Laporan Paktikum

INTERNET OF THINGS

Nama: Adriano Rafaelo Lumbantoruan

NIM: 11323035

Program Studi: DIII Teknologi Informasi

INSTITUT TEKNOLOGI DEL

FAKULTAS VOKASI

1. **Inisialisasi**

```
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)
```

Konstruktor kelas Simulator menerima parameter settings_file, yang merupakan jalur ke file JSON berisi pengaturan. Di dalamnya, ada pengaturan klien default yang diinisialisasi dengan nilai-nilai tertentu.

- Pengaturan Klien Default: Menginisialisasi objek ClientSettings dengan nilai default untuk pengaturan klien MQTT seperti clean, retain, qos, dan time_interval.
- Memuat Topik: Memanggil metode load_topics untuk memuat topik-topik dari file pengaturan JSON.
- Parameter : settings_file: Jalur ke file konfigurasi JSON.

- Atribut: default_client_settings: Sebuah instance default dari ClientSettings dengan nilai yang telah ditetapkan (sesi bersih, flag retain, level QoS, dan interval waktu).

topics: Daftar topik yang dibuat dengan memuat konfigurasi dari file pengaturan yang ditentukan. Kelas ini menyimpan pengaturan untuk klien MQTT, yang dapat mencakup: clean: Menentukan apakah sesi bersih harus digunakan (boolean). retain: Menunjukkan apakah pesan harus disimpan (boolean). qos: Level Quality of Service untuk pengiriman pesan (integer, biasanya 0, 1, atau 2) time_interval: Interval waktu untuk pengiriman pesan (integer, dalam detik).

2. Membaca Pengaturan Klien

```
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)
    )
```

Metode ini bertanggung jawab untuk membaca pengaturan klien dari sebuah dictionary. Jika pengaturan tertentu tidak ada, maka nilai default akan digunakan. Ini berguna untuk memastikan bahwa setiap topik dapat memiliki pengaturan klien yang sesuai.
Python. Metode ini membuat dan mengembalikan objek ClientSettings, menggunakan nilai dari dictionary pengaturan atau kembali ke nilai default jika kunci tertentu tidak ada.

3. Memuat Topik (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
```

Metode ini adalah inti dari kelas Simulator. Ia bertugas membaca file JSON dan membuat daftar objek Topic. Berikut adalah langkah-langkah yang dilakukan dalam metode ini:

- Membaca File JSON: Menggunakan fungsi json.load() untuk memuat konten file JSON ke dalam dictionary Python.
- Mengambil Pengaturan Broker: Mengambil informasi broker seperti URL, port, dan versi protokol dari file JSON.
- Membaca Setiap Topik: Melalui loop, setiap topik dibaca dari konfigurasi dan diproses berdasarkan jenisnya (single, multiple, atau list).
- Membuat Objek Topic: Untuk setiap topik, objek Topic dibuat dengan parameter yang sesuai.

4. Menjalankan Simulator (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()
```

Metode run bertanggung jawab untuk memulai semua topik yang telah dimuat. Ini dilakukan

dengan cara berikut:

- **Memulai Setiap Topik**: Melalui loop, setiap topik dipanggil metode start() untuk
- memulai proses terkait.

Menunggu Semua Thread Selesai: Menggunakan metode join() pada setiap topik untuk memastikan bahwa semua thread telah selesai dieksekusi sebelum melanjutkan

5. Menghentikan Simulator (stop)

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

Metode ini digunakan untuk menghentikan semua topik yang sedang berjalan. Ini dilakukan dengan cara berikut:

Menghentikan Setiap Topik: Melalui loop, setiap topik dipanggil metode stop() untuk menghentikan proses terkait.

Kesimpulan

Kelas Simulator adalah alat yang kuat untuk mengelola dan mensimulasikan komunikasi MQTT berdasarkan konfigurasi dinamis. Dengan memanfaatkan kelas ini:

- Anda dapat dengan mudah mengatur berbagai skenario pengujian untuk aplikasi IoT.
- Kelas ini mendukung berbagai jenis topik (single, multiple, list), sehingga fleksibel dalam penggunaannya.
- Dengan menggunakan file JSON sebagai konfigurasi, Anda dapat dengan cepat menyesuaikan pengaturan tanpa perlu mengubah kode sumber secara langsung.

Kelas ini sangat berguna dalam konteks pengembangan dan pengujian aplikasi berbasis MQTT di lingkungan IoT atau sistem komunikasi terdistribusi lainnya.