API, définition et usage.

Kossi

07/01/2020

API : qu'est-ce que c'est ?

Une **API**, ou Application Programming Interface, est une interface ou un protocole de communication entre différentes parties d'un programme informatique destiné à simplifier la mise en œuvre et la maintenance des logiciels.

Une classe d'APIs très utiles est celle des APIs basées sur le Web, dite **API Web**. Une *API Web* est un serveur que vous pouvez utiliser pour récupérer et envoyer des données à l'aide de code. Les API sont le plus souvent utilisées pour récupérer des données, et ce sera en grande partie l'objet de ce cours.

Lorsque vous naviguez sur https://www.google.com/ , vous envoyez actuellement une requête qui ressemble à :

Request URL: http://google.com/
Request Method: GET
Status Code: 200
Remote Address: 172.217.18.
Referrer Policy: no-referrer-when-downgrade
Cache-Control: public, max-age=2592000
Content-Length: 219
Content-Type: text/html; charset=UTF-8
Date: Mon, 06 Jan 2020 03:08:42 GMT
Expires: Wed, 05 Feb 2020 03:08:42 GMT
Location: http://www.google.com/

Pour extraire des données du web, certains sites web offrent une **API** documentée qui peut être requêtée par HTTP.

HTTP (HyperText Transfer Protocol) est un protocole conçu pour permettre la communication entre clients et serveurs. Lorsque vous surfez sur le Web, votre navigateur Web peut être le client, et une application Web hébergée sur un ordinateur distant peut être le serveur. Il existe différentes méthodes HTTP : **GET**, **POST**, **PUT**, **DELETE** ... Les plus utilisées sont **GET** et **POST**.

Surfer sur le net sans son navigateur web

- Ouvrir un terminal sur votre OS (Win + "cmd" + Enter ou Ctrl+Alt+T sur Linux)
- Taper: `curl -X GET www.google.com`
- Vous verrez la réponse du serveur sur votre écran. C'est un code HTML brut qui pourrait être visionné par n'importe quel lecteur HTML.

Envoyer une requête `GET`

Pour faire une requête `GET`, nous utiliserons le module **requests** et sa fonction **requests**.get(), qui nécessite un argument: l'**URL** (**U**niversal **R**essource **L**ocator) de notre requête.

Lorsque vous envoyez une requête `GET` au serveur, vous recevez normalement une réponse HTTP. En Python et avec le module **requests**, cette réponse http est représentée sous forme d'un objet `requests.models.Response` comme vous pouvez le voir ci-dessous :

```
import requests
# Nous envoyons une requête `GET` à l'URL 'http://api.worldbank.org/v2/country/br'
reponse = requests.get("http://api.worldbank.org/v2/country/br")
print("La reponse de la requête GET est un objet: {}".format(type(reponse)))
```

```
## La reponse de la requête GET est un objet: <class 'requests.models.Response'>
```

Ce type d'objet possède un attribut **status_code**; c'est le **code d'état** de votre requête et il vous permet de vérifier si la requête a réussi ou pas.

```
code_etat = reponse.status_code
print("code  d'état : {}".format(code_etat))
```

```
## code  d'état : 200
```

Le protocole HTTP stipule différents types de **code d'état** dont les plus importants sont :

- **200** : succès de la requête ;
- **301** et **302** : redirection, respectivement permanente et temporaire ;
- **401** : utilisateur non authentifié ;
- **403** : accès refusé ;
- **404** : page non trouvée (vous le reconnaîtrez sûrement);
- **500** et **503** : erreur serveur ;
- **504** : le serveur n'a pas répondu.

Visiter ce lien pour une liste plus complète.

`Question : Alors, l'URL précédemment requêtée existe-t-elle ou pas ?`

Récupérer le contenu de la réponse du serveur

Même si le code d'état est un moyen simple pour s'enquérir de l'état de votre requête, il ne vous fournit pas l'information concrète. Le but premier quand on on communique avec le serveur, c'est de pouvoir récupérer nos données pour d'autres usages plus ou moins complexes. Où sont alors les données dans notre cas ?

Tout d'abord, rappelons que notre requête ci-dessus s'adresse aux serveurs de la banque mondiale. Cette dernière a mis a disposition des acteurs économiques une API permettant d'extraire de leurs différentes bases des informations variées. Typiquement, le **point de terminaison** (**endpoint** en anglais) http://api.worldbank.org/v2/country/br est une sorte d'interface où l'on peut demander à la banque mondiale de nous fournir des informations spécifiques sur un pays. Dans l'URL précédent, le pays d'intérêt est le Brésil qui est symbolisé par le **br** à la toute fin. En changeant ce paramètre **br**, qui est le code ISO 3166 du Brésil, nous pouvons avoir des informations sur n'importe quel pays. Remplacez par exemple le **br** par **ci** et vous aurez des informations à propos de la Côte d'Ivoire.

Informations, informations ! Mais où sont-elles, vous vous dites sûrement. Patience ! En effet l'objet **Response** retourné par le serveur contient tout ce dont nous avons besoin. Il possède notamment un champ **text** qui stocke la réponse brute du serveur.

```
contenu = reponse.text
print(contenu)
```

```
## <?xml version="1.0" encoding="utf-8"?>
## <wb:countries page="1" pages="1" per_page="50" total="1" xmlns:wb="http://www.worldbank.org">
##   <wb:country id="BRA">
##     <wb:iso2Code>BR</wb:iso2Code>
##     <wb:name>Brazil</wb:name>
##     <wb:region id="LCN" iso2code="ZJ">Latin America &amp; Caribbean </wb:region>
##     <wb:adminregion id="LAC" iso2code="XJ">Latin America &amp; Caribbean (excluding high income)</wb
##     <wb:incomeLevel id="UMC" iso2code="XT">Upper middle income</wb:incomeLevel>
##     <wb:lendingType id="IBD" iso2code="XF">IBRD</wb:lendingType>
##     <wb:capitalCity>Brasilia</wb:capitalCity>
##     <wb:longitude>-47.9292</wb:longitude>
##     <wb:latitude>-15.7801</wb:latitude>
##   </wb:country>
## </wb:countries>
```

La banque mondiale nous renvoie donc neuf informations clés sur le pays demandé :

- Code alpha-3 ISO 3166-1 à 3 lettres
- Code alpha-2 ISO 3166-1 à 2 lettres
- Nom du pays
- Région: ID, nom et code à deux lettres de la Banque mondiale
- Niveau de revenu: ID, nom et code à deux lettres de la Banque mondiale
- Type de prêt: ID, nom et code à deux lettres de la Banque mondiale
- Capitale du pays
- Longitude de la capitale
- Latitude de la capitale

Requête paramétrée

Mais tout n'est pas encore gagné, les données telles que présentées sont sous un format non structuré et se prêtent difficilement à des analyses statistiques. Nous aurions aimé que la banque mondiale nous renvoie des données sous forme de table excel comme nous en avons bien l'habitude en statistique. Pour réaliser cet objectif, nous avons deux possibilités:

- **Faire la mise en forme des données par nous même**. Le format actuel des données s'appelle **format xml** et il y a des librairies **python** qui permettent de le "parser". Nous reviendrons sur ce point un peu plus tard dans le cours mais pour le moment, nous n'avons encore rien vu dans ce cours qui nous permettrait de faire cette tâche quelque peu incongrue.
- **Trouver un moyen de dire à la banque mondiale que nous préférons des données tabulaires**: les requêtes HTTP possèdent toutes des paramètres que l'on peut modifier pour affiner les résultats renvoyés par le serveur. Dans le cas de l'API de la banque mondiale, le paramètre **format** permet de spécifier le format de la réponse renvoyé par le serveur. Comment ça s'utilise alors ?

```
reponse = requests.get("http://api.worldbank.org/v2/country/br?format=json")
contenu = reponse.text
print(contenu)
```

```
## [{"page":1,"pages":1,"per_page":"50","total":1},[{"id":"BRA","iso2Code":"BR","name":"Brazil","region
```

Vous avez vu le changement opéré au niveau de l'URL ? C'est exactement de cette manière que l'on spécifie les paramètres dans une requête HTTP. Ce n'est pas une spécificité de l'API de la banque mondiale mais de toutes les requêtes HTTP. Vous auriez sûrement remarqué le paramètre *?q* utilisé par google pour envoyer les recherches à ses serveurs !

Le format JSON

Le nouveau format est nettement plus lisible même s'il n'est toujours pas assez tabulaire: c'est du **json**. Le format json s'apparente aux dictionnaires python et se présente ici comme un mélange entre des listes, de chaînes de caractères et de valeurs numériques. Pour gérer ce type de format, il existe un module **python** repondant exactement au même nom, c'est-à-dire que le nom du module est **json**.

```
import json
reponse_json = json.loads(contenu)
print(reponse_json)
```

```
## [{'page': 1, 'pages': 1, 'per_page': '50', 'total': 1}, [{'id': 'BRA', 'iso2Code': 'BR', 'name': 'Bra
```

La fonction **json.loads** prend en entrée un texte (chaîne de caractères) au format json et renvoie un objet python qui vous permet d'accéder aux différents champs contenus dans le json initial. Ainsi, pour afficher quelques enregistrements parmi les résultats renvoyés par le serveur on procède comme suit:

```
import json
reponse_json = json.loads(contenu)
print("Premier enregistrement: {}".format(reponse_json[0]))
```

```
## Premier enregistrement: {'page': 1, 'pages': 1, 'per_page': '50', 'total': 1}
```

```
print("Dernier enregistrement: {}".format(reponse_json[-1]))
```

```
## Dernier enregistrement: [{'id': 'BRA', 'iso2Code': 'BR', 'name': 'Brazil', 'region': {'id': 'LCN', '
```

Si vous essayez d'indexer comme ci-dessus le contenu renvoyé par le serveur sans appeler **json.loads** au préalable, vous n'obtiendrez que des lettres de l'alphabet qui le composent.

NOTE: L'objet **Response** dispose en soi une fonction **json** qui vous formate la reponse du serveur au format json sans que vous n'ayez besoin de le faire par vous même. Vous vous doutez bien que sous le capot, cette fonction utilise la librairie *json* que nous venons de voir.

```
reponse = requests.get("http://api.worldbank.org/v2/country/br?format=json")
contenu = reponse.json()
print("Premier enregistrement: {}".format(contenu[0]))
```

```
## Premier enregistrement: {'page': 1, 'pages': 1, 'per_page': '50', 'total': 1}
```

```
print("Dernier enregistrement: {}".format(contenu[-1]))
```

```
## Dernier enregistrement: [{'id': 'BRA', 'iso2Code': 'BR', 'name': 'Brazil', 'region': {'id': 'LCN', '
```

Résumé

En définitive, nous avons appris dans ce tutoriel introductif à :

- Définir une **API**
- Envoyer une requête **GET** en python
- Vérifier le **code d'état** d'une requête
- Récupérer le contenu de la réponse renvoyée par le serveur
- Spécifier les paramètres dans une requête HTTP
- Gérer les chaînes de caractères au format **json**