Crawler 구현 문서

Twitter 및 Naver & Daum Blog를 crawling 한다.

1. DB
   1. DB 연결

mongod --dbpath D:\Project\ITSMINE\mongoDB\ --logpath D:\Project\ITSMINE\mongoDB\log\log.txt –directoryperdb

* 1. DB 구조

1. API 사용
   1. Naver
      1. 요청 URL

http://openapi.naver.com/search

* + 1. 요청 변수

|  |  |  |
| --- | --- | --- |
| 변수 | 값 | 설명 |
| Key | String | Api 키값(필수) |
| Target | String | 필수, Blog(블로그), webkr(웹문서),  cafearticle (카페) |
| Query | String | 질의, UTF-8 기본 |
| Display | Int | 검색 출력 건수 10~100 |
| Start | Int | 시작위치 1~1000 |
| Sort | String | Sim(유사도순), date(날짜순) |
| Api key | a4c87ac3748a15f672bb5f644443115f | |

* + 1. 응답 결과

|  |  |  |
| --- | --- | --- |
| 필드 | 값 | 설명 |
| rss | - | 디버그를 쉽게 하고 RSS 리더기만으로 이용할 수 있게 하기 위해 만든 RSS 포맷의 컨테이너이며 그 외의 특별한 의미는 없습니다. |
| channel | - | 검색 결과를 포함하는 컨테이너입니다. 이 안에 있는 title, link, description 등의 항목은 참고용으로  무시해도 무방합니다. |
| lastBuildDate | datetime | 검색 결과를 생성한 시간입니다. |
| total | integer | 검색 결과 문서의 총 개수를 의미합니다. |
| start | integer | 검색 결과 문서 중, 문서의 시작점을 의미합니다. |
| display | integer | 검색된 검색 결과의 개수입니다. |
| item | - | 개별 검색 결과이며, title, link, description, bloggername, bloggerlink을 포함합니다. |
| title | string | 검색 결과 문서의 제목을 나타냅니다. 제목에서 검색어와 일치하는 부분은 태그로 감싸져 있습니다. |
| link | string | 검색 결과 문서의 하이퍼텍스트 link를 나타냅니다. |
| description | string | 검색 결과 문서의 내용을 요약한 패시지 정보입니다. 문서 전체의 내용은 link 를 따라가면,  읽을 수 있습니다. 패시지에서 검색어와 일치하는 부분은 태그로 감싸져 있습니다. |
| bloggername | string | 검색 결과 블로그 포스트를 작성한 블로거의 이름입니다. |
| bloggerlink | string | 검색 결과 블로그 포스트를 작성한 블로거의 하이퍼텍스트 link입니다. |

|  |
| --- |
| <rss version="2.0">  <channel>  <title>Naver Open API - blog ::'sample'</title>  <link>http://search.naver.com</link>  <description>Naver Search Result</description>  <lastBuildDate>Tue, 25 Jun 2013 16:10:04 +0900</lastBuildDate>  <total>132893</total>  <start>1</start>  <display>10</display>  <item>  <title>메고/<b>SAMPLE</b> SALE</title>  <link> http://openapi.naver.com/l?AAABWISw6DIBgGT/OzNAoWVyygsffg8VGSalWKJtxeOpnFZI4TuSqaDWlJ+vkPM5HpWak7lFu2N/ugKj+4sZdO8OAaAB6RWyFD8EP7U2QpI6pUyk5CE381f7DZp+5rL+TOb+sNWgbE7WoAAAA=  </link>  <description>  장마라더니 비가 잠시 소강 상태네요~~~ 패브릭 가방 <b>SAMPLE</b> SALE 들어갑니다.  한 개씩만 있는 제품이구요~~ 재고 소진 후에 주문시에는 SALE 가격 적용 안 되네요~~~^^\*  찜콩 안 되구요~ 반품,교환 안되니깐...  </description>  <bloggername>MEGO</bloggername>  <bloggerlink>http://blog.naver.com/mego72</bloggerlink>  </item>  <item>  <title>메고/<b>SAMPLE</b> SALE/ 패브릭 프린트백/ 진저백st</title>  <link>  http://openapi.naver.com/l?AAABWISw6DIBgGT/OzNAoWVyygsffg8VGSalWKJtxeOpnFZI4TuSqaDWlJ+vkPM5HpWak7lFu2N/ugKj+4sZdO8OAaAB6RWyFD8EP7U2QpI6pUyk5CE381f7DZp+5rL+TOb+sNWgbE7WoAAAA=  </link>  <description>  장마라더니 비가 잠시 소강 상태네요~~~ 패브릭 가방 <b>SAMPLE</b> SALE 들어갑니다.  한 개씩만 있는 제품이구요~~ 재고 소진 후에 주문시에는 SALE 가격 적용 안 되네요~~~^^\*  찜콩 안 되구요~ 반품,교환 안되니깐 많이...  </description>  <bloggername>MEGO</bloggername>  <bloggerlink>http://blog.naver.com/mego72</bloggerlink>  </item>  </channel>  </rss> |

* + 1. Python code

|  |
| --- |
| \_\_author\_\_ = 'KCP' \_\_date\_\_ = '2015-05-06'  **from** urllib **import** parse, request **import** xml.etree.ElementTree **as** et   **def getsearchresult**(target, query):  *"""Return a xml tree or xml root element  Keyword argument:  target -- Request URL  query -- Dictionary for search  """* xmldata = request.urlopen(target+'?'+parse.urlencode(query)).read().decode('utf-8')  xmltree = et.fromstring(xmldata)  **return** xmltree  a = getsearchresult('http://openapi.naver.com/search', {'key': 'a4c87ac3748a15f672bb5f644443115f', 'target': 'blog', 'query': '제주도', 'display': 10, 'start': 1, 'sort': 'sim'}) |

* + 1. 특이사항
  1. Daum
     1. 요청 URL
     2. 요청 변수
     3. 응답 결과
     4. Python code
     5. 특이사항
  2. Twitter
     1. 요청 URL
     2. 요청 변수
     3. 응답 결과
     4. Python code
     5. 특이사항
  3. Tour API
     1. 요청 URL
     2. 요청 변수
     3. 응답 결과
     4. Python code
     5. 특이사항