// 데이터 처리

데이터 저장 형식 : json 도큐먼트

## 1. REST API

RESTful : 자원별 고유 URL, http 메서드 GET, POST, PUT, DELETE

6.x 이전 : http:// 호스트 : 포트 / 인덱스 / 도큐먼트 타입 / 도큐먼트 id

7.0 이후 : http:// 호스트 : 포트 / 인덱스 / _doc / 도큐먼트 id
고정자

## 2. CRUD

PUT : 입력

"result" : "created"

→ 이미 있는 id에 또 입력 시 덮어쓰고 "result" : "updated"

_doc 대신 _create : 중복 id 덮어쓰기 X 오류 O

GET : 가져오기

URL 입력하면 내용이 _source 에 나타남

DELETE : 삭제

① 도큐먼트 단위

return : "result" : "deleted"

인덱스 남아있어 GET 하려고 함 : "found" : false

② 인덱스 단위

return : "acknowledged" : true

인덱스 없는데 GET : "type" : "index_not_found_exception"

"status" : 404

POST : 수정 (PUT은 안됨)

- 인덱스 / _doc 까지만 입력 → 도큐먼트 id 자동생성
- 인덱스 / _update / 도큐먼트 id
  : 원하는 필드의 내용만 업데이트
  doc 지정자 ex) { "doc" : { "message" : "안녕하세요" } }
  _version 의 값 증가

* 실제로 내부에서는 전체 내용 가져와서 새로 PUT

## 3. _bulk API

따로따로 하는 것보다 속도가 빠르다

index, create, update, delete (내용X, 명령문만)
↳ 명령문과 데이터문을 한줄씩 순서대로 입력

ex) POST _bulk
{ "index" : { "_index" : "test", "_id" : "1" } }
{ "field" : "value one" }
⋮

파일에 저장된 내용 실행
→ 명령을 .json 파일로 저장하고 curl

$ curl -XPOST "http://localhost:9200/_bulk" -H 'Content-Type: application/json' ~ json' --data-binary @bulk.json

## 4. _search API

쿼리를 통한 검색 기능 : GET 인덱스명/_search

↳ 쿼리 입력, 없을 시 match_all (전체검색)

- URI 검색

GET 인덱스명/_search?q=검색어 → field:value

field 지정해 검색하는 것이 좋다.

[ hits.total.value ⇒ 문서 개수
hits.hits:[ ] ⇒ 정확도(relevancy) 높은 문서 10개

AND, OR, NOT 사용가능

- 데이터 본문 (Data Body) 검색

검색 쿼리를 데이터 본문으로 입력.

Elasticsearch의 QueryDSL 사용, json 형식 → match 쿼리 가장 많이 씀

ex) GET test/_search

```
{
  "query" : {
            "match" : {
                      "field" : "value"
                      }
            }
}
```

- 멀티테넌시 (Multitenancy)

여러 인덱스를 한꺼번에 묶어서 검색

쉼표, 로 나열하거나 와일드카드 * 문자로 묶는다.

// 검색과 쿼리 - Query DSL (Domain Specific Language)

텀(Term)으로 분석 과정 거쳐 저장 → 풀 텍스트 검색 (Full Text Search)

- Full Text Query

match_all : 해당 인덱스의 모든 도큐먼트

match : field에 value가 포함된 도큐먼트

→ 여러개 넣을시 default OR ⇒ operator 옵션 설정 가능

match_pharse : 공백 포함 정확히 일치하는 도큐먼트

⇒ slop으로 다른 단어 끼어들기 개수 설정

query_string : URI 검색의 루씬 문법

## • Bool 복합 Query

must : 쿼리가 참인 도큐먼트 검색 → AND, OR과 비슷하지만 같지는 X

must_not : 쿼리가 거짓인 도큐먼트 검색

should : 검색결과 중 이 쿼리에 해당하는 도큐먼트 점수↑

filter : 쿼리가 참인 도큐먼트 검색, but 스코어 계산 X

must 보다 속도↑ 캐싱 가능

## • Relevancy

Score : 검색 조건과 얼마나 일치하는지

(Best Matching) BM25 알고리즘

: $score(D, Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})}$

① TF (Term Frequency)

: 도큐먼트 내에 검색된 term 많을수록 점수↑

② IDF (Inverse Document Frequency)

: 검색한 term 가진 도큐먼트 많을수록 (희소성↓) 점수↓

③ Field Length

: 짧은 필드(ex. 제목)에 포함된 term 점수↑

## Exact Value Query

Full Text : score 기반으로 relevancy 높은 결과

↔ 정확값 (Exact Value) : 검색 조건의 참/거짓 여부만 판단.

bool 쿼리의 filter 내부에서 사용하면 스코어에 영향 X

└ 검색에 조건은 추가하지만 스코어에는 영향 없도록 제어

ex) 쇼핑몰에서 키워드 검색하며 생산업체 필터링

keyword : 문자열은 keyword 형식으로 저장하면 정확값 검색