Web Scraping

APIs? Where we're going, we don't need APIs.

Richard E Sarkis
Rochester's Python User Group
June 17th, 2016 Meeting

Reasons

- Preclude any manual data copy-pasting from a website
- If an API to a website isn't provided
- If an API is restrictive monetarily, or in capability

Problems

- Fragility: Websites don't rely on semantic description of it's content, relying instead of relative placement and HTML tag attributes
- TOS: Website may not take kindly to high-throughput scraping
- Portability: You're writing your own API using one language. It'll take work to port it to another language, usually this work is done for you in an official web API for a service

Tools

- Web browser (Firefox, Safari, Chrome, etc) with decent dev tools, or equivalent
    - and, a website you wish to scrape!
- Python 3 (requests, BeautifulSoup)
- requestb.in

Web Dev Tools

Web Browser Dev Tools

- The major web browser: Firefox, Chrome and Safari have built-in web developer interfaces
- Excellent starter tool for reverse-engineering client-server communication
- Inspectors for HTML/CSS/JS
- Network tracing for resource loading and metadata

Demo

Using Firefox

Request Testing

- requestb.in assist a web developer in debugging their HTTP requests

https://requestb.in

Demo

Using requestsb.in

Python

Requests

Python: Requests

- Simple, clean module for HTTP(S) session handling
- Easily handle cookies, POST data, query parameters in URLs all in Pythonic style

http://docs.python-requests.org/en/master/

Python: Requests

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type":"User"...'
>>> r.json()
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

Demo

Requests Module

Python

BeautifulSoup

Python: BeautifulSoup

BeautifulSoup…

- provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree
- sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility

https://www.crummy.com/software/BeautifulSoup/

Demo

BeautifulSoup

Case Study

Static Websites
GET Submissions

Case Study: Static Websites

- `crime_report.py`
  - Scrape incident reports from the Monroe County website (tricky)
- `faculty_list.py`
  - Scrape the faculty list from the U of R Physics & Astronomy department webpage

Case Study

Dynamic Websites (GET & POST Submissions)

Case Study: Dynamic Websites

- `arxiv.py`
  - Submit a search query to retrieve papers matching the query
- `google_form.py`
  - A 'robot' form submitter for a Google Form

Conclusion

Q & A