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


import sys
import json
import datetime

from LIB import db
from LIB import influxdb
from LIB import storage

DEBUG = False # True

## 作成
def create_channel(id, name, owner):
    print("create_channel:" + name)

    s = db.DB()

    # チャネル存在確認
    c = s.select(db.DsChannel, id)
    print(c)
    if c != None:
        print("Channel " + id + " is exists.")
        return None

    # チャネル登録
    c = db.DsChannel(id=id, name=name, owner=owner)
    s.insert(c)
    print(s)

    # InfluxDB に Bucket を作成する
    idb_rec = s.select(db.DsInfluxDB, "InfluxDB")
    print("idb_rec=",idb_rec)
    if idb_rec != None:
        url = "http://" + idb_rec.address + ":" + str(idb_rec.port)
        print("url = " + url)
        idb = influxdb.InfluxDB(url, idb_rec.organization, idb_rec.token)
        print(idb)
```

```
41              idb.create_bucket(id)
42
43          # Storage(MinIO)に Bucket を作成する
44          st_rec = s.select(db.DsStorage, "MinIO")
45          print("st_rec=",st_rec)
46          if st_rec != None:
47              url = "http://" + st_rec.address + ":" + str(st_rec.port)
48              print("url = " + url)
49              st = storage.Storage(url, st_rec.access_key_id, st_rec.secret_access_key)
50              print(st)
51              st.create_bucket(id)
52
53          return c
54
55
56      ## 削除
57      def delete_channel(id):
58          print("delete_channel:" + id)
59          s = db.DB()
60          c = s.select(db.DsChannel, id)
61          print(c)
62          if c == None:
63              print("Channel " + id + " is NOT exists.")
64              return None
65          s.delete(c)
66
67          # InfluxDB の Bucket を削除する
68          idb_rec = s.select(db.DsInfluxDB, "InfluxDB")
69          print("idb_rec=",idb_rec)
70          if idb_rec != None:
71              url = "http://" + idb_rec.address + ":" + str(idb_rec.port)
72              print("url = " + url)
73              idb = influxdb.InfluxDB(url, idb_rec.organization, idb_rec.token)
74              idb.delete_bucket(id)
75
76          # Storage(MinIO)の Bucket を削除する
77          st_rec = s.select(db.DsStorage, "MinIO")
78          print("st_rec=",st_rec)
79          if st_rec != None:
80              url = "http://" + st_rec.address + ":" + str(st_rec.port)
```

```
81                  print("url = " + url)
82                  st = storage.Storage(url, st_rec.access_key_id, st_rec.secret_access_key)
83                  print(st)
84                  st.delete_bucket(id)
85
86
87      ## 一覧
88      def list_channel():
89          print("list_channel")
90          s = db.DB()
91          channel = s.list(db.DsChannel)
92          for c in channel:
93              print(c.name,c.id)
94
95          return channel
96
97
98      ## 情報取得
99      def get_channel(id):
100         print("get_channel")
101         s = db.DB()
102         c = s.select(db.DsChannel, id)
103         print(c)
104         return c
105
106
107     """
108     create_channel("TTT", "TT name2", "tsuyo2")
109     create_channel("TTT2", "TT name3", "tsuyo3")
110
111     get_channel("TTT22")
112
113     channel = list_channel()
114     for c in channel:
115         print("L", c.id,c.name,c.owner)
116
117     change_channel_name("TTT", "UPDATED NAE")
118     channel = list_channel()
119     for c in channel:
120         print("L2", c.id,c.name,c.owner)
```

```
121     delete_channel("TTT")
122     channel = list_channel()
123     for c in channel:
124         print("L3", c.id,c.name,c.owner)
125
126     delete_channel("TTT2")
127     """
```