
# 웹문서 크롤링
# (crawler, spider, web robot)

국민대학교 소프트웨어학부

강 승 식

# Crawling in Python

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re
pages = set()
def getLinks(pageUrl):
    global pages
    html = urlopen("http://en.wikipedia.org"+pageUrl)
    bsObj = BeautifulSoup(html, "html.parser")
    for link in bsObj.findAll("a", href=re.compile("^(/wiki/)")):
        if 'href' in link.attrs:
            if link.attrs['href'] not in pages:
                # 새 페이지를 발견
                newPage = link.attrs['href']
                print(newPage)
                pages.add(newPage)
                getLinks(newPage)
getLinks("")
```

출처: "파이썬으로 웹 크롤러 만들기", 한빛미디어

# BeautifulSoup

- Python package for parsing HTML and XML documents
  - It creates a parse tree for parsed pages
  - Useful for web scraping

```
# anchor extraction from html document
from bs4 import BeautifulSoup
import urllib2

webpage = urllib2.urlopen('http://en.wikipedia.org/wiki/Main_Page')
soup = BeautifulSoup(webpage, 'html.parser')
for anchor in soup.find_all('a'):
    print(anchor.get('href', '/'))
```

# Regular Expression

- https://en.wikipedia.org/wiki/Regular_expression
- https://ko.wikipedia.org/wiki/%EC%A0%95%EA%B7%9C_%ED%91%9C%ED%98%84%EC%8B%9D (정규표현식)
- Example

  [A-Za-z0-9\._+]+@[A-Za-z]+.(com|org|edu|net)

  (?:\.) {2,}(?=[A-Z])

  - blank 2개 이상인 것 중에서 '.'(마침표) 뒤에, 그리고 대문자 앞에 blank가 2개 이상인 것만 match
- <참고> vi 사용법 퀴즈 (gvim 또는 vim)
  - http://cafe.naver.com/sskangpl/32
  - https://ko.wikipedia.org/wiki/Vi
  - https://en.wikipedia.org/wiki/Comparison_of_text_editors

# Crawling in Windows MFC Library

```
#include <urlmon.h>

HRESULT hr = URLDownloadToFile(
        NULL,   // A pointer to the controlling IUnknown interface
        url,
        filePath,
        0,      // Reserved. Must be set to 0.
        pBindStatusCallback );
```

# Crawling in Java

```
package crawler;

import java.io.*;
import java.net.*;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.util.*;

public class test {
 public static void main(String[] args) {
  // TODO Auto-generated method stub
  URL url;

  try {
   url = new URL("http://www.naver.com");

   BufferedReader br;
   BufferedWriter bw;
   String l;
```

```
br = new BufferedReader(new InputStreamReader(url.openStream(),"utf-8"));
bw = new BufferedWriter(new FileWriter("text11.txt"));

while ((l = br.readLine()) != null) {
 Pattern p = Pattern.compile("<img[^>]*src=[\'"]?([^>\'"]+)[\'"]?[^>]*>");
 Matcher mc = p.matcher(l);
 while(mc.find()) bw.write(mc.group(1));
}
br.close();
bw.close();

} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}}}
```

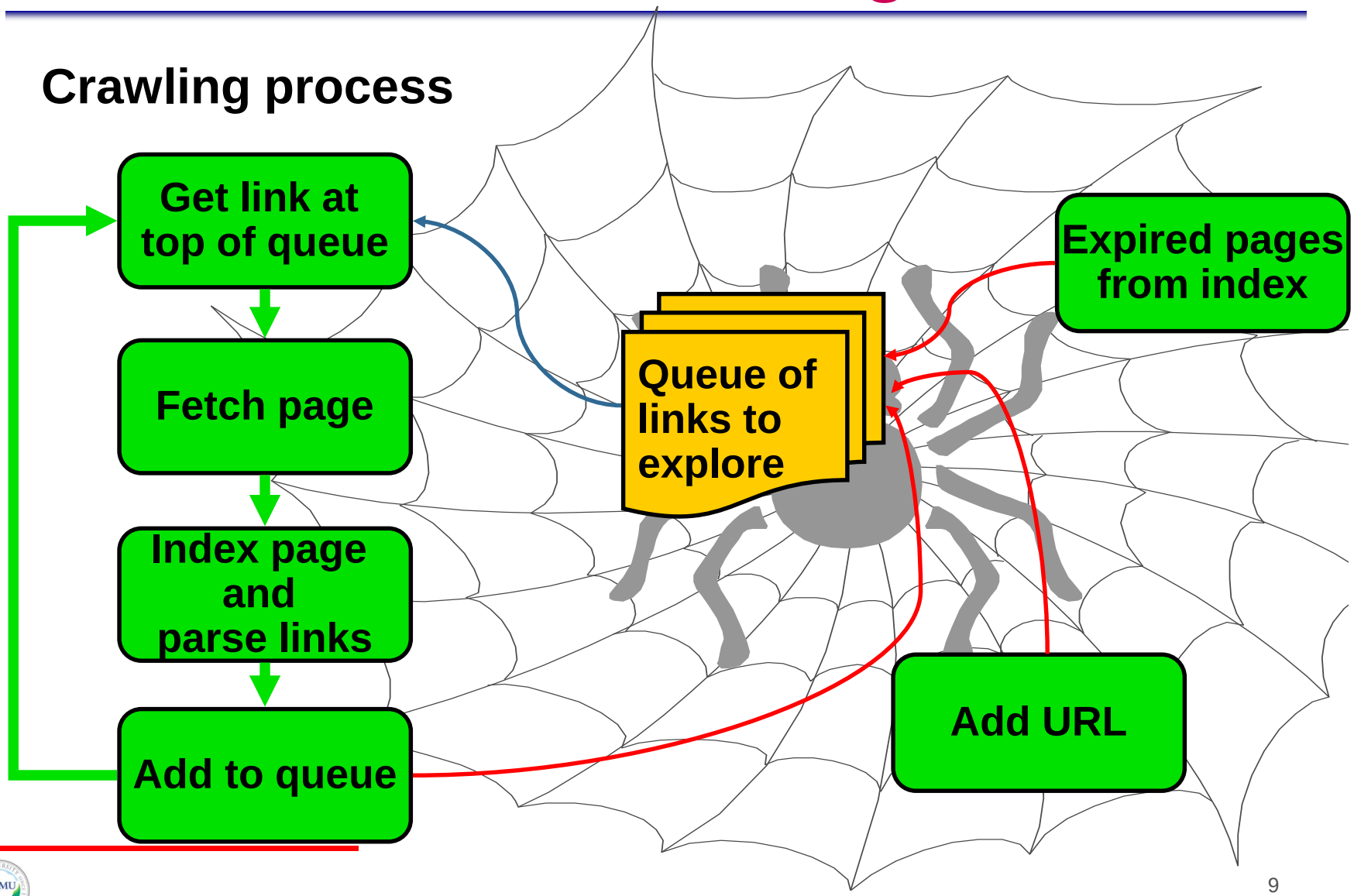
출처: http://develop88.tistory.com/51 [왕 Blog]

# 웹 문서 수집

- 목표 : 사용자 요구에 적합한 문서 수집
  - Static: html, text, image, audio …
  - Dynamic: DB access

- 논점
  - URL 리스트 확보
    - Hyperlink, 모든 웹서버(IP)
  - Static page 수집 방법
  - Dynamic page 학습 방법

# Web crawling



# Queuing discipline

- Standard graph exploration:
  - Random
  - BFS
  - DFS (+ depth limits)

- Priority based on query-independent ranking
  - Highest in-degree
  - Highest potential PageRank

# Load balancing

- Internal
  - Response time
  - Size of answers
  - No. of threads, no. of open onnections, etc
- External
  - Server overload
  - Queuing discipline