Getting data!

The problem!

Data exists in different sources and different formats and we have to work with whatever format we get

or

We perform data analysis with data in the format we have, not the format we want!

Sources

local data

csv files
pdf files
xls files

web data

json
xml
html

database servers

mysql
postgres
mongoDB

RESTful Web Services

REST: Representational State Transfer

"A network of web pages connected through links and HTTP commands (GET, POST, etc.)"

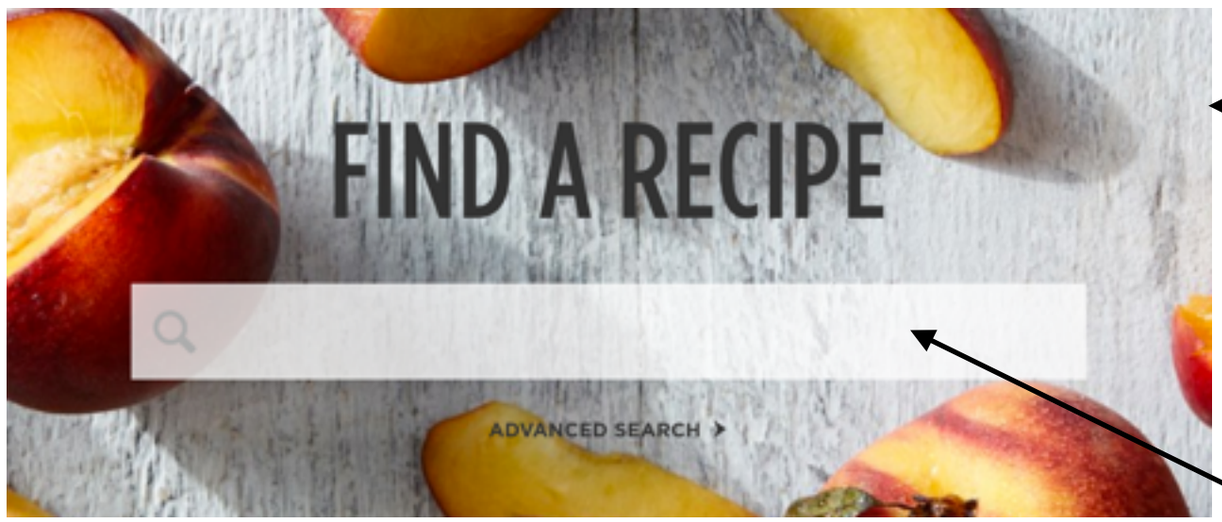
RESTful: A web service that conforms to the REST standards

RESTful Web Services

Increasingly, web services are RESTful

* Send an HTTP request query
* Get back an HTML or JSON text object

Example: Epicurious

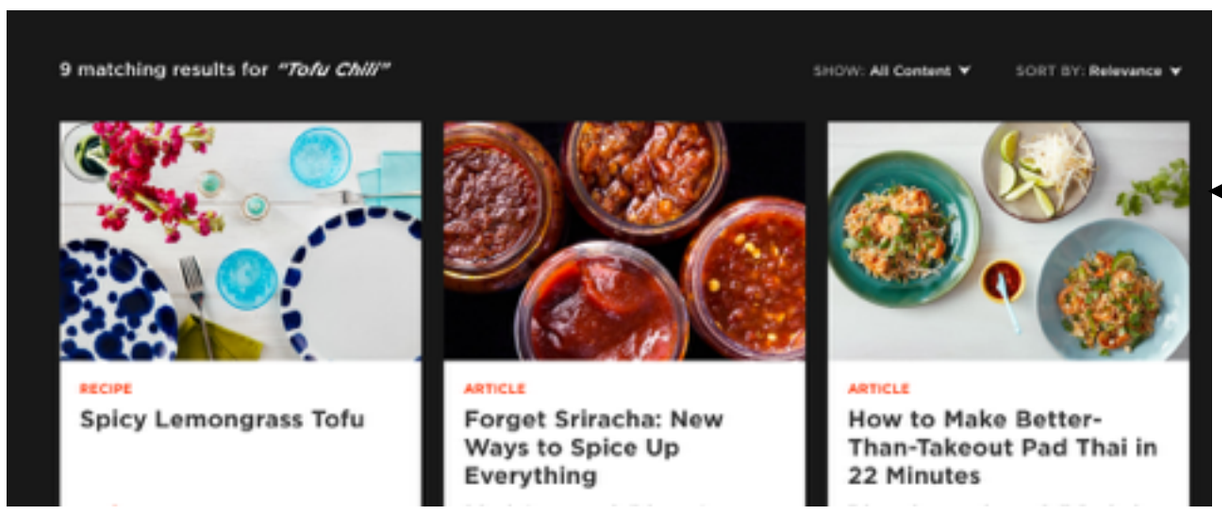


http://www.epicurious.com

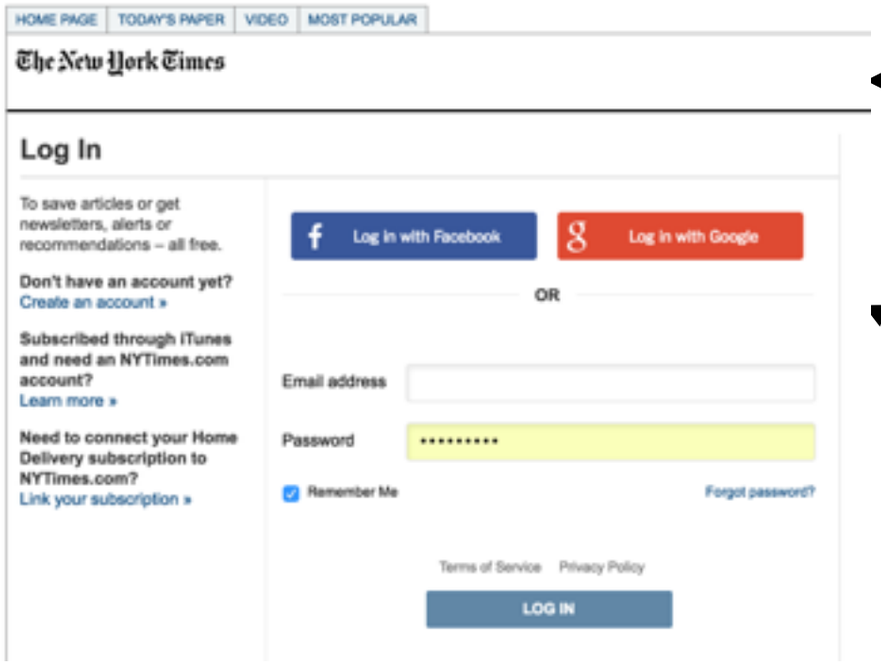
type “Tofu Chili” in search box

http GET

new url:
http://www.epicurious.com/search/Tofu%20Chili



Example: NYTIMES login



https://myaccount.nytimes.com/auth/login

type Email address and Password in the form

http POST

new url:
http://www.nytimes.com/

Example: Google GEOCODING API

HTTP GET request
with a JSON response

https://maps.googleapis.com/maps/api/geocode/json?address=Columbia_University,_New_York,_NY

All google API requests take the form:

<api_url>/<response_type>?<parameters>

json (or xml)

address=Columbia_University,_New_York,_NY

https://maps.googleapis.com/maps/api/geocode/

RESTful Web Services

Because RESTful Web Services are standardized, we can programmatically:

* send an http request and get an http response
* parse the response and extract data

Getting web data

* Send an http request and get an http response
  * requests
  * urllib.requests (urllib2 on python2)
* parse the response and extract data
  * json library
  * lxml library
  * BeautifulSoup, Selenium (for html data)

http requests

requests: Python library for handling http requests and responses

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

http://cn.python-requests.org/zh_CN/latest/

using requests

* Import the library

```
import requests
```

* Construct the url

```
url = "http://www.epicurious.com/search/Tofu+Chili"
```

* Send the request and get a response

```
response = requests.get(url)
```

* Check if the request was successful

```
if response.status_code == 200:
"SUCCESS"!!!!
else:
"FAILURE"!!!
```

JSON

JavaScript Object Notation

- Standard for "serializing" data objects, usually in files
- Human-readable, useful for data interchange
- Also useful for representing and storing semistructured data
- Stored as plain text (python str)

Basic JSON Constructs

JSON	Python
number	int,float
string	str
Null	None
true/false	True/False
Object	dict
Array	list

python json library

Converts a json string to an equivalent python type

str

```
import json
json_data = '[{"b": [2, 4], "c": 3.0, "a": "A"}]'
python_data = json.loads(json_data)
```

list

json.dumps(data) converts python data to json

requests and json

The response object handles json

```
address="Columbia_University,_New_York,_NY"
url="https://maps.googleapis.com/maps/api/geocode/json?address=%s" % (address)
response = requests.get(url).json()
```

response now points to a
python data object

requests and json

Let’s take a look at the JSON object returned by Google Geocoding API

Working with json

Problem 1: Write a function that takes an address as an argument and returns a (latitude, longitude) tuple

```
get_lat_lng(address_string)
```

Working with json

Problem 2: Write a function that takes a possibly incomplete address as an argument and returns a list of tuples of the form (complete address, latitude, longitude)

```
get_lat_lng(address_string)
```

xml

Extensible Markup Language

- Tree structure
- Tagged elements (nested)
- Attributes
- Text/Content (leaves of the tree)

xml: Example

```
<Bookstore>
  <Book ISBN="ISBN-13:978-1599620787" Price="15.23" Weight="1.5">
    <Title>New York Deco</Title>
    <Authors>
      <Author>
        <First_Name>Richard</First_Name>
        <Last_Name>Berenholtz</Last_Name>
      </Author>
    </Authors>
  </Book>
  <Book ISBN="ISBN-13:978-1579128562" Price="15.80">
    <Remark>
    Five Hundred Buildings of New York and over one million other books are available for Amazon Kindle.
    </Remark>
    <Title>Five Hundred Buildings of New York</Title>
    <Authors>
      <Author>
        <First_Name>Bill</First_Name>
        <Last_Name>Harris</Last_Name>
      </Author>
      <Author>
        <First_Name>Jorg</First_Name>
        <Last_Name>Brockmann</Last_Name>
      </Author>
    </Authors>
  </Book>
</Bookstore>
```

lxml: Python xml library

xml tree object definition

```
from lxml import etree
root = etree.XML(data)
print(root.tag)
```

prints the root tag (Bookstore in our example)

http://lxml.de/1.3/tutorial.html

lxml: Python xml library
Examining the tree

```
print(etree.tostring(root, pretty_print=True).decode("utf-8"))
```

lxml: Iterating over children of a tag

root is a collection of children so we can iterate over it

```
for child in root:
    print(child.tag)
```

lxml: Iterating over elements

```
for element in root.iter():
    print(element.tag)
```

iter is an 'iterator'. it generates a sequence of elements in the order they appear in the xml code

lxml: Iterating over elements

iterates over the tree matching only those elements that have "Author" as the tag

```
for element in root.iter("Author"):
    print(element.find('First_Name').text,element.find('Last_Name').text)
```

returns the first element that has "First_Name" as the tag

lxml: using XPath

XPath: expression for navigating through an xml tree

```
for element in root.findall('Book/Title'):
    print(element.text)
```

lxml: Finding by attribute value

```
root.find('Book[@Weight="1.5"]/Authors/Author/First_Name').text
```

Problem

Print first and last names of all authors of the book with ISBN=ISBN-13:978-1579128562

HTML

HyperText Markup Language

- Formats text
- Tagged elements (nested)
- Attributes
- Derived from SGML
- Closely related to XML
- Can contain runnable scripts

HTML/CSS

Study this on your own!