

LAPORAN PRAKTIKUM Internet Of Things

MQTT SERVER



**Franklyn Aldo Ignatia Lumbantoruan
11323010
D-III Teknologi Informasi**

**INSTITUT TEKNOLOGI DEL FAKULTAS
VOKASI**

2024/2025

File simulator.py

Code Program

```
mqtt-simulator > simulator.py > Simulator > load_topics
1 import json
2 from topic import Topic
3 from data_classes import BrokerSettings, ClientSettings
4
5 class Simulator:
6     def __init__(self, settings_file):
7         self.default_client_settings = ClientSettings(
8             clean=True,
9             retain=False,
10            qos=2,
11            time_interval=10
12        )
13        self.topics = self.load_topics(settings_file)
14
15    def read_client_settings(self, settings_dict: dict, default: ClientSettings):
16        return ClientSettings(
17            clean=settings_dict.get('CLEAN_SESSION', default.clean),
18            retain=settings_dict.get('RETAIN', default.retain),
19            qos=settings_dict.get('QOS', default.qos),
20            time_interval=settings_dict.get('TIME_INTERVAL', default.time_interval)
21        )
22
23    def load_topics(self, settings_file):
24        topics = []
25        with open(settings_file) as json_file:
26            config = json.load(json_file)
27            broker_settings = BrokerSettings(
28                url=config.get('BROKER_URL', 'localhost'),
29                port=config.get('BROKER_PORT', 1883),
30                protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
31            )
32            broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
33            # read each configured topic
34            for topic in config['TOPICS']:
35                topic_data = topic['DATA']
36                topic_payload_root = topic.get('PAYLOAD_ROOT', {})
37                topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
38                if topic['TYPE'] == 'single':
39                    # create single topic with format: /(PREFIX)
40                    topic_url = topic['PREFIX']
41                    topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
42                elif topic['TYPE'] == 'multiple':
43                    # create multiple topics with format: /(PREFIX)/(id)
44                    for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
45                        topic_url = topic['PREFIX'] + '/' + str(id)
46                        topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
47                elif topic['TYPE'] == 'list':
48                    # create multiple topics with format: /(PREFIX)/(item)
49                    for item in topic['LIST']:
50                        topic_url = topic['PREFIX'] + '/' + str(item)
51                        topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
52            return topics
53
54    def run(self):
55        for topic in self.topics:
56            print(f'Starting: {topic.topic_url} ...')
57            topic.start()
58        for topic in self.topics:
59            # workaround for Python 3.12
60            topic.join()
61
62    def stop(self):
63        for topic in self.topics:
64            print(f'Stopping: {topic.topic_url} ...')
65            topic.stop()
66
```

Penjelasan :

1. `import json`

- Ini untuk mengambil alat (library) yang bisa membaca dan mengolah data dalam format JSON. JSON itu seperti 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 dijalankan pertama kali pas kita buat objek Simulator. Dia butuh file pengaturan (settings) yang isinya konfigurasi apa aja yang harus diatur.

```
self.default_client_settings = ClientSettings(  
    clean=True,  
    retain=False,  
    qos=2,  
    time_interval=10  
)  
self.topics = self.load_topics(settings_file)
```

- 6.
- 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 membaca pengaturan dari file atau data yang dikasih, terus kembalikan pengaturan klien yang udah digabung sama pengaturan default.

```
    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)  
    )
```

9.

- Ini bagian yang mengembalikan pengaturan klien yang sudah dibaca dari file.

10. `def load_topics(self, settings_file):`

- Fungsi ini membaca file pengaturan, terus buat semua topik yang perlu ada di simulasi. Topik-topik ini penting untuk komunikasi nanti.

11. `topics = []`

- Ini membuat daftar kosong untuk menampung semua topik yang bakal dibuat.

12. `with open(settings_file) as json_file:`

- Ini membuka file pengaturan yang kita kasih tadi dan baca isinya.

13. `config = json.load(json_file)`

- Ini untuk membaca file JSON tadi dan ubah isinya jadi bentuk data yang bisa dipakai di Python.

```
broker_settings = BrokerSettings(  
    url=config.get('BROKER_URL', 'localhost'),  
    port=config.get('BROKER_PORT', 1883),  
    protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311  
)
```

14.

- Ini perintah ambil pengaturan tentang broker (misalnya alamat server dan protocol yang dipakai).

```
broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
```

15.

- Ini perintah untuk membaca pengaturan klien yang digunakan buat komunikasi ke broker.

```
for topic in config['TOPICS']:
```

16.

- Sekarang kita akan mulai membaca tiap topik yang ada di file pengaturan, di bagian TOPICS.

```
    topic_data = topic['DATA']
```

17.

- Ambil data yang ada di masing-masing topik.

```
    topic_payload_root = topic.get('PAYLOAD_ROOT', {})
```

18.

- Cek apakah ada pengaturan khusus yang disebut PAYLOAD_ROOT, kalau tidak ada, gunakan default kosong.

```
    topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
```

19.

- Baca pengaturan klien buat topik ini. Jadi, setiap topik bisa punya pengaturan klien sendiri-sendiri.

```
    if topic['TYPE'] == 'single':
```

20.

- Cek jenis topik ini, apakah "single" (satu topik aja).

```
        topic_url = topic['PREFIX']
```

21.

- 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 masukan ke daftar topics.

23. `elif topic['TYPE'] == 'multiple':`

- Kalau jenis topiknya "multiple", artinya akan ada beberapa topik dengan ID yang berbeda-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 akan ada topik dengan nama-nama yang ada dalam daftar (list).

27. `for item in topic['LIST']:`

- Perulangan untuk membaca tiap item dalam list dan buat topik dari setiap item tersebut.

28. `topic_url = topic['PREFIX'] + '/' + str(item)`

- Gabungkan item ke URL topik, misalnya /home/first_item, /home/second_item, dan seterusnya.

29. `return topics`

- Kembalikan daftar semua topik yang udah dibuat.

30. `def run(self):`

- Fungsi ini untuk memulai menjalankan simulasi, yaitu menjalankan semua topik yang sudah dibuat.

31. `for topic in self.topics:`

- Lakukan perulangan buat mulai setiap topik satu per satu.

32. `topic.start()`

- Mulai topik tertentu supaya topiknya mulai jalan.

33. `for topic in self.topics:`

- Perulangan lagi, tapi kali ini untuk memastikan semua topik selesai jalan.

34. `topic.join()`

- Ini untuk nunggu sampai topik selesai dijalankan, agar tidak ada yang ketinggalan.

35. `def stop(self):`

- Fungsi ini untuk menghentikan semua topik yang lagi jalan.

36. `for topic in self.topics:`

- Perulangan untuk menghentikan setiap topik.

37. `topic.stop()`

- Untuk menstop topik yang sedang berjalan.