put(account, keys, values)関数
入力:
account　　処理を実行するアカウント
keys　　パラメータ名の配列
values　　パラメータ値の配列
※keysとvaluesは同一インデックスの値がペアで、key.value形式のパラメータ情報となる

出力:なし

get(keys, values)関数
入力:
keys　　パラメータ名の配列
values　　パラメータ値の配列
※keysとvaluesは同一インデックスの値がペアで、key.value形式のパラメータ情報となる
出力:任意の構造体データ(※辞書形式で表現可能な形式)

リスト1

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


import sys
import json
import datetime
import uvicorn

from fastapi import FastAPI
from fastapi import Request
from fastapi.responses import JSONResponse
from fastapi.security import OAuth2PasswordBearer, OAuth2PasswordRequestForm
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from fastapi.staticfiles import StaticFiles
from routers import channel_rt, data_rt, file_rt, bchain_data_rt, nft_rt
from APP import event


IP="0.0.0.0"
PORT=8000


# 初期化
app = FastAPI()

```

```
# イベント処理ミドルウェア
evt = event.Event()
@app.middleware("http")
async def process_event(request: Request, call_next):
    print("START process_event", request)

    # 処理前イベント
    res = evt.pre_event(request)

    # NFT対応の特殊処理
    if 'MyPhotoNFT' in res:
        r = res['MyPhotoNFT']
        print(r)

        if r['status'] != 0:
            print("NO call_next")
            return JSONResponse(content=r['data'], status_code=403)

    # 処理実態
    response = await call_next(request)

    # 処理後イベント
    res = evt.post_event(request, response)

    print("END process_event", response)

    return response


# CORS対応
origins = [
    "http://localhost",
    "http://localhost:"+str(PORT),
]
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)


# API処理)の登録
app.include_router(data_rt.router)
app.include_router(channel_rt.router)
app.include_router(bchain_data_rt.router)
app.include_router(file_rt.router)
app.include_router(nft_rt.router)


# システム情報
class SystemInfo(BaseModel):
    System = "My Data Server"
```

```
    Version = "1.0b"
    Release = "2022/08/31"

@app.get("/info", response_model=SystemInfo)
def read_root():
    return SystemInfo()


# 画面表示用の静的ファイル
app.mount("/dataserver", StaticFiles(directory="html"), name="html")


@app.get("/")
async def read_root():
    return {"My Data Server"}

if __name__ == "__main__":
    uvicorn.run(app, host=IP, port=PORT)
```

第2部第2章リスト1(mydata_server.py)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


import sys
import json

from LIB import db
from LIB import ethereum

DEBUG = False # True


class Event:
    # コンストラクタ
    def __init__(self):
        print("Constructor of Event")
        self.event_list = {}
        self.eth_rec = None

        s = db.DB()
        # Ethereumへの接続設定
        eth_rec = s.select(db.DsEthereum, "Ethereum")
        if DEBUG : print("eth_rec=",eth_rec)

        url = "http://" + eth_rec.address + ":" + str(eth_rec.port)
```

```
27            account = eth_rec.account
28            password = eth_rec.password
29            private_key = eth_rec.private_key
30            self.eth = ethereum.Ethereum(url, account, password, private_key)
31
32            # イベント登録
33            self.events = s.list(db.DsEvent)
34            for rec in self.events:
35                if DEBUG : print(rec.id, rec.name, rec.func_id)
36
37                rec.key = rec.method + rec.url
38                rec.call = self.call_contract
39
40                # Function情報の取得
41                rec.func = s.select(db.DsFunction, rec.func_id)
42                if DEBUG : print(rec.func)
43                if rec.func:
44                    # コントラクトの実行準備
45                    self.eth.set_event_contract(rec.func_id, rec.func.address, rec.func.abi)
46
47            if DEBUG:
48                for rec in self.events:
49                    print(rec,id,rec.method,rec.url,rec.call,rec.func)
50
51
52        # デストラクタ
53        def __del__(self):
54            print("Destructor of Event")
55
56
57        # コントラクト実行
58        def call_contract(self, evt, request, response):
59            print("call_contract", evt, request, response)
60            print("event = ", evt.id, evt.name)
61
62            account = None
63            if evt.func:
64                params = {}
65                params['name'] = evt.id
66                params['url'] = request.url.path
67                params['method'] = request.method
68                params['user'] = request.client.host
69
70                if 'user' in request.query_params:
71                    params['user'] = request.query_params['user']
72                if 'account' in request.query_params:
73                    params['account'] = request.query_params['account'].lower()
74                    account = params['account']
75                if 'need_token' in request.query_params:
76                    params['need_token'] = request.query_params['need_token']
77
78                print("PARAM for eth.put_event_contract:", evt.func_id, params)
79                self.eth.put_event_contract(evt.func_id, account, params)
80
```

```
            res = {}

            return res


    # イベント実行チェック
    def check_event(self, request, check_point):
        # リクエストと情報取り出し
        method = request.method
        path = request.url.path
        user = request.client.host
        print(method, path, user)

        # 実行するイベントを探す
        event_list = []
        for evt in self.events:
            if DEBUG : print(evt.id, evt.method, evt.url, method, path, check_point)
            if check_point.lower() == 'pre' and evt.act_pre != True:
                continue
            elif check_point.lower() == 'post' and evt.act_post != True:
                continue

            if evt.method != '*' and evt.method != method:
                continue
            if evt.url != '*' and path.startswith(evt.url) != True:
                continue

            if DEBUG :print(evt.id, "in event list")

            event_list.append(evt)

        return event_list


    # 処理前イベント
    def pre_event(self, request):
        if DEBUG : print("pre_event", request)

        # 実行イベントを探す
        event_list = self.check_event(request, 'pre')
        if DEBUG : print(event_list)

        # イベント実行
        res = {}
        for evt in event_list:
            r = evt.call(evt, request, None)
            if r : res[evt.id] = r

        return res


    # 処理後イベント
    def post_event(self, request, response):
        if DEBUG : print("post_event", request, response)
```

```
135
136             # 実行イベントを探す
137             event_list = self.check_event(request, 'post')
138             if DEBUG : print(event_list)
139
140             # イベント実行
141             res = {}
142             for evt in event_list:
143                 r = evt.call(evt, request, response)
144                 if r : res[evt.id] = r
145
146             return res
```

リスト2 APP/event.py

・put(account, keys, values)

入力:

account	処理を実行するアカウント
keys	パラメータ名の配列
values	パラメータ値の配列

※keysとvaluesは同一インデックスの値がペアで、key.value形式のパラメータ情報となる

出力:なし

・get(keys, values)

入力:

keys	パラメータ名の配列
values	パラメータ値の配列

※keysとvaluesは同一インデックスの値がペアで、key.value形式のパラメータ情報となる

出力:任意の構造体データ(※辞書形式で表現可能な形式)

リスト3

```
 1  // SPDX-License-Identifier: MIT
 2  pragma solidity ^0.8.7;
 3
 4
 5  contract MyAuditEvent {
 6
 7      struct Audit {
 8         string url;
 9         string method;
10         string user;
11         uint timestamp;
12         address  account;
```

```
    }

    struct AuditInfo {
        string name;
        Audit[] ad_list;
    }

    mapping(string => AuditInfo) audit_list;

    function put(address account, string[] calldata keys, string[] calldata values) public {
        string memory name;
        string memory url;
        string memory method;
        string memory user;
        for (uint i = 0; i < keys.length; i++) {
            if (keccak256(bytes(keys[i])) == keccak256(bytes('name'))) {
                name = values[i];
            }
            else if (keccak256(bytes(keys[i])) == keccak256(bytes('url'))) {
                url = values[i];
            }
            else if (keccak256(bytes(keys[i])) == keccak256(bytes('method'))) {
                method = values[i];
            }
            else if (keccak256(bytes(keys[i])) == keccak256(bytes('user'))) {
                user = values[i];
            }
        }

        AuditInfo storage ad_info = audit_list[name];
        ad_info.name = name;
        Audit memory ad = Audit(url, method, user, block.timestamp, account);
        ad_info.ad_list.push(ad);
    }

    function get(string[] calldata keys, string[] calldata values) public view returns(AuditInfo memory){
        // 今回は、取得範囲の指定などはオミット
        string memory name;
        for (uint i = 0; i < keys.length; i++) {
            if (keccak256(bytes(keys[i])) == keccak256(bytes('name'))) {
                name = values[i];
            }
        }
        return audit_list[name];
    }

}
```

リスト4 MyAuditEvent.sol

```
-------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------
mysql> select id,address from ds_function where id = 'MyAuditEvent';
+--------------+--------------------------------------------+
| id           | address                                    |
+--------------+--------------------------------------------+
| MyAuditEvent | 0xD9f6E51f96d488175cFf9bC509E9AFb588925259 |
+--------------+--------------------------------------------+
1 row in set (0.00 sec)

mysql>
mysql> select abi from ds_function where id = 'MyAuditEvent';
-------------------------------------------------------------------------------+
| abi
-------------------------------------------------------------------------------+
| [
        {
                "inputs": [
                        {
                                "internalType": "address",
                                "name": "account",
                                "type": "address"
~~ 省略 ~~

                "stateMutability": "view",
                "type": "function"
        }
] |
-------------------------------------------------------------------------------+
1 row in set (0.01 sec)

mysql>
-------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------
```

リスト5

```
-------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------
(mydata_server) % cd LIB
(mydata_server) % ./setevent.py --id AUDIT --name "audit event" --url '*' --method '*' --act_post --func_id MyAuditEvent
-------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------
```

リスト6

```
-------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------
mysql> select * from ds_event;
+-------+-------------+------+--------+---------+----------+--------------+
| id    | name        | url  | method | act_pre | act_post | func_id      |
+-------+-------------+------+--------+---------+----------+--------------+
| AUDIT | audit event | *    | *      |       0 |        1 | MyAuditEvent |
+-------+-------------+------+--------+---------+----------+--------------+
1 row in set (0.00 sec)

mysql>
-------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------
```

リスト7

```
-------------------------------------------------------------------------------------------------------------------------------------------------------
-----------------------
#!/usr/bin/env python
# -*- coding: utf-8 -*-


import sys
import json
import datetime

from LIB import db
from LIB import ethereum

# Ethereumへのアクセス情報取得
s = db.DB()
eth_rec = s.select(db.DsEthereum, "Ethereum")
print("eth_rec=",eth_rec)
if eth_rec != None:
    url = "http://" + eth_rec.address + ":" + str(eth_rec.port)
    account = eth_rec.account
    password = eth_rec.password
    private_key = eth_rec.private_key
else:
    url = "http://192.168.3.3:8545"
    account = "0x70f9e0445a572d62eabf6d6669f69283558d7e2f"
    password = "dataprovider"
    private_key = "c2f7b10487844a8d0f708a2397a63e0aa2b2695351775fe5548557bc71280e28"

eth = ethereum.Ethereum(url, account, password, private_key)

# コントラクト設定
```

```
event_id = "AUDIT"
func_id = "MyAuditEvent"
func_rec = s.select(db.DsFunction, func_id)
eth.set_event_contract(func_id, func_rec.address, func_rec.abi)

params = {"name":event_id}
res = eth.get_event_contract(func_id, params)
print(res)
```

リスト8

```
(mydata_server) % TOP=$HOME/Desktop/tsuyo/CQ/mydata_server/
(mydata_server) % export PYTHONPATH="$TOP:%TOP/LIB:$TOP/APP:$PYTHONPATH"
(mydata_server) % export MYDATASERVER_LIB_CONFIG_PATH=$TOP/LIB
(mydata_server) % ./get_event_contract.py
Constructor of DB
select
eth_rec= <LIB.db.DsEthereum object at 0x7ff46d5b3340>
Constructor of Ethereum
select
('', [])        ★ <- ここが出力で、まだ何もない
Destructor of DB
db_disconnect
Destructor of Ethereum
%
```

リスト9

```
(mydata_server) % ./get_event_contract.py

Constructor of DB
select
eth_rec= <LIB.db.DsEthereum object at 0x7fb30ddb3370>
Constructor of Ethereum
select
('AUDIT', [('/docs', 'GET', '192.168.3.3', 1663134080, '0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/openapi.json', 'GET',
```

```
'192.168.3.3', 1663134088, '0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/data/', 'GET', '192.168.3.3', 1663134159,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f')])
Destructor of DB
db_disconnect
Destructor of Ethereum
(mydata_server) %
```

```
1. ('/docs', 'GET', '192.168.3.3', 1663134080, '0x70F9E0445A572D62EaBF6d6669f69283558D7E2f')
2. ('/openapi.json', 'GET', '192.168.3.3', 1663134088, '0x70F9E0445A572D62EaBF6d6669f69283558D7E2f')
3. ('/data/', 'GET', '192.168.3.3', 1663134159, '0x70F9E0445A572D62EaBF6d6669f69283558D7E2f')
```

```
(mydata_server) % ./get_event_contract.py

Constructor of DB
select
eth_rec= <LIB.db.DsEthereum object at 0x7f87c05c4400>
Constructor of Ethereum
select
('AUDIT', [('/docs', 'GET', '192.168.3.3', 1663134080, '0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/openapi.json', 'GET',
'192.168.3.3', 1663134088, '0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/data/', 'GET', '192.168.3.3', 1663134159,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/channel/html', 'GET', '192.168.3.3', 1663134774,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/channel.html', 'GET', '192.168.3.3', 1663134814,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/js/mylib.js', 'GET', '192.168.3.3', 1663134815,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/css/channel.css', 'GET', '192.168.3.3', 1663134820,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/channel/', 'GET', '192.168.3.3', 1663134824,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f')])
Destructor of DB
db_disconnect
Destructor of Ethereum
(mydata_server) %
```

```
------------------------------------------------------------------------------------------------------------------------
----------------------
(mydata_server) % ./get_event_contract.py

Constructor of DB
select
eth_rec= <LIB.db.DsEthereum object at 0x7fc9066e03a0>
Constructor of Ethereum
select
('AUDIT', [('/docs', 'GET', '192.168.3.3', 1663134080, '0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/openapi.json', 'GET',
'192.168.3.3', 1663134088, '0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/data/', 'GET', '192.168.3.3', 1663134159,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/channel/html', 'GET', '192.168.3.3', 1663134774,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/channel.html', 'GET', '192.168.3.3', 1663134814,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/js/mylib.js', 'GET', '192.168.3.3', 1663134815,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/css/channel.css', 'GET', '192.168.3.3', 1663134820,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/channel/', 'GET', '192.168.3.3', 1663134824,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/data.html', 'GET', '192.168.3.3', 1663134861,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/dataserver/js/chart.js', 'GET', '192.168.3.3', 1663134863,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/data/', 'GET', '192.168.3.3', 1663134871,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/data/', 'GET', '192.168.3.3', 1663134872,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/data/', 'GET', '192.168.3.3', 1663134932,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/data/', 'GET', '192.168.3.3', 1663134989,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/channel', 'POST', '192.168.3.3', 1663135006,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/channel/', 'POST', '192.168.3.3', 1663135008,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f'), ('/channel/', 'GET', '192.168.3.3', 1663135013,
'0x70F9E0445A572D62EaBF6d6669f69283558D7E2f')])
Destructor of DB
db_disconnect
Destructor of Ethereum
(mydata_server) %
------------------------------------------------------------------------------------------------------------------------
----------------------
```

リスト13