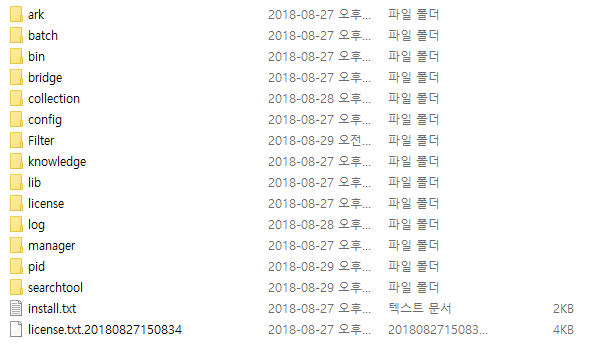
**1. SF-1 엔진 설치 폴더 구성**



1. batch: 사용되는 cmd script

2. bin: cmd나 관리도구에서 실행하는 파일(exe)들을 모아놓은 폴더 // isc(검색기), cmanager(괸리도구), indexer(색인기)

3. bridge: 외부시스템(DB, WEB, LOD(linked open data), file의 외부와 연결다리 역할로서 수집기(스파이더(정보수집기), 크롤러(주기적 데이터 수집)), scd 생성

4. collection: 수집 데이터를 저장하는 공간. type내에서 scd - mode에 따라 다른 디렉토리로 저장 / 색인기를 돌릴 경우 원본 SCD파일은 backup으로 이동한다.

5. config: 수집&검색엔진&관리도구의 기본 경로, 순서, 시스템 설정 (xml로 구성: 구조화로 다양한 설정이 가능하기 때문)

6. Filter: 검색과정에서 필터링이 된 자료

7. Knowledge: NLP를 위한 사전이 수록된 곳, 와이즈넛에서는 기본적으로 중국어, 영어, 일본어, 한국어가 제공된다. 그 외 언어는 UTF-8을 기반으로한 검색기능은 제공되지만, 그 이상의 검색은 사전이 필요하다.

8. lib: queryAPI530와 웹브라우저와 검색엔진 간의 request와 response를 수행하는 API등의 라이브러리를 제공

9. license: 와이즈넛 라이선스 파일, 서버당 하나의 라이선스, 2,000~2,500만원 상당의 가격

10. log: 각종 로그를 기록할 수 있는 곳. 설정에 따라

11. manager: 관리도구 디렉토리 // 기본 설정

**2. DB 수집 Config 및 수집 결과 파일 (SCD)**

* *D:\wisenut\sf-1\config\collection\korchamtest.xml*
* cmd

D:\wisenut\sf-1\batch에서 *bridge.cmd ../config/config.xml db korchamtest static*

□ Source: 수집 대상 설정

- charset: SCD파일의 문자코드(와이즈넛에서는 전 과정을 한글을 위해서 UTF-8로 통일)

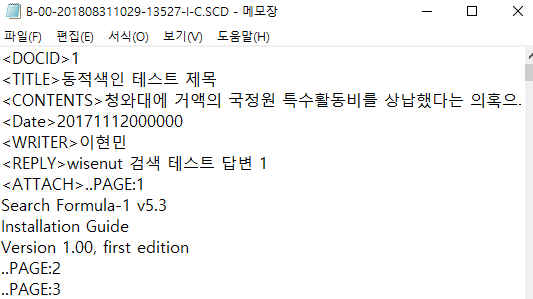
- id: 브리지 실행 시 해당 type내에서 사용

- type: 수집 type (DB, 외부 데이터 등)

□ SCDdir path: 수집한 데이터를 SCD화 하여 저장할 위치를 지정

□ CatalogInfo: 생성하는 SCD의 목록구조를 설정 // 아래의 결과물에서의 순서와 동일하다. DOCID(문서ID), TITLE, CONTENTS, DATE .. 순

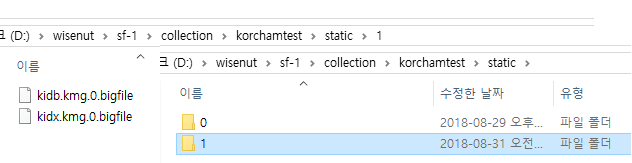
- path에 저장된 결과물



□ SCD Naming: Date-제목의 중복(치명적 오류를 발생)을 막기 위해 ms단위까지 표시한다. 그럼에도 중복이 발생할 수 있는 경우의 수가 있기에, 별도의 index num을 부여한다.

-TYPE: I: Insert(입력), U: Update(수정), D: Delete(삭제), R: Replace(경로변경)

**3. Static 색인 파일 및 현재 타겟 정보**



* cmd

*D:\wisenut\sf-1\batch에서 indexer.cmd ../config/config.xml korchamtest static*

scd를 backup으로 이동 후에 collection/korchamtest/ 내로 색인파일이 저장된다.

□ 색인의 구조(역 인덱스 정보)

책의 뒷 장 인덱스처럼, 데이터 내의 “단어-빈도-해당문서의 위치”로 나타낸 색인의 구조. 색인은 이러한 구조로 되어 있기 때문에, 빠른 속도로 해당 검색어Query를 탐색할 수 있다.

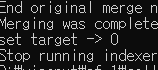
□ Static, Dynamic, Merge 모드

Static: 수집색인에서 정적인 수집과 색인을 진행 (전수 수행) // insert type

Dynamic: 수집색인에서 동적인 수집과 색인을 진행하는 것으로 변동데이터만 수집색인한다. recursive algorithm에서 반복적인 결과값을 저장해두고 반복때마다 사용하는 Dynamic programming의 dynamic과 개념이 비슷하다. // Update, Delete, Replace type

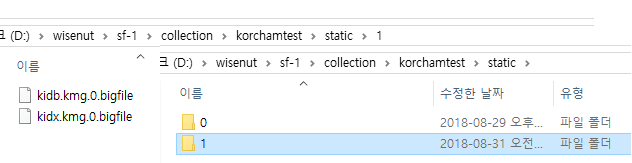
Merge: delete type의 경우, 실제 원본 파일을 삭제하지 않고, 삭제 인덱스만 보내고 기존의 위치에 계속해서 저장한다. 기존의 파일을 삭제할 경우, 저장공간에 빈 공간이 생겨 문제가 생길 수 있기 때문에, 이러한 방식을 채택하였다. 이러한 남아있는 삭제 데이터들은 반복된 dynamic 수집색인 이후 Merge모드로 수집색인을 진행하여 정리한다.(일주일에 1번 정도)

□ Fault tolerance



0과 1로 나뉜 이유는, Fault tolerance(폴트 톨러런스)의 기능이다. 0번과 1번 작업 영역에서 한 영역에서 문제가 발생 시, 다른 영역에서 기존 색인작업 데이터를 쓰기 위한 방법이다. 어떠한 데이터를 0번에서 계속 작업을 하면서, dynamic모드로 많은 양이 쌓였을 경우, Merge 이후 1번 타겟으로 변경하여 작업을 진행한다. 만약 1번 타겟에 버그가 발생하였을 경우, 기존의 0번 타겟을 임시로 사용을 한다.

**3. Static 색인 파일 및 현재 타겟 정보**



* cmd

*D:\wisenut\sf-1\batch에서 indexer.cmd ../config/config.xml korchamtest static*

scd를 backup으로 이동 후에 collection/korchamtest/ 내로 색인파일이 저장된다.

□ 색인의 구조(역 인덱스 정보)

책의 뒷 장 인덱스처럼, 데이터 내의 “단어-빈도-해당문서의 위치”로 나타낸 색인의 구조. 색인은 이러한 구조로 되어 있기 때문에, 빠른 속도로 해당 검색어Query를 탐색할 수 있다.

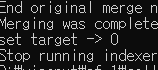
□ Static, Dynamic, Merge 모드

Static: 수집색인에서 정적인 수집과 색인을 진행 (전수 수행) // insert type

Dynamic: 수집색인에서 동적인 수집과 색인을 진행하는 것으로 변동데이터만 수집색인한다. recursive algorithm에서 반복적인 결과값을 저장해두고 반복때마다 사용하는 Dynamic programming의 dynamic과 개념이 비슷하다. // Update, Delete, Replace type

Merge: delete type의 경우, 실제 원본 파일을 삭제하지 않고, 삭제 인덱스만 보내고 기존의 위치에 계속해서 저장한다. 기존의 파일을 삭제할 경우, 저장공간에 빈 공간이 생겨 문제가 생길 수 있기 때문에, 이러한 방식을 채택하였다. 이러한 남아있는 삭제 데이터들은 반복된 dynamic 수집색인 이후 Merge모드로 수집색인을 진행하여 정리한다.(일주일에 1번 정도)

□ Fault tolerance



0과 1로 나뉜 이유는, Fault tolerance(폴트 톨러런스)의 기능이다. 0번과 1번 작업 영역에서 한 영역에서 문제가 발생 시, 다른 영역에서 기존 색인작업 데이터를 쓰기 위한 방법이다. 어떠한 데이터를 0번에서 계속 작업을 하면서, dynamic모드로 많은 양이 쌓였을 경우, Merge 이후 1번 타겟으로 변경하여 작업을 진행한다. 만약 1번 타겟에 버그가 발생하였을 경우, 기존의 0번 타겟을 임시로 사용을 한다.

**4. 통합검색 페이지 UI**

□ 검색의 과정

ISC(Query) -> SC(searcher) -> querying – filtering – sorting – ranking -> return

□ 검색어 처리 method

TOKEN: 특수문자나 띄어쓰기 단위로 문자를 끊어서 단어 검색 -> 영미권 언어나 카테고리 정보에서 활용

NGRAM(LIKE): 문장을 일정 길이에 따라 자르고(2자, 3자.. n자) 단어 파악, 검색 -> 모델명이나, 부품번호와 같은 부분에서 활용

kma: 형태소 분석 + 사전 -> 한국어와 같은 언어에서 의미의 최소단위로 자르고, 정확한 언어 해석에 사용 ex) 대학생선교회

□ 랭킹: 검색이 되어 리턴이 되었다 하여도, 원하는 정보가 있지 않으면 효용성이 하락. 이를 위해 정확한 의미 파악을 위한 형태소 분석과 랭킹정보를 통해 정확한 정보를 리턴

필드가중치: 제목필드와 내용필드에서 검색된 내용에 가중치를 따로 주어 우선순위로 리턴 (보통, 제목 단어가 중요도가 높다)

그 외에 필드포함수, 단어거리 정보, 빈도수 등의 방법이 존재



□ ISC: 검색기 // 검색기는 많은 이유로 자주 꺼질 수 있기 때문에, 실행되어 있는 관리도구(Cmanager)가 자동 재시작해주는 기능을 한다.

> ISC는 업무 분산을 위해 쿼리 프로세서의 멀티 스레드 형식으로 Searcher를 사용한다.

> 포트 설정은 config.xml 내의 SFconfig에서 port 설정

> 수동실행시, d:/wisenut/sf-1/batch ../config/config.xml test

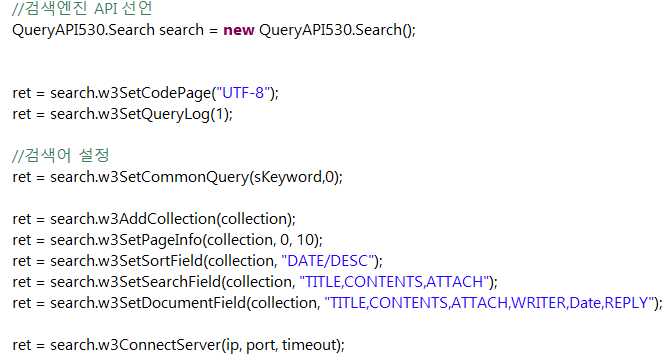
□ 검색엔진의 흐름: Webbrowser API에서 검색어입력 -> 검색엔진에 request -> API에서 색인데이터를 Response 객체에 담아 리턴 -> http화면에 출력(UI 구성)

□ TOMCAT 설치: PC를 개인서버화하여 운영할 수 있도록 하는 오픈소스 WAS. 이클립스를 통해 SERVER를 열고, localhost:8080(설정포트)를 통해 페이지에 접근할 수 있다.

New – Project – Web – Dynamic Web Project(동적페이지: 동적으로 움직이는 페이지) - 톰캣 폴더 경로 설정

Server Start – Work Space내에서 UI수정 (JSP)

Workspace 내에서 KORCHAM – WebContent – search\_page.jsp (JSP 수정 // <% 자바코드 %> + HTML)



□ sf-1/lib 내의 QueryAPI530를 search 인스턴스로 호출 (검색어를 받아 색인DB로 연결하는 역할)

SetCodePage//QueryLog(1): UTF-8방식과 검색어 기록유무 method

SetSortField: 정렬방식 method / SearchField: 검색어가 검색할 필드(TITLE만 있을 시, TITLE만 검색)

DocumentField: 문서의 필드를 정하는 Method(\*항목마다 띄어쓰기가 될 경우 버그가 발생한다.)

ConnectServer: (127.0.0.1(ip), 7000(설정된 포트), 5\*1000(ms))

□ EMB000021700b7e

Search\_result (검색어 검색 결과) 내용에서 ./result/result\_content.jsp를 include하여 해당 내용을 출력

□ result\_content.jsp // for 문을 통해서 내용 출력(result count(i) 루프) // html태그 내에 JAVA코드 삽입 (<%= %>)

<dd class=*"lh18"*> TITLE :<%=search.w3GetField(collection, "TITLE", i) %> </dd>