LAPORAN PRAKTIKUM
INTERNET OF THINGS

MQTT Server

Ayu Enissa Maretty Sinaga
11323040
DIII-Teknologi Informasi

INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI
Tahun Ajaran 2024/2025

1. Imports

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

- json: Untuk membaca file konfigurasi dalam format JSON.
- Topic: Kelas yang merepresentasikan topik MQTT. Kemungkinan berisi metode untuk memulai (start) dan menghentikan (stop) topik.
- BrokerSettings dan ClientSettings: Kelas data yang digunakan untuk menyimpan pengaturan broker dan klien MQTT.

2. Konstruktor

```
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)
```

- settings_file: File JSON yang memuat konfigurasi simulator.
- self.default_client_settings: Pengaturan default untuk klien MQTT.
    - clean=True: Membuat sesi baru setiap kali terhubung ke broker.
    - retain=False: Pesan tidak akan disimpan di broker setelah dikirim.
    - qos=2: QoS level 2, menjamin pesan dikirim tepat satu kali.
    - time_interval=10: Interval waktu pengiriman pesan dalam detik.
- self.topics: Daftar objek topik yang dimuat dari konfigurasi

3. Metode 1

```
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)
    )
```

- Membaca pengaturan klien dari sebuah dictionary.
- Jika kunci tertentu tidak ada dalam dictionary, nilai default akan digunakan.
    - **CLEAN_SESSION**: Apakah klien menggunakan sesi bersih.
    - **RETAIN**: Menentukan apakah pesan harus disimpan di broker.
    - **QOS**: QoS level untuk pengiriman pesan.
    - **TIME_INTERVAL**: Waktu antar pengiriman pesan.

4. Load_topics

```python
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
```

- topics: Daftar objek topik yang akan dibuat.
- BrokerSettings: Membaca konfigurasi broker dari JSON:
    - BROKER_URL: URL broker MQTT.
    - BROKER_PORT: Port broker.
    - PROTOCOL_VERSION: Versi protokol MQTT.

Jenis Topik:

a. single:
- Membuat satu topik dengan format /{PREFIX}.

b. multiple:
- Membuat rentang topik dari RANGE_START hingga RANGE_END dengan format /{PREFIX}/{id}.

c. list:
- Membuat beberapa topik dari daftar LIST dengan format /{PREFIX}/{item}.

5. Run

```python
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()
```

- Memulai semua topik dalam daftar self.topics.
- start(): Memulai topik (misalnya, membuka koneksi MQTT, mulai mengirim pesan).
- join(): Memastikan thread dari topik selesai sebelum melanjutkan.

6. Stop

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

- Menghentikan semua topik.
- **stop()**: Menghentikan proses topik (misalnya, menutup koneksi MQTT).

7. Kesimpulan
   - Simulator ini digunakan untuk:
   - Membaca konfigurasi broker, klien, dan topik dari file JSON.
   - Mengatur berbagai jenis topik MQTT (single, multiple, list).
   - Memulai dan menghentikan simulasi topik-topik MQTT sesuai pengaturan.

   Fungsi Utama

   - run(): Memulai simulasi.
   - stop(): Menghentikan simulasi.