Article
Lucas Enard · Sept 30, 2022 9m de lecture

Open Exchange

Réalisation complète d'IRIS en Python : DataTransformation d'un CSV vers un serveur FHIR

Dans ce GitHub nous recueillons des informations à partir d'un csv, nous utilisons une DataTransformation pour les transformer en un objet FHIR, puis nous sauvegardons ces informations sur un serveur FHIR, et tout cela en utilisant uniquement Python.

The objective is to show how easy it is to manipulate data into the output we want, here a FHIR Bundle, in the IRIS full Python framework.

1. Fhir-orga-dt

2. Préalables

Assurez-vous que git et Docker desktop sont installé.

Si vous travaillez à l'intérieur du conteneur, comme il est montré dans 3.3., vous n'avez pas besoin d'installer fhirpy et fhir.resources.

Si vous n'êtes pas dans le conteneur, vous pouvez utiliser pip pour installer fhirpy et fhir.resources.
Vérifiez fhirpy et fhir.resources pour plus d'information.

3. Installation

3.1. Installation pour le développement

Clone/git tire le repo dans n'importe quel répertoire local, par exemple comme indiqué ci-dessous :

```
git clone https://github.com/LucasEnard/fhir-client-python.git
```

Ouvrez le terminal dans ce répertoire et lancez :

```
docker build .
```

3.2. Portail de gestion et VSCode

Ce référentiel est prêt pour VS Code.

Ouvrez le dossier fhir-client-python cloné localement dans VS Code.

Si vous y êtes invité (coin inférieur droit), installez les extensions recommandées.

3.3. Avoir le dossier ouvert à l'intérieur du conteneur

Vous pouvez être *à l'intérieur* du conteneur avant de coder si vous le souhaitez.

Pour cela, il faut que docker soit activé avant d'ouvrir VSCode.

Ensuite, dans VSCode, lorsque vous y êtes invité (coin inférieur droit), rouvrez le dossier à l'intérieur du conteneur afin de pouvoir utiliser les composants python qu'il contient.

La première fois que vous effectuez cette opération, cela peut prendre plusieurs minutes, le temps que le conteneur soit préparé.

Si vous n'avez pas cette option, vous pouvez cliquer dans le coin inférieur gauche et cliquer sur Ouvrir à nouveau dans le conteneur puis sélectionner De Dockerfile

Plus d'informations ici

En ouvrant le dossier à distance, vous permettez à VS Code et à tous les terminaux que vous ouvrez dans ce dossier d'utiliser les composants python dans le conteneur.

4. Serveur FHIR

Pour compléter cette présentation, nous allons utiliser un serveur fhir.

Ce serveur fhir était déjà intégré lorsque vous avez cloné et construit le conteneur.

L'url est localhost:52773/fhir/r4

5. Présentation pas à pas

Présentation complète de la réalisation d'IRIS en Python.

5.1. Messages and objects

Les objets et les messages contiennent les informations entre nos services, nos processus et nos opérations.

Dans le fichier obj.py nous devons créer une classe de données qui correspond au csv, ceci sera utilisé pour garder l'information avant de faire la DataTransformation.

Dans notre exemple, le csv organisation.csv ressemble à ceci,

```
active;name;city;country;system;value
true;Name1;city1;country1;phone;050678504
false;Name2;city2;country2;phone;123456789
```

Par conséquent, l'objet ressemblera à ceci,

```
@dataclass
# > Cette classe représente une organisation simple
class BaseOrganization:
    active:bool = None
    name:str = None
    city:str = None
    country:str = None
    system:str = None
    value:str = None
```

Dans le fichier msg.py, nous aurons deux types de requêtes, la première contenant les informations d'une organisation avant la DataTransformation et la seconde contenant les informations de l'organisation après la DataTransformation.

5.2. Service aux entreprises

Dans le fichier bs.py, nous avons le code qui nous permet de lire le csv et pour chaque ligne du csv (donc pour chaque organisation), le mapper dans un objet que nous avons créé plus tôt.
Ensuite, pour chacune de ces lignes (organisation), nous créons une requête et l'envoyons à notre processus pour effectuer la DataTransformation.

```
# Nous ouvrons le fichier
with open(self.path + self.filename,encoding="utf-8") as csv:
    # Nous le lisons et le mappons en utilisant l'objet BaseOrganization précédemment
 utilisé
    reader = DataclassReader(csv, self.fhir_type ,delimiter=";")
    # Pour chacune de ces organisations, nous pouvons créer une requête et l'envoyer
au processus
    for row in reader:
```

```
        msg = OrgaRequest()
        msg.organization = row
        self.send_request_sync('Python.ProcessCSV',msg)
```

5.3. Service aux entreprises

Dans le fichier bp.py, nous avons la DataTransformation, qui convertit un simple objet python contenant peu d'informations en un objet FHIR R4

Voici les étapes pour effectuer une DataTransformation en utilisant du python embarqué sur notre organisation simple,

```
# Création de l'objet Organisation
organisation = Organisation()

# Mappage des informations de la demande à l'objet Organisation
organization.name = base_orga.name

organization.active = base_orga.active

## Création de l'objet Adresse et mappage des informations
## de la demande à l'objet Addresse
adresse = Addresse()
adress.country = base_orga.country
adress.city = base_orga.city

### Configuration de l'adresse de notre organisation à celle que nous avons créée
organization.address = [adress]

## Création de l'objet ContactPoint et mise en correspondance
## des informations de la demande avec l'objet ContactPoint
telecom = ContactPoint()
telecom.value = base_orga.value
telecom.system = base_orga.system

### Configuration de la télécommunication de notre organisation à celle que nous avon
s créée
organization.telecom = [telecom]

# Maintenant, notre DT est achevé, nous avons une organisation d'objets qui est
# un objet FHIR R4 et qui contient toutes nos informations csv.
```

Après cela, notre mappage est terminé et notre DT fonctionne.

Maintenant, nous pouvons envoyer cette ressource FHIR R4 nouvellement créée à notre FhirClient qui est notre opération.

5.4. Opération d'un entreprise

Dans le fichier bo.py nous avons le FhirClient, ce client crée une connexion à un serveur fhir qui contiendra les informations recueillies par le csv.

Dans cet exemple, nous utilisons un serveur fhir local qui n'a pas besoin d'une clé api pour se connecter.

Pour s'y connecter, nous devons utiliser la fonction oninit,

```
if not hasattr(self,'url'):
    self.url = 'localhost:52773/fhir/r4'

self.client = SyncFHIRClient(url=self.url)
```

Maintenant, lorsque nous recevons un message/demande, nous pouvons, en trouvant le type de ressource de la ressource que nous envoyons avec notre demande au client, créer un objet lisible par le client, puis le sauvegarder sur le serveur fhir.

```
# Obtenez le type de ressource de la demande ( ici "Organisation" )
resource_type = request.resource["resource_type"]

# Créer une ressource de ce type en utilisant les données de la demande
resource = construct_fhir_element(resource_type, request.resource)

# Sauvegarder la ressource sur le serveur FHIR en utilisant le client
self.client.resource(resource_type,**json.loads(resource.json())).save()
```

Il est à noter que le client fhir fonctionne avec n'importe quelle ressource de FHIR R4 et pour utiliser et modifier notre exemple, nous devons seulement changer la DataTransformation et l'objet qui contient les informations csv.

5.5. Conclusion de la présentation

Si vous avez suivi ce parcours, vous savez maintenant exactement comment lire un csv d'une représentation d'une ressource FHIR R4, utiliser une DataTransformation pour le transformer en un véritable objet FHIR R4 et le sauvegarder sur un serveur.

6. Création d'une nouvelle DataTransformation

Ce référentiel est prêt à être codé en VSCode avec les plugins InterSystems.
Ouvrez /src/python pour commencer à coder ou utiliser l'autocomplétion.

Étapes pour créer une nouvelle transformation

Pour ajouter une nouvelle transformation et l'utiliser, la seule chose que vous devez faire est d'ajouter votre csv nommé Patient.csv (par exemple) dans le dossier src/python/csv.

Ensuite, créez un objet dans src/python/obj.py appelé BasePatient qui met en correspondance votre csv.

Maintenant, créez une requête dans src/python/msg.py appelée PatientRequest qui a une variable resource de type BasePatient.

L'étape finale est la DataTransformation, pour cela, allez dans src/python/bp.py et ajoutez votre DT. Ajoutez d'abord if isinstance(request, PatientRequest): puis mappez votre ressource de requête à une fhir.resource Patient.

Maintenant si vous allez dans le portail de gestion et changez le paramètre du ServiceCSV pour ajouter filename=Patient.csv vous pouvez juste démarrer la réalisation et voir votre transformation se dérouler et votre client envoyer les informations au serveur.

Étapes détaillées pour créer une nouvelle transformation

Si vous n'êtes pas sûr de ce qu'il faut faire ou de la manière de le faire, voici une création étape par étape d'une nouvelle transformation :

Créez le fichier Patient.csv and le dossier src/python/csv pour les remplir avec le suivant:

```
family;given;system;value
FamilyName1;GivenName1;phone;555789675
FamilyName2;GivenName2;phone;023020202
```

Notre CSV contient un nom de famille, un prénom et un numéro de téléphone pour deux patients.

Dans src/python/obj.py écrivez :

```
@dataclass
class BasePatient:
    family:str = None
    given:str = None
    system:str = None
    value:str = None
```

Dans src/python/msg.py écrivez :

```
from obj import BasePatient
@dataclass
class PatientRequest(Message):
    resource:BasePatient = None
```

Dans src/python/bp.py écrivez :

```
from msg import PatientRequest
from fhir.resources.patient import Patient
from fhir.resources.humanname import HumanName
```

Dans src/python/bp.py dans la fonction on<u>r</u>equest écrivez :

```
if isinstance(request,PatientRequest):
    base_patient = request.resource

    patient = Patient()

    name = HumanName()
    name.family = base_patient.family
    name.given = [base_patient.given]
    patient.name = [name]
```

```
    telecom = ContactPoint()
    telecom.value = base_patient.value
    telecom.system = base_patient.system
    patient.telecom = [telecom]

    msg = FhirRequest()
    msg.resource = patient

    self.send_request_sync("Python.FhirClient", msg)
```

Maintenant si vous allez dans le portail de gestion et changez le paramètre du ServiceCSV pour ajouter filename=Patient.csv vous pouvez juste arrêter et redémarrer la production et voir votre transformation se dérouler et votre client envoyer les informations au serveur.

7. Ce qu'il y a dans le référentiel

7.1. Dockerfile

Le dockerfile le plus simple pour démarrer un conteneur Python.

Utilisez docker build . pour construire et rouvrir votre fichier dans le conteneur pour travailler à l'intérieur de celui-ci.

7.2. .vscode/settings.json

Fichier de paramètres.

7.3. .vscode/launch.json

Fichier de configuration si vous voulez déboguer.

#Bases de données #CSV #FHIR #Importation et exportation de données #Python #InterSystems IRIS for Health
Voir l'application sur InterSystems Open Exchange

URL de la source:https://fr.community.intersystems.com/post/r%C3%A9alisation-compl%C3%A8te-diris-en-python-datatransformation-dun-csv-vers-un-serveur-fhir