あなたのスキルは社会に役立つ
エンジニアだからできる社会貢献

東日本大震災の発生直後に発足したHack For Japanや「市民が主体となって自分たちの街の課題を技術で解決するコミュニティ作り支援」を掲げるCode for Japanのメンバーを始めとして、日本各地で技術を活用した社会貢献活動が行われています。本連載では、防災や減災、地域の活性化や課題解決、そして人材育成など、「エンジニアだからできる社会貢献」の取り組みをお届けします。

第134回 EU発のオープンソース都市OS、FIWAREとは？

●関 治之（せき はるゆき） Code for Japan https://code4japan.org/
Twitter @hal_sk GitHub @halsk
●潮 周悟（うしお しゅうご） Code for Japan https://code4japan.org/
Mail ushio.s@gmail.com Twitter @ushio_s GitHub @ushios

FIWAREとは

みなさんは、FIWARE（ファイウェア）というオープンソースプラットフォームをご存じでしょうか。FIWAREは、EU（欧州連合）が多大な税金を投入し、官民連携による投資によって開発されているオープンソースのデータ連携用ソフトウェア群です。

EU圏にはさまざまな国が存在しており、人口の大小もさまざまです。投資体力のある国とそうでない国もあります。そこで、国の格差があまり広がらないよう、「特定の国や企業グループ間だけでデータを流通させるのではなく、共通のデータモデルを定義し、標準APIによるデータ交換インターフェースやデータ連携をするためのミドルウェア層をオープンソースで提供することで、オープンなデータ流通プラットフォームを作り上げよう」という目的で、2007年にFIWAREプロジェクトがスタートしました。2016年には、FIWAREを継続して強化・普及を推進するために非営利団体「FIWARE Foundation」が設立されています。

FIWAREは現在、ヨーロッパを中心に、さまざまな国でスマートシティや企業間データ連携のために使われています。日本においても、政府の「デジタル田園都市国家構想推進交付金」で利用が推奨されたことから、FIWAREの採用自治体が増えてきています。

Code for Japanでも、市民参加型まちづくりであるMake our Cityプロジェクト[注1]の中で、FIWAREを拡張したオープンソースのデータ連携プラットフォームをいくつかの自治体に提供しています。このプラットフォームはいくつかの民間企業や自治体とともに開発しており、スマートシティのオープンソースマーケットやコミュニティを作ることを目指しています。

FIWARE Orion Context Broker

FIWAREは特定のソフトウェアを示す言葉ではなく、さまざまなツールの総称です。利用目的に合わせて必要なツールを組み合わせて使います（**図1**）。

簡単に例を挙げますと、コンテキスト処理として、クラウドとエッジ上でIoTサービスを動的に調整する分散実行フレームワークであるFogFlow、コンテキストの永続化などを目的としたCygnusやSTH-Cometなどがあります。

FIWAREの中核となるのは、データの交換を行うコンテキストマネジメント部分です。本節では、コンテキストマネジメントの1つであるFIWARE Orion Context Broker（以下、Orion）について詳しく解説します。

注1） https://www.code4japan.org/activity/moc

NGSIv2 API

OrionはAPIとしてFIWARE NGSI version 2（NGSIv2）を実装しています[注2]。NGSIv1やNGSI-LDなど、別のバージョンも存在していますが、少し違ったものになりますので検索などを行う際はお気をつけください。

●コンテキストのエンティティ

NGSIv2では、1つのデータの塊をエンティティ（Entity）と呼びます。エンティティはidと型（type）を持ち、各エンティティはこれらの組み合わせによって、一意に識別されるようになっています（図2）。

●コンテキストの属性

エンティティは属性（Attributes）を持ち、属性は属性名（attribute name）、属性型（attribute type）、属性値（attribute value）、およびメタデータ（metadata）を持ちます。

●コンテキストのメタデータ

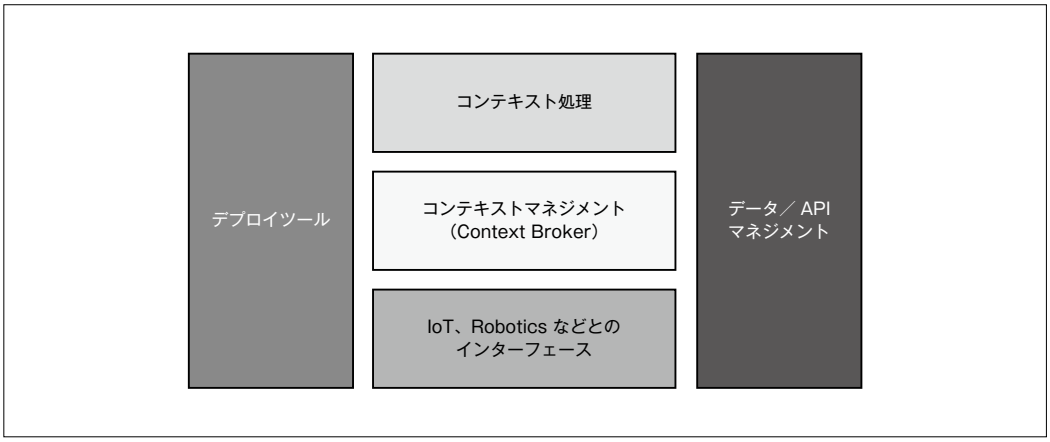
メタデータはメタデータ名（metadata name）、メタデータ型（metadata type）、メタデータ値（metadata value）を持ちますが、ネストされたデータが含まれることを予期していません。ここでは詳しい説明は省きます。

◆ ◆ ◆

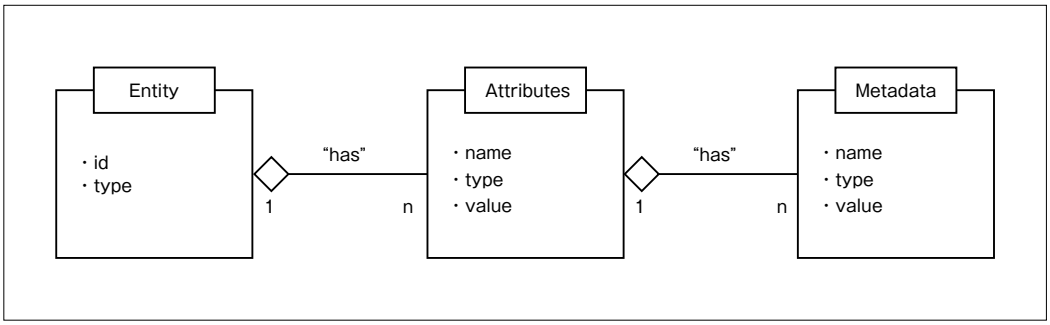
これらをふまえたうえでエンティティを

注2）https://fiware-orion.readthedocs.io/en/1.2.0/user/ngsiv2_implementation_notes/index.html

◆図1　FIWAREプラットフォームの構成要素



◆図2　NGSIデータモデルのおもな要素



※https://fiware-orion.letsfiware.jp/user/orion-api/#context-data-modelling-and-exchangeより引用。

JSONで表記すると、**リスト1**のようになります。

Orionの起動

Orionの実行には、Orionサーバを用意しなければなりません。Code for Japanとして、どなたでも登録、利用していただけるサンドボックス環境[注3]を用意していますが、データ作成・削除などの取得以外の操作に関してはユーザー登録・認証が必要になりますので、ここではローカル環境にDocker Composeを利用し起動する方法を紹介します。

はじめに、docker-compose.ymlを作成します。**リスト2**のように書いたら、**docker compose up**コマンドで起動しましょう[注4]。

起動したことを確認するために、/versionのエンドポイントも叩きます。version情報などが返ってきたら起動は成功です（**図3**）。

CRUD操作

Orionでは、そのインターフェースを通して、REST-APIでよく言われているデータのCRUD操作（作成、読み出し、更新、削除）、またその更新情報の通知など、コンテキスト情報のライフサイクル全体を管理できます。

●作成

作成したいエンティティをJSON形式でPOSTします。**リスト3**は、JavaScriptのコード例です。コードを実行し、201 Createdのステータスコードが返ってくれば、作成に成功しています。

●取得

/作成したエンティティのディレクトリ/登録したエンティティのIDをGETすることにより、エンティティを取得できます。実行し、作成の際に送ったJSONと同じものが返ってきたら取得も成功です。

注3）**https://app.sandbox.makeour.city/**

注4）Make our CityのGitHubにも用意しています。**https://github.com/makeOurCity/docker-compose-orion-v2/blob/main/docker-compose.yml**

◆**リスト1　エンティティの例**

```
{
  “id”: “entityID”,
  “type”: “entityType”,
  “name”: {
    “type”: “Text”,
    “value”: “entityName”
  }
}
```

◆**リスト2　docker-compose.yml**

```
version: "3.5"
services:

  orion:
    image: fiware/orion:3.7.0
    ports:
      - 1026:1026
    command: [ "-dbhost", "mongodb" ]
    environment:
      ORION_MONGO_HOST: mongodb:27018
      ORION_PORT: 1026
      ORION_LOG_LEVEL: INFO
    depends_on:
      - mongodb

  mongodb:
    image: mongo:3.6
    ports:
      - 27018:27018
    volumes:
      - ./data/mongodb:/data/db
      - ./data/configdb:/data/configdb
```

◆**図3　起動確認**

```
$ docker compose up
$ curl localhost:1026/version
{
"orion" : {
  "version" : "3.7.0",
  "uptime" : "0 d, 0 h, 0 m, 13 s",
  "git_hash" : "8b19705a8ec645ba1452cb97847a5615f0b2d3ca"
```

●更新

更新の種類はいくつかあります。

- 1つの属性の値の上書き
- 属性の追加
- エンティティの複数の属性を同時に上書き
- 複数のエンティティの属性を上書き
- 更新演算子を使用しての属性の値の上書き

●削除

削除についても複数種類があります。

- エンティティの属性の削除
- エンティティ自体の削除

◆ ◆ ◆

誌面の都合上簡単な説明に留めましたが、実際のコードもMake our CityのGitHubリポジトリ[注5]に用意していますのでぜひ試してみてください。

その他

CRUD操作以外にも、同時に削除と追加を行う/v2/op/updateなどありますので、ぜひLet's FIWAREのチュートリアル[注6]やAPIリファレンス[注7]などを読んで、実際にContext Brokerを使用してみることをお勧めします。また、ブラウザ上でも利用可能で、APIを簡単に叩けるPostman[注8]が用意してあったり、実際の例に則したように、Orionを使用できるFIWARE APIウォークスルー[注9]やAPI全体の仕様をSwagger UI[注10]で公開したりしていますので、そちらもご覧いただければ幸いです。

注5） https://github.com/makeOurCity/docker-compose-orion-v2/blob/main/src/

注6） https://www.letsfiware.jp/fiware-tutorials/

◆リスト3　エンティティの作成の例

```
const axios = require("axios");

const data = {
  id: "urn:ngsi-ld:MEASUREMENT:id:1",
  type: "DeviceMeasurement",
  numValue: {
    type: "Number",
    value: 20.5,
  },
};

axios
  .post("http://localhost:1026/v2/entities", data)
  .then((res) => {
    console.log(res);
  });
```

おわりに

```
while (Japan.recovering
        we.hack();
}
```

今回はOrionだけの紹介になりましたが、先にも少しだけ紹介したとおり、FIWAREは特定のソフトウェアを示す言葉ではなく、さまざまなツールの総称です。Orionだけではできることも限られており、FIWAREのほかのソフトウェアと組み合わせることでさまざまなデータを取り扱い、データ構造を変更したり、集計、統計なども動的に行ったり、グラフィカルに表示したりすることができるようになります。

それは、Orionを利用し公開されているデータであれば、自分でもFIWAREのそのほかのツールを使い、自分や都市にとって便利なものを用意することができる、ということでもあります。そのために、FIWAREのいろいろなアプリケーションを導入し、便利なものを提供するということもできますが、まずはOrionにデータを投入し、それを公開するということだけでも意味があると思っています。

本稿を読んでFIWAREに興味を持った方は、ぜひCode for JapanのSlackワークスペース[注11]にご参加ください。SD

注7） https://fiware-orion.letsfiware.jp/orion-api/

注8） https://www.postman.com/fiware/workspace/fiware-foundation-ev-s-public-workspace/collection/513743-dd7b3ba8-5aa3-405a-b054-47608a483f77

注9） https://fiware-orion.letsfiware.jp/user/walkthrough_apiv2/#_2

注10） https://swagger.lab.fiware.org/

注11） https://www.code4japan.org/activity/community