LAPORAN PRAKTIKUM
Internet of Things

PENJELASAN CODE MQTT

11323019 Febiola Hutagalung

INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI
TAHUN AJARAN 2024/2025

Simlator.py digunakan untuk mengatur dan mengelola simulasi pengiriman pesan menggunakan protokol MQTT (Message Queuing Telemetry Transport).

```
1   import json
2   from topic import Topic
3   from data_classes import BrokerSettings, ClientSettings
4
```

- Import json digunakan untuk membaca file konfigurasi dalam format JSON.
- from topic import Topic: Mengimpor kelas Topic yang tampaknya digunakan untuk merepresentasikan topik MQTT.
- from data_classes import BrokerSettings, ClientSettings: Mengimpor BrokerSettings dan ClientSettings dari modul data_classes yang mungkin merepresentasikan pengaturan broker MQTT dan klien.

```
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)
```

- default_client_settings: Menentukan pengaturan default untuk klien MQTT seperti:
  - clean: Menandakan apakah klien memulai sesi baru setiap kali terhubung (True).
  - retain: Menentukan apakah pesan perlu dipertahankan di broker (False).
  - qos: Quality of Service, di sini nilai 2 artinya tingkat kehandalan tertinggi.
  - time_interval: Interval waktu untuk pengiriman pesan dalam detik (10 detik).
- self.topics: Daftar topik yang akan dimuat dari settings_file menggunakan metode load_topics.

```
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      )
```

Metode ini membaca pengaturan klien dari dictionary settings_dict dan mengembalikannya sebagai objek ClientSettings. Jika nilai tertentu tidak ditemukan, akan digunakan nilai default dari default.

```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
```

- Membaca file konfigurasi settings_file (dalam format JSON).
- broker_settings: Membuat objek BrokerSettings untuk menyimpan informasi seperti URL broker, port, dan versi protokol.
- broker_client_settings: Menggunakan read_client_settings untuk mengatur pengaturan klien untuk broker.
- Konfigurasi untuk setiap topik diambil dari konfigurasi file:
    - Jika tipe topik adalah single, topik ditambahkan dengan URL menggunakan PREFIX.
    - Jika tipe topik adalah multiple, beberapa topik dibuat berdasarkan rentang dari RANGE_START hingga RANGE_END.
    - Jika tipe topik adalah list, beberapa topik dibuat berdasarkan elemen-elemen dalam LIST.

```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 ini digunakan untuk menjalankan setiap topik yang telah dikonfigurasi. Memanggil metode start() pada setiap topik untuk memulai simulasi. Memanggil join() untuk memastikan semua topik berjalan.

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

Digunakan untuk menghentikan semua topik yang sedang berjalan.

Kesimpulan:

Kode ini mensimulasikan pengiriman pesan MQTT ke berbagai topik yang diatur dalam file konfigurasi. Kelas Simulator bertanggung jawab untuk mengelola pengaturan broker, klien, serta menginisiasi dan menghentikan topik yang sesuai. Pengaturan dapat diubah melalui file JSON yang berisi informasi mengenai broker dan topik-topik yang perlu disimulasikan.