

LAPORAN PRAKTIKUM INTERNET OF THINGS

“MQTT SERVER”



Daniel Haganta Ginting

11323002

TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI DEL

FAKULTAS VOKASI

TA 2024/2025

```

mqtt-simulator > simulator.py
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()

```

Kode ini merupakan implementasi dari sebuah simulator yang mengatur topik-topik untuk komunikasi menggunakan protokol MQTT.

Penjelasan:

Import Statements

```
1 import json
2 from topic import Topic
3 from data_classes import BrokerSettings, ClientSettings
```

- json: Digunakan untuk memuat pengaturan dari file JSON.
- Topic: Kelas yang diimpor dari modul topic, yang kemungkinan berfungsi untuk mewakili dan mengelola topik-topik MQTT.
- BrokerSettings dan ClientSettings: Kelas yang diimpor dari modul data_classes, digunakan untuk menyimpan pengaturan broker dan klien MQTT.

Kelas Simulator

```
class Simulator:
    def __init__(self, settings_file):
```

Kelas Simulator bertanggung jawab untuk mengelola seluruh simulasi. Konstruktor `__init__` menerima nama file pengaturan dan memuat topik-topik berdasarkan pengaturan tersebut.

Pengaturan Klien Default

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

Menetapkan pengaturan klien MQTT default, termasuk:

- clean: Menentukan apakah sesi bersih.
- retain: Menentukan apakah pesan harus disimpan.
- qos: Quality of Service level.
- time_interval: Interval waktu untuk pengiriman pesan.

Metode read_client_settings

```
def read_client_settings(self, settings_dict: dict, default: ClientSettings):
```

Metode ini membaca pengaturan klien dari dictionary dan mengembalikannya dalam bentuk objek ClientSettings. Jika pengaturan tidak ada, menggunakan pengaturan default.

Metode load_topics

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

Metode ini memuat topik-topik dari file JSON. Ini mencakup:

- Membaca pengaturan broker (URL, port, dan versi protokol).
- Mengiterasi melalui setiap topik yang dikonfigurasi dalam file JSON.
- Membuat objek Topic berdasarkan tipe topik (single, multiple, list).

Metode run

```
def run(self):
```

- Metode ini menjalankan simulasi dengan memulai setiap topik dan menunggu hingga semuanya selesai.
- Memanggil metode start dan join pada setiap topik.

Metode stop

```
def stop(self):
```

Metode ini menghentikan semua topik yang sedang berjalan dengan memanggil metode stop pada setiap topik.