< project 코딩 간 배운 점 >

1. groot-app 실습 간 배운점

- npm설치의 중요성
  · 10번이 넘는 시행착오의 반복, 해결은?
    $ git clone https://github.com/hyperledger/education.git
    $ cd education/LFS171x/fabric-material/tuna-app 으로 처음부터 재 다운로드
    -> tuna-app에 있는 내용을 그대로 groot-app으로 복사해 $npm install
  · 그래도 error가 발생! 그놈에 버전이 문제다
    -> Node.js 문제일 경우?
    $ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
    $ sudo apt-get install -y nodejs 로 nodejs를 깔아주기
    $ node –v으로 버전 확인(무조건! 8이상이어야 함)
    $ sudo npm cache clean –f로 남은 캐시데이터들을 삭제
    $ sudo npm install –g n 모듈 설치
    $ sudo n 8.9.4 하면 node@8.9.4 버전 설치완료
    $ npm rebuild로 최종 업데이트 해주면 끝!
    -> 다른 문제일 경우?(npm v6에서 제공하는 기능 사용)
    $ npm audit으로 사용하는 npm 모듈의 취약점을 검사해 나오는 high문제들 해결!!

- npm을 제대로 설치해서 노란 네모박스에 successful이 뜬다면 드디어 성공 한 것!
  $ npm install –g도 해서 전체적으로 한 번 더 설치를 완료 후
  $ startFabric.sh 실행을 통해 fabric network 구동!
  $ docker ps로 6개 다 떴는지 확인(chaincode, peer, orderer, cli, couchDB, ca)
  $ node registerAdmin.js

```
groot@GROOT-Server4:~/groot/groot-app$ node registerAdmin.js
 Store path:/home/groot/.hfc-key-store
(node:2230) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
Successfully enrolled admin user "admin"
Assigned the admin user to the fabric client ::{"name":"admin","mspid":"Org1MSP","roles":null,"affiliation":"","enrollmentSecret":"","enrollment":{"signingIdentity":"7307c30abbd5dd877e18184e77af37a9e4f6fb4b54f
373b7b599120f56be463c","identity":{"certificate":"-----BEGIN CERTIFICATE-----\nMIICATCCAaigAwIBAgIUG0k2Y1FkLiPMCXDSkl8nJm984MUwCgYIKoZIzj0EAwIw\nczELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNh
\nbiBGcmFuY2lzY28xGTAXBgNVBAoTEG9yZzEuZXhhbXBsZS5jb20xHDAaBgNVBAMT\nE2NhLm9yZzEuZXhhbXBsZS5jb20wHhcNMTgxMjI2MDQ1NTAwWhcNMTkxMjI2MDUw\nMDAwWjAhMQ8wDQYDVQQLEwZjbGllbnQxDjAMBgNVBAMTBWFkbWluMFkwEwYHKoZI\nzj0CAQYIK
oZIzj0DAQcDQgAE03omEizbnNx9647zVaPSuVnXsNZjYLgJ3syIJ4/w\nP0yw0whsJCCkivx7vUgtbJ3KEM12FYzfOPiZDRsxfX2hAaNsMGowDgYDVR0PAQH/\nBAQDAgeAMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFC3yDPMXJiH9Rrc5aqk2tecZ\nnFuwjMCsGA1UdIwQkMCKA
IEI5qg3NdtruuLoM2nAYUdFFBNMarRst3dusalc2Xkl8\nMAoGCCqGSM49BAMCA0cAMEQCIGFCn4BH2bsIzy7/iLtMWIxPETbqqP36BQZJLg0k\npDmnAiAg3hCRxcx15HDWXVjewcu64giHHqlY04B2yjmjmldBVg==\n-----END CERTIFICATE-----\n"}}}
```

  $ node registerAdmin.js

```
groot@GROOT-Server4:~/groot/groot-app$ node registerUser.js
 Store path:/home/groot/.hfc-key-store
(node:2520) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
Successfully loaded admin from persistence
Successfully registered user1 - secret:ScfVEZAfcGze
Successfully enrolled member user "user1"
User1 was successfully registered and enrolled and is ready to intreact with the fabric network
```

  $ cd ~/.hfc-key-store에서 $ll로 key값 제대로 생성되었는지 확인>_<

```
groot@GROOT-Server4:~/.hfc-key-store$ ll
total 32
drwxrwxr-x  2 groot groot 4096 Dec 26 14:08 ./
drwxr-xr-x 12 groot groot 4096 Dec 26 15:17 ../
-rw-rw-r--  1 groot groot  246 Dec 26 14:08 3a03d2b89ecad4c74465b0a58fbb2b2cadcb3710d74ed9340ede8d75f9a3086f-priv
-rw-rw-r--  1 groot groot  182 Dec 26 14:08 3a03d2b89ecad4c74465b0a58fbb2b2cadcb3710d74ed9340ede8d75f9a3086f-pub
-rw-rw-r--  1 groot groot  246 Dec 26 13:59 7307c30abbd5dd877e18184e77af37a9e4f6fb4b54f373b7b599120f56be463c-priv
-rw-rw-r--  1 groot groot  182 Dec 26 13:59 7307c30abbd5dd877e18184e77af37a9e4f6fb4b54f373b7b599120f56be463c-pub
-rw-rw-r--  1 groot groot  986 Dec 26 13:59 admin
-rw-rw-r--  1 groot groot 1180 Dec 26 14:08 user1
```

  $ node server.js 로 서버 구동!!!!

2. Hyperledger-Fabric과 DJango 연동 간 배운점

- 연동은 어려운 게 아님.
- DJango에서 입력받은 값을 Fabric의 입력 값으로 전달해주는 개념
- 현재 Fabric은 python을 제공하지 않기 때문에 Fabric의 controller.js, routes.js, server.js만 수정 후 DJango의 views.py에서 연결 실시
- routes.js에서 지정해준 url주소와 controller.js에서 지정해준 데이터 파싱 방법('-'로 스플릿)을 통해 DJango의 client로부터 입력받은 값을 전달!!
  requests이용 값 전달(맨 위에 import requests)

```
# Hyperledger-Fabric으로 데이터 전송@@@@@@@@@@
#     0         1        2       3        4        5          6           7
# Technology   Sort   Company   Term   Content   Client   Cont_term   Enroll_date
fabric = "http://210.107.78.150:8000/add_cont/" + contract.title + "-" + contract.sort + "-" +
                                                  User.objects.get(user_id=com_name) + "-" + contract.term + "-" +
                                                  "Content" + "-" + "Client(없으면 null)" + "-" +
                                                  "3" + "-" + "2020.01.01"
f = requests.get(fabric)
print(f.text) # cmd 창에 보여질 값
```

  이렇게 해주면 입력받은 값들이 Hyperledger-Fabric Network에 전달되고 peer를 통해 block과 WS에 기록됨(couchDB 확인으로 알 수 있음 : http://210.107.78.150:5984/_utils)
- requests 모듈? python에서 HTTP요청을 보내는 모듈

  1. GET 요청할 때 parameter 전달법

```
params = {'param1': 'value1', 'param2': 'value'}
res = requests.get(URL, params=params)
```

```
In [7]: params = {'key':'value'}

In [8]: res = requests.get('http://www.tistory.com', params=params)

In [9]: res.url
Out[9]: u'http://www.tistory.com/?key=value'
```

응답 객체인 res 를 통해서 내가 실제로 던진 URL이 뭔지 확인해보았다. 내가 준 URL과 파라미터를 requests 모듈이 엮어서 적절한 새로운 요청을 만든 것이다. 내가 직접 URL을 저렇게 타이핑하는 것보다 파라미터를 딕셔너리 형식으로 정리하고 requests 모듈에 던져주는 것이 훨씬 좋다고 생각한다.

  위 사진처럼 전달해도 될 듯 / f.text말고 f.json()도 가능!!
  https://dgkim5360.tistory.com/entry/python-requests (← URL 참고해서 공부)

- requests와 유사한 크롤링에서 배웠던 urllib 공부
  urllib 라이브러리는 python에서 웹과 관련된 데이터를 쉽게 이용하게 도와주는 역할.
  · 총 4개의 모듈이 존재
    ① request : 웹 문서 불러오기
      * urllib.request.urlopen("--주소--") : 웹에서 얻은 데이터에 대한 객체 반환
      * urllib.request.urlopen("--주소--").getheaders() : 해당 웹서버의 정보를 list형식으로 반환
      * urllib.request.urlopen("--주소--").status : 웹페이지의 상태 확인(200~500대 숫자로 반환)
      * urllib.request.urlopen("--주소--").read() : 웹페이지의 HTML 코드를 출력

    ② parse : url에 한글 검색어 입력 가능
      * 한글로 검색하기 위해서는 인코딩을 해야 하는데 이를 도와주는 모듈
      * urllib.parse.quote_plus("--입력 값--") : 웹이 알아들을 수 있도록 인코딩 해 줌(공백을 +로 처리)
      * urllib.parse.quote("--입력 값--") : 웹이 알아들을 수 있도록 인코딩 해 줌(공백을 %20으로 처리)

- Class has no objects member(클래스에 객체 맴버가 없다는 오류가 뜰 때는?)

Install `pylint-django` using `pip` as follows

```
pip install pylint-django
```

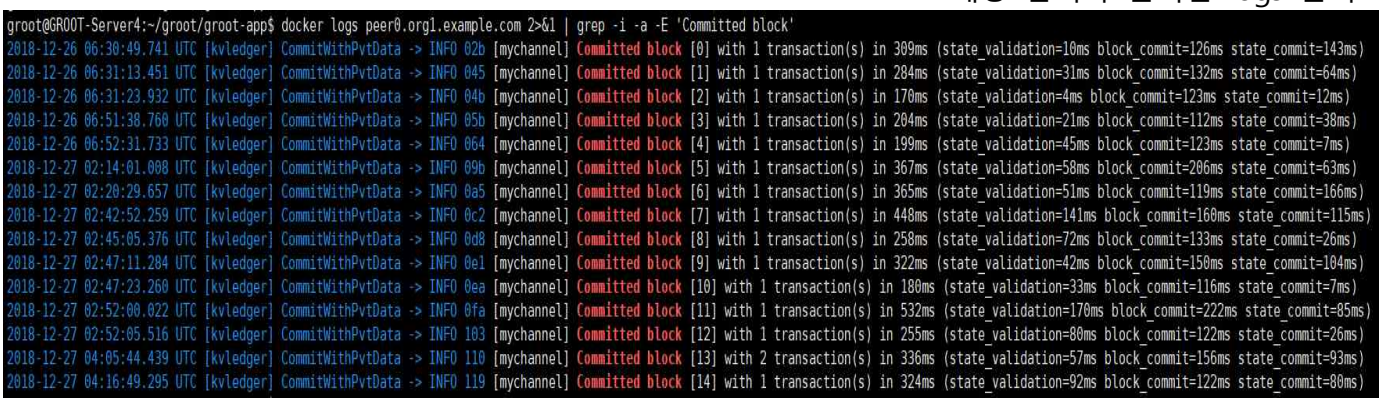
Then in Visual Studio Code goto: **User Settings** (Ctrl + , or File > Preferences > Settings if available) Put in the following (please note the curly braces which are required for custom user settings in VSC):

```
{"python.linting.pylintArgs": [
    "--load-plugins=pylint_django"
],}
```

pylint란 Python 코드의 오류를 검사하고 적절한 Python 코딩 패턴을 권장하며 널리 사용되는 도구로, Python 프로젝트용 Visual Studio에 통합되어 있다.(이거 깔고 환경변수 설정해주면 오류 해결!)
코드 분석의 방법으로는 정적 분석과 동적 분석이 있는데 정적분석은 코드를 실행하지 않은 상태에서 분석하는 것이며, 디버깅이나 유닛테스트 등은 동적분석이다.(코드를 실행한 상태에서 분석하는 것)
pylint '--파일이름--' 치면 정적분석의 결과가 쫙~ 나옴!
참고 : http://www.sysnet.pe.kr/Default.aspx?mode=2&sub=0&detail=1&pageno=0&wid=11119&rssMode=1&wtype=0

3. Block 쌓이는 내용 보기

- $ docker logs peer0.org1.example.com 2>&1 | grep -i -a -E 'Committed block'
해당 단어가 들어간 logs 출력



치면 블록높이를 볼 수 있음(몇 개가 쌓였는지)

· 2>&1 의미? 쉘 스크립트에서 에러 메세지를 화면에 표시하지 않도록 하는 것
0(표준입력), 1(표준출력), 2(표준에러)
http://blogger.pe.kr/369 ← 블로그 참고(GOOD!!)

· grep 의미? 파일이나 표준입력으로부터 패턴을 찾아 해당 라인(행)을 출력해주는 역할
파일의 내용이나 콘솔의 출력물 중 특정 문자열을 찾는데 사용!
어떤 실행결과로부터(| 전에 입력된) 문자를 파싱하기위한 용도로 많이 사용
http://jybaek.tistory.com/704 ← 블로그 참고

-i 옵션 : 문자열의 대소문자 구분 안 함(--ignore-case)
-a 옵션 : 바이너리파일을 텍스트파일처럼 처리하게 해줌(--text)
기본적으로 grep는 바이너리 파일을 처리 할 수 없음
-E 옵션 : 확장 정규표현식으로 패턴 해석(--extended-regexp)
결국 egrep와 같은 의미!
http://geundi.tistory.com/113 ← 블로그 참고

4. vi 에디터 사용법

- 주석 추가 및 삭제(https://norux.me/30 참고)
  · vim의 편리한 기능 중 하나인 key mapping을 사용해 주석 처리 및 삭제
    * vi ~/.vimrc를 열어 파일 수정
    * ctrl + c를 누르면 주석처리를 하고
      ctrl + x를 누르면 주석을 지우게끔
    * '<.'> 옆의 문자는 비주얼 모드에서 블록의 범위를 뜻함
      norm i// 노말모드에서 앞에 //를 추가하겠다
      norm 2x 노말모드에서 앞에 2글자를 지우겠다
- https://www.linux.co.kr/linux/vi/vi.html ← 참고 블로그
- http://mintnlatte.tistory.com/189 ← 참고 블로그(들여쓰기 관련)
    * vi ~/.vimrc 파일에서 기본 설정해줘서 이제는 사용만 하면 됨
    * 들여쓰기 원하는 줄을 vim으로 블록지정하고 >>입력!
    * 들여쓰기 취소하기 원하는 줄을 vim으로 블록지정하고 <<입력!
- 문자열 치환(:시작줄,끝줄s/찾을패턴/바꿀문자/옵션)

2. 바꾸기 기초 [edit]

Vi 에서 문자(열) 바꾸기는 콜론 모드에서 's'ubstitute 명령을 사용한다.

```
:(시작줄),(끝줄)s/찾을패턴/바꿀스트링/옵션
```

- 시작줄, 끝줄 : 바꾸기를 할 범위를 행번호로 지정. "."는 현재 커서가 있는 줄. "$" 는 제일 마지막 줄을 의미
- 찾을 패턴, 바꿀 스트링 - 말 그대로... 전자를 찾아 후자로 치환한다. 앞은 "pattern" 이고 뒤는 "string" 임에 주목. 가장 큰 차이점은, 전자에는 "정규표현식"을 사용할 수 있다는 것.
- 옵션 :
  - g : global - 한 줄에 패턴이 여러 번 나오면 모두 바꾼다. 지정하지 않으면 첫번째 패턴만 치환
  - i : ignore case - 대소문자 구분을 하지 않는다.
  - c : confirm - 검색된 모든 문자열에 대해서 바꿀지 말지를 물어본다.

예:

```
:5,10s/a/b/      - 5번째 줄부터 10번째 줄까지 각 줄의 첫번째 "a" 를 "b" 로 바꾼다.
:.,.+10s/a/b/g   - 현재 줄부터 (현재 행번호+10)번째 줄까지 모든 "a" 를 "b" 로 바꾼다.
:1,$s/a/b/c      - 첫번째 줄부터 마지막 줄까지 (즉 문서 전체) 각 줄의 "a" 를 "b" 로 바꾸되, 사용자에게 확인을 받는다.
:%s/a/b/gi       - 역시 문서 전체에서 "a" 와 "A" 를 "b" 로 바꾼다.
:%s/Hello/Good Morning/g - 당연히... 두 글자 이상의 문자열도 검색 및 치환이 가능하다.
```