TP SD
Service web RESTful

14 novembre 2023

L'objectif de ce TP est de manipuler un service web RESTful orienté navigabilité. La manipulation se fait via un outil en ligne de commande, `curl`, et une bibliothèque Python, `requests`.

1 Exécution simple

Cet exercice consiste à installer et faire fonctionner un service web structurant des données sur des personnes et les ouvrages qu'ils ont réalisés.

1. Lancer un serveur MongoDB avec `docker run --name some-mongo -p 27017:27017 -d mongo`. On peut vérifier qu'il n'y pas d'erreur avec `docker ps` et en consultant les logs `docker logs some-mongo`.
2. Installer la bibliothèque Python Eve (framework REST) avec `pip3 install eve`.
3. Reprendre le code source disponible dans l'archive fournie.
4. Lancer l'exécution du service avec `python3 run.py`, puis charger les données tests avec `python3 client.py`.
5. Réaliser la méthode GET depuis un navigateur et vérifier que la requête est bien prise en compte du coté serveur.

> On accède au service en rentrant l'adresse `http://localhost:5000` dans un navigateur.

Alternativement, on peut lancer le serveur avec `docker-compose up --build`, puis poursuivre avec le chargement des données.

2 Requêtes avec `curl`

Dans un premier temps, on étudiera comment se structurent les données fournies par le service web avec le client HTTP `curl`. Puis, on s'intéressera aux différents types de requêtes ainsi qu'aux codes de statut.

> Attention, il peut être nécessaire de désactiver le proxy : `http_proxy=`.
>
> On peut aussi étudier le fonctionnement de `curl` par le biais d'un service externe tel que `https://jsonplaceholder.typicode.com/`.

2.1 Requêtes GET

Exécuter la commande suivante pour faire une requête simple de type GET : `curl --verbose http://localhost:5000`. Analyser la partie liée à HTTP dans l'affichage. Pour analyser la partie liée au corps XML, il peut être utile d'utiliser `curl http://localhost:5000 | jq` ou `curl http://localhost:5000 | python3 -m json.tool` à défaut. Comment obtenir un affichage encore plus détaillé ?

On trouve la documentation de l'affichage dans `man curl`. On peut avoir plus d'information avec l'option `--trace -` à la place de `--verbose`.

Des liens sont fournies à chaque réponse GET. Les parcourir et identifier comment les adresses des ressources se structurent. Réaliser un diagramme montrant les changements d'états possibles pour le client. Faire le lien avec les scripts Python utilisés.

La navigabilité au sein de l'interface est au cœur du principe HATEOAS et cela prend la forme d'une section `_links` dans chaque réponse GET. Voir aussi https://docs.python-eve.org/en/stable/features.html#hateoas.

Le fichier `settings.py` documente le schéma d'accès aux données.

2.2 Requêtes de modification

Rajouter une personne avec la requête suivante :

```
curl --header "Content-Type: application/json" \
     --data '{"firstname": "Barbara", "lastname": "Snow"}' \
     http://localhost:5000/persons
```

Pourquoi ne précise t-on pas qu'il s'agit d'une requête de type POST ?

La documentation mentionne que l'option `--data` implique une requête de type POST.

On souhaite faire une requête GET qui n'accepte comme format de retour que le JSON. Écrire la requête qui indique cette contrainte dans l'en-tête de la requête.

```
curl -H "Accept: application/json" \
     --output out.json \
     http://localhost:5000/persons
```

En analysant le résultat de la requête précédente, rajouter un autre client avec une adresse complète en utilisant le format JSON.

```
curl -H "Content-Type: application/json" \
     -d '{"firstname":"Louis-Claude",
       "lastname":"Canon",
       "location":{
         "number":16,
         "street":"route de Gray",
         "city":"Besançon"}}' \
     http://localhost:5000/persons
```

Réaliser une requête permettant de changer l'auteur d'un ouvrage donné. Quel entête est-il nécessaire d'utiliser pour ce changement et pourquoi ? Pour finir, supprimer un auteur donné. Quel(s) problème(s) de cohérence pourrait-on identifier dans ce service ?

```
curl -X PATCH -H "Content-Type: application/json" \
     -H "If-Match: a8d692294d1b472f1b873df9d49d1de7477d0f7c" \
     -d '{"owner": "61d46e46dc46facf267b16d2"}' \
     http://127.0.0.1:5000/works/61d46e47dc46facf267b16eb
curl -X DELETE \
     -H "If-Match: f9ba97365f57cc1103785ab41e8abb325c09a2bd" \
     http://127.0.0.1:5000/persons/61d32405e84369bd18de66cf
```

Un entête `If-Match` mal défini empêche la suppression pour éviter des problèmes de cohérence en cas de concurrence. Voir aussi https://docs.python-eve.org/en/stable/features.html#data-integrity-and-concurrency-control.

Il reste cependant un problème de cohérence au niveau de la contrainte d'intégrité référentielle des ouvrages vers les auteurs : lors de la suppression des auteurs, les ouvrages correspondants ne sont pas impactés.

2.3 Codes de statut

Analyser les codes de statut déjà obtenus et identifier leurs significations.

On retrouve la description des codes sur https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml :

200 OK : succès lors d'une requête en lecture.

201 Created : succès d'une création.

204 No Content : succès d'une requête sans contenu de retour.

404 Not Found : ressource non trouvée.

412 Precondition Failed : préconditions envoyées par la requête non vérifiées.

422 Unprocessable Content : le contenu ne peut être traité.

428 Precondition Required : la requête doit être conditionnelle.

3 Bibliothèque Python `requests`

En repartant du code fourni dans `client.py`, écrire un code Python qui permet de vérifier si la contrainte d'intégrité référentielle est bien respectée.

```python
persons = requests.get('http://localhost:5000/persons').json()['_items']
ids = [p['_id'] for p in persons]

works = requests.get('http://localhost:5000/works').json()['_items']
refs = [w['owner'] for w in works]
for r in refs:
    if r not in ids:
        print('Missing author', r)
```

Écrire un script Python qui supprime les ouvrages correspondant à chaque auteur manquant pour rétablir la contrainte d'intégrité référentielle.

```
persons = requests.get('http://localhost:5000/persons').json()['_items']
ids = [p['_id'] for p in persons]

works = requests.get('http://localhost:5000/works').json()['_items']
for w in works:
    if w['owner'] not in ids:
        requests.delete('http://localhost:5000/works/' + w['_id'],
            headers={'If-Match': w['_etag']})
```

4 Diagramme d'états-transitions

Réaliser le diagramme de transitions correspondant aux états du client avec les requêtes de type GET, POST et DELETE seulement.