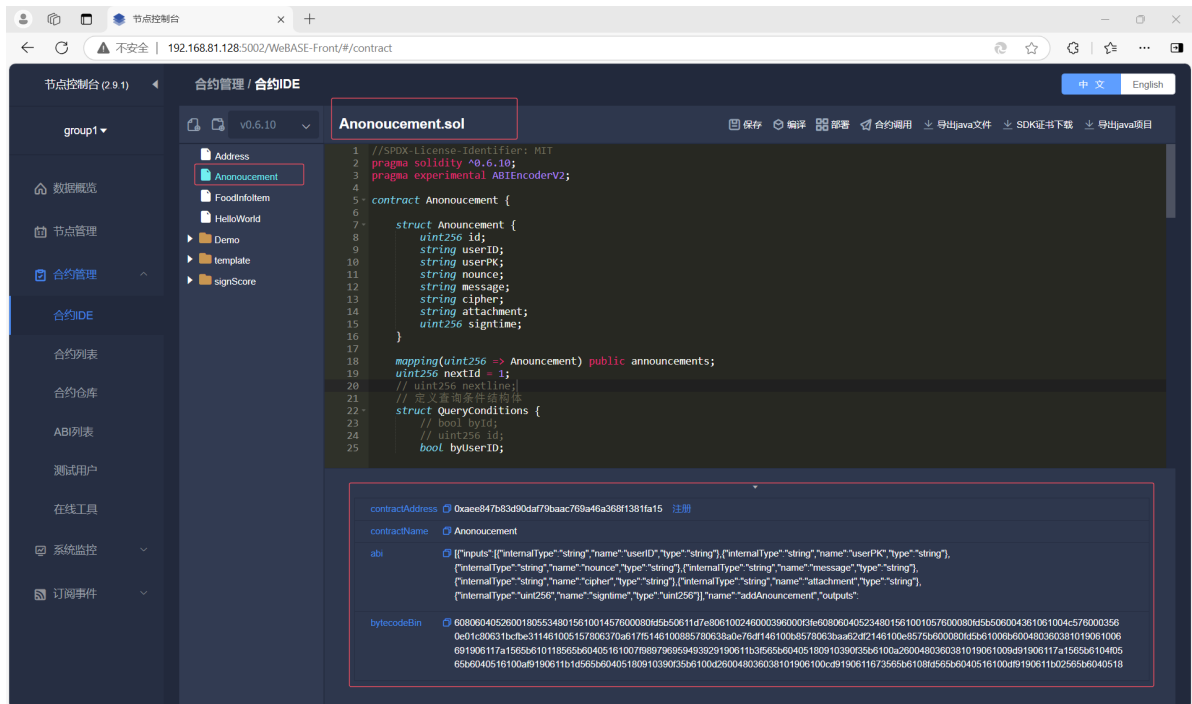
```

```

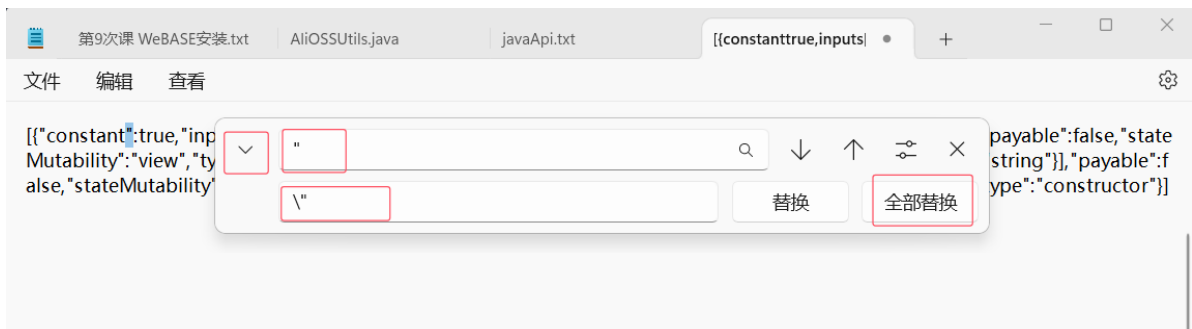
    for (uint256 j = 0; j < foundCount; j++) {
        result[j] = announcements[ids[j]];
    }

    return result;
}
}

```



(1) 这里的abi要将其中的 " 替换成 ",可以打开一个记事本,放入abi,使用快捷键ctrl+f, 全部替换。



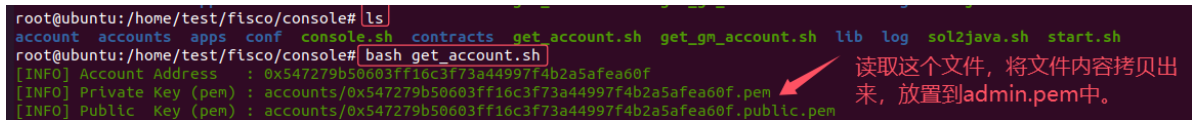
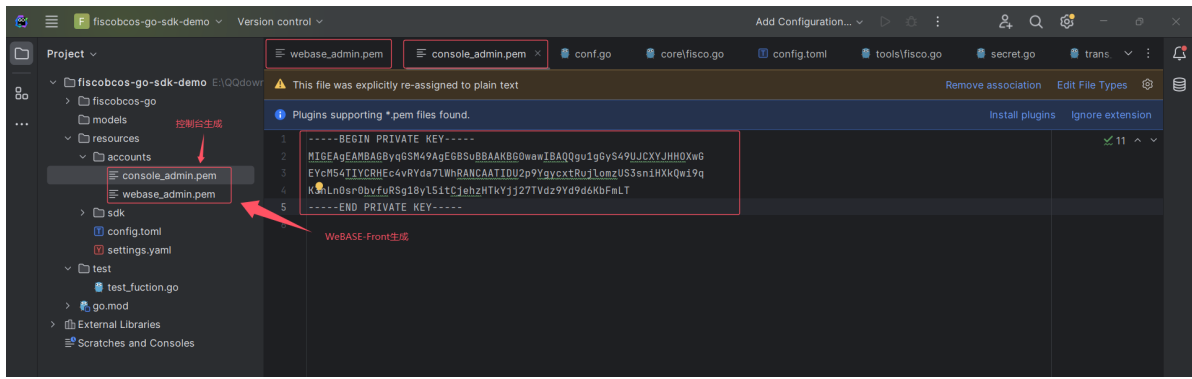
(2) 也可以不用替换, 直接将abi赋值给whole.Config.Fisco.Abi["contract1"]。(这里的"contract1"是一个map的键与settings配置文件中的保持一致) 换做其他的合约换成对应的 键 就好。

### 3.连接FISCOBCOS的网络配置文件读取:

配置fiscobcos链的连接:

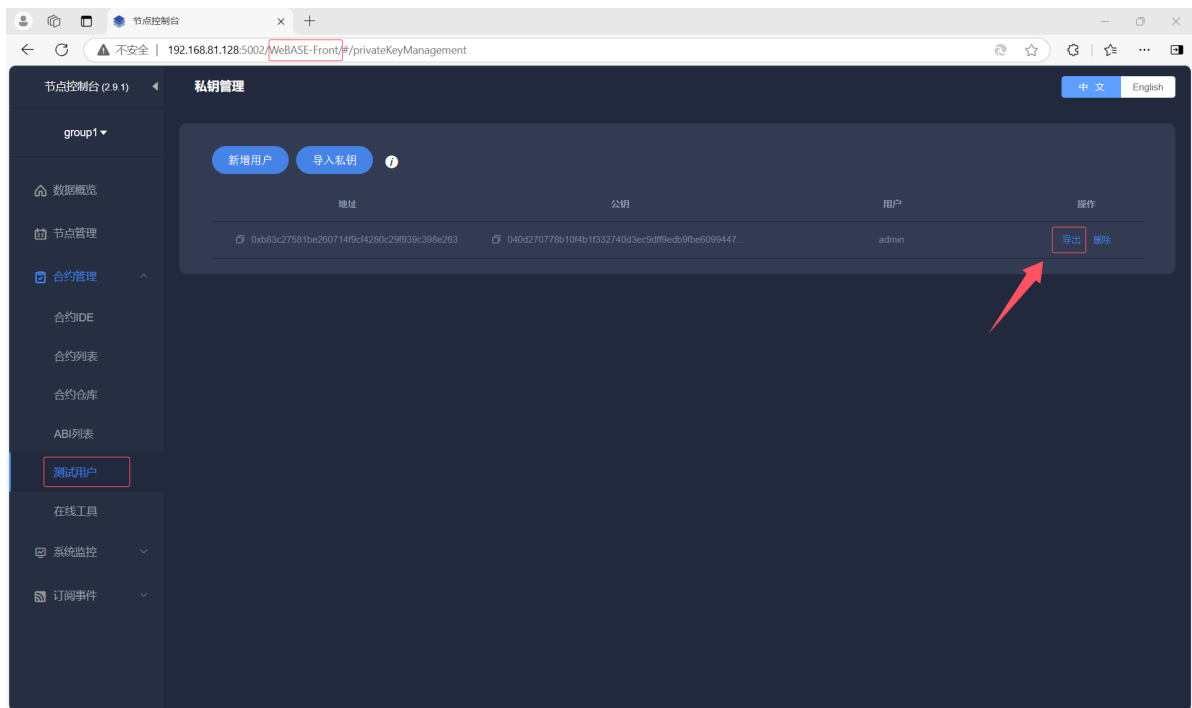
生成私钥:

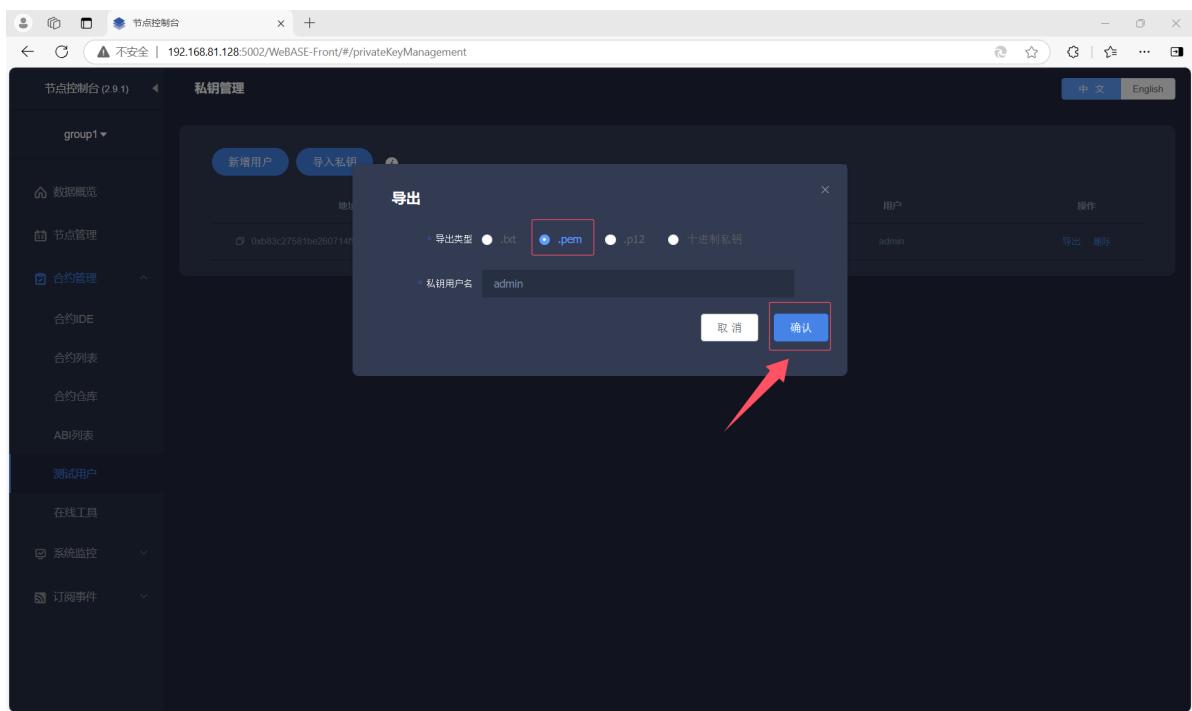




或者从WeBASE-Front上，拿取，下载到项目目录的（fisco-go-sdk-demo/fiscobcos/accounts）中：

下载完之后，记得将私钥文件改个名字（与config.toml中的配置项保持一致）。



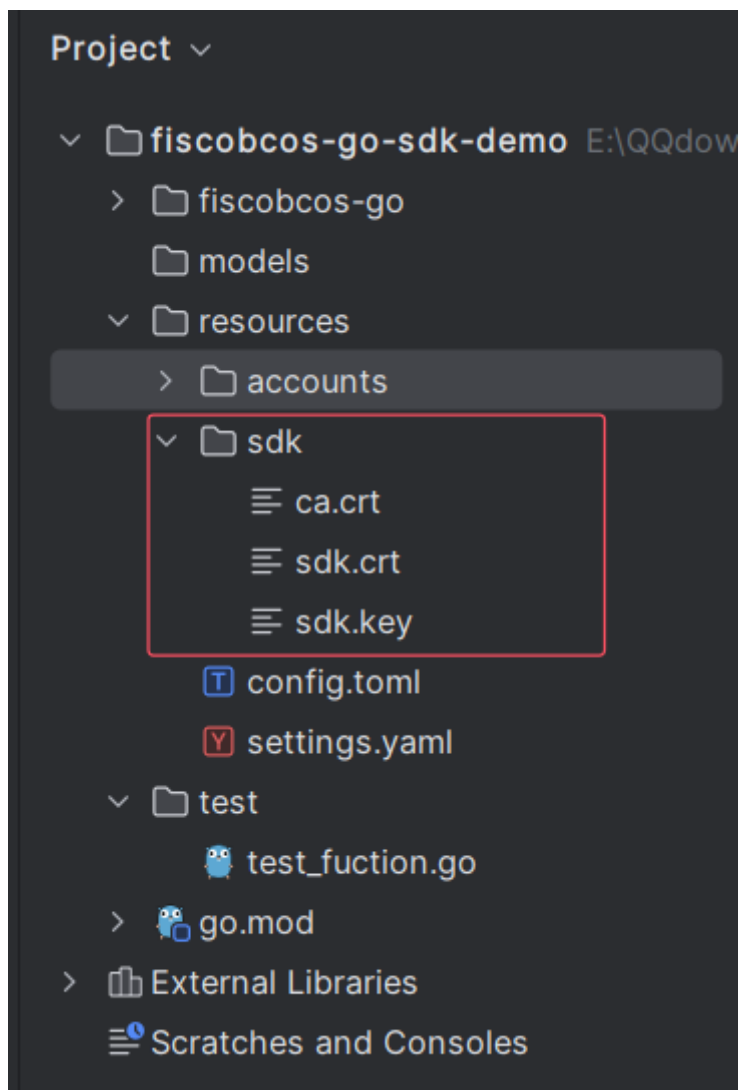


**注意：**

这里导出后下载到自己的项目目录fisco-go-sdk-demo/fiscobcos/accounts) 中。

记得改文件名称（与config.toml中的配置项保持一致）。

拷贝节点证书文件：



例如：

```
root@ubuntu:/home/test/fisco# ls nodes/127.0.0.1/sdk  
ca.crt  sdk.crt  sdk.key
```

网络连接配置文件config.toml（有要修改的部分，认真看图）：

```
[Network]
#type rpc or channel
Type="channel"
# 三个节点证书，使用相对路径
CAFile="resources/sdk/ca.crt"
Cert="resources/sdk/sdk.crt"
Key="resources/sdk/sdk.key"
# if the certificate context is not empty, use it, otherwise read from the
certificate file
# multi lines use triple quotes
CAContext='''
KeyContext='''
CertContext='''

[[Network.Connection]]
NodeURL="192.168.81.128:20200" # 节点的地址
GroupID=1 # 群组id
# [[Network.Connection]]
# NodeURL="127.0.0.1:20200"
# GroupID=2
```

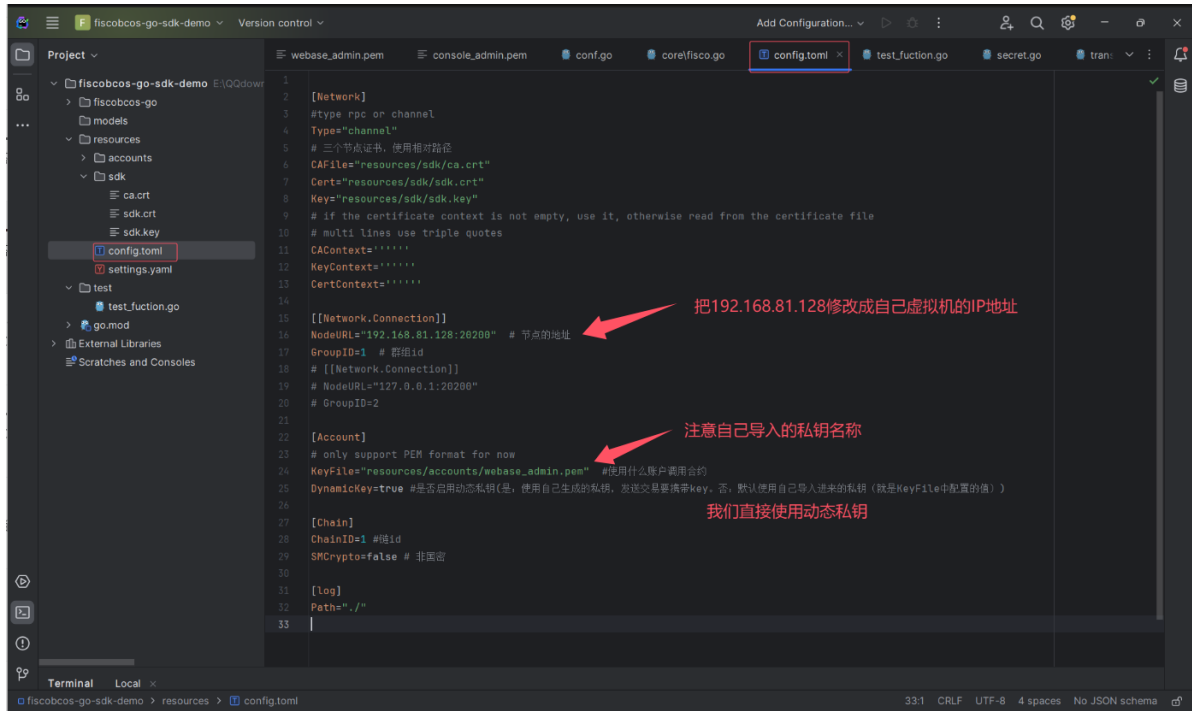
```

[Account]
# only support PEM format for now
KeyFile="resources/accounts/webase_admin.pem" #使用什么账户调用合约
DynamicKey=true #是否启用动态私钥(是: 使用自己生成的私钥, 发送交易要携带key。否: 默认使用自己
导入进来的私钥(就是KeyFile中配置的值))

[Chain]
ChainID=1 #链id
SMCrypto=false # 非国密

[log]
Path="./"

```



注意事项:

启用动态私钥的话, 只能使用携带私钥的发送交易的函数, 如: (`SendTransactionByKey`, `SendCallByKey`)。

不启用动态私钥的话, 只能使用不携带私钥的发送交易的函数, 如: (`SendTransaction`, `SendCall`)。

## 4.HelloWorld的调用 (通过crypto/ecdsa包下的函数生成私钥, 发送交易。)

在main函数中调用WeBASE-Front上的合约:

```

package main

import (
    "crypto/ecdsa"
    "fiscobcos-go-sdk-demo/fiscobcos-go/core"
    "fiscobcos-go-sdk-demo/fiscobcos-go/tools"
    "fmt"
)

```

```

const (
    HelloWorld = "contract1"
)

func main() {
    core.InitConf("resources/settings.yaml")
    core.InitClient("resources/config.toml")
    core.InitSession(HelloWorld)

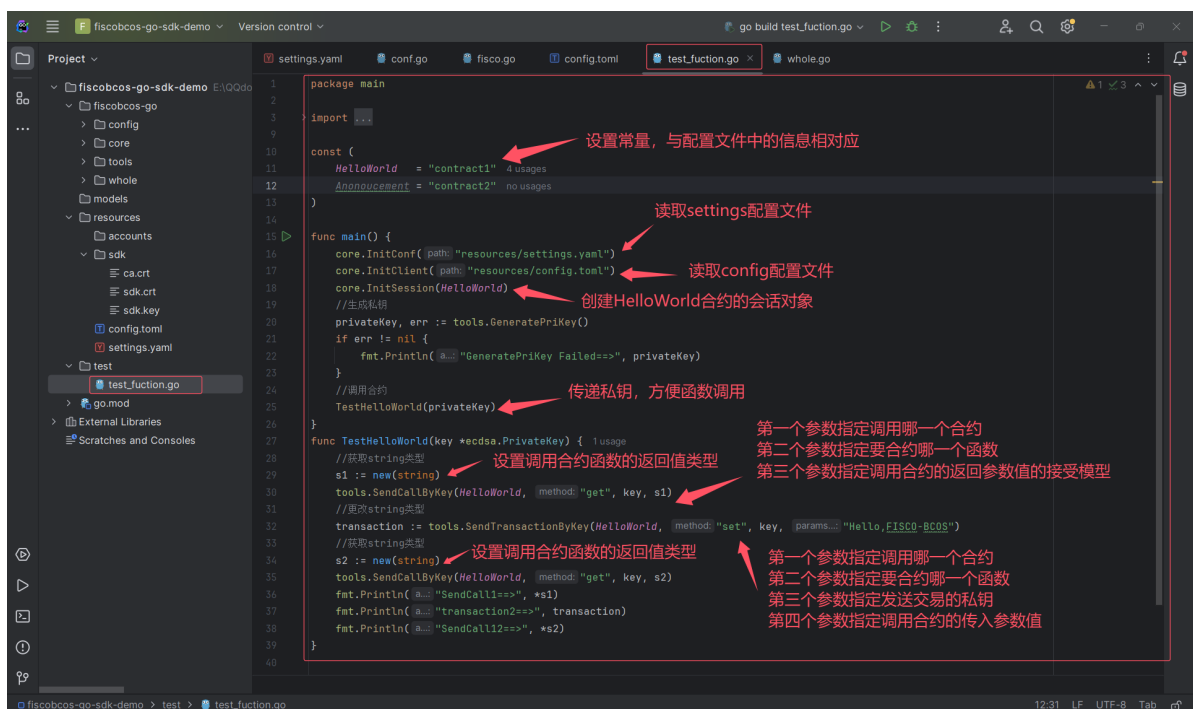
    //生成私钥
    privateKey, err := tools.GeneratePriKey()
    if err != nil {
        fmt.Println("GeneratePriKey Failed==>", privateKey)
    }
    //调用合约
    TestHelloWorld(privateKey)
}

func TestHelloWorld(key *ecdsa.PrivateKey) {
    //获取string类型
    s1 := new(string)
    tools.SendCallByKey(HelloWorld, "get", key, s1)

    //更改string类型
    transaction := tools.SendTransactionByKey(HelloWorld, "set", key,
        "Hello,FISCO-BCOS")

    //获取string类型
    s2 := new(string)
    tools.SendCallByKey(HelloWorld, "get", key, s2)
    fmt.Println("SendCall1==>", *s1)
    fmt.Println("transaction2==>", transaction)
    fmt.Println("SendCall12==>", *s2)
}

```



启动项目：

注意事项：

- (1) 启动程序时，如果报错了，执行go mod tidy。
- (2) 如果还是报错，根据报错信息拉取对应的工具包。

例如：



```
Run go build test_fuction.go x
> E:\Blockchain\Go\go_project\library\pkg\mod\github.com\ffisiclo-lbclols\go-sdk@v1.1.1\crypto\sm3\sm3.go:8:2: missing go.sum entry for module providing package github.com/s
go_get_fiscobcos-go-sdk-demo/fiscobcos-go/tools
E:\Blockchain\Go\go_project\library\pkg\mod\github.com\ffisiclo-lbclols\go-sdk@v1.1.1\abi\abi.go:25:2: missing go.sum entry for module providing package github.com/ethereum/
go_get_fiscobcos-go-sdk-demo/fiscobcos-go/core
E:\Blockchain\Go\go_project\library\pkg\mod\github.com\ffisiclo-lbclols\go-sdk@v1.1.1\abi\numbers.go:24:2: missing go.sum entry for module providing package github.com/ether
go_get_github.com/FISCO-BCOS/go-sdk/abi/v1.1.1
E:\Blockchain\Go\go_project\library\pkg\mod\github.com\ffisiclo-lbclols\go-sdk@v1.1.1\abi\event.go:25:2: missing go.sum entry for module providing package github.com/ethereum
go_get_fiscobcos-go-sdk-demo/fiscobcos-go/tools
E:\Blockchain\Go\go_project\library\pkg\mod\github.com\ffisiclo-lbclols\go-sdk@v1.1.1\core\types\bloom9.go:23:2: missing go.sum entry for module providing package github.com
go_get_fiscobcos-go-sdk-demo/fiscobcos-go/tools
E:\Blockchain\Go\go_project\library\pkg\mod\github.com\ffisiclo-lbclols\go-sdk@v1.1.1\core\types\transaction_signing.go:27:2: missing go.sum entry for module providing packa
go_get_github.com/FISCO-BCOS/go-sdk/core/types@v1.1.1
E:\Blockchain\Go\go_project\library\pkg\mod\github.com\ffisiclo-lbclols\go-sdk@v1.1.1\core\types\log.go:24:2: missing go.sum entry for module providing package github.com/eti
go_get_github.com/FISCO-BCOS/go-sdk/client/v1.1.1
```

调用成功：



```
Run go build test_fuction.go x
GOROOT=D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8 #gosetup
GOPATH=E:\Blockchain\Go\go_project\library;E:\Blockchain\Go\go_project\workspace #gosetup
D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8\bin\go.exe build -o C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe E:\QQdownload\Go\GoWeb\fisco
bcos-go-sdk\fiscobcos-go-sdk-demo\test\test_fuction.go #gosetup
C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe
2024/11/16 22:21:14 config yamlFile load Init success
Client初始化完成
Session初始化完成
Generate PrivateKey: b57b3c6b603f83ee94d1e3ca10f9ebd0b1cd474b573081d38804aa085ae8de
SendCall1==> HelloWorld
transaction2==> true
SendCall12==> Hello,FISCO-BCOS
调用成功
Process finished with the exit code 0
```

## 5. Anonouncement的调用（通过crypto/ecdsa包下的函数生成私钥，发送交易。）

addAnouncement（添加公告）功能测试：

```
package main

import (
    "crypto/ecdsa"
    "fiscobcos-go-sdk-demo/fiscobcos-go/core"
    "fiscobcos-go-sdk-demo/fiscobcos-go/tools"
    "fmt"
    "math/big"
    "time"
)

const (
    HelloWorld    = "contract1"
    Anonouncement = "contract2"
)

func main() {
    core.InitConf("resources/settings.yaml")
```

```

core.InitClient("resources/config.toml")
core.InitSession(Anonouncement)
//生成私钥
privateKey, err := tools.GeneratePriKey()
if err != nil {
    fmt.Println("GeneratePriKey Failed==>", privateKey)
}
//调用合约
TestAnnouncement(privateKey)
}

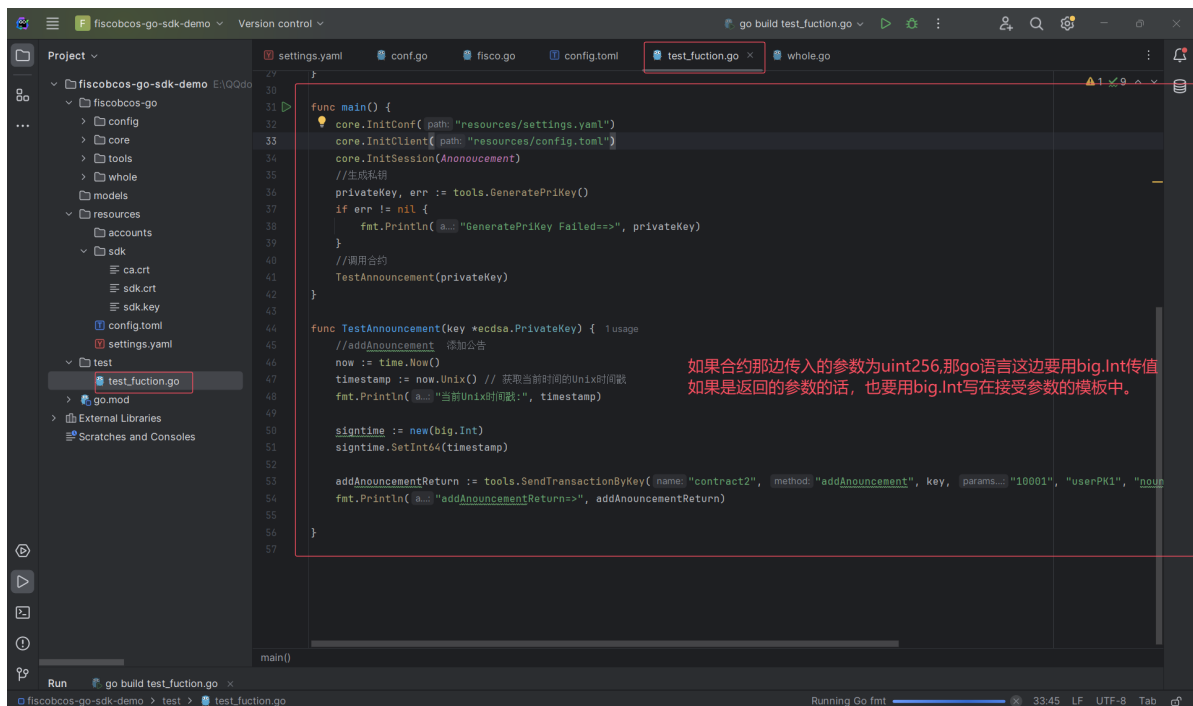
func TestAnnouncement(key *ecdsa.PrivateKey) {
    //addAnnouncement 添加公告
    now := time.Now()
    timestamp := now.Unix() // 获取当前时间的Unix时间戳
    fmt.Println("当前Unix时间戳:", timestamp)

    signtime := new(big.Int)
    signtime.SetInt64(timestamp)

    addAnnouncementReturn := tools.SendTransactionByKey(Anonouncement,
"addAnnouncement", key, "10001", "userPK1", "nounce1", "message1", "cipher1",
"attachment1", signtime)
    fmt.Println("addAnnouncementReturn=>", addAnnouncementReturn)
}

```

下图中的TestAnnouncement中的"contract2"可替换为Anonouncement



执行结果：

```
Run go build test_fuction.go x
GOROOT=D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8 #gosetup
GOPATH=E:\Blockchain\Go\go_project\library:E:\Blockchain\Go\go_project\workspace #gosetup
D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8\bin\go.exe build -o C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe E:\QQdownload\Go\GoWeb\fisco
bcos-go-sdk\fiscobcos-go-sdk-demo\test\test_fuction.go #gosetup
C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe
2024/11/16 22:59:07 config yamlFile load Init success
Client初始化完成
Session初始化完成
Generate PrivateKey: f0eb6f644ae12136ca1db936236389c621ec6e1722f020e9541bc2aa979bd0d45
当前Unix时间戳: 1731769147
addAnouncementReturn=> true 调用成功
Process finished with the exit code 0
```

announcements功能测试:

(1) 创建接受数据的结构体:

```
package contract

import "math/big"

type Output struct {
    Id          *big.Int
    UserID      string
    UserPK      string
    Nounce      string
    Message     string
    Cipher      string
    Attachment  string
    Signtime    *big.Int
}
```

路径: fiscobcos-go-sdk-demo/models/contracts/anonoucement.go



```
1 package contract
2
3 import "math/big"
4
5 type Output struct { 1 usage
6     Id          *big.Int
7     UserID      string
8     UserPK      string
9     Nounce      string
10    Message     string
11    Cipher      string
12    Attachment  string
13    Signtime    *big.Int
14 }
15
```

返回数据为uint256类型，\*big.Int能接受这个数据。所以接受数据的变量类型要为\*big.Int

由于返回值有多个参数，所以可以用结构体去接收。

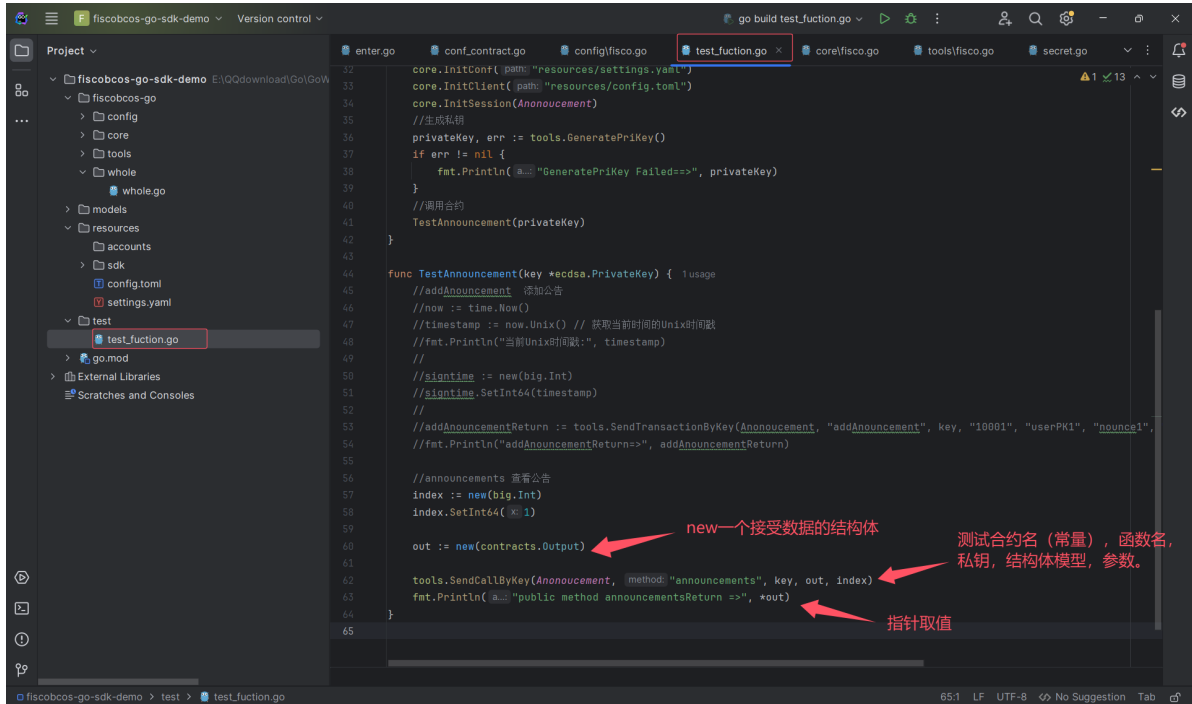
## (2)运行的代码

```
func TestAnnouncement(key *ecdsa.PrivateKey) {
    //addAnnouncement 添加公告
    //now := time.Now()
    //timestamp := now.Unix() // 获取当前时间的Unix时间戳
    //fmt.Println("当前Unix时间戳:", timestamp)
    //
    //signtime := new(big.Int)
    //signtime.SetInt64(timestamp)
    //
    //addAnnouncementReturn := tools.SendTransactionByKey(Anonouncement,
    "addAnnouncement", key, "10001", "userPK1", "nounce1", "message1", "cipher1",
    "attachment1", signtime)
    //fmt.Println("addAnnouncementReturn=>", addAnnouncementReturn)

    //announcements 查看公告
    index := new(big.Int)
    index.SetInt64(1)

    out := new(contracts.Output)
```

```
tools.SendCallByKey(Anonouncement, "announcements", key, out, index)
fmt.Println("public method announcementsReturn =>", *out)
}
```



listAnouncement功能测试（在进行测试时，测试人员使用addAnouncement函数，上传了多个公告）：

(1) 不用查询条件进行查询（返回所有数据）：

```
func TestAnnouncement(key *ecdsa.PrivateKey) {
    //addAnouncement 添加公告
    //now := time.Now()
    //timestamp := now.Unix() // 获取当前时间的Unix时间戳
    //fmt.Println("当前Unix时间戳:", timestamp)
    //
    //signtime := new(big.Int)
    //signtime.SetInt64(timestamp)
    //
    //addAnouncementReturn := tools.SendTransactionByKey(Anonouncement,
    "addAnouncement", key, "10001", "userPK1", "nounce1", "message1", "cipher1",
    "attachment1", signtime)
    //fmt.Println("addAnouncementReturn=>", addAnouncementReturn)

    //announcements 查看公告
    //index := new(big.Int)
```

```

//index.SetInt64(1)
//
//out := new(contracts.Output)
//
//tools.SendCallByKey(Anonouncement, "announcements", key, out, index)
//fmt.Println("public method announcementsReturn =>", *out)

//listAnouncement (分页获取数据)
ans := new([]contracts.Output)
byUserID := false //是否使用UserID, 来进行查询
userID := "10005"

byTimeRange := false //是否使用时间范围来查询
startTime := new(big.Int)
startTime.SetInt64(0) //设置起始时间

end_now := time.Now()
end_timestamp := end_now.Unix() // 获取当前时间的Unix时间戳
fmt.Println("当前Unix时间戳:", end_timestamp)

endTime := new(big.Int)
endTime.SetInt64(end_timestamp) //设置结束时间
pageSize := new(big.Int) //每页大小
pageSize.SetInt64(10)
startIndex := new(big.Int) //起始索引
startIndex.SetInt64(0)

tools.SendCallByKey(Anonouncement, "listAnouncement", key, ans, byUserID,
userID, byTimeRange, startTime, endTime, pageSize, startIndex)
fmt.Println("listAnouncementReturn Length =>", len(*ans))
fmt.Println("listAnouncementReturn =>", *ans)
}

```

The screenshot shows the IDE interface with the file `test_fuction.go` open. The code is a Go test function that calls `tools.SendCallByKey` to test the `listAnouncement` function. The code is annotated with Chinese comments explaining the parameters and the expected return type.

Red boxes highlight the following sections:

- Line 66-70:** `ans := new([]contracts.Output)`. Comment: 智能合约函数返回类型为一个结构体数组, 所以创建一个结构体数组来接收数据 (The smart contract function returns a type of a struct array, so create a struct array to receive the data).
- Line 71-73:** `byUserID := false`. Comment: 第一次测试, 不使用查询条件, 进行查询 (First test, do not use query conditions, perform query).
- Line 87-88:** `fmt.Println("listAnouncementReturn Length =>", len(*ans))` and `fmt.Println("listAnouncementReturn =>", *ans)`. Comment: 获取数组长度和数组存储的值 (Get array length and array stored value).

执行结果:

```
Run go build test_fuction.go x

GOROOT=D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8 #gosetup
GOPATH=E:\Blockchain\Go\go_project\library;E:\Blockchain\Go\go_project\workspace #gosetup
D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8\bin\go.exe build -o C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe E:\Qqdownload\Go\GoWeb\fisco
bcos-go-sdk\fiscobcos-go-sdk-demo\test\test_fuction.go #gosetup
C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe

2024/11/17 14:44:41 config yamlFile load Init success
Client初始化完成
Session初始化完成
Generate PrivateKey: 3493752ff9def81472da0eed2d8a57ea7bacf2f9c344edfed174e1da980d6cc0 调用成功
当前Unix时间戳: 1731825881
listAnnouncementReturn Length => 10
listAnnouncementReturn => [{1 10001 userPK1 nounce1 message1 cipher1 attachment1 1731564456} {2 10002 userPK2 nounce2 message2 cipher2 attachment2 1731566505} {3 10003 userPK3 nounce3 message3 cipher3 attachment3 1731566540} {4 10004 userPK4 nounce4 message4 cipher4 attachment4 1731566594} {5 10005 userPK5 nounce5 message5 cipher5 attachment5 1731566622} {6 10006 userPK6 nounce6 message6 cipher6 attachment6 1731566651} {7 10006 userPK6 nounce6 message6 cipher6 attachment6 1731574620} {8 10006 userPK6 nounce6 message6 cipher6 attachment6 1731586311} {9 10006 userPK6 nounce6 message6 cipher6 attachment6 1731586353} {10 10006 userPK6 nounce6 message6 cipher6 attachment6 1731586621}]

Process finished with the exit code 0
```

(2) 使用UserID进行查询:

注意事项: 将byUserID的值赋值为true

```
Run go build test_fuction.go x

GOROOT=D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8 #gosetup
GOPATH=E:\Blockchain\Go\go_project\library;E:\Blockchain\Go\go_project\workspace #gosetup
D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8\bin\go.exe build -o C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe E:\Qqdownload\Go\GoWeb\fisco
bcos-go-sdk\fiscobcos-go-sdk-demo\test\test_fuction.go #gosetup
C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe

2024/11/17 14:46:12 config yamlFile load Init success
Client初始化完成
Session初始化完成
Generate PrivateKey: f10c2cc2fc07937824d8cee4576cac6dc512e17b6aa91351170e1c8aeeaa64b3d 调用成功
当前Unix时间戳: 1731825972
listAnnouncementReturn Length => 1
listAnnouncementReturn => [{5 10005 userPK5 nounce5 message5 cipher5 attachment5 1731566622}]

Process finished with the exit code 0
```

(3) 使用startTime,endTime进行查询:

注意事项:

将byTimeRange的值赋值为true, 将byUserID的值赋值为false

endTime (终止时间) 的设置要注意

```
Run go build test_fuction.go x

GOROOT=D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8 #gosetup
GOPATH=E:\Blockchain\Go\go_project\library;E:\Blockchain\Go\go_project\workspace #gosetup
D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8\bin\go.exe build -o C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe E:\Qqdownload\Go\GoWeb\fisco
bcos-go-sdk\fiscobcos-go-sdk-demo\test\test_fuction.go #gosetup
C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\__go_build_test_fuction.go.exe

2024/11/17 14:47:28 config yamlFile load Init success
Client初始化完成
Session初始化完成
Generate PrivateKey: 5896c9b47f05fc4a4f45a83748e45cc514db4c459ec7b65c09641e39921080fb 调用成功
当前Unix时间戳: 1731826048
listAnnouncementReturn Length => 10
listAnnouncementReturn => [{1 10001 userPK1 nounce1 message1 cipher1 attachment1 1731564456} {2 10002 userPK2 nounce2 message2 cipher2 attachment2 1731566505} {3 10003 userPK3 nounce3 message3 cipher3 attachment3 1731566540} {4 10004 userPK4 nounce4 message4 cipher4 attachment4 1731566594} {5 10005 userPK5 nounce5 message5 cipher5 attachment5 1731566622} {6 10006 userPK6 nounce6 message6 cipher6 attachment6 1731566651} {7 10006 userPK6 nounce6 message6 cipher6 attachment6 1731574620} {8 10006 userPK6 nounce6 message6 cipher6 attachment6 1731586311} {9 10006 userPK6 nounce6 message6 cipher6 attachment6 1731586353} {10 10006 userPK6 nounce6 message6 cipher6 attachment6 1731586621}]

Process finished with the exit code 0
```

(4) 使用UserID和startTime,endTime进行查询:

注意事项:

(1) 将byTimeRange和byUserID的值赋值为true

(2) 会根据UserID和startTime,endTime进行查询, 把所有符合条件的值返回

```
Run go build test_fuction.go x
GOROOT=D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8 #gosetup
GOPATH=E:\Blockchain\Go\go_project\library;E:\Blockchain\Go\go_project\workspace #gosetup
D:\Tool\GoTool\GoSDK_1.22.8\go1.22.8\bin\go.exe build -o C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\___go_build_test_fuction.go.exe E:\QQdownload\Go\GoWeb\fisco
bcos-go-sdk\fiscobcos-go-sdk-demo\test\test_fuction.go #gosetup
C:\Users\20405\AppData\Local\JetBrains\GoLand2023.2\tmp\GoLand\___go_build_test_fuction.go.exe
2024/11/17 14:49:10 config yamlFile load Init success
Client初始化完成
Session初始化完成
Generate PrivateKey: 2943fe81ed91ae8455a5bd575506e59df4741d05035d6e274d19a380b1c23798
当前Unix时间戳: 1731826150
listAnnouncementReturn Length => 1
listAnnouncementReturn => [{5 10005 userPKS nounceS messageS cipherS attachmentS 1731566622}]

Process finished with the exit code 0
```

调用成功

## 6.通过WeBASE导出的私钥发送交易

注意事项：

代码中不展示具体的调用函数过程，但会通过私钥计算出公钥和地址，与webase上导出的文件作对比

具体函数调用过程，只需要通过将各类型的私钥，转换成\*ecdsa.PrivateKey类型的私钥，就可以调用SendCallByKey和SendTransactionByKey(将私钥作为参数传入就行)。

详细内容见FISCOBCOS中的GoSDK操作---(搭建项目版本)，注意由于两个文档的目录结构不同，要注意函数在哪个包下。

## 三，小结：

合约中传入和返回参数	Go语言中使用对应类型
string	string或者*string
uint / uint256	*big.Int
uint8	*uint8
address	*common.address(用这个包： <a href="https://github.com/ethereum/go-ethereum/blob/master/common/address.go">github.com/ethereum/go-ethereum/common</a> )
bool	*bool
uint[] / uint256[]	[]*big.Int
string[]	[]*string
结构体（例如：User）	*User
结构体数组（例如：User[]）	[]*User

注意事项：

go语言中声明传入、传出的参数的模型时，可以直接：new一个类型 赋值给 变量

例如：num:= new(big.Int)

