블록체인 (Blockchain)

03. 개발환경 세팅, Blockchain 클래스 작성

소프트웨어 꼰대 강의

노기섭 교수

(kafa46@cju.ac.kr)

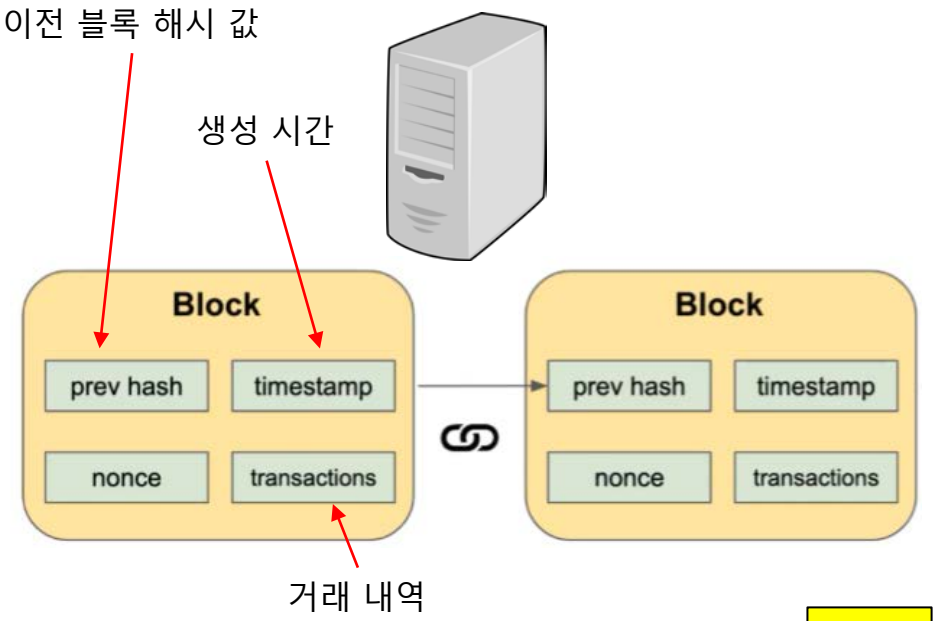
Blockchain 구현 및 작동

일단 서버가 돌아가는지 확인

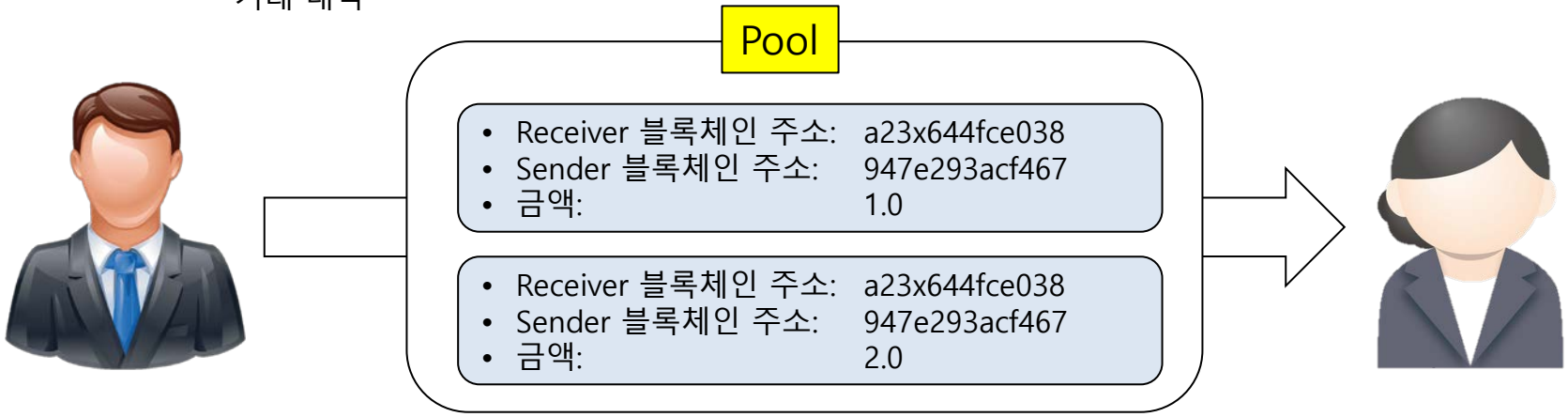
터미널에서 서버 실행

- (venv)$ export FLASK_APP=mining # 윈도우일 경우 export 대신 set 명령어 사용
- (venv)$ export FLASK_DEBUG=True # 윈도우일 경우 export 대신 set 명령어 사용
- (venv)$ flask run –h 0.0.0.0 –p 8000

구현해야 하는 상황

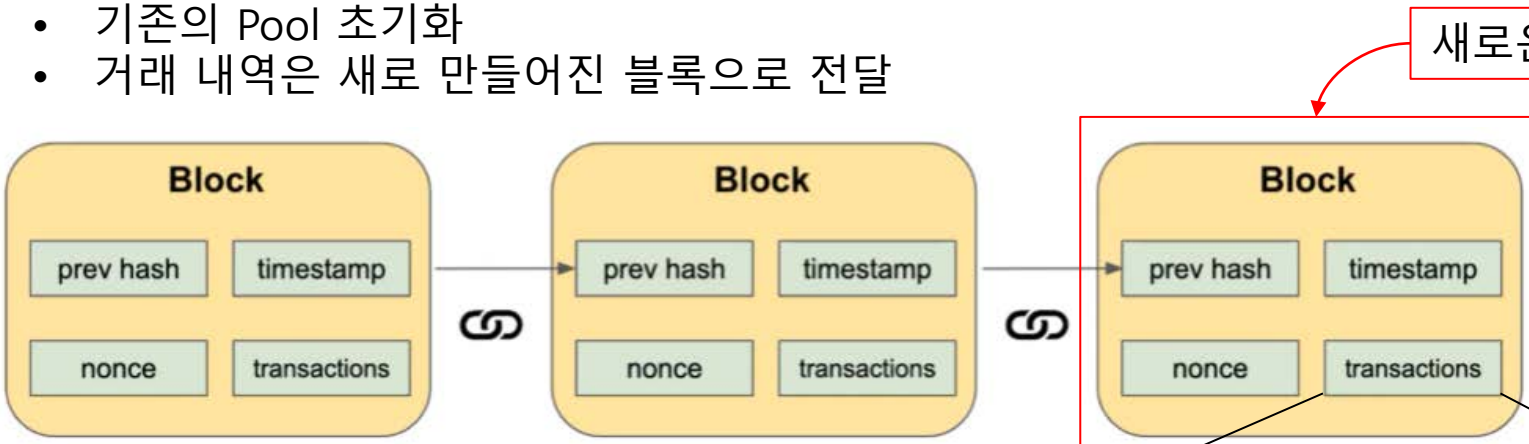


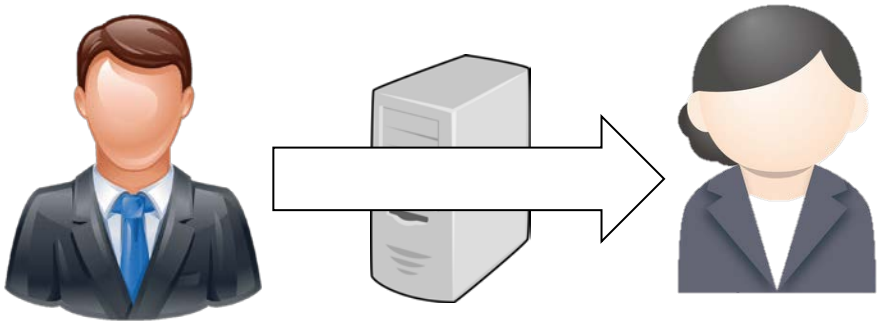
- Pool에 거래내역(transaction)들을 포함
- 마이닝을 통해 nonce 구하기
- 새로운 블록을 만들면 기존의 Pool은 초기화 되고 기존 거래내역은 다음 블록으로 전달
  (다음 슬라이드 참조)



새로운 Block 생성된 상황

- 기존의 Pool 초기화
- 거래 내역은 새로 만들어진 블록으로 전달





- Receiver 블록체인 주소: a23x644fce038
- Sender 블록체인 주소: 947e293acf467
- 금액: 1.0

- Receiver 블록체인 주소: a23x644fce038
- Sender 블록체인 주소: 947e293acf467
- 금액: 2.0

Pool

빈 리스트로 초기화!

블록체인 클래스 작성 (blockchain.py)

blockchain.py

```python
class BlockChain:
    '''블록체인 클래스'''
    def __init__(self) -> None:
        pass

    def create_genesis_block(self,) -> bool:
        '''Genesis Block 생성'''
        block_exist = Block.query.all()
        if block_exist:
            print({
                'status': 'Fail to create genesis block',
                'error': 'Block(s) aleady exist'
            })
            return False

        genesis_block = Block(
            prev_hash=blockchain_utils.hash({}),
            nonce=0,
            timestamp=time.time(),
        )
        db.session.add(genesis_block)
        db.session.commit()

        return True


    def create_block(self, nonce:int, prev_hash:str=None):
        '''블록체인에서의 새로운 단위 블록 생성'''
        try:
            db.session.add(
                Block(
                    prev_hash=prev_hash,
                    nonce=nonce,
                    timestamp=time.time(),
                )
            )
            db.session.commit()
            return True
        except Exception as e:
            print('Fail to block on database')
            print(f'Error: {e}')
            return False
```

블록체인 유틸리티 작성 (/utils/blockchain_utils.py)

blockchain_utils.py

```python
import hashlib
import json
import collections

from mining.blockchain import BlockChain
from mining.models import Block, Transaction


def print_blockchain(chains):
    '''블록체인 보기좋게 출력'''
    for idx, chain in enumerate(chains):
        print(f"\n\n{'===' * 5} Blockchain {idx} {'===' * 5}")
        for k, v in chain.items():
            print(f'{k:15}{v}')
        print(f"{'***' * 3} End of blockchain {idx} {'***' * 3}")

def sorted_dict_by_key(unsorted_dict: dict):
    return collections.OrderedDict(
        sorted(unsorted_dict.items(), key=lambda keys: keys[0])
    )

def get_blockchain():
    '''데이터베이스로부터 블록체인 정보 가져오기'''
    blockchain_eixst = Block.query.all()
    if not blockchain_eixst:
        block_chain = BlockChain()
        block_chain.create_genesis_block()
    return build_blockchain_json()
```

blockchain_utils.py
작성 후 main_views.py에 적용했는지
확인해 주세용^^

```python
def build_blockchain_json() -> dict:
    blocks = Block.query.filter(
        Block.timestamp,
    ).order_by(Block.timestamp)
    result_dic = {
        'chain': [],
        'transaction_pool': [],
    }

    for block in blocks:
        result_dic['chain'].append(
            {
                'nonce': block.nonce,
                'prev_hash': block.prev_hash,
                'timestamp': block.timestamp,
                'transactions': get_transaction_list(block),
            }
        )
    last_block = Block.query.filter(
        Block.timestamp,
    ).order_by(Block.timestamp.desc()).first()
    result_dic['transaction_pool'] = get_transaction_list(last_block)
    return result_dic


def get_transaction_list(block: Block) -> list:
    transaction_exist = Transaction.query.all()
    if not transaction_exist:
        return [] # 빈 리스트 리턴

    transaction_list = []
    transactions = block.transactions
    for transaction in transactions:
        transaction_list.append(
            {
                'send_blockchain_addr': transaction.send_addr,
                'recv_blockchain_addr': transaction.recv_addr,
                'amount': transaction.amount,
            }
        )
    return transaction_list


def hash(block: dict) -> str:
    sorted_block = json.dumps(block, sort_keys=True)
    return hashlib.sha256(sorted_block.encode()).hexdigest()
```

서버 실행

터미널에서 서버 실행

- (venv)$ export FLASK_APP=mining # 윈도우일 경우 export 대신 set 명령어 사용
- (venv)$ export FLASK_DEBUG=True # 윈도우일 경우 export 대신 set 명령어 사용
- (venv)$ flask run –h 0.0.0.0 –p 8000

Shell Script 작성하기

- run_mining.sh 파일 생성
  - 터미널에서 입력했던 순서대로 작성

쉘 코드로 실행해 보기

- (venv)$. run_mining.sh

다음 강의

➔ 거래 클래스 작성

(Transfer.py)

수고하셨습니다 ..^^..