T. Cicchini y S. Pinto

Clase N° 19

Web scraping

Motivación

En la clase pasada, vimos que ciertas aplicaciones o páginas web tienen desarrolladas una o varias APIs que nos permiten interactuar con las mismas.

Imaginemos que, además:

- La aplicación con la que queremos trabajar nos facilita la comunicación mediante determinadas reglas

Motivación

Pero, ¿si no hay API o no la usamos?

Motivación

A no desanimarse, podemos scrapear las páginas y obtener así la info que querramos de las mismas

Motivación

Porque si las páginas **tienen** un montón de información, eso significa que...

Generalidades

Al final de esta clase, veremos cómo scrapear páginas webs. Pero para esto, resulta necesario ahondar en los siguientes temas:

- Comprender el lenguaje en el cual se montan las páginas web
- Identificar los elementos de cada página que nos interesan
- Librerías de Python que nos permitan acceder a dicho contenido
- Armado de códigos que nos permitan extraer información de manera automatizada

Estructura código HTML

HTML (Hyper Text Markup Language) es uno de los lenguajes más utilizados para la creación de páginas web.

Estructura código HTML

HTML (Hyper Text Markup Language) es uno de los lenguajes más utilizados para la creación de páginas web.

Con este lenguaje, compuesto de elementos, describimos la estructura de una página web.

Estructura código HTML

HTML (Hyper Text Markup Language) es uno de los lenguajes más utilizados para la creación de páginas web.

Con este lenguaje, compuesto de elementos, describimos la estructura de una página web.

De esta forma, un intérprete de html, como por ejemplo un navegador, va a saber cómo mostrarnos la información.

Estructura código HTML

HTML (Hyper Text Markup Language) es uno de los lenguajes más utilizados para la creación de páginas web.

Con este lenguaje, compuesto de elementos, describimos la estructura de una página web.

De esta forma, un intérprete de html, como por ejemplo un navegador, va a saber cómo mostrarnos la información.

Cada elemento etiqueta con un determinado nombre algún contenido determinado. El nombre de esta etiqueta y el atributo de la misma nos permiten encontrar el contenido buscado.

Estructura código HTML

Existen algunos tags específicos que siempre veremos:

Estructura código HTML

Existen algunos tags específicos que siempre veremos:

- ** **
- <p>
- <a>
- <h...>

Se utiliza para marcar secciones en el código

Estructura código HTML

Existen algunos tags específicos que siempre veremos:

-
- **<p>**
- <a>
- <h...>

Se utiliza para marcar párrafos

Estructura código HTML

Existen algunos tags específicos que siempre veremos:

-
- <p>
- **<a>**
- <h...>

Se utiliza para introducir links

Estructura código HTML

Existen algunos tags específicos que siempre veremos:

-
- <p>
- <a>
- **<h...>**

h1...h6 se utilizan para marcar encabezados

Estructura código HTML

Existen algunos tags específicos que siempre veremos:

-
- <p>
- <a>
- <h...>

A su vez, también tenemos algunos atributos frecuentes:

Estructura código HTML

Existen algunos tags específicos que siempre veremos:

-
- <p>
- <a>
- <h...>

A su vez, también tenemos algunos atributos frecuentes:

- **class**
- href

define la clase de un tag, lo que permite mapearlo con un estilo dado un css

Estructura código HTML

Existen algunos tags específicos que siempre veremos:

-
- <p>
- <a>
- <h...>

A su vez, también tenemos algunos atributos frecuentes:

- class
- **href**

define el link al cual efectivamente el tag nos referencia (en general dentro de un tag <a>)

Estructura código HTML

Header

```
<!DOCTYPE html>
<html>

    <head>
        <title> Página HTML </title>
    </head>

    <body>

        <div class="parrafos">
                <p> Algún párrafo </p>
                <p> Otro párrafo </p>
        </div>

        <a href="otra_pagina"> Link a otra página </a>

    </body>

</html>
```

Cuerpo

Página HTML

Algún párrafo

Otro párrafo

Link a otra página

Vista en el navegador

Mini ejemplo de estructura HTML

https://www.w3schools.com/html/default.asp

CSS

CSS (Cascading Style Sheets) es el lenguaje de programación utilizado para estilizar una página web.

https://www.w3schools.com/css/default.asp

CSS

CSS (Cascading Style Sheets) es el lenguaje de programación utilizado para estilizar una página web.

Es decir, le da información al intérprete de html sobre cómo tiene que mostrar los tags según sus atributos y nombres.

https://www.w3schools.com/css/default.asp

CSS

CSS (Cascading Style Sheets) es el lenguaje de programación utilizado para estilizar una página web.

Es decir, le da información al intérprete de html sobre cómo tiene que mostrar los tags según sus atributos y nombres.

Ahorra bastante trabajo en términos de homogeneizar páginas web e históricamente fue un salvataje para quienes programaban aplicaciones. Básicamente, es posible darle distintos estilos a una página cambiando el css en vez de cambiando el html directamente.

https://www.w3schools.com/css/default.asp

JavaScript

Lenguaje de scripting de las páginas web.

Es el que usualmente se utiliza para definir el efecto de ciertas acciones, como por ejemplo, apretar un botón, posicionar el puntero sobre alguna sección, ir a hacia el final de una página, etc.

Es un lenguaje orientado a objetos: la página donde estamos, la ventana, los botones, entre otros, son objetos con métodos y atributos asociados.

https://www.w3schools.com/js/

JavaScript

Importante! Toda acción que hacemos sobre una paǵina web es un método asociado a un objeto.
Es decir, todo se puede programar explícitamente!

Por ejemplo: scrollear una página hasta una posición (X,Y) es llamar al método scrollTo() del objeto window, es decir:

window.scrollTo(X,Y)

Esto lo podemos probar en la consola web (F12 en cualquier navegador) .

https://www.w3schools.com/js/

Pero entonces, ¿cuántos lenguajes necesito?

Pero entonces, ¿cuántos lenguajes necesito?

Pero entonces, ¿cuántos lenguajes necesito?

En realidad, en una página web podremos encontrarnos con los tres lenguajes, cada uno utilizado con la siguiente finalidad:

- **HTML** para definir el **contenido** de las páginas
- **CSS** para especificar el **formato** en el cual se mostrará el contenido
- **JavaScript** para programar el **comportamiento** de las mismas

¿Cómo extraer datos de las páginas?

Para extraer datos de las páginas con python tenemos las siguientes librerías:

- Requests (misma librería que para las APIs)
- BeautifulSoup
- Scrapy
- Selenium

BeautifulSoup

Parsea el contenido de la respuesta de un request.get como un html. Nos sirve para:

- Navegar por el html usando la estructura de tags.

- Encontrar tags con determinados nombres o atributos.

https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Selenium

Permite comunicarse con el javascript de la página, es decir, con el contenido dinámico de una página, por ejemplo:

- Apretar botones
- Scrollear

Desventajas:

- Es más lento dado que simula un navegador

Importante! Esperar a que se cargue la página cada vez que interactuemos si la navegación es dinámica.

https://selenium-python.readthedocs.io/

Scrapy

Permite scrapear al mismo tiempo varias páginas, por lo que la principal ventaja es que:

- Es muy rápido... pero muy rápido...
- Permite redireccionar la salida a archivos bien formateados (json, xml)

Desventajas:

- No maneja tan bien contenido dinámico (al menos si lo hace, no es tan fácil de implementar).
- Es un toque más difícil de aprender (la curva de aprendizaje es más lenta).

https://scrapy.org/

¿Cómo identificamos dónde está cada elemento?

AFIP. Cambios en el monotributo: qué pasará desde hoy con los montos a pagar

La AFIP aprobó esta noche la resolución que reglamenta los cambios en el sistema ~~impositivo~~ simplificado; qué trámites habrá que hacer

Por Silvia Stang

Las 7 respuestas. El detrás de escena de una

- Nos paramos sobre un dato que queremos extraer.
- Botón derecho del mouse.
- **“Inspeccionar elemento”**

¿Cómo identificamos dónde está cada elemento?

- Se abre un “inspector” que nos dice en qué parte del código HTML está la información que seleccionamos.

¿Cómo identificamos dónde está cada elemento?

Con las librerías de python mencionadas vamos a tener múltiples formas de identificar un elemento:

- Por tag asociado.
- Por atributos (como por ejemplo, la clase).

Además a veces en vez de apuntarle a un elemento de interés, le vamos a apuntar a algún elemento padre/madre y a partir de ahí navegar.

Otras formas de buscar en una página

Xpath: forma compacta para buscar elementos.

Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore **Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
</body>
</html>
```

The form elements can be located like this:

```
login_form = driver.find_element_by_xpath("/html/body/form[1]")
login_form = driver.find_element_by_xpath("//form[1]")
login_form = driver.find_element_by_xpath("//form[@id='loginForm']")
```

Ejemplo Selenium

Truco: podemos probarlo en la consola web (tocando F12) a ver si funciona y detecta bien los elementos. Ej: escribir en la consola $x(‘//div’) devuelve todos los “divs”.

https://www.w3schools.com/xml/xpath_intro.asp

Otras formas de buscar en una página

CSS Selector: otra forma análoga a XPath. Cada librería tiene una forma de buscar a partir de estos selectores. Misma sintaxis que archivos CSS.

Selector	Example	Example description
.class	.intro	Selects all elements with class="intro"
.class1.class2	.name1.name2	Selects all elements with both *name1* and *name2* set within its class attribute
.class1 .class2	.name1 .name2	Selects all elements with *name2* that is a descendant of an element with *name1*
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element.class	p.intro	Selects all <p> elements with class="intro"
element,element	div, p	Selects all elements and all <p> elements
element element	div p	Selects all <p> elements inside elements

```
<html>
 <body>
  <p class="content">Site content goes here.</p>
</body>
</html>
```

The “p” element can be located like this:

```
content = driver.find_element_by_css_selector('p.content')
```

Ejemplo Selenium

https://www.w3schools.com/cssref/css_selectors.asp

¿Algunos trucos?

- **Delay en el scrapeo**. Acciones hechas a tiempos aleatorios. Incluso simular acciones aleatorias.
  https://www.scraping-bot.io/top-7-web-scraping-tips/

  En general, tener paciencia. No hacer muchos requests en poco tiempo, esperar un poco entre acciones...

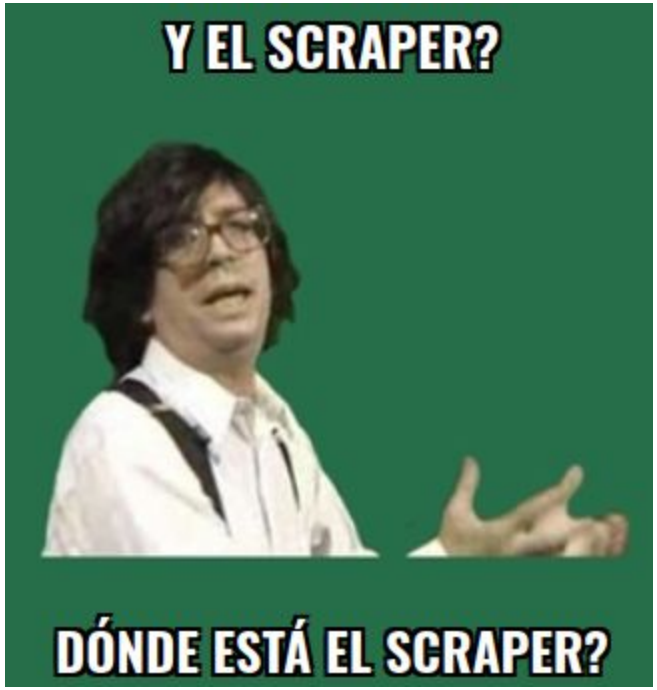
- **Resetear las cookies todo el tiempo.** Toda información que quede guardada en el sitio puede ser mala para scrapear sistemáticamente.
- **Simular que soy un navegador.** Algunas librerías de scrapeo sugieren hacerse pasar como un navegador “User-Agent = Mozilla Firefox”
  https://dev.to/hhsm95/using-user-agent-to-scraping-data-lli

¿Algunos trucos?

- **Navegar sin encabezados (headless).** Básicamente que la ventana de navegación no se abra.
- **Esconder el IP.** De alguna manera mostrarle a la página que la requests no vienen siempre del mismo lado. https://www.kdnuggets.com/2018/02/web-scraping-tutorial-python.html

Otras páginas con tips por allí:

- https://www.scraperapi.com/blog/5-tips-for-web-scraping/
- https://www.codementor.io/blog/python-web-scraping-63l2v9sf2q


Y EL SCRAPER?
DÓNDE ESTÁ EL SCRAPER?