

Internet Of Things



Disusun Oleh :

Nama : Felix Natanael Butarbutar

NIM : 11323018

Diploma III Teknologi Informasi IT Del 2024

Sitoluama, Laguboti 2024

Code:

```
simulator.py X
mqtt-simulator > simulator.py > Simulator > read_client_settings
1 import json
2 from topic import Topic
3 from data_classes import BrokerSettings, ClientSettings
4
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)
14
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         )
22
23     def load_topics(self, settings_file):
24         topics = []
25         with open(settings_file) as json_file:
26             config = json.load(json_file)
27             broker_settings = BrokerSettings(
28                 url=config.get('BROKER_URL', 'localhost'),
29                 port=config.get('BROKER_PORT', 1883),
30                 protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
31             )
32             broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
33             # read each configured topic
34             for topic in config['TOPICS']:
35                 topic_data = topic['DATA']
36                 topic_payload_root = topic.get('PAYLOAD_ROOT', {})
37                 topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
38                 if topic['TYPE'] == 'single':
39                     # create single topic with format:://{PREFIX}
40                     topic_url = topic['PREFIX']
41                     topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
42                 elif topic['TYPE'] == 'multiple':
43                     # create multiple topics with format:://{PREFIX}/{id}
44                     for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
45                         topic_url = topic['PREFIX'] + '/' + str(id)
46                         topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
47                 elif topic['TYPE'] == 'list':
48                     # create multiple topics with format:://{PREFIX}/{item}
49                     for item in topic['LIST']:
50                         topic_url = topic['PREFIX'] + '/' + str(item)
51                         topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
52         return topics
53
54     def run(self):
55         for topic in self.topics:
56             print(f'Starting: {topic.topic_url} ...')
57             topic.start()
58         for topic in self.topics:
59             # workaround for Python 3.12
60             topic.join()
61
62     def stop(self):
63         for topic in self.topics:
64             print(f'Stopping: {topic.topic_url} ...')
65             topic.stop()
66
```

Penjelasan:

Kelas *Simulator*

Kelas ini bertugas mengelola simulasi komunikasi berbasis message broker dengan protokol MQTT. Kelas ini mengatur topik-topik (topics) yang akan digunakan dalam komunikasi berdasarkan file konfigurasi JSON. Saat diinisialisasi, kelas ini membuat pengaturan klien MQTT bawaan menggunakan objek *ClientSettings* dan memuat daftar topik menggunakan metode *load_topics*. Setiap topik dikonfigurasi berdasarkan jenisnya (misalnya, single, multiple, atau list) untuk menentukan bagaimana pesan akan diteruskan ke broker MQTT.

Konstruktor *_init_*

Konstruktor kelas bertugas menginisialisasi nilai bawaan untuk pengaturan klien MQTT dengan membuat objek *ClientSettings* yang memiliki properti seperti *clean*, *retain*, *qos*, dan *time_interval*. Selanjutnya, file konfigurasi dibaca melalui metode *load_topics*, yang menghasilkan daftar objek *Topic* berdasarkan isi file JSON tersebut.

Metode *read_client_settings*

Metode ini membaca pengaturan klien MQTT dari sebuah dictionary dan menghasilkan objek *ClientSettings*. Metode ini memastikan bahwa setiap properti seperti *CLEAN_SESSION*, *RETAIN*, *QOS*, dan *TIME_INTERVAL* disesuaikan dengan nilai yang diberikan dalam file konfigurasi. Jika tidak ada nilai yang ditemukan, maka nilai bawaan dari parameter *default* akan digunakan. Hasil akhirnya adalah objek *ClientSettings* dengan nilai yang sudah dikonfigurasi. Metode *load_topics*

Metode ini membaca file JSON konfigurasi dan membuat daftar topik berdasarkan pengaturan yang ada.

Pertama, pengaturan broker MQTT seperti URL, port, dan versi protokol disimpan dalam objek *BrokerSettings*. Pengaturan klien umum dibaca menggunakan metode *read_client_settings*. Selanjutnya, metode ini memproses setiap topik dalam konfigurasi. Jika tipe topik adalah *single*, maka topik dibuat dengan format *{PREFIX}*. Jika tipe multiple, maka beberapa topik dibuat berdasarkan rentang *RANGE_START* hingga *RANGE_END*, dengan format *{PREFIX}/{id}*. Sementara itu, tipe *list* menghasilkan beberapa topik berdasarkan elemen dalam *LIST*, dengan format *{PREFIX}/{item}*. Semua topik yang dihasilkan disimpan dalam daftar dan dikembalikan oleh metode ini.

Metode *run*

Metode ini digunakan untuk memulai semua topik yang telah dikonfigurasi. Untuk setiap topik dalam daftar *self.topics*, metode *start()* dipanggil untuk memulai komunikasi. Setelah itu, metode *join()* dipanggil untuk memastikan semua topik berjalan secara bersamaan. Metode ini penting untuk menjalankan simulasi komunikasi dengan broker MQTT secara serentak.

Metode *stop*

Metode ini digunakan untuk menghentikan semua topik yang sedang berjalan. Untuk setiap topik dalam daftar *self.topics*, metode *stop()* dipanggil untuk mengakhiri komunikasi. Metode ini memastikan semua topik dihentikan dengan rapi.