Gestion des données

Cours 6 - Données sur le web

Olivier Schwander
<olivier.schwander@sorbonne-universite.fr>

UPMC - LIP6

2021-2022

Service web

Site web classique

- Navigateur qui demande des pages
- Serveur qui renvoie les pages

Programmer les échanges

- Programme qui récupère des informations

Examples

- Applications mobiles

Architecture

Client/serveur

- Client: une application quelconque
- Serveur: toujours un serveur web

Réponses

- Plus des pages web
- Données structurées

Réponses

Données stockées dans divers formats

- XML
- Json
- Binaire
- Propriétaire

Utilisation

- Comme un fichier classique
- On peut oublier d'où vient l'information

Protocoles

Representational state transfer (REST)

- Question codée dans l'URL

http://example.com/age/capitaine?unit=year&format=json

Autres: SOAP et WSDL

- Plus complexe
- Plus structuré

Hypertext Transfer Protocol (HTTP)

(version ultra-simplifiée)

Requêtes GET

- Demander une information
- Requête la plus classique

Requêtes POST

- Envoyer une information
- Soumission d'un formulaire

Uniform Resource Locator (URL)

Format

http://example.com/chemin/vers/la/ressource?arg1=valeur1&arg2=valeur2

- `http`: protocole
- `example.com`: serveur
- `chemin/vers/la/ressource`: identifiant de la ressource
- `?`: tout ce qui suit est un argument
- `arg1=valeur1&arg2=valeur2`: arguments

Examples

- http://www.bing.com/search?q=http
- http://www-connex.lip6.fr/~schwander/enseignement/2015-2016/m2stat_bi/index.html

Requêtes

En général

- Récupérer des données: GET
- Envoyer des données: POST

Cas particuliers

- Petites données à envoyer: GET
- Requête compliquée: POST
- Mots de passes: POST

Arguments

- Seules les requêtes GET prennent des arguments dans l'URL
- Autre mécanisme pour POST

En Python

Requests: HTTP for Humans

Warning: Recreational use of other HTTP libraries may result in dangerous side-effects, including: security vulnerabilities, verbose code, reinventing the wheel, constantly reading documentation, depression, headaches, or even death. (source: http://docs.python-requests.org/en/master/)

Exemples

Base

```
>>> import requests
>>> result = requests.get("http://www.bing.com/search?q=http")
>>> print(result.text)
```

Json

- Requests se charge du chargement des données

```
>>> import requests
>>> r = requests.get('https://api.github.com/events')
>>> r.json()
[{u'repository': {u'open_issues': 0, u'url': 'https://github.com
```

- On récupère directement une valeur Python

Trouver le service web

Documentation

- Si le service est officiellement supporté

Reverse engineering

- Analyse des requêtes d'une page web
- Étude d'une application mobile

Services web

Cas idéal

- Gentil fournisseur de service
- Documentation, formats ouverts, accès autorisé

~~Parfois~~ Souvent

- Pas de service web public
- Format incompréhensible
- Conditions d'utilisation incompatibles
- Pas l'info qui vous intéresse

Que faire sans service web ?

Extraction des données

- La page web contient les données
- Récupérons-les manuellement

Étapes

- Faire des requêtes HTTP
- Extraire des données dans le HTML

Démarche

Comprendre les URL

- Surfer normalement sur le site
- Remplir les formulaire, regarder où on arrive
  http://www.bing.com/search?q=chat
  http://www.bing.com/search?q=chien
- Intuition, essais, erreurs

Comprendre la structure du HTML

- À quelle endroit du code est l'info ?
- Comment l'identifier à coup sûr ?
- Comment est-elle stockée ?

HyperText Markup Language (HTML)

(version ultra-simplifiée)

Du XML

- (à peu près)
- Structure d'arbre
- Des nœuds, appelés balises

Les balises

- `<p>...</p>`: paragraphe de texte
- `<a href=http://destination>titre</a>`: lien hypertexte
- `<div>...</div>`: container
- `<span>...</span>`: mise en forme
- et plus encore

Exemple

```
<!DOCTYPE html>
<html>
 <head>
  <title>
   Exemple de HTML
  </title>
 </head>
 <body>
  Ceci est une phrase avec un
    <a href="cible.html">hyperlien</a>.
  <p>
   Ceci est un paragraphe où il n'y a pas d'hyperlien.
  </p>
 </body>
</html>
```

Trouver son chemin

Attributs: meta-données sur les balises

- `id`: identifiant unique d'un nœud
- `class`: partagé par plusieurs nœuds

Utilité normale: mise en forme

- Nœud `id=maintitle` en rose avec des poneys
- Nœud `id=privacy` en tout petit
- Nœuds `class=specialoffer` en rouge clignotant

Corollaire

- On peut trouver les informations intéressantes

En python

```
>>> import requests
>>> from bs4 import BeautifulSoup
>>> url = "https://fr.wikipedia.org/wiki/Uniform_Resource_Locato
>>> page = requests.get(url)
>>> print(page.status_code)
200
>>> soup = BeautifulSoup(page.text, "html.parser")
>>> soup.title
<title>Uniform Resource Locator \u2014 Wikip\xe9dia</title>
>>> soup.title.string
u'Uniform Resource Locator \u2014 Wikip\xe9dia'
>>> soup.find("h2")
<h2>Sommaire</h2>
```

Recherche et filtres

Trouver un nœud

- `soup.find("div")`
- `soup.find("div", class="offer)`
- `soup.find("div", id="firstHeading")`

Trouver tous les nœuds

- `soup.find_all("div")`

Parents, enfants

- `.find_parents()`, `.find_parent()`
- `.content[0]`
- `.next_siblings .previous_siblings`

Inspecteur Firefox

- Accessible avec F12
- Possibilité de pointer sur la page pour trouver le nœud correspondant