Ask Company

크롤링 차근차근 시작하기

Overview

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

크롤링

- 사람이 웹페이지에 접속해서 정보를 읽어들이듯이
- 인터넷 상에 흩어져있는 자료들을 사람 대신에 프로그램을 통해 서핑하며 수집&가공
- 이때 프로그램 구성에 따라 서핑능력의 차이가 발생.
  - 대표적으로 자바스크립트 혹은 ActiveX 처리 여부

"Web scraping is a computer software technique of extracting information from websites."

크롤링으로 해볼 수 있는 것

- Bot 프로그램과 연계하기 참 좋습니다.
- 오늘 날씨정보를 읽어와서, 메세지 알림
- 공연예매 빈 좌석이 생기면, 메세지 알림 혹은 자동 예매
- 중단 예정인 서비스가 있는 데, 자료백업을 지원하지 않는다면, 직접 백업
- 회사/학교 식당 메뉴를 찾아보기 힘들다면, 식사 1시간 전에 오늘의 메뉴 메세지 알림
- 가사가 들어있지 않은 MP3파일에 자동 가사 기입
- 국회의원 입법현황을 읽어와서 => 분석 (Pandas 등)

주의: 정보를 읽어올 수 있다고 해서, 마음대로 활용할 수 있다는 것은 아닙니다. 저작권에 유의해주세요.

API가 있다면, API를 쓰세요.

- 크롤링은 정식으로 허용된 방법은 아니기에, 언제라도 막힐 수 있으며, 사이트 변경에 대한 크롤링 로직을 직접 관리/개선
  - 컨텐츠 활용에 대한 부분은 법적인 이슈
- API는 해당 서비스 제공자가 정식으로 제공하는 서비스
  - 해당 API에서 원하는 기능을 제공하지 않을 수도 있습니다.
  - API를 활용하더라도, 컨텐츠 활용에 대한 부분은 법적인 이슈
  - 대개 requests 라이브러리를 통해 처리 가능
- 다양한 API
  - 인스타그램 API
  - 네이버 : 블로그 검색 API, 지도 API

다양한 웹페이지 구현

똑같아 보이는 웹페이지라 할지라도,구현방법이 모두 다를 수 있습니다.

- 필요한 컨텐츠가 HTML을 통해 표현이 모두 되어있을 경우 - **가장 쉽고 빠른 처리**
- 필요한 컨텐츠가 서버로부터 JSON응답을 받아서 처리할 경우
- 필요한 컨텐츠가 JavaScript를 통해 처리할 경우
- 인증이 필요한 웹페이지
- 다양한 웹 프론트엔드 프레임워크를 통해 표현되는 페이지
- ActiveX가 있는 페이지
- ~~Flash로 구현된 페이지 등등등~~

이 외에도 정말 다양한 방식으로 구현한 웹페이지가 있습니다. 그 웹페이지에 맞게 적절하게 크롤링 루틴을 적용해야합니다.

복잡한 크롤링의 경우, 서비스 개발자의 의도를 이해할 수 있어야 합니다.

기본 개발환경

Python 최신버전

- python.org 혹은 anaconda.com 에서 다운받아 설치
  - 설치 시에 "환경변수 PATH에 추가" 옵션 필히 체크 (강력 추천)
- 추가로 필요한 라이브러리
  - requests : 심플한 HTTP 클라이언트
  - beautifulsoup4 : HTML Parser
  - jupyter : Jupyter Notebook을 포함한 Jupyter 환경
- 노트북 서버 구동
  - 쉘> jupyter notebook

http(s)

World Wide Web

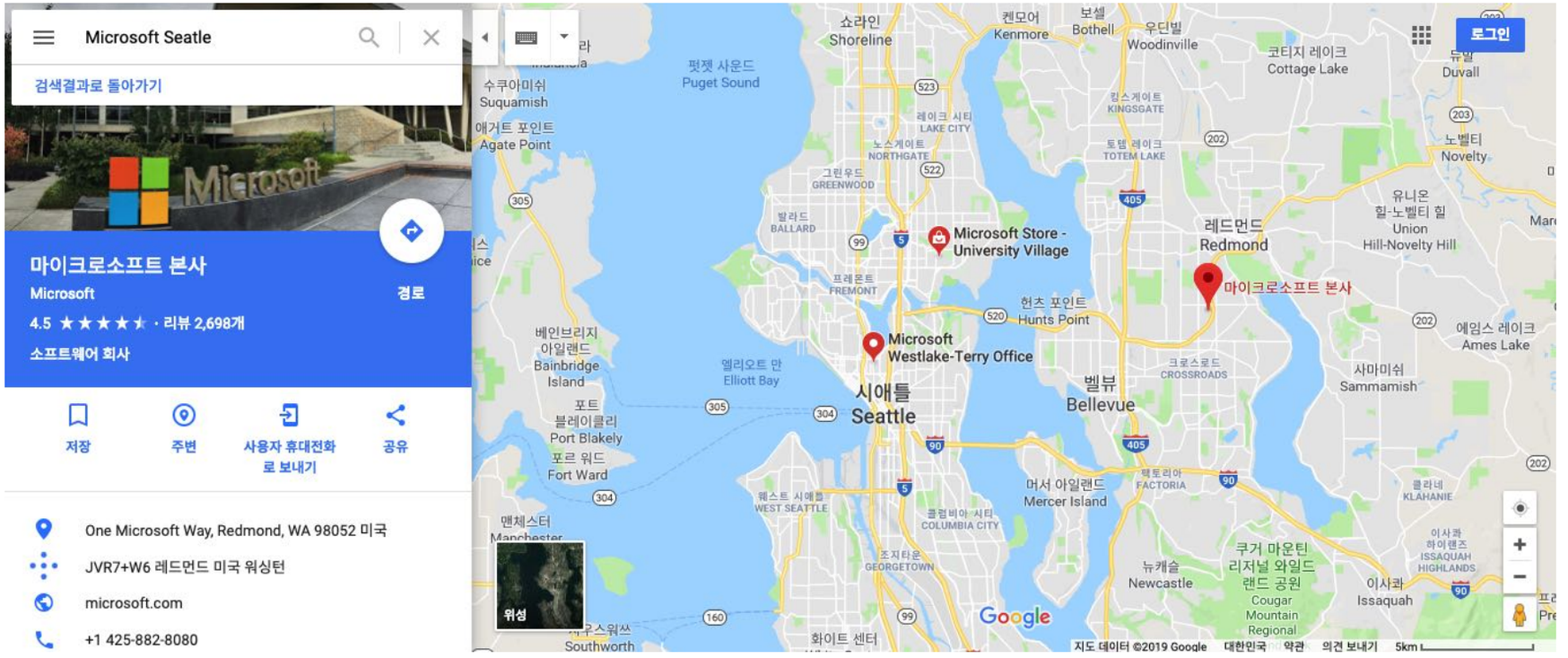
- 프로토콜 : HTTP 혹은 HTTPS
  - Hyper Text Transfer Protocol
- 최근 서비스는 웹 기반의 서버/클라이언트 구조에서 시작
  - 웹프레임워크를 통한 **생산성 극대화**
  - PC 애플리케이션의 시대에서 → 웹 애플리케이션의 시대.
- 서비스에 따라 차후 socket 기반의 서버 구조로 변경하기도.
  - ex) 카카오톡 "겁나빠른황소" 프로젝트 : https://www.bloter.net/archives/88245

Web Page

- 정적인 성격의 웹 사이트
  - HTML 위주
  - JavaScript는 약간의 양념 정도 사용

Web Application

- 하나의 페이지에서 사용자와 Interaction하는 UI를 제공
  - 브라우저 단에서 구동되는 JavaScript를 통해 UI를 구현
  - Heavy JavaScript



다양한 http(s) 클라이언트

- GUI 브라우저
  - Chrome, Firefox, Safari, Internet Explorer, Opera 등
- CLI 브라우저 : w3m, elinks, lynx 등
- CLI 브라우저 (단순요청) : curl, wget 등
- 파이썬 라이브러리/프레임워크
  - requests, httpie 등
  - selenium : 웹브라우저 제어
  - scrapy : 크롤링 프레임워크
    - requests, selenium 등을 활용

http(s)의 구조

선 클라이언트 요청 → 후 서버 응답

1. 클라이언트가 먼저 요청하면, 서버는 이에 응답
   - 대개 웹페이지 접속, 새로고침, 링크 클릭을 통한 페이지 전환
   - Ajax 요청, 앱 API 요청
2. 클라이언트 요청없이, 서버에서 먼저 클라이언트로 데이터 전송 불가
   - 물론 웹소켓이라는 기술이 있긴 합니다만, 이것도 클라이언트 측에서 먼저 서버로 연결을 하고 있어야 합니다.
3. 서버 : 클라이언트 요청을 처리하고, 그에 대한 응답 생성 (장고/플라스크/nginx/아파치 웹서버)
   - 다양한 응답 포맷 : **html**, css, javascript, jpg 등
4. 클라이언트 : 서버로부터의 응답을 받아서, 로직에 따라 처리
   - 브라우저 : 서버로부터 HTML응답을 받아, 이를 해석하여 그래픽으로 표현. 필요하다면 서버로 javascript, css, jpg 등을 추가 요청/다운로드. 자바스크립트를 통해 서버로 Ajax 요청을 전달
   - Android/iOS 앱 : 내부에 웹브라우저(WebView)를 내장하기도 하고, 네이티브 코드로 앱 API를 호출하여 대개 JSON응답을 받아서 처리
5. 계속 반복

웹페이지 샘플

모두 같은 모습을 한 웹페이지이지만, 구현이 모두 다릅니다.

- 레벨1: 단순 HTML
  - https://askdjango.github.io/lv1/
- 레벨2: Ajax를 통해 데이터를 받아와서, 렌더링
  - https://askdjango.github.io/lv2/
- 레벨3: 자바스크립트 객체 값을 활용하여, 렌더링
  - https://askdjango.github.io/lv3/
- 이 외에도 정말 다양한 방식의 구현이 존재.

레벨1) 단순 HTML (샘플 예시)

```
import requests
from bs4 import BeautifulSoup

html = requests.get('https://askdjango.github.io/lv1/').text
soup = BeautifulSoup(html, 'html.parser')

print("시즌2")
for tag in soup.select('#s2_course_list > .course > a'):
    print(tag.text, tag['href'])

print("시즌1")
for tag in soup.select('#s1_course_list > .course > a'):
    print(tag.text, tag['href'])
```

레벨2) Ajax 렌더링 (샘플 예시)

```
import requests

res = requests.get('https://askdjango.github.io/lv2/data.json')
course_dict = res.json()

print("시즌2")
for course in course_dict['s2']:
    print(course['name'], course['url'])

print()
print("시즌1")
for course in course_dict['s1']:
    print(course['name'], course['url'])
```

레벨3) 자바스크립트 렌더링 (샘플 예시)

```
import json
import re
import requests

res = requests.get('https://askdjango.github.io/lv3/')
html = res.text

# 생략 (코드는 나중에 다루겠습니다.) ➔ Hint: 정규 표현식와 js2xml 라이브러리
```

인생은 짧습니다.
파이썬/장고를 쓰세요.

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

- Ask Company