Utiliser une API

Dernière mise à jour le : 05/01/2024

Qu'est-ce qu'une API ?

> Une **API** (*Application Programming Interface*, ou *interface de programmation pour application*) est une interface *logicielle* qui permet de « connecter » un logiciel ou un service à un autre logiciel ou service afin d'échanger des données et des fonctionnalités.
>
> Les API offrent de nombreuses possibilités, comme la portabilité des données, la mise en place de campagnes de courriels publicitaires, des programmes d'affiliation, l'intégration de fonctionnalités d'un site sur un autre ou l'*open data*. Elles peuvent être gratuites ou payantes.
>
> *Définition fournie par la CNIL*

Ces interfaces, généralement offertes par des services Web, sont le plus souvent accompagnées d'une documentation complète décrivant comment les programmes *consommateurs* peuvent se servir des fonctionnalités du programme *fournisseur*.

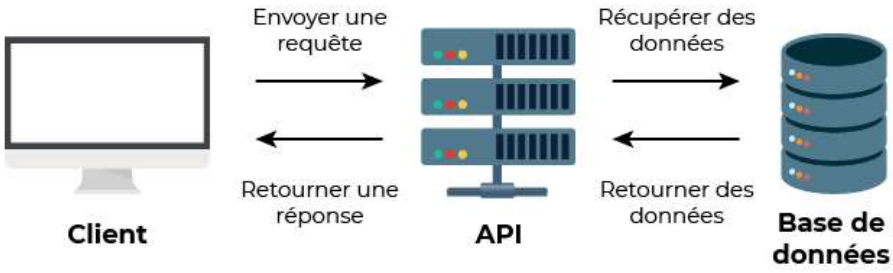
De plus en plus de services en ligne disposent d'API permettant aux utilisateurs d'écrire des programmes se servant des fonctionnalités desdits services. Par exemple :

- l'API d'un site de données ouvertes permet à des programmes de récupérer ces données et de les exploiter pour créer des applications web ou smartphone par exemple.
- l'API d'un service de paiement en ligne fournit ses services à d'autres applications qui peuvent alors intégrer ce service de paiement en ligne (sans avoir besoin de le programmer soi-même) ;
- l'API d'OpenAI permet à des programmes d'utiliser les modèles d'IA développés par OpenAI (ChatGPT, DALL-E, ...)
- une API d'application météorologique permet à des programmes de récupérer des informations sur le temps, la température pour des villes données ;
- l'API de X (Twitter) permet à des programmes de récupérer des tweets et de les intégrer à un autre site ;
- l'API de Spotify permet de créer des applications basées sur les données de Spotify (morceaux en streaming, playlists, recommandations, etc.) ;
- etc.

> Toutes les API ne sont pas nécessairement accessibles via des services Web. En effet, dans l'industrie contemporaine du logiciel, les applications informatiques se servent de nombreuses interfaces de programmation, car la programmation réutilise des briques de fonctionnalités fournies par des logiciels tiers. Par exemple, le module `sqlite3` de Python est en réalité une API pour interagir avec les bases de données au format SQLite ; le module `os` permet d'interagir avec le système d'exploitation via son API, etc.

API Web

Les API disponibles via des services Web fonctionnent selon le modèle client-serveur. Elles utilisent désormais, pour la plupart, le format JSON (pour *JavaScript Object Notation*, soit « notation d'objet JavaScript ») pour échanger des informations (auparavant c'est le format XML qui était utilisé).

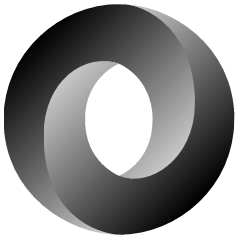


Fonctionnement d'une API Web

Crédits : Cours OpenClassrooms Adoptez les API REST pour vos projets web, diffusé sous licence CC BY-SA, Kassandre Pedro & Raye Schiller.

On va étudier rapidement le format JSON avant de voir quelques exemples d'utilisation.

Données échangées au format JSON



Logo du format JSON

Crédits : Douglas Crockford, Public domain, via Wikimedia Commons

Lorsqu'un programme (consommateur) utilise une API pour intéragir avec un programme (fournisseur), ces deux programmes échangent des données selon le modèle client/serveur. Ces données sont la plupart du temps au format `JSON` qui est un format de données structurées (structure arborescente) utilisant la syntaxe des objets du langage JavaScript.

Le format JSON est un *fichier texte* facilement manipulable par d'autres langages (que JavaScript) car il est basé sur des paires clé/valeur et est donc très proche d'un point de vue syntaxique des dictionnaires de Python, des tableaux associatifs de PHP, etc.

Voici un exemple de fichier JSON appelé `BobAndFriends.json` :

```
{
    "name":  "Bob",
    "favoriteNumber" : 3,
    "isProgrammer": false,
    "defaults": null,
    "hobbies": {
        "music": ["guitar", "piano", "trompet"],
        "sport": ["football", "table tennis"]
        },
    "friends": [{
        "name":  "Mary",
        "favoriteNumber" : 100,
        "isProgrammer": true,
        "defaults": null,
        "hobbies": {
            "cinema": ["action movies", "romantic movies"],
            "sport": ["baseball"]
            }
        },
```

```
        {
        "name":  "John",
        "favoriteNumber" : 0,
        "isProgrammer": true,
        "defaults": ["sore loser", "impatient"],
        "hobbies": {
            "music": ["saxophone", "flute"],
            "cinema": ["horror movies", "romantic movies"]
            }
        }]
}
```

Remarques importantes :

- Les clés doivent absolument êtres écrites entre des guillemets doubles ("") et non entre apostrophes (') ;
- Les valeurs peuvent être :
    - des booléens : `true` ou `false` (en minuscule) ;
    - `null` : constante spéciale indiquant une absence de valeur ;
    - des nombres entiers ou flottants ;
    - des chaînes de caractères ;
    - des tableaux délimités par des crochets (comme en Python) ;
    - des objets JSON délimités par des accolades (comme les dictionnaires de Python).

Manipuler un fichier JSON avec Python

Pour importer dans Python le contenu d'un fichier JSON, on peut utiliser la fonction `load` du module `json` à partir d'un fichier ouvert.

```
import json

# ouverture du flux de lecture
fichier = open("BobAndFriends.json")
# mémorisation du fichier json dans le dictionnaire `contenu`
contenu = json.load(fichier)
# on ferme le flux de lecture
fichier.close()
```

La variable `contenu` est un alors un dictionnaire Python :

```
>>> type(contenu)
<class 'dict'>
>>> contenu
{'name': 'Bob', 'favoriteNumber': 3, 'isProgrammer': False, 'defaults': None, 'hobbies': {'music': ['g
```

Remarque : Vous constaterez que la fonction `load` du module `json` a transformé le mot `null` du fichier JSON en la constante `None` de Python et a transformé `true` et `false` en les booléens `True` et `False` de Python. Les doubles guillemets ont aussi été transformés en simples guillemets comme le permet la syntaxe Python.

On peut alors facilement manipuler les données comme n'importe quel dictionnaire Python :

```
>>> contenu['hobbies']
{'music': ['guitar', 'piano', 'trompet'], 'sport': ['football', 'table tennis']}
>>> contenu['hobbies']['music'][1]
piano
```

Exemples de requêtes vers une API en Python

Les exemples ci-dessous résument ce qui est fait de manière détaillée dans le TP de ce chapitre, que vous allez faire pour terminer.

À faire

Faire le TP sur l'utilisation d'une API accessible sur Capytale avec le code fournit par votre enseignant. Le TP est également dans la bibliothèque publique, il est intitulé *TNSI/Th4/Ch3 : Utiliser une API [TP]*.

Utilisation du module `requests`

Pour exécuter des requêtes en Python on peut utiliser la bibliothèque `requests` : documentation officielle (voir en priorité la section *Quickstart*) ou une traduction française d'une ancienne version (problèmes potentiels de compatibilité pour des fonctionnalités avancées).

Pour installer `requests`, il suffit d'exécuter la commande suivante dans un Terminal :

```
pip install requests
```

Requêtes vers une API de données ouvertes

La ville d'Angers fournit une API permettant de récupérer différents jeux de données sur la ville d'Angers : https://data.angers.fr/pages/home/

Placer des parkings sur une carte

Par exemple, on peut récupérer des données sur les parkings de la ville via une requête GET à l'URL : https://data.angers.fr/api/explore/v2.1/catalog/datasets/angers_stationnement/records?limit=20.

On peut alors créer un programme qui récupère ces données :

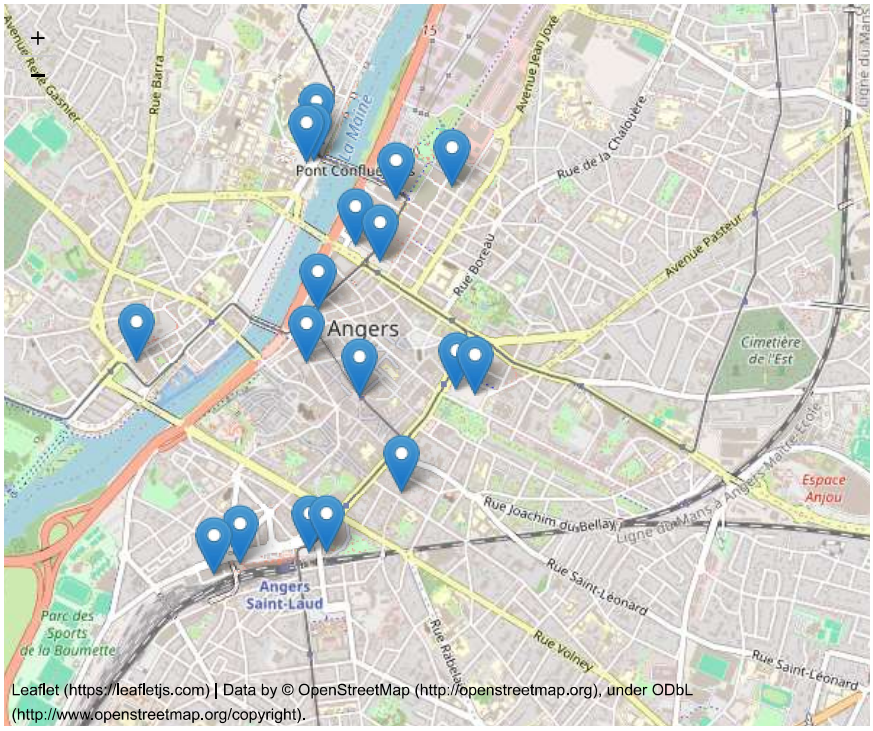
```
import requests

url = "https://data.angers.fr/api/explore/v2.1/catalog/datasets/angers_stationnement/records?limit=20"
reponse = requests.get(url)  # envoi de la requête GET
parkings = reponse.json()  # conversion de la réponse au format JSON en un dictionnaire Python
```

Le dictionnaire `parkings` contient alors toutes ces données :

```
>>> parkings
{'total_count': 18,
 'results': [
    {'id': '49007-P-010', 'nom': 'Parking du Mail', 'insee': '49007', 'adresse': '2 AVENUE DU ONZE NOV
    {'id': '49007-P-011', 'nom': 'Parking St Laud', 'insee': '49007', 'adresse': '7 ESPLANADE DE LA GA
    ...
 ]
}
```

On peut par exemple utiliser ces données pour créer une carte (grâce au module `folium` qui exploite l'API d'OpenStreetMap) pour positionner les différents parkings :



Connaître la disponibilité en temps réel dans les parkings

Un autre jeu de données permet de récupérer les données en temps réel sur les places disponibles dans ces différents parkings. On peut alors créer une application qui affiche cela pour faciliter le stationnement des usagers :

```
Parking Mail : il reste 177 places
Parking Bressigny : il reste 73 places
Parking Larrey : il reste 23 places
Parking Confluences : il reste 15 places
Parking Saint Serge Patinoire : il reste 170 places
Parking Leclerc : il reste 106 places
Parking Mitterrand Maine : il reste 85 places
Parking Quai : il reste 61 places
Parking Ralliement : il reste 153 places
Parking Republique : il reste 161 places
Parking Moliere : il reste 89 places
Parking Berges De Maine : il reste 119 places
Parking Marengo : il reste 242 places
Parking Saint Laud 2 : il reste 331 places
Parking Saint Laud : il reste 31 places
Parking Haras Public : il reste 68 places
Parking Maternite : il reste 44 places
Parking Mitterrand Rennes : il reste 56 places
```

Traduire des textes d'une langue à une autre

L'API **LibreTranslate** (voir code source : https://github.com/LibreTranslate/LibreTranslate) permet de traduire un texte (et même un fichier) d'une langue à une autre.

Il est dans ce cas nécessaire d'envoyer au serveur le texte de départ à traduire. Pour cela, on utilise une requête de type POST en passant en paramètre le texte à traduire (mais aussi la langue source et la langue cible). Une fois la traduction effectuée, le serveur renvoie un fichier JSON contenant la traduction :

```
import requests

url = "https://translate.terraprint.co/translate"
parametres = {'q': "L'informatique c'est vraiment trop bien !", 'source': 'auto', 'target': 'en'}
r = requests.post(url, data=parametres)
traduction = r.json()
```

La traduction est alors disponible dans le dictionnaire `traduction` :

```
>>> traduction
{'detectedLanguage': {'confidence': 97.0, 'language': 'fr'}, 'translatedText': 'Computer science is so
>>> traduction['translatedText']
Computer science is so good!
```

Bilan

- Une API est une interface qui permet de faire communiquer deux programmes informatiques entre eux.
- Les services disposant d'une API permettent donc à tous les autres de récupérer des données afin de les utiliser dans ses propres applications.
- Les API sont accompagnées d'une documentation qui détaille comme fonctionne l'API : seul l'interface du service est nécessaire, nul besoin de connaître l'implémentation qui se cache derrière.
- La plupart des API disponibles via une interface Web fonctionnent selon le modèle client-serveur : on les contacte via une requête HTTP (de type GET, POST, etc.) et la réponse contient les données demandées, le plus souvent au format JSON (fichier texte dont la syntaxe est celle des objets du langage JavaScript).
- Des modules Python comme `requests` permettent d'interroger ces API, libre à chacun d'utiliser les données récupérés pour en faire quelque chose d'utile.
- En réalité, la plupart des applications complexes font intervenir et interagir plusieurs API :
  - voir cette vidéo de *Cookie connecté* sur les API : https://youtu.be/T0DmHRdtqY0
  - pour aller plus loin, la vidéo *Les APIs pour débutants* du Wagon présente les API et donne des exemples concrets d'utilisation : https://youtu.be/0FQ6w4CO5Nw.

Références :

- Documentations des modules `json` et `requests` : https://docs.python.org/fr/3/library/json.html et https://requests.readthedocs.io/en/latest/
- Définition de la CNIL d'une API
- Articles Wikipédia : Interface de programmation, JSON

Germain BECKER, Lycée Emmanuel Mounier, ANGERS

Voir en ligne : info-mounier.fr/terminale_nsi/langages_prog/utiliser-une-api