**实验11 编写Fabric链码**

**【实验介绍】**

现有一个业务场景，多个商户之间组织成一个联盟，使用Fabric进行简单的存证。其中，他们需要存证的内容包括账单数据，这里，从简单出发，每一条账单数据的格式为< id, amount >，id为一个具有独⼀性的标识字段，用于区分不同的账单；amount则为金额。简单定义了数据存储模型，那么还有数据操作，这里他们会有账单发布、账单查询，以及简单的账单统计需求。

这个联盟找到你，让你根据上述场景完成一个链码，以满足他们的应用需求。

关于数据类型，其中只涉及两个字段，id与amount，其中，amount使用int类型即可，而id字段仅仅用于唯一性标识，你可以简单的使用不冲突的数字或者字母即可。

你可以继续使用之前的e2e案例中的网络架构进行实验，但是你需要替换其中的链码以及脚本文件 script. sh中的调⽤参数以及查询参数，根据你自己的链码实现细节进行相应修改；你也可以不使用e2e 中的网络架构，自己手动进行网络部署(e2e\_cli的部署配置文件是docker-compose-cli.yaml。你可以参照这个来部署)。

Fabric网络节点：

1) 客户端节点：代表用户实体，通过其与Peer节点和orderer排序节点交互，完成交易。

2) Peer节点： Peer节点主要就是为了维护和更新区块链网络状态，它可以根据更细化的职能分为背书节点(Endorser)，模拟执行客户端发来的交易，并返回背书响应；记账节点(Committer)，对排序服务确定的新的共识块进行验证并记入账本数据库中；锚节点(Anchor)，与通道内其他成员通信的节点，通道内的每个组织都需要Anchor节点；主节点(Leader)，组织内代表组织与排序服务节点通信的节点。

3) Orderer排序节点：负责接收客户端发来的背书签名的交易，对一组未认证过的交易进行打包共识生成新的区块，将其广播给Peer节点。e2e案例中的zookeeper和kafka都是配合orderer节点做共识服务的。

4) CA节点：证书颁发节点，因为fabric是联盟链，并不是说完全的不可信网络，所以需要证书来验证某个组织或者节点的身份是否可信，CA节点就负责提供这种服务，用于身份确认和签名等流程中。

可以根据个人需求，拉取相应的节点镜像来配置网络。

**【实验要求】**

需要你实现的功能点主要包括：

（1）存储账单数据< id, amount > 根据id查询某一条账单的金额；

（2）当前的账单总额；

（3）金额最大的一笔账单的信息（比如返回其id与amount）；

（4）金额最小的一笔账单的信息（比如返回其id与amount）；

（5）上述功能点根据实现与否按点给分。自己的想法进行额外的功能点添加，按点加分，根据其复杂度、难度、创新等酌情加分。

这是一道开放性的题目，仅说明需求，具体代码实现方案up to you；关于链码的编写，你可以使用Java，也可以使用Go（推荐感兴趣的同学使用Go，并不会涉及太多语法特性，了解一下语法就能用了；Fabric的Java链码容易出现一些莫名的bug，比如上一个实验中，完成链码部署后，在cli容器中进行手动调用时会出现调用成功但是数据没发生改变的情况）。

如果选择使用Go进行链码编写，那你可以继续使用之前1.1版本的Fabric进行部署和测试；如果选择使用Java进行链码编写，那请使用后来的1.3版本Fabric进行部署和测试.关于Go的上手，这里推荐两个入门教程Go语言入门教程、Go 语言教程。

建议在代码中添加针对性的注释。

**【实验准备】**

在原有的fabric环境下进行修改即可。

**【实验过程】**

本实现是在e2e案例的基础上修改完成的，主要修改的有两个部分：

一是链码，地址为：hyperledger/fabric/examples/chaincode/go/chaincode\_example02

二是脚本，地址为：hyperledger/fabric/examples/e2e\_cli/scripts/script.sh

本实现主要是通过提供的接口ChaincodeStubInterface的PutState和GetState函数来实现记录账户数据的功能，且对于Sum，Max，Min这类特殊值也采用该函数，所以取id时，不能取这些特殊值，负责会影响正确结果,仅供参考。

相关函数：

GetFunctionAndParameters() (string, []string) 将字符串数组的参数分为两部分，数组第一个字是Function，剩下的都是Parameter

PutState(key string, value []byte) error 增改数据

GetState(key string) ([]byte, error) 查询数据

**1 chaincode\_example02**

init():初始化，添加一个账单，并且更新对应的Sum，Max和Min，init函数会在脚本中运行，如图11-1所示。

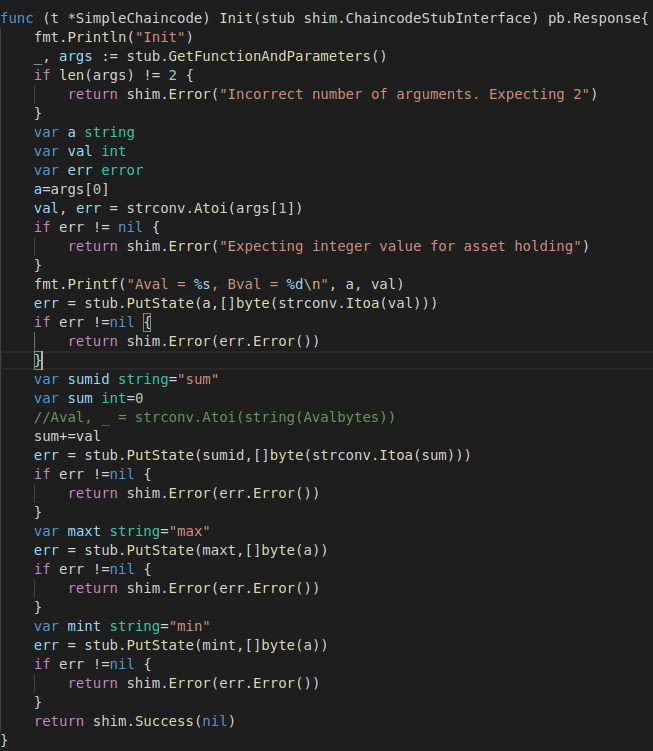


图11-1 Init()

Invoke():用一个统一的Invoke函数来调用所有功能，如图11-2所示。

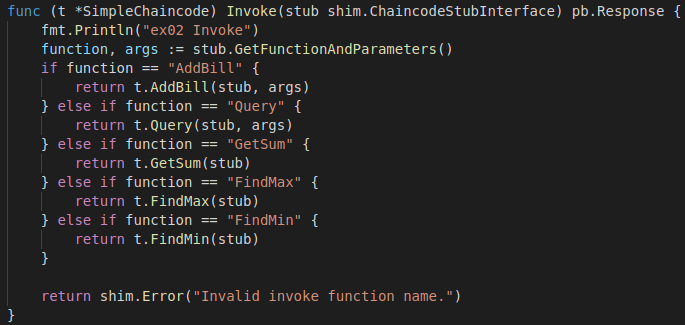


图7-12 Iovoke()

AddBill():添加账单，与已有的Max和Min账单进行比较并更新，更新Sum，如图7-13所示。

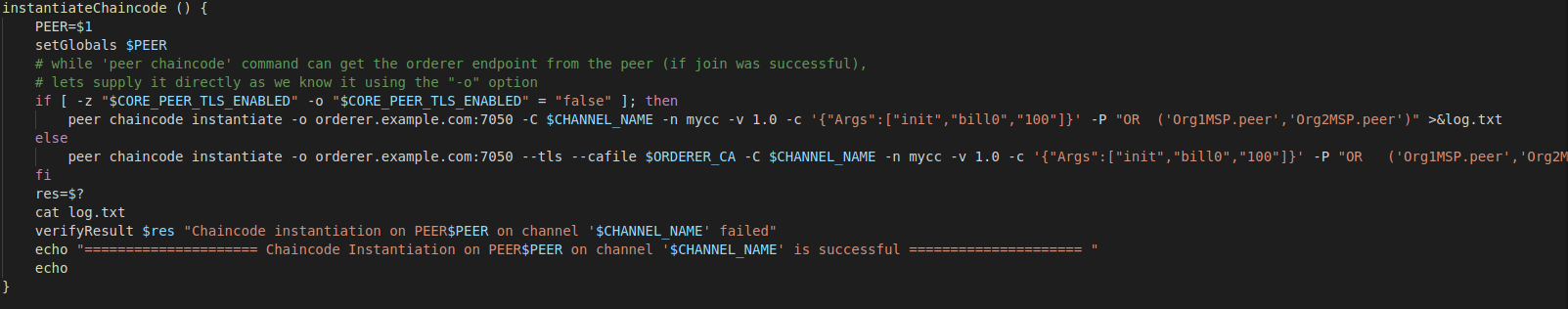


图7-13 AddBill()

Query(), GetSum(), FindMax(), FindMin()和e2e\_cli中原来提供的代码基本 一致，输出可以按自己想要的格式来改，就不贴代码了。

**2 script.sh**

这里修改很简单，你想让你的案例一开始运行时做什么操作就做什么操作，可以从原来的script.sh的最后看出有哪些操作，我这里就简单的初始化了，其他的操作都是后面自己通过命令来操作，如图7-14所示。

图7-14 脚本中初始化函数instantiateChaincode()

改成你自己写的init函数里面指定的参数格式就行，这里可以看出我们初始化了一个账单bill0，其金额为100。

**3 执行过程**

启动:

|  |
| --- |
| ./network\_setup.sh up |

进入cli客户端：

|  |
| --- |
| docker exec -it cli bash |

查看账户bill0的资产：

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -c '{"Args":["Query","bill0"]}' |

2022-05-25 10-42-27 的屏幕截图

图7-15

和我们初始化的bill0账单的金额相同

插入一条新账单<bill1,101>

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -c '{"Args":["AddBill","bill1","101"]}' |

2022-05-25 10-41-49 的屏幕截图

图7-16

查看金额总和:

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -c '{"Args":["GetSum"]}' |

2022-05-25 10-42-07 的屏幕截图

图7-17

查看金额最大的账单:

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -c '{"Args":["FindMax"]}' |

2022-05-25 10-42-07 的屏幕截图

图7-18

查看金额最小的账单:

|  |
| --- |
| peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -c '{"Args":["FindMin"]}' |

2022-05-25 10-42-27 的屏幕截图

图7-19

**【实验小结】**

本实验设计了一个实际的业务场景，让大家尝试对fabric的e2e业务案例的实现代码进行修改，使其符合新的需求。通过对业务逻辑链码的修改和其联盟网络配置情况的查看和部署以及前几个相关实验中对于fabric框架的介绍，帮助大家可以简单地实现链码的编写和部署，让大家对于fabric如何实现基于许可的区块链网络和相应的业务逻辑的实现过程有一个大致的了解。

**【习题】**

1. 简单地说明一下客户端申请调用链码函数进行后fabric的运行流程。

2. 你可以根据自己的想法进行额外的功能点添加，按点加分，根据其复杂度、难度、创新等酌情加分。

**【参考文献】**

1. 邹均，张海宁，唐屹，李磊等. 区块链技术指南. 机械工业出版社
2. 冯翔. 区块链开发实战: Hyperledger Fabric 关键技术与案例分析 Key technology and case analysis for Hyperledger Fabric[M]. 机械工业出版社, 2018.
3. Hyperledger Community. Hyperledger Fabric(OL). (2020-06-01) [2020-07-12]. <https://github.com/hyperledger/fabric>.
4. Hyperledger Community. Smart Contracts and Chaincode(OL). (2020-08-20) [2020-12-11]. <https://github.com/hyperledger/fabric-docs-i18n/blob/release-2.2/docs/locale/es/source/smartcontract/smartcontract.md>.