Espejos de sitios webs con python, curl y wget

Como tal vez sepamos, con este último **wget** y el parámetro **--mirror** bien podemos realizar un sitio web espejo. El asunto está en que si necesitamos extraer datos específicos y en hacerlo plenamente funcional sustituyendo el motor que lo publica pues para eso es esta entrada. ***Veamos.***

Entorno con Python 3

Para garantizar que nuestro código corra en cualquier máquina debemos establecer entornos virtuales de programación, esencialmente esto se logra instalando **pip3** (permite instalar diversos programas hechos en Python) junto con otras librerías de desarrollo, acompañados finalmente de **python3-venv**:

```
$ sudo apt-get install -y python3-pip
$ sudo apt-get install build-essential libssl-dev libffi-dev python-dev
$ sudo apt-get install -y python3-venv
```

Inicializando un proyecto

Una vez tengamos instalado el entorno de virtualización, creamos una carpeta con el nombre de nuestro proyecto e inicializamos; la última línea será la que tenemos que repetir cada vez que vayamos a trabajar en ese proyecto y su entorno. Además, con solo usar los comandos **python** y **pip** haremos uso de manera automática del entorno que hayamos configurado:

```
$ mkdir nombre_proyecto
$ cd nombre_proyecto
$ source nombre_proyecto/bin/activate
```

Instalación de Requests, Beautiful Soup y httplib2

Para activar de nuevo nuestro entorno entramos en la carpeta donde hallamos inicializado el proyecto:

```
$ cd nombre_proyecto
$ . my_env/bin/activate
(nombre_proyecto) $
```

Podremos entonces instalar Requests a nuestro entorno:

```
(nombre_proyecto) $ pip install requests
```

Para luego instalar Beautiful Soup:

```
(nombre_proyecto) $ pip install beautifulsoup4
```

Por último instalamos **httplib2**:

```
$ pip install httplib2
```

Descargando una página web

Ya en este punto comenzamos a crear nuestro espejo de sitio web con una simple página, para ello, todavía ejecutando nuestro entorno:

```
(e)  prewebs  $ python
```

```
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> url = 'https://pypi.org/project/beautifulsoup4/'
>>> page = requests.get(url)
>>> page.status_code
200
>>> page.text
```

Desgranando con Beautiful Soup

Acá comenzamos a utilizar Beautiful Soup para hallar las secciones que nos interesan:

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(page.text, 'html.parser')
```

De esta manera podremos ver de manera muy bien ordenada el código HTML:

```
>>> print(soup.prettify())
```

También podemos ver solamente los párrafos con la siguiente instrucción:

```
>>> soup.find_all('p')
```

Como el resultado es una lista de Python, podremos llamar específicamente un párrafo específico:

```
>>> soup.find_all('p')[5].get_text()
'Author: Leonard Richardson'
```

Para este ejemplo podemos extraer las diferentes versiones que ha tenido Beautiful Soup si especificamos con el siguiente parámetro:

```
>>> soup.find_all(class_='release__version')
[

   4.9.3



,

   4.9.2



,

   4.9.1
   .
   .
   .


]
```

Como podemos observar, ya tenemos las herramientas necesarias para extraer solamente los enlaces web y poder así extraer el resto de las páginas del sitio web.

Utilizando httplib2

En la sección anterior expusimos como tener en memoria la página web completa para poder examinar todo su contenido, pero obvio, por ser completa "gasta" mucha más memoria. Si solamente necesitamos extraer los enlaces web podemos escribir el siguiente código:

```
>>> import httplib2
>>> from bs4 import BeautifulSoup, SoupStrainer
>>> http = httplib2.Http()
```

```
>>> status, response = http.request('https://pypi.org/project/beautifulso
up4/')
>>> for link in BeautifulSoup(response, parse_only=SoupStrainer('a')):
...   if link.has_attr('href'):
...     print(link['href'])
#content
/
/help/
/sponsor/
/account/login/
/account/register/
/help/
/sponsor/
/account/login/
/account/register/
/project/beautifulsoup4/
#description
#history
#files
http://www.crummy.com/software/BeautifulSoup/bs4/
http://www.crummy.com/software/BeautifulSoup/bs4/download/
https://libraries.io/pypi/beautifulsoup4
...
```

Debemos tener en cuenta que recibiremos varios tipos de enlaces diferentes:

- Los que comienzan con "/", que son la base URL más el ítem recibido, ***son los que estamos buscando***.
- Los que comienzan con "#" que son anclas en la misma página web, estos los podemos ignorar.
- Los que comienzan con "http", que son enlaces externos los cuales también podemos obviar.
- Los enlaces para enviar correo electrónico, comienzan con "mailto".

Dicho esto, el código que necesitamos es:

```
>>> for link in BeautifulSoup(response, parse_only=SoupStrainer('a')):
...   if link.has_attr('href'):
...     if link['href'].startswith("/"):
...       print(link['href'])
```

```
...
```

Incluso podemos refinar más la búsqueda, si buscamos que además el enlace contenga una palabra clave:

```
>>> for link in BeautifulSoup(response, parse_only=SoupStrainer('a')):
...   if link.has_attr('href'):
...     if link['href'].startswith("/") and link['href'].find("palabra_cl
ave")>0 :
...       print(link['href'])
...
```

O también dos palabras claves al mismo tiempo:

```
>>> for link in BeautifulSoup(response, parse_only=SoupStrainer('a')):
...   if link.has_attr('href'):
...     if link['href'].startswith("/") and (link['href'].find("p_clave1"
)>0 or link['href'].find("p_clave2")>0):
...       print(link['href'])
...
```

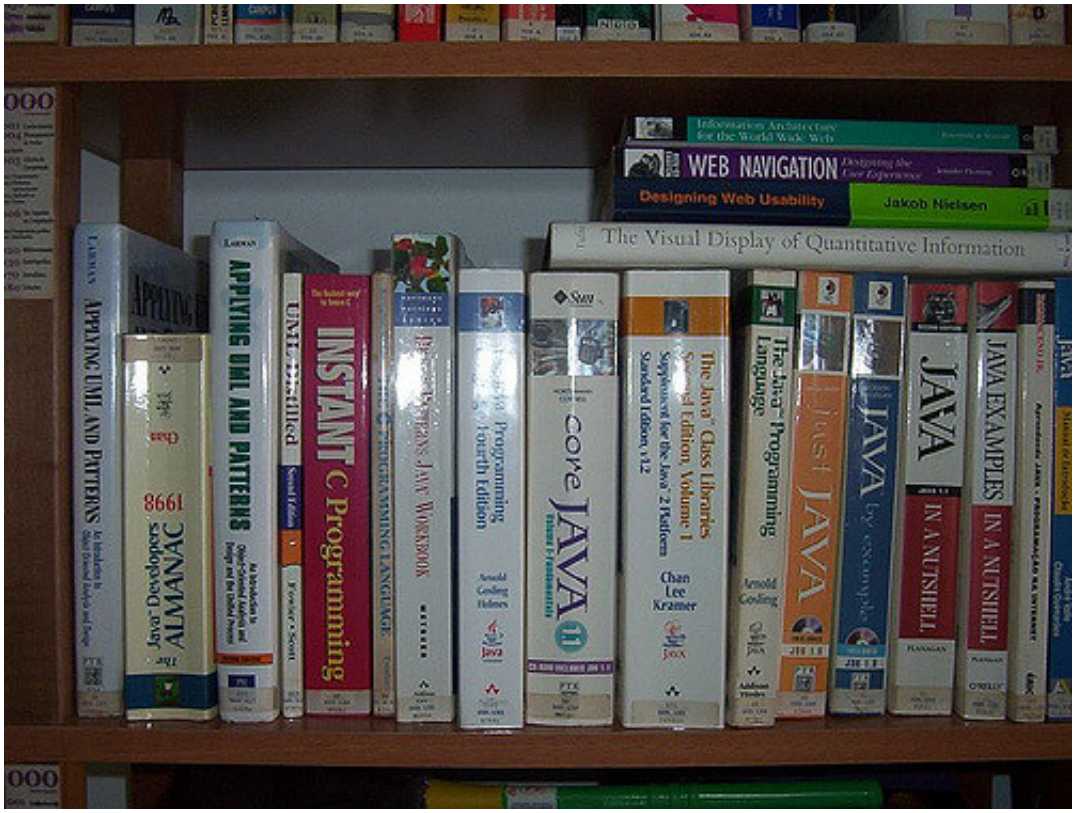
Guardando los enlaces en un fichero

A fin de que **wget** pueda comenzar a descargar las páginas que nos interesan, primero debemos pasar el URL en dos partes, la base y la página principal a descargar (generalmente "index.html") para poder escribir el URL completo en un archivo:

```
>>> import httplib2
>>> from bs4 import BeautifulSoup, SoupStrainer
>>> http = httplib2.Http()
>>> URL_base = 'https://pypi.org'
>>> status, response = http.request('https://pypi.org/project/beautifulso
up4/')
>>> f = open("enlaces.txt", "a")
```

```
>>> for link in BeautifulSoup(response, parse_only=SoupStrainer('a')):
...   if link.has_attr('href'):
...     if link['href'].startswith("/") and (link['href'].find("p_clave1"
)>0 or link['href'].find("p_clave2")>0):
...       print(link['href'])
...       f.write(URL_base + link['href'])
>>> f.close()
```

a



Fuentes consultadas

En idioma inglés



- «How To Install Python 3 and Set Up a Local Programming Environment on Ubuntu 16.04» https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-local-programming-environment-on-ubuntu-16-04.
- «How To Work with Web Data Using Requests and Beautiful Soup with Python 3» https://www.digitalocean.com/community/tutorials/how-to-work-with-web-data-using-requests-and-beautiful-soup-with-python-3.
- «Beautiful Soup Documentation» https://www.crummy.com/software/BeautifulSoup/bs4/doc/.
- «httplib2 0.19.0» https://pypi.org/project/httplib2/ .
- «retrieve links from web page using python and Beautiful Soup» https://stackoverflow.com/questions/1080411/retrieve-links-from-web-page-using-python-and-beautifulsoup.
- «How To Scrape Web Pages with Beautiful Soup and Python 3» https://www.digitalocean.com/community/tutorials/how-to-scrape-web-pages-with-beautiful-soup-and-python-3.
- «How to make an offline mirror copy of a website with wget» https://alvinalexander.com/linux-unix/how-to-make-offline-mirror-copy-website-with-wget/.
- «GNU Wget 1.21.1-dirty Manual» https://www.gnu.org/software/wget/manual/wget.html#index-required-images_002c-downloading.
- «How to Use wget, the Ultimate Command Line Downloading Tool» https://www.howtogeek.com/281663/how-to-use-wget-the-ultimate-command-line-downloading-tool/.
- «rsnapshot» https://rsnapshot.org/.
- «Rsnapshot» https://ubuntu.com/server/docs/tools-rsnapshot.
- «rsnapshot» https://wiki.archlinux.org/index.php/Rsnapshot.
- «curl.1 the man page» https://curl.se/docs/manpage.html#-z.

- «curl tutorial» https://curl.se/docs/manual.html.
- «CURL SPEAKS ETAG» https://daniel.haxx.se/blog/2019/12/06/curl-speaks-etag/.
- «Python File Open» https://www.w3schools.com/python/python_file_open.asp.
- «».
- «».
- «».
- «».



Crédito de la imagen Gerd Altmann, trabajo, licencia de uso:Pixabay