005链码 API 介绍

> 孔壹学院：国内区块链职业教育领先品牌
> 官方网址：http://www.kongyixueyuan.com/

网址

https://godoc.org/github.com/hyperledger/fabric/core/chaincode/shim

参数读取API

- GetFunctionAndParameters 提取调用链码交易中的参数，其中第一个作为被调用的函数名称，剩下的参数作为函数的执行参数

`func (stub *ChaincodeStub) GetFunctionAndParameters() (function string, params []string)`

```
# {"Args":["set","tom","100"]}
fn, args := stub.GetFunctionAndParameters()
fmt.Println(fn, args)

# 输出结果
set ["tom", "100"]
```

- GetStringArgs 提取链码交易的指定参数

`func (stub *ChaincodeStub) GetStringArgs() []string`

```
# {"Args":["set","tom","100"]}

args = stub.GetStringArgs()
fmt.Println(args)

# 输出结果
["set","tom","100"]
```

账本状态交互API

- PutState 在账本中添加或更新一对键值。

`func (stub *ChaincodeStub) PutState(key string, value []byte) error`

```
    err := stub.PutState("str",[]byte("hello"))
    if err != nil {
        fmt.Println("str PutState error: "+err.Error())
    }else{
        fmt.Println("str PutState success!")
    }
```

- GetState 负责查询账本，返回指定键的对应值

`func (stub *ChaincodeStub) GetState(key string) ([]byte, error)`

```
    strValue , err := stub.GetState("str")
    if err != nil {
        fmt.Println("str GetState error: "+err.Error())
    }else {
        fmt.Printf("str value: %s \n",string(strValue))
    }
    # 输出结果
    str value: hello
```

- DelState 删除一对键值

`func (stub *ChaincodeStub) DelState(key string) error`

```
    err = stub.DelState("str")
```

- GetStateByRange 查询指定范围内的键值，startKey为起始key，endKey为终止key

`func (stub *ChaincodeStub) GetStateByRange(startKey, endKey string) (StateQueryIteratorInterface, error)`

```
    err := stub.PutState("str",[]byte("hello"))
    err = stub.PutState("str1",[]byte("hello1"))
    err = stub.PutState("str2",[]byte("hello2"))
    resultIterator , err := stub.GetStateByRange("str" , "str2")

    defer resultIterator.Close()
    fmt.Println("-----start resultIterator-----")
    for resultIterator.HasNext() {
        item, _ := resultIterator.Next()
        fmt.Println(string(item.Value))
    }
    fmt.Println("-----end resultIterator-----")

    # 运行结果
    -----start resultIterator-----
    hello
    hello1
    -----end resultIterator-----
```

- GetHistoryForKey 返回某个键的历史记录

`func (stub *ChaincodeStub) GetHistoryForKey(key string) (HistoryQueryIteratorInterface, error)`

```
    historyIterator,err := stub.GetHistoryForKey("str")
    defer historyIterator.Close()
    fmt.Println("-----start historyIterator-----")
    for resultIterator.HasNext() {
        item, _ := historyIterator.Next()
        fmt.Println(string(item.TxId))
        fmt.Println(string(item.Value))
    }
    fmt.Println("-----end historyIterator-----")
```

其他API

- CreateCompositeKey 给定一组属性，将这些属性组合起来构造一个复合键

`func (stub *ChaincodeStub) CreateCompositeKey(objectType string, attributes []string) (string, error)`

```
    indexName := "sex~name"
    indexKey , err := stub.CreateCompositeKey(indexName,[]string{"boy","xiao
wang"})

    value := []byte{0x00}
    stub.PutState(indexKey,value)
    fmt.Println(indexKey)
    indexKey , err = stub.CreateCompositeKey(indexName,[]string{"boy","xiaol
i"})
    stub.PutState(indexKey,value)
    fmt.Println(indexKey)
    indexKey , err = stub.CreateCompositeKey(indexName,[]string{"girl","xiao
fang"})
    fmt.Println(indexKey)
    stub.PutState(indexKey,value)

    # 运行结果
    sex~nameboyxiaowang
    sex~nameboyxiaoli
    sex~namegirlxiaofang
```

- SplitCompositeKey 给定一个复合键，将其拆分为复合键所用的属性

`func (stub *ChaincodeStub) SplitCompositeKey(compositeKey string) (string, []string, error)`

- GetStateByPartialCompositeKey 根据局部的复合键返回所有的匹配的键值

`func (stub *ChaincodeStub) GetStateByPartialCompositeKey(objectType string, attributes []string) (StateQueryIteratorInterface, error)`

```

    resultIterator,err = stub.GetStateByPartialCompositeKey(indexName, []str
ing{"boy"})
    defer resultIterator.Close()
    fmt.Println("-----start resultIterator-----")
    for resultIterator.HasNext() {
        item, _ := resultIterator.Next()

        objectType, compositeKeyParts, err := stub.SplitCompositeKey(item.Ke
y)
        if err != nil {
            return shim.Error(err.Error())
        }
        fmt.Println("objectType: "+objectType)
        fmt.Println("sex : "+compositeKeyParts[0])
        fmt.Println("name : "+compositeKeyParts[1])

    }
    fmt.Println("-----end resultIterator-----")
    # 运行结果
    -----start resultIterator-----
    objectType: sex~name
    sex : boy
    name : xiaoli
    objectType: sex~name
    sex : boy
    name : xiaowang
    -----end resultIterator----
```

- GetQueryResult 对状态数据库进行富查询，仅有couchDB支持

`func (stub *ChaincodeStub) GetQueryResult(query string) (StateQueryIteratorInterface, error)`

```
    resultIterator , err = stub.GetQueryResult("{\"selector\": {\"sex\": \"boy\"}}" )

    defer resultIterator.Close()
    fmt.Println("-----start resultIterator-----")
    for resultIterator.HasNext() {
        item, _ := resultIterator.Next()
        fmt.Println(string(item.Value))
    }
    fmt.Println("-----end resultIterator-----")
```

- InvokeChaincode 调用另一个链码中的Invoke方法

`func (stub *ChaincodeStub) InvokeChaincode(chaincodeName string, args [][]byte, channel string) pb.Response`

```
    # chaincode_example02 中 a向b 转账
    trans:=[][]byte{[]byte("invoke"),[]byte("a"),[]byte("b"),[]byte("11")}
    stub.InvokeChaincode("mycc",trans,"mychannel")
```