Orion Context broker

Agenda

- Descripción
- Modelo NSGI
- Arquitectura
- JSON
- REST
- OCB Query Language
- Datos geo-referenciados
- Suscriptor

Descripción

Orion Context Broker (OCB) permite administrar todo el ciclo de vida de la información en el conjunto de Fiware.

A través del OCB es posible registrar entidades de contexto y administrarlos a través de actualizaciones y consultas. Además, permite suscribirse a la información de contexto para que cuando se produzca alguna modificación se reciba una notificación.

Información de contexto

Es el valor del atributo que caracteriza una entidad en una aplicación. Esta información proviene de diferentes fuentes y protocolos.

Modelo

De acuerdo a NSGI:

Entidades son la representación virtual de todos los objetos físicos en el mundo.

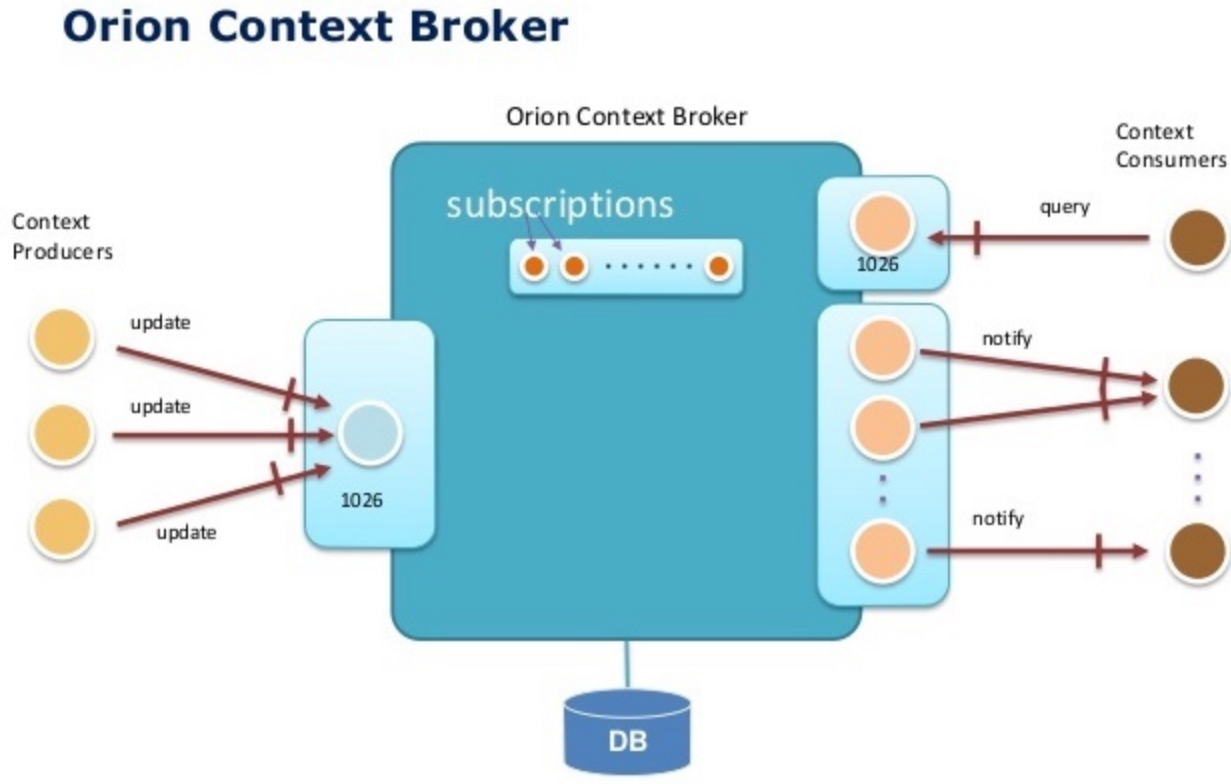
Cualquier información disponible sobre las entidades físicas son expresadas en forma de atributos de una manera virtual.

Iniciar máquina virtual con OCB

```
# Utilizando la terminal.
#### Set up
# crear carpeta
mkdir curso
# cambiar de directorio
cd curso
# bajar estructura de las máquinas virtuales
git clone https://github.com/CarlosUrteaga/Fiware
# Cambiar a directorio Fiware
cd Fiware
cd vm-fiware-orion
# iniciar máquina virtual de Orion
vagrant up
# conectar a máquina virutal por medio de SSH
vagrant ssh
#iniciar docker de OCB
docker-compose up
```

Arquitectura

OCB es un intermediario entre los productores (dispositivos) y los consumidores (aplicaciones).

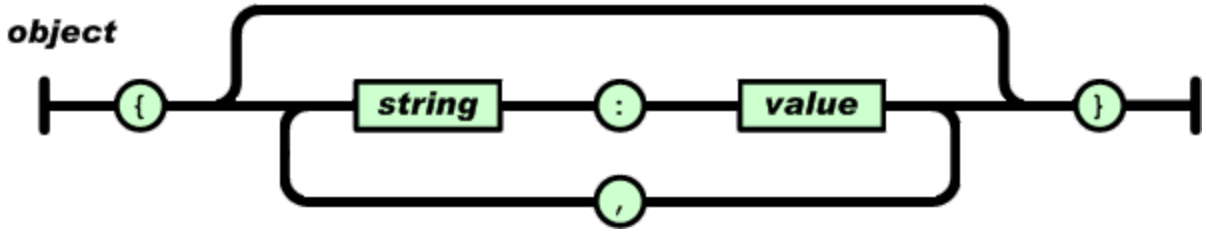


- Los productores (estaciones) son responsables de actualizar la información en la base de datos.
- Los consumidores realizan consultas para obtener el estado actual. Además se tiene la posibilidad obtener notificaciones de atributos particulares.

El formato de datos que utiliza el OCB es de tipo JSON.

JSON

El formato de datos JSON (Java Script Object Notation) es ligero para el intercambio de información, además de ser fácil de leer y de procesar.



String está definido como las propiedades de las entidades.

value es el atributo de tipo booleano, objeto JSON, arreglo, número o nulo.

JSON:

```
{ "entidad": atributo, "entidad":atributo, ...}
```

Ejemplo:

José tiene 21 años y está cursando con tres materias: Física, Matemáticas y Economía.

```
{
        "nombre":"José",
        "edad":21,
        "Materias":
        {
                "Materias1":"Física",
                "Materias2":"Matemáticas",
                "Materias3":"Economía"
        }
}
```

REST

OCB realiza consultas a la base de datos. Las típicas consultas son insertar, modificar y obtener.

OCB contiene una interfaz tipo web para realizar dichas consultas. Este es un servicio web de tipo REST (REpresentational state Transfer).

La forma de interactuar con este tipo de servicio web es a través de una URL, tipo de acción y un cuerpo con información.

Este tipo de servicios tiene la flexibilidad de obtener información de forma arborecente. Es decir, es posible obtener | actualizar | borrar información de una entidad completa o sólo valores de una entidad en especifico.

Ejercicio

Los siguientes ejercicios serán realizados utilizando el programa Insomnia.

Dentro de insomina se creará una carpeta con el nombre Operaciones-Comunes.

POST

Para generar una nueva petición que inserte una entidad en la base de datos se usará la operación *post*, el nombre sugerido es inserta-entidad.

```
En el campo de URL se pone la dirección donde se encuentra

        post http://url:1026/v2/entities
```

Crear un objeto tipo lugar de estacionamiento que se encuentra en la planta baja en el pasillo A lugar 13 que se encuentra opcuapdo

Ejemplo:

```
post: http://localhost:1026/v2/entities
Header:
        Content-type application/json

Body:
        {
                "id": "lugar01",
                "type": "ParkingSpot",
                "floor" :{
                        "value":"PB",
                        "type":"text"
            },
                "name" :{
                        "value":"A-13"
            },
                "status" :{
                        "value":"libre"
            }
        }
```

El id, representa el identificador del objeto mientras que type permite inferir el tipo atributo.

Para cada id existen varios Type, es decir, puede existir varios objetos con el mismo id siempre y cuando sean de distinto tipo.

Para los campos de atributos, muchas veces se puede inferir el tipo de acuerdo al valor, pero es una buena práctica incluir el tipo de dato (*type*).

GET

La operación utilizada para obtener información de la base de datos es el get. Es posible duplicar la consulta anterior y renombrarla, con el nombre obten-todas-entidades.

Para este caso, sólo se especifica el URL, sin body ni Content-type.

```
        get http://{url}:1026/v2/entities
```

ejemplo:

```
        get http://localhost:1026/v2/entities
```

No obstante si se está consultando un OCB en la nube, será necesario agregar en el campo X-Auth-Token el token que les fue asignado.

```
        get url   http://{url}:1026/v2/entities
headers X-Auth-Token    1O11Qj4FReeHTs0Rb5hVLYwKNHFbbu
```

Para consultar una sola entidad, el URL se especifica con el id.

```
get http://{url}:1026/v2/entities/{id}
```

ejemplo:

```
get http://localhost:1026/v2/entities/lugar01
```

Esta consulta permite notar que la base de datos infiere el tipo datos insertados.

Actualizar valores

Es posible actualizar varios valores o uno solo. En el caso de varios valores se utiliza el formato JSON mientras que en el otro se utiliza el *type* del valor a insertar.

```
        put http://{url}:1026/v2/entities/{id}
        Header:
        Body:
```

Ejemplo:

```
        put http://localhost:1026/v2/entities/lugar01/att
        Header
                Content-type application/json
Body:
{
    "name" :{
                "value":"A15"
    },
        "floor" :{
                "value":"PB"
    },
        "status" :{
                "value":"ocupado"
    }
}
```

Modificación de un solo valor.

```
        put http://{url}:1026/v2/entities/{id}/attrs/{val
        Header:
        Body:
```

Ejemplo:

```
        put http://localhost:1026/v2/entities/lugar01/att
        Header
                Content-type text/plain
Body:
"free"
```

Delete

Es posible borrar atributos e identidades.

Para borrar un atributo se utiliza el comando Delete:

```
    delete http://url:1026/v2/entities/{id}/attrs/{value}
```

Para borrar una se utiliza la siguiente expresión:

```
    delete http://url:1026/v2/entities/{id}
```

Metadatos

Los metadatos se agregan en la nueva versión de NGSI. Simplemente se pone información adicional con la misma estructura. Por ejemplo, si se está manejando temperatura, se podría agregar algo así:

```
{
        "id":"lugar01",
        "type":"ParkingSpot",
        "temperature" :{
                "value":23,
                "metadata":{
                        “precision”: {
                                “type”:xxx,
                                “value”: xxx
                        }
                }
        ...
```

```
                        }
                    }
                },
        {
                    "id": "PalacioNacional",
                    "type": "PointOfInterest",
                    "category": {
                            "type": "Text",
                            "value": "Edificio",
                            "metadata": {}
                    },
                    "location": {
                            "type": "geo:point",
                            "value": "19.432336, -99.1
                            "metadata": {}
                    },
                    "name": {
                            "type": "Text",
                            "value": "Palacio Nacional
                            "metadata": {}
                    },
                    "postalAddress": {
                            "type": "StructuredValue"
                            "value": {
                                    "addressCountry":
                                    "addressLocality"
                                    "addressRegion": "
```