API Web / Python

Laurent Signac – CC-BY-SA – 21-01-21 1604 5f9c0924bb5e37b5352d

La plupart des services Web proposent maintenant une API (Application Programming Interface), rendant le service accessible à un programme plutôt qu'à une personne derrière une machine. C'est grâce à l'existence de ces API qu'il existe par exemple plusieurs clients (qui ne sont pas des navigateurs) pour un service particulier (Twitter par exemple).

L'utilisation de certains services nécessite une phase d'authentification (Facebook, Twitter, la plupart des services Google), et d'autres non, comme OpenStreetMap, OpenFoodFacts… L'utilisation peut être gratuite ou payante.

L'idée derrière ces API est d'accéder via une méthode GET, POST ou PUT du protocole HTTP (voir *Couche Application*) à une URL particulière permettant d'effectuer une opération ou de récupérer un résultat dans un format facile à analyser (par exemple JSON ou XML).

Le site web openfoodfacts.org[1] propose par exemple une API permettant d'obtenir des informations sur des produits alimentaires. La requête (GET) est de la forme : https://world.openfoodfacts.org/api/v0/product/code-barre.json

Le champ `code-barre` doit être rempli avec le code du produit.

Le résultat obtenu est au format JSON, facilement exploitable par un programme informatique :


```
{
  "product": {
    "amino_acids_tags": [],
    "ingredients_text": "Café torréfié moul'YArabica Origine Pérou biologique et
                         Max Havelaar Ingrédient : 100% café Arabica Origine Pérou
                         certifié biologique et Max Havelaar.",
    "nutriscore_grade": "b",
    "unique_scans_n": 28,
    "nutriments": {
      "saturated-fat": 0,
      "proteins_100g": 0,
      "sodium": 0,
      "energy_100g": 0,
    },
    "product_name_fr": "Café Pur Arabica Pérou Doux Et Fruité Bio",
    "image_front_thumb_url":
       "https://static.openfoodfacts.org/images/products/20713096/front_fr.19.100.jpg",
  },
  "code": "20713096",
  "status": 1,
  "status_verbose": "product found"
}
```


Le dictionnaire contient par exemple le nom du produit, un lien vers une photo de l'emballage, et différentes valeurs nutritionnelles (ici à 0).

Voici un exemple de programme Python qui récupère ces données et les affiche.

```
import urllib.request as req
url = "https://world.openfoodfacts.org/api/v0/product/20713096.json"
u = req.urlopen(url)
content = u.read()
```

[1] https://openfootfacts.org

```
jsonstr = content.decode('utf8')
print(jsonstr)
```

Dans le code qui précède `content` contient un objets de type `bytes`. On le transforme en chaîne de caractères avec la méthode `decode`, en supposant que l'encodage utilisé est ici UTF8. La chaîne obtenue est au format JSON est peut être affichée ou transformée en objet Python (ici un dictionnaire) à l'aide du module `json` :

```
import json
data = json.loads(jsonstr)
urlproduit = data['product']['image_front_thumb_url']
sodium = data['product']['nutriments']['sodium']
print("URL Image produit", urlproduit)
print("Teneur en sodium :", sodium)
```

Notons que le module `requests` (non inclus dans la bibliothèque standard) permet de simplifier la récupération des données (son utilisation est encouragée) :

```
import requests
url = "https://world.openfoodfacts.org/api/v0/product/20713096.json"
content = requests.get(url)
data = content.json()
sodium = data['product']['nutriments']['sodium']
print("Teneur en sodium :", sodium)
```