LAPORAN PRAKTIKUM
Internet Of Things

MTQQ SIMULATOR

Febyanti Hutahaean
11323055
D-III Teknologi Informasi

INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI
2024/2025

File simulator.py

- **Code Program**

mqtt-simulator > simulator.py > Simulator > load_topics

```python
import json
from topic import Topic
from data_classes import BrokerSettings, ClientSettings

class Simulator:
    def __init__(self, settings_file):
        self.default_client_settings = ClientSettings(
            clean=True,
            retain=False,
            qos=2,
            time_interval=10
        )
        self.topics = self.load_topics(settings_file)

    def read_client_settings(self, settings_dict: dict, default: ClientSettings):
        return ClientSettings(
            clean=settings_dict.get('CLEAN_SESSION', default.clean),
            retain=settings_dict.get('RETAIN', default.retain),
            qos=settings_dict.get('QOS', default.qos),
            time_interval= settings_dict.get('TIME_INTERVAL', default.time_interval)
        )

    def load_topics(self, settings_file):
        topics = []
        with open(settings_file) as json_file:
            config = json.load(json_file)
            broker_settings = BrokerSettings(
                url=config.get('BROKER_URL', 'localhost'),
                port=config.get('BROKER_PORT', 1883),
                protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
            )
            broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
            # read each configured topic
            for topic in config['TOPICS']:
                topic_data = topic['DATA']
                topic_payload_root = topic.get('PAYLOAD_ROOT', {})
                topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
                if topic['TYPE'] == 'single':
                    # create single topic with format: /{PREFIX}
                    topic_url = topic['PREFIX']
                    topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
                elif topic['TYPE'] == 'multiple':
                    # create multiple topics with format: /{PREFIX}/{id}
                    for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
                        topic_url = topic['PREFIX'] + '/' + str(id)
                        topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
                elif topic['TYPE'] == 'list':
                    # create multiple topics with format: /{PREFIX}/{item}
                    for item in topic['LIST']:
                        topic_url = topic['PREFIX'] + '/' + str(item)
                        topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
        return topics

    def run(self):
        for topic in self.topics:
            print(f'Starting: {topic.topic_url} ...')
            topic.start()
        for topic in self.topics:
            # workaround for Python 3.12
            topic.join()

    def stop(self):
        for topic in self.topics:
            print(f'Stopping: {topic.topic_url} ...')
            topic.stop()

```

■ **Penjelasan** :

1. ```
   import json
   ```
   - Ini untuk mengambil alat (library) yang bisa membaca dan mengolah data dalam format JSON. JSON itu kayak format file yang banyak dipakai buat menyimpan data, misalnya pengaturan aplikasi.
2. ```
   from topic import Topic
   ```
   - Ini untuk mengambil kelas Topic dari file lain. Kelas Topic ini dipakai buat mengatur topik-topik yang ada dalam simulasi nanti.
3. ```
   from data_classes import BrokerSettings, ClientSettings
   ```
   - Ini juga untuk mengambil dua kelas, yaitu BrokerSettings dan ClientSettings, yang isinya pengaturan tentang broker (misalnya server yang jadi tempat komunikasi) dan pengaturan klien (misalnya perangkat yang terhubung ke server).
4. ```
   class Simulator:
   ```
   - Ini adalah tempat utama, yaitu kelas Simulator. Semua yang berkaitan dengan menjalankan simulasi ada di sini.
5. ```
   def __init__(self, settings_file):
   ```
   - Ini adalah bagian awal, yang bakal dijalanin pertama kali pas kita buat objek Simulator. Dia butuh file pengaturan (settings) yang isinya konfigurasi apa aja yang harus diatur.
6. ```
   self.default_client_settings = ClientSettings(
       clean=True,
       retain=False,
       qos=2,
       time_interval=10
   )
   self.topics = self.load_topics(settings_file)
   ```
   - Ini bikin pengaturan default buat klien, yang mencakup hal-hal kayak apakah sesi harus bersih, jenis komunikasi yang dipakai, dan interval waktu.
7. ```
   self.topics = self.load_topics(settings_file)
   ```
   - Setelah itu, dia panggil fungsi load_topics buat baca dan bikin semua topik yang ada di file pengaturan.
   ```

8.
```
def read_client_settings(self, settings_dict: dict, default: ClientSettings):
```
   - Fungsi ini baca pengaturan dari file atau data yang dikasih, terus kembalikan pengaturan klien yang udah digabung sama pengaturan default.

9.
```
return ClientSettings(
    clean=settings_dict.get('CLEAN_SESSION', default.clean),
    retain=settings_dict.get('RETAIN', default.retain),
    qos=settings_dict.get('QOS', default.qos),
    time_interval= settings_dict.get('TIME_INTERVAL', default.time_interval)
)
```
   - Ini bagian yang ngembaliin pengaturan klien yang udah dibaca dari file.

10.
```
def load_topics(self, settings_file):
```
   - Fungsi ini baca file pengaturan, terus buat semua topik yang perlu ada di simulasi. Topik-topik ini penting buat komunikasi nanti.

11.
```
topics = []
```
   - Ini bikin daftar kosong buat nampung semua topik yang bakal dibuat.

12.
```
with open(settings_file) as json_file:
```
   - Ini buka file pengaturan yang kita kasih tadi dan baca isinya.

13.
```
config = json.load(json_file)
```
   - Ini buat baca file JSON tadi dan ubah isinya jadi bentuk data yang bisa dipakai di Python.

14.
```
broker_settings = BrokerSettings(
    url=config.get('BROKER_URL', 'localhost'),
    port=config.get('BROKER_PORT', 1883),
    protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
)
```
   - Ini perintah ambil pengaturan tentang broker (misalnya alamat server dan protokol yang dipakai).

15.
```
broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
```
   - Ini printah buat baca pengaturan klien yang digunakan buat komunikasi ke broker.

16.
```
for topic in config['TOPICS']:
```
   - Sekarang kita bakal mulai baca tiap topik yang ada di file pengaturan, di bagian TOPICS.

17. `topic_data = topic['DATA']`
    - Ambil data yang ada di masing-masing topik.
18. `topic_payload_root = topic.get('PAYLOAD_ROOT', {})`
    - Cek apakah ada pengaturan khusus yang disebut PAYLOAD_ROOT, kalau nggak ada, pakai default kosong.
19. `topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)`
    - Baca pengaturan klien buat topik ini. Jadi, setiap topik bisa punya pengaturan klien sendiri-sendiri.
20. `if topic['TYPE'] == 'single':`
    - Cek jenis topik ini, apakah "single" (satu topik aja).
21. `topic_url = topic['PREFIX']`
    - Kalau jenisnya single, buat URL topik dengan format: /PREFIX (misalnya /home).
22. `topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))`
    - Buat objek Topic baru dan masukin ke daftar topics.
23. `elif topic['TYPE'] == 'multiple':`
    - Kalau jenis topiknya "multiple", artinya bakal ada beberapa topik dengan ID yang beda-beda.
24. `for id in range(topic['RANGE_START'], topic['RANGE_END']+1):`
    - Buat topik dengan ID yang dimulai dari RANGE_START sampai RANGE_END.
25. `topic_url = topic['PREFIX'] + '/' + str(id)`
    - Tambahin ID ke URL topiknya, jadi formatnya misalnya /home/1, /home/2, dan seterusnya.
26. `elif topic['TYPE'] == 'list':`
    - Kalau jenis topiknya "list", artinya bakal ada topik dengan nama-nama yang ada dalam daftar (list).
27. `for item in topic['LIST']:`
    - Perulangan buat baca tiap item dalam list dan buat topik dari setiap item tersebut.

28. `topic_url = topic['PREFIX'] + '/' + str(item)`
    - Gabungin item ke URL topik, misalnya /home/first_item, /home/second_item, dan seterusnya.
29. `return topics`
    - Kembaliin daftar semua topik yang udah dibuat.
30. `def run(self):`
    - Fungsi ini buat mulai menjalankan simulasi, yaitu menjalankan semua topik yang udah dibikin.
31. `for topic in self.topics:`
    - Lakukan perulangan buat mulai setiap topik satu per satu.
32. `topic.start()`
    - Mulai topik tertentu supaya topik itu mulai jalan.
33. `for topic in self.topics:`
    - Perulangan lagi, tapi kali ini buat pastiin semua topik selesai jalan.
34. `topic.join()`
    - Ini buat nunggu sampai topik selesai dijalankan, biar nggak ada yang ketinggalan.
35. `def stop(self):`
    - Fungsi ini buat berhentiin semua topik yang lagi jalan.
36. `for topic in self.topics:`
    - Perulangan buat berhentiin setiap topik.
37. `topic.stop()`
    - Stop topik yang sedang berjalan.

CATATAN :Secara keseluruhan, kode ini ngatur simulasi topik-topik yang dipakai dalam komunikasi sistem, terutama dalam hal pengaturan broker dan kliennya. Jadi, aplikasi ini bisa baca pengaturan dari file, buat topik-topik sesuai dengan pengaturan itu, jalankan simulasi, dan bisa stop kapan aja.