Web Scraping in Python

Internet map (2005)

What is

- **Data scraping** (“to scrape” = “raschiare”):

  Extract data from human-readable output, using an ad-hoc software (**crawler**)

- **Web scraping**: Extract data from websites
- **Web crawling** (“to crawl” = “avanzare lentamente”):

  Same, but focused on automatically browsing the web

- Normally, data transfer between programs is accomplished using **data structures** suited for automated processing by computers, not people
- Scraping ≈ **parsing**. But often ignores images or multimedia data, display formatting, redundant labels, superfluous commentary etc.
- Crawler ≈ **spider** (somewhat lighter) ≈ **bot** (automated)

Why is important

- **PageRank** (Google): How to rank web pages in search engine results?
  - *"A node is important if linked from other important and link parsimonious nodes or if it is highly linked"*

    - US Patent US7058628B1 (expired)

  ⇒ we need to navigate the web graph and extract hypertext links. Also relate the results to keywords.

- Scale-free **web topology** (Barabasi):
  - *"I had to wait two more years, until Jeong joined my research lab and built our own crawler, to get the data that allowed us to discover scale-free networks. Had we gotten the data in 1996, as I originally hoped to, we might have discovered it three years earlier."*

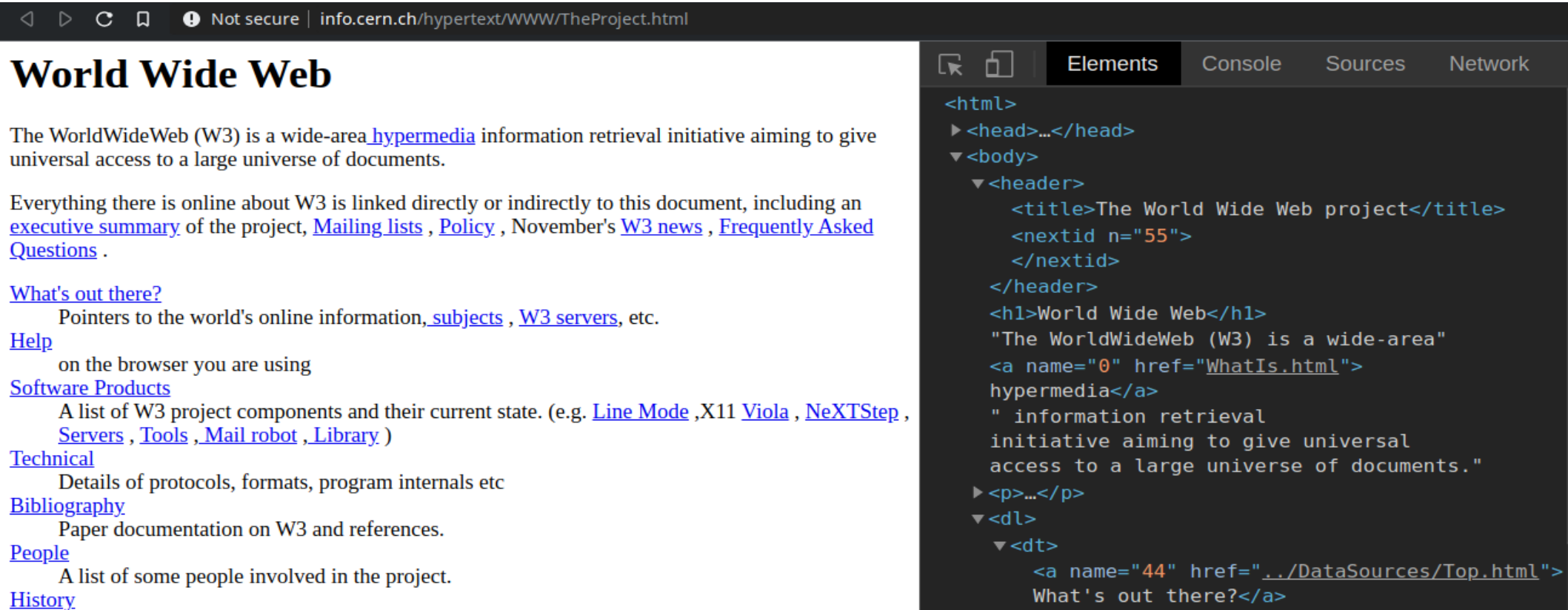
    - "Network Science" book

What this lecture…

- …is about:
  - The **Web** (HTML, DOM, CSS, XPath), REST APIs (XML, JSON) and the **browser inspection** tools
  - Python crawling & parsing libs (request, beautifulsoup, selenium, **scrapy**) and asynchronous features
  - Use-cases and examples (**fbcrawl**, twint)
- …is not about (but you may still want to know):
  - **DB admin** (store data) & **sysadmin** (schedule, log)
  - JavaScript libs
  - Commercial options
- Aims of lecture:
  - Build **basic dictionary & skills**, with a practical cut
  - Learn how to **get data** for a **research/commercial project**
  - Know **where to look**, for uncovered use-cases

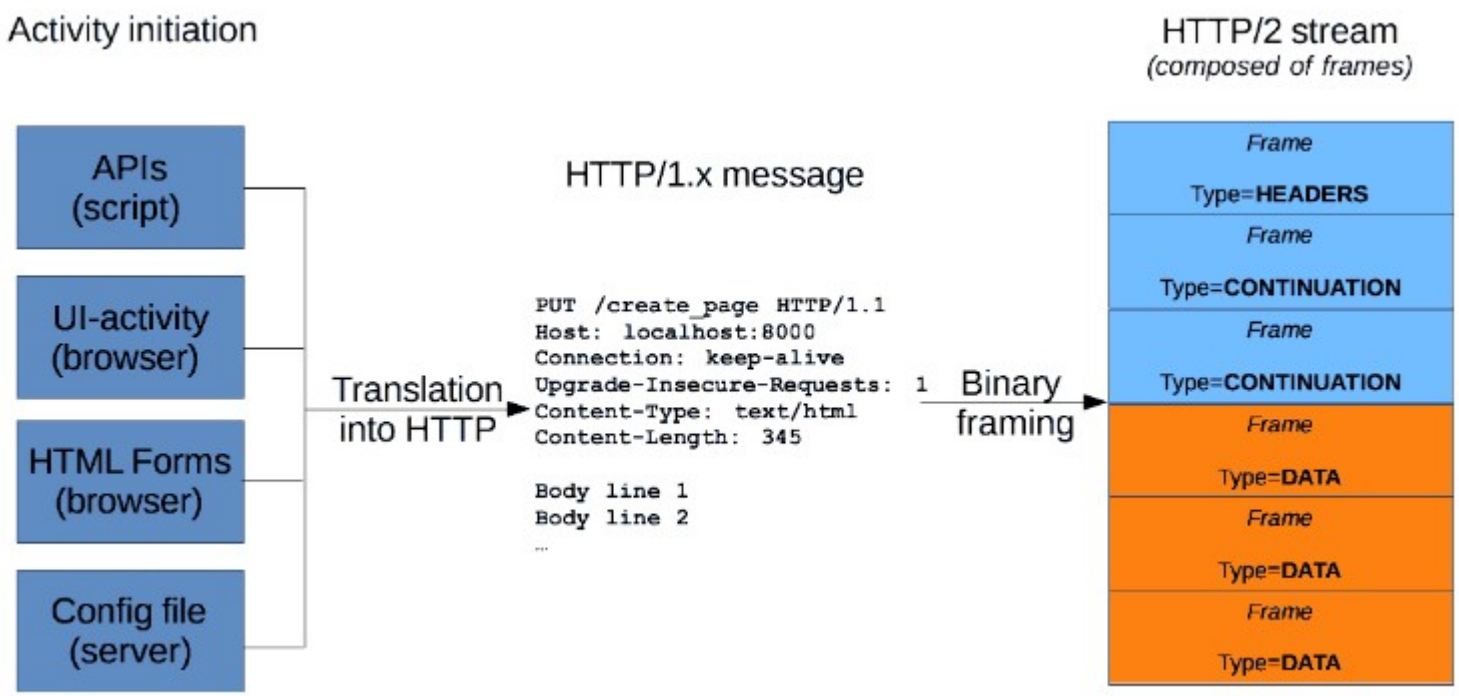
The Web (as it used to be)

- **Web pages**: documents (text) that have been formatted in Hypertext Markup Language (HTML)
- **HTTP** (Hypertext Transfer Protocol): the protocol (set of methods) used to transfer web resources (web pages, media, information...):
  - **GET**: **requests** a page from a server, yields a web page as **response**
  - **POST**: **asks** the web server to accept data enclosed in the **body** of the request message, most likely for storing it (often used when uploading a file or when submitting a web form)



HTTP Messages

- HTTP messages are how data is exchanged between a server and a client
- 2 types of messages: **requests** sent by the client to trigger an action on the server, and **responses**, the answer from the server.
- A browser, proxy, or web server provide HTTP messages through APIs (for browsers), config files (for proxies or servers) and such
- **Request = headers + body**



Browser inspection tools

- The browser performs request that can be inspected

Elements Console Sources **Network** Performance Memory Application Security

Preserve log Disable cache Online

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

50 ms 100 ms 150 ms 200 ms 250 ms 300 ms 350 ms 400 ms 450 ms 500 ms

Name

TheProject.html

Headers Preview Response Timing

General

Request URL: `http://info.cern.ch/hypertext/WWW/TheProject.html`

Request Method: `GET`

Status Code: `304 Not Modified`

Remote Address: `188.184.64.53:80`

Referrer Policy: `no-referrer-when-downgrade`

Response Headers view source

Connection: `close`

Date: `Tue, 04 Feb 2020 12:14:52 GMT`

ETag: `"40521e06-8a9-291e721905000"`

Server: `Apache`

Request headers

- Request headers leak information and directives to the server

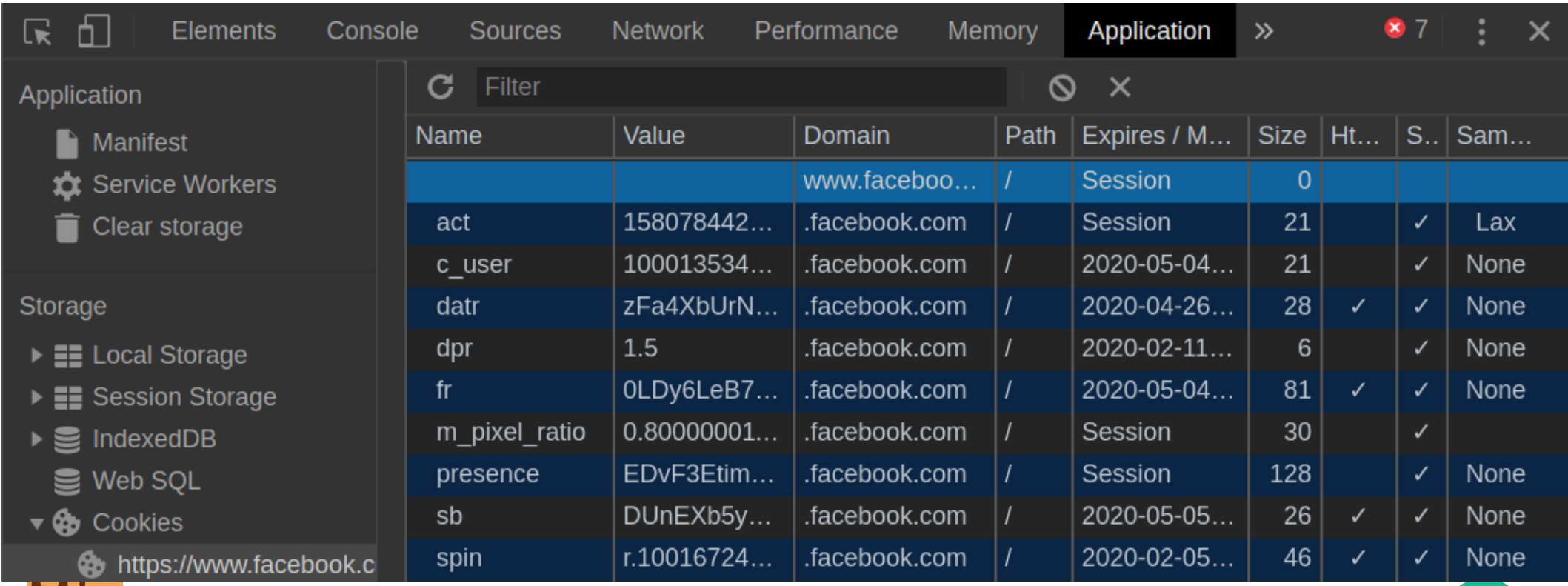
```
▼ Request Headers    view source
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apn
  g,*/*;q=0.8,application/signed-exchange;v=b3
  Accept-Encoding: gzip, deflate
  Accept-Language: en,en-US;q=0.9,en-GB;q=0.8,it;q=0.7
  Cache-Control: max-age=0
  Connection: keep-alive
  Host: info.cern.ch
  If-Modified-Since: Thu, 03 Dec 1992 08:37:20 GMT
  If-None-Match: "40521e06-8a9-291e721905000"
  Upgrade-Insecure-Requests: 1
  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
  hrome/77.0.3865.65 Safari/537.36
```

- They can also have cookies:

```
Headers   Preview   Response   Cookies
cookie: CONSENT=YES+IT.it+20160904-14-0;
```

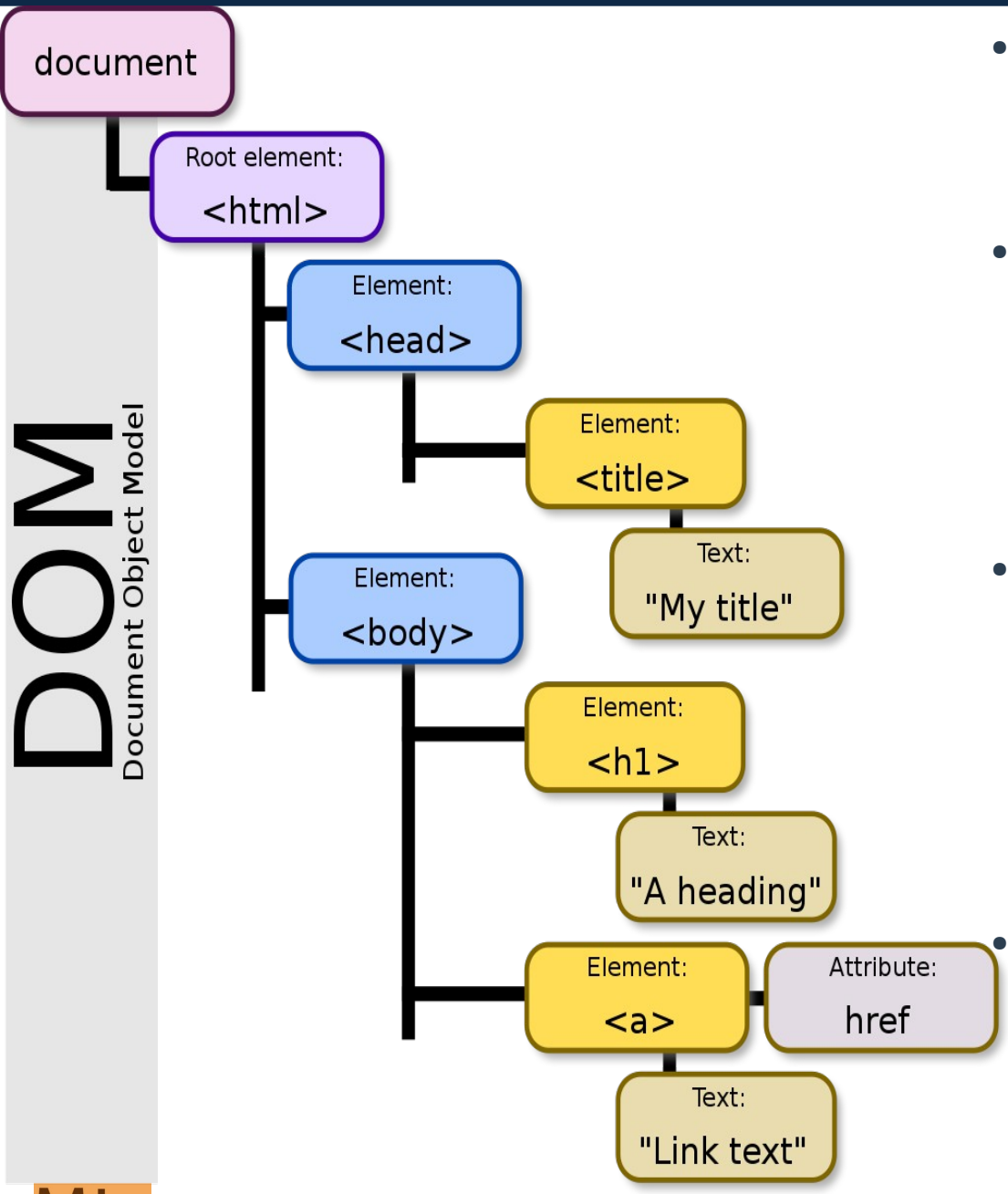
Cookies

- **Cookies**: data, stored in small **text files**, on your computer.
- When a web server has sent a web page to a browser, the connection is shut down, and **the server forgets** everything about the user. Cookies were invented to solve the problem "**how to remember** information about the user". When a user visits a web page, the username can be stored in a cookie (saved in **name-value pairs**)



Name	Value	Domain	Path	Expires / M…	Size	Ht…	S..	Sam…
		www.faceboo…	/	Session	0			
act	158078442…	.facebook.com	/	Session	21		✓	Lax
c_user	100013534…	.facebook.com	/	2020-05-04…	21		✓	None
datr	zFa4XbUrN…	.facebook.com	/	2020-04-26…	28	✓	✓	None
dpr	1.5	.facebook.com	/	2020-02-11…	6		✓	None
fr	0LDy6LeB7…	.facebook.com	/	2020-05-04…	81	✓	✓	None
m_pixel_ratio	0.80000001…	.facebook.com	/	Session	30		✓	
presence	EDvF3Etim…	.facebook.com	/	Session	128		✓	None
sb	DUnEXb5y…	.facebook.com	/	2020-05-05…	26	✓	✓	None
spin	r.10016724…	.facebook.com	/	2020-02-05…	46	✓	✓	None

Response: Web pages and DOM



- **HTML** (Hypertext Markup Language) is the markup language for constructing web pages
- **DOM**: each **web page** can be seen as a **tree** structure, where each **node** is an **HTML tag**, which is linked to its respective properties/fields/attributes
- The DOM is just a (language-independent) **model**, but is conceptually important since it allows to “see” (browse & select) the document much like a file-system
- **Selector APIs, class selectors and XPath** are built on top of the DOM model

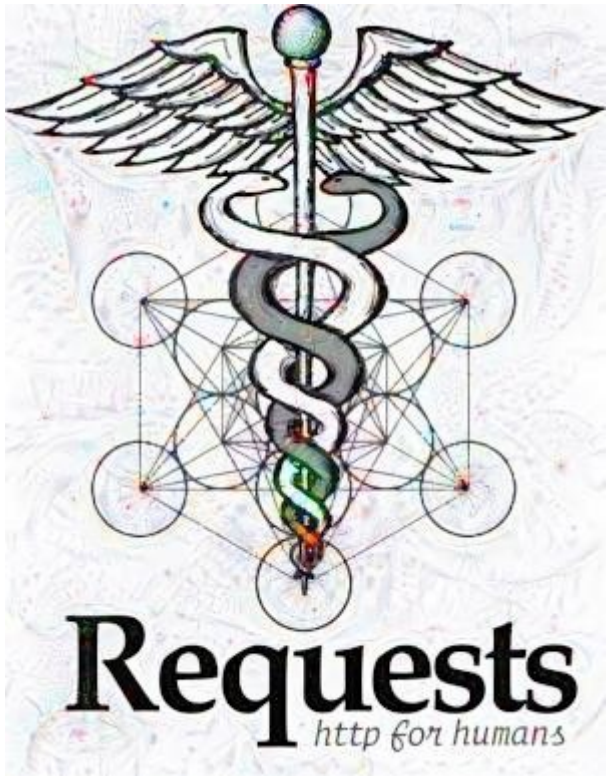
HTML and XML

- **HTML** doesn’t provide a specification to define new element and it uses predefined tags
- **XML** is a language that enables a user to **define a data structure** where values are assigned in each field in the structure
- **OpenOffice, RSS**, Atom, **SOAP**, **SVG**, and **XHTML** have been developed with XML syntax . XML has also provided the base language for communication protocols such as **XMPP (Jabber)**
- **XML** was designed to store and transport data
- XML-defined standards: **AJAX**, **DOM**, **XPath**, **XSLT**, **XQuery**

```
▼<note>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
 </note>
```

Requests and parsing in Python

- **Notebook**



The Web today

- With **HTML5** and **CSS3** you can build beautiful, responsive, static websites (Jekyll, Hugo etc.) which serve content to a user (animations, videos etc.), such as https://caos.space/

- If you (web developer) want to interact with user actions, you need more than HTML. You can write a **Web application**. For example a PHP script is executed on the server, and the plain HTML result is sent back to the browser.

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

- **JavaScript** is the programming language for the Web:
  - **Add new HTML** to the page, change the existing content, modify styles
  - React to user actions, run on mouse clicks, pointer movements, key presses
  - Send requests over the network to remote servers, download and upload files (so-called **AJAX** and COMET technologies).
  - Get and set **cookies**, ask questions to the visitor, show messages.
  - Remember the data on the client-side (“local storage”).

REST APIs and JSON

- **REST** (**Representational State Transfer**) is a structural design approach for crafting loosely attached applications using HTTP which is often implemented in the growth of web services
- REST web services do not impose any rule concerning how it needs to be put into practice at a subordinate level; it places high-level design guiding principle
- **JSON** (**JavaScript Object Notation**), and it's designed to store and transport data. JSON is designed to store and organize data similar to XML, but JSON is smaller, faster, and easier to parse than XML

```
{
  "students": [
    {
      "firstName": "David",
      "lastName": "Crosby"
    },
    {
      "firstName": "Alex",
      "lastName": "Rodriguez"
    },
    {
      "firstName": "Anna",
      "lastName": "Weston"
    }
  ]
}
```

XMLHttpRequest

- All modern browsers have a built-in **XMLHttpRequest** object to request data from a server
- The XMLHttpRequest object is a **developer's dream**, because you can:
  - **Update a web page** without reloading the page
  - Request data from a server - **after the page has loaded**
  - Receive data from a server - after the page has loaded
  - **Send data** to a server - in the background

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
       // Typical action to be performed when the document is ready:
       document.getElementById("demo").innerHTML = xhttp.responseText;
    }
};
xhttp.open("GET", "filename", true);
xhttp.send();
```

- It's a **scraper's nightmare**!

Selenium to the rescue

- **Selenium** (webdriver) is a tool that automates browser's behavior
- https://selenium-python.readthedocs.io
- Notebook

Scrapy

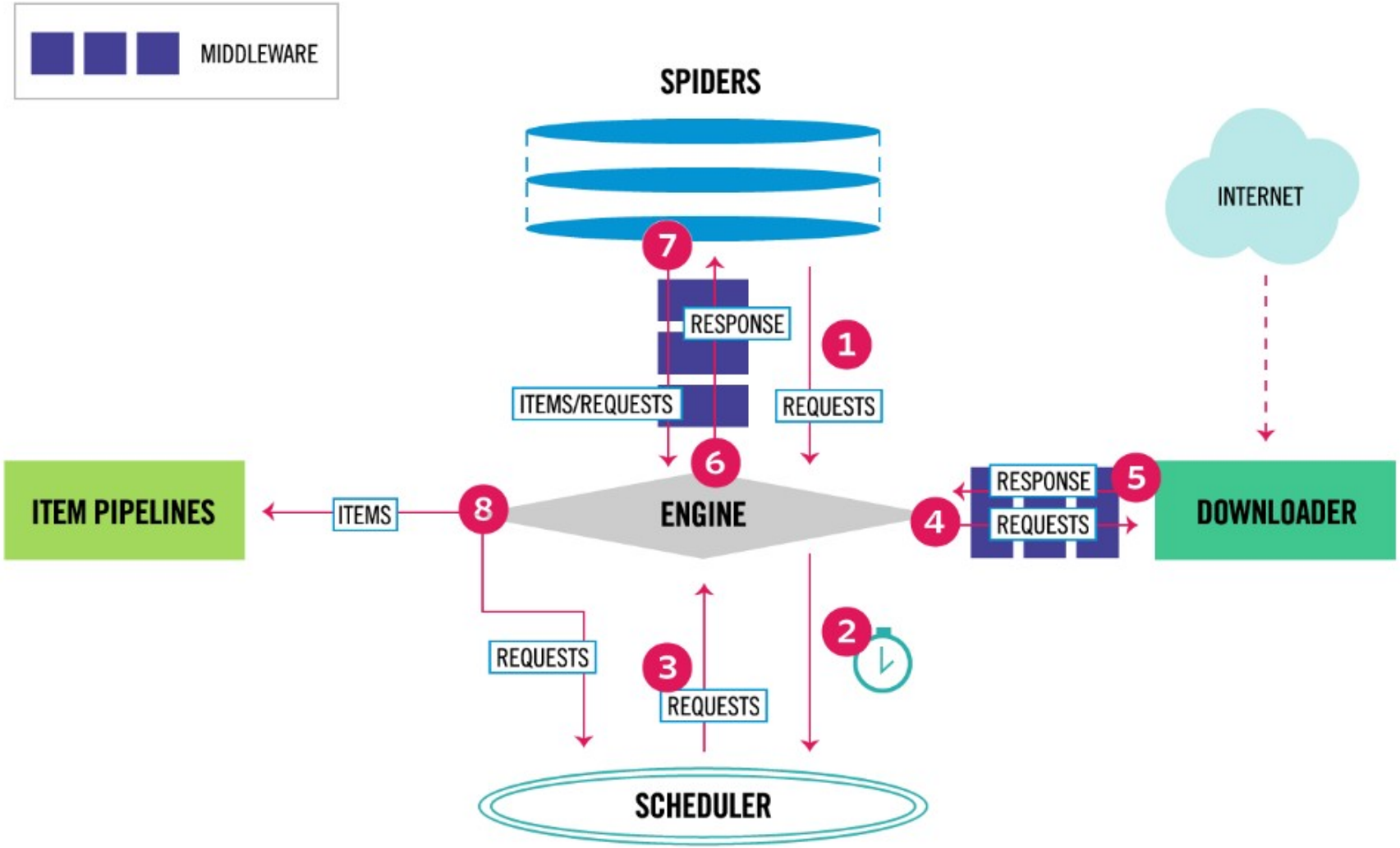
Scrapy

An open source and collaborative framework for extracting the data you need from websites.
In a fast, simple, yet extensible way.

- **Scrapy**: Advanced crawling engine based on the **twisted** framework
- Just a few years back, the developers only had forking and threaded servers to handle concurrent connections. Yet when network sessions would reach hundreds or even thousands, those generated too many separate, resource-consuming processes to be efficient. Twisted is a Now, we have an **asynchronous** Python framework for event-driven programming

Scrapy examples (newscrapy)

- https://docs.scrapy.org/en/latest/intro/tutorial.html



Scrapy

- Scrapy-selenium middleware
- Escort example
- Scrapy-useragents

Fbcrawl

- https://github.com/rugantio/fbcrawl

Twint

- https://github.com/twintproject/twint