Lección 18: Web Scrapping

Índice

Anterior

Web Scrapping

Web scrapping es el conjunto de técnicas utilizadas para extraer datos de páginas web. En definitiva de simular la visita a una página web por un navegador, extraer información de la web y hacerla accesible de forma estructurada. El web scrapping es útil cuando se necesita extraer información/funcionalidad de forma automática de una web que no tiene una API.

> **AVISO:** El web scrapping puede ser ilegal y no estar permitido por los términos y condiciones de algunas páginas web. Como mínimo se debería consultar el robots.txt de la página web objetivo para comprobar las limitaciones al web scrapping.

Aunque hay otras, la librería de python más conocida para hacer Web Scrapping es (Beautiful Soup)[https://www.crummy.com/software/BeautifulSoup/bs4/doc/]. Esta librería se utiliza en combinación con *requests*^.

Para instalarla:

```
pip install beautifulsoup4
```

Paso a paso

De forma general, los pasos a dar para hacer Web Scrapping son:

1. Identificar los elementos de la web

Las webs son documentos estructurados en HTML. Los elementos de la web se estructuran y definen mediante etiquetas HTML. Cada etiqueta contiene instrucciones sencillas que indican al navegador cómo dar formato al texto y a definir los diversos elementos de la página web. Al aplicar estas etiquetas de marcado a los diferentes elementos del texto, se indica al navegador cómo mostrarlos al usuario, lo que permite crear páginas web estructuradas y con un diseño coherente.

El primer paso es identificar qué elementos de la web contienen la información/datos que se quieren extraer. Para ello se visita la web con un navegador y se inspecciona el código hasta identificar los elementos HTML. De cada uno de los elementos es interesante anotar la *etiqueta HTML*, la *clase* y/o el *id*.

2. Identificar las peticiones que permiten descargar la página

Es necesario identificar qué peticiones HTTP será necesario replicar con *requests* para que el servidor devuelva la página HTML de la que se desea extraer

información. Es interante identificar la url de destino, parámetros en url, el método HTTP (GET, POST, …), las cabeceras (user-agent, …) y datos necesarios para que el servidor devuelva la página HTML adecuada.

Para ello, la mejor herramienta es de nuevo el navegador y las herramientas de desarrollo.

3. Replicar la petición con *requests*

Ya en Python se utilizará *requests* para replicar la petición HTTP necesaria:

```
import requests
from bs4 import BeautifulSoup

headers = {
        'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0'
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
        'Accept-Language': 'es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3',
        'DNT': '1',
        'Connection': 'keep-alive',
        'Cookie': 'cookie-agreed=0',
        'Upgrade-Insecure-Requests': '1'
    }

url = ''
req = requests.get(url, headers=headers)
print(req.url)
if req.status_code == 200:
    html = BeautifulSoup(req.text, 'html.parser') # Beautiful Soup para parsear el html y po
```

4. Buscar y parsear con *Beautiful Soup* los elementos identificados

Con *BeautifulSoup* ahora es necesario localizar los elementos interesantes y parsear el contenido:

```
resoluciones = html.find_all('div', {'class': 'layout__region--content'})
    for resolucion in resoluciones:
        titulo = ''
        link = ''
        fecha = ''
        tag_titulo = resolucion.find('div', {'class': 'field--name-title'})
        if tag_titulo:
            titulo = tag_titulo.getText()
        tag_link = resolucion.find('a',{'class': ''})
        print(tag_link)
        if tag_link:
            link = '' + tag_link['href']
        tag_fecha = resolucion.find('time')
```

```
        if tag_fecha:
            fecha = tag_fecha['datetime']
        print((titulo,link,fecha))
```

Si fuera, necesario podrían descargarse todos los enlaces:

```
import requests
from bs4 import BeautifulSoup
import csv
from urllib.parse import urlparse


headers = {
        'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:84.0) Gecko/20100101 Firefox/84.0'
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.
        'Accept-Language': 'es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3',
        'DNT': '1',
        'Connection': 'keep-alive',
        'Cookie': 'cookie-agreed=0',
        'Upgrade-Insecure-Requests': '1'
    }


def download(url, filepath):
    with requests.get(url, stream = True, timeout=200) as r:
        with open(filepath, 'wb') as f:
            for chunk in r.iter_content(chunk_size=1024):
                f.write(chunk)



url = '' # TODO: Rellenar url objetivo
parsed_url = urlparse(url)
req = requests.get(url, headers=headers, timeout=200)
if req.status_code == 200:
    html = BeautifulSoup(req.text, 'html.parser') # Beautiful Soup para parsear el html y po
    with open('./datos.csv', mode='wt', encoding='utf-8') as f:
        csv_file = csv.writer(f, delimiter = ',')
        csv_file.writerow(['titulo', 'enlace', 'fecha'])
        resoluciones = html.find_all('div', {'class': 'layout__region--content'})
        for resolucion in resoluciones:
            titulo = ''
            link = ''
            fecha = ''
            tag_titulo = resolucion.find('div', {'class': 'field--name-title'})
            if tag_titulo:
                titulo = tag_titulo.getText()
```

```
tag_link = resolucion.find('a',{'class': ''})
if tag_link:
    link = f'{parsed_url.scheme}://{parsed_url.hostname}/' + tag_link['href']
tag_fecha = resolucion.find('time')
if tag_fecha:
    fecha = tag_fecha['datetime']
if all([titulo,link,fecha]):
    csv_file.writerow([titulo, link, fecha])
    download(link, f"{titulo}.pdf")
```

Proyecto de Ejemplo: Obtener los datos de una empresa a partir del NIF con WebScrapping

El **objetivo del proyecto** es crear un **módulo de python** con una función que pueda reutilizarse desde otros proyectos y que el módulo pueda llamarse por sí mismo para obtener los datos de una empresa a partir de su NIF.

El módulo se llamará empresas.py y contendrá una única funcion:

1. Una función *get*. Recibe como parámetro de entrada una lista de NIFs a buscar. Devuelve un diccionario/json con los datos de las empresas que haya localizado.

El módulo se podrá llamar desde línea de comando especificando tantos NIFs como se quieran: NIF1 NIF2 NIF3 …

Solución: Encuentra una posible solución en (empresas)[https://github.com/mercaderd/curso_python/tree/main/18_WebScrapping/empresas]

Siguiente