Scraping

*

저녁이 있는 프로젝트
오상훈
6 Hours, 1 Month

Web Scraping

- Web상에 존재하는 Contents를 수집하는 작업 (프로그래밍으로 자동화 가능)
- HTML 페이지를 가져와서, HTML/CSS등을 파싱하고, 필요한 데이터만 추출하는 기법
- Open API를 호출해서, 받은 데이터 중 필요한 데이터만 추출하는 기법
- Selenium등 브라우저를 프로그래밍으로 조작해서 가능.
- 용어 이해
  - ➢ 크롤링(crawling) : 스크래핑 기술 중 하나.
    - 웹 크롤러(crawler)는 조직적, 자동화된 방법으로 WWW를 탐색하는 컴퓨터 프로그램.
    - 여러 인터넷 페이지(문서, html 등)를 수집/분류해 저장한 후 쉽게 찾게 인덱싱.
- 파싱(parsing)
    - 어떤 페이지(문서, html 등) 안에 특정 패턴이나 순서로 추출해 정보를 가공.
    - 일련의 문자열을 의미있는 토큰(token)으로 분해하고 이들로 이루어진 파스트리(parse tree)를 만드는 과정.

```
import requests
from bs4 import BeautifulSoup
res = requests.get('http://media.daum.net/economic/')
res.status_code, res.content
soup = BeautifulSoup(res.content, 'html.parser')
links = soup.select('a[href]')
type(links), links                    # list type
```

find_all

❖ find() : 가장 먼저 검색되는 태그만 반환

❖ find_all() : 전체 태그 list 반환

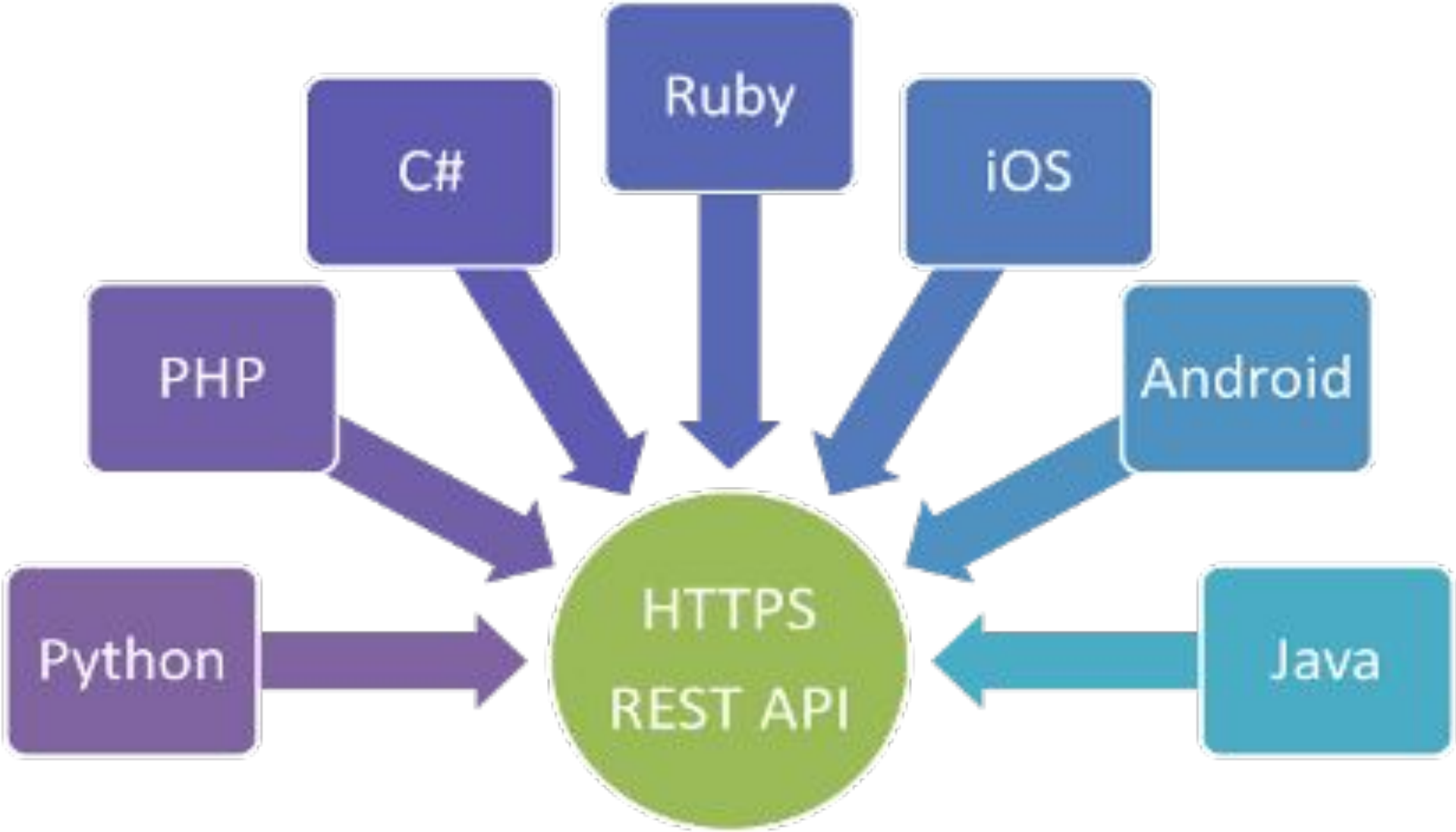
```
html = "<html> <body> \
    <h1 class='public_class_name' id='h1_id_name'>[1]크롤링이란?</h1> \
    <p class='public_class_name' id='p01_id_name'>웹페이지에서 필요한 데이터를 추출하는 것</p> \
    <p id='p02_id_name' align='center'>파이썬을 중심으로 다양한 웹크롤링 기술 발달</p> \
 </body> </html>"
>>> title_data = soup.find_all('h1')                    # tag로 검색
>>> type(title_data), title_data, title_data[0].string
>>> title_data = soup.find_all(id='h1_id_name')          # id로 검색
>>> title_data, title_data[0].get_text()
>>> title_data = soup.find_all('p', class_='public_class_name')# tag와 class로 검색
>>> title_data, title_data[0].string, title_data[0].attrs
[<p class="public_class_name" id="p01_id_name">웹페이지에서 ... 추출하는 것</p>]
웹페이지에서 필요한 데이터를 추출하는 것
{'class': ['public_class_name'], 'id': 'p01_id_name'}
>>> title_data = soup.find_all('p', attrs = {'align': 'center'})    # 속성:속성값으로 검색
>>> title_data, title_data[0].string
```

Web API

- Application Programming Interface 약자로, 특정 프로그램을 만들기 위해 제공되는 모듈(함수 등)을 의미
- Open API: 공개 API라고도 불리우며, 누구나 사용할 수 있도록 공개된 API (주로 Rest API 기술을 많이 사용함)
- Rest API: Representational State Transfer API의 약자로, HTTP프로토콜을 통해 서버 제공 기능을 사용할 수 있는 함수를 의미
- 일반적으로 XML, JSON의 형태로 응답을 전달(원하는 데이터 추출이 수월)

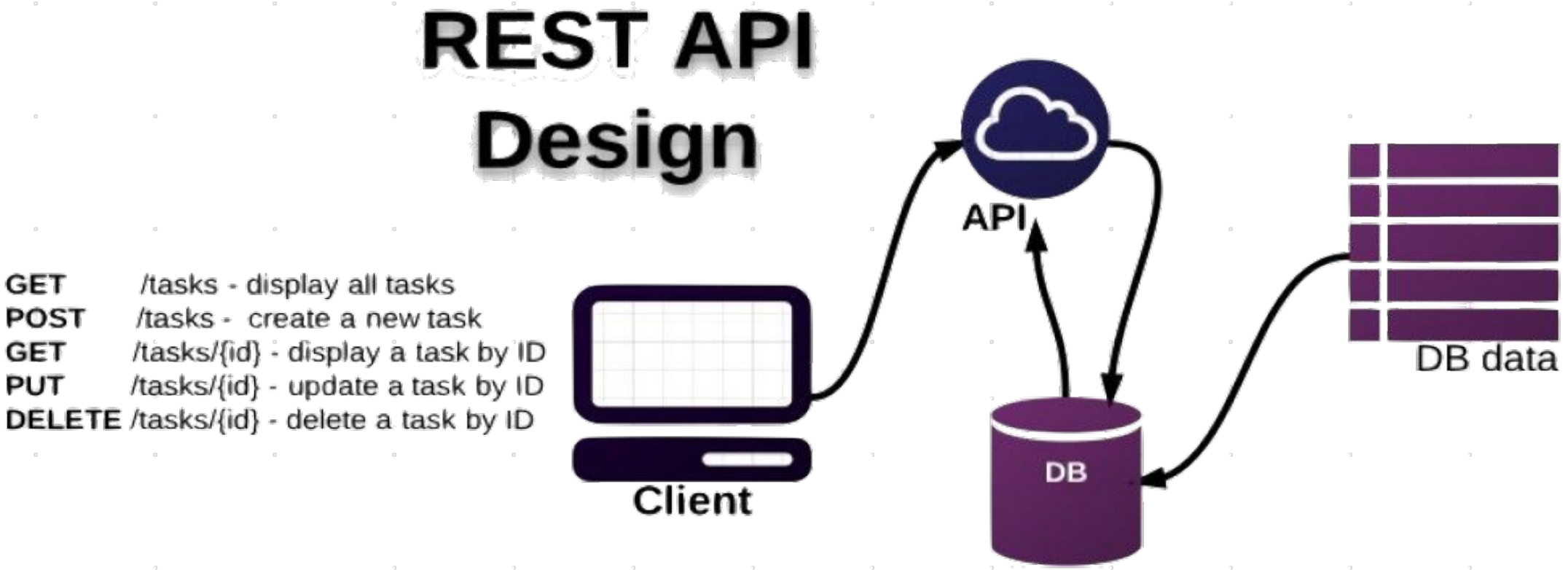
```
>>> info_url = 'http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=109'
>>> response = requests.get(info_url)
>>> soup = BeautifulSoup(response.content, 'html.parser')
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0">
<channel>
<title>기상청 육상 중기예보</title>
…
>>> locations = soup.find_all('location')
>>> for location in locations:
>>>   print(location.find('city').text, ":", location.find('wf').text)
서울 : 구름많음
인천 : 구름많음
수원 : 구름많음
```

REST API



REST(Representational State Transfer)

❖ 필요한 정보만 요청



REST(Representational State Transfer)

- RIA(Rich Internet Application) 대두 : 서버 역할 축소
  - XML-RPC, SOAP
  - REST : WWW 인기
- 특징 : 리소스가 전부
  - 클라이언트 - 서버 분리, 스테이트리스, 유니폼 인터페이스, 계층화된 시스템, 코드-온-디맨드
- 회원가입 : www.openweathermap.org
  - **Get appid** : SignIn > API Keys > Create Key
- Using PostMan in Chrome
  - Tool (https://chrome.google.com/webstore)

Web API

- HTTP 프로토콜: WWW에서 문서 전송을 위한 상호 약속 규칙
- request(요청) / response(응답) 으로 구성
- browser(클라이언트)가 요청하면 web server(서버)가 HTML 파일이나 다른 자원(이미지, 텍스트, 동영상 등)을 응답으로 전송
  - GET 방식 : 데이터 전달을 URL 내에서 함
    https://search.naver.com/search.naver?where=nexearch&sm=top_hty&fbm=1
  - POST 방식 : 데이터 전송을
    - FORM Tag 통해서 함(사용자에게 직접적으로 노출되지 않음)
      예) ID, 비밀번호 전달의 경우
- API Status Codes
  - 200: Everything went **okay**, and the result has been returned (if any).
  - 400: The server thinks you made a **bad request**. This can happen when you don't send along the right data, among other things.
  - 401: The server thinks you're **not authenticated**. Many APIs require login ccredentials, so this happens when you don't send the right credentials to access an API.
  - 403: The resource you're trying to **access is forbidden**, you don't have the right permissions to see it.
  - 404: The resource you tried to **access wasn't found on the server**.
  - 503: The server is **not ready** to handle the request

JSON & XML

- 알아 가기
  - 왜 자료 교환 시 String 타입 사용 불편한가 ?
- Both JSON and XML can be used to receive data from a web server
- JSON vs XML
  - For AJAX applications, JSON is faster and easier than XML

ex) {"employees":[→ key:value, Map - { }, List - [ ]
{ "firstName":"John", "lastName":"Doe" },
{ "firstName":"Anna", "lastName":"Smith" },
{ "firstName":"Peter", "lastName":"Jones" }
]}

ex) <employees>
<employee>
<firstName>John</firstName> <lastName>Doe</lastName>
</employee>
<employee>
<firstName>Anna</firstName> <lastName>Smith</lastName>
</employee>
...
</employees>

Try - JSON & XML

- 회원가입 : www.openweathermap.org
  - **Get appid** : SignIn > API Keys > Create Key
- GET
  - type in address in chrome : chrome://apps/
  - Menu > API > Current weather data in www.openweathermap.org
    ex) api.openweathermap.org/data/2.5/weather?q=London&**appid=yours**
  - insert url value in address and then Enter
- POST
  - click ‘header’ > insert Content-Type : application/json
  - insert URL value in address > Click Button ‘Send’
    ex) api.openweathermap.org/data/2.5/weather?q=London&**appid=yours**
  - Click ‘Code’

REST API - Doc

서비스명	로그인
RETURN MAP	member_json

논리명	I/O	필수여부	물리명	타입	테이블명	필드명	입력값	비고
회원 ID	I	O	login_id	VARCHAR(20)	member			
비밀번호	I	O	login_pwd	VARCHAR(20)	member			
디바이스 맥	I	O	device_mac	varchar(200)				
디바이스 type	I	O	device_type	tinyint	member			
디바이스 Number	I	O	device_no	VARCHAR(200)	member			
결과	O		resultNum	VARCHAR(10)				true / false
회원 ID	O		uid	VARCHAR(20)	member			
회원명	O		name	VARCHAR(20)	member			
회원등급	O		member_type	INT				B(교수) / C(학생)
회원사진	O		mem_img	varchar(50)				
회원 키 값	O		member_idx	INT	member			
최종접속일	O		last_login	VARCHAR(50)	member			

테스트URL	/widzet/login?login_id=mem01&login_pwd=1111
비고	테스트 관리자 정보 : admin / 1234 라스트 로그인 체크 회원이미지 경로 : /upload/member 폴더


수고하셨습니다.