Scraping Website Content

This blog post was translated by ChatGPT.

There are many existing tools available to scrape website content. However, using them doesn't help us understand the underlying process. When dealing with complex or specific websites at work, these tools might not yield the desired results. We need to reinvent the wheel to better learn and use these tools effectively.

Let's also look at some existing tools.

Data Miner

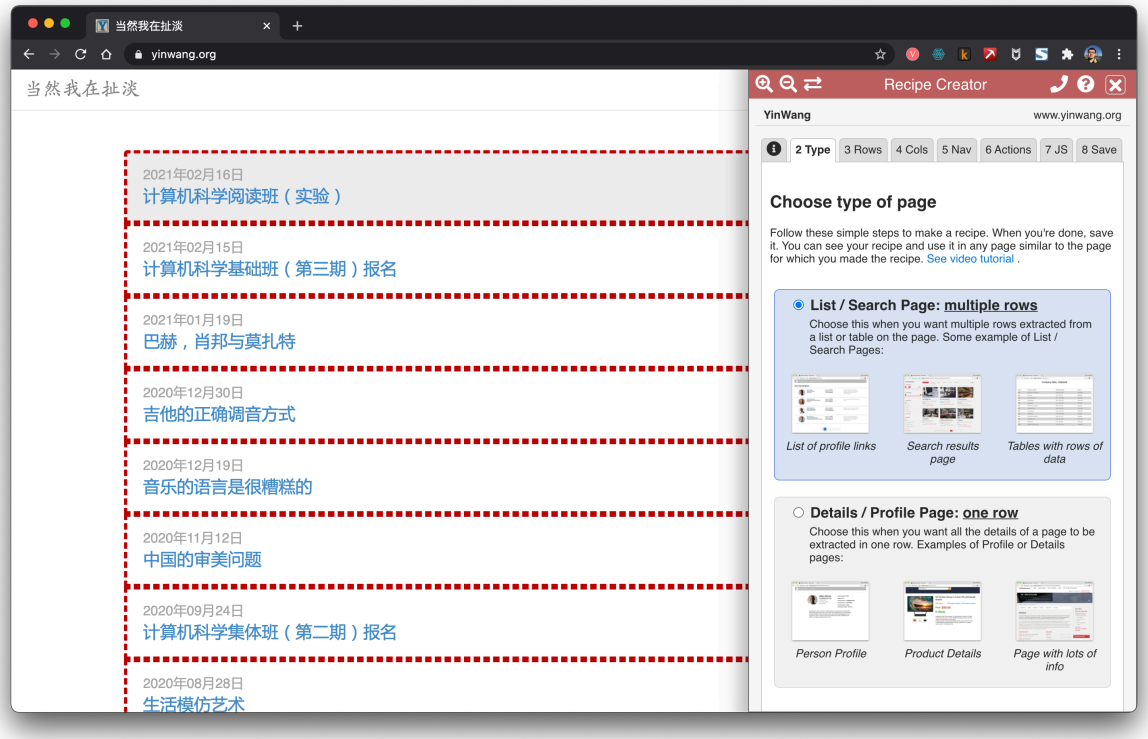

Figure 1: miner

`Data Miner` is a very convenient Chrome extension for scraping links and content.

getbook

`getbook` is a handy tool for creating eBooks.

```
pip install getbook
```

book.json:

```
{
  "uid": "book",
  "title": "Hello World",
  "author": "Armin",
  "chapters": [
    "http://lucumr.pocoo.org/2018/7/13/python/",
    "http://lucumr.pocoo.org/2017/6/5/diversity-in-technology",
  ]
}
```

```
getbook -f ./book.json --mobi
```

This makes it easy to turn some links into an eBook. By using `Data Miner` and `getbook`, one can scrape links and convert them into an eBook conveniently.

Feynman's Lectures on Physics

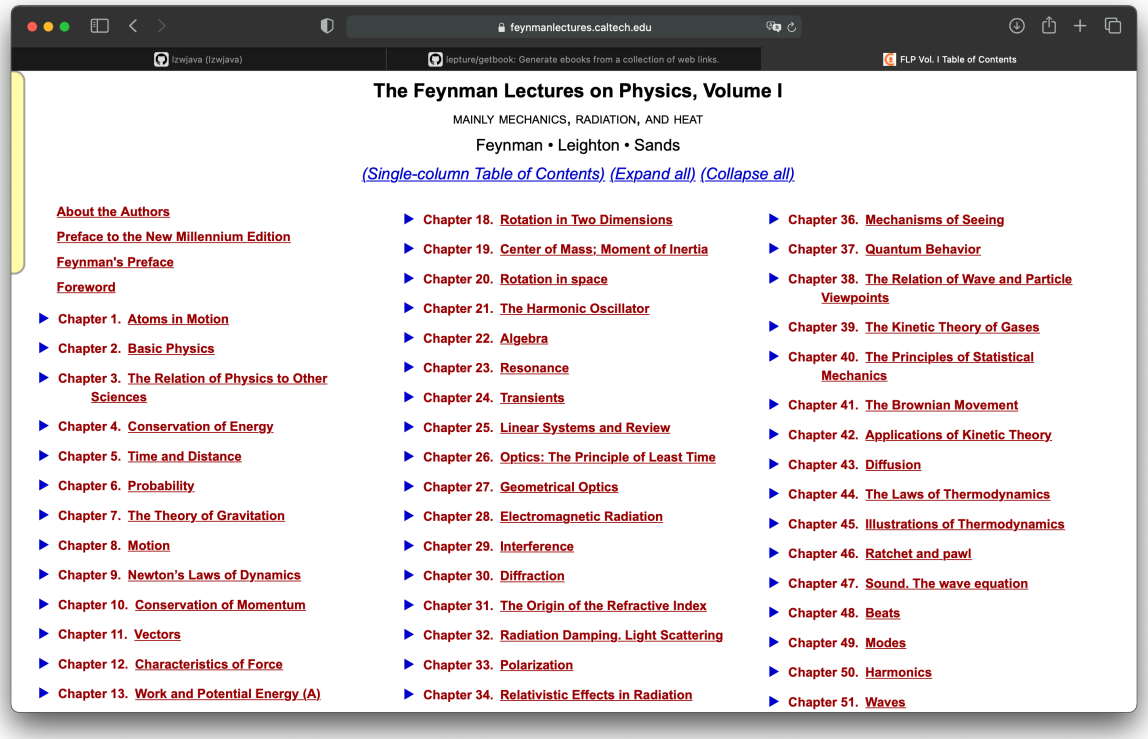

Figure 2: fl

In the chapter "Practical: Turning Feynman's Lectures on Physics Webpages into an eBook," we learned how to turn an `html` webpage rendered with `mathjax` into an eBook. Here, we continue this project to see how to get all the webpages. Feynman's Lectures on Physics has three volumes. The image above shows the table of contents for the first volume.

> http.client —HTTP protocol client
>
> Source code: Lib/http/client.py
>
> This module defines classes which implement the client side of the HTTP and HTTPS protocols. It is normally not used directly —the module urllib.request uses it to handle URLs that use HTTP and HTTPS.
>
> See also: The Requests package is recommended for a higher-level HTTP client interface.

It is evident that `requests` is a higher-level interface.

```
import requests

def main():
    r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
    print(r.status_code)

main()
```

```
401
```

```
import requests

def main():
    r = requests.get('https://github.com')
    print(r.status_code)
    print(r.text)

main()
```

```
200
<html>
  ...
</html>
```

After trying, it shows that the `requests` interface works.

```
<div class="toc-chapter" id="C03">
   <span class="triangle">

   </span>
   <a class="chapterlink" href="javascript:Goto(1,3)">
    <span class="tag">
     Chapter 3.
    </span>
    The Relation of Physics to Other Sciences
   </a>
   <div class="sections">
    <a href="javascript:Goto(1,3,1)">
     <span class="tag">
      3-1
     </span>
     Introduction
    </a>
    <a href="javascript:Goto(1,3,2)">
     <span class="tag">
      3-2
     </span>
     Chemistry
    </a>
    <a href="javascript:Goto(1,3,3)">
     <span class="tag">
      3-3
     </span>
     Biology
    </a>
    <a href="javascript:Goto(1,3,4)">
     <span class="tag">
      3-4
     </span>
     Astronomy
    </a>
    <a href="javascript:Goto(1,3,5)">
```

```
          <span class="tag">
           3-5
          </span>
          Geology
         </a>
         <a href="javascript:Goto(1,3,6)">
          <span class="tag">
           3-6
          </span>
          Psychology
         </a>
         <a href="javascript:Goto(1,3,7)">
          <span class="tag">
           3-7
          </span>
          How did it get that way?
         </a>
        </div>
       </div>
```

This is the `html` code for Chapter 3 on the table of contents page. To scrape the link for each chapter, `<a href="javascript:Goto(1,3,7)">`, it shows a `javascript` hyperlink.

`https://www.feynmanlectures.caltech.edu/I_03.html`

It turns out that the path for each chapter follows a pattern. `I_03.html` represents Volume 1, Chapter 3.

```
import requests
from bs4 import BeautifulSoup
from multiprocessing import Process


def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'scraping {url}')
    r = requests.get(url)
    if r.status_code != 200:
```

```
        raise Exception(r.status_code)
    soup = BeautifulSoup(r.text, features='lxml')
    f = open(f'./chapters/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

def main():
    for i in range(52):
        p = Process(target=scrape, args=(i+1,))
        p.start()
        p.join()

main()
```

Next, let's continue writing the scraping code. Here, `Process` is used.

```
    raise RuntimeError('''
RuntimeError:
        An attempt has been made to start a new process before the
        current process has finished its bootstrapping phase.

        This probably means that you are not using fork to start your
        child processes and you have forgotten to use the proper idiom
        in the main module:

            if __name__ == '__main__':
                freeze_support()
                ...

        The "freeze_support()" line can be omitted if the program
        is not going to be frozen to produce an executable.

def main():
    for i in range(52):
        p = Process(target=scrape, args=(i+1,))
        p.start()
    p.join()
```

```
if __name__ == "__main__":
    main()

def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(52)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('Time: ', stop - start)


if __name__ == "__main__":
    main()
```

```
scraping https://www.feynmanlectures.caltech.edu/I_01.html
scraping https://www.feynmanlectures.caltech.edu/I_04.html
...
scraping https://www.feynmanlectures.caltech.edu/I_51.html
scraping https://www.feynmanlectures.caltech.edu/I_52.html
Time:  9.144841699
```

```
<div class="figure" id="Ch1-F1">
        <img src="img/FLP_I/f01-01/f01-01_tc_big.svgz">
        <div class="caption empty">
         <span class="tag">
          Figure 1–1
         </span>
        </div>
</div>
```

```
import requests
from bs4 import BeautifulSoup
from multiprocessing import Process
import timeit


def scrape(chapter):
    if chapter < 1 or chapter > 52:
```

the most information in the fewest words? I believe it is the *atomic hypothesis* (or the atomic *fact* , or whatever you wish to call it) that *all things are made of atomsâlittle particles that move around in perpetual motion, attracting each other when they are a little distance apart, but repelling upon being squeezed into one another* . In that one sentence, you will see, there is an *enormous* amount of information about the world, if just a little imagination and thinking are applied.

Figure 1â1

To illustrate the power of the atomic idea, suppose that we have a drop of water a quarter of an inch on the side. If we look at it very closely we see nothing but waterâsmooth, continuous water. Even if we magnify it with the best optical microscope availableâroughly two thousand timesâthen the water drop will be roughly forty feet across, about as big as a large room, and if we looked rather closely, we would *still* see relatively smooth waterâbut here and there small football-shaped things swimming back and forth. Very interesting. These are paramecia. You may stop at this

Figure 3: fig

```
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'scraping {url}')
    r = requests.get(url)
    if r.status_code != 200:
        raise Exception(r.status_code)
    soup = BeautifulSoup(r.text, features='lxml')
    f = open(f'./chapters/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()


def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(52)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
```

```
    print('Time: ', stop - start)


if __name__ == "__main__":
    main()
```

Let’ s look at the links.

```
    imgs = soup.find_all('img')
    for img in imgs:
        print(img)
```

```
scraping https://www.feynmanlectures.caltech.edu/I_01.html
<img id="TwitLink" src=""/>
<img id="FBLink" src=""/>
<img id="MailLink" src=""/>
<img id="MobileLink" src="


"/>
<img id="DarkModeLink" src=""/>
<img id="DesktopLink" src=""/>
<img src="img/camera.svg"/>
<img src="img/FLP_I/f01-00/f01-00.jpg"/>
<img data-src="img/FLP_I/f01-01/f01-01_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-02/f01-02_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-03/f01-03_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-04/f01-04_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-05/f01-05_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-06/f01-06_tc_big.svgz"/>
<img class="first" data-src="img/FLP_I/f01-07/f01-07_tc_iPad_big_a.svgz"/>
<img class="last" data-src="img/FLP_I/f01-07/f01-07_tc_iPad_big_b.svgz"/>
<img data-src="img/FLP_I/f01-08/f01-08_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-09/f01-09_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-10/f01-10_tc_big.svgz"/>
```

https://www.feynmanlectures.caltech.edu/img/FLP_I/f01-01/f01-01_tc_big.svgz

```
Forbidden

You don't have permission to access this resource.
```

```
Apache/2.4.38 (Debian) Server at www.feynmanlectures.caltech.edu Port 443
```

```
% pip install selenium
Collecting selenium
  Using cached selenium-3.141.0-py2.py3-none-any.whl (904 kB)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/site-packages (from selenium) (1.24.1
Installing collected packages: selenium
Successfully installed selenium-3.141.0
```

```
export CHROME_DRIVER_HOME=$HOME/dev-env/chromedriver
export PATH="${PATH}:${CHROME_DRIVER_HOME}"
```

```
% chromedriver -h
Usage: chromedriver [OPTIONS]

Options
  --port=PORT                    port to listen on
  --adb-port=PORT                adb server port
  --log-path=FILE                write server log to file instead of stderr, increases log level to INF
  --log-level=LEVEL              set log level: ALL, DEBUG, INFO, WARNING, SEVERE, OFF
  --verbose                      log verbosely (equivalent to --log-level=ALL)
  --silent                       log nothing (equivalent to --log-level=OFF)
  --append-log                   append log file instead of rewriting
  --replayable                   (experimental) log verbosely and don't truncate long strings so that t
  --version                      print the version number and exit
  --url-base                     base URL path prefix for commands, e.g. wd/url
  --readable-timestamp           add readable timestamps to log
  --enable-chrome-logs           show logs from the browser (overrides other logging options)
  --allowed-ips                  comma-separated allowlist of remote IP addresses which are allowed to
```

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.expected_conditions import presence_of_element_located

with webdriver.Chrome() as driver:
```

```
    wait = WebDriverWait(driver, 10)
    driver.get("https://google.com/ncr")
    driver.find_element(By.NAME, "q").send_keys("cheese" + Keys.RETURN)
    first_result = wait.until(presence_of_element_located((By.CSS_SELECTOR, "h3>div")))
    print(first_result.get_attribute("textContent"))


from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui.WebDriverWait
from selenium.webdriver.support.expected_conditions import presence_of_element_located
import urllib

def main():
    driver = webdriver.Chrome()
    wait = WebDriverWait(driver, 10)
    driver.get("https://www.feynmanlectures.caltech.edu/I_01.html")
    elements = driver.find_elements(By.TAG_NAME, "img")
    # print(dir(elements[0]))
    print(driver.page_source)
    i = 0
    for element in elements:
        # src = element.get_attribute('src')
        element.screenshot(f'images/{i}.png')
        i +=1
    driver.close()
main()
```

```
from bs4 import BeautifulSoup
from multiprocessing import Process
import timeit
from pathlib import Path
from selenium import webdriver
from selenium.webdriver.common.by import By

def img_path(chapter):
```

```
    return f'./chapters/{chapter}/img'

def img_name(url):
    splits = url.split('/')
    last = splits[len(splits) - 1]
    parts = last.split('.')
    name = parts[0]
    return name

def download_images(driver: webdriver.Chrome, chapter):
    path = img_path(chapter)
    Path(path).mkdir(parents=True, exist_ok=True)

    elements = driver.find_elements(By.TAG_NAME, "img")
    for element in elements:
        src = element.get_attribute('src')
        name = img_name(src)
        element.screenshot(f'{path}/{name}.png')

USER_AGENT = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, like Gecko) '

def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    driver = webdriver.Chrome()
    driver.get(url)
    page_source = driver.page_source
    Path(f'./chapters/{chapter_str}').mkdir(parents=True, exist_ok=True)
    print(f'scraping {url}')

    download_images(driver, chapter_str)

    soup = BeautifulSoup(page_source, features='lxml')
    imgs = soup.find_all('img')
```

```
    for img in imgs:
        if 'src' in img.attrs or 'data-src' in img.attrs:
            src = ''
            if 'src' in img.attrs:
                src = img.attrs['src']
            elif 'data-src' in img.attrs:
                src = img.attrs['data-src']
                del img.attrs['data-src']
            name = img_name(src)
            img.attrs['src'] = f'img/{name}.png'

    f = open(f'./chapters/{chapter_str}/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

    driver.close()


def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(2)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('Time: ', stop - start)

if __name__ == "__main__":
    main()
```

```
scraping https://www.feynmanlectures.caltech.edu/I_01.html
scraping https://www.feynmanlectures.caltech.edu/I_02.html
Time:  21.478510914999998
```

```
 errpipe_read, errpipe_write = os.pipe()
OSError: [Errno 24] Too many open files
```

```
 % ulimit a
ulimit: invalid number: a
lzw@lzwjava feynman-lectures-mobi % ulimit -a
-t: cpu time (seconds)              unlimited
-f: file size (blocks)              unlimited
-d: data seg size (kbytes)          unlimited
-s: stack size (kbytes)             8192
-c: core file size (blocks)         0
-v: address space (kbytes)          unlimited
-l: locked-in-memory size (kbytes)  unlimited
-u: processes                       2784
-n: file descriptors                256

12
download_images
12
mathjax2svg
latexs 128
make_svg 0
insert_svg 0
make_svg 1
insert_svg 1
make_svg 2
insert_svg 2
make_svg 3
insert_svg 3
convert

12
download_images
12
mathjax2svg
latexs 0
latexs 0
convert
Time:  11.369145162

% grep --include=\*.html -r '\$' *
```

```
43/I_43.html:a long period of time $T$, have a certain number, $N$, of hits. If we
43/I_43.html:number of collisions is proportional to the time $T$. We would like to
43/I_43.html:We have written the constant of proportionality as $1/\tau$, where
43/I_43.html:$

\tau$ will have the dimensions of a time. The constant $\tau$ is the
43/I_43.html:there are $60$ collisions; then $\tau$ is one minute. We would say
43/I_43.html:that $\tau$ (one minute) is the
```

错误 E21018: 解析文件中的内容时，创建改进的 Mobi 域名失败。内容：<In earlier chapters > 文件中： /pr
提醒 W28001: Kindle 阅读器不支持内容中指定的 CSS 样式。正在删除 CSS 属性： 'max-width' 文件中： /pri
提醒 W29004: 强制关闭的已打开标签为： 文件中： /private/var/folders/_3/n3b7c
提醒 W29004: 强制关闭的已打开标签为： <p amzn-src-id="975"> 文件中： /private/var/folders/_3/n3b7dq8x

提醒 W14001: 超链接出现问题，尚未解决： /private/var/folders/_3/n3b7dq8x6652drmx6_d3t3bh0000gr/T/97c
提醒 W14001: 超链接出现问题，尚未解决： /private/var/folders/_3/n3b7dq8x6652drmx6_d3t3bh0000gr/T/97c
提醒 W14001: 超链接出现问题，尚未解决： /private/var/folders/_3/n3b7dq8x6652drmx6_d3t3bh0000gr/T/97c

```
<span class="disabled" href="#Ch1-F1">
          1−1
</span>

Rasterizing 'OEBPS/84b8b4179175f097be1180a10089107be75d7d85.svg' to 1264x1011
Rasterizing 'OEBPS/23a4df37f269c8ed43f54753eb838b29cff538a1.svg' to 1264x259
Traceback (most recent call last):
  File "runpy.py", line 194, in _run_module_as_main
  File "runpy.py", line 87, in _run_code
  File "site.py", line 39, in <module>
  File "site.py", line 35, in main
  File "calibre/utils/ipc/worker.py", line 216, in main
  File "calibre/gui2/convert/gui_conversion.py", line 41, in gui_convert_override
  File "calibre/gui2/convert/gui_conversion.py", line 28, in gui_convert
  File "calibre/ebooks/conversion/plumber.py", line 1274, in run
  File "calibre/ebooks/conversion/plugins/mobi_output.py", line 214, in convert
  File "calibre/ebooks/conversion/plugins/mobi_output.py", line 237, in write_mobi
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 55, in __call__
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 142, in rasterize_spine
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 152, in rasterize_item
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 185, in rasterize_external
```

```
  File "calibre/ebooks/oeb/base.py", line 1092, in bytes_representation
  File "calibre/ebooks/oeb/base.py", line 432, in serialize
TypeError: cannot convert 'NoneType' object to bytes

% kindlepreviewer feynman-lectures-on-physics-volumn-1.epub -convert
Checking specified arguments.
Pre-processing in progress.
Processing 1/1 book(s).
Book converted with warnings! : /Users/lzw/projects/feynman-lectures-mobi/feynman-lectures-on-physics-v
Post-processing in progress.
Writing output/log files to /Users/lzw/projects/feynman-lectures-mobi/output

Cleaning up manifest...
Trimming unused files from manifest...
Creating AZW3 Output...
Serializing resources...
Splitting markup on page breaks and flow limits, if any...
Creating KF8 output
    Generating KF8 markup...
Tag table has no aid and a too large chunk size. Adding anyway.
Tag table has no aid and a too large chunk size. Adding anyway.
Tag table has no aid and a too large chunk size. Adding anyway.
    Compressing markup...
    Creating indices...
```