Fabric Project Document

2015004302 곽상원

목차

---

1. Source code description -Line별로 설명

1) Chaincode

▶Item.java: 중고거래의 대상이 되는 물품 asset을 정의

16~20: 물품의 상태를 나타내기위한 열거형으로 YET은 등록만 된 상태, ON_SALE은 판매등록이 된 상태, DONE은 거래가 종료된 상태를 의미한다.

22~32: 물품의 정보를 담는 변수들이다. name은 제조사, owner는 소유주, state는 물품의 상태, price은 가격이다.

33~48: 물품의 정보를 가져오기위한 get함수들이다.

▶Token.java: 각 user가 지닌 토큰을 표현하기위한 asset으로 user마다 1개씩 생성된다.

15~19: Token의 정보를 담는 변수들이다. token은 토큰의 개수, owner는 이 토큰의 소유주를 나타낸다.

21~27: 물품의 정보를 가져오기위한 get함수들이다.

▶Number.java: 현재 등록된 물품과 user의 수를 저장하는 asset으로 Item과 Token asset의 key값을 결정하기위해 사용된다.

15~18: inum은 현재 등록된 물품의 수, unum은 현재 등록된 user의 수

20~26: Number의 정보를 가져오기위한 get함수들이다.

▶ItemQueryResult.java: 물품에 대한 query 요청시, 그에 대한 응답을 담기위해 쓰인다.

▶TradeItem.java: 실제 logic이 구현돼있다.

43~46: Error에 대한 열거형으로 ITEM_NOT_FOUND은 해당 물품이 없을 때, ITEM_ALREADY_EXISTS는 물품이 이미 존재할 때 사용된다.

48~53(initLedger()): fabric network를 시작할 때 실행되는 함수로, adminToken()과 numInit()을 호출한다.

55~63(adminToken()): 토큰의 개수는 1000000, 소유주는 admin인 Token을 생성한다. 이때 Token의 key값은 “TOKEN1”이다.

65~73(numInit()): 네트워크 시작 시에 물품은 0개, user수는 admin 1명이므로 이를 저장하는 Number asset을 생성한다.
76~92(tokenNum()): owner 정보를 전달하면 그 owner의 Token에 해당하는 번호를 반환해준다. 예를 들어 kim의 토큰이 "TOKEN15"에 해당하면 15를 반환한다. 이때 탐색범위는 "TOKEN1"부터 "TOKEN99"이다.
만약 소유주에 해당하는 토큰이 존재하지 않으면 0을 반환한다.
94~125(earnToken()): owner 정보를 전달하면 그 owner에 대한 Token을 생성하고, admin이 owner에게 토큰 100개를 전달하는 함수이다. 98~102에서 이미 100토큰을 받았다면 "User already receive!"라는 error message를 예외로 발생시킨다. 103~122는 admin의 토큰 개수는 100개 줄이고, owner의 Token을 Number를 이용해 생성하고 Number를 갱신한다. 예를 들어 Number의 inum=0, unum=1이라고 하면, 소유주가 "kim"이고 토큰 개수가 100개인 Token을 생성할 때 (unum+1)=2이므로 "TOKEN2"가 key값이 된다. 그리고 unum이 2로 바뀌었으므로 Number를 갱신한다.
127~146(registerItem()): 소유주 owner와 제조사 name 정보를 받으면 그에 해당하는 물품을 등록하는 함수이다. 131~143은 Item을 생성할 때 Number를 이용해 생성하고 Number를 갱신한다. 예를 들어 Number inum=0, unum=2라고 하면, 소유주가 "kim"이고 제조사가 "NIKE"인 Item을 생성할 때 (inum+1)=1이므로 "ITEM1"이 key값이 된다. 그리고 inum이 1로 바뀌었으므로 Number를 갱신한다.
148~163(sellMyItem()): 물품의 key값과 가격 price 정보를 받으면 그에 해당하는 물품을 판매 등록하는 함수이다. 152~156은 key에 해당하는 물품이 없는 경우 ITEM_NOT_FOUND error message를 발생시킨다. 157~160은 해당 물품의 state를 YET에서 ON_SALE로 바꾸는 것이다.
165~202(buyUserItem()): 물품의 key값과 새로운 소유주 newOwner 정보를 받으면 그에 해당하는 물품을 구매하는 함수이다. 170~174는 key에 해당하는 물품이 없는 경우 ITEM_NOT_FOUND error message를 발생시킨다. 175~186은 newOwner의 Token에서 해당 물품을 구매하기에 충분한 토큰이 있는지 확인하는 것으로, 토큰이 부족하면 "Short Money!" error message를 발생시킨다. 189~191은 구매자인 newOwner의 토큰 개수를 물품 가격만큼 깎아주고, 193~199는 판매자에게 물품의 가격만큼 토큰을 더해준다. 마지막으로 201에서 changeItemOwner()를 실행한다.
204~215(changeItemOwner()): 물품의 key값과 새로운 소유주 newOwner 정보를 받으면 그에 해당하는 물품의 소유주를 newOwner로 바꾸는 함수이다.
217~236(getMyItems()): 소유주인 owner 정보를 받으면 owner가 등록한 모든 물품을 보여주는 함수이다. 탐색범위는 "ITEM1"부터 "ITEM99"까지로, 229~231에서 물품의 owner와 인자로 전달받은 owner가 같은 경우만 결과에 담는다. 그러면 이를 JSON형식으로 parsing해서 결과로 반환한다.
238~253(getAllItems()): 등록된 모든 물품을 보여주는 함수이다. 탐색범위는 "ITEM1"부터 "ITEM99"까지다.
255~272(getAllRegisteredItems()): 판매 등록된 모든 물품을 보여주는 함수이다. 탐색범위는 "ITEM1"부터 "ITEM99"까지로, 265~267에서 물품의 state가 ON_SALE인 경우만 결과에 담는다. 그러면 이를 JSON형식으로 parsing해서 결과로 반환한다.

274~291(getAllOrderedItems()): 판매가 완료된 모든 물품을 보여주는 함수이다. 탐색범위는 "ITEM1"부터 "ITEM99"까지로, 284~286에서 물품의 state가 DONE인 경우만 결과에 담는다. 그러면 이를 JSON형식으로 parsing해서 결과로 반환한다.

2) Fabric Node SDK

▶enrollAdmin.js: "admin"을 등록하고 이를 wallet에 추가한다.

▶registerUser.js: name정보를 받으면 이에 해당하는 user를 등록하고 wallet에 추가한다.

▶invoke.js: 함수이름 fun, user이름 name, 그 외 인자인 args를 받으면 이 정보를 이용해 특정 함수를 invoke한다. 42~54는 각 함수이름에 대한 호출방식을 별도로 설정한 모습이다.

▶query.js: 함수이름 fun, user이름 name을 받으면 이 정보를 이용해 get함수를 호출한다. 41~48에서 각 함수이름에 대한 호출방식을 별도로 설정하고 그 결과를 JSON객체로 변환하는 모습이다.

3) Route(index.js)

11~66: '/' URL로 data가 get되면 홈페이지를 렌더링한다. 13~17에서 렌더링할 부분에 대한 변수를 선언했다. 20~36은 user가 정해진 경우에만 실행되는 영역으로, getMyItems()를 이용해 user의 물품목록과 판매할 수 있는 물품목록을 얻는다.
37~43은 getAllRegisteredItems()를 이용해 구매할 수 있는 물품목록을 얻는다.
45~50은 getAllItems()를 이용해 등록된 모든 물품목록을 얻는다.
52~63은 getAllRegisteredItems()와 getAllOrderedItems()를 이용해 판매등록 및 판매가 완료된 모든 물품목록을 얻는다.
마지막으로 64에서 여태까지 얻은 데이터를 홈페이지에 렌더링한다.

79~82: '/earnToken' URL로 data가 post되면 earnToken()을 호출하고 res.redirect('/')로 홈페이지를 렌더링한다.

84~90: '/registerItem' URL로 data가 post되면 registerItem()을 호출하고 res.redirect('/')로 홈페이지를 렌더링한다.

92~101: '/sellMyItem' URL로 data가 post되면 sellMyItem()을 호출하고 res.redirect('/')로 홈페이지를 렌더링한다.

103~108: '/buyUserItem' URL로 data가 post되면 buyUserItem()을 호출하고 res.redirect('/')로 홈페이지를 렌더링한다.

2. 실행 및 실행결과

▶실행방법

1) npm install //필요한 node module을 설치한다.
2) **./network_start.sh //fabric 네트워크를 생성하고 initLedger를 실행해준다.**
3) npm start //서버 실행

----------localhost:3000에 접속----------

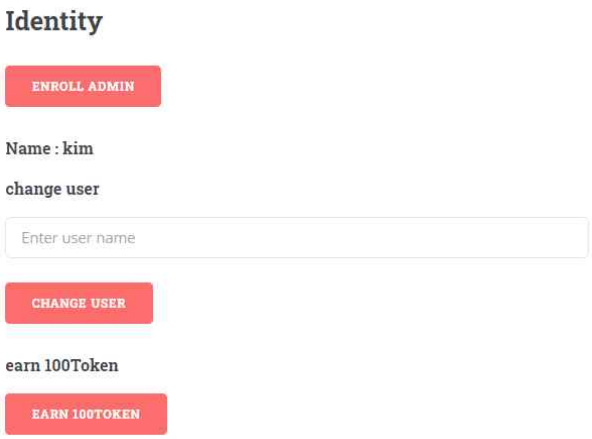
4) "ENROLL ADMIN"버튼을 눌러 admin 계정을 생성한다.
5) 사용자 명을 입력하고 CHANGE USER를 누른다.
6) **EARN 100TOKEN을 누른다. // 이때 사용자의 Token이 생성되므로 반드시 눌러야한다.**
7) 자유행동...

▶시나리오 및 실행결과

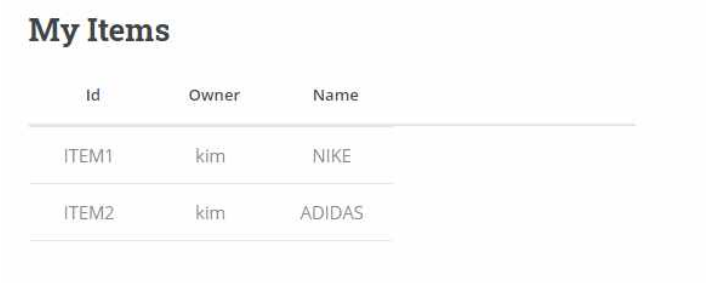
1) kim이 사용자로 등록하고 EARN 100TOKEN
2) 2가지 물품을 등록하고 그 중 1개를 판매 등록한다.
3) park이 사용자로 등록하고 EARN 100TOKEN
4) park이 판매 등록된 물품을 구매한다.

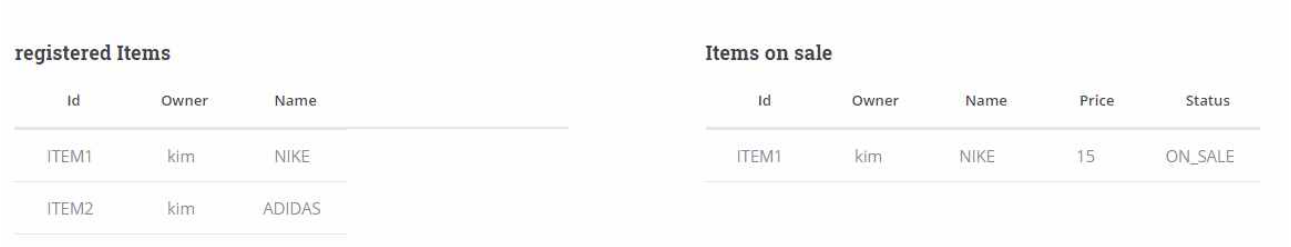
---

1) kim이 사용자로 등록하고 EARN 100TOKEN

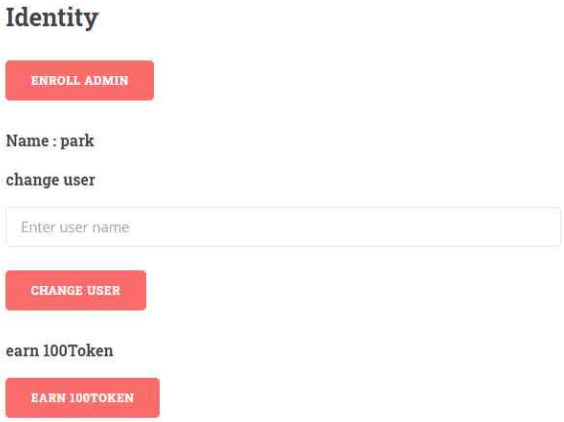


2) 2가지 물품을 등록하고 그 중 1개를 판매 등록한다.



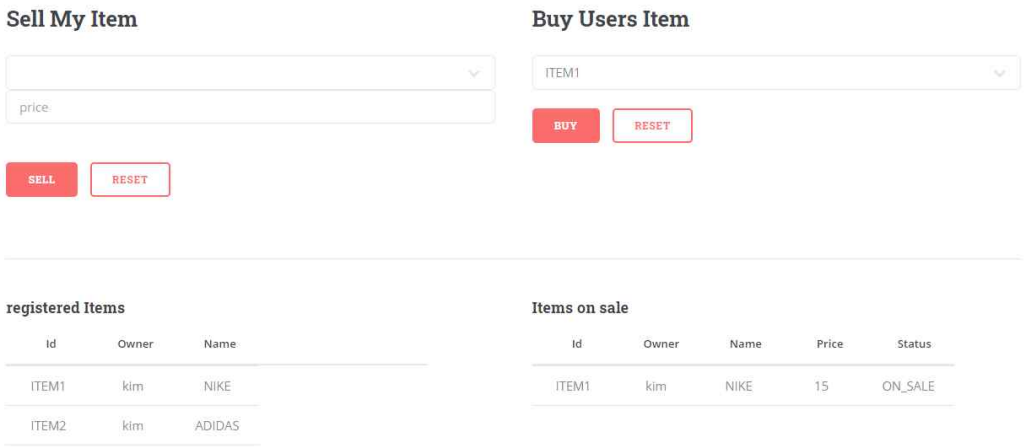


3) park이 사용자로 등록하고 EARN 100TOKEN



4) park이 판매 등록된 물품을 구매한다.

-구매 전



-구매 후

My Items

Id	Owner	Name
ITEM1	park	NIKE

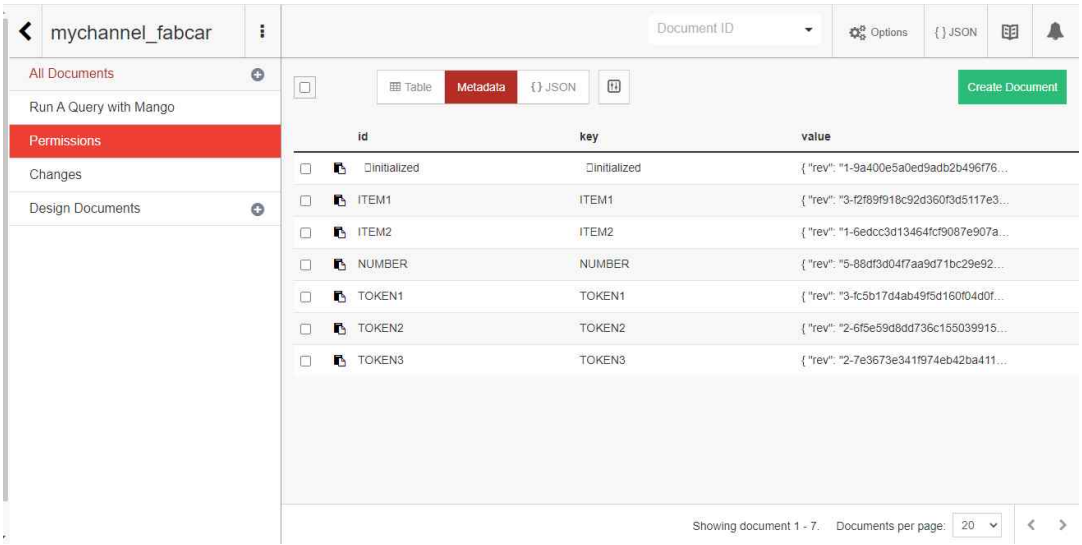
registered Items

Id	Owner	Name
ITEM1	park	NIKE
ITEM2	kim	ADIDAS

Items on sale

Id	Owner	Name	Price	Status
ITEM1	park	NIKE	15	DONE

-CouchDB "mychannel_fabcar"의 결과



ITEM1은 "NIKE", ITEM2는 "ADIDAS"
TOKEN1은 "admin", TOKEN2는 "kim", TOKEN3는 "park"의 토큰 정보를 담은 asset이다.

```
{
  "_id": "ITEM1",
  "_rev": "3-f2f89f918c92d360f3d5117e3c0ab1b5",
  "name": "NIKE",
  "owner": "park",
  "price": 15,
  "state": "DONE",
  "~version": "CgMBDAA="
}
```

ITEM1의 owner와 state가 변한 것을 확인할 수 있다.