

LAPORAN PRAKTIKUM IOT



**Prapanca R Panjaitan
11323027
D-III TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI
2024/2025**

```

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

```

```

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

```

Kode di atas adalah implementasi kelas `Simulator` yang bertugas untuk memuat, mengelola, dan menjalankan topik-topik yang dikonfigurasi dalam sebuah file JSON. Berikut penjelasan setiap bagian:

Kelas `Simulator` dimulai dengan metode `init`, yang diinisialisasi dengan file pengaturan (`settings_file`). Metode ini mengatur pengaturan klien default dengan membuat objek `Client Settings`, dimana atribut seperti `clean`, `retain`, `qos`, dan `time_interval` ditentukan nilai awalnya. Setelah itu, ia memuat daftar topik dengan memanggil metode `load_topics`.

Metode `read_client_settings` digunakan untuk membaca pengaturan klien dari dictionary pengaturan yang diberikan. Metode ini memeriksa apakah setiap atribut ada di dictionary (`settings_dict`), dan jika tidak, akan menggunakan nilai dari pengaturan default yang diberikan (`default`).

Metode `load_topics` memuat file pengaturan yang diberikan, membaca isinya sebagai JSON, lalu mengatur konfigurasi broker menggunakan objek `BrokerSettings`. Setelah itu, metode ini membaca setiap topik yang terdefinisi dalam file JSON. Berdasarkan tipe topik (`single`, `multiple`, atau `list`), ia membuat topik-topik dengan format URL yang sesuai dan menambahkannya ke dalam daftar `topics`. Tipe `single` membuat satu topik berdasarkan `PREFIX`, tipe `multiple` membuat beberapa topik dengan rentang ID, dan tipe `list` membuat beberapa topik berdasarkan item dalam daftar `LIST`.

Metode `run` bertugas menjalankan semua topik dalam daftar. Untuk setiap topik, ia mencetak pesan "Starting" dengan URL topik, lalu memanggil metode `start` untuk memulai proses topik. Setelah semua topik dimulai, metode `join` dipanggil untuk memastikan prosesnya selesai.

Metode `stop` bertanggung jawab menghentikan semua topik. Untuk setiap topik, ia mencetak pesan "Stopping" dengan URL topik, lalu memanggil metode `stop` untuk mengakhiri prosesnya.

Kode ini mengatur bagaimana simulator memuat pengaturan dari file JSON, mengelola topik-topik berdasarkan konfigurasi yang diberikan, lalu menjalankan atau menghentikannya sesuai kebutuhan.