Intro to Web Scraping

Players Meeting

Brendan Houng & Prabesh KC

2019/08/08

Overview

- Why Web Scraping
- Parsing HTML
- Downloading Web Pages
- HTTP
- REST APIs
- Examples

  a) AFL

  b) Reddit

  c) Fitness Passport

Why Learn Web Scraping?

- access to data
- useful for understanding the web, including for using APIs and building websites / dashboards
- have some ready to use examples

amazon
UBER
Microsoft
Google
f
TESLA
David Parkins

Parsing HTML (1)

Web pages consist of HTML, CSS and Javascript

- Hypertext Markup Language is essentially a series of tags. Roughly similar to latex, xml, json.

```
 <p> Some paragraph of text </p>

opening tag: <p>

    content between tags: text, numbers, images

closing tag: </p>
```

- Cascading Style Sheets to customise presentation of web page
- Javascript allows for interactivity

Parsing HTML (2)

- A typical HTML page structure

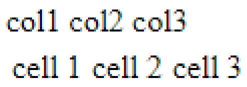
```
<html>

<head>
    <title> title at top of browser  </title>
</head>
<body>
   <h1> Title </h1>

   <p> some paragraph of text  </p>

</body>
</html>
```

```
Common tags:
- list items  <ul> and <li>
- links <a href="someurl" > click me </a>
- font style bold <b>, <i>
- divs <div> and ids <id> for layouts. e.g 2 columns, menu panels.
attributes <div="a" class="b">
```

Parsing HTML (3)

Assume that most data we are interested in will be available in table format

```
<table>
   <thead>
     <tr>
         <th> col1 </th>
         <th> col2 <th>
         <th> col3 </th>
     </tr>
   </thead>
   <tbody>
      <tr>
        <td> cell 1 </td>
        <td> cell 2 </td>
        <td> cell 3 </td>
      </tr>
   </tbody>
 </table>
```

Code results in this exciting table: 

col1 col2 col3
cell 1 cell 2 cell 3

Downloading Web Pages

wget and curl

wget

```
$ wget http://theage.com.au --no-check-certificate
```

recursion, follow links

```
$ wget -r
```

no parent

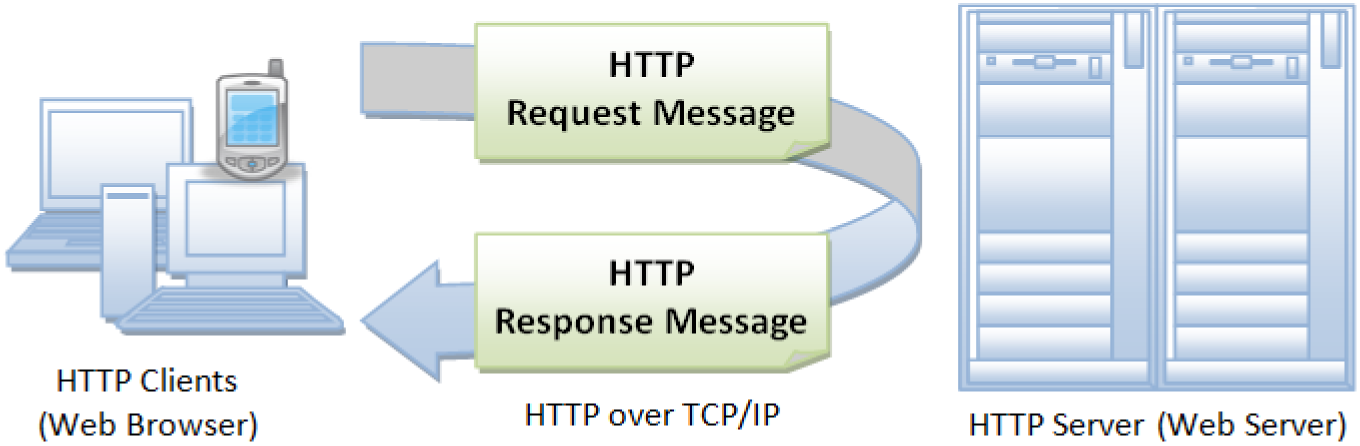
```
$ wget -r --no-parent
```

wait between retrieval, limit rate

```
$ wget -r --no-parent -w 10 --limit-rate=20k
```

Reference

HTTP (1)


HTTP
Request Message
HTTP
Response Message
HTTP Clients
(Web Browser)
HTTP over TCP/IP
HTTP Server (Web Server)

HTTP (2), requests

- Uniform Resource Locator (URL):

```
protocol://hostname:port/path-and-file-name
```

- Request - browser translates URL into protocol, sending request to server:

```
GET /docs/index.html HTTP/1.1
Host: www.nowhere123.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

HTTP (3), responses

- Response - what is returned from the server

```
HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 08:56:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
...
Accept-Ranges: bytes
Content-Length: 44
Content-Type: text/html

<html><body><h1>It works!</h1></body></html>
```

Reference

Restful APIs (1)

Representational State Transfer architectural style

Application Program Interface

Standard for how a web server responds in relation to requests (HTTP methodologies defined in RFC 2616 Protocol)

HTTP and Restful APIs are stateless, meaning that responses do not track state

4 types of requests, 2 of which you need to know:

- GET for retrieving pages. most pages. e.g. return of a list of items
- POST for submitting data, creating a resource through a HTML form. Creating an account, entering details

Restful APIs (2)

A useful abstraction of the 4 types of requests is:

Create -> Post

Read -> Get

Update -> Put

Delete -> Delete

Resulting in another acronym for the common CRUD web apps.

The alternative protocol is Simple Object Access Protocol (SOAP), which strings messages through a sequence of steps (e.g. nodes for addressing, security, format independence)

Reference

Practical Examples

1. AFL scraping (Python/BeautifulSoup)
2. Reddit (R/rvest)
3. fitness passport

See Jupyter Notebooks/Rmarkdown

Other options:

Python Scrapy

Selenium for handling javascript

Sikuli