Fastcampus Web Programming SCHOOL

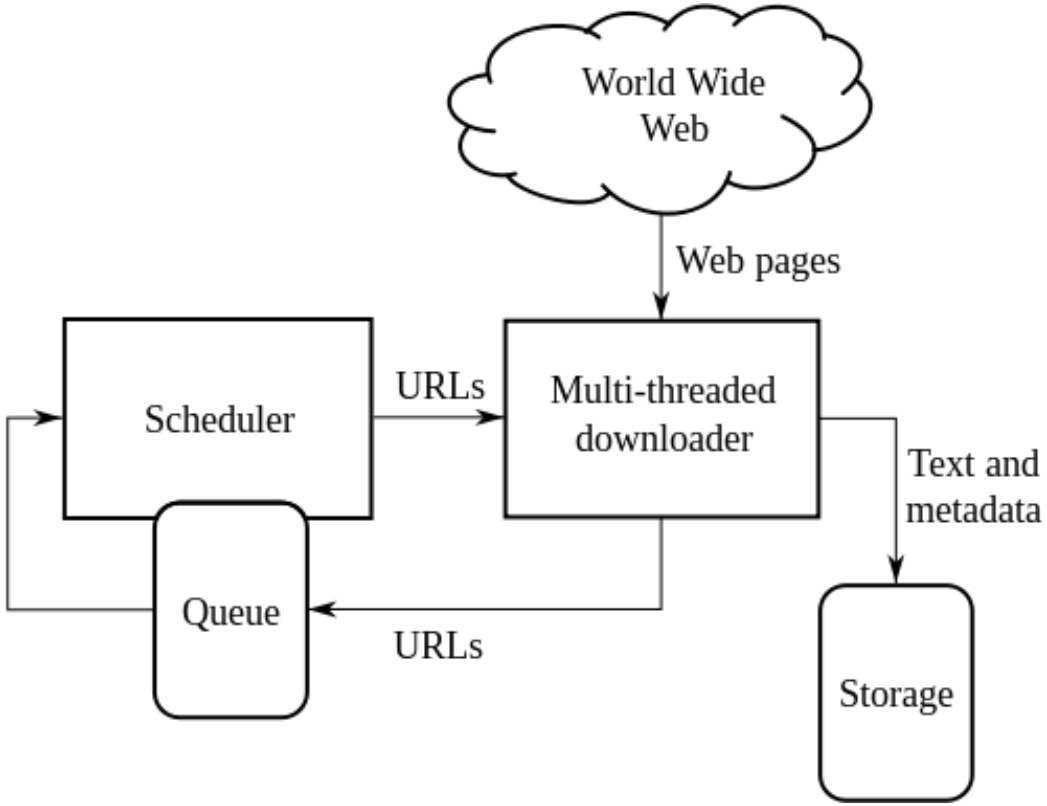
Web scraping, Micro Framework

Get static page content from web

Crawling, Scraping, Parsing

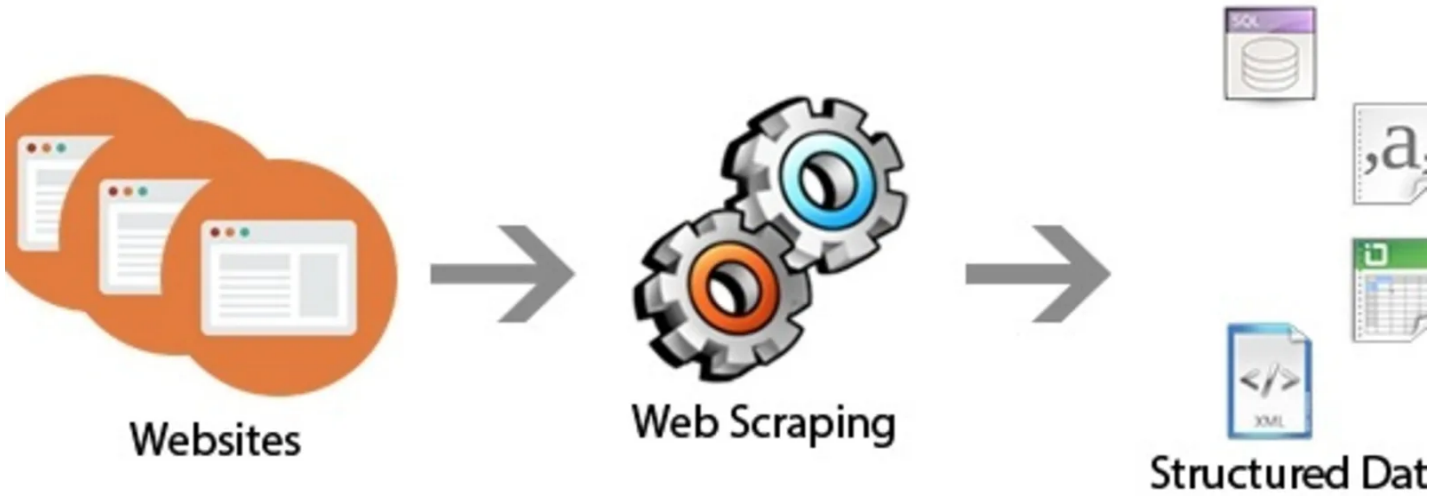
Crawling

Crawling: 조직적 자동화된 방법으로 월드 와이드 웹을 탐색하는 것



Scraping

Scraping: 데이터를 수집하는 행위



Parsing

Parsing: 문장 혹은 문서를 구성 성분으로 분해하고 위계관계를 분석하여 문장의 구조를 결정하는 것

Caution!!

저작권 침해 위반 소지

- 웹사이트 운영자의 크롤링 금지 룰을 어길경우
- 월권하여 데이터베이스에 접근
- 타인의 경제적 이익을 침해할 경우
- 개인정보를 수집할 경우(전화번호, 주소, ..)

Requirements

- requests
- beautifulsoup4
- selenium

How to request information??

```
$ pip install requests
```

```
requests.get(url)
```

How to parse from response data??

```
>>> url="https://www.google.com/"
>>> response = requests.get(url)
>>> response
>>> response.status_code
>>> response.encoding
>>> response.text
>>> response.json()
>>> response.headers
```

request API with requests

simple request

https://api.kurly.com/v2/categories?ver=1

with token

https://api.kurly.com/v3/home/products/2718?&ver=1613284621182

with beautifulsoup

```
$ pip install beautifulsoup4
```

```
$ pip install lxml
```

```
import requests
from bs4 import BeautifulSoup
import lxml


html = request.get().text
soup = BeautifulSoup(html, 'lxml')
lis = soup.find('li')
for li in lis:
    print(li.get_text())
```

Top Box Office data from rotten tomatoes

https://www.rottentomatoes.com

find headline news from the guardian

https://www.theguardian.com/uk/technology

requests, beautifulsoup으로 구할 수 없는 경우

- rendering 후 클라이언트에 의해 데이터가 호출될 경우(ajax, react-router, ..)
- SPA(Single Page Application)

Do it yourself

Selenium Library를 활용하여 평화나라에서 아이패드를 검색했을 때 나오는 결과물을 가져오는 플로우를 구현하세요.

Flask

Web Framework

- 웹서비스를 제공하기 위해 필요한 기능들을 모아둔 클래스와 라이브러리의 모임

Web Frameworks built with python

- Full-stack
  - Django
  - Pyramid
  - Web2py
- Microframework
  - **Flask**
  - Bottle
- Async
  - Tornado
  - Sanic

Flask - start app

```
$ pip install flask
```

```
from flask import Flask


app = Flask(__name__)

@app.route('/')
def index():
        return 'hello world!'

if __name__ == '__main__':
        app.run(host='0.0.0.0', port=8080, debug=True)
```

```
$ python server.py
```

for cloud env like c9.io

```
from flask import Flask
import os

app = Flask(__name__)

@app.route('/')
def index():
    return 'hello world!'

app.run(host=os.getenv('IP', '0.0.0.0'),port=int(os.getenv('PORT', 8080)))
```

Flask - route

```
from flask import Flask


app = Flask(__name__)

@app.route('/')
def index():
        return 'hello'

@app.route('/about')
def about():
        return 'about'

if __name__ == '__main__':
        app.run(host='0.0.0.0', port=8080, debug=True)
```

Flask - render

```
from flask import Flask, render_template


app = Flask(__name__)

@app.route('/')
def index(name=None):
        return render_template('index.html', name=name)

@app.route('/about')
def about(name=None):
        return render_template('about.html', name=name)

if __name__ == '__main__':
        app.run(host='0.0.0.0', port=8080, debug=True)
```

Flask - render

```
/
  server.py
  /templates
    index.html
    about.html
```

Flask with BeautifulSoup

```
from bs import BeautifulSoup

def index():
    ...
    .(some code).
    ...
```

route with querystring, path

```
@app.route('/user/<string:name>')
def user(name=None):
    return render_template('user.html', msg=name)
```

```
@app.route('/users')
def users():
    querystring = request.args
    return render_template('users.html', rows=querystring)
```

form with sqlite

```
@app.route('/movies', methods=['GET', 'POST'])
def movies():
    if request.method == 'GET':
        conn = lite.connect('./data/data.db')
        conn.row_factory = lite.Row

        cur = conn.cursor()
        cur.execute("SELECT * FROM Movies;")

        rows = cur.fetchall()
        conn.close()
        return render_template('movies.html', rows=rows)

...
```

```
...

    elif request.method == 'POST':
        try:
            input_name = request.form["movie-name"]
            input_year = request.form["movie-year"]
            input_studio = request.form["movie-studio"]
            with lite.connect('./data/data.db') as conn:
                cur = conn.cursor()
                cur.execute("""
                        INSERT INTO Movies(name, year, studio)
                        VALUES(?,?,?);
                        """,
                        (input_name, input_year, input_studio))
                conn.commit()
                msg = "Success"
        except:
            conn.rollback()
            msg = "Failed to Save"
        finally:
            return render_template('movies.html', msg=msg)
```

simple api server with flask, mlab

```
mongo_uri = "mongodb://strongadmin:admin1234@ds135844.mlab.com:35844/mydbinstance"

@app.route('/api/v1/item')
def api():
    client = MongoClient(mongo_uri)
    db = client.mydbinstance
    items = db.bigbang
    try:
        query = {}
        projection= {
                "_id":0,
                "title":1,
                "item":1,
                }
        result = list(items.find(query, projection))
    except:
        result = "failed"
    finally:
        return jsonify({"items":result})
```