

Web scraping

gencat

Departament d'Economia i Finances

Dra. Laia Subirats - Professora Agregada dels estudis
d'Informàtica, Multimèdia i Telecomunicació
18 setembre 2024, Barcelona

Índex

1. **Introducció:** Objectius del curs, conceptes bàsics de web scraping, antecedents que inclouen exemples de web scraping i opcions de software per a la recopilació i anàlisi de dades. *Pràctica 1+Kahoot 1+Demo+ Pràctica 2*
2. **Ètica:** privadesa i consentiment, estudis de cas i debat.
3. **Eines d'obtenció d'informació de la web en Python:** taules, imatges, etc.
4. **Cas pràctic d'Escales de vaixells al Port de Tarragona.** *Pràctica 3+Pràctica 4*
5. **Eines d'obtenció d'informació de la web en R:** taules, imatges, etc.
6. **Cas pràctic de Taules de l'Impost a la Renda de les Persones Físiques,** publicades per l'Agència Estatal d'Administració Tributària. *Pràctica 5*
7. *Test*

Introducció

Objectius

1. Conèixer el **significat i els potencials beneficis** del web scraping.
2. Ser capaç d'**avaluar la dificultat** de fer web scraping en un lloc web determinat.
3. Ser capaç de fer web scraping simple, utilitzant **Python i R**
4. Conèixer **programaris no-code** que fan web scraping com **Octoparse**.
5. Ser capaç d'**emmagatzemar les dades obtingudes** d'internet en un format interoperable.
6. Ser capaç de **solucionar els principals obstacles** implementats per a evitar el web scraping.
7. Conèixer els **principals aspectes legals** relacionats amb el web scraping.
8. Conèixer diferents **casos d'èxit o usos pràctics** del web scraping.

Perquè fem web scraping?

1. Quan les dades **no estan disponibles o no hi ha API**
2. Quan existeix una API, però aquesta no és **gratuïta**, mentre que l'accés a la pàgina web sí que ho és.
3. Quan l'**API limita** el nombre d'accessos per segon, per dia, etc.
4. Quan l'**API no permet recuperar tota la informació d'interès** que es mostra a la pàgina web.

Cal recordar que la informació pot obtenir-se d'una **API**, d'una **pàgina web**, d'una **aplicació mòbil**, d'uns **feeds o RSS** (rich site summary) o exportació de dades en **arxius** (cal verificar que no estan desactualitzats).

Definició

El web scraping consisteix en la **construcció d'un agent** que permeti descarregar, analitzar i organitzar dades procedents d'internet, de manera **automàtica**. Gràcies a l'ús d'aquesta tècnica, podem dissenyar un **script** que desenvolupi una sèrie de tasques repetitives amb les quals emmagatzemar informació d'interès de manera estructurada i molt més eficientment que si ho féssim manualment, **accelerant el procés** i **evitant errors** produïts en el procés de copiar i enganxar.

Avaluació inicial

1. L'arxiu robots.txt
2. El mapa del lloc web
3. La seva grandària
4. La tecnologia emprada
5. El propietari del lloc web

1. Exemples robots.txt

a) Exclusió de tres directoris

```
User-agent: *  
Disallow: /cgi-bin/  
Disallow: /tmp/  
Disallow: /~joe/
```

b) Exclusió de tots els robots

```
User-agent: *  
Disallow: /
```

c) Permís d'accés complet a tots els robots

```
User-agent: *  
Disallow:
```

d) Permís d'accés a un sol robot

```
User-agent: Google  
Disallow:  
User-agent: *  
Disallow: /
```

e) Exclusió de pàgines concretes

```
User-agent: *  
Disallow: /~joe/junk.html  
Disallow: /~joe/foo.html  
Disallow: /~joe/bar.html
```


2. Mapa lloc web

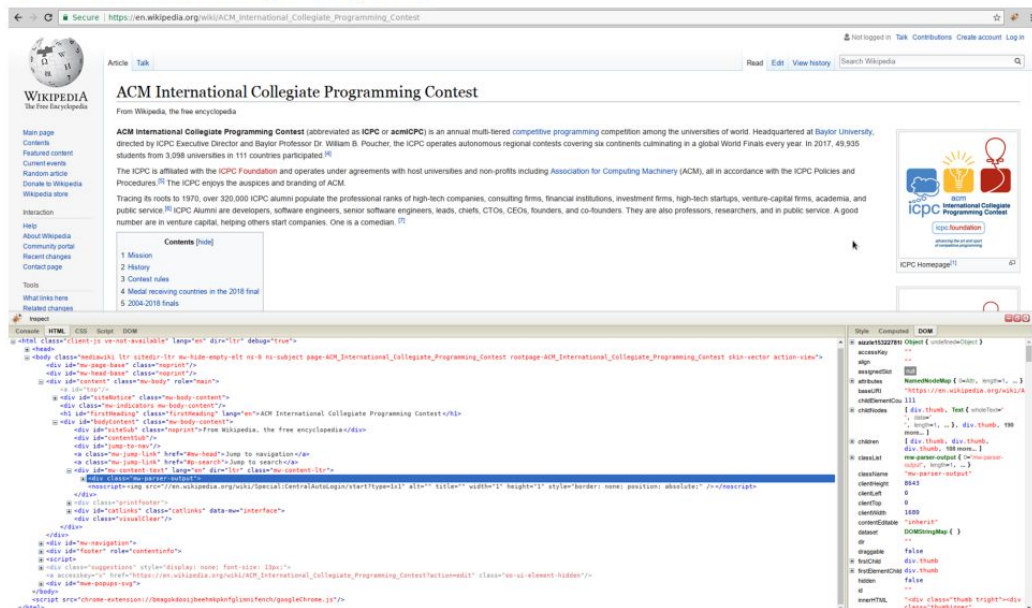
- Etiqueta d'obertura <urlset>, dins de la qual s'ha d'especificar l'espai de nom (estàndard de protocol).
- Cada URL s'ha d'especificar entre les etiquetes <url> i </url>, com una etiqueta XML principal.
- Dins de cada etiqueta primària <url>, una entrada secundària <loc> especifica l'adreça URL.
- Etiqueta de tancament </urlset>.

A continuació es mostra un exemple de *sitemap* en què se'n pot observar l'estructura.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com</loc>
    <lastmod>2005-01-01</lastmod>
    <changefreq>monthly</changefreq>
    <priority>0.8</priority>
  </url>
</urlset>
```

2. Mapa lloc web

Figura 2. Anàlisi mitjançant Firebug Lite de la pàgina web



Font: Viquipèdia

3. Grandària, 4. tecnologia i 5. propietari

Una manera ràpida de verificar la grandària d'una pàgina web es fa amb la **cerca avançada a Google**, amb la paraula clau site, del lloc web d'interès.

La **tecnologia** es pot analitzar després d'instal·lar l'eina **builtwith** de Python per exemple, mitjançant la instrucció `pip3 install builtwith`.

Propietari amb Python amb **whois**:

```
pip3 install python-whois  
import whois
```

```
Web scraping print(whois.whois('https://www.wordpress.com'))
```

Comunicació client i servidor web: Petició HTTP

Connection. Especifica el tipus de connexió amb el servidor HTTP. Normalment, com es mostra en l'exemple, el seu valor és keep-alive.

Accept. Fa referència al tipus de continguts o fitxers acceptats com a resposta; generalment, text/html.

User-agent. Conté informació sobre la petició, això és, sobre el navegador utilitzat, el sistema operatiu, etc.

Accept-encoding. Especifica el tipus de codificacions (encodings) admeses (el servidor pot comprimir la resposta).

Accept-language. El servidor indica els idiomes acceptats.

Cookie. Un altre concepte important són les galetes (cookies), ja que permeten establir preferències que persisteixen al llarg de diferents pàgines web.

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Encoding: gzip, deflate, br

Accept-Language: ca,en-US;q=0.7,en;q=0.3

Cache-Control: max-age=0

Connection: keep-alive

Cookie: WMF-Last-Access-Global=26-Jul-...; WMF-Last-Access=26-Jul-2018

Host: www.wikipedia.org

If-Modified-Since: Mon, 23 Jul 2018 10:07:54 GMT

If-None-Match: W/"12742-571a7d1a715ba"

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 6.1; W...) Gecko/20100101 Firefox/60.0

Comunicació client i servidor web: Resposta HTTP

2XX. Peticions reeixides. Aquesta classe de codi d'estat indica que la petició ha estat rebuda correctament, entesa i acceptada. Un altre exemple d'aquest tipus és el 201 Created, que indica que la petició ha estat completada i ha donat pas a la creació d'un nou recurs.

3XX. Redireccions. En aquest cas, el client ha de fer una acció addicional per a completar la petició. Un exemple d'aquest tipus és el codi 300 Multiple Choices, el qual indica opcions múltiples que el client ha de seleccionar: presentant diferents opcions de format per a la visualització de vídeos, llistant arxius amb diferents extensions, etc.

4XX. Errors del client. La sol·licitud conté una sintaxi incorrecta o no es pot processar. Així, el codi 404 Not Found fa referència a un recurs no oposat i s'utilitza quan el servidor web no és capaç de trobar la pàgina o recurs sol·licitats.

5XX. Errors del servidor, en completar una sol·licitud aparentment vàlida. El codi 500 Internal Server Error és comunament emès per aplicacions incrustades en servidors web que generen contingut dinàmicament; per exemple, aplicacions muntades en Tomcat, quan es troben amb situacions d'error alienes a la naturalesa del servidor web.

age: 77899

backend-timing: D=206 t=1532521483616492

cache-control: s-maxage=86400, must-revalidate, max-age=3600

content-encoding: gzip

content-type: text/html

date: Thu, 26 Jul 2018 10:03:04 GMT

etag: W/"12742-571a7d1a715ba"

last-modified: Mon, 23 Jul 2018 10:07:54 GMT

server: mw1264.eqiad.wmnet

strict-transport-security: max-age=106384710; includeSubDomains; preload

vary: Accept-Encoding

via: 1.1 varnish (Varnish/5.1), 1.1...1), 1.1 varnish (Varnish/5.1)

x-analytics: WMF-Last-Access=26-Jul-2018;WM...ss-Global=26-Jul-2018;https=1

x-cache: cp1065 hit/10, cp3040 hit/1, cp3040 hit/147455

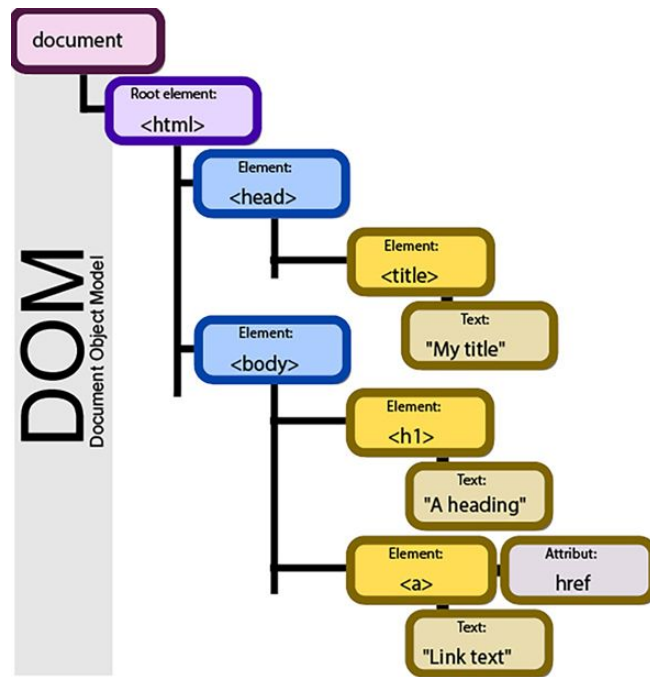
x-cache-status: hit-front

x-client-ip: 84.88.76.3

X-Firefox-Spdy: h2

x-varnish: 548206840 24383145, 516973151 525052522, 403431753 292805460

Document object model (DOM)



XML

<https://es.slideshare.net/slideshow/xml-42789471/42789471>

Pràctica 1: Exploració pràctica d'una pàgina web mitjançant eines de desenvolupament



Principals reptes del web scraping

1. Modificació **user-agent**
2. Gestió d'inici de **sessió i galetes**
3. **Espaiat de peticions HTTP**
4. **Múltiples adreces IP**
5. **Paranys de l'aranya**

Resolució d'obstacles amb web scraping

1. Modificació user-agent



```
import requests

headers = {

    "Accept":
    "text/html,application/xhtml+xml,application/xml;q
    =0.9,image/webp,\

    */*;q=0.8",

    "Accept-Encoding": "gzip, deflate, sdch, br",

    "Accept-Language": "en-US,en;q=0.8",

    "Cache-Control": "no-cache", "dnt": "1", "Pragma":
    "no-cache", "Upgrade-Insecure-Requests": "1",

    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac
    OS X 10_12_3) AppleWebKit/5\ 37.36 (KHTML, like
    Gecko) Chrome/56.0.2924.87 Safari/537.36" } r =
    requests.get("http://www.example.com",
    headers=headers)
```

Resolució d'obstacles amb web scraping

2. Gestió d'inici de **sessió** (logins) i de **galetes** (cookies) de sessió

Per a gestionar el seguiment de galetes configurades pel servidor, la llibreria Requests disposa de l'objecte **session**, que permet agregar aquestes galetes de manera automàtica a les posteriors peticions fetes en el mateix lloc.

```
import requests
session = requests.Session() session.post("http://example.com/login",
data=dict(email="me@domain.com",password="secret_value"))
<!-- peticions realitzades amb session afegeixen automàticament les cookies--> r =
session.get("http://example.com/protected_page")
```

Resolució d'obstacles amb web scraping

3. Espaiat peticions HTTP

`time.sleep(10)`

configuració de temporitzacions (timeouts) i altres excepcions

4. Ús de múltiples adreces IP. Cal dir que l'ús d'aquests servidors sol implicar un cost addicional, al voltant de 40 USD per cada 100 adreces IP, segons Brody (2017)

El codi següent mostra un exemple de configuració de temporitzacions mitjançant la llibreria Requests:

```
try:
    <!-- esperar fins a 10 segons-->
    requests.get("http://example.com", timeout=10)
except requests.exceptions.Timeout:
    pass
```

Un altre problema habitual és el de les connexions caigudes (*broken connections*). Si la connexió o el servidor cauen inesperadament, la petició HTTP feta es trobarà en un estat ambigu que no retornarà una resposta útil. Per a solucionar aquest problema, la llibreria Requests permet afegir una excepció, semblant a la utilitzada en la gestió de temporitzacions, que evitarà que el codi utilitzat deixi de funcionar.

A continuació, se'n mostra un exemple:

```
try:
    requests.get("http://example.com")
except requests.exceptions.RequestException:
    pass
```

Resolució d'obstacles amb web scraping

5. Evitar paranyes d'aranya (spider trap)

Alguns llocs generen dinàmicament el seu contingut de manera que poden tenir un **nombre infinit de pàgines web**. Per exemple, si el lloc conté un calendari amb enllaços al mes i any següents, el mes següent també tindrà un enllaç al mes següent, i així successivament. Atès que el nostre rastrejador seguirà en principi qualsevol enllaç del lloc que no hagi vist abans, es veurà atrapat en una successió infinita d'enllaços, coneguda com a parany d'aranya (spider trap).

Una manera senzilla d'evitar els paranyes d'aranya consisteix a registrar la profunditat de la pàgina, definida com el nombre d'enllaços que s'han seguit per a arribar-hi. Així, en **definir prèviament una profunditat màxima**, quan s'aconsegueix aquest llimitar, el rastrejador deixa d'agregar enllaços a la cua.

```
def link_crawler(..., max_depth=2):  
    max_depth = 2  
    seen = {}  
  
    ...  
    depth = seen[url]  
    if depth != max_depth:  
        for link in links:  
            if link not in seen:  
                seen[link] = depth + 1  
                crawl_queue.append(link)
```

Millors pràctiques

- 1) Abans de fer web scraping, **verifiqueu si ja existeix una API** que permeti recuperar la informació d'interès, sense limitacions de descàrrega.
- 2) **No analitzeu l'HTML manualment.** L'ús de llibreries com BeautifulSoup facilita considerablement la tasca.
- 3) **No satureu de peticions el servidor web**, ja que això augmentarà les probabilitats de ser bloquejats. Així mateix, com que l'administrador de la web (webmaster) es pot adonar que s'està fent una gran quantitat de peticions a la seva pàgina, pot ser interessant contactar-hi per a trobar la manera de treballar conjuntament.
- 4) **Modifiqueu el user agent**, ja que molts llocs revisen aquesta capçalera per a prevenir el web scraping.
- 5) **Reviseu el navegador.** Si desconexem la causa d'un problema, pot ser interessant obrir una nova sessió en el navegador, preferiblement utilitzant els modes d'«incògnit» o de «navegació privada» (private browsing) per a assegurar que el conjunt de galetes es troba buit. Així mateix, es pot utilitzar la instrucció curl per a depurar els casos més complexos.

Millors pràctiques

- 6) **Assumiu que el web scraper deixarà de funcionar.** Les pàgines web són dinàmiques, per la qual cosa pot ser molt útil implementar un codi que de seguida proporcioni advertiments detallats quan algun fragment deixi de funcionar.
- 7) Tingueu en compte la **qualitat i robustesa de les dades** obtingudes. La International Data Management Association del Regne Unit (DAMA UK) defineix la qualitat de les dades a partir de sis dimensions.
- 8) Recordeu els **aspectes legals** associats al web scraping, amb l'objectiu de fer un bon ús de les dades obtingudes.

Bibliografia

1. Acodemy (2015). Learn Web Scraping With Python In A Day: The Ultimate Crash Course to Learning the Basics of Web Scraping With Python In No Time. CreateSpace Independent Publishing Platform.
2. Bosch, O. (2017). An introduction to web scraping, IT and Legal aspects.
3. Brody, H. (2017). The ultimate guide to web scraping. LeanPub.
4. Broucke, S. Vanden; Baesens, B. (2018). Practical Web Scraping for Data Science. Springer.
5. Cavallo, A.; Rigobon, R. (2016). The Billion Prices Project: Using Online Prices for Measurement and Research. Journal of Economic Perspectives (vol. 30, núm. 2, pàg. 151-178).
6. Dale, K. (2016). Data Visualization with Python and JavaScript. O'Reilly.
7. Heydt, M. (2018). Python Web Scraping Cookbook: Over 90 proven recipes to get you scraping with Python, microservices, Docker, and AWS. Packt Publishing.
8. Kouzis-Loukas, D. (2016). Learning Scrapy. Packt Publishing. Lawson, R. (2015). Web Scraping with Python. Packt Publishing Ltd.
9. Mitchell, R. (2015). Web Scraping with Python: Collecting Data from the Modern Web. O'Reilly.
10. Munzert, S.; Rubba, C.; Meißner, P.; Nyhuis, D. (2014). Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining. Hoboken, NJ; Chichester; West Sussex: John Wiley & Sons.
11. Nair, V.G. (2014). Getting started with BeautifulSoup. Packt Publishing Ltd. Open Source Collaborative framework in Python. <https://scrapy.org>
12. Blogs de Towards data science <https://towardsdatascience.com>

Octoparse: <https://www.octoparse.com>

The image shows the Octoparse website homepage. The header includes the Octoparse logo, navigation links (Solutions, Templates, Download, Pricing, Resources), and user options (Login, Start a free trial, EN). The main heading is "Easy Web Scraping for Anyone", followed by the tagline: "Octoparse is your no-coding solution for web scraping to turn pages into structured data within clicks." Below this are two buttons: "Start a free trial" and "Watch a demo". A large, semi-transparent screenshot of the Octoparse software interface is overlaid on the bottom half of the page. The interface shows a dashboard with a sidebar, a main area with a "Hello, Octoparse. Glad to see you here!" message, a search bar, and a "Start" button. It also displays a "What's a custom task?" section, a "What's a test template?" section, and a "Running" status with a progress bar and a "30230" value. A "Cloud Task" button is visible. At the bottom of the interface, there is a cookie consent banner that says "This website uses cookies to ensure you get the best experience." with "Accept" and "Close" buttons.

Refining web scraping job adds: 770 records from job ads with 5240 identified relations to NCS

Refining preferred label to **5-digit ESCO level** in order to be able to integrate with the platform (ESCO code is maintained to 4-digit ESCO level)

Refining job titles:

Change acronyms such as CRM (customer relationship management) and BI (Business intelligence)

Remove words such as advisor, adviser, manager, specialist, consultant and assistant

Remove words such as part-time, hours, multiple positions.

Remove words such as Dublin, Ireland and EMEA

Improving the **modularity** of the library so it can be used easily for scraping several cities



ESCO code	ESCO Preferred Label	accountability	adaptability	coaching	communication
2166	technical director	1	1	1	1
2431	hospitality revenue manager	1	1	1	1
1420	tobacco shop manager	1	0	1	0
2422	housing policy officer	1	1	1	1
3322	commercial sales representative	1	1	1	1
5249	rental service representative in machinery	1	0	1	1
4225	customer service representative	1	0	0	1
2423	recruitment consultant	1	0	1	0
3343	administrative assistant	1	1	0	1
2153	telecommunications engineer	1	0	1	1

Refining web scraping MOOCs: 345 records from courses with 569 identified relations to NCS

The objective was to **reduce false results** not related to the soft skill

To avoid false results, we implemented **thresholds using NCS and buzzwords**

We generated two **scores**:

- Number of times that the NCS and the buzzwords appeared in the *description*, and
- Number of times that NCS and the buzzwords appear in the *title*

Observation: capital letters are converted to non-capital in order to be able to compare strings and that root of words were used (for instance the string “adaptab” is searched in a word instead of “adaptability”)



	accountability	adaptability	coaching	communication	conflict resolution	creativity	critical thinking
https://www.mooc-list.com/course/assessment-learning-co	1	0	0	0	0	0	0
https://www.mooc-list.com/course/setting-expectations-ass	1	0	0	0	0	0	0
https://www.mooc-list.com/course/setting-stage-success-ey	1	0	0	0	0	0	0
https://www.mooc-list.com/course/coaching-practices-cours	1	0	1	0	0	0	0
https://www.mooc-list.com/course/installing-accountability	1	0	0	0	0	0	0
https://www.mooc-list.com/course/becoming-successful-lea	1	0	0	0	0	0	0
https://www.mooc-list.com/course/becoming-part-globalise	0	1	0	1	0	0	0
https://www.mooc-list.com/course/conversations-inspire-cc	0	0	1	0	0	0	0
https://www.mooc-list.com/course/coaching-conversations-	0	0	1	0	0	0	0
https://www.mooc-list.com/course/leadership-coaching-thr	0	0	1	0	0	0	0
https://www.mooc-list.com/course/wellness-coaching-powe	0	0	1	0	0	0	0
https://www.mooc-list.com/course/coaching-digital-learning	0	0	1	0	0	0	0
https://www.mooc-list.com/course/player-centred-coaching	0	0	1	0	0	0	0
https://www.mooc-list.com/course/coaching-knowledge-ers	0	0	1	0	0	0	0
https://www.mooc-list.com/course/team-coaching-futurelei	0	0	1	0	0	0	0
https://www.mooc-list.com/course/coaching-teachers-prom	0	0	1	0	0	0	0
https://www.mooc-list.com/course/managing-coach-courser	0	0	1	0	0	0	0
https://www.mooc-list.com/course/coaching-strategies-sale	0	0	1	0	0	0	0
https://www.mooc-list.com/course/cartography-esri	0	0	1	0	0	0	0
https://www.mooc-list.com/course/swimcoach-university-ic	0	0	1	0	0	0	0
https://www.mooc-list.com/course/guerrilla-literacy-learnei	0	0	1	0	0	0	0
https://www.mooc-list.com/course/introduction-communici	0	0	0	1	0	0	0
https://www.mooc-list.com/course/oral-communication-eng	0	0	0	1	0	0	0
https://www.mooc-list.com/course/understanding-russians-	0	0	0	1	0	0	0
https://www.mooc-list.com/course/communication-major-si	0	0	0	1	0	0	0
https://www.mooc-list.com/course/engaging-persuasive-an	0	0	0	1	0	0	0
https://www.mooc-list.com/course/communication-theory-i	0	0	0	1	0	0	0

4rt Congrés d'Economia i Empresa de Catalunya 2024



4rt Congrés d'Economia i Empresa de Catalunya 2024



Que guanyi el/la millor!

Kahoot!

Demo Octoparse + Pràctica 2 <https://www.octoparse.com>



Ètica

- 1) **Incompliment de termes i condicions.** La majoria de pàgines web publiquen una sèrie de termes i condicions o acords de llicència d'usuari que, sovint, aborden explícitament l'accés al seu contingut mitjançant rastrejadors (scrapers). Amb això, es pretén crear un incompliment de la responsabilitat contractual en establir un contracte entre el propietari del lloc web i l'craper. No obstant això, la publicació d'aquests termes en un lloc web pot no ser suficient per a mostrar que un rastrejador ha incomplert les condicions, si no hi ha una acceptació activa per part d'aquest. Per això, l'ús d'una casella de verificació explícita o enllaç del tipus «Accepto» obliga el web scraper a acceptar activament els termes. De la mateixa manera, en els llocs en els quals és necessari iniciar sessió, la creació d'un compte sol incloure un acord explícit dels termes i condicions.
- 2) **Infracció de drets d'autor o marca registrada.** A Estats Units, la legislació legal de l'ús raonable (fair use) permet l'ús limitat de material protegit per drets d'autor sota certes condicions, sense el permís explícit del titular dels drets esmentats. Així, els usos amb aquestes finalitats com la paròdia, la crítica, els comentaris o la recerca acadèmica es consideren ús legítim. No obstant això, la majoria d'usos comercials es consideren una infracció.
- 3) **Llei de frau i abús informàtic.** Existeixen diverses lleis federals i estatals que prohibeixen l'accés a l'ordinador d'una altra persona. En resum, aquestes lleis afirmen que «qui accedeix intencionadament a un ordinador sense autorització [...] i com a resultat de tal conducta causa dany» està incomplint la llei.

4) **Violació de domicili.** Aquest terme es refereix a un delictes civil en el qual una entitat interfereix en la propietat personal d'un individu i hi causa pèrdua de valor o dany. El 1999, aquesta llei es va aplicar en un cas judicial entre Ebay i Bidder's Edge.

5) **Protocol d'exclusió de robots.** Es tracta d'un estàndard industrial que permet a una pàgina web disposar d'un arxiu robots.txt on es proporcionen instruccions sobre qui pot accedir al lloc i a quines pàgines es pot accedir. Encara que aquest arxiu té un valor legal limitat, abans de rastrejar qualsevol pàgina web és aconsellable verificar si el propietari hi està d'acord, per a evitar futurs problemes legals.

6) **Llei de drets d'autor del mil·lenni digital i llei CAN-SPAM.** Aquestes lleis també han estat utilitzades en alguns casos judicials relacionats amb web scraping.

Recomanacions

- **Obtenir permís per escrit.** La millor pràctica per a evitar problemes legals consisteix a obtenir permís escrit del propietari d'un lloc web, en el qual s'especifiqui fins a quin punt se'n pot extreure informació.
- **Verificar les condicions d'ús.** Aquestes inclouran sovint disposicions explícites contra l'extracció automàtica de dades. Així mateix, les API d'un lloc web solen anar acompanyades dels seus propis termes d'ús, per la qual cosa també és aconsellable revisar aquests casos.
- **Rastrear només informació pública.** Generalment, quan un lloc web exposa informació públicament, sense ser necessari acceptar una sèrie de termes i condicions, es considera que l'ús moderat de web scraping és adequat. Aquells llocs en els quals és necessari iniciar sessió per a accedir a la informació d'interès, per contra, són més delicats des del punt de vista legal.
- **No causar dany.** No sobrecarregar el servidor amb gaires peticions, mantenir-se allunyat dels equips protegits i no intentar accedir als servidors als quals no es té accés.
- **Utilitzar la informació extreta de manera justa.** No utilitzar amb finalitats comercials les dades protegides per drets d'autor

Python

Requests

Purpose: Simplifies sending HTTP requests and handling responses.

Usage: Fetches web pages to be parsed.

```
import requests
```

```
response = requests.get('https://example.com')  
html = response.text
```

BeautifulSoup (bs4)

Purpose: Parses HTML and XML documents to extract data.

Usage: Works well with static content and is often used in conjunction with `requests`.

```
from bs4 import BeautifulSoup
```

```
soup = BeautifulSoup(html, 'html.parser')  
title = soup.title.string
```

lxml

Purpose: Provides high-performance parsing of HTML and XML documents.

Usage: Faster and more powerful than BeautifulSoup for large documents.

```
from lxml import etree
```

```
tree = etree.HTML(html)  
title = tree.xpath('//title/text()')
```

Scrapy

Purpose: A powerful and versatile web scraping framework.

Usage: Handles large-scale scraping projects, supports data extraction, web crawling, and more.

```
import scrapy
```

```
class MySpider(scrapy.Spider):  
    name = 'myspider'  
    start_urls = ['https://example.com']
```

```
def parse(self, response):  
    yield {'title': response.css('title::text').get()}
```


Selenium

Purpose: Automates browser actions and can handle dynamic content.

Usage: Used for scraping sites that rely on JavaScript to load content.

```
from selenium import webdriver
```

```
driver = webdriver.Chrome()  
driver.get('https://example.com')  
title = driver.title  
driver.quit()
```

Cas pràctic port de Tarragona

Port de Tarragona

<https://www.porttarragona.cat/APTEstadisticas>



Port de Tarragona

<http://practicosdetarragona.org/o/informacion-a-buques>





```
#!/pip install requests beautifulsoup4
import requests
from bs4 import BeautifulSoup
import pandas as pd

# URL of the webpage to scrape
url = "http://practicosdetarragona.org/o/informacion-a-buques"

# Send a GET request to the webpage
response = requests.get(url)

# Check if the request was successful
if response.status_code == 200:
    # Parse the HTML content using BeautifulSoup
    soup = BeautifulSoup(response.content, 'html.parser')

    # Find all tables on the page
    tables = soup.find_all('table')
```

```
# List to store data for DataFrame
all_data = []

for table in tables:
    # Extract table headers
    headers = [th.get_text(strip=True) for th in table.find_all('th')]

    # Extract table rows
    rows = table.find_all('tr')
    for row in rows:
        cells = row.find_all('td')
        if cells:
            cell_texts = [cell.get_text(strip=True) for cell in cells]
            all_data.append(cell_texts)

# Create a DataFrame from the collected data
if all_data:
    df = pd.DataFrame(all_data)
    # Display the DataFrame
    print(df)

    # Optionally, save the DataFrame to a CSV file
    df.to_csv('scraped_data.csv', index=False)
else:
    print("No data found in the tables.")
else:
    print(f"Failed to retrieve the webpage. Status code: {response.status_code}")
```

Exercici amb Selenium i Duckduckgo

<https://sites.google.com/chromium.org/driver/downloads?authuser=0>

Troba els resultats de web scraping laia subirats a

https://duckduckgo.com/?t=h_&q=web+scraping+laia+subirats&ia=web

<https://github.com/laiasubirats/cibersecurity>

Qüestions?

Índex

1. **Revisió apunts robots.txt i XML** (30 min)
2. **Eines d'obtenció d'informació de la web en R: taules, imatges, etc.** (30 min)
3. **Cas pràctic de Taules de l'Impost a la Renda de les Persones Físiques, publicades per l'Agència Estatal d'Administració Tributària.** *Pràctica 5* (1h)
4. **Projecte** (1h)
5. **Test** (30 min)
6. **I després del web scraping què?** (1h)

R

Rvest

The `rvest` package, part of the tidyverse, is widely used for web scraping. It provides functions to read HTML content and extract data from web pages easily.

Key functions include `read_html()`, `html_nodes()`, and `html_text()`.

```
library(rvest)
url <- "https://example.com"
page <- read_html(url)
titles <- page %>% html_nodes("h1") %>%
  html_text()
```

httr

The `httr` package provides functions for working with HTTP, making it useful for handling more complex web requests and APIs.

Key functions include `GET()`, `POST()`, and `content()`.

```
library(httr)
url <- "https://api.example.com/data"
response <- GET(url)
data <- content(response, "text")
```

XML2

The `xml2` package is useful for parsing and working with XML and HTML. It is often used in conjunction with `rvest`.

Key functions include `read_html()`, `xml_find_all()`, and `xml_text()`.

```
library(xml2)
page <- read_html("https://example.com")
titles <- xml_find_all(page, "//h1") %>%
  xml_text()
```

Rcurl

The `Rcurl` package provides a comprehensive interface to the libcurl library, allowing for more control over web requests. Key functions include `getURL()`, `postForm()`, and `getcurlHandle()`.

```
library(RCurl)
url <- "https://example.com"
page <- getURL(url)
```

jsonlite

The `jsonlite` package is useful for parsing and creating JSON data, often necessary when working with APIs. Key functions include `fromJSON()` and `toJSON()`.

```
library(jsonlite)
url <- "https://api.example.com/data"
json_data <- fromJSON(url)
```


RSelenium

<https://cran.r-project.org/web/packages/RSelenium/vignettes/basics.html>

Cas pràctic impost de la renda

Impost a la Renda de les Persones Físiques, publicades per l'Agència Estatal d'Administració Tributària en R

Basat en un desenvolupament publicat pel 'Perfil de la ciutat'

- <https://www.perfilciutat.net/articles/fent-web-scraping-amb-r-a-la-agencia-a-tributaria>
- <https://www.dropbox.com/scl/fo/rzpxhbeaepr0nssg2mxoq/ALMso-Uh8g97zU4QQWQRLzo?rlkey=tf3o0xsaucwwiob6cxmqsdjzv&e=1&dl=0>
- <https://github.com/fawno/AEAT>
- <https://atc.gencat.cat/es/tributs/irpf/index.html>
- Cal instal·lar R i Rstudio

<https://atc.gencat.cat/es/tributs/irpf/index.html>



```
# Install and load the required packages
```

```
install.packages("rvest")
```

```
install.packages("dplyr")
```

```
library(rvest)
```

```
library(dplyr)
```

```
# Define the URL of the page you want to scrape
```

```
url <- "https://atc.gencat.cat/es/tributs/irpf/index.html"
```

```
# Read the HTML content of the page
```

```
page <- read_html(url)
```

```
# Extract the tables
```

```
tables <- page %>%
```

```
  html_nodes("table") %>%
```

```
  html_table(fill = TRUE)
```

```
# Print the number of tables
```

```
extracted
```

```
print(length(tables))
```

```
# Print the first table to inspect it
```

```
print(tables[[1]])
```

```
# Optionally convert the first table  
to a data frame
```

```
table1 <- as.data.frame(tables[[1]])
```

```
# Print the data frame
```

```
print(table1)
```

Amb l'ajuda de ChatGPT!! Algunes consideracions que dona

1. The `fill = TRUE` parameter in `html_table` is used to ensure that the table is filled correctly even if some rows have missing columns.
2. You can access other tables by indexing into the tables list, e.g., `tables[[2]]` for the second table, and so on.

Always check the webpage structure using your web browser's developer tools (F12) to ensure that the CSS selectors used in the script match the structure of the webpage.

Imprimir següent taula

```
# Print the first table to inspect it  
print(tables[[2]])
```

```
# Optionally convert the first table to a data frame  
table2 <- as.data.frame(tables[[2]])
```

```
# Print the data frame  
print(table2)
```

Projecte



Test



Aplicacions de web scraping

1. Price Monitoring and Comparison
2. Market Research
3. Lead Generation
4. SEO and Content Aggregation
5. Financial Data Gathering
6. News Monitoring
7. **Job Market Analysis**
8. Real Estate and Property Listings
9. Academic Research
10. **Social Media Monitoring**
11. Travel and Hospitality
12. **Sentiment Analysis**
13. Public Data and Government Reports
14. Competitor Analysis

I després del web scraping què? → Anàlisi de dades

Processament del llenguatge natural

Análisis de la depresión y la ansiedad causadas por un aborto usando datos de Twitter

<https://openaccess.uoc.edu/handle/10609/138932>

Mining Facebook Data of People with Rare Diseases: A Content-Based and Temporal Analysis

<https://www.mdpi.com/1660-4601/15/9/1877>

Anàlisi de les dades del Dia Mundial de les Malalties Minoritàries a Twitter

<https://openaccess.uoc.edu/handle/10609/139566>

Análisis de datos de Twitter en el Día Mundial de la Salud

<https://openaccess.uoc.edu/handle/10609/124966>

I després del web scraping què? → Anàlisi de dades

Classificació d'ossos llocs arqueològics




<https://www.mdpi.com/2076-3417/9/22/4960>

Predicció rendiment acadèmic

<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0282306>

Qüestions?

FORMAR TRANS- FORMAR

 UOC.universitat
 @UOCuniversitat
 UOCuniversitat



Universitat Oberta
de Catalunya



 gencat

Departament d'Economia i Finances