

크롤링 차근차근 시작하기

HTTP(s) 요청 및 응답

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

우리는 크롤링을 하고자 합니다.

- 크롤링할 대상들은 웹 WWW 에 존재
 - 보다 나은 크롤링을 위해, 웹 WWW에 대한 이해가 필수
 - HTTP(S) 프로토콜 : Client/Server 구조
 - 다양한 포맷의 응답
 - ex) HTML, CSS, JavaScript, Mixed HTML/CSS/JavaScript, Image, PDF 등
- HTTP Client/Server
 - Client : HTTP 클라이언트 역할을 하는 프로그램
 - ex) 대개의 웹브라우저, curl, wget, requests 라이브러리 등
 - Server : HTTP 서버 역할을 하는 프로그램
 - ex) Apache 웹서버, IIS 웹서버, Spring 웹서버, Django 웹서버 등등

HTTP 프로토콜

- 어떤 포맷의 데이터라도 전송할 수 있도록 설계
 - HTML, CSS, JavaScript, Image, Video 등
- URI를 통해 자원의 위치 지정
 - `https://www.askcompany.kr/r/django/`
 - `https` → 프로토콜 (혹은 `http`)
 - `www.askcompany.kr` → 서버 도메인
 - `/r/django/` → 요청할 자원의 이름
 - 현재 서버 및 다른 서버의 자원도 지정 가능

웹 개발에서의 주요 개발 언어

- 웹 백엔드
 - 어떤 언어/프레임워크를 사용하든지 상관이 없습니다.
 - HTTP 요청에 맞춰 적절히 응답만 할 수 있다면, OK
- 웹 프론트엔드
 - 웹브라우저는 웹서버로 요청을 보내고, 웹서버는 상황에 따라 HTML/CSS/JavaScript 및 Image, Text, PDF, Excel 등의 응답
 - 웹브라우저는 HTML코드/CSS코드/JavaScript코드를 처리하여, 그래픽으로 보여주는 능력이 있습니다.
 - 대개 웹 프론트엔드 개발이라 한다면, HTML/CSS/JavaScript를 조합한 개발을 뜻하며, 최근에는 JavaScript의 비중이 급격히 높아졌습니다.

웹프론트엔드를 크게 3가지 성격으로 분류

- 웹 페이지
 - HTML/CSS/JavaScript 중에 HTML 비중이 크며, JavaScript는 거의 X
- JavaScript가 조금 가미된 웹 페이지
 - HTML/CSS/JavaScript 중에 HTML 비중이 크지만, 어느 정도 JavaScript를 사용
- 웹 애플리케이션
 - 사용자와 UI Interaction
 - HTML/CSS/JavaScript 중에 JavaScript의 비중이 크며, 주요 콘텐츠 데이터들을 JavaScript를 통해 서버에서 받아오며, 표현

HTTP 요청을 하는 다양한 방법

- 웹브라우저 접속
- 새로그침
- 자바스크립트를 통한 요청 (ex: Ajax)
- 앱 API : 앱 http 클라이언트를 통한 요청
- HTML Form 전송

상황에 따른 2가지 처리 방법

- 리소스에 직접 HTTP 요청
 - 응답 HTML에 찾고자 하는 콘텐츠가 있을 경우
 - 찾고자 하는 콘텐츠가 JSON이며, JSON응답을 받는 URL을 알 경우
 - requests 활용하여 처리 가능
- 리소스
 - 최초 응답 HTML에 찾고자 하는 콘텐츠가 없고, JavaScript를 통해 처리될 경우
 - selenium 활용하여 처리 가능
 - → 이 경우에도 분석에 따라, requests 활용 가능
 - 참고) <https://www.youtube.com/watch?v=dhxqn6RZkm0>











네이버 실시간 검색어 HTML 응답 문자열

```

▼<div class="ah_roll PM_CL_realtimeKeyword_rolling_base" aria-hidden="false">
  <h3 class="blind">급상승 검색어 검색어</h3>
  ▼<div class="ah_roll_area PM_CL_realtimeKeyword_rolling" queryid=
"C1551320582937536074">
    ▼<ul class="ah_l" queryid="C1551320871457550319">
      ▼<li class="ah_item">
        ▼<a href="#" class="ah_a" data-clk="lve.keyword">
          <span class="ah_r">1</span>
          <span class="ah_k">토틀넵 첼시</span> == $0
        </a>
      </li>
      ▼<li class="ah_item">
        ▼<a href="#" class="ah_a" data-clk="lve.keyword">
          <span class="ah_r">2</span>
          <span class="ah_k">시나문물</span>
        </a>
      </li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
      ▶<li class="ah_item">...</li>
    </ul>
  </div>
</div>

```

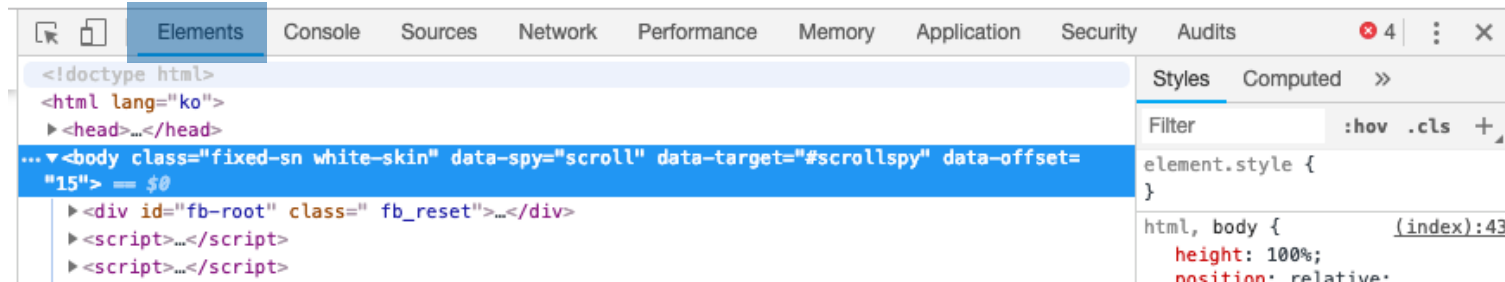
응답을
네트워크
그 즉시

급상승 검색어		DataLab. 급상승 트래킹 >
1~10위	11~20위	
1 토트넘 첼시		
2 시나몬물		
3 첼시 토트넘		
4 김예진		
5 빅히트샵		
6 김건우		
7 b1a4		
8 바르셀로나 레알마드리드		
9 항거:유관순 이야기		
10 엘클라시코		

응답을 받은 브라우저를 HTML를 처음부터 1줄씩 해석해가며, 그래픽으로 보여줍니다.
네트워크 접근이 필요한 부분 (CSS/JavaScript/Image/Video URL)을 만나면, 이에 대해
그 즉시 추가 HTTP요청

HTML 코드 확인하는 방법

- 웹 서버로부터의 초기 응답
 - 웹브라우저에서는 "페이지 소스 보기" 메뉴
 - requests 활용 시에는 `requests.get("...").content`
- 웹 페이지에서의 현재 상황
 - 브라우저의 "개발자 도구" → Elements 탭
 - JavaScript에 의해 페이지 내용이 변경되었을 수가 있습니다.
 - Selenium 혹은 requests



자주 가시는 사이트의 페이지 소스 보기

- 웹브라우저로 접속해서, 페이지 소스보기를 해보세요.
 - "페이지 소스 보기"의 내용이 서버로부터의 원본 HTML 응답입니다.
- 특정 웹사이트에서 우클릭이 안 되는 것은 자바스크립트로 우클릭을 막아놓은 것입니다.
 - 크롬 브라우저를 쓰신다면 Enable Right Click 확장을 써보세요.

<https://chrome.google.com/webstore/detail/enable-right-click/hhojmcideegachlhfgfdhailpfhgknjm?hl=ko>

다양한 HTTP 메서드

패킷을 어떻게 구성하느냐의 차이

Method	설명
GET (크롤링에서 주로 사용)	리소스 요청
POST (크롤링에서 주로 사용)	대개 리소스 추가요청이나 수정/삭제 요청으로도 사용
PUT	대개 리소스 수정 요청
DELETE	대개 리소스 삭제 요청
HEAD	HTTP 헤더 정보만 요청. 해당 자원 존재여부 확인 목적.
OPTIONS	웹서버가 지원하는 메서드 종류 반환 요청
TARCE	클라이언트의 요청을 그대로 반환.

GET/POST 만을 살펴봅니다.

- GET 요청 : "엽서"에 비유.
 - 엽서는 주소에 함께 메시지를 남깁니다.
 - 물건을 보낼 수는 없어요. (프로토콜 설계상 파일 업로드 불가능합니다.)
 - 잘 설계된 서비스에서는 주로 **조회** 요청 시에 사용
- POST 요청 : "소포"에 비유
 - 소포는 박스 안에 메시지나 물건을 같이 보낼 수 있어요.
 - 파일 업로드 지원
 - 잘 설계된 서비스에서는 주로 **추가/수정/삭제** 요청 시에 사용

HTTP 요청/응답 패킷 형식

- 요청 패킷

- 요청 헤더 : 클라이언트에서 필요한 헤더 Key/Value를 세팅한 후, 요청 전달
- 첫번째 빈 줄 : 헤더와 Body 구분자로서 활용
- Body : 클라이언트에서 필요한 Body를 세팅하여, 요청 전달
 - ex) 업로드할 파일 내용 등

- 응답 패킷

- 응답 헤더 : 서버에서 필요한 헤더 Key/Value를 세팅한 후, 응답 전달
- 첫번째 빈 줄 : 헤더와 Body 구분자로서 활용
- Body : 서버에서 필요한 Body를 세팅하여, 응답 전달
 - ex) HTML 문자열, 이미지 데이터, PDF 데이터 등

헤더란?

- HTTP 요청/응답 시에 헤더정보가 Key/Value 형식으로 세팅
- 대개의 브라우저에서는 다음 헤더를 설정
 - User-Agent : 브라우저 종류
 - Referer : 이전 페이지 URL (어떤 페이지를 거쳐서 왔는가?)
 - Accept-Language : 어떤 언어의 응답을 원하는가?
 - Authorization : 인증정보
- 크롤링 시에는 User-Agent 헤더와 Referer를 커스텀하게 설정할 필요가 있습니다.
 - 서비스에 따라 User-Agent헤더와 Referer헤더를 통해, 응답을 거부하기도 합니다. (ex: 네이버웹툰)

httpie를 통해, 응답 헤더를 직접 봅시다.

- 설치 프로그램
 - pip install httpie
- 명령 프롬프트에서 실행할 명령
 - ex) http --headers https://www.naver.com

```
▶ http --headers https://www.naver.com
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, must-revalidate
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Thu, 28 Feb 2019 03:03:50 GMT
P3P: CP="CAO DSP CURa ADMa TAIa PSAa OUR LAW STP PHY ONL UNI PUR FIN COM NAV INT DEM STA PRE"
Pragma: no-cache
Referrer-Policy: unsafe-url
Server: NWS
Set-Cookie: PM_CK_loc=dfd34323664c3c627d5d9701ad0d24376d8b4cd6d2fee0e1ea794dad54f58cb2; Expires=Fri, 01 Mar 2019 03:03:50 GMT; Path=/; HttpOnly
Strict-Transport-Security: max-age=63072000; includeSubdomains
Transfer-Encoding: chunked
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
```

웹 요청/응답 시에는 이와 같은 프로토콜의 문자열을 서로 주고 받아야 합니다.

- 이를 쉽게 수행할 수 있도록 도와주는 것
 - → 웹 클라이언트 라이브러리, 웹 브라우저
- requests
 - 첫 응답만 받음. 추가 요청이 없음.
 - 단순한 요청에 최적화.
 - HTML응답을 받더라도, 이에 명시된 이미지/CSS/JavaScript 추가 다운을 수행 X => 직접 다운로드 요청은 가능
- selenium : 웹브라우저 자동화 툴
 - javascript/css 지원, 기존 GUI 브라우저 자동화 라이브러리
 - 사람이 웹서핑하는 것과 동일한 환경, 대신에 리소스를 많이 먹음
 - 웹브라우저에서 HTML에 명시된 이미지/CSS/JavaScript를 모두 자동 다운로드/적용

쓸 수만 있다면 requests를 쓰는 것이 ...

- 적은 리소스, 유연한 구동환경, 빠른 처리.
- 아래와 같은 응답을 받은 경우
 - requests는 서버에 추가요청을 하지 않습니다.
 - selenium은 기존 브라우저를 활용하기 때문에, 최소한 CSS 1개, JavaScript 2개, 이미지 1개에 대해 추가로 요청을 합니다.

```
<!doctype html>
<html>
  <head>
    <title>Ask Company VOD</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" />
    <script src="https://code.jquery.com/jquery-2.2.4.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
  </head>
  <body>
    
  </body>
</html>
```

Selenium은 언제?

- 생각해보기
 - 로그인 처리가 필요한가요?
 - javascript 처리가 필요한가요?
 - 브라우저에 보면 데이터가 있는 데, requests로 응답을 받아보면 데이터가 없나요?
- 이도저도 고민하기 싫다면, selenium을 쓰시면 됩니다.
 - 단 리소스를 많이 먹고, requests보다 처리 로직이 복잡할 수도 있습니다.

requests 활용 실전코드 (1)

받은 파일을 직접 열어보세요. 어떤가요?

소스코드 편집기로 열어보세요.

```
import os
import requests

# URL 소스 : https://comic.naver.com/webtoon/detail.nhn?titleId=20853&no=1164&weekday=tue
# - 마음의 소리 : 1160. 그 인형

image_url = ('https://image-comic.pstatic.net/webtoon/20853/1164/2019030416'
             '5422_16ead90819a164bac058e845343ee514_IMAG01_1.jpg')

response = requests.get(image_url)
image_data = response.content # 응답 원본 데이터 (bytes 타입)
filename = os.path.basename(image_url)

with open(filename, 'wb') as f:
    print('writing to {} ({} bytes)'.format(filename, len(image_data)))
    f.write(image_data)
```

<https://gist.github.com/allieus/4df42cc7c569cb248d81f82872278710> 의 Case 1 코드

다운받은 파일을 실제로 열어보면 ...

writing to 20190304165422_16ead90819a164bac058e845343ee514_IMAGE01_1.jpg (3342 bytes)

이미지인데, 용량이 3KB ???

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<style>
a:link      {font:8pt/11pt verdana; color:red}
a:visited   {font:8pt/11pt verdana; color:#4e4e4e}
</style>
<title>HTTP 403 Forbidden</title>
</head>
<script>
function Homepage(){
// in real bits, urls get returned to our script like this:
// res://shdocvw.dll/http_404.htm#http://www.DocURL.com/bar.htm
```

응답의 정체는 ...
Image가 아니라, HTML 문자열.

요청을 한다고 해서
무조건 응답을 하는 것은 아닙니다.

서버가 어떻게 대응하고 있는 지에 대해서는
예측하여 테스트해봐야하며,
서버측 대응은 언제라도 변경될 수 있습니다.

requests 활용 실전코드 (2) - 대응하기

Case by case

```
import os
import requests

# URL 소스 : https://comic.naver.com/webtoon/detail.nhn?titleId=20853&no=1164&weekday=tue
# - 마음의 소리 : 1160. 그 인형

image_url = ('https://image-comic.pstatic.net/webtoon/20853/1164/2019030416'
             '5422_16ead90819a164bac058e845343ee514_IMAG01_1.jpg')

episode_url = 'https://comic.naver.com/webtoon/detail.nhn?titleId=20853&no=1164&weekday=tue'
headers = {'Referer': episode_url}

response = requests.get(image_url, headers=headers)
image_data = response.content # 응답 원본 데이터 (bytes 타입)
filename = os.path.basename(image_url)

with open(filename, 'wb') as f:
    print('writing to {} ({} bytes)'.format(filename, len(image_data)))
    f.write(image_data)
```

```
writing to 20190304165422_16ead90819a164bac058e845343ee514_IMAG01_1.jpg (150285 bytes)
```

<https://gist.github.com/allieus/4df42cc7c569cb248d81f82872278710> 의 Case 2 코드

요약

- 크롤링할 콘텐츠를 다양한 방법(requests, selenium등)으로 요청하여, 그 콘텐츠가 포함된 문자열 뭉치를 획득
- 그 문자열 뭉치가 ...
 - HTML 안에 포함이 되어있다면, HTML Parser (BeautifulSoup4 등) 로 쉽게 처리 가능
 - HTML 이외 영역에 있다면, 정규표현식 등의 방법으로 추출 가능

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

- Ask Company