Web Scraping Formulario

Beautiful Soup
Selenium
Scrapy

Frank Andrade

Web Scraping Formulario

El web scraping nos permite extraer data de la web. Antes de aprender Beautiful Soup, Selenium o Scrapy, vamos a revisar conceptos básicos de HTML.

HTML básico para Web Scraping

Analicemos el siguiente elemento HTML.

Este es solo un elemento HTML, pero el documento HTML detrás de una página web tiene varios elmentos como este.

Código HTML ejemplo

```
<article class="main-article">
 <h1> Titanic (1997) </h1>
 <p class="plot"> 84 years later ... </p>
 <div class="full-script"> 13 meters. You ... </div>
</article>
```

El documento HTML está estructurado con "nodos". Cada rectángulo debajo representa un nodo (elemento, atributo o texto)

- "Hermanos" son nodos con los mismos padres.
- El hijo de un nodo y los hijos de sus hijos son llamados sus "descendientes". Del mismo modo, el padre de un nodo y el padre de su padre son llamados "ancestros".
- Es recomendado buscar elementos en este orden
  a. ID
  b. Class name
  c. Tag name
  d. Xpath

Beautiful Soup

Flujo de Trabajo

```
Importar librerías
 from bs4 import BeautifulSoup
 import requests

Obtener páginas
 result=requests.get("www.google.com")
 result.status_code #obtener status
 result.headers #obtener encabezados

Contenido de la página
 contenido = result.text

Crear soup
 soup=BeautifulSoup(contenido,"lxml")

HTML en formato legible
 print(soup.prettify())

Encontrar un elemento
 soup.find(id="mi_id")

Encontrar elementos
 soup.find_all("a")
 soup.find_all("a","css_class")
 soup.find_all("a",class_="mi_class")
 soup.find_all("a",attrs={"class":
                          "mi_class"})

Obtener texto
 ejemplo=elemento.get_text()
 ejemplo=elemento.get_text(strip=True,
                           separator=' ')

Obtener atributos
 ejemplo = elemento.get('href')
```

XPath

Necesitamos aprender XPath para hacer web scraping con Selenium y Scrapy.

XPath Sintaxis

Un XPath usualmente contiene un tag, nombre de atributo y valor de atributo.

```
//tag[@Atributo="Valor"]
```

Veamos algunos ejemplos de como localizar el elemento article, el titulo de la película y transcript del código HTML que vimos antes.

```
//article[@class="main-article"]
//h1
//div[@class="full-script"]
```

XPath Funciones y Operadores

XPath funciones

```
//tag[contains(@Atributo, "Valor")]
```

XPath Operadores: and, or

```
//tag[(expresion 1) and (expresion 2)]
```

XPath Caracteres Especiales

/	Selecciona los hijos del nodo ubicado a la izquiera de este caracter
//	Especifica que el nodo a emparejar puede estar en cualquier nivel del documento
.	Especifica que el contexto actual debería ser usado (el nodo referencia)
..	Selecciona a un nodo padre
*	Caracter comodín que selecciona todos los elementos sin importar el nombre
@	Selecciona un atributo
()	Indica una agrupación dentro de un XPath
[n]	Indica que un nodo con index "n" debe ser seleccionado

Selenium

Flujo de Trabajo

```
from selenium import webdriver
web="www.google.com"
path='introduce ruta del chromedriver'
driver = webdriver.Chrome(path)
driver.get(web)
```

Encontrar un elemento
```
driver.find_element_by_id('nombre')
```

Encontrar elementos
```
driver.find_elements_by_class_name()
driver.find_elements_by_css_selector
driver.find_elements_by_xpath()
driver.find_elements_by_tag_name()
driver.find_elements_by_name()
```

Cerrar driver
```
driver.quit()
```

Obtener el texto
```
data = elemento.text
```

Espera Implícita
```
import time
time.sleep(2)
```

Espera Explícita
```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

WebDriverWait(driver,5).until(EC.element_to_be_clickable((By.ID,
'id_name'))) #esperar 5 segundos hasta poder encontrar elemento
```

Opciones: Headless mode, cambiar tamaño de ventana
```
from selenium.webdriver.chrome.options import Options
opciones = Options()
opciones.headless = True
opciones.add_argument('window-size=1920x1080')
driver = webdriver.Chrome(path,options=opciones)
```

Puedes encontrar tutoriales sobre
Python en YouTube o Medium
YT: www.youtube.com/andradefrank
Medium: frank-andrade.medium.com

Scrapy

Scrapy es el framework más complete de web scraping en Python. Para configurarlo revisa la documentación de Scrapy.

Crear un Proyecto y Spider
Para crear un nuevo proyecto, corre el siguiente comando en el terminal o cmd
```
scrapy startproject mi_primer_spider
```
Para crear un nuevo spider, primero cambia el directorio
```
cd mi_primer_spider
```
Crear un spider
```
scrapy genspider ejemplo ejemplo.com
```

La plantilla básica
Cuando creamos un spider, obtenemos una plantilla con el siguiente contenido.

```
import scrapy
class ExampleSpider(scrapy.Spider):
    name = 'ejemplo'
    allowed_domains = ['ejemplo.com']
    start_urls = ['http://ejemplo.com/']

    def parse(self, response):
        pass
```

Clase

Método Parse

La clase es contruida con la data que introducimos en el comando previo, pero el método parse tenemos que construirlo nosotros.

Buscando elementos
Para buscar elementos con Scrapy, usa el argumento "response" del método parse
```
response.xpath('//tag[@Atributo="Valor"]')
```
Obtener texto
Para obtener el elemento texto usamos text() y luego .get() o .getall(). Por ejemplo:
```
response.xpath('//h1/text()').get()
response.xpath('//tag[@Atributo="Valor"]/text()').getall()
```

Devolver la data extraída
Para ver la data extraída tenemos que usar la palabra clave yield

```
def parse(self, response):
    title = response.xpath('//h1/text()').get()

    # Devolver data extraída
    yield {'titles': title}
```

Correr el spider y exportar data a CSV o JSON
```
scrapy crawl ejemplo
scrapy crawl ejemplo -o nombre_archivo.csv
scrapy crawl ejemplo -o nombre_archivo.json
```