

감히 이런 제목을...?

Everything of Crawling for Big Data Analysis using Python

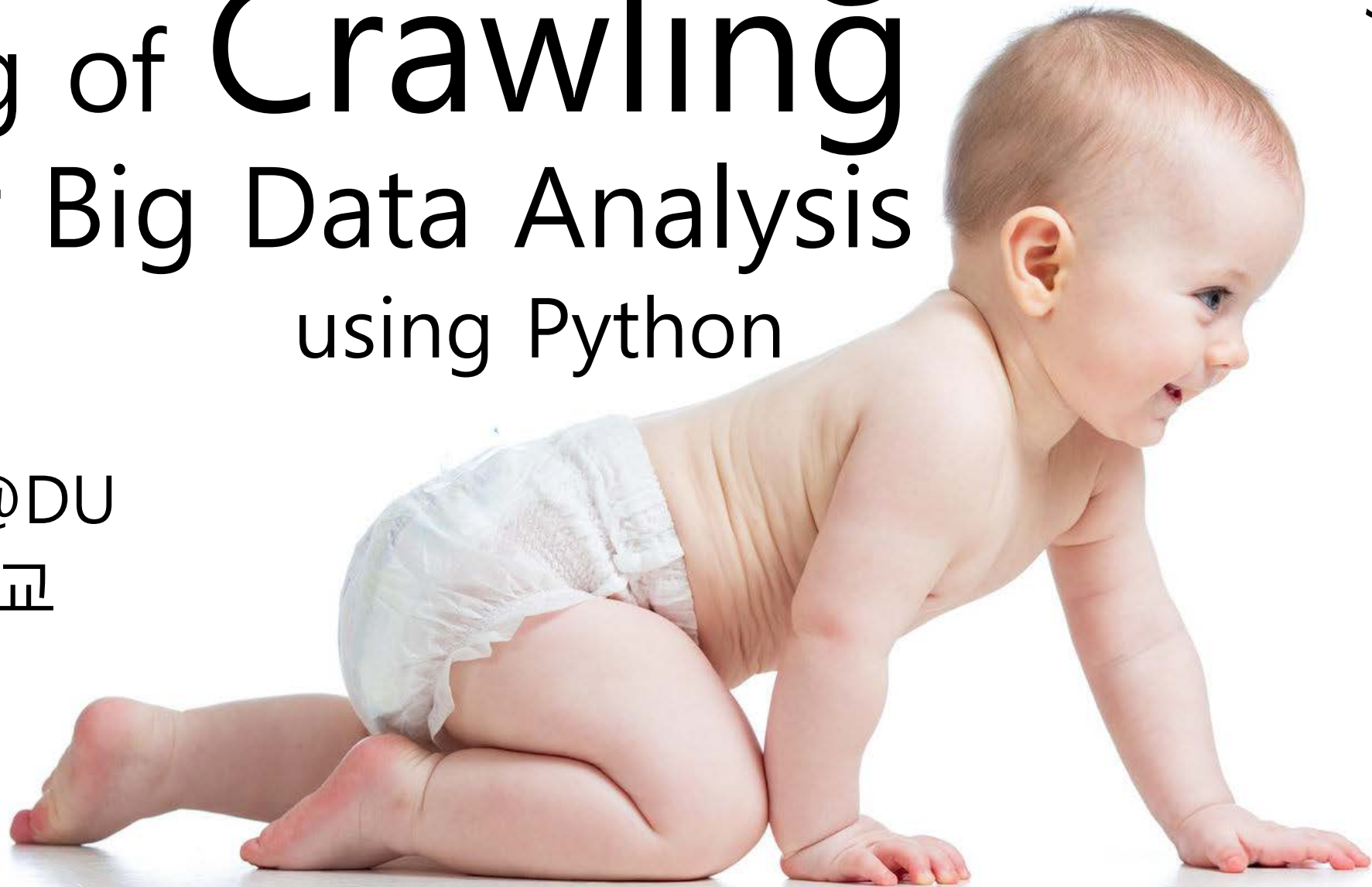
사실 Scraping

20180701 Sun @DU

KISTI / 한양대학교

김영진 金英珍

Young Jin Kim





Crawling?

http://www

Uniform **R**esource **L**ocator

World **W**ide **W**eb Project(1991)

Hyper **T**ext **T**ransfer **P**rotocol

그 외에 보안을 강화한 HTTPS, 파일전송을 위한 FTP 등

http://www

scheme://[user:password@]host[:port]][/]path[?query][#fragment]

예1) <http://www.somehost.com/a.gif>

- IP 혹은 Domain name 정보가 필요한 형태 (www.somehost.com에 있는 a.gif를 가리키고 있음)

예2) <ftp://id:pass@192.168.1.234/a.gif>

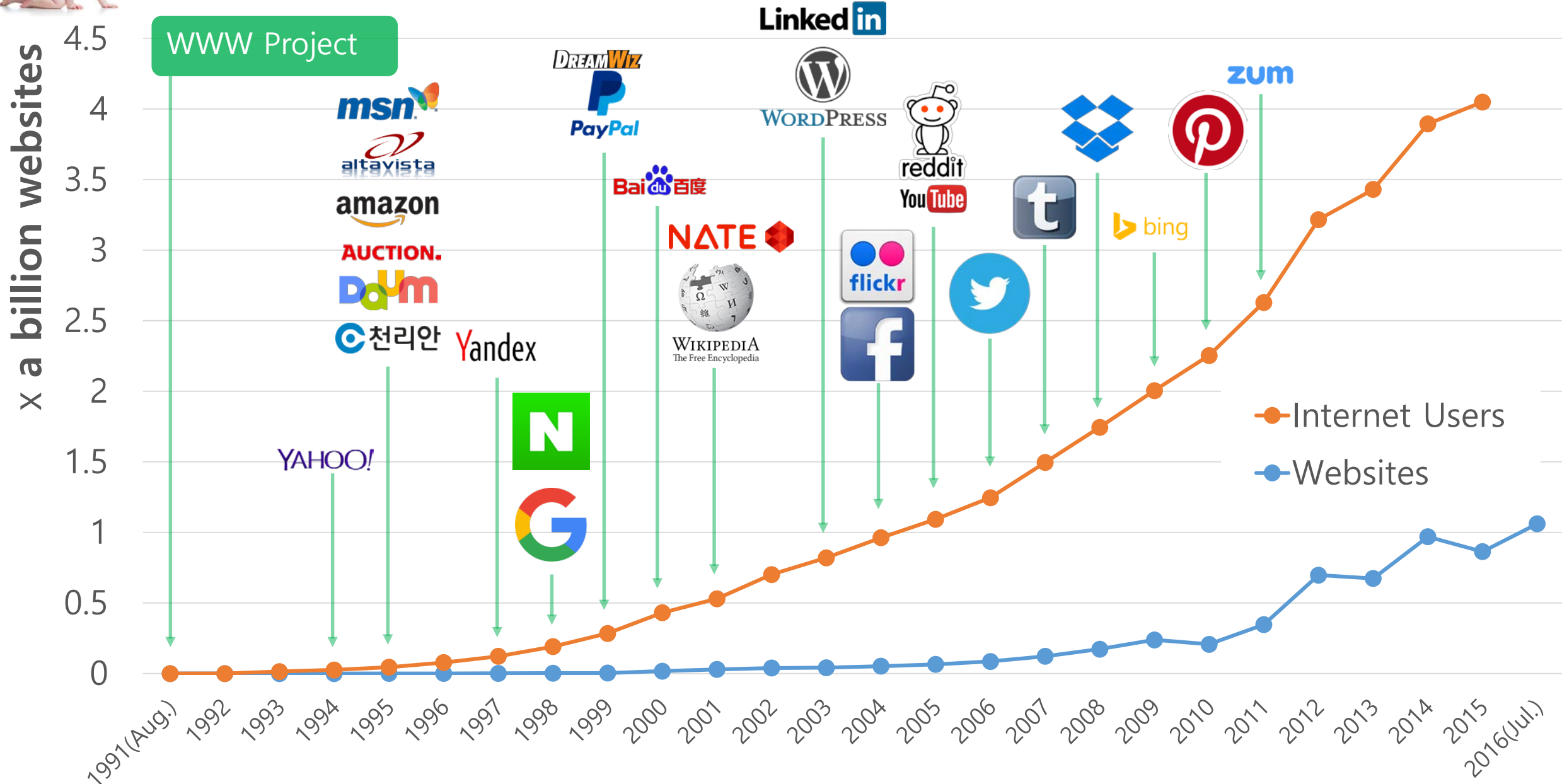
- IP 혹은 Domain name 정보가 필요한 형태 (192.168.1.234에 있는 a.gif를 가리키고 있음)

예3) <mailto:somebody@mail.somehost.com>

- IP정보가 필요없는 프로토콜 (mailto 프로토콜은 단지 메일을 받는 사람의 주소를 나타냄)



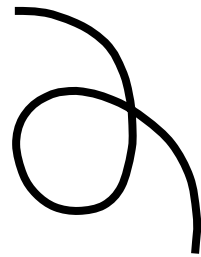
웹페이지는 셀 수 없을 정도로 많다.



Markup Language(HTML 등)으로 쓰여진 웹 페이지를
웹 브라우저를 통해서 읽는다.

URL : www.example.net

Webpage



Web browser



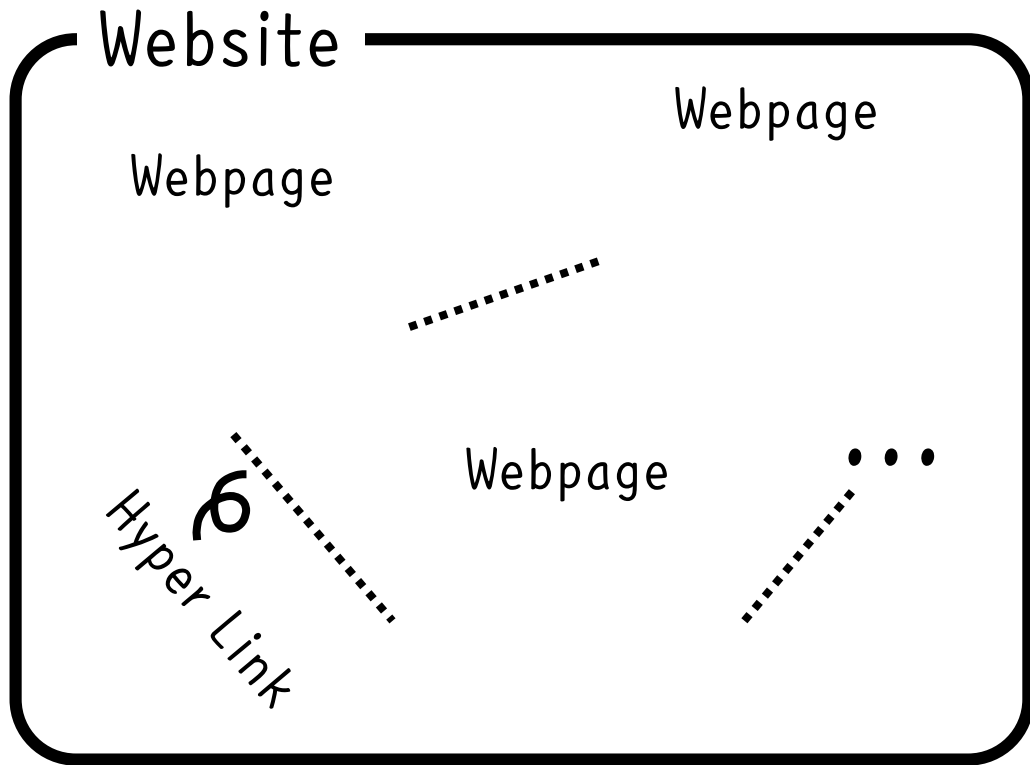
Markup Language

```
<!DOCTYPE html>  
<html>  
<!-- created 2010-01-01 -->  
<head>  
  <title>sample</title>  
</head>  
<body>  
  <p>Voluptatem accusantium  
  totam rem aperiam.</p>  
</body>  
</html>
```

HTML

WWW > Internet > Website > Webpage

이러한 웹 페이지들은 서로 하이퍼링크를 통해 연결되어 있다.
이런 웹페이지들의 의미 있는 묶음을 웹 사이트라고 한다.



=

Website

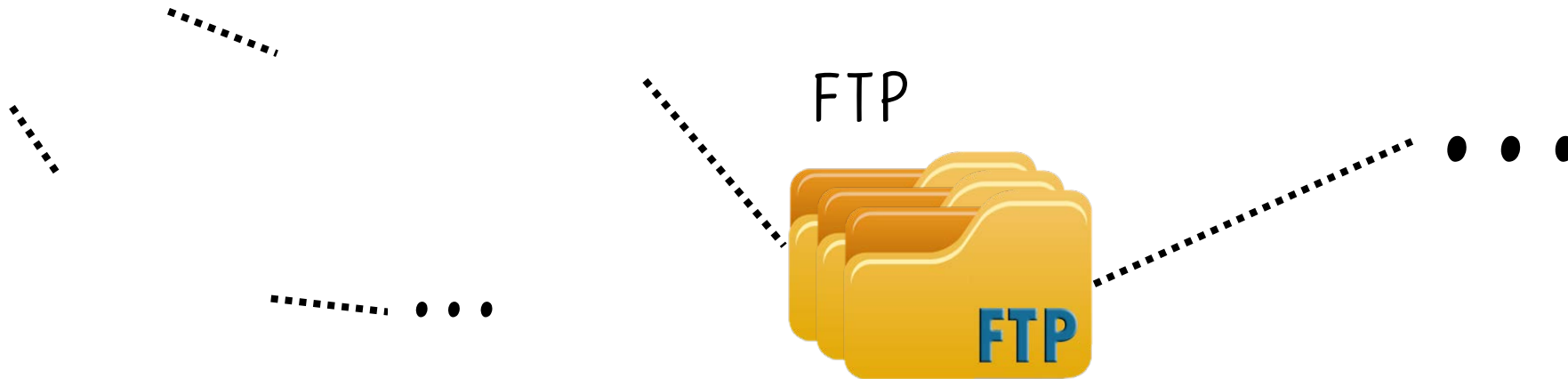
World Wide Web

Intranet

이런 웹 사이트들 또한 서로 연결 되어있고, 다른 프로토콜과도 이어져 있을 수 있다.
여기에 독립적인 Intranet이 따로 존재해 있고, 이를 모두 묶어서 WWW이라 부른다.

Internet via HTTP and etc

Website



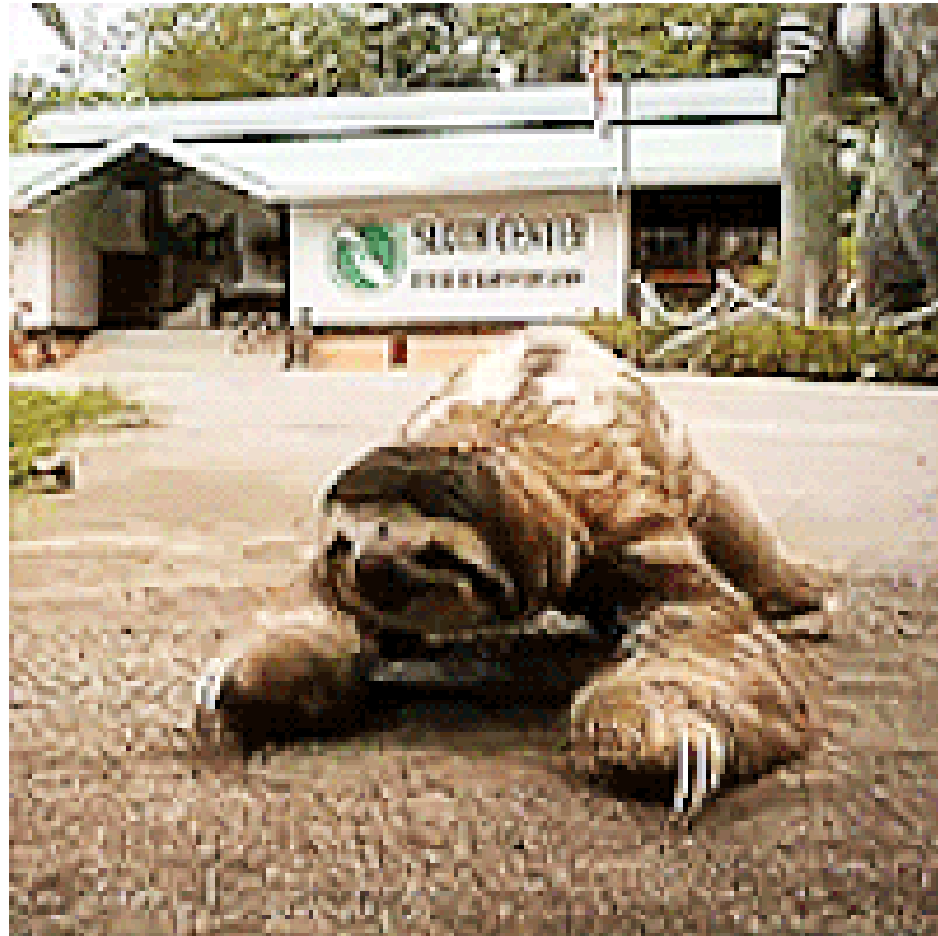
Internet : 여러가지 networks의 통합을 뜻하는 Inter + Network의 준말이다. 초창기엔 FTP, 최근엔 HTTP가 가장 대중적으로 쓰이고 있고, 보안을 강화한 HTTPS도 사용 중이다. 이 외에도 Telnet, POP3, SSH 등 다양한 프로토콜이 존재한다. 군대, 각 기업 내에서는 Intranet도 사용한다.

WWW : World Wide Web의 준말, URL로 주어진 web 문서의 주소가 Hyper text (HTML; Hypertext Markup Language)에 의해 연결되고 Internet 등을 통해 접근할 수 있는 정보 공간을 총칭해서 뜻한다. 줄여서 보통 web이라 부른다.

URL : Uniform Resource Locator의 약어로, 웹사이트 주소 (WWW 프로토콜에 해당하는 주소) 뿐 아니라, computer network 상의 모든 주소를 뜻한다. 원래는 IP를 포함한 숫자 주소로 나타내었으나, 이는 너무 복잡하여 문자를 사용하여 간결히 나타낸 도메인(domain)을 사용하는 것이 대중화 되었다. 우리가 흔히 말하는 URL은 도메인을 포함한 웹 페이지 주소를 뜻한다.

Website : 일정한 webpages의 의미 있는 묶음을 말한다. 좀 더 기술적으로는 하나의 web server 상에 hosting 되는 문서를 뜻한다. 한 website 안에는 여러 개의 webpages이 존재할 수 있다. 이들은 하이퍼 링크를 통해서 서로 다른 웹페이지들로 이동할 수 있다.

Crawling? 기어가기??

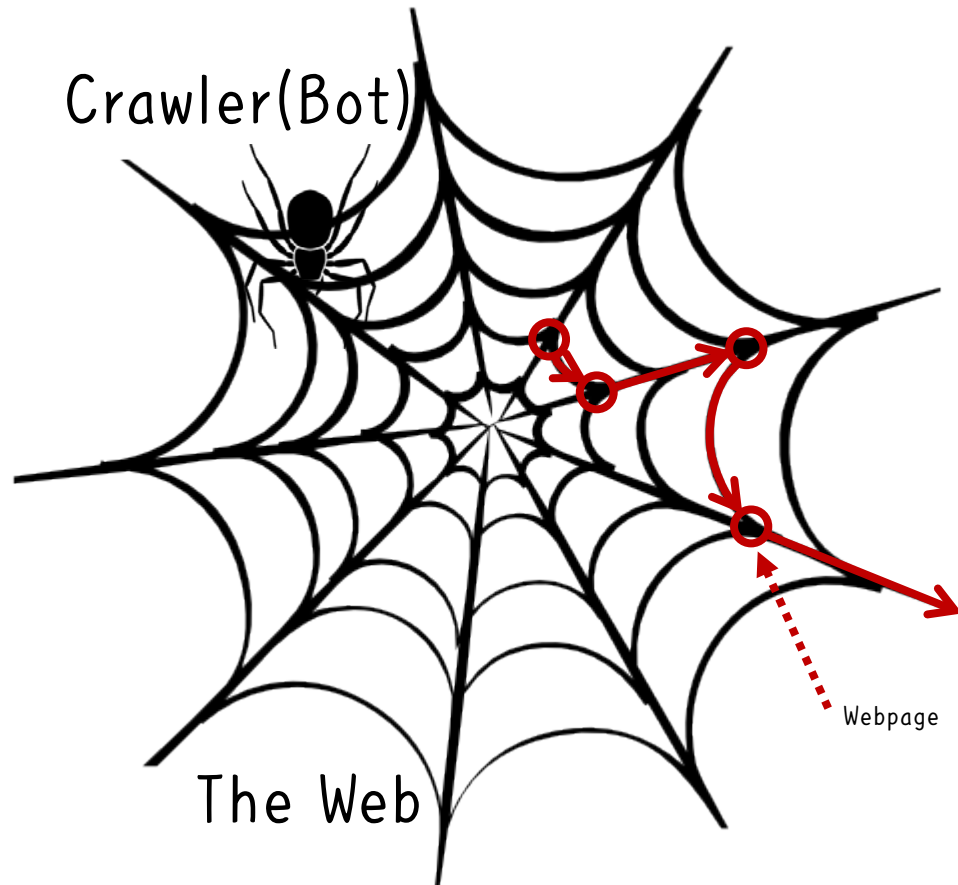


1.1 웹 크롤링이란?

우리가 하고 싶은 건 Web crawling Web scraping

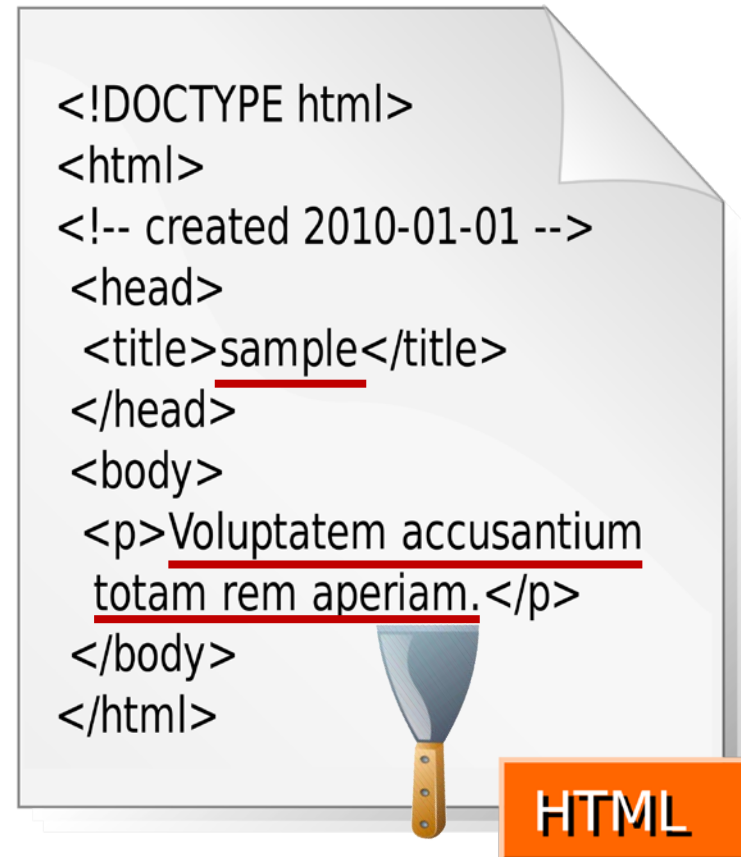
Web crawling?

Bot이 Web을 Link를 통해 돌아다니는 것



Web scraping!

Webpage에서 원하는 자료를 긁어오는 것



1.1 웹 크롤링이란?

우리가 하고 싶은 건 Web crawling Web Scraping

URL을 통해 웹 페이지에 접속하여 HTML 문서에서 정보 읽어 오기

To developer

Web Indexing (by Web Crawler)

구글과 네이버의 차이는 대체 어디서 나는 걸까? robots.txt은 뭘까?

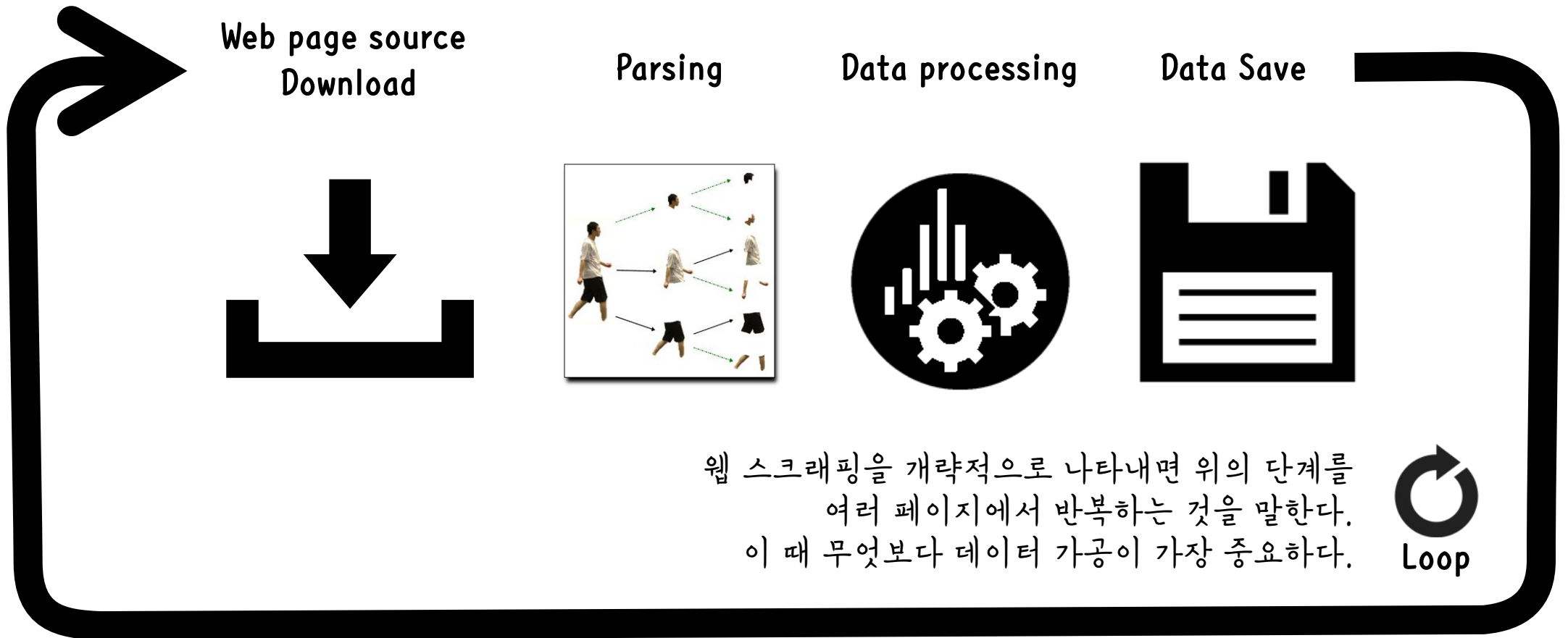
흔히 말하는 검색엔진의 성능을 담당하는 algorithm의 한 부분을 뜻한다. 각 검색엔진은 성능이 모두 다르다. 각 엔진에서 사용하는 algorithm의 성능 차이 때문, 이 차이는 여러 webpages 중 중요한 pages을 걸러내는 방식에서 발생한다. 중복되는 pages, 불필요한 pages 등을 효율적으로 처리해야 한다. crawler가 web 문서에서 URL을 보고 내부 algorithm을 이용해 machine learning을 통해 학습한다. 이에 따라 미리 검색결과를 저장해 놓은 뒤 보여준다. 이것이 Web indexing이다.

To researcher

Web Scraping (using Parser)

웹페이지에서 데이터를 **노.가.다. 없.이. 얻을 수 있을까?** Web Scraping이 맞음

data가 넘쳐나는 시대, 내가 원하는 정보만, 손쉽게(?) 반복적으로 얻어오는 방법을 뜻한다. URL의 규칙성을 이용하거나, 가지고 있는 일련의 URLs를 통해 HTML 문서를 읽어와서 그 중 우리가 원하는 data를 반복적으로 추출해 오는 것을 말한다. 앞서 개발자들에게 필요한 전체 algorithm이 아닌 훨씬 기초적인 부분만 사용된다. 초기에는 Python 선두주자로서, 여러 packages(scrapy, beautifulsoup) 를 제공하였고, 이후, R, Java 등 왜만한 주요 언어에서는 scraping이 가능하게 되었다.



1.3 파싱 (parsing)

먼저 HTML이 어떻게 생겼는지 이해하자.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>제목</title>
  </head>
  <body>
    <h1>제목</h1>
    <p>단락</p>
    <div class="ex">...</div>
    <a href="URL">링크</a>
  </body>
</html>
```

1. DTD 선언; Document Type Definition

```
<!DOCTYPE html>
```

2. 태그와 요소

```
<h1>...</h1> ; h1 요소
```

; <start tag> + child node + <end tag>

3. 자주 쓰이는 요소

```
<p>단락</p> ; p(Paragraph)
```

```
<div class="ex"> ...</div>
```

;div(Division)

```
<a href="URL">링크</a> ;a(Anchor)
```

...

우리에게 이런 태그들의 의미보단, 구조의 이해가 중요하다.

<http://html5tutorial.github.io/> - HTML5 자습서

1.3 파싱 (parsing)

먼저 HTML이 어떻게 생겼는지 이해하자.

이제 실제 HTML 문서들이 어떻게 생겼는지 확인 해보자.

www.naver.com 에 접속해서 다음을 수행해보자.

- IE와 Microsoft Edge

: **F12**

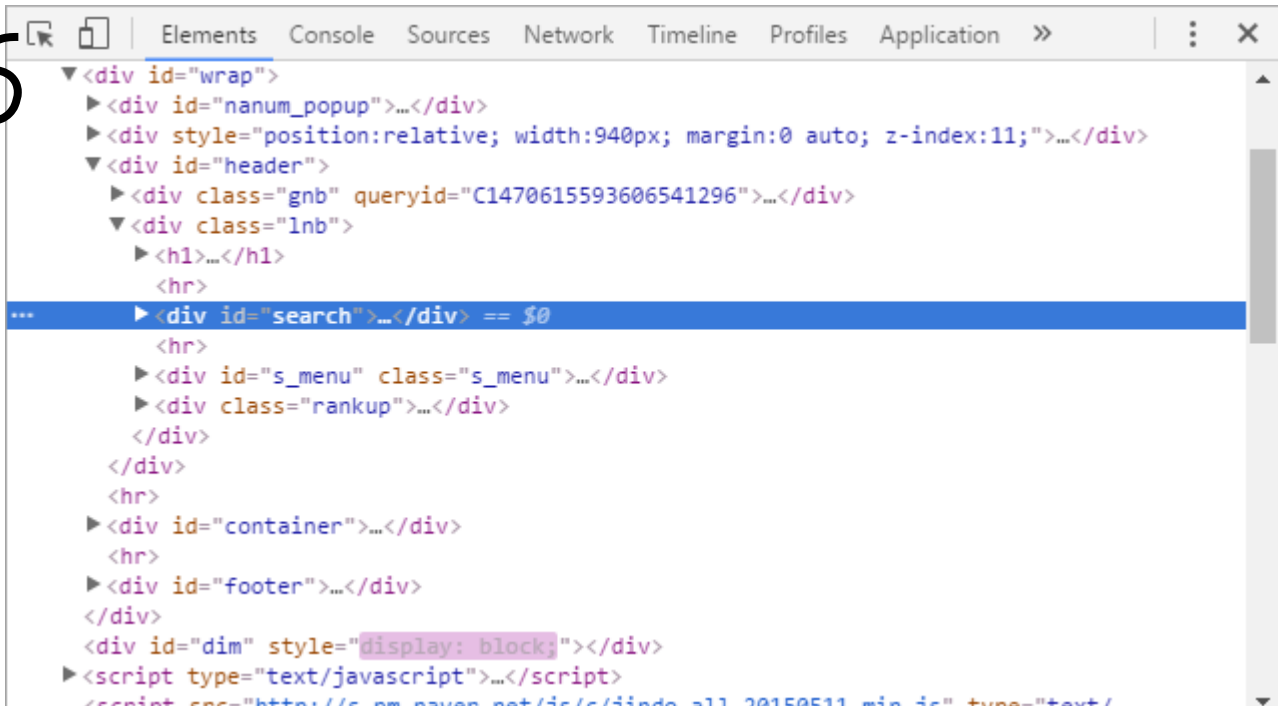
- Firefox와 Chrome

: **Ctrl** + **Shift** + **I**

- Safari

: **Cmd** + **Option** + **I**

요거 유용함!



```
<div id="wrap">
  <div id="nanum_popup">...</div>
  <div style="position:relative; width:940px; margin:0 auto; z-index:11;">...</div>
  <div id="header">
    <div class="gnb" queryid="C1470615593606541296">...</div>
    <div class="lnb">
      <h1>...</h1>
      <hr>
      <div id="search">...</div> == $0
      <hr>
      <div id="s_menu" class="s_menu">...</div>
      <div class="rankup">...</div>
    </div>
  </div>
  <hr>
  <div id="container">...</div>
  <hr>
  <div id="footer">...</div>
</div>
<div id="dim" style="display: block;"></div>
<script type="text/javascript">...</script>
<script src="http://s.nm.naver.net/js/c/findo_all_20150511_min.js" type="text/
```

1.3 파싱 (parsing)

이러한 웹 문서의 소스(HTML 등)에서 구문을 분석하는 것

Scraping에서 Parsing이란?

HTML이나 XML, JavaScript 등으로 쓰여진 소스들을 각 **요소별로 나누는 것**이 때 이러한 parsing을 해주는 것을 parser라고 부른다.

Crawling 및 Scraping을 지원하는 모든 언어는 기본적으로 이런 parser가 존재한다. parser가 꼭 필요한 것은 아니지만, parser가 있다면 훨씬 편하고 빠른 시간에 원하는 결과물을 얻을 수 있다.

Parser는 우리가 원하는 data를 tag를 통해서 찾아내고, 해당하는 내용들을 불러오는 것이다.

1.3 파싱 (parsing)

이러한 웹 문서의 소스(HTML 등)에서 구문을 분석하는 것

자, 그럼

이제 배워 봅시다.

1,4 Reference

시간 날 때 읽어보면 좋은 내가 공부했던 사이트들

<http://alwarm.tistory.com/16> - 검색엔진의 역사

<http://www.bloter.net/archives/166941> - 구글이 어떻게 웹페이지를 수집하냐면...

<http://id002.tistory.com/96> - 웹 문서 크롤링이란?

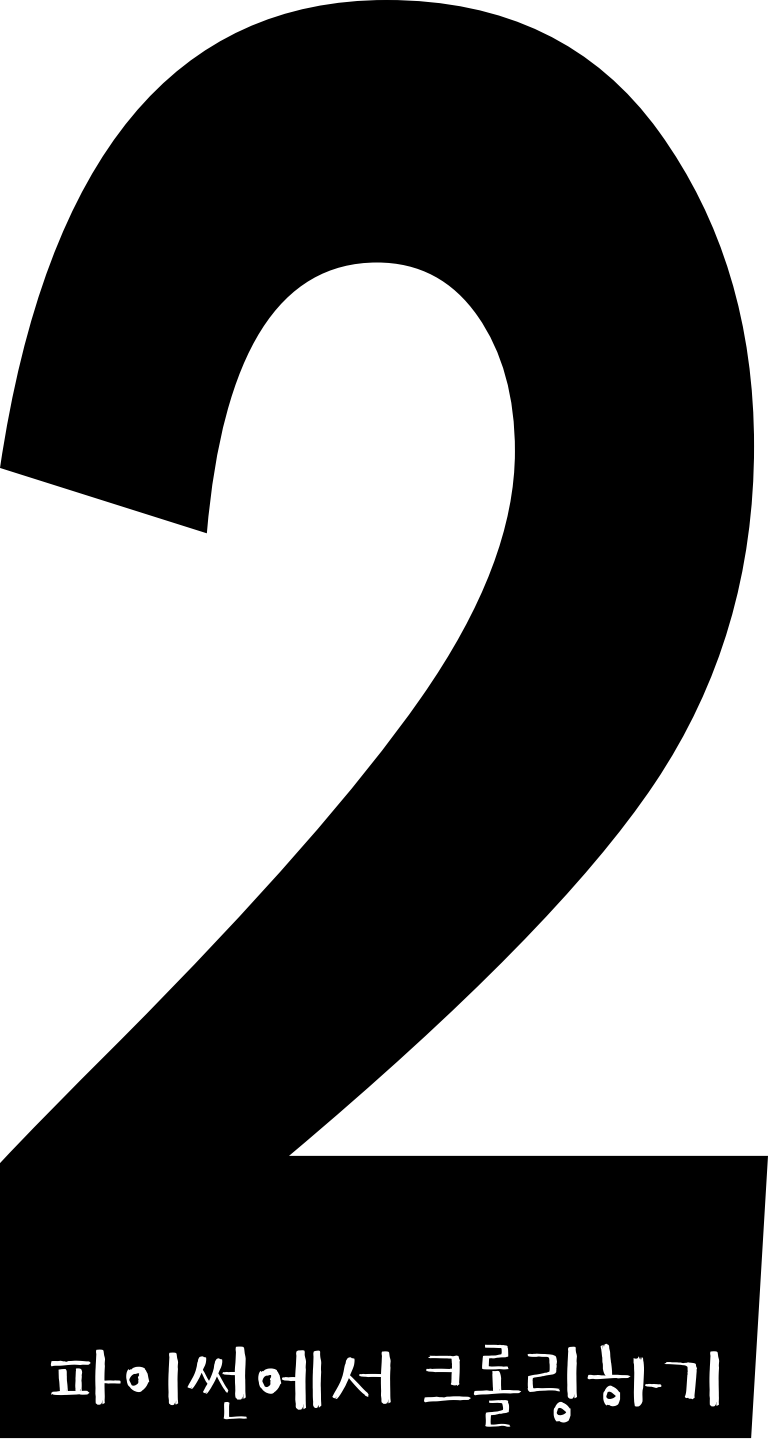
<http://lyb1495.tistory.com/104> - 웹크롤러 아키텍처

https://en.wikipedia.org/wiki/Robots_exclusion_standard - 로봇 배제 표준

https://en.wikipedia.org/wiki/Web_crawler - 웹 크롤러

<http://www.slideshare.net/lucypark/the-beginners-guide-to-54279917> - The beginner's guide to 웹 크롤링

<http://html5tutorial.github.io/> - HTML5 자습서



파이썬에서 크롤링하기



Crawling using Python

BeautifulSoup; Parser의 집합

Scraping을 위한 최소한의 기능을 모아놓았다.

쉽고 빠르다. (~~심지어 매뉴얼도 한글화 되어있다.~~)

python3을 지원하기 때문에, utf-8을 지원하고, 한글을 편하게 사용할 수 있다.

You didn't write that awful page. You're just trying to get some data out of it. Beautiful Soup is here to help. Since 2004, it's been saving programmers hours or days of work on quick-turnaround screen scraping projects.

Beautiful Soup

"A tremendous boon." -- Python411 Podcast

[[Download](#) | [Documentation](#) | [Hall of Fame](#) | [Source](#) | [Discussion group](#)]

If Beautiful Soup has saved you a lot of time and money, the best way to pay me back is to check out [Constellation Games, my sci-fi novel about alien video games](#). You can [read the first two chapters for free](#), and the full novel starts at 5 USD. Thanks!

If you have questions, send them to [the discussion group](#). If you find a bug, [file it](#).

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:

1. Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application
2. Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8. You don't have to think about encodings, unless the document doesn't specify an encoding and Beautiful Soup can't detect one. Then you just have to specify the original encoding.
3. Beautiful Soup sits on top of popular Python parsers like [lxml](#) and [html5lib](#), allowing you to try out different parsing strategies or trade speed for flexibility.

Beautiful Soup parses anything you give it, and does the tree traversal stuff for you. You can tell it "Find all the links", or "Find all the links of class `externalLink`", or "Find all the links whose urls match `foo.com`", or "Find the table heading that's got bold text, then give me that text."

Valuable data that was once locked up in poorly-designed websites is now within your reach. Projects that would have taken hours take only minutes with Beautiful Soup.

Interested? [Read more](#).

Download Beautiful Soup

The current release is [Beautiful Soup 4.5.1](#) (August 2, 2016). You can install Beautiful Soup 4 with `pip install beautifulsoup4`. It's also available as the `python-beautifulsoup4` package in Debian, Ubuntu, and Fedora.

Beautiful Soup 4 works on both Python 2 (2.7+) and Python 3.

Beautiful Soup is licensed under the MIT license, so you can also download the tarball, drop the `bs4/` directory into almost any Python application (or into your library path) and start using it immediately. (If you want to do this under Python 3, you will need to manually convert the code using `2to3`.)

Beautiful Soup 3



#0 설치하기

*작업환경은 Ubuntu 16.04, python3
python3에서 한글(유니코드)의 인식이 잘 된다.

각 운영체제마다 설치법이 약간씩 다르다.

pip을 이용한 간편한 설치법

*Windows에서는 PATH 환경변수에 Python의 경로가 추가되어 있어야한다.

```
$ pip install beautifulsoup4 # UNIX 계열 운영체제  
C:\> pip install beautifulsoup4 # Windows
```

pip이 아니라 기본 package manager를 통한 설치방법

```
$ sudo apt-get install python3-bs4 # Ubuntu, Debian  
$ sudo yum install python3-beautifulsoup4 # Fedora, CentOS  
$ sudo pacman -S python-beautifulsoup4 # Arch Linux
```

2.2.1 Scraping by BS

BeautifulSoup을 이용한 Scraping - #1 URL 읽기 & Parsing

#1 URL에서 읽어 오기

*이 코드를 복붙 하면 되지 않는다.
바로 따옴표 "이 문제가 되기 때문이다.

```
# -*- coding: utf-8 -*-  
import requests  
from bs4 import BeautifulSoup
```

```
url = 'http://...'  
html = requests.get(url).text  
soup = BeautifulSoup(html, 'html.parser')
```

```
print(soup.prettify())      #읽어온 HTML 문서를 중간 결과로 확인
```

2.2.1 Scraping by BS

BeautifulSoup을 이용한 Scraping - #1 URL 읽기 & Parsing

#1 URL에서 읽어 오기

```
# -*- coding: utf-8 -*-  
import requests  
from bs4 import BeautifulSoup
```

```
url = 'http://...'  
html = requests.get(url).text  
soup = BeautifulSoup(html, 'html.parser')
```

```
print(soup.prettify())    #읽어온 HTML 문서를 중간 결과로 확인
```

*맨 처음에 이 코드가 utf-8을 대상으로 짜여졌다는 의미이다.

2.2.1 Scraping by BS

BeautifulSoup을 이용한 Scraping - #1 URL 읽기 & Parsing

#1 URL에서 읽어 오기

```
# -*- coding: utf-8 -*-
import requests
from bs4 import BeautifulSoup

url = 'http://...'
html = requests.get(url).text
soup = BeautifulSoup(html, 'html.parser')

print(soup.prettify())
```

*urllib는 url과 관련된 함수들이 들어있는 package
urlopen은 urllib안의 request module에 들어있다.
python2의 경우에는 urllib.request가 아니라 urllib

#읽어온 HTML 문서를 중간 결과로 확인

#1 URL에서 읽어 오기

```
# -*- coding: utf-8 -*-  
import requests  
from bs4 import BeautifulSoup
```

*BeautifulSoup을 bs4 package에서 불러온다.

```
url = 'http://...'  
html = requests.get(url).text  
soup = BeautifulSoup(html, 'html.parser')
```

```
print(soup.prettify())    #읽어온 HTML 문서를 중간 결과로 확인
```

#1 URL에서 읽어 오기

```
# -*- coding: utf-8 -*-  
import requests  
from bs4 import BeautifulSoup
```

```
url = 'http://...'
```

*읽고 싶은 사이트의 주소를 url에 저장한다.

```
html = requests.get(url).text
```

```
soup = BeautifulSoup(html, 'html.parser')
```

```
print(soup.prettify())
```

#읽어온 HTML 문서를 중간 결과로 확인

#1 URL에서 읽어 오기

```
# -*- coding: utf-8 -*-  
import requests  
from bs4 import BeautifulSoup
```

```
url = 'http://...'
```

```
html = requests.get(url).text
```

```
soup = BeautifulSoup(html, 'html.parser')
```

```
print(soup.prettify())
```

#읽어온 HTML 문서를 중간 결과로 확인

*url을 get으로 읽어서 그 안에 있는 text를 뽑는다.

#1 URL에서 읽어 오기

```
# -*- coding: utf-8 -*-  
import requests  
from bs4 import BeautifulSoup
```

```
url = 'http://...'  
html = requests.get(url).text  
soup = BeautifulSoup(html, 'html.parser')
```

*BeautifulSoup() 함수는 res를 html.parser를 이용해 parsing을 하여 soup에 저장한다.

```
print(soup.prettify())    #읽어온 HTML 문서를 중간 결과로 확인
```

#1 URL에서 읽어 오기

```
# -*- coding: utf-8 -*-  
import requests  
from bs4 import BeautifulSoup  
  
url = 'http://...'  
html = requests.get(url).text  
soup = BeautifulSoup(html, 'html.parser')  
  
print(soup.prettify())
```

*최근 웹 사이트들 중에는 트래픽 절약을 위하여 들여쓰기와 탭을 꼭 필요한 부분을 제외하고 모두 제거하는 uglify 작업을 하는 곳이 많다. 이럴 때 prettify() 함수는 soup 자료 구조를 들여쓰기 된 HTML 문서로 보여준다.

#읽어온 HTML 문서를 중간 결과로 확인

#2 필요한 부분 뽑아 내기 ; using **CSS Selector**

다시 한번, 웹 문서를 살펴보자. 이번엔 소스보기 (앞에서 print한 것을 볼 수 있다.)

***복습!**

개발자 도구 (검사)

소스보기

- IE와 Microsoft Edge : **F12** **Ctrl** + **U**
- Firefox와 Chrome : **Ctrl** + **Shift** + **I** **Ctrl** + **U**
- Safari (in OS X) : **Cmd** + **Option** + **I** **Cmd** + **Option** + **u**

원하는 홈페이지에 가서 해보자.

#2 필요한 부분 뽑아 내기 ; using CSS Selector

CSS 선택자를 알고 나면 복잡한 함수들 필요없이 .selector() 하나만 가지고 요래 저래 요리할 수 있다!

```
<body class="emall_body_em">  
  <div id="skip">  
    <ul>  
      ...
```

soup.select('body div ul')

또는

soup.select('.emall_body_em #skip ul') : 그런 경우 class나 id를 이용하면 된다!

*class는 앞에 .을, id는 앞에 #을 붙인다.

#2 필요한 부분 뽑아 내기 ; ~~커의 selector로 되긴 하지만...~~

BeautifulSoup에서 parsing을 하는데 가장 많이 쓰이는 것은 다음과 같다.

문법

```
soup = BeautifulSoup(target, 'parser')
```

```
soup.find_all('tag', item="value")
```

```
soup.get('item')
```

```
soup.tag
```

예시

```
soup = BeautifulSoup(res, 'html.parser')
```

```
soup.find_all('div', class_="singer")
```

```
soup.get('href')
```

```
soup.string
```

*가능한 item : `class_`, href, title, id, ...

** `class`는 이미 python에서 사용하고 있는 이름이기 때문에, `class_`를 쓴다.

#2 필요한 부분 뽑아 내기

```
soup = BeautifulSoup(target, 'parser')
```

BeautifulSoup은 urlopen으로 읽어온 가장 row data를 먼저 각 tag 별로 parsing을 해준다. 가능한 parser는 html.parser, lxml, xml, html5lib가 존재한다. 각 parser의 특징은 매뉴얼에 잘 나와있다.

```
soup = BeautifulSoup(res, 'html.parser')
```

#2 필요한 부분 뽑아 내기 ; .find_all()

```
soup.find_all('tag', item="value")
```

.find_all() 함수는 가장 중요하다고 볼 수 있다. parsing된 문서에서 해당하는 tag와 그 아래 딸린 아들내용들을 모두 찾아준다. item을 입력하지 않아도 된다. item을 입력 시 그 item이 value의 값과 동일한 tag들만 찾아준다. 결과 값은 리스트 형태로 넘겨준다. [좀 더 자세한 사용법은 홈페이지를 참조](#)

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/#css-selectors>

```
soup.find_all('div', class_="singer")  
soup.find_all('a')
```

*가능한 item : `class_`, href, title, id, ...

** class는 이미 python에서 사용하고 있는 이름이기 때문에, `class_`를 쓴다.

#2 필요한 부분 뽑아 내기 ; .get("") 또는 [""]

```
soup.get('item')
```

.get() 함수는 item에 해당하는 value를 반환해준다. 즉 item이 href였다면 url을, title이었다면 제목을 반환해주는 것이다.

이의 약식 표현으로 예시처럼 ['item']처럼 사용해도 된다.

다만, soup 안에 여러 개의 item 값이 있어도, 한 가지 값만 보여준다

```
soup.get('href')
```

```
soup['href']
```

*가능한 item : `class_`, href, title, id, ...

** class는 이미 python에서 사용하고 있는 이름이기 때문에, `class_`를 쓴다.

#2 필요한 부분 뽑아 내기 ; .tag 또는 .text

`soup.tag`

.tag() 함수는 하위 tag를 찾아준다. 아주 간편하게 하위 tag로 갈 수 있지만, 만약에 같은 이름을 가진 하위 tag들이 여러 개라면, 한 가지 요소만 찾아준다. 따라서 여러 개가 존재하는 경우 .find_all() 함수를 이용한다.

`soup.div`

`soup.text` *요소 내 그냥 text는 이렇게 뽑으면 된다.

*가능한 tag : 하위에 있는 모든 tag명과 parent를 하면 바로 상위 태그로 갈 수 있다.

#3 이외에 필요한 python 문법: 파일 읽고쓰기, list형, for 문

```
f = open("result.txt", 'w')
f.write(variable + ', ' + variable + '\n')
f = open("result.txt", 'r')
f.read()
f.close()
```

```
for i in list:
    command...
```

*파일 읽고 쓰기

**list형 변수는 x[0]부터 x[n]까지 존재

***for loop은 list형의 처음 자료부터 끝까지 i의 형태로 loop을 돌려준다. 즉 처음엔 list[0], 두 번째는 list[1]과 같은 값으로 계속 i를 이용하여 command를 수행한다.

#4 실전으로...

자 이제 실전으로 들어가 봅시다.

배운 함수들을 이용하여 파이썬에서 실습을 해 봅시다.

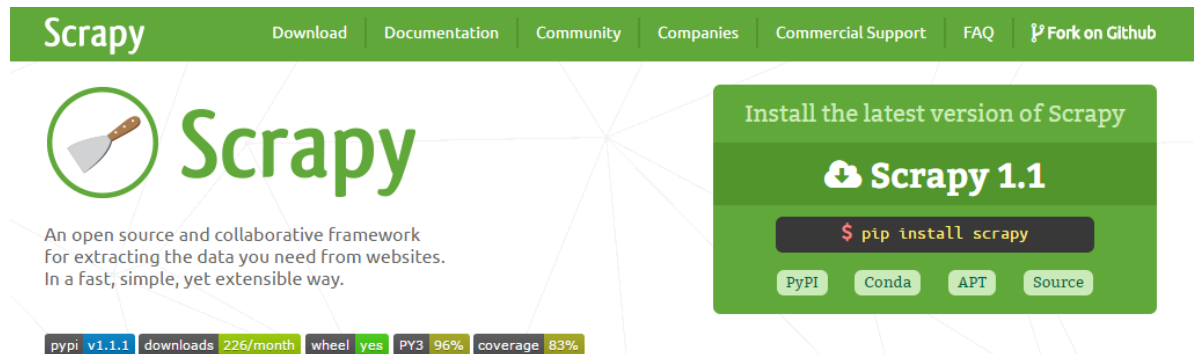


2.3 Beyond the BS4

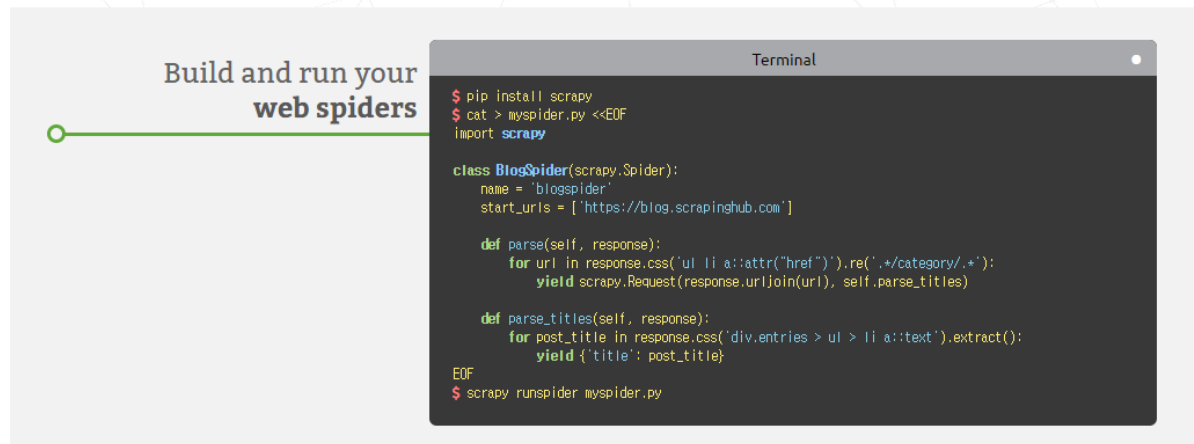
More than BeautifulSoup!? - Scrapy

이것 말고도 Scrapy라는 더 방대한 프로그램도 존재한다.
하지만 훨씬 복잡하기 때문에, 여기서 생략!

꿀팁: 유튜브에서 oneq의 강좌를 참고!!



The image shows the Scrapy website homepage. At the top, there is a navigation bar with links for Download, Documentation, Community, Companies, Commercial Support, FAQ, and Fork on Github. The main content area features the Scrapy logo (a green circle with a scraper icon) and the text "Scrapy". Below the logo, it says "An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way." To the right, there is a green box that says "Install the latest version of Scrapy" and "Scrapy 1.1", with a button for "\$ pip install scrapy" and links for PyPI, Conda, APT, and Source. At the bottom, there are statistics for pypi v1.1.1, downloads 226/month, wheel yes, PY3 96%, and coverage 83%.



The image shows a terminal window titled "Terminal" with the following code:

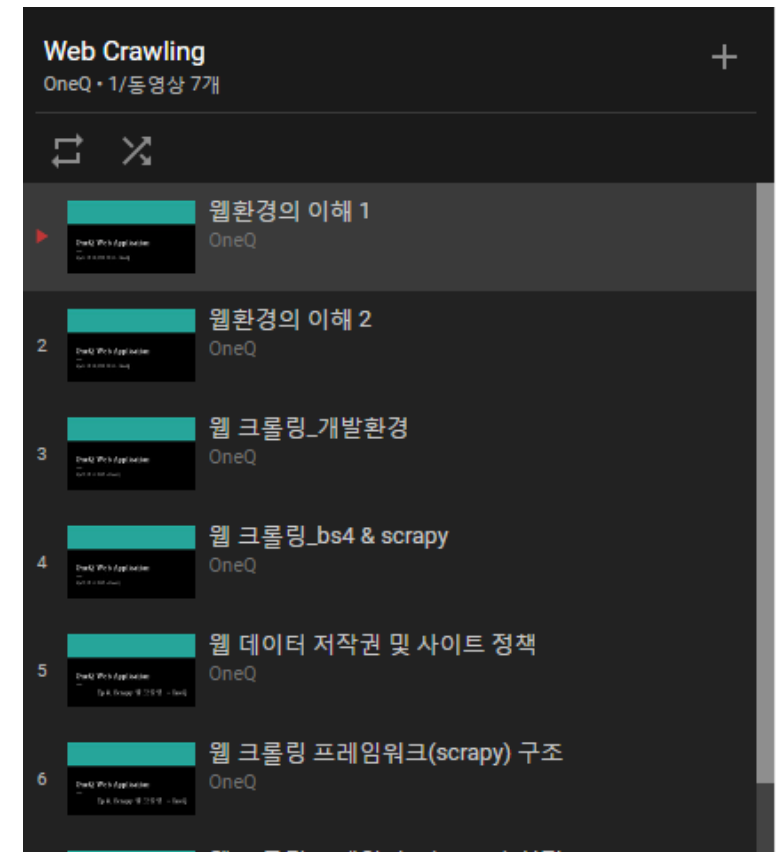
```
Build and run your web spiders

Terminal
$ pip install scrapy
$ cat > myspider.py <<EOF
import scrapy

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

    def parse(self, response):
        for url in response.css('ul li a::attr("href")').re('.*/category/*'):
            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}
EOF
$ scrapy runspider myspider.py
```



The image shows a YouTube playlist titled "Web Crawling" with 7 videos. The videos are:

1. 웹환경의 이해 1 (OneQ)
2. 웹환경의 이해 2 (OneQ)
3. 웹 크롤링_개발환경 (OneQ)
4. 웹 크롤링_bs4 & scrapy (OneQ)
5. 웹 데이터 저작권 및 사이트 정책 (OneQ)
6. 웹 크롤링 프레임워크(scrapy) 구조 (OneQ)
7. 웹 크롤링 프레임워크(scrapy) 사용법 (OneQ)

Scrapy; Crawling을 하기 위한 더 많은 기능 제공

Scraping뿐 아니라 Crawling을 위해 더 많은 기능을 제공한다.
(scrapy보다 더) 다양한 기능을 제공한다.

또한 당연히 parser로 BS4를 사용할 수 있다. (import 해다 쓰면 되니까)

*Linux Scheduler 이용하여 정기적으로 작업

*그 결과를 메일링으로 정기적으로 보낼 수 있음

등 등 훨씬 방대한 기능을 제공한다.

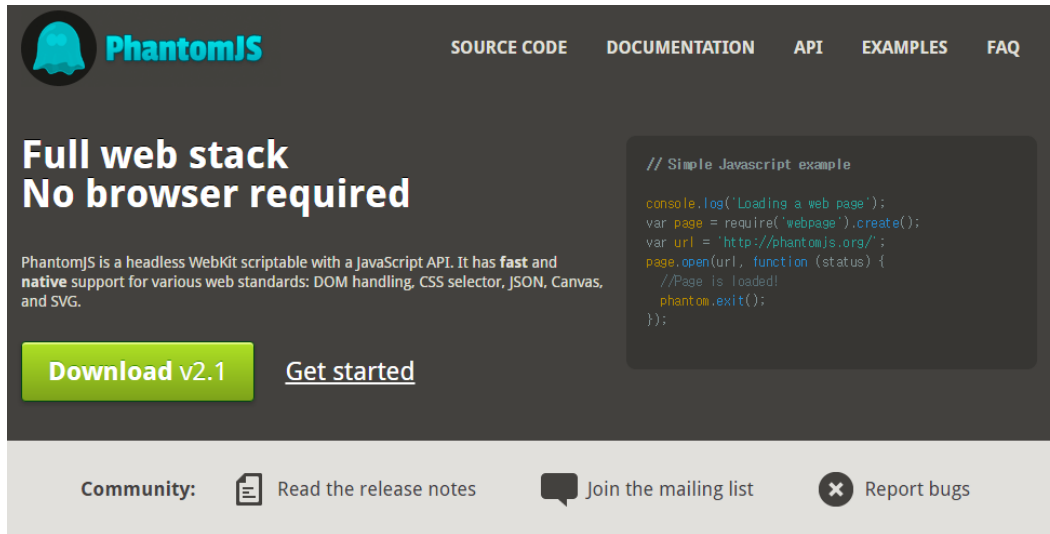
웹 문서는 두 가지로 나뉜다. JavaScript이거나, 혹은 아니거나

일반적인 HTML 문서로 짜여져 있는 것은 어렵지 않게 Scraping을 할 수 있다. 이유는 html 문서의 모든 요소들을 바로 읽을 수 있기 때문이다.

하지만, JavaScript로 짜여진 경우는 이야기가 다르다.
잘은 모르지만 액션 (*.do)를 사용한다.

이런 경우는 액션을 호출 한 뒤의 html문서를 읽어올 수 없기 때문에,
이런 문서를 동적으로 불러오기 위해서는?

PhantomJS 또는 Selenium이 필요하다.



The screenshot shows the PhantomJS website homepage. At the top, there is a navigation bar with links for SOURCE CODE, DOCUMENTATION, API, EXAMPLES, and FAQ. The main heading reads "Full web stack No browser required". Below this, a short paragraph describes PhantomJS as a headless WebKit scriptable with a JavaScript API. A green "Download v2.1" button and a "Get started" link are visible. A code block shows a simple JavaScript example for loading a web page. At the bottom, there is a "Community" section with links to "Read the release notes", "Join the mailing list", and "Report bugs".

PhantomJS is an optimal solution for

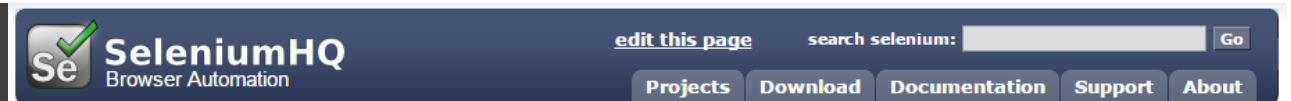
HEADLESS WEBSITE TESTING
Run functional tests with frameworks such as Jasmine, QUnit, Mocha, Capybara, WebDriver, and many others. [Learn more](#)

SCREEN CAPTURE
Programmatically capture web contents, including SVG and Canvas. Create web site screenshots with thumbnail preview. [Learn more](#)

PAGE AUTOMATION
Access and manipulate webpages with the standard DOM API, or with usual libraries like JQuery. [Learn more](#)

NETWORK MONITORING
Monitor page loading and export as standard HAR files. Automate performance analysis using YSlow and Jenkins. [Learn more](#)

PhantomJS is used in the test workflow of various open-source projects: [Bootstrap](#), [CodeMirror](#), [Ember.js](#), [jQuery Mobile](#), [Less.js](#), [Modernizr](#), [YUI3](#), and [many more](#).



The screenshot shows the SeleniumHQ website header. It features the SeleniumHQ logo (a green checkmark over the letters 'Se') and the text "Browser Automation". To the right, there is a search bar labeled "search selenium:" with a "Go" button. Below the search bar are navigation buttons for "Projects", "Download", "Documentation", "Support", and "About".

What is Selenium?

Selenium automates browsers. That's it! What you do with that power is entirely up to you. Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) also be automated as well.

Selenium has the support of some of the largest browser vendors who have taken (or are taking) steps to make Selenium a native part of their browser. It is also the core technology in countless other browser automation tools, APIs and frameworks.

Which part of Selenium is appropriate for me?

Selenium WebDriver



If you want to

- create robust, browser-based regression automation suites and tests
- scale and distribute scripts across many environments

Selenium IDE



If you want to

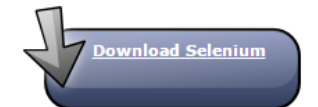
- create quick bug reproduction scripts
- create scripts to aid in automation-aided exploratory testing



Selenium is a suite of tools to automate web browsers across many platforms.

Selenium...

- runs in [many browsers](#) and [operating systems](#)
- can be controlled by many [programming languages](#) and [testing frameworks](#).



2.4 For Active pages

JavaScript...고생끝!!

하지만, 안쓰고도 할 수 있다.

2. 여기를 누르면 아래처럼

그.렇.지.만, 사이트에 대한 이해를 조금만 하면 된다.

1. 요기로 가보자.

예시를 위해 CU의 best상품을 가져왔다.

개발자 도구(검사)를 실행하여 Network 탭으로 가서 분석하고자 하는 페이지를 누르면 동적으로 보이는 반응들을 체크 할 수 있다.

여기서는 *.do 파일을 살펴봐야 한다.

2. 여기를 누르면 아래처럼

1. 요기로 가보자.

3. 클릭

Name	Status	Type	Initiator	Size	Time	Timeline - Start Time
bestAjax.do	200	xhr	jquery-1.10...	24.5 ...	126 ...	
font.css	200	style...	jquery-1.10...	(from...	5 ms	
common.css	200	style...	jquery-1.10...	(from...	5 ms	
prod_list.js?_=147787...	200	xhr	jquery-1.10...	898 B	14 ms	
NanumGothic-Regula...	200	font	jquery-1.10...	(from...	25 ms	

하지만, 안쓰고도 할 수 있다.

그.렇.지.만, 사이트에 대한 이해를 조금만 하면 된다. 4. 여기 요기저기를 왔다 갔다 한다.



예시를 위해 CU의 best상품을 가져왔다.
 CU 예제의 경우엔 bestAjax.do의 변화를
 Form Data에서 살펴보면 된다.
 전체 / 간편식사 등을 다녀보고,
 맨 아래 더보기 등을 누르면

pageIndex, searchMainCategory가 변하는
 것을 확인할 수 있다.

6에 해당하는 view source를 누른다.

<http://cu.bgfretail.com/product/best.do?category=product&depth2=2>

5. 요기를 보자

6. 여길 누른다
전체의 경우

더 보기 누른 경우

하지만, 안쓰고도 할 수 있다.

그.렇.지.만, 사이트에 대한 이해를 조금만 하면 된다.

7을 복사하고, 8을 더블클릭 하면 나오는
<http://cu.bgfretail.com/product/bestAjax.do>
의 주소 끝에 ?를 붙이고 복사한 것을 이어 붙인다.

pageIndex 등의 파라미터를 바꿔가며 체크한다.
이 페이지의 경우 다음과 같이 입력하면 된다.

<http://cu.bgfretail.com/product/bestAjax.do?pageIndex=1&searchMainCategory=>
*pageIndex를 1~2까지 하면 모든 상품을 볼 수 있다. (for loop)으로 전부 체크!
**searchMainCategory는 널 값을 줘야만 제대로 뜨는 것을 확인 할 수 있다.

8. 더블 클릭

7. 요기 복사

9. I got it!



뽀로로미니카 7,000원



페브리즈허브향 7,100원

잘. 된. 다.

상황에 따라 다양한 사이트에 대해 이해를 하고,
Network tab을 응용하여 사용하면 된다.

하지만 모든 경우에 되는 것은 아니다.

이건 일부분에 먹히는 편 법...
그러니 안되면,

강 셀레니움이나 팬텀JS 고고싱...ㅠ_ㅠ

조홍

좌절감이 차나어를 키우는 것이다!



using Web API

Web API (Application Programming Interface)란 웹 애플리케이션 개발에서 다른 서비스에 요청을 보내고 응답을 받기 위해 정의된 명세

facepy

facebook-sdk

python-twitter



2.6 Reference

시간 날 때 읽어보면 좋은 내가 공부했던 사이트들

<http://jhjeong.com/python/2016/01/28/beautifulsoup-web-crawling> - beautifulsoup을 사용한 웹 크롤링

<http://younghwanshin.com/pythonbeautifulsoup%EB%A5%BC-%ED%99%9C%EC%9A%A9%ED%95%9C-%EC%9B%B9%ED%81%AC%EB%A1%A4%EB%A7%81-%EB%8F%84%EC%A0%84%EA%B8%B0/> - beautifulsoup을 사용한 웹 크롤링 도전기

<http://rednooby.tistory.com/4> - [Python] 파이썬을 이용한 크롤링

<http://creativeworks.tistory.com/entry/PYTHON-3-Tutorials-24-%EC%9B%B9-%ED%81%AC%EB%A1%A4%EB%9F%AClike-Google-%EB%A7%8C%EB%93%A4%EA%B8%B0-1-How-to-build-a-web-crawler> - [PYTHON 3] Tutorials 24. 웹 크롤러(like Google) 만들기 1 - How to build a web crawler

<https://vnthf.github.io/blog/crawling/> - 크롤링 해보기 (scrapy)

<http://uslifelog.tistory.com/45> - [Scrapy] 웹사이트 크롤링해서 파일 저장 하기(분양정보수집사례)

<http://m.blog.naver.com/engk0924/220652380312> - 파이썬을 활용한 웹크롤링1

2.6 Reference

시간 날 때 읽어보면 좋은 내가 공부했던 사이트들

<https://www.youtube.com/watch?v=dhxq6RZkm0> - [AskDjango] 국회 사이트, 국회의원 목록 크롤링

감사합니다.

kimyoungjin06@gmail.com

<https://www.facebook.com/Young.Jin.Kim.06>

<http://kimyoungjin06.wixsite.com/youngergerman06>