HIVEMQ
MQTT CLIENT

Implémentation d'un client MQTT en Java

Introduction:

- *La bibliothèque client **HIVEMQ MQTT** conception moderne de l'implémentation de MQTT pour Java en s'appuyant sur des framework modernes:*
  - ***Netty** pour le traitement du réseautage*
  - ***RxJava** pour gérer le streaming asynchronique des messages.*

- *La bibliothèque fournit des implémentations distinctes d'API:*
  - ***blocage,***
  - ***Asynchrone***

- *fonctionnalités*
  - ***MQTT 3.1.1***
  - ***MQTT 5.0***

Installation bibliothèque « HiveMq »:

- *La bibliothèque HiveMQ est disponible par Gradel:*

```
implementation 'com.hivemq:hivemq-mqtt-client-shaded:1.3.0'
```

Instance de MqttClient: (API asynchrone)

```
private Mqtt3AsyncClient client;

client = MqttClient.builder()
        .useMqttVersion3()
        .serverHost(SERVER_URI)
        .serverPort(PORT)
        .sslWithDefaultConfig()
        .buildAsync();
```

Crée un générateur pour un client MQTT.
- *Retour: le constructeur créé pour un client MQTT.*

interface MqttClient

Utilise MQTT version 3.
Retour: le constructeur pour le Mqtt3Client.

interface MqttClientBuilder

Définit l'hôte du serveur auquel se connecter.
- *Paramètres : l'hôte du serveur.*
- *Retour: le constructeur.*

interface MqttClientBuilderBase

Définit le port du serveur auquel se connecter.
- *Paramètres : port – le port du serveur.*
- *Retour: le constructeur.*

interface MqttClientBuilderBase

Définit la configuration du transport sécurisé sur la configuration par défaut. Cela signifie que les chiffrements et les protocoles par défaut du système sont utilisés.
- *Retour: le constructeur.*

interface MqttClientBuilderBase

Génère le Mqtt3AsyncClient.
- *Retour: le Mqtt3AsyncClient intégré. (CompletableFuture.runAsync())*

interface Mqtt3ClientBuilder

Instance de MqttClient: (API asynchrone)

```
client.connectWith()
      .simpleAuth()
      .username(USER_NAME)
      .password(PASSWORD)
      .applySimpleAuth()
      .send()
      .whenComplete((mqtt3ConnAck, throwable) ->
        …
      );
```

Construit le Mqtt3Connect .
interface Mqtt3AsyncClient

Eléments du CompletableFuture.
`CompletableFuture.complete()`

Exécute le theard associé au CompletableFuture.
`CompletableFuture.get()`

Le theard terminé on récupère les paramètres.
`mqtt3ConnAck` -> returnCode=SUCCESS, sessionPresent=false

```
MqttConnAck{returnCode=SUCCESS, sessionPresent=false}
```

Instance de MqttClient: (API asynchrone)

```
{"end_device_ids":{"device_id":"eui-0025ca0a0000853e","applicatior…
```

```
client.subscribeWith()
      .topicFilter(TOPIC)
      .qos(MqttQos.AT_MOST_ONCE)
      .callback(mqtt3Publish ->
        …
      )
      .send()
      .whenComplete((mqtt3SubAck, throwable) ->
        …
      );
```

AT_MOST_ONCE -> Qos=0
AT_LEAST_ONCE -> Qos=1
EXACTLY_ONCE -> Qos=2

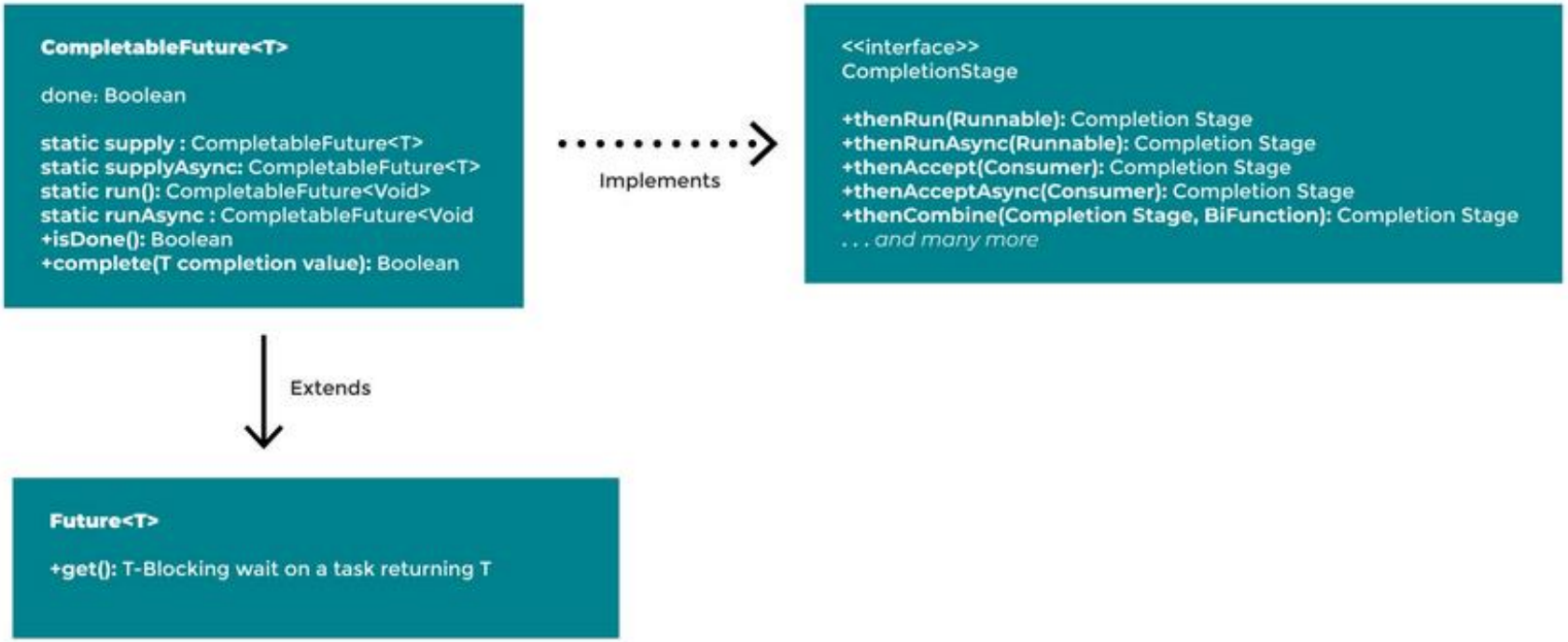
Définit un callback pour les messages en publication correspondants à l'abonnements.

- ***Paramètres : callback pour les messages de publication correspondants.***

Le theard terminé on récupère les paramètres.
throwable -> booléen état de la souscription au topic

```
Mqtt Subscription Ok -> eu1.cloud.thethings.network!
```

Instance de MqttClient: (API asynchrone)



CompletableFuture:

Est utilisé pour la programmation asynchrone en Java. La programmation asynchrone est un moyen d'écrire du code non bloquant en exécutant une tâche sur un thread distinct du thread d'application principal et en informant le thread principal de sa progression, de son achèvement, ou de son échec.