LAPORAN PRAKTIKUM

IOT

MQTT SIMULATOR

Kevin Christian B. Rumapea
11323012
DIII-Teknologi Informasi

1. **Konstruktor __init__**

```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)
```

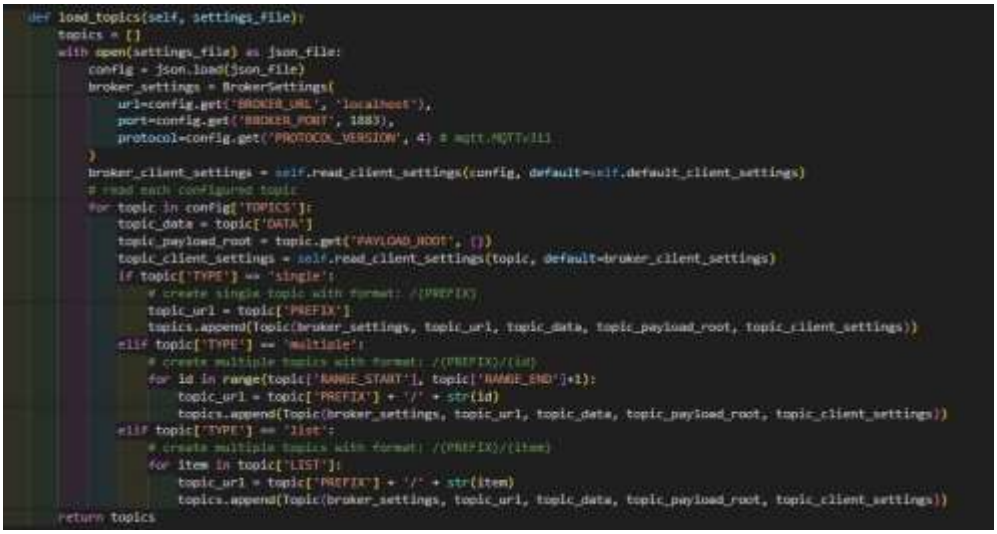
Penjelasan:
Kelas Simulator dirancang untuk mensimulasikan komunikasi dengan broker MQTT berdasarkan pengaturan yang diberikan dalam file konfigurasi JSON. **Konstruktor __init__** bertugas menginisialisasi pengaturan default untuk klien MQTT menggunakan ClientSettings. Nilai default seperti clean=True (mengaktifkan sesi bersih), retain=False (tidak menyimpan pesan), qos=2 (kualitas layanan maksimal), dan time_interval=10 detik digunakan jika nilai tertentu tidak ditentukan dalam file konfigurasi. Selain itu, konstruktor juga memanggil metode load_topics untuk membaca file JSON dan menghasilkan daftar topik MQTT yang akan digunakan dalam simulator.

2. **read_client_settings**

```python
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)
    )
```

Penjelasan:
Metode read_client_settings bertugas membaca pengaturan klien dari dictionary konfigurasi. Jika nilai tertentu tidak ditemukan dalam dictionary, metode ini akan menggunakan nilai default dari parameter default. Misalnya, jika pengaturan CLEAN_SESSION tidak ditemukan, nilai default.clean (yang diatur pada konstruktor) akan digunakan. Metode ini kemudian mengembalikan sebuah objek ClientSettings dengan nilai pengaturan yang telah diperoleh.

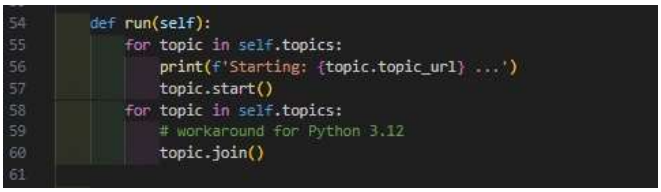
3. **load_topics**



Penjelasan:

Metode load_topics bertanggung jawab untuk membaca file konfigurasi JSON yang diberikan melalui settings_file dan menghasilkan daftar topik MQTT. Pertama, file JSON dibuka dan dimuat menggunakan json.load. Setelah itu, pengaturan broker (seperti BROKER_URL, BROKER_PORT, dan PROTOCOL_VERSION) dikonfigurasi melalui objek BrokerSettings. Selanjutnya, pengaturan klien broker dibaca menggunakan read_client_settings. Setiap topik dalam file konfigurasi diproses berdasarkan tipe-nya. Jika tipe adalah single, hanya satu topik yang dibuat dengan format /{PREFIX}. Untuk tipe multiple, topik dibuat berdasarkan rentang ID dengan format /{PREFIX}/{id}. Sedangkan untuk tipe list, topik dihasilkan dari elemen-elemen dalam daftar LIST dengan format /{PREFIX}/{item}. Semua topik ini disimpan dalam daftar dan dikembalikan.

4. **Metode run**



Penjelasan:

Metode run bertugas untuk memulai semua topik MQTT yang telah dibuat dalam simulator. Pada implementasinya, metode ini pertama-tama menggunakan iterasi untuk memanggil fungsi start() pada setiap objek Topic yang tersimpan di atribut self.topics. Fungsi ini memulai komunikasi untuk setiap topik berdasarkan pengaturan yang telah dikonfigurasi. Setelah semua topik dijalankan, metode ini kemudian memanggil join() untuk setiap topik. Fungsi join() digunakan untuk memastikan bahwa thread yang terkait dengan setiap topik selesai sebelum metode ini berakhir, sehingga operasi dapat dilakukan secara sinkron. Secara keseluruhan, metode run memastikan bahwa semua topik dimulai dengan benar dan eksekusi tidak berlanjut hingga semua operasi topik selesai.

5. **Metode stop**

```
61
62      def stop(self):
63          for topic in self.topics:
64              print(f'Stopping: {topic.topic_url} ...')
65              topic.stop()
66
```

Penjelasan:

Metode stop dirancang untuk menghentikan semua operasi pada topik yang sedang berjalan. Sama seperti pada metode run, metode ini menggunakan iterasi untuk mengakses setiap objek Topic dalam daftar self.topics. Pada setiap iterasi, fungsi stop() dipanggil untuk menghentikan operasi pada topik tersebut. Sebelum menghentikan, sebuah pesan dicetak ke konsol untuk memberi informasi bahwa topik tertentu sedang dihentikan. Metode stop digunakan untuk memastikan bahwa simulator dapat berhenti dengan aman dan memastikan semua topik telah dihentikan dengan benar.

Metode **run** dan **stop** bekerja secara berpasangan untuk mengontrol operasi simulator. Metode run digunakan untuk memulai simulasi, sementara stop memastikan bahwa semua operasi dihentikan dengan cara yang aman dan terkontrol. Kombinasi ini memberikan fleksibilitas dalam pengelolaan siklus hidup simulator MQTT