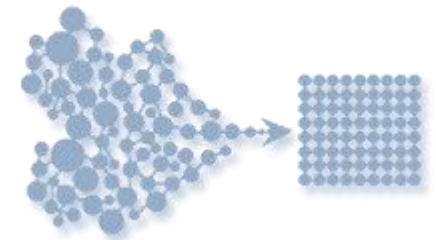


# TALLER DE WEB SCRAPING



# Web Scraping

- **Web scraping** es un proceso informático que sirve para **extraer de manera automatizada información de páginas web**.
- El web scraping está muy relacionado con la indexación de la web, y es una técnica universal adoptada por la mayoría de los motores de búsqueda.
- El web scraping se enfoca sobre todo en la **transformación de datos sin estructura en la web** (como el formato HTML) **en datos estructurados** que pueden ser almacenados y analizados en un repositorio central.
- El web scraping se utiliza de forma profusa hoy en día. Algunos usos del web scraping son la comparación de precios en tiempo real, la monitorización de datos relacionados con el clima, la detección de cambios en sitios webs o la integración de datos en sitios webs.



# ¿Es legal hacer web scraping?

Se ha discutido mucho sobre la legalidad de extraer de forma automática contenido de terceros.

Lo primero que hay que recordar es que **el web scraping es sólo una herramienta**, y que por tanto la discusión no debe centrarse en la herramienta sino el uso que se hace posteriormente de los datos extraídos.



En 2017, el tema volvió a retomar fuerza como consecuencia de la sentencia en EE.UU., que obligó a **LinkedIN** a levantar el bloqueo que impedía a la empresa **hiQ Labs** extraer datos de los perfiles públicos de esta red social de profesionales.

En España, dichas cuestiones han sido analizadas por nuestros tribunales a raíz de los procedimientos civiles interpuestos por la línea aérea **Ryanair** frente a tres agencias de viaje en línea que utilizaban dichas técnicas para ofrecer comparativas de precios (y la posibilidad de adquirir directamente los billetes) a los usuarios de sus sitios web a cambio de una comisión.

- 9 de octubre de 2012 — Ryanair contra Atrápalo
- 30 de octubre de 2012 — Ryanair contra eDreams
- 7 de mayo de 2014 — Ryanair contra LastMinute

Ryanair basaba sus acciones principalmente en tres fundamentos, que sin embargo **fueron sucesivamente rechazados por el Tribunal Supremo**.

## 1. Vulneración de los términos y condiciones de uso del sitio web

- Ryanair alegaba que los términos y condiciones de su web prohibían expresamente el uso de técnicas de web scraping. Sin embargo, **el Tribunal Supremo rechazó este argumento al considerar que las agencias de viaje no habían aceptado los términos y condiciones de forma expresa.**
- Por tanto, navegar por un sitio web no supone la aceptación expresa de los términos y condiciones del sitio web.





## 2. Vulneración de los derechos sobre base de datos

- Los datos que se muestran en la web no es lo mismo que la Base de Datos.
- Ryanair alegaba que su sitio web contiene una base de datos (conforme a la definición de la Ley de Propiedad Intelectual y de la Directiva 96/9) protegida por derechos de autor.
- El Tribunal Supremo también desestimó este argumento, ya que:
  1. No existe base de datos en el sentido en que se define en la Ley de Propiedad Intelectual y de la Directiva 96/9.
  2. En cualquier caso, la base de datos no estaría protegida por derechos de autor ya que no se cumple el requisito de la originalidad.
  3. La base de datos tampoco estaría protegida por derecho sui generis puesto que Ryanair no había probado dicha inversión sustancial en la obtención, la verificación y la presentación de los contenidos de la base de datos, sino sólo en la mera creación de la información, que no está protegida por este derecho.

### 3. Competencia desleal basada en la extracción ilícita de información de la base de datos.

- No hay competencia desleal si los datos extraídos no se usan para replicar el mismo negocio.
- Ryanair alegaba que las agencias de viaje cometían actos de competencia desleal por aprovechamiento ilícito de los esfuerzos competitivos de Ryanair.
- El Tribunal Supremo también desestimó este motivo, insistiendo en la ausencia de base de datos y que, por tanto, la "extracción" para competir no puede producirse.



A la vista de estas tres sentencias, el uso de técnicas de web scraping está sometido al cumplimiento de tres requisitos:

1. Ausencia de aceptación expresa de los términos y condiciones que prohíban el web scraping.
2. Ausencia de una base de datos en el sentido de la Ley de Propiedad Intelectual y la Directiva 96/9.
3. Evitar la extracción, definida como la transferencia permanente o temporal de toda o de una parte sustancial de la base de datos a otro soporte distinto de la base de datos original.

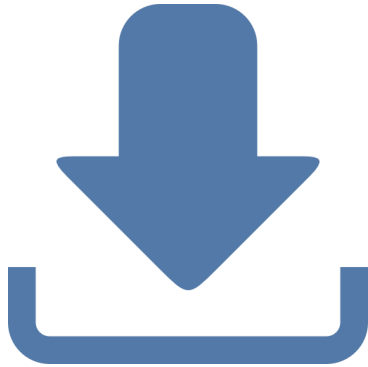




Independientemente de las consideraciones legales, existen una serie de normas de etiqueta que es conveniente cumplir cuando creamos un scraper:

- Revisar el archivo robots.txt.
- Revisar el apartado “términos y condiciones”.
- Identificar de forma correcta a nuestro scraper (user-agent).
- Limitar el número de peticiones por segundo.





**DESCARGAR**



**PARSEAR**



**GUARDAR**



# Descargar URLs con Python

→ urllib

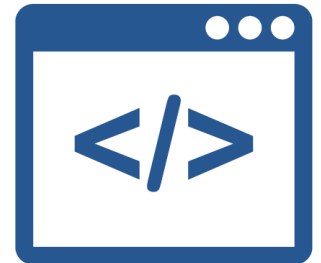
→ urllib2

→ requests

```
import requests  
page = requests.get('https://www.medialab-prado.es')
```

Estrategias para obtener los datos que queremos:

- CSS Selectors
- Xpath Selectors
- Expresiones regulares



# CSS Selectors

En CSS, los selectors son patrones que se usan para seleccionar los elementos de un document HTML a los que queremos aplicar un estilo.

```
p {  
    text-align: center;  
    color: red;  
}
```

```
p.italic {  
    font-style: italic;  
}
```

Selector	Ejemplo	Descripción
.class	.intro	Selecciona los elementos con class="intro"
#id	#firstname	Selecciona el element con id="firstname"
*	*	Selecciona todos los elementos
element	p	Selecciona todos los elementos <p> de la página
element, element	div, p	Selecciona todos los elementos <div> y todos los <p>
element element	div p	Selecciona todos los elementos <p> que están dentro de elementos <div>
element > element	div > p	Selecciona todos los elementos <p> cuyo padre sea un elemento <div>
element+element	div + p	Selecciona todos los elementos <p> que están colocados después de <div>
[attribute]	[target]	Selecciona todos los elementos que tengan un atributo 'target'
[attribute=value]	[target=_blank]	Selecciona todos los elementos con "target=_blank"

# CSS Selectors

- Abrir Google Chrome
- Abrir la dirección <http://www.marca.com/futbol/primer clasificacion.html>
- Abrir la consola de desarrollo:



Pulsar F12



Cmd + Opción + I

- En la consola probar lo siguiente:
  - `document.querySelector('.equipo')`
  - `document.querySelector('.equipo').innerText`
  - `document.querySelectorAll('.equipo')`
- ¿Cómo extraerías el resto de datos?
- <http://flukeout.github.io>

# BeautifulSoup

BeautifulSoup es una librería de Python para parsear documentos html usando selectores.

```
import requests
from bs4 import BeautifulSoup

page = requests.get(http://www.marca.com/futbol/primer clasificacion.html)
soup = BeautifulSoup(page)
fila = soup.select("tr")
equipos = soup.findAll("td", { "class" : "equipo" })
```



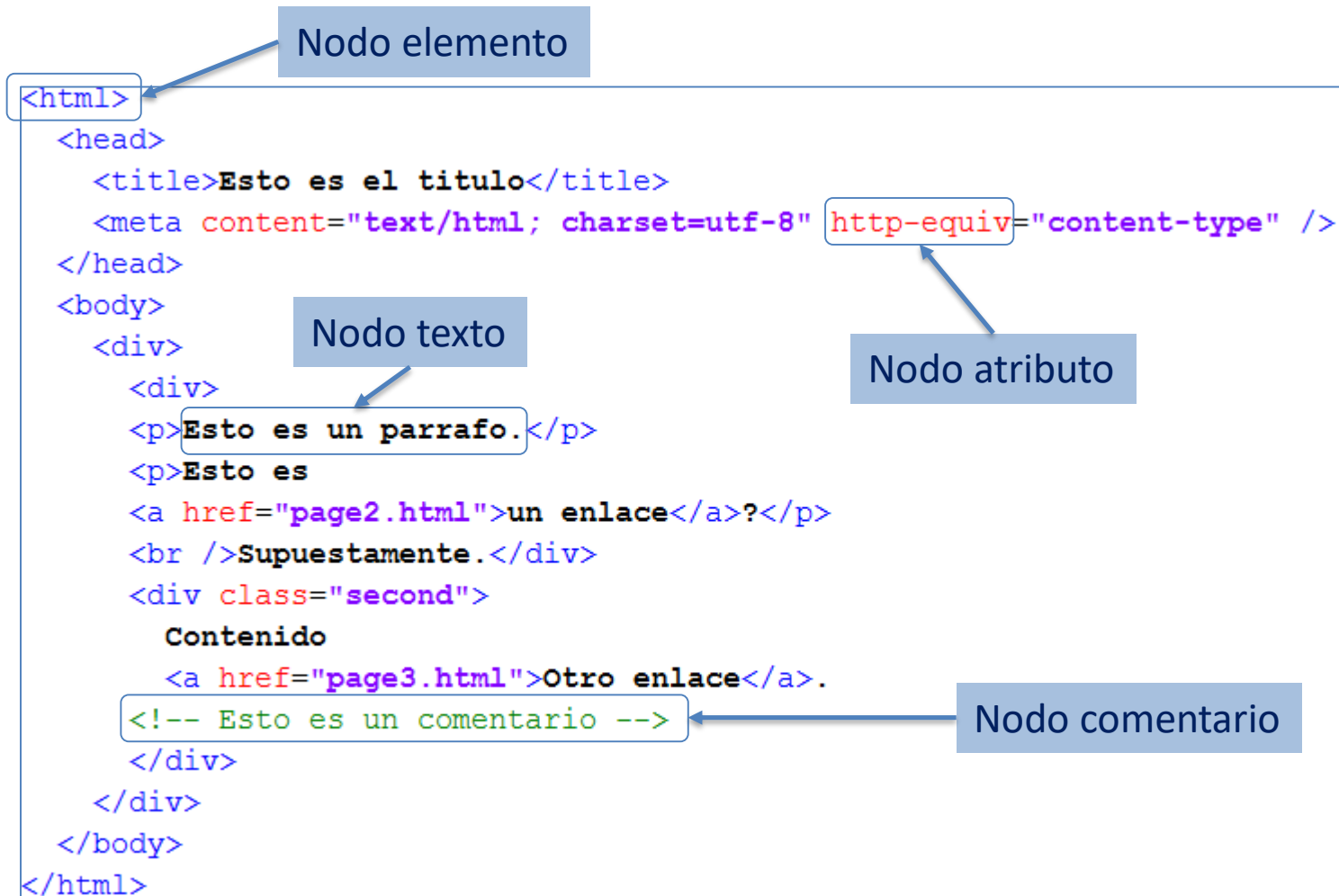
- XPath (XML Path Language) es un lenguaje que permite construir expresiones que recorren y procesan un documento XML.
- Permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML.
- Teniendo en cuenta que HTML es un tipo de XML, podemos usar Xpath para parsear páginas html.
- Cuando hablamos del modelo de datos XPath nos referimos a un árbol compuesto por nodos que pueden ser de cuatro tipos\*:
  - Nodos **ELEMENTO** (<p>...</p> )
  - Nodos **ATRIBUTO** (href="page.html" )
  - Nodos de **TEXTO** ("Some Title" )
  - Nodos **COMENTARIO** (<!-- a comment --> )

\* Hay otros tipos de nodos que nosotros no vamos a ver

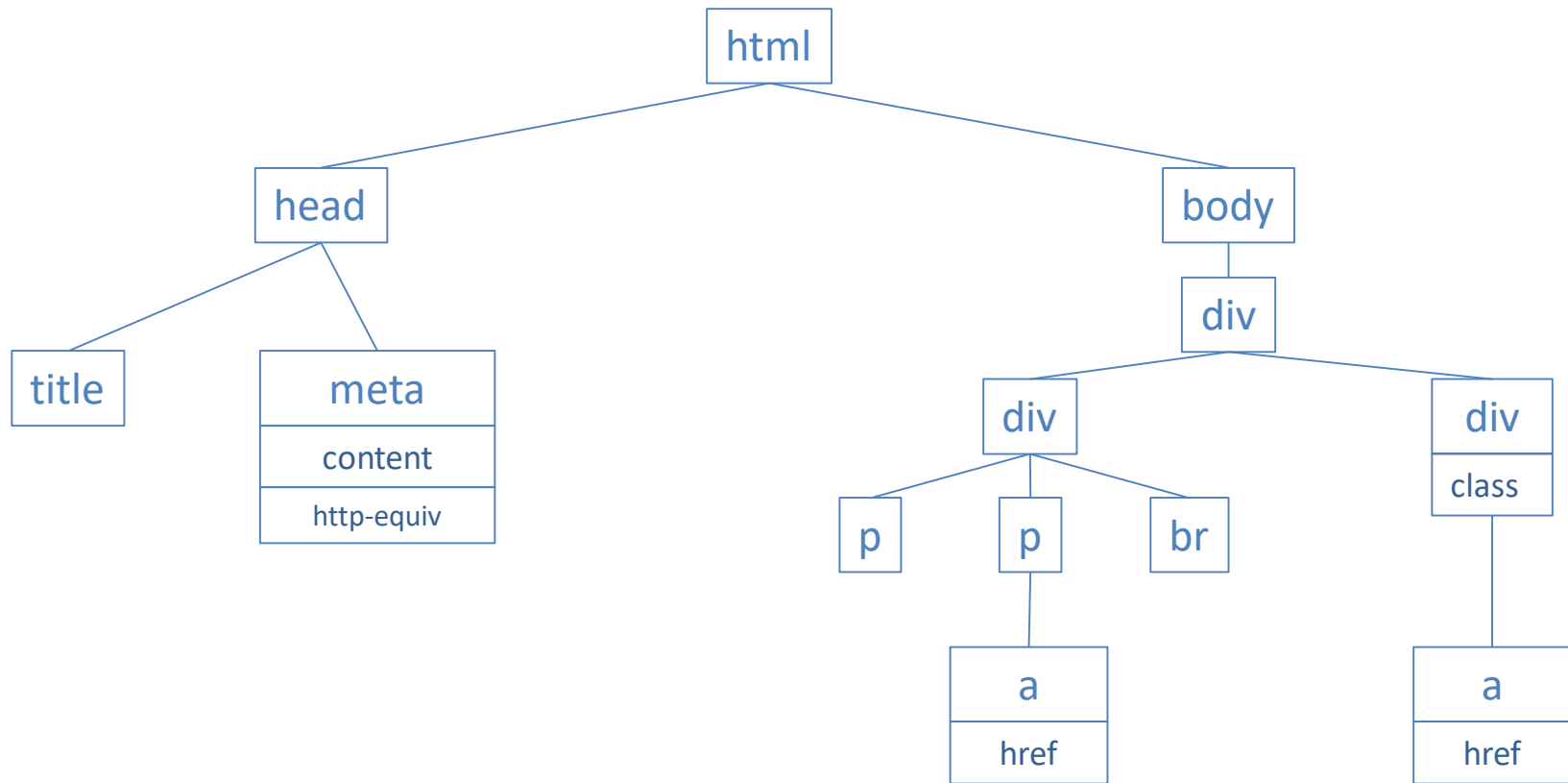
# Xpath Selectors

```
<html>
  <head>
    <title>Esto es el titulo</title>
    <meta content="text/html; charset=utf-8" http-equiv="content-type" />
  </head>
  <body>
    <div>
      <div>
        <p>Esto es un parrafo.</p>
        <p>Esto es
        <a href="page2.html">un enlace</a>?</p>
        <br />Supuestamente.</div>
        <div class="second">
          Contenido
          <a href="page3.html">Otro enlace</a>.
          <!-- Esto es un comentario -->
        </div>
      </div>
    </body>
  </html>
```

# Xpath Selectors



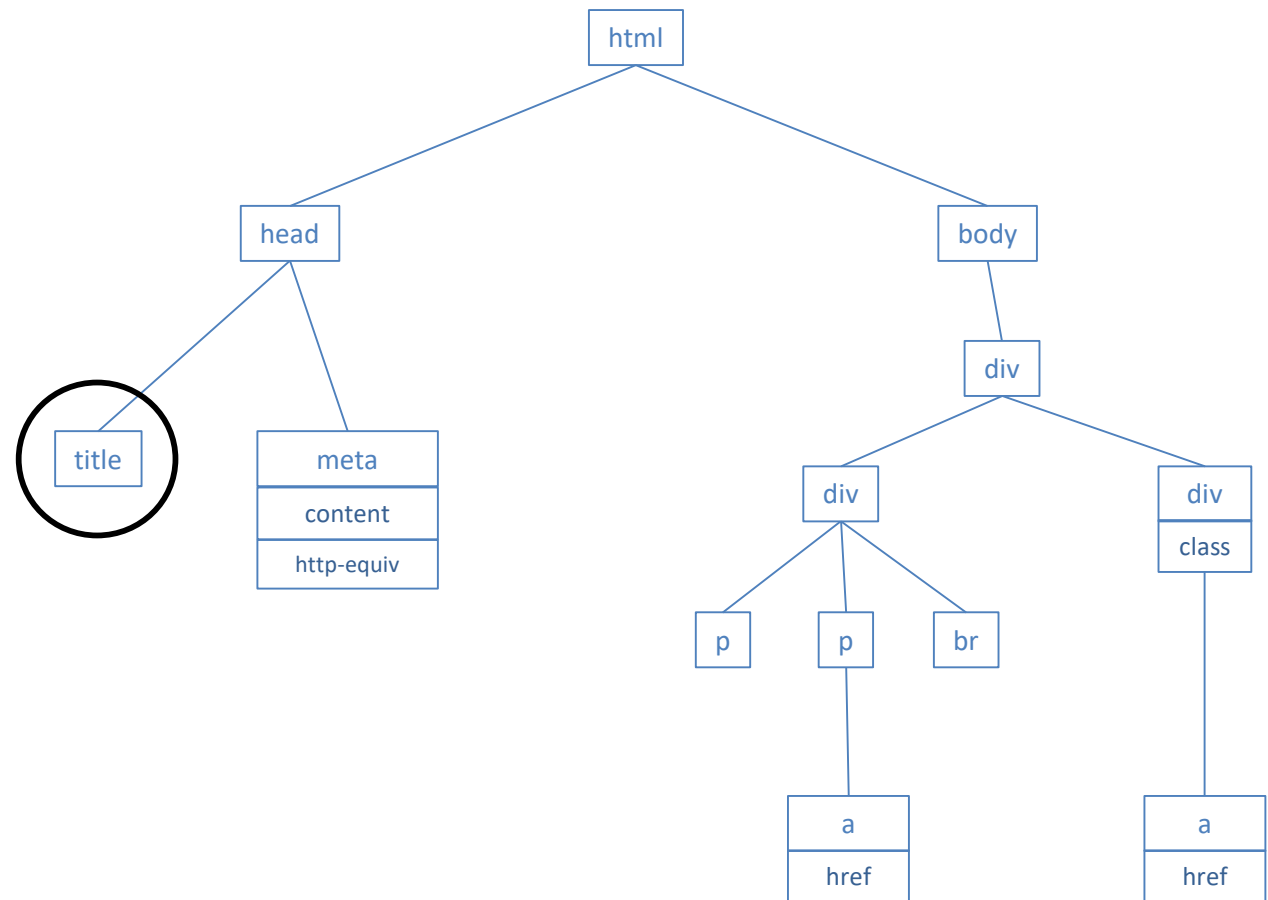
# Xpath Selectors



# Xpath Selectors

Las expresiones Xpath nos permiten movernos por la estructura de árbol usando “/” y “//”.

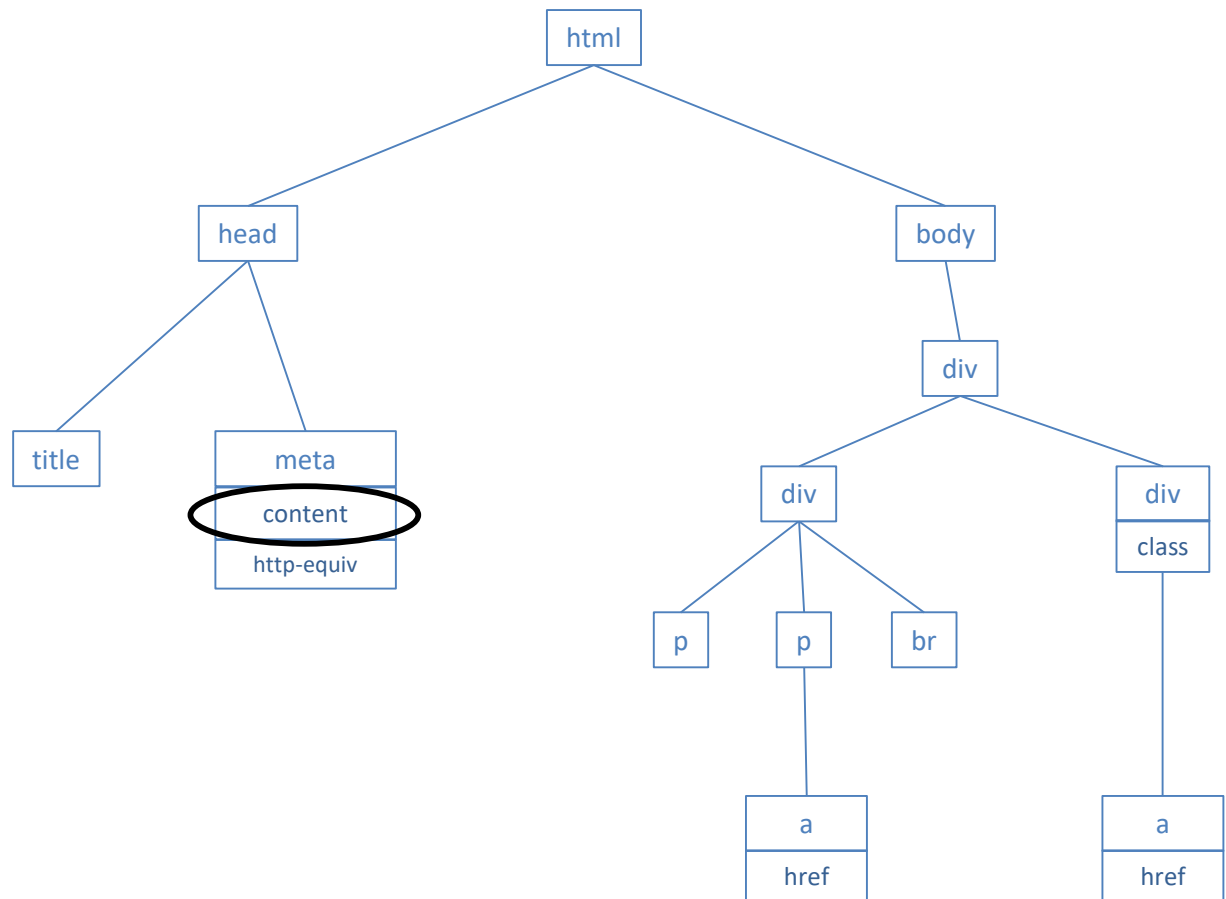
/html/head/title



# Xpath Selectors

Las expresiones Xpath nos permiten movernos por la estructura de árbol usando “/” y “//”.

`//meta/@content`



# Xpath Selectors

## Localización

- “/” → descendiente directo (justo un nivel por debajo).
- “//” → descendiente (cualquier nivel por debajo).
- “paso1/paso2/paso3” → ruta relativa.
- “/paso1/paso2/paso3” → ruta absoluta (a partir del nodo raíz).
- Intentar usar rutas relativas siempre que sea posible.

# Xpath Selectors

## Atributos

- “@” → selecciona nodo atributo.
- /html/head/meta/@content
- //meta/@content
- //a/@href

## Filtros

- //div /div[@class=“second”]
- //div/div[1]

## Nodos texto

- //div/a/text()



# Xpath Selectors

## Ejemplo en python

```
from lxml import html
import requests

page = requests.get('http://www.imdb.com/chart/top')
tree = html.fromstring(page.content)

pelis = tree.xpath('//td[@class="titleColumn"]/a/text()')
```

# Expresiones Regulares

Las Expresiones Regulares son secuencias de caracteres que forma un patrón de búsqueda, utilizadas principalmente para la búsqueda de patrones de cadenas de caracteres en textos.

[abc]	Cualquier carácter de los que están entre corchetes
[^abc]	Cualquier carácter que NO sea de los que están entre corchetes
[0-9]	Cualquier carácter en el rango definido
[^0-9]	Cualquier carácter que NO esté en el rango
(x y)	Alternativas (uno u otro)

.	Cualquier carácter (menos salto de línea)
\w	Carácter letra ~ [a-zA-Z]
\W	Carácter NO letra
\d	Carácter numérico ~ [0-9]
\D	Carácter NO numérico
\s	Carácter separador (espacio, tabulador,...)
\S	Carácter no separador

n+	Al menos una aparición de n
n*	Cero o más ocurrencias de n (voraz)
n?	Cero o más ocurrencias de n (no voraz)
^n	Cadena que comienza por n
n\$	Cadena que termina en n

( )	Grupo. Captura una subcadena
-----	------------------------------

# Expresiones Regulares

Librería para usar expresiones regulares

```
import requests
import re

url = "https://app.uem.es/Docente/servlet/uem.profesores.notas.ServletPool?notasAsignaturas=v"
req = requests.get(url)
htmldata = req2.text

m = re.search('var titulos=new Array\((.*?)\);', htmldata, re.UNICODE|re.MULTILINE|re.DOTALL)
listadoTitulos = m.group(1)
#print listaAsignaturas
titulos = re.findall('(.*?)', listadoTitulos, re.UNICODE|re.MULTILINE|re.DOTALL)
```

Expresión regular

Modificadores / Opciones

Un bot (aféresis de “robot”) es un programa informático que inspecciona páginas de la World Wide Web de forma metódica y automatizada, imitando en muchas ocasiones el comportamiento humano.



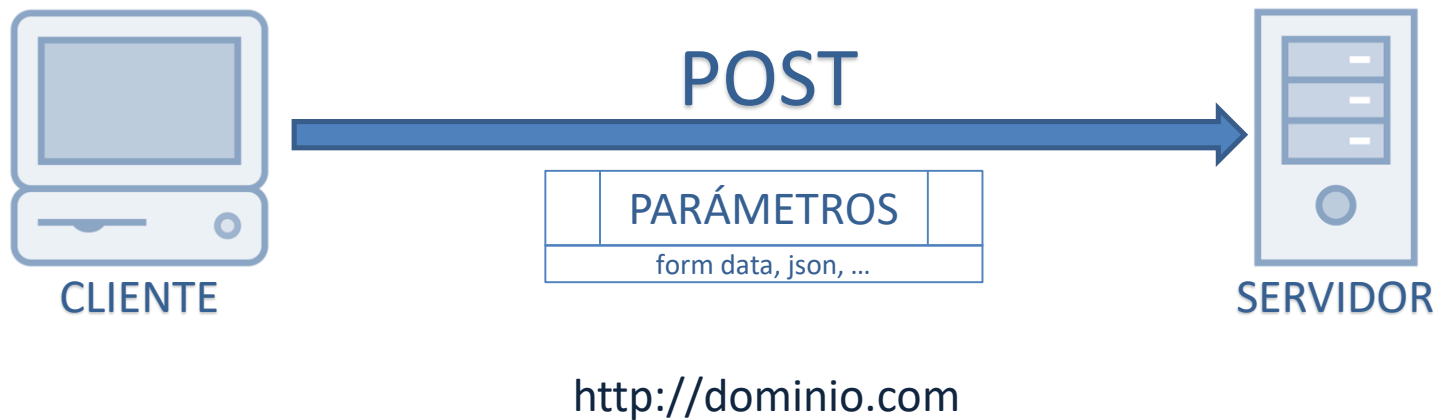
Web scraper avanzado con las siguientes capacidades:

- Scraping de múltiples páginas enlazadas.
- Uso de parámetros.
- Interrogar formularios.



- Scraping de múltiples páginas enlazadas.
- Uso de parámetros.
- Interrogar formularios.

# HTTP : GET Vs POST



# HTTP : GET Vs. POST

- GET es el método en el que los parámetros van incluidos en la URL de la página web.

`http://dominio.com?cp=28670&fecha=22/05/2017`

- POST es el método que habitualmente se utiliza para enviar parámetros. Los datos van en el cuerpo de la petición http en vez de en a la URL de la página solicitada.

`http://dominio.com`

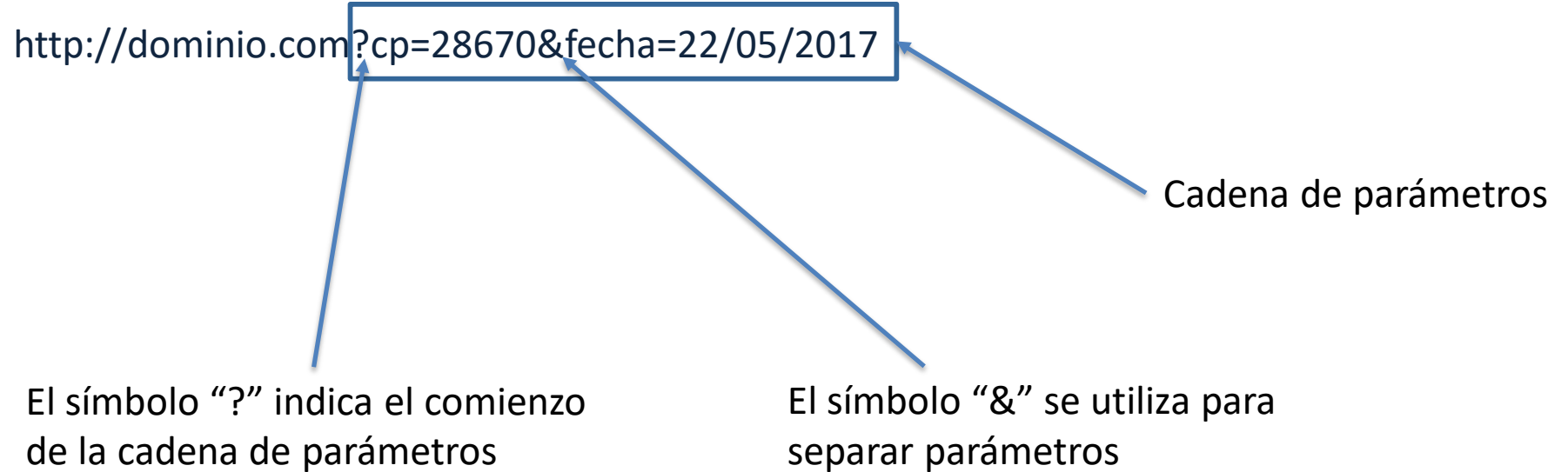
`{ cp : 28670, fecha: 22/05/2017 }`

- En muchas ocasiones, GET y POST son intercambiables: aunque una web tenga un formulario que envía los datos usando POST, si probamos a enviarlos mediante GET, el servidor devuelve la misma respuesta.



# HTTP : GET

http://dominio.com?cp=28670&fecha=22/05/2017



The diagram shows the URL 'http://dominio.com?cp=28670&fecha=22/05/2017'. A blue box highlights the query string portion '?cp=28670&fecha=22/05/2017'. Three blue arrows point from text labels to specific parts of the URL: one from 'El símbolo “?” indica el comienzo de la cadena de parámetros' to the question mark, one from 'El símbolo “&” se utiliza para separar parámetros' to the ampersand, and one from 'Cadena de parámetros' to the entire boxed query string.

Cadena de parámetros

El símbolo “?” indica el comienzo de la cadena de parámetros

El símbolo “&” se utiliza para separar parámetros

# HTTP : GET

- Si los parámetros contienen caracteres distintos de letras y números, hay que codificarlos antes de enviarlos para evitar problemas:

- URLENCODE

- nombre=Enrique Puertas&cp=28670&fecha=22/05/2017

- nombre=Enrique%20Puertas&cp=28670&fecha=22%2F05%2F2017

- BASE64

- comentario=El hotel era demasiado pequeño y el desayuno no tenía zumo de naranja

- comentario=RWwgaG90ZWwgZXJhIGRlbWFzaWFkbyBwZXF1ZcOxbyB5IGVsIGRlc2F5dW5vIG5vIHRIbsOtYSB6dW1vIGRlIG5hcmFuamENCg==

# HTTP : POST

“action” define la dirección a la que se envían los datos

```
<form action="http://www.aprendebigdata.com" method="post">
```

```
<p>
```

```
<label for="nombre">Nombre: </label>
```

```
<input type="text" name="nombre" id="nombre">
```

Parámetro de tipo texto (“nombre=...”)

```
<br/> <br/>
```

```
<label for="apellido">Apellido: </label>
```

```
<input type="text" name="apellido" id="apellido">
```

```
<br/> <br/>
```

```
<label for="email">Email: </label>
```

```
<input type="text" name="email" id="email">
```

```
<br/> <br/>
```

```
<input type="radio" name="sexo" id="varon" value="Varón">
```

```
<label for="varon">Varón: </label>
```

```
<br/> <br/>
```

```
<input type="radio" name="sexo" id="mujer" value="Mujer">
```

```
<label for="mujer">Mujer: </label>
```

```
<br/> <br/>
```

```
<input type="submit" value="Enviar">
```

```
<input type="reset">
```

```
</p>
```

```
</form>
```

Parámetro de tipo opción (“sexo=Varón” o “sexo=Mujer”)

# HTTP POST con Python

```
import requests
```

```
page = 'http://www.undominio.com'
```

```
datosUsuario = {'nombre':'Enrique', 'apellidos':'Puertas',  
'email':'uncorre@gmail.com', 'sexo':'Varón'}
```

```
req = requests.post(page, data=datosUsuario)  
req2 = requests.get(page, data=datosUsuario)
```

# Interrogando formularios

The screenshot displays a web browser window showing IMDb movie listings. The top section lists movies with their ratings and descriptions. Below the listings, the Chrome DevTools Network tab is open, showing a list of requests. The 'Form Data' request is highlighted, and its details are shown in the right-hand pane.

**Movie Listings:**

- 4. Stranger Things (2016- )**  
45 min | Drama, Fantasy, Horror  
★ 8,3 [Rate this](#)  
A family of power-hungry thousand year old vampires look to take back the city that they built and dominate all those who have done them wrong.  
Stars: Joseph Morgan, Daniel Gillies, Claire Holt, Phoebe Tonkin  
Votes: 84.524
- 4. Stranger Things (2016- )**  
55 min | Drama, Fantasy, Horror  
★ 9,0 [Rate this](#)  
When a young boy disappears, his mother, a police chief, and his friends must confront terrifying forces in order to get him back.  
Stars: Winona Ryder, David Harbour, Finn Wolfhard, Millie Bobby Brown  
Votes: 302.432

**Network Tab:**

- Filter:  ☐ Regex ☐ Hide data URLs ☒ All XHR JS CSS Img Media Font Doc WS Manifest Other
- Timeline: 10000 ms, 20000 ms, 30000 ms, 40000 ms, 50000 ms, 60000 ms, 70000 ms, 80000 ms, 90000 ms, 100000 ms, 110000 ms, 120000 ms
- Name:
  - ☐ process
  - ☐ title?genres=horror&title\_type=tv\_series,mini\_series
  - ☐ consumersite-3586589641\_CB508966526\_.css
  - ☐ consumer-navbar-mega-238568768\_CB532297092\_.css
  - ☐ other-3780135229\_CB530008515\_.css
  - ☐ starbarwidget-2454701167\_CB522736579\_.css
  - ☐ watchlistButton-3806422028\_CB531876201\_.css
  - ☐ tarnhelm-4000355955\_CB533829584\_.js
  - ☐ DAsf-1.34\_FX1\_V275812351\_.js
  - ☐ ue-full-11e51f253e8ad9d145f4ed644b40f692\_V1\_.js
  - ☐ ads?gdfp\_req=1&correlator=4061578497069584&output=json\_html&callback=tinygpt.callback&impl=fifs&json\_a=1&iu\_parts=4215%2Cimdb2.co
  - ☐ imdbpro\_logo\_nb-3000473826\_CB530713470\_.png
  - ☐ imdbpro\_menu\_user-4091932618\_CB530713443\_.png
  - ☐ film-184890147\_CB522736516\_.png
  - ☐ A1EVAM02EL85F8:225-1512810-1327015:1J3PBWH575EN17CFSCXF:www.imdb.com\$uedata=s:%2Fuedata%2F225-1512810-1327015%2F%3FId%26
  - ☒ navbar\_sprite-580289297\_V\_.png
  - ☐ data:image/png;base...
  - ☐ view?xai=AKA0jstWAPz8XGa6RU04PiUPNgREBsAV0azMd\_Qzc5fQuOsPz7ul86c72PXLbsIR-5dGUIS\_vHdcacruzAhZIBFFVFXTRXoLOc6cqsWKajpY93:
  - ☐ imdb\_global\_sprite-1857117761\_V\_.png
- 67 requests | 78.1 KB transferred | Finish: 23.56 s | DOMContentLoaded: 3.20 s | Load: 3.98 s

**Form Data (Highlighted Request):**

- view source
- view URL encoded
- realm: title
- title:
- title\_type: tv\_series
- title\_type: mini\_series
- release\_date-min:
- release\_date-max:
- user\_rating-min:
- user\_rating-max:
- num\_votes-min:
- num\_votes-max:
- genres: horror
- boxoffice\_gross\_us-min:
- boxoffice\_gross\_us-max:
- keywords:
- locations:
- moviemeter-min:
- moviemeter-max:
- plot:
- role:
- runtime-min:

# Parseando webs con paginación

Podemos encontrar diferentes tipos de paginación:

- Enlace simple
- Números de páginas
- Numeración más “siguiente”
- Scroll infinito



# Parseando webs con paginación

```
import requests

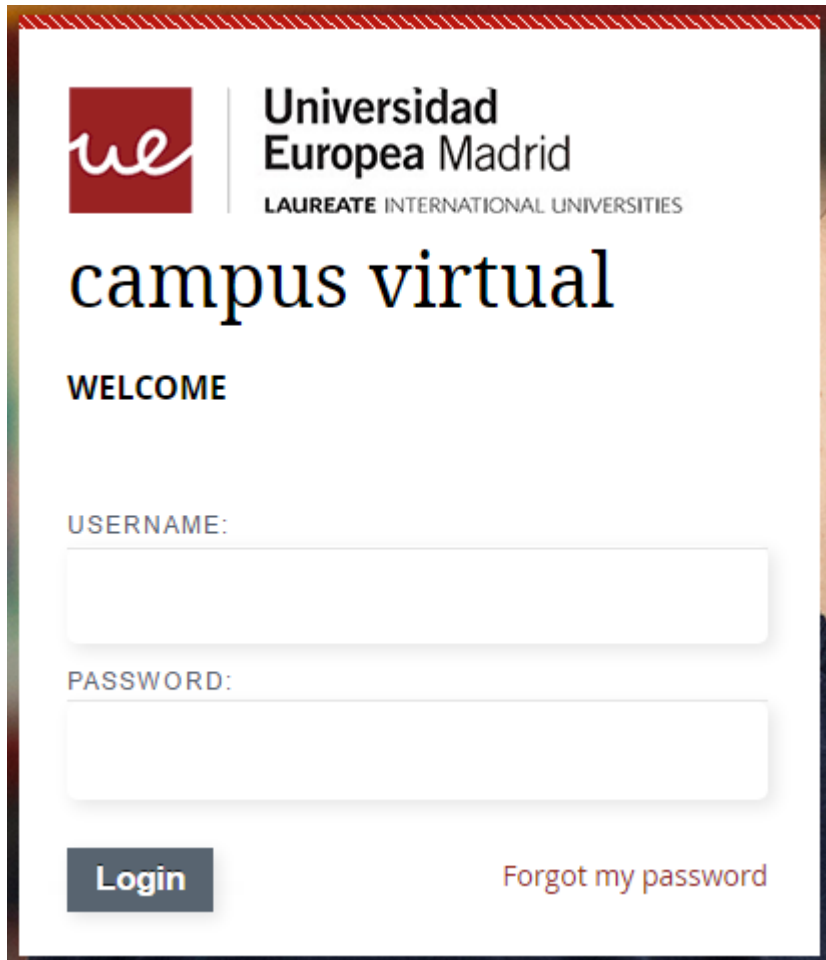
base_page = 'http://www.aprendebigdata.com/page-%s'
for i in range(1, 50):
    urlPage = base_page % i
    page = requests.get(urlPage)
```

Número fijo de  
páginas

```
from lxml import html
import requests

// [...]
page = requests.get('http://.....')
tree = html.fromstring(page.content)
enlaceSiguiente = tree.xpath('//div[@class="nav"]/a/@href')
```

Buscamos el  
enlace a la  
siguiente página



The image shows a login page for the 'campus virtual' of Universidad Europea Madrid. At the top left is the university's logo, a red square with white 'ue' text. To its right, the text 'Universidad Europea Madrid' is displayed in a bold, black, sans-serif font, with 'LAUREATE INTERNATIONAL UNIVERSITIES' in a smaller font below it. The main heading 'campus virtual' is in a large, black, serif font. Below this, the word 'WELCOME' appears in a bold, black, sans-serif font. The login section contains two labels, 'USERNAME:' and 'PASSWORD:', in a small, grey, sans-serif font, each followed by a white input field with a light grey border. At the bottom left is a dark grey 'Login' button with white text. To its right is a red link that says 'Forgot my password'.

- Se trata igual que un formulario.
- Al introducir usuario y contraseña se genera una sesión.
- Al hacer scraping hay que tener en cuenta en las peticiones la sesión.



# Sesiones con Python

```
url = "http://www.midominio.com/zonausuarios"
s = requests.Session()
s.headers['User-Agent'] = 'Mozilla/5.0 (Macintosh;...'

user = 'epuertas'
pwd = '12345678'
logindata = {'username':user,'password':pwd}
req = s.post(url, data=logindata)

if req.status_code == 200 :
    print 'login OK'
else:
    print 'Login incorrecto'
    quit()

req = s.get('http://www.midominio.com/zonausuarios/contactos')

htmldata = req.text
print htmldata
```

# Cómo evitar el bloqueo

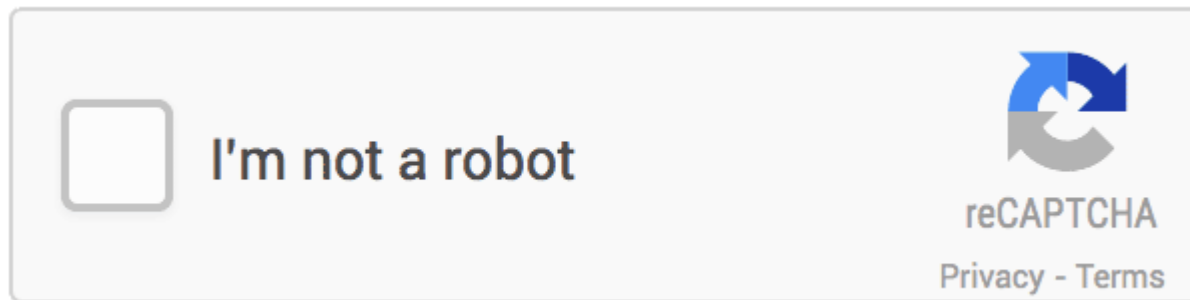
Algunos consejos para evitar ser bloqueados por los servidores:

- No hacer demasiadas peticiones simultaneas o en muy poco tiempo desde la misma IP.
- No hacer peticiones usando intervalos de tiempo exactos -> usar tiempos aleatorios.
- Evitar cadenas user-agent extrañas o características de bots.
- Usar http referer adecuado.



# CAPTCHAS

Captcha o CAPTCHA son las siglas de Completely Automated Public Turing test to tell Computers and Humans Apart (prueba de Turing completamente automática y pública para diferenciar ordenadores de humanos).



# CAPTCHAS

Diferentes enfoques para resolver captchas:

- Soluciones técnicas
  - OCR
  - Visión Artificial y Aprendizaje Automático
- Soluciones manuales
  - Death by Captcha ([www.deathbycaptcha.com](http://www.deathbycaptcha.com))  
Fácil de integrar mediante API  
1.39\$ por cada 1000 captchas