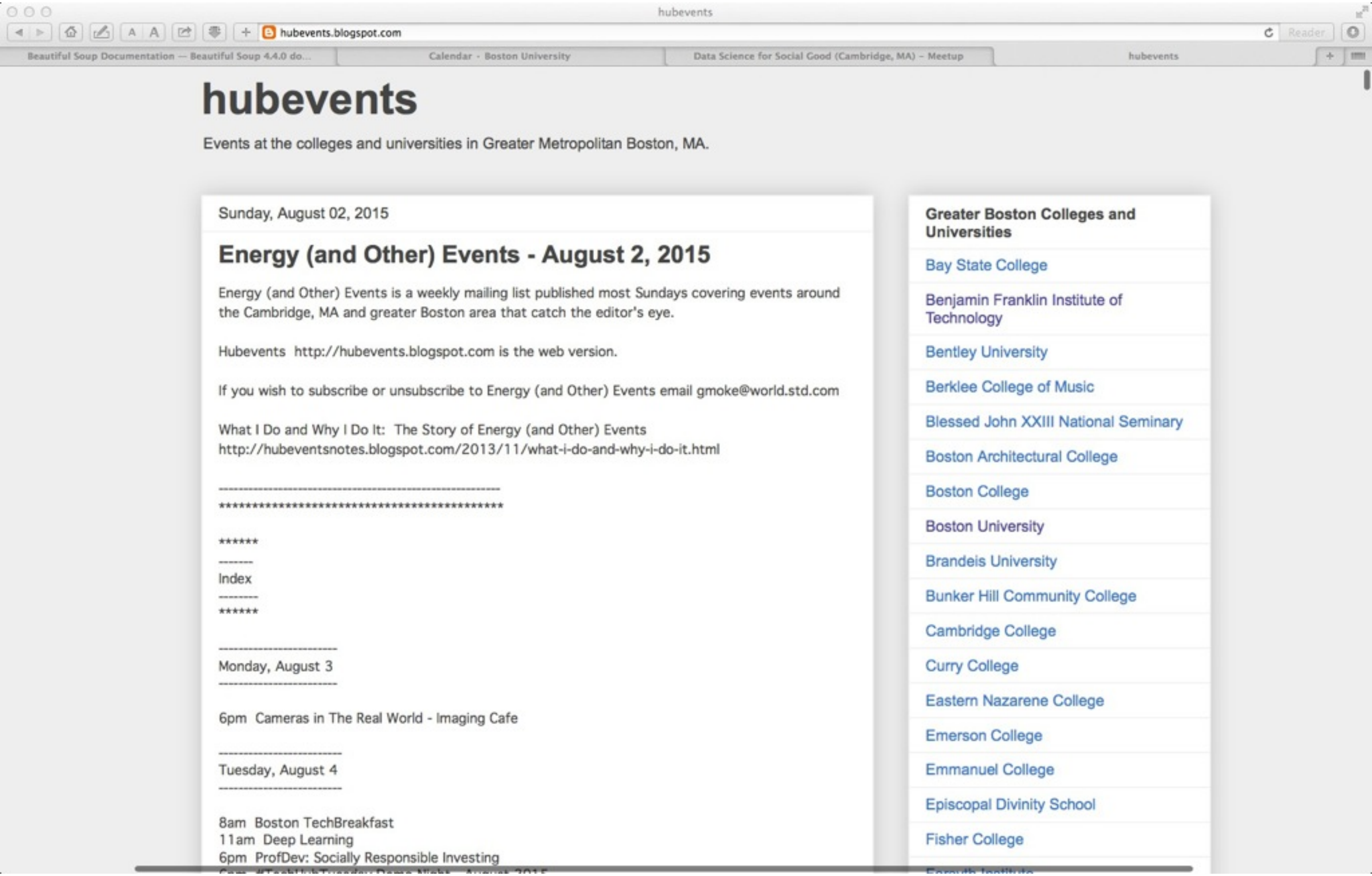
Web Scraping with Python: `Requests` and `Beautiful Soup`

Jonathan Halverson, Ph.D.
halverson.jonathan@gmail.com

Boston Data-Con
Saturday, August 8, 2015


hubevents
hubevents.blogspot.com
Reader
Beautiful Soup Documentation — Beautiful Soup 4.4.0 do...
Calendar - Boston University
Data Science for Social Good (Cambridge, MA) - Meetup
hubevents

hubevents

Events at the colleges and universities in Greater Metropolitan Boston, MA.

Sunday, August 02, 2015

Energy (and Other) Events - August 2, 2015

Energy (and Other) Events is a weekly mailing list published most Sundays covering events around the Cambridge, MA and greater Boston area that catch the editor's eye.

Hubevents http://hubevents.blogspot.com is the web version.

If you wish to subscribe or unsubscribe to Energy (and Other) Events email gmoke@world.std.com

What I Do and Why I Do It: The Story of Energy (and Other) Events
http://hubeventsnotes.blogspot.com/2013/11/what-i-do-and-why-i-do-it.html

----
****

******

Index

******

Monday, August 3

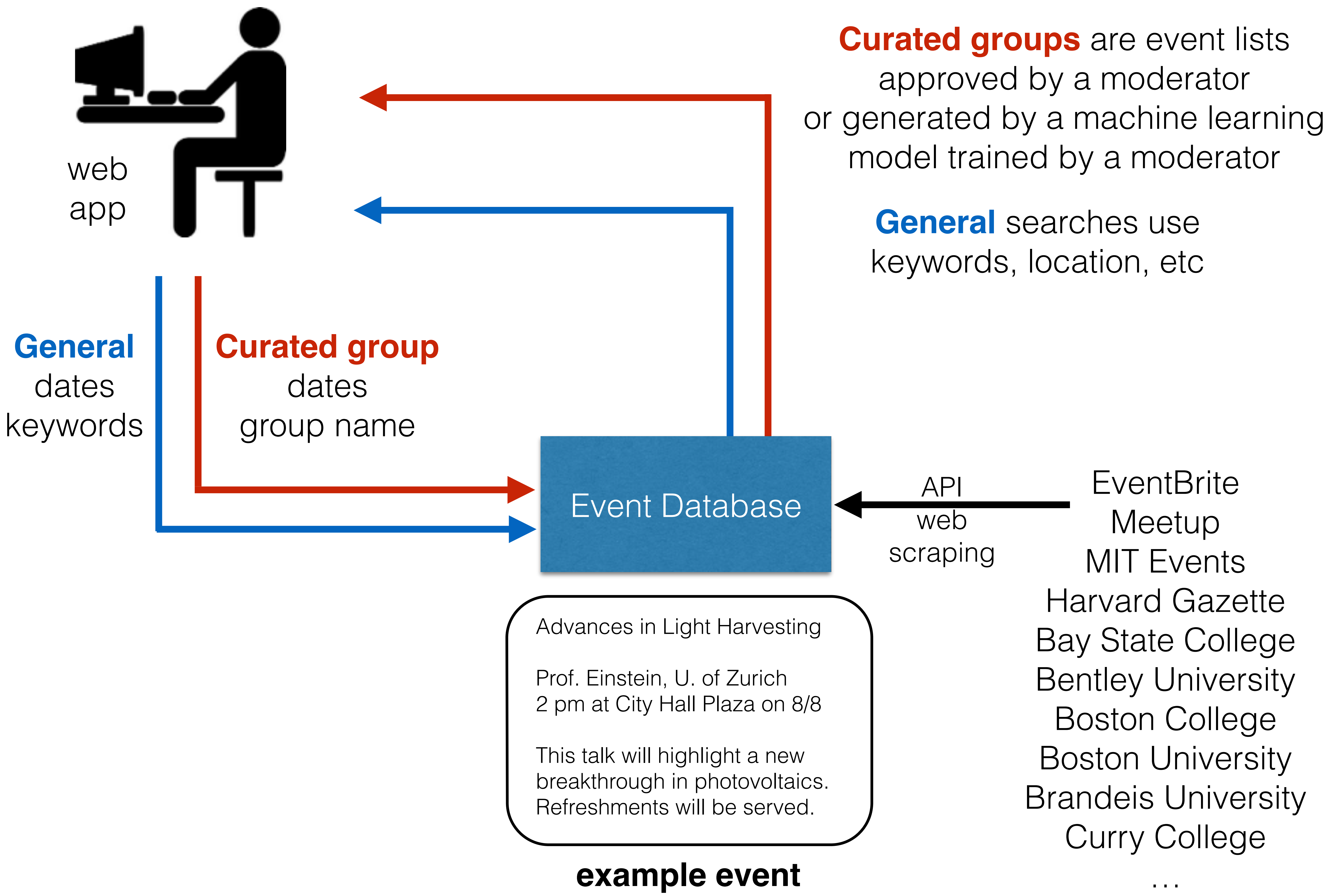
6pm Cameras in The Real World - Imaging Cafe

Tuesday, August 4

8am Boston TechBreakfast
11am Deep Learning
6pm ProfDev: Socially Responsible Investing

Greater Boston Colleges and Universities

- Bay State College
- Benjamin Franklin Institute of Technology
- Bentley University
- Berklee College of Music
- Blessed John XXIII National Seminary
- Boston Architectural College
- Boston College
- Boston University
- Brandeis University
- Bunker Hill Community College
- Cambridge College
- Curry College
- Eastern Nazarene College
- Emerson College
- Emmanuel College
- Episcopal Divinity School
- Fisher College



Curated groups are event lists approved by a moderator or generated by a machine learning model trained by a moderator

General searches use keywords, location, etc

Advances in Light Harvesting

Prof. Einstein, U. of Zurich
2 pm at City Hall Plaza on 8/8

This talk will highlight a new breakthrough in photovoltaics. Refreshments will be served.

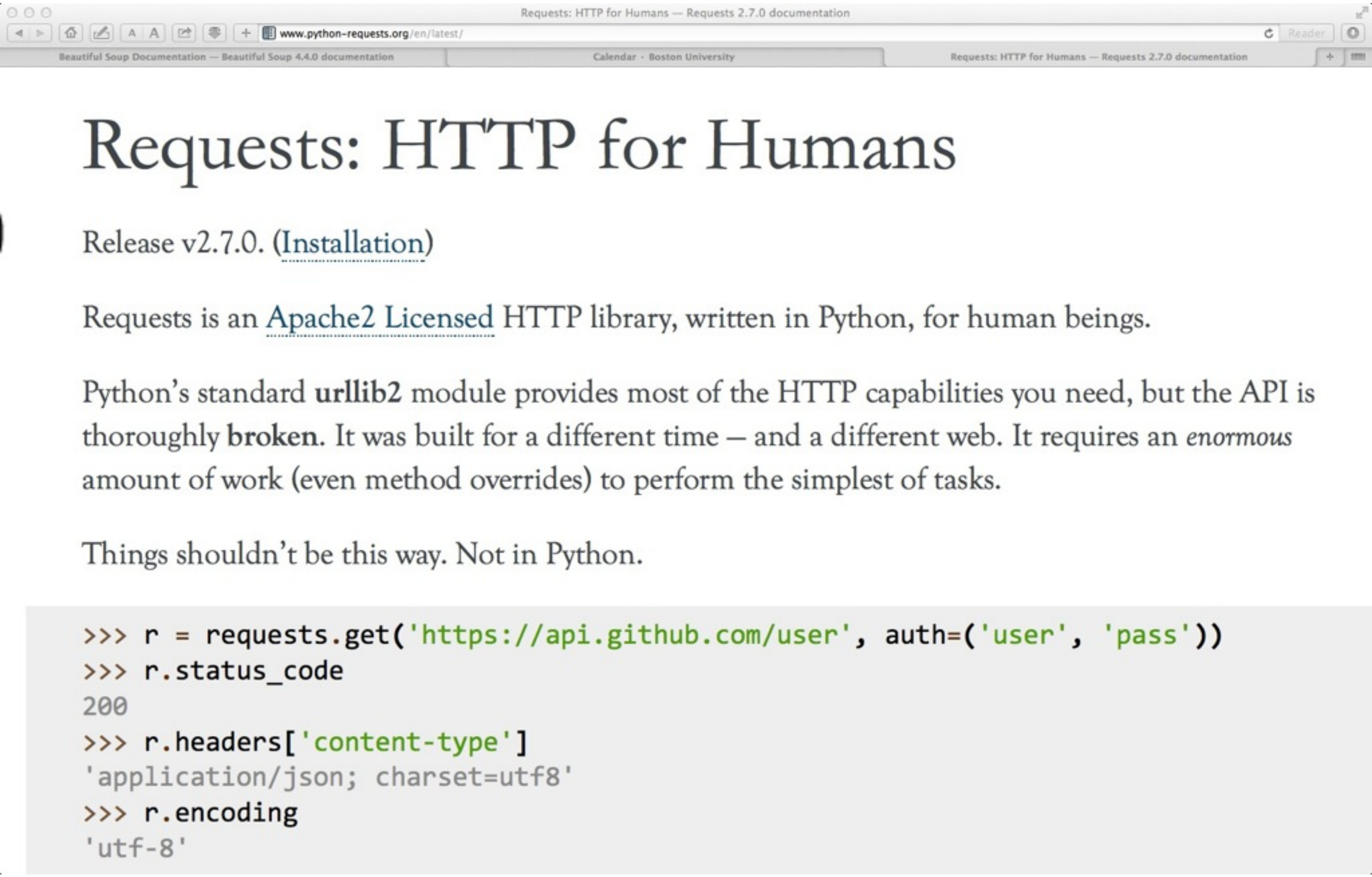
example event

Q. What to do when there's no API?
A. "*The website is the API.*"

www.mbta.com

Use `requests` to download webpages

Asheesh Laroia
Web scraping: Reliably and efficiently pull data from pages that don't expect it
https://www.youtube.com/watch?v=52wxGESwQSA


Requests: HTTP for Humans — Requests 2.7.0 documentation
www.python-requests.org/en/latest/
Reader
Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation
Calendar · Boston University
Requests: HTTP for Humans — Requests 2.7.0 documentation

Requests: HTTP for Humans

Release v2.7.0. (Installation)

Requests is an Apache2 Licensed HTTP library, written in Python, for human beings.

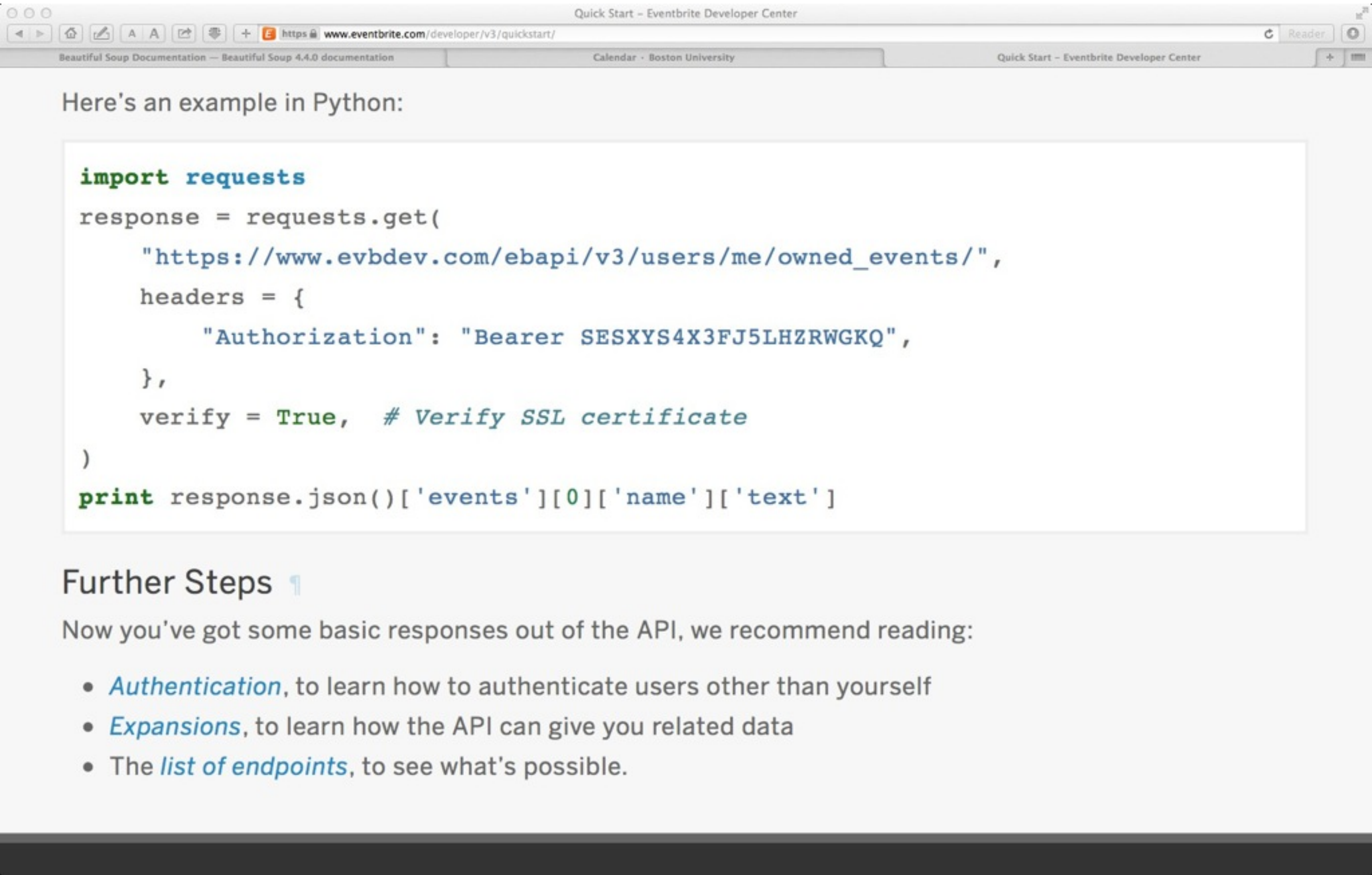
Python's standard **urllib2** module provides most of the HTTP capabilities you need, but the API is thoroughly **broken**. It was built for a different time – and a different web. It requires an *enormous* amount of work (even method overrides) to perform the simplest of tasks.

Things shouldn't be this way. Not in Python.

```
>>> 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'
```

[live demo of using requests on the mbta homepage]

EventBrite API using `requests`


Quick Start – Eventbrite Developer Center
https www.eventbrite.com/developer/v3/quickstart/
Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation
Calendar · Boston University
Quick Start – Eventbrite Developer Center

Here's an example in Python:

```
import requests
response = requests.get(
    "https://www.evbdev.com/ebapi/v3/users/me/owned_events/",
    headers = {
        "Authorization": "Bearer SESXYS4X3FJ5LHZRWGKQ",
    },
    verify = True,  # Verify SSL certificate
)
print response.json()['events'][0]['name']['text']
```

Further Steps ¶

Now you've got some basic responses out of the API, we recommend reading:

- *Authentication*, to learn how to authenticate users other than yourself
- *Expansions*, to learn how the API can give you related data
- The *list of endpoints*, to see what's possible.

mechanize

```
import re
import mechanize

br = mechanize.Browser()
br.open("http://www.example.com/")
# follow second link
response1 = br.follow_link(text_regex=r"cheese\s*shop", nr=1)
assert br.viewing_html()
print br.title()
print response1.geturl()
print response1.info()   # headers
print response1.read()   # body
```

- Easy HTML form filling.
- Convenient link parsing and following.
- Browser history (.back() and .reload() methods).
- The Referer HTTP header is added properly (optional).
- Automatic observance of **robots.txt**.
- Automatic handling of HTTP-Equiv and Refresh.

Terms of Use

“As always, be sure to review the site’s terms of use/service and respect the robots.txt file before starting any scraping job.

Make sure to adhere to ethical scraping practices by not flooding the site with numerous requests over a short span of time. Treat any site you scrape as if it were your own.”

www.mbta.com/robots.txt

www.eventbrite.com/robots.txt

scrapy

```
import scrapy

class BlogSpider(scrapy.Spider):
    name = 'blogspider'
    start_urls = ['http://blog.scrapinghub.com']

    def parse(self, response):
        for url in response.css('ul li a::attr("href")').re(r'.*/\d\d\d\d/\d\d/$'):
            yield scrapy.Request(response.urljoin(url), self.parse_titles)

    def parse_titles(self, response):
        for post_title in response.css('div.entries > ul > li a::text').extract():
            yield {'title': post_title}

$ scrapy runspider myspider.py
```

- Good for scraping entire websites

Beautiful Soup

- Beautiful Soup is a Python library for pulling data out of HTML and XML files (even if invalid tags)

```
soup = BeautifulSoup(some_html, some_html_parser)
```

1. HTMLParser (stdlib)
2. xml.dom.minidom (stdlib)
3. regular expressions
4. html5lib
5. lxml.html

- Beautiful Soup extracts data from HTML using a parser
- Avoid regular expressions for parsing HTML
- html5lib and lxml are superior

[live demo of writing a scraper for bu.edu/calendar]

Summary

- Use `requests` for small jobs and `scrapy` for large
- `mechanize` is good for interacting with forms
- Scripts can do whatever browsers can do
- See python `spiker-money` for scraping JavaScript-rich sites
- See `selenium`
- Look to ssh and tsocks to deal with per-IP address query limits
- Websites can find bots through behavior profiling and having cookies disabled