LAPORAN PRAKTIKUM
INTERNET OF THINGS

“MQTT SERVER”

NATASYA DIAN ELENA SIAHAAN
11323053
TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI
TA 2024/2025

mqtt-simulator > simulator.py

```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)

    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)
        )

    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

    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()

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

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

Penjelasan:

1. **Import Statements**

```
import json
from topic import Topic
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.

2. **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.

3. **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.

4. **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.

5. **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).

6. **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.

7. **Metode stop**

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

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