< 하이퍼레져 페브릭6 >

1. marble_chaincode_private.go 추가 작성 후 업그레이드 하기

Q) marble03_private 디렉토리를 만들어 추가된 CC 작성 후 upgrade하기

① .json 파일 수정하기(collectionMarbleStatus 추가 / "Org2"만 가능하게끔)

```
{
    "name": "collectionMarbleStatus",
    "policy": "OR('Org2MSP.member')",
    "requiredPeerCount": 0,
    "maxPeerCount": 3,
    "blockToLive":1000000
}
```

② chaincode 수정

- struct 추가(marbleStatusPDC) : Status값은 'stock', 'sold out', 'ready' 등등 값이 가능

```
type marbleStatusPDC struct {
    ObjectType string `json:"docType"`
    Name       string `json:"name"`
    Status     string `json:"status"`
}
```

- Invoke 함수에 case문 추가 후 "checkStatus" 함수 구현(func : readMarbles 그대로 복붙)

```
case "checkStatus":
    //check a marble status
    return t.checkStatus(stub, args)
```

: 말 그대로 readStatus하는 것(입력받은 값을 출력해주는 역할 할 수 있도록 함수 구현)

- initMarble 함수에 초기 값 하나 더 추가(5-price)

```
func (t *SimpleChaincode) initMarble(stub shim.ChaincodeStubInterface, args []string) pb.Response {
    var err error

    //  0-name  1-color  2-size  3-owner  4-price  5-price
    // "asdf",  "blue",  "35",   "bob",   "99",    "stack"
    if len(args) != 6 {
        return shim.Error("Incorrect number of arguments. Expecting 6")
    }
```

: 즉, 하나 값을 더 받았기 때문에 한 번 더 쪼개줄 것(.json 이용 PutStatus를 만들어주기)

A) 위에서 한 내용에 따라 코드 수정 후 업데이트!

① Invoke에서 select~case 문 작성한 것을 바탕으로 checkStatus라는 함수 생성

```
func (t *SimpleChaincode) checkStatus(stub shim.ChaincodeStubInterface, args []string) pb.Response {
    var name, jsonResp string
    var err error

    if len(args) != 1 {
        return shim.Error("Incorrect number of arguments. Expecting name of the marble to query")
    }

    name = args[0]
    valAsbytes, err := stub.GetPrivateData("collectionMarbleStatus", name) //get the marble from chaincode state
    if err != nil {
        jsonResp = "{\"Error\":\"Failed to get state for " + name + "\"}"
        return shim.Error(jsonResp)
    } else if valAsbytes == nil {
        jsonResp = "{\"Error\":\"Marble does not exist: " + name + "\"}"
        return shim.Error(jsonResp)
    }

    return shim.Success(valAsbytes)
}
```

: json에서 작성한 collectionMarbleStatus 함수를 가져와서 그 이름(get)에 해당하는 값을 반환(=status)
: 에러가 발생할 경우(null이 아닐 경우) status 받아오는데 문제 생긴 것
: 가져오려는 값(status)인 valAsbytes가 null일 경우 marble이 존재하지 않는 것
: 둘 다 이상이 없으면 response로 valAsbytes(=status) 값을 그대로 출력!!

② initMarble()에서 데이터 하나 추가해 총 6개 값을 받도록 했기 때문에 그 안에 내용 수정

```
if len(args[5]) == 0 {
    return shim.Error("6th argument must be a non-empty string")
}
marbleName := args[0]
color := strings.ToLower(args[1])
owner := strings.ToLower(args[3])
size, err := strconv.Atoi(args[2]) // 문자를 숫자로 변경할 때 쓰이는 Atoi는 반드시 2개의 값을 반환하므로 err도 써줄 것(저 값 안 쓸꺼면 _로)
if err != nil {
    return shim.Error("3rd argument must be a numeric string")
}
price, err := strconv.Atoi(args[4])
if err != nil {
    return shim.Error("5th argument must be a numeric string")
}
status := strings.ToLower(args[5]) // 생성과 동시에 할당(:=)
```

: if args[5](=status)의 길이가 0이면 제대로 입력하라는 에러 문구 출력
: 입력받은 args[5]의 값을 status라는 변수로 지정(strings 타입으로 생성과 동시에 할당!)

```
objectType = "marbleStatus" // 단순 이름 지정이므로 struct타입과 꼭 같을 필요 없음
marbleStatus := &marbleStatusPDC{objectType, marbleName, status} // 구조체의 주소지정(&)
marbleStatusBytes, err := json.Marshal(marbleStatus)
if err != nil {
    return shim.Error(err.Error())
}
err = stub.PutPrivateData("collectionMarbleStatus", marbleName, marbleStatusBytes)
if err != nil {
    return shim.Error(err.Error())
}
```

: init()으로 입력받은 값을 collectionMarbleStatus에 저장하기 위해 PutPrivateData 구현
: marbleName에 해당하는 marbleStatusBytes값을 입력(Marshal함수를 통해 바이트형식으로 변환 후)

③ delete()에 status 삭제하는 코드 추가

```
//  Delete private status of marble
err = stub.DelPrivateData("collectionMarbleStatus", marbleName)
if err != nil {
    return shim.Error(err.Error())
}
```

④ **$ ./byfn.sh down**으로 모든 채널,서버 죽이고
$ ./byfn.sh up –c mychannel –s couchdb를 이용해 couchdb를 갖는 새로운 시스템 기동
state(저장소)
: 하게 되면 channel 생성, join, anchor-peer 생성, chaincode install, instantiate까지 자동으로 실행!!
: 여기서 생성되는 체인코드는 sample_example02로 내장된 코드(byfn.sh에서 삭제해버리면 작동 안함)
: byfn.sh 파일은 네트워크 기동을 쉽게 해주기 위한 스크립트 파일일 뿐!!
$ docker ps로 네트워크 잘 띄워졌는지 확인
$ docker exec –it cli bash를 이용해 docker로 이동!!(이 안에서 install 등 작업!)
interprete terminal
$ peer chaincode list —installed 와 peer chaincode list —instantiated –C mychannel
: 해보면 byfn.sh 파일에 내장되어 있는 기본값인 mycc만 설치되어 있는 것을 확인 가능
: 이건 별개로, marble관련 체인코드를 설치하고 인스턴쓰화 해줘야함!!
marblesp:1.0을 install하고, instantiate(강사님 github 참고!)
: down했으니까 어제 상태로 되돌리기 한 것
: Org1에서 price(보임!) Org2에서 price(에러!)

---------- 현재상황 ----------

install	marblesp : 1.0 mycc : 1.0
instance	**marblesp : 1.0** mycc : 1.0

⑤ marblesp 2.0 install
$ peer chaincode install -n marblesp -v 2.0 -p github.com/chaincode/marbles03_private/go
: 각 peer별로 환경변수 바꿔가며 install 작업 실시!(총 4번)

⑥ marblesp 2.0 upgrade
$ export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
$ peer chaincode upgrade -o orderer.example.com:7050 --tls —cafile $ORDERER_CA -C mychannel -n marblesp -v 2.0 -c '{"Args":["init"]}' -P "OR('Org1MSP.member','Org2MSP.member')" --collections-config $GOPATH/src/github.com/chaincode/marbles03_private/collections_config.json
: instantiate를 upgrade로만 바꿔서 실행(기존 marblesp 1.0 인스턴스를 업그레이드 하는 작업)
: 이후 installed된 list 확인해보면 3개 나옴(mycc, marblesp 1.0, marblesp 2.0)
: 해당 채널에 instantiated된 list 확인해보면 2개 나옴(mycc, marblesp 2.0) => 1.0은 2.0으로 덮어씌워짐

⑦ invoke(initMarble)
$ peer chaincode invoke -o orderer.example.com:7050 —tls —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 marblesp –c '{"Args":["initMarble","marble2","blue","35","tom","99","stock"]}'
: 이제 upgrade 한 chaincode를 바탕으로 invoke(initMarble 함수 이용) 작업 실시
: 만약 "marble1"으로 생성하면 error발생(why? marblesp 1.0에서 이미 생성한 데이터이기 때문!)
: 그러므로, "marble2"로 바꾸고, 맨 마지막에 "stock"이라는 status 정보 추가해서 작성할 것(총 6개 argument)

⑧ query
$ peer chaincode query -C mychannel -n marblesp -c '{"Args":["checkStatus", "marble"]}'
: Org2에서 하면 정상작동! Org1로 바꿔 쿼리 날리면 미 작동!(josn 파일에서 그렇게 설정했기 때문에)

```
root@af9ceab37b62:/opt/gopath/src/github.com/hyperledger/fabroot@af9ceab37b62:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode query -C mychannel -n marblesp -c '{"Args":["checkStatus", "marble"]}'
{"docType":"marbleStatus","name":"marble","status":"stock"}
root@af9ceab37b62:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode query -C mychannel -n marblesp -c '{"Args":["checkStatus", "marble"]}'
Error: endorsement failure during query. response: status:500 message:"{\"Error\":\"Failed to get state for marble\"}"
```

---------- 현재상황 ----------

install	marblesp : 2.0 marblesp : 1.0 mycc : 1.0
instance	**marblesp : 2.0** mycc : 1.0