测试链码

确定进入CLI容器中

```
$ sudo docker exec -it cli bash
```

Peer加入应用通道后, 可以执行链码相关操作,进行测试

链码在调用之前, 必须先经过安装和实例化两个步骤, 部署到Peer节点上.

检查环境变量是否正确设置

```
echo $CHANNEL_NAME
```

设置环境变量

```
export CHANNEL_NAME=mychannel
```

链码使用之前必须:

1. 将其安装在指定的节点上
2. 安装完成后要对其进行实例化
3. 调用链码(查询, 执行事务)

安装并实例化Chaincode

安装:

```
peer chaincode install -n mycc -v 1.0 -p
github.com/chaincode/chaincode_example02/go/
```

参数说明:

-n: 指定要安装的链码的名称

-v: 指定链码的版本

-p: 指定要安装的链码的所在路径

实例化:

```
peer chaincode instantiate -o orderer.example.com:7050 --tls --
cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/t
lsca.example.com-cert.pem -C $CHANNEL_NAME -n mycc -v 1.0 -c
'{"Args":["init","a", "100", "b","200"]}' -P "OR
('Org1MSP.peer','Org2MSP.peer')"
```

实例化完成后, 用户即可向网络中发起交易

参数说明:

-o: 指定Oderer节点地址

--tls: 开启TLS验证

--cafile: 指定TLS_CA证书路径

-n: 指定要实例化的链码名称

-v:指定要实例化的链码的版本号

-C: 指定通道名称

-c: 实例化链码时指定的参数

-P: 指定背书策略

查询

```
peer chaincode query -C $CHANNEL_NAME -n mycc -c '{"Args":
["query","a"]}'
```

输出结果:`Query Result: 100`

参数说明:

-n: 指定要调用的链码名称

-C: 指定通道名称

-c: 指定调用链码时所需要的参数

func query(account string){

}

账本保存数据以 key-value方式保存

调用/转账

```
peer chaincode invoke -o orderer.example.com:7050  --tls --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrg
anizations/example.com/orderers/orderer.example.com/msp/tlscacerts/t
lsca.example.com-cert.pem  -C $CHANNEL_NAME -n mycc -c '{"Args":
["invoke","a","b","10"]}'
```

参数说明:

-o: 指定orderer节点地址

--tls: 开启TLS验证

--cafile: 指定TLS_CA证书路径

-n: 指定链码名称

-C: 指定通道名称

-c: 指定调用链码的所需参数

func invoke(accoutF string, accountT stirng, amount string)

查询a账户的金额

```
peer chaincode query -C $CHANNEL_NAME -n mycc -c '{"Args":
["query","a"]}'
```

输出结果: `Query Result: 90`

练习

将github.com/chaincode/sacc/的链码安装并实例化,

查询a账户的余额

使用set方法设置a账户的余额

使用get方法获取a账户的余额

实例化时只需要两个参数, 不需要指定操作名称

两个参数: 账户名, 金额

set

账户名, 金额

get

账户名, 金额