3 Python web scrapers and crawlers

Check out these great Python tools for crawling and scraping the web, and parsing out the data you need.

28 Sep 2017 | Jason Baker (Red Hat) (/users/jason-baker) | 753 | 4 comments

***Image credits :** You as a Machine. Modified by Rikki Endsley. CC BY-SA 2.0.*

In a perfect world, all of the data you need would be cleanly presented in an open and well-documented format that you could easily download and use for whatever purpose you need.

In the real world, data is messy, rarely packaged how you need it, and often out-of-date.

More Python Resources

- What is an IDE? (https://www.redhat.com/en/topics/middleware/what-is-ide?intcmp=7016000000127cYAAQ)
- Cheat sheet: Python 3.7 for beginners (https://opensource.com/downloads/cheat-sheet-python-37-beginners?intcmp=7016000000127cYAAQ)
- Top Python GUI frameworks (https://opensource.com/resources/python/gui-frameworks?intcmp=7016000000127cYAAQ)
- Download: 7 essential PyPI libraries (https://opensource.com/downloads/7-essential-pypi-libraries?intcmp=7016000000127cYAAQ)
- Red Hat Developers (https://developers.redhat.com/?intcmp=7016000000127cYAAQ)
- Latest Python content (https://opensource.com/tags/python?intcmp=7016000000127cYAAQ)

Often, the information you need is trapped inside of a website. While some websites make an effort to present data in a clean, structured data format, many do not. Crawling (https://en.wikipedia.org/wiki/Web_crawler), scraping locked away in a proprietary database.

Sooner or later, you're going to find a need to do some crawling and scraping to get the data you need, and almost certainly you're going to need to do a little coding to get it done right. How you do this is up to you, but I've found the Python community to be a great provider of tools, frameworks, and documentation for grabbing data off of websites.

Before we jump in, just a quick request: think before you do, and be nice. In the context of scraping, this can mean a lot of things. Don't crawl websites just to duplicate them and present someone else's work as your own (without permission, of course). Be aware of copyrights and licensing, and how each might apply to whatever you have scraped. Respect robots.txt (http://www.robotstxt.org/) files. And don't hit a website so frequently that the actual human visitors have trouble accessing the content.

With that caution stated, here are some great Python tools for crawling and scraping the web, and parsing out the data you need.

Pyspider

Let's kick things off with pyspider (https://github.com/binux/pyspider), a web-crawler with a web-based user interface that makes it easy to keep track of multiple crawls. It's an extensible option, with multiple backend databases and message queues supported, and several handy features baked in, from prioritization to the ability to retry failed pages, crawling pages by age, and others. Pyspider supports both Python 2 and 3, and for faster crawling, you can use it in a distributed format with multiple crawlers going at once.

Licensed under the Apache 2 license, pyspyder is still being actively developed on GitHub.

MechanicalSoup

MechanicalSoup (https://github.com/hickford/MechanicalSoup) is a crawling library built around the hugely-popular and incredibly versatile HTML parsing library Beautiful Soup (https://www.crummy.com/software/BeautifulSoup/). If your crawling needs are fairly simple, but require you to check a few boxes or enter some text and you don't want to build your own crawler for this task, it's a good option to consider.

MechanicalSoup is licensed under an MIT license. For more on how to use it, check out the example source file example.py (https://github.com/hickford/MechanicalSoup/blob/master/example.py) on the project's GitHub page. Unfortunately, the project does not have robust documentation at this time

Scrapy

Scrapy (https://scrapy.org/) is a scraping framework supported by an active community with which you can build your own scraping tool. In addition to scraping and parsing tools, it can easily export the data it collects in a number of formats like JSON or CSV and store the data on a backend of your choosing. It also has a number of built-in extensions for tasks like cookie handling, user-agent spoofing, restricting crawl depth, and others, as well as an API for easily building your own additions.

For an introduction to Scrapy, check out the online documentation code base can be found on GitHub (https://github.com/scrapy/scrapy) under a 3-clause BSD license.

If you're not all that comfortable with coding, Portia (https://github.com/scrapinghub/portia) provides a visual interface that makes it easier. A hosted version is available at scrapinghub.com (https://portia.scrapinghub.com/).

Others

- Cola (https://github.com/chineking/cola) describes itself as a “high-level distributed crawling framework” that might meet your needs if you're looking for a Python 2 approach, but note that it has not been updated in over two years.

- Demiurge (https://github.com/matiasb/demiurge), which supports both Python 2 and Python 3, is another potential candidate to look at, although development on this project is relatively quiet as well.

- Feedparser (https://github.com/kurtmckee/feedparser) might be a helpful project to check out if the data you are trying to parse resides primarily in RSS or Atom feeds.

- Lassie (https://github.com/michaelhelmick/lassie) makes it easy to retrieve basic content like a description, title, keywords, or a list of images from a webpage.

- RoboBrowser (https://github.com/jmcarp/robobrowser) is another simple library for Python 2 or 3 with basic functionality, including button-clicking

This is far from a comprehensive list, and of course, if you're a master coder you may choose to take your own approach rather than use one of these frameworks. Or, perhaps, you've found a great alternative built for a different language. For example, Python coders would probably appreciate checking out the Python bindings (https://selenium-python.readthedocs.io/) for Selenium (https://github.com/SeleniumHQ/selenium) for sites that are trickier to crawl without using an actual web browser. If you've got a favorite tool for crawling and scraping, let us know in the comments below.

Topics :

Python (/tags/python) **Internet** (/tags/internet)

Programming (/tags/programming)

(/users/jason-baker)

About the author

Jason Baker - I use technology to make the world more open. Linux desktop enthusiast. Map/geospatial nerd. Raspberry Pi tinkerer. Data analysis and visualization geek. Occasional coder. Cloud nativist. Civic tech and open government booster.

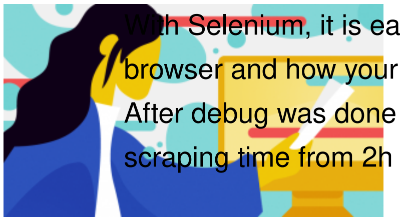
• More about me (/users/jason-baker)

Recommended reading

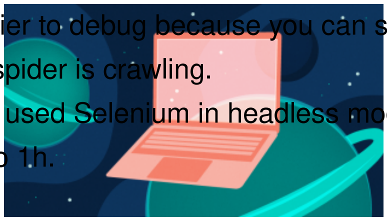
Improve your time management with Jupyter (/article/20/9/calendar-jupyter?utm_campaign=intrel)

Use Python to solve a charity's business problem (/article/20/9/solve-problem-python?utm_campaign=intrel)

How this open source test framework evolves with .NET (/article/20/9/testing-net-fixie?utm_campaign=int



Create a slide deck using Jupyter Notebooks (/article/20/9/presentation-jupyter-notebooks?utm_campaign=intrel)



Build a remote management console using Python and Jupyter Notebooks (/article/20/9/remote-management-

Managing a non-profit organization's supply chain with Groovy (/article/20/9/groovy?utm_camp

4 Comments

vigneshlkp on 30 Sep 2017 0

Good

Sasa Buklijas on 03 Oct 2017 1

A few years ago I started with Beautiful Soup.

For one recent project, started 2 years ago and still in daily use, I used Selenium.
With Selenium, it is easier to debug because you can see what is happening in a browser and how your spider is crawling.
After debug was done I used Selenium in headless mode (with phantomjs), it reduced scraping time from 2h to 1h.

Dan Hemberger on 01 Nov 2017 2

Thanks for the summary Jason! By the way, the documentation of MechanicalSoup has improved significantly in the past few months. There's now an extensive Read the Docs site: https://mechanicalsoup.readthedocs.io/en/latest/ (https://mechanicalsoup.readthedocs.io/en/latest/)

Jacksonp2008 on 04 Nov 2017 0

What's the best spider that will index into elasticsearch 5+ ?