

엘라스틱서치 이해하기

Moon Yong Joon

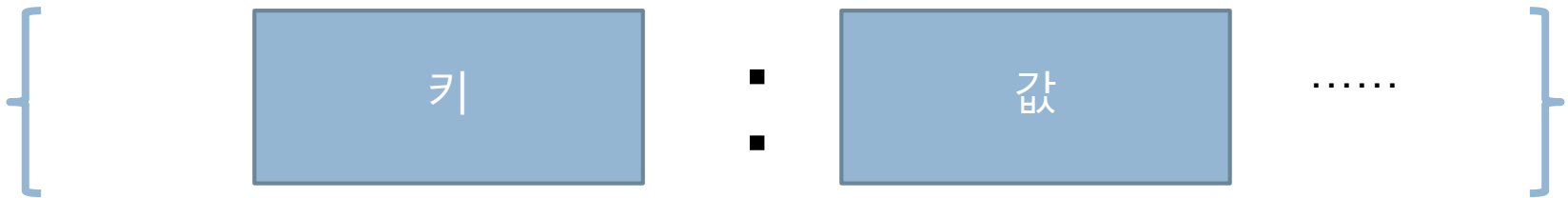
기초



JSON

JSON

JSON은 키값쌍으로 데이터를 보관하며 키는 해쉬처리가 되는 유일한 값이 되어야 함



JSON 스타일

JSON은 키와 값을 구성함 값에는 문자열, 숫자, 객체가 들어갈 수 있음

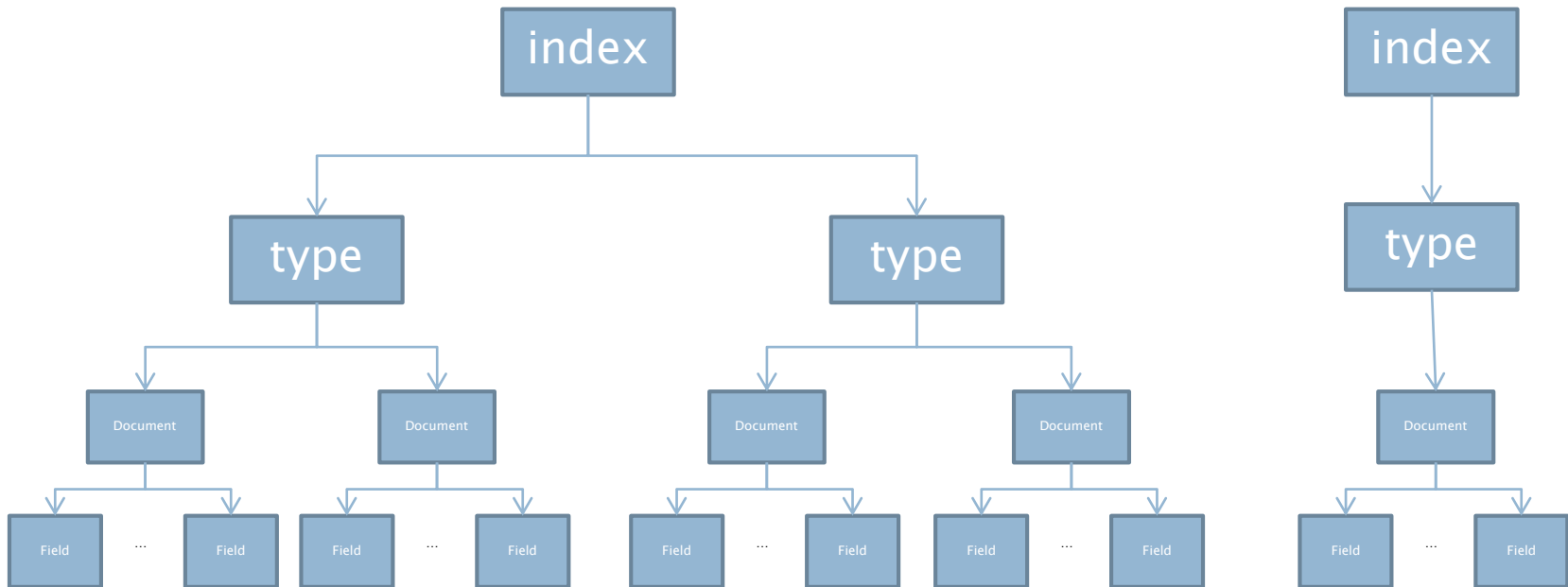
```
{
  "email": "john@smith.com",
  "first_name": "John",
  "last_name": "Smith",
  "info": {
    "bio": "Eco-warrior and defender of the weak",
    "age": 25,
    "interests": [ "dolphins", "whales" ]
  },
  "join_date": "2014/05/01"
}
```



데이터 구조 기본

데이터구조

데이터는 index, type, document, field로 구성



데이터 구조

RDB와 비교한 데이터 구조

| RDB | Elasticsearch |
|----------|---------------|
| DATABASE | INDEX |
| TABLE | TYPE |
| ROW | DOCUMENT |
| COLUMN | FIELD |
| SCHEMA | MAPPING |

데이터구조 : 예시

데이터 구조에 대해 elasticsearch-head 도구로 조회

The screenshot shows the Elasticsearch Head web interface. At the top, the URL is `http://localhost:9200/` and the cluster health is **yellow (5 of 10)**. The left sidebar contains navigation tabs: Overview, Indices, and Browser. The 'Browser' tab is active, showing a list of indices (All Indices, books) and types (book, itbook). The main panel displays search results for the 'books' index. The search bar shows 'Structured Query [+]' and 'Any Request [+]'.

Search results summary: Searched 5 of 5 shards, 6 hits, 0.002 seconds

| _index | _type | _id | _score | title | date | pages | category | price |
|--------|--------|-----|--------|---------------------|------------|-------|----------|-------|
| books | book | 25 | 1 | 123 | | | | |
| books | book | 12 | 1 | xxx | | | | |
| books | book | 10 | 1 | Elasticsearch Guide | 2014-05-01 | 300 | ICT | |
| books | itbook | 1 | 1 | big data | | 300 | | 30000 |
| books | book | 1 | 1 | Elasticsearch Guide | 2014-05-01 | 5000 | ICT | |
| books | book | 11 | 1 | Elasticsearch Guide | 2014-05-01 | 300 | ICT | |



Curl 처리

데이터 처리방법

elasticsearch에 HTTP 프로토콜을 이용해 데이터 생성, 조회, 변경, 삭제

```
curl -X{메소드} http://host:port/{인덱스}{타입}/{다큐먼트 id} -d "{데이터}"
```

| Method | 설명 |
|--------|-----------------------|
| PUT | 특정 id에 대한 새로운 다큐먼트 생성 |
| POST | 특정 id에 대한 다큐먼트 갱신 |
| GET | 특정 id에 대한 새로운 다큐먼트 조회 |
| DELETE | 다큐먼트 등 데이터 구조 삭제 |
| HEAD | 저장된 다큐먼트에 대해 조회 |



크롬(postman)

크롬 내의 설치

Chrome 웹 스토어

스토어 검색

추천

앱

게임

확장 프로그램

테마

유형

Chrome 앱

웹사이트

카테고리

전체

특성

오프라인 실행 가능

Google 제공

무료

Android에서

Google 드래

평점

★★★★★

★★★★★

★★★★★

★★★★★

이 사용자가 함께 설치한 앱

Postman
www.getpostman.com 에서 제공
★★★★★ (6617) | 개발자 도구 | 사용자 2,041,254명

개요 | 리뷰 | 지원 | 관련 프로그램

Build APIs faster

GET Request

GET Request +

The HTTP GET request method is meant to retrieve data from a server. The data is identified by a unique URI (Uniform Resource Identifier). A GET request can pass parameters to the server using "Query String Parameters". For example, in the following request:

http://www.example.com/fetchers/handwritten

The parameter "hand" has the value "written".

This endpoint returns the HTTP headers, request parameters and the complete URI requested.

GET https://echo.getpostman.com/gettest=123 Params Send Save

Authorization Headers Body Pre request Script Tests Generate Code

Type No Auth

Body Cookies Headers (12) Tests (404) Status: 200 OK Time: 1047 ms

Pretty Raw Preview JSON Save Response

1-6
2-4
3
4
5-1
"args": {
 "test": "123"
},
"headers": {

오프라인 실행 가능

기기와 호환 가능

Supercharge your API workflow with Postman! Build, test, and document your APIs faster. More than a million developers already do....

Supercharge your API workflow with Postman!

Build, test, and document your APIs faster. More than a million developers already do.

The idea for Postman arose while the founders were working together, and were

웹사이트

악용사례 신고

추가 정보

버전: 4.3.1

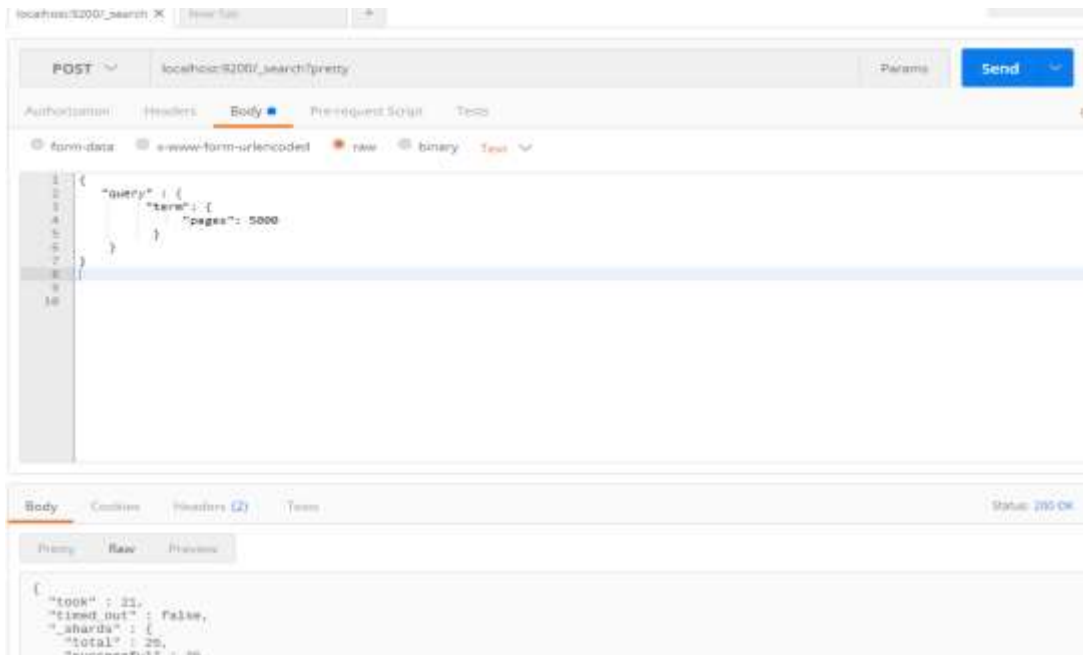
업데이트 날짜: 2016년 6월 8일

크기: 6.32MiB

언어: English

데이터 처리방법

http 메소드를 세팅하고 주소창에 url를 설정하고 raw 데이터에 json으로 입력해서 처리



Get 메소드에서 json 데이터를 넣을 수가 없어 post 메소드로 처리

Windows 처리시 주의사항

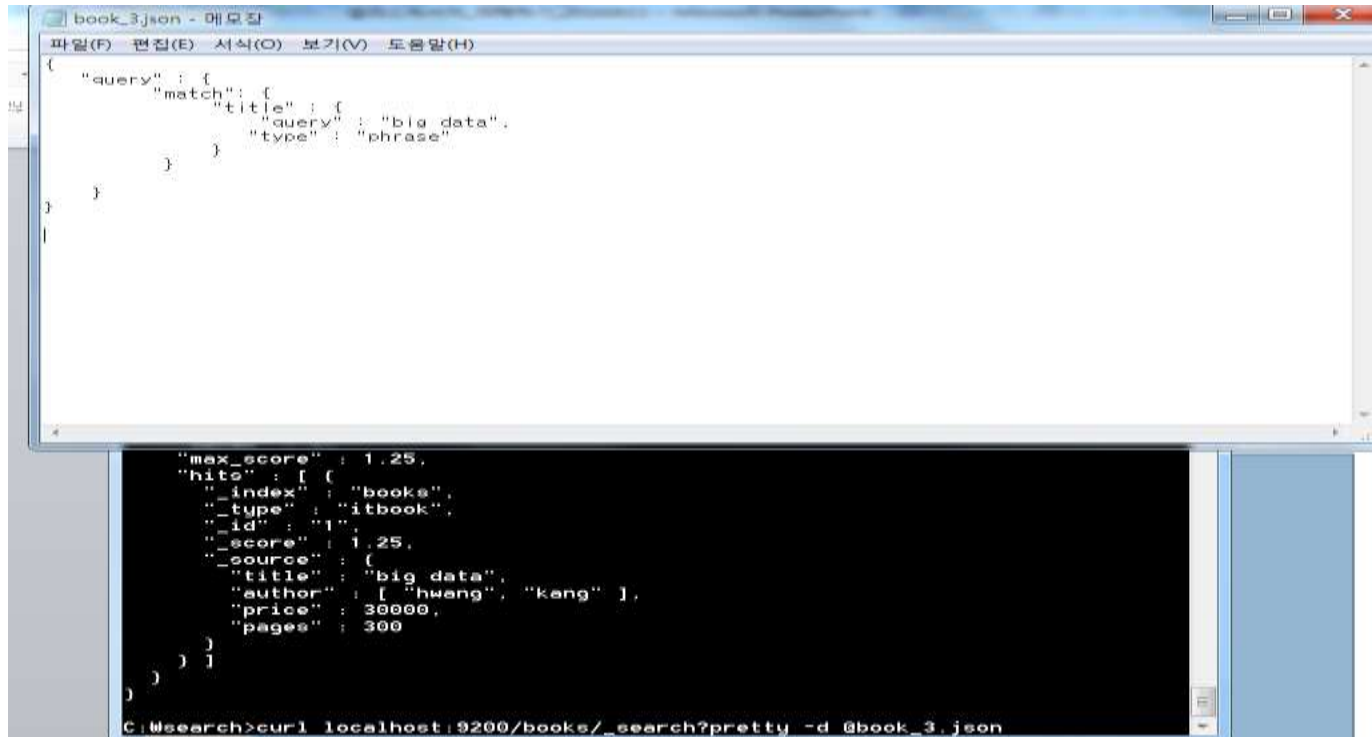
Window 처리

Command 창에서 “을 \”로 변환해서 처리하면 됨

파일처리시에는 \”를 “ 로 처리해야 함

Window 처리 : 파일이용

파일로 처리하면 windows 커맨드에서 \”를 사용하지 않아도 됨



```
book_3.json - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

{
  "query": {
    "match": {
      "title": {
        "query": "big data",
        "type": "phrase"
      }
    }
  }
}

"max_score" : 1.25,
"hits" : [ (
  "_index" : "books",
  "_type" : "itbook",
  "_id" : "1",
  "_score" : 1.25,
  "_source" : {
    "title" : "big data",
    "author" : [ "hwang", "keng" ],
    "price" : 30000,
    "pages" : 300
  }
) ]
)

C:\Wsearch>curl localhost:9200/books/_search?pretty -d @book_3.json
```

Window 처리 : 생성

Command 창에서 “을 \”로 변환해서 처리하면
됨

#실제 값을 입력

```
C:\>curl -XPUT http://localhost:9200/books/book/1 -d "{ \"title\": \"Elasticsearch Guide\", \"author\": \"kim\", \"date\": \"2014-05-01\", \"pages\": 250 }"
```

#처리결과

```
{\"_index\":\"books\", \"_type\":\"book\", \"_id\":\"1\", \"_version\":1, \"_shards\":{\"total\":2, \"successful\":1, \"failed\":0}, \"created\":true}
```


Window 처리 : 조회

조회

```
#조회값 입력
C:\>curl -XGET http://localhost:9200/books/book/1?pretty=true
{
  "_index" : "books",
  "_type" : "book",
  "_id" : "1",
  "_version" : 1,
  "found" : true,
  "_source" : {
    "title" : "Elasticsearch Guide",
    "author" : "kim",
    "date" : "2014-05-01",
    "pages" : 250
  }
}
```

FIELD

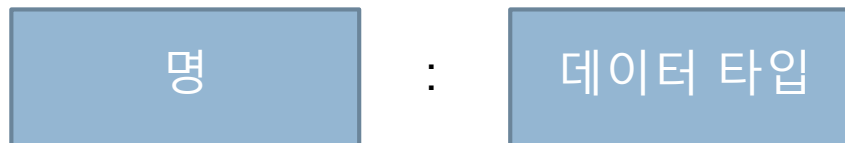
Moon Yong Joon



Field 구조

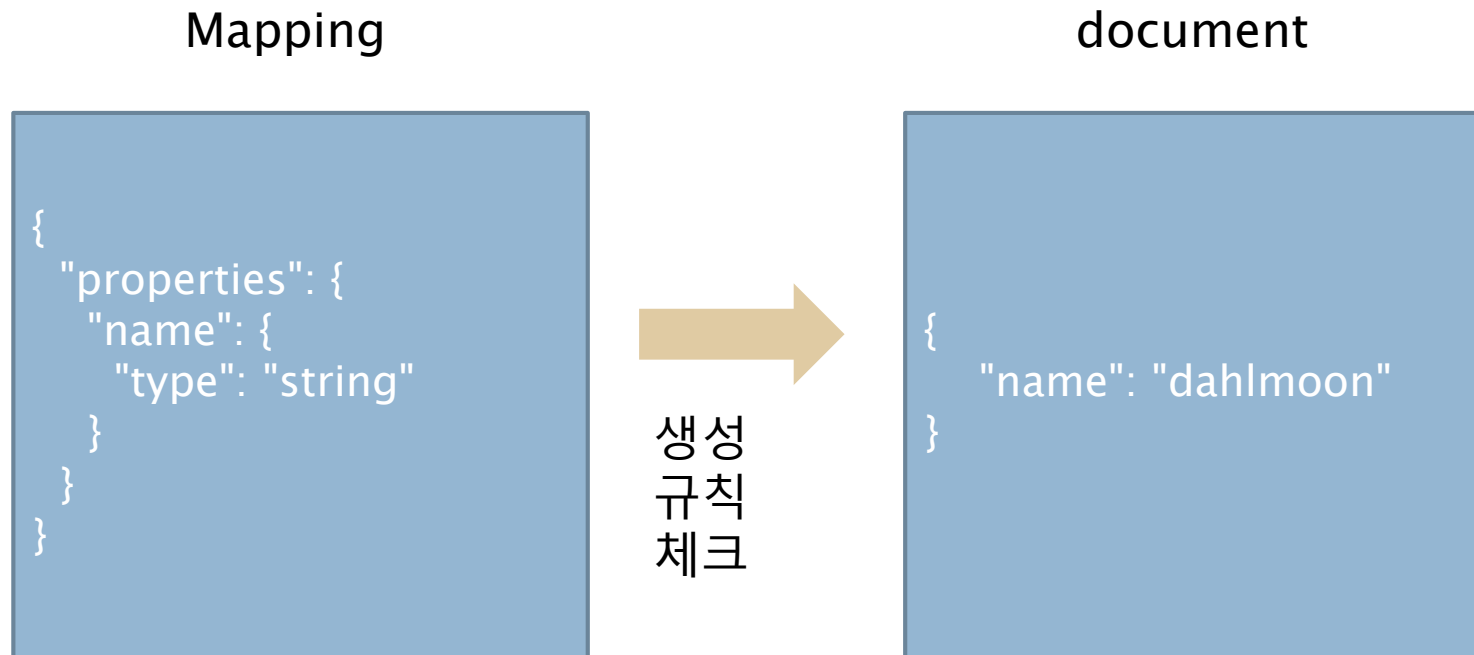
Field 구조

필드는 키와 값으로 구성되며 키에는 이름, 값에는 데이터 타입이 정의(Mapping)되고 실제 저장시에 해당 데이터 타입을 체크함



Field : Mapping과 Document

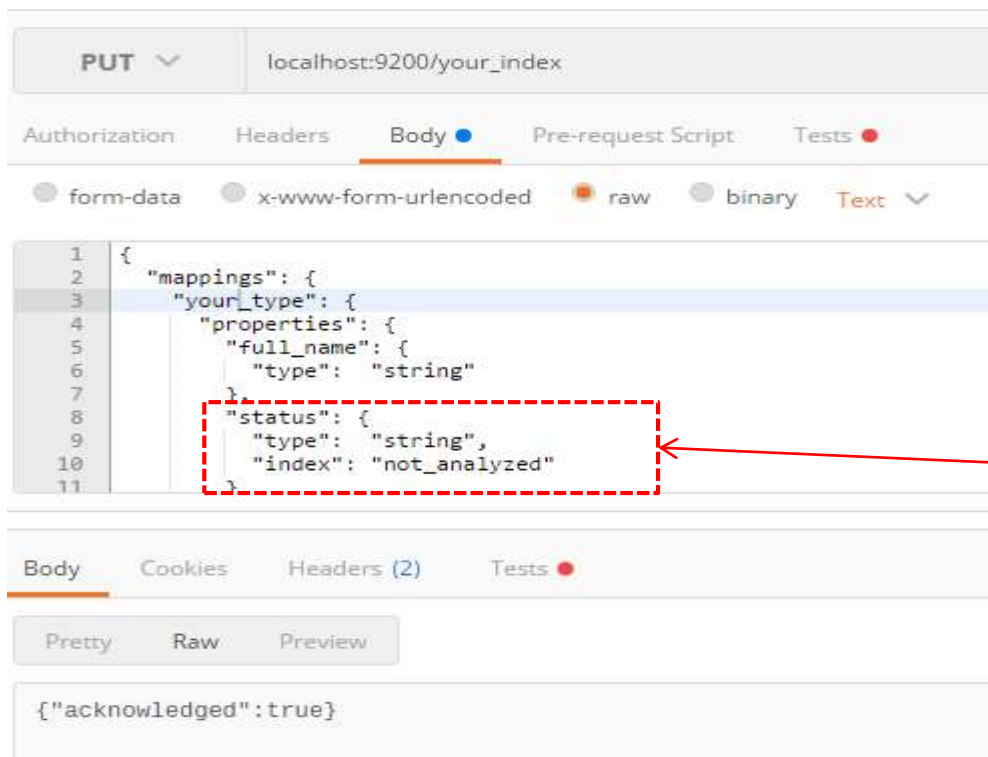
Type 에 정의된 mapping은 document를 생성하는
규약이므로 실제 정의된 것대로 다큐먼트가 생성되
어야 함



String 데이터 타입

String 정의 : index/type 생성

Index와 type 내의 full_name과 status 생성
full_name은 full text(analyzed) 처리하고 status
는 keyword(not_analyzed) 처리



Status 필드는 분석
하지 않도록 지정 하
나의 keyword로 인
식

String 정의 : 다큐먼트 생성

Index와 type 내의 full_name과 status 생성

The screenshot displays a REST client interface with a PUT request to `localhost:9200/your_index/your_type/1`. The request body is a JSON object with `full_name` and `status` fields. The response shows a 201 status with a JSON object containing index and type information.

Request:

```
PUT localhost:9200/your_index/your_type/1
```

Body:

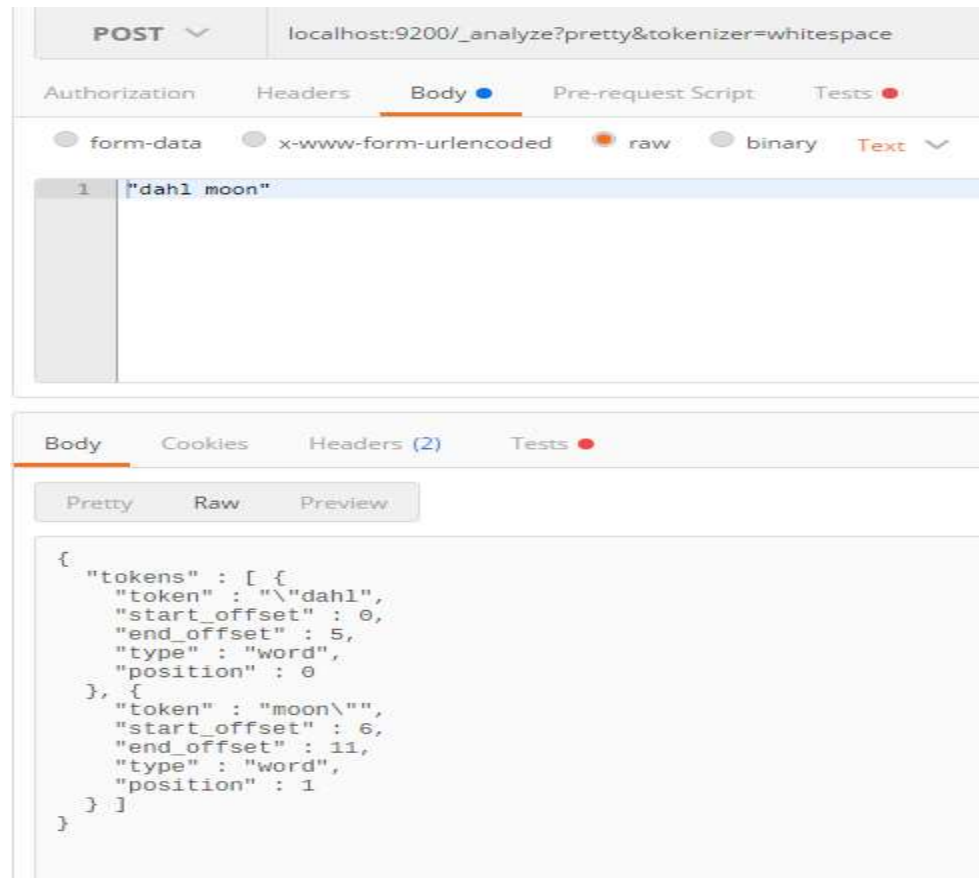
```
{
  "full_name": "dahl moon",
  "status": "Dahl Moon "
}
```

Response:

```
{ "_index": "your_index", "_type": "your_type", "_id": "1", "_version": 1, "_shards": { "total": 2, "successful": 1, "failed": 0 }, "created": true }
```

String 정의 : analyze

_analyze api를 이용해서 문자열을 분석



String의 옵션

String 필드에 아래의 옵션이 있으므로 mapping 정의시 세팅 필요

| | |
|-----------------|--|
| store | _source 에 데이터 저장을 하지 않을때 사용 |
| index | analyzed : 필드에 분석기 적용, not_analyzed : 필드에 분석기를 사용하지 않을 때 적용, No : 필드를 색인하지 않을 경우 |
| boost | 필드에 가중치 부여해서 검색결과의 우선순위에 영향 |
| null_value | 해당 필드가 없을 경우 기본값을 설정 |
| analyzer | 데이터 색인과 문자열검색에 사용될 분석기 지정, index 옵션에 analyzed가 지정된 경우 사용 |
| Index_analyzer | 데이터 색인에 사용될 분석기 지정 |
| search_analyzer | 문자열 검색에 사용될 분석기 지정. 데이터가 인덱스에 색인된 이후에도 변경가능 |
| include_in_all | _all 매핑필드가 적용된 경우 색인여부를 지정 |
| ignore_above | 지정된 값보다 큰 크기의 토큰의 문자열은 색인하지 않음 not_analyzed때 유용 |



Numeric 데이터 타입

Numeric 정의

필드를 정의하고 type에 integer/float으로 지정

```
PUT my_index
{
  "mappings": {
    "my_type": {
      "properties": {
        "number_of_bytes": {
          "type": "integer"
        },
        "time_in_seconds": {
          "type": "float"
        }
      }
    }
  }
}
```

| | |
|---------|---|
| long | A signed 64-bit integer with a minimum value of -2^{63} and a maximum value of $2^{63}-1$. |
| integer | A signed 32-bit integer with a minimum value of -2^{31} and a maximum value of $2^{31}-1$. |
| short | A signed 16-bit integer with a minimum value of $-32,768$ and a maximum value of $32,767$. |
| byte | A signed 8-bit integer with a minimum value of -128 and a maximum value of 127 . |
| double | A double-precision 64-bit IEEE 754 floating point. |
| float | A single-precision 32-bit IEEE 754 floating point. |

Number

숫자는 정수인 `byte`, `short`, `integer`, `long`과 실수인 `float`, `double`을 지원

| | |
|------------------|--|
| store | _source 에 데이터 저장을 하지 않을때 사용 |
| index | No : 필드를 색인하지 않을 경우 |
| boost | 필드에 가중치 부여해서 검색결과의 우선순위에 영향 |
| null_value | 해당 필드가 없을 경우 기본값을 설정 |
| include_in_all | _all 매핑필드가 적용된 경우 색인여부를 지정 |
| ignore_malformed | false 지정일 경우 숫자필드에 잘 못된 값이 들어오면 에러처리 true 일 경우는 강제로 저장 |
| coerce | false 지정일 경우 숫자가 아니면 오류 true일 경우 문자열은 숫자로 변환 , 정수필드에 실수가 들어오면 정수로 변환 |



Date 데이터 타입

Date 의 기본 정의

필드를 정의하고 type에 boolean으로 지정

```
PUT my_index
{
  "mappings": {
    "my_type": {
      "properties": {
        "date": {
          "type": "date"
        }
      }
    }
  }
}
```

Date 의 multi 필드 정의

Format 필드에 세부적인 date 요건을 표시

```
PUT my_index
{
  "mappings": {
    "my_type": {
      "properties": {
        "date": {
          "type": "date",
          "format": "yyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch_millis"
        }
      }
    }
  }
}
```

Date(날짜) : symbol

년(yyyy), 월(mm), 일(dd), 시(hh), 분(mm), 초(ss), 밀리초(SSS) 단위로 설정

| | | | |
|---|-----------------|---|-------------|
| G | 서기-AD/BC | A | 오전/오후 AM/PM |
| C | 세기 | h | 시간 - 12 |
| Y | 년도 | H | 시간- 24 |
| w | 년도 기준이 주 | k | 시계시간(1~24) |
| e | 요일(숫자) - 1은 월요일 | m | 분 |
| E | 요일(텍스트)- Mon | s | 초 |
| y | 년도 | S | 밀리초 |
| M | 월 | z | 타임존-UTC |
| d | 일- 달기준 | Z | 타임존= +000 |
| D | 일 - 년기준 | | |

Date 의 문서 생성 및 검색

필드를 정의하고 type에 boolean으로 지정

```
PUT my_index/my_type/1  
{ "date": "2015-01-01" }
```

This document uses a plain date.

```
PUT my_index/my_type/2  
{ "date": "2015-01-01T12:10:30Z" }
```

This document includes a time.

```
PUT my_index/my_type/3  
{ "date": 1420070400001 }
```

This document uses milliseconds-since-the-epoch.

```
GET my_index/_search  
{  
  "sort": { "date": "asc" }  
}
```

Note that the sort values that are returned are all in milliseconds-since-the-epoch.

Asc 대로 sort
처리

Date(날짜)

숫자는 정수인 byte, short, integer, long과 실수인 float, double을 지원

| | |
|------------------|-----------------------------|
| store | _source 에 데이터 저장을 하지 않을때 사용 |
| index | No : 필드를 색인하지 않을 경우 |
| boost | 필드에 가중치 부여해서 검색결과의 우선순위에 영향 |
| null_value | 해당 필드가 없을 경우 기본값을 설정 |
| include_in_all | _all 매핑필드가 적용된 경우 색인여부를 지정 |
| ignore_malformed | true 일 경우는 강제로 저장 |
| format | 입력된 날짜 형식을 설정 |



Bool 데이터 타입

Boolean 의 정의

필드를 정의하고 type에 boolean으로 지정

```
PUT my_index
{
  "mappings": {
    "my_type": {
      "properties": {
        "is_published": {
          "type": "boolean"
        }
      }
    }
  }
}
```

| | |
|--------------|---|
| False values | false, "false", "off", "no", "0", "" (empty string), 0, 0.0 |
| True values | Anything that isn't false. |

Boolean 의 문서 생성 및 검색

필드를 정의하고 type에 boolean으로 지정

```
PUT my_index/my_type/1
{
  "is_published": true
}

GET my_index/_search
{
  "query": {
    "term": {
      "is_published": 1
    }
  }
}
```

조회시 true 일
경우 1로 처리

Boolean

불린은 true와 false 둘중에 하나만 가능

| | |
|----------------|-----------------------------|
| store | _source 에 데이터 저장을 하지 않을때 사용 |
| index | No : 필드를 색인하지 않을 경우 |
| boost | 필드에 가중치 부여해서 검색결과의 우선순위에 영향 |
| null_value | 해당 필드가 없을 경우 기본값을 설정 |
| include_in_all | _all 매핑필드가 적용된 경우 색인여부를 지정 |



Binary 데이터 타입

Binary 정의 및 문서 생성

Binary 타입은 binary 값만 처리

```
PUT my_index
{
  "mappings": {
    "my_type": {
      "properties": {
        "name": {
          "type": "string"
        },
        "blob": {
          "type": "binary"
        }
      }
    }
  }
}

PUT my_index/my_type/1
{
  "name": "Some binary blob",
  "blob": "U29tZSBiaW5hcnkgYmxvYg=="
}
```

binary

이미지 등과 같은 base64 형식의 binary 데이터
가능 binary 필드는 색인이 되지 않으며 검색도
불가능

| | |
|--------------------|-----------------------------|
| store | _source 에 데이터 저장을 하지 않을때 사용 |
| compress | true 일 경우 압축 |
| compress_threshold | 입력된 값보다 클 경우 압축 |



object type

객체(object)의 형태

객체 타입의 값을 저장 객체는 데이터와 기능을 가지는 자료구조 단, 기능을 포함하지 않은 객체를 저장함

```
{  
  "필드명" : {  
    "필드명" : 값  
    "필드명" : 값  
  }  
}
```

객체(object) 타입 생성

Manager 필드는 객체 타입, name도 inner 객체 타입

```
PUT my_index
{
  "mappings": {
    "my_type": {
      "properties": {
        "region": {
          "type": "string",
          "index": "not_analyzed"
        },
        "manager": {
          "properties": {
            "age": { "type": "integer" },
            "name": {
              "properties": {
                "first": { "type": "string" },
                "last": { "type": "string" }
              }
            }
          }
        }
      }
    }
  }
}
```

```
PUT my_index/my_type/1
{
  "region": "US",
  "manager": {
    "age": 30,
    "name": {
      "first": "John",
      "last": "Smith"
    }
  }
}
```

객체(object)

객체 타입의 값을 저장 객체는 데이터와 기능을 가지는 자료구조 단, 기능을 포함하지 않은 객체를 저장함

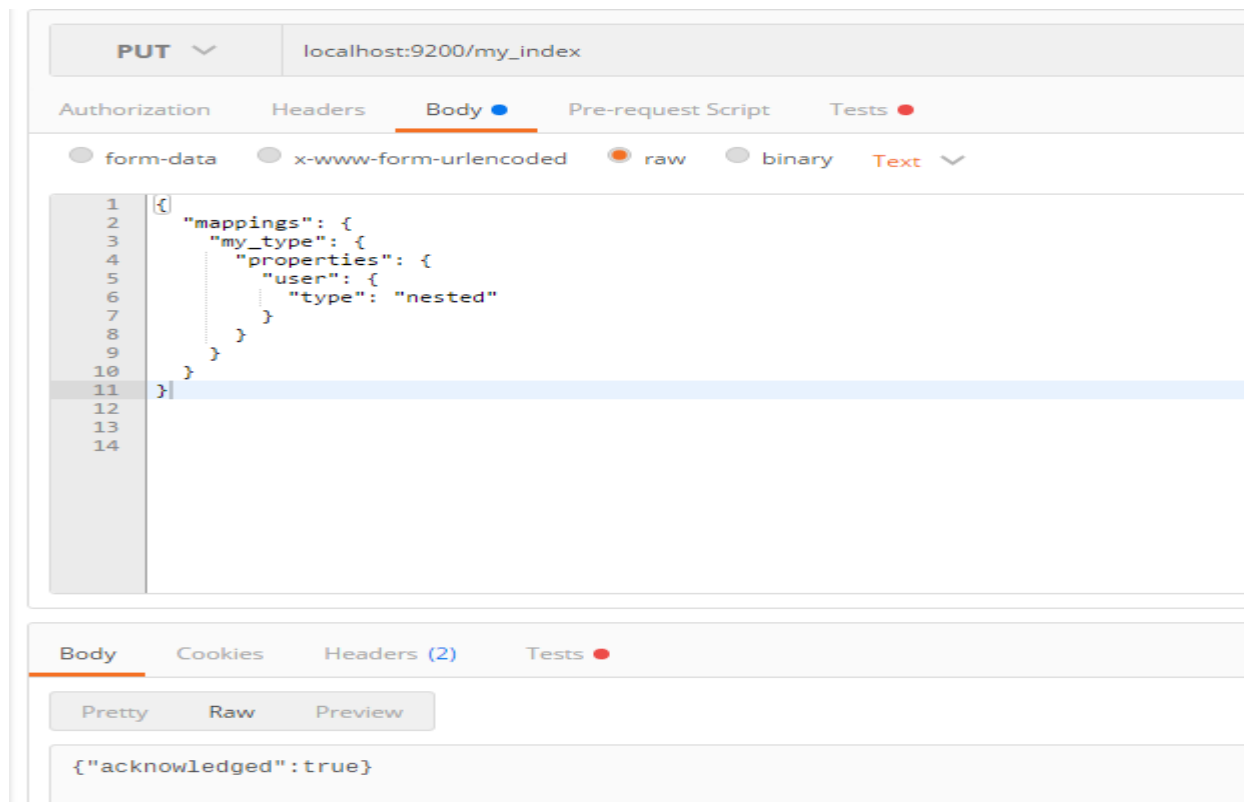
| | |
|----------------|--|
| dynamic | true : 동적변경을 허용. 정의되지 않는 것을 처리함 정의되지 않은 것은 검색 불가 false: 정의되지 않는 것은 처리하지 않음 strict: 정의되지 않은 것은 입력조차 불가능 |
| enabled | false 일 경우 색인하지 않은 데이터 입력시 특정 필드 배제용으로 사용하면 유용 |
| include_in_all | _all 매핑필드가 적용된 경우 색인여부를 지정 |

A horizontal decorative bar consisting of an orange rectangle on the left and a blue rectangle on the right.

nested object type

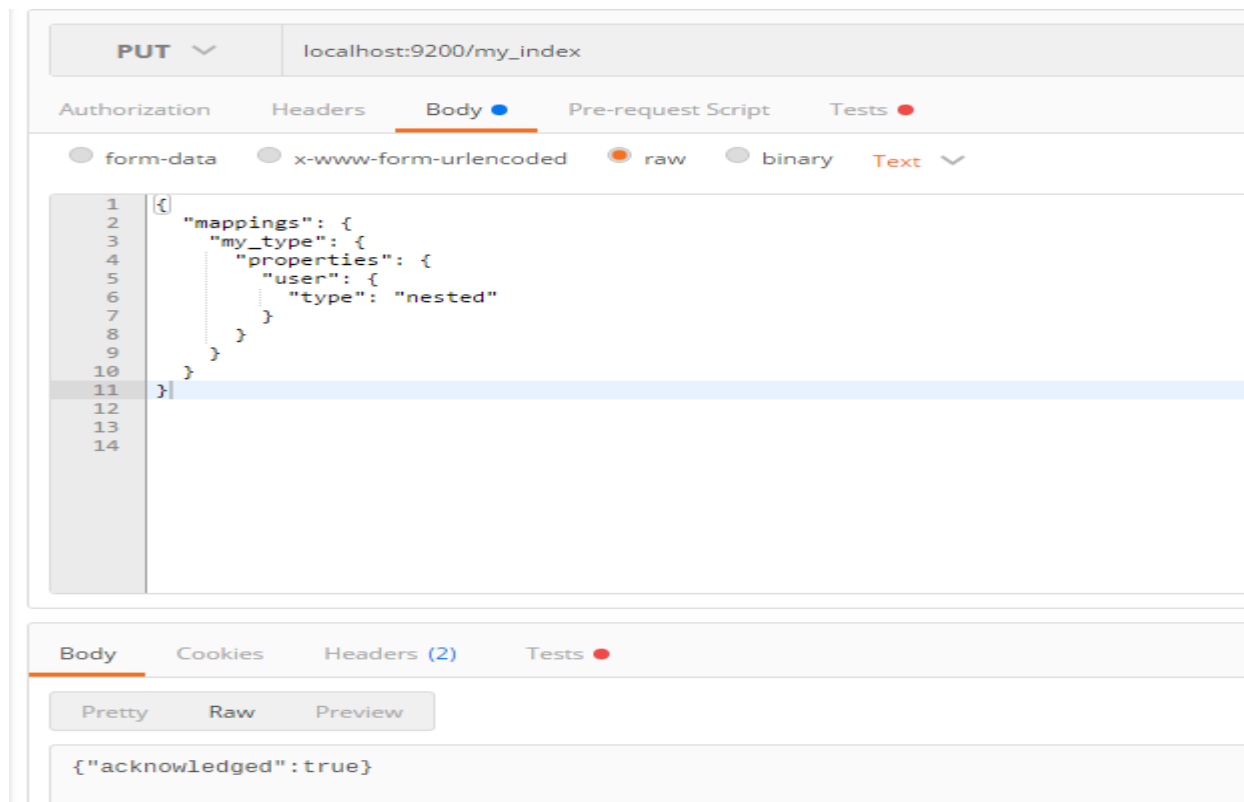
중첩(nested) 정의

Mapping 정의시 중첩을 보관할 field 내의 type 에 nested로 저장해야 한다



중첩(nested) 정의

Mapping 정의시 중첩을 보관할 field 내의 type 에 nested로 저장해야 한다



중첩(nested)의 형태

객체 타입과 유사하지만 독립된 데이터로 유지해 저장하는 타입을 nested라고 함

```
{  
  "필드명" : [{  
    "필드명" : 값  
    "필드명" : 값  
  },  
  {  
    "필드명" : 값  
    "필드명" : 값  
  },  
]  
}
```

중첩(nested) 다큐먼트 생성

Mapping 정의시 중첩을 보관할 field 내의 type 에 nested로 저장해야 한다

The screenshot shows a REST client interface with a PUT request to `localhost:9200/my_index/my_type/1`. The request body is a nested JSON document:

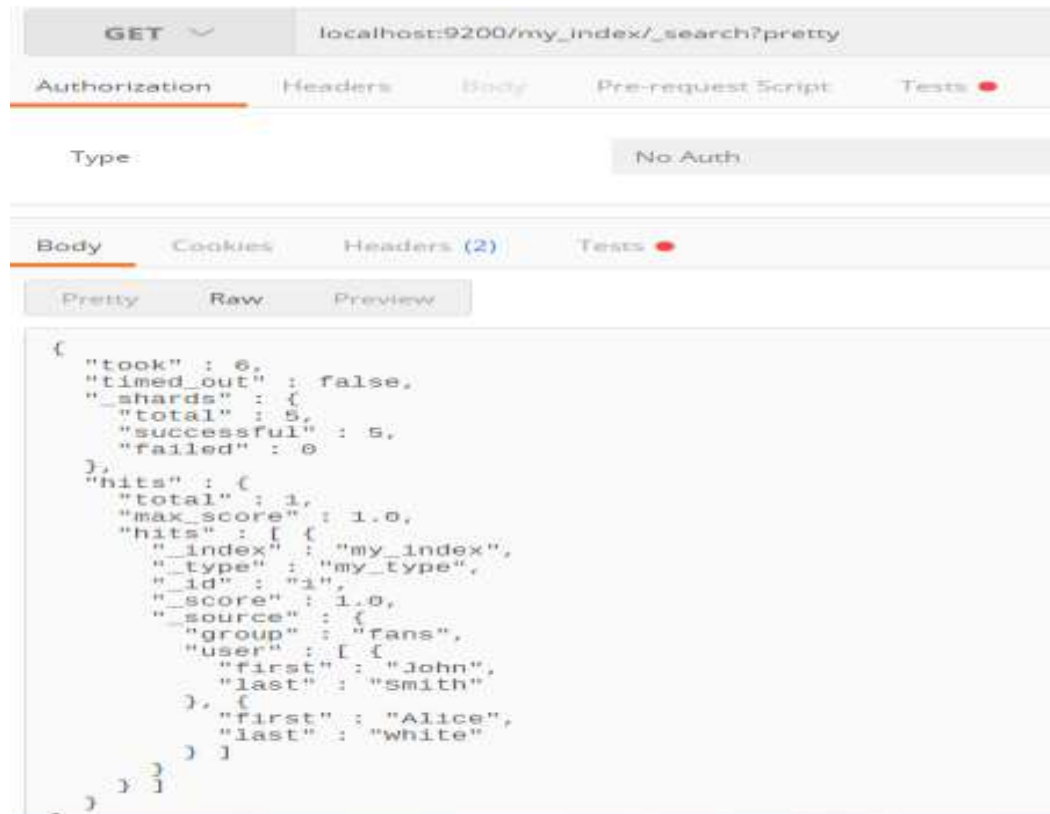
```
1 {  
2   "group" : "fans",  
3   "user" : [  
4     {  
5       "first" : "John",  
6       "last" : "Smith"  
7     },  
8     {  
9       "first" : "Alice",  
10      "last" : "White"  
11    }  
12  ]  
13 }  
14  
15  
16
```

The response status is 201 Created. The response body is a JSON object:

```
{ "_index": "my_index", "_type": "my_type", "_id": "1", "_version": 1, "_shards": { "total": 2, "successful": 1, "failed": 0 }, "created": true }
```

중첩(nested) 다큐먼트 조회

중첩 객체에 대한 조회



The screenshot displays a REST client interface with the following details:

- Method:** GET
- URL:** localhost:9200/my_index/_search?pretty
- Authorization:** No Auth
- Body Tab:** Active, showing a pretty-printed JSON response.

The JSON response is as follows:

```
{
  "took" : 6,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "my_index",
      "_type" : "my_type",
      "_id" : "1",
      "_score" : 1.0,
      "_source" : {
        "group" : "fans",
        "user" : [ {
          "first" : "John",
          "last" : "Smith"
        }, {
          "first" : "Alice",
          "last" : "White"
        } ]
      }
    } ]
  }
}
```

중첩(nested) 객체 검색

중첩 객체 내부 필드를 이용해서 검색



```
1 {
2   "query": {
3     "nested": {
4       "path": "user",
5       "query": {
6         "bool": {
7           "must": [
8             { "match": { "user.first": "Alice" } }
9           ]
10        }
11      }
12    }
13  }
14 }
```



```
{
  "took" : 14,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.4054651,
    "hits" : [ {
      "_index" : "my_index",
      "_type" : "my_type",
      "_id" : "1",
      "_score" : 1.4054651,
      "_source" : {
        "group" : "fans",
        "user" : [ {
          "first" : "John",
          "last" : "Smith"
        }, {
          "first" : "Alice",
          "last" : "White"
        } ]
      }
    } ]
  }
}
```



Array type

ARRAY

Elasticsearch에서, 전용의 배열 형식이 없지만
사용시 모든 필드는 기본적으로 0 개 이상의 값을
포함. 배열의 모든 값은 동일한 데이터 유형

an array of strings: ["one", "two"]

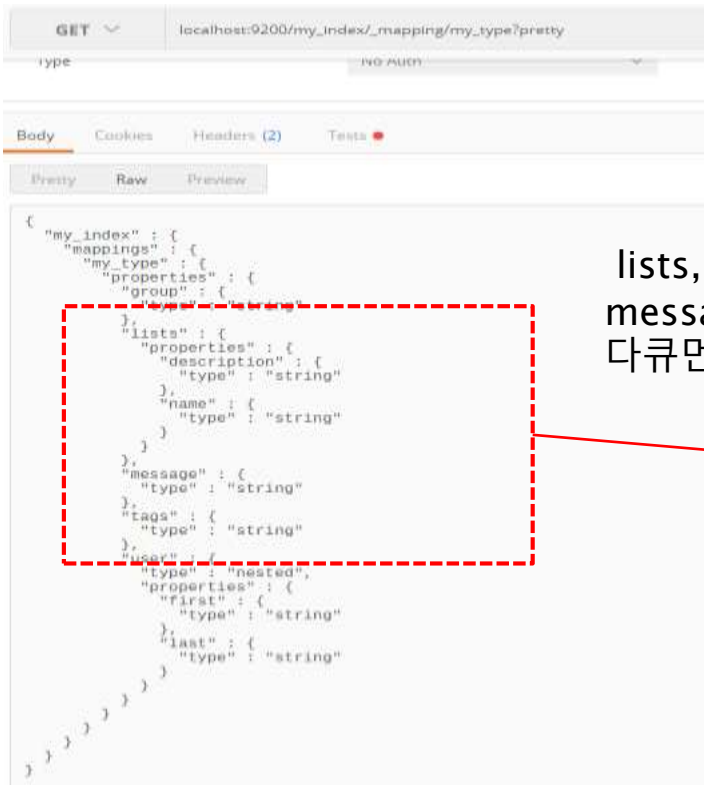
an array of integers: [1, 2]

an array of arrays: [1, [2, 3]] which is the equivalent of [1, 2, 3]

an array of objects: [{ "name": "Mary", "age": 12 }, { "name": "John", "age": 10 }]

Mapping 확인

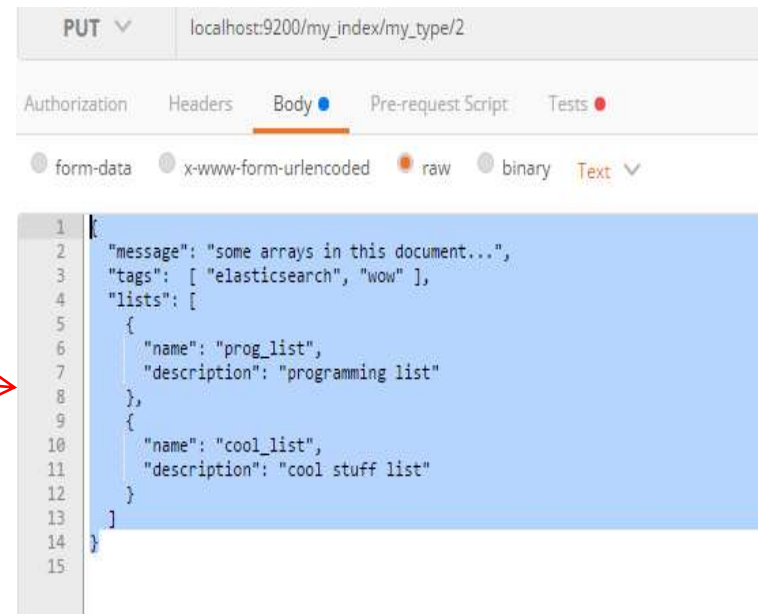
Array 타입이라도 mappings에는 코어타입으로 정의되어 있음



The screenshot shows a REST client interface with a GET request to `localhost:9200/my_index/_mapping/my_type?pretty`. The response is a JSON object defining the mapping for 'my_index'. A red dashed box highlights the 'lists' field within the 'my_type' mapping, which is defined as an array of objects with 'name' and 'description' properties, both of type 'string'.

```
{
  "my_index" : {
    "mappings" : {
      "my_type" : {
        "properties" : {
          "group" : {
            "type" : "string"
          },
          "lists" : {
            "properties" : {
              "description" : {
                "type" : "string"
              },
              "name" : {
                "type" : "string"
              }
            }
          },
          "message" : {
            "type" : "string"
          },
          "tags" : {
            "type" : "string"
          },
          "user" : {
            "type" : "nested",
            "properties" : {
              "first" : {
                "type" : "string"
              },
              "last" : {
                "type" : "string"
              }
            }
          }
        }
      }
    }
  }
}
```

lists, tags,
message에 대한
다큐먼트 생성



The screenshot shows a REST client interface with a PUT request to `localhost:9200/my_index/my_type/2`. The request body is a JSON document that includes a 'message' field, a 'tags' array, and a 'lists' array containing two objects with 'name' and 'description' properties.

```
1 {
2   "message": "some arrays in this document...",
3   "tags": [ "elasticsearch", "wow" ],
4   "lists": [
5     {
6       "name": "prog_list",
7       "description": "programming list"
8     },
9     {
10      "name": "cool_list",
11      "description": "cool stuff list"
12    }
13  ]
14 }
15
```

다큐먼트 생성

Array와 non-array로 처리해도 기본적으로 동일한 매핑

```
PUT localhost:9200/my_index/my_type/2

Authorization Headers Body Pre-request Script Tests
form-data x-www-form-urlencoded raw binary Text

1 {
2   "message": "some arrays in this document...",
3   "tags": [ "elasticsearch", "wow" ],
4   "lists": [
5     {
6       "name": "prog_list",
7       "description": "programming list"
8     },
9     {
10      "name": "cool_list",
11      "description": "cool stuff list"
12    }
13  ]
14 }
15
```

```
PUT localhost:9200/my_index/my_type/3

Authorization Headers Body Pre-request Script Tests
form-data x-www-form-urlencoded raw binary Text

1 {
2   "message": "no arrays in this document...",
3   "tags": "elasticsearch",
4   "lists": {
5     "name": "prog_list",
6     "description": "programming list"
7   }
8 }
9
```

다큐먼트 조회

Tags 필드가 array이므로 그 안의 wow로 검색하여 조회



```
GET localhost:9200/my_index/_search?pretty&q=wow&df=tags

{
  "took": 4,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 2,
    "max_score": 0.19178301,
    "hits": [
      {
        "_index": "my_index",
        "_type": "my_type",
        "_id": "2",
        "_score": 0.19178301,
        "_source": {
          "message": "some arrays in this document...",
          "tags": [ "elasticsearch", "wow" ],
          "lists": [
            {
              "name": "prog_list",
              "description": "programming list"
            },
            {
              "name": "cool_list",
              "description": "cool stuff list"
            }
          ]
        }
      },
      {
        "_index": "my_index",
        "_type": "my_type",
        "_id": "1",
        "_score": 0.19178301,
        "_source": {
          "message": "some arrays in this document...",
          "tags": [ "elasticsearch", "wow" ],
          "lists": [
            {
              "name": "prog_list",
              "description": "programming list"
            },
            {
              "name": "cool_list",
              "description": "cool stuff list"
            }
          ]
        }
      }
    ]
  }
}
```



좌표 type

좌표(geo_point)

지도 상의 위치 정보 즉 점을 담아 표현하는 구조

| | |
|-------------------|---|
| lat_lon | true일 경우 하위필드에 lat,lon에 저장 |
| geohash | true일 경우 하위필드에 geohash에 저장 |
| normalize | true일 경우 위도와 경도 값의 표준화 여부를 결정 |
| normalize_lat | false일 경우 위도를 표준화하지 않음 |
| normalize_lon | false일 경우 경도를 표준화하지 않음 |
| validate | normalize 옵션이 false일 경우 true 세팅되면 유효하지 않는 위도와 경도에 대해 오류 |
| validate_lat | true 일 경우 유효하지 않은 위도에 대해 오류 |
| validate_lon | true 일 경우 유효하지 않은 경도에 대해 오류 |
| geohash_prefix | true 일 경우 입력된 좌표를 위치해시값으로 변환 저장 |
| geohash_precision | geohash_prefix가 true일 경우 위치해시에 대한 정밀값 저장 |

위치모형(geo_shape)

지도 상의 위치 정보 즉 선, 원, 사각형, 다각형을
담아 표현하는 구조

| | |
|----------------|-----------------------------|
| precision | 위치모형에 대한 정밀값 저장 |
| distance_error | 다각형에 대한 허용할 오류의 한계치를 소수로 설정 |



다중 필드 옵션

다중필드(multi field)

기존 필드를 여러 형태로 색인할 수 있도록 fields 옵션을 사용해서 정의

```
{
  "properties" : {
    "필드명" : {
      "필드옵션"
      "fields" : {
        "하위필드명" : {
          "하위필드옵션"
        }
      }
    }
  }
}
```

토큰 수(token_count)

다중필드 정의를 이용해서 실제 저장된 필드의 토큰에 대한 분할된 수를 저장

```
{
  "properties" : {
    "필드명" : {
      "필드옵션"
      "fields" : {
        "tokens" : {
          "type" : "token_count",
          "store" : true,
          "analyzer" : "standard"
        }
      }
    }
  }
}
```

토큰 count 처리 예시: 1

Mapping 생성

GET localhost:9200/my_index/_mappings?pretty

Authorization Headers Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (2) Tests

Pretty Raw Preview

```
{
  "my_index" : {
    "mappings" : {
      "my_type" : {
        "properties" : {
          "name" : {
            "type" : "string",
            "fields" : {
              "length" : {
                "type" : "token_count",
                "analyzer" : "standard"
              }
            }
          }
        }
      }
    }
  }
}
```

중첩 필드에
length를 지정하고
token_count 타입
을 할당

토큰 count 처리 예시: 2

다큐먼트를 2개 생성

PUT localhost:9200/my_index/your_type/1

Authorization Headers Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary Text

1 { "name": "John Seftin" }

Body Cookies Headers (2) Tests

Pretty Raw Preview

```
[{"_index": "my_index", "_type": "your_type", "_id": "1", "_version": 1, "shards": {"total": 2, "successful": 1, "failed": 0}, "created": true}]
```

PUT localhost:9200/my_index/your_type/2

Authorization Headers Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary Text

1 { "name": "Rachel Alice Williams" }

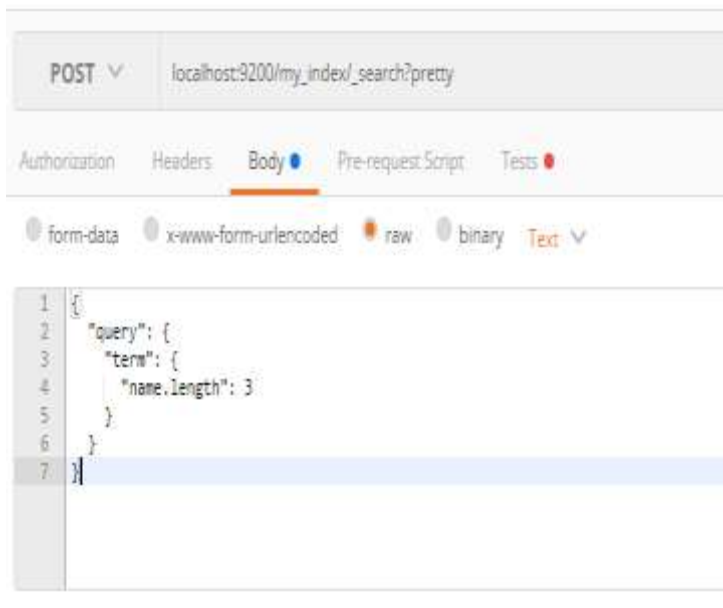
Body Cookies Headers (2) Tests

Pretty Raw Preview

```
{ "_index": "my_index", "_type": "your_type", "_id": "2", "_version": 1, "shards": {
```

토큰 count 처리 예시: 3

token_count가 3개인 것을 검색하지만 검색결과에는 표시되지 않음



```
POST localhost:9200/my_index/_search?pretty

{
  "query": {
    "term": {
      "name.length": 3
    }
  }
}
```



```
Pretty Raw Preview

{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.30685282,
    "hits" : [ {
      "_index" : "my_index",
      "_type" : "my_type",
      "_id" : "2",
      "_score" : 0.30685282,
      "_source" : {
        "name" : "Rachel Alice Williams"
      }
    } ]
  }
}
```



필드 복사 옵션

필드복사(copy_to)

특정 필드의 값을 다른 필드로 복사, 복사될 필드는 정의시 store 옵션이 true여야 함

```
{
  "properies": {
    "필드명1": {
      "type": "string", "copy_to": "pk_data"
    },
    "필드명2": {
      "type": "string", "copy_to": "pk_data"
    },
    "pk_data": {
      "type": "string", "store": true
    }
  }
}
```

필드명1과 필드명2
를 pk_data 필드에
복사되어 저장됨

데이터 생성 및 갱신



인덱스 생성

Index

인덱스(dahl)를 지정해서 생성 후 조회하면 dahl에 대한 세부 정보가 나옴

```
curl XPUT localhost:9200/dahl
```

```
curl XGET localhost:9200/dahl
```

```
{ "dahl": { "aliases": {}, "mappings": {}, "settings": { "index": { "creation_date":  
"1464920538785", "number_of_shards": "5", "number_of_replicas": "1", "uuid":  
"46BdJFAFQxCcD0Kii0ldWw", "version": { "created": "2030399" } } },  
"warmers": {} }}
```



타입 생성

Type

Mapping 을 정의하고 type을 생성

```
curl XPUT localhost:9200/dahl/_mapping/moon -d '{
  "properties": {
    "name": {
      "type": "string"
    }
  }
}'
```

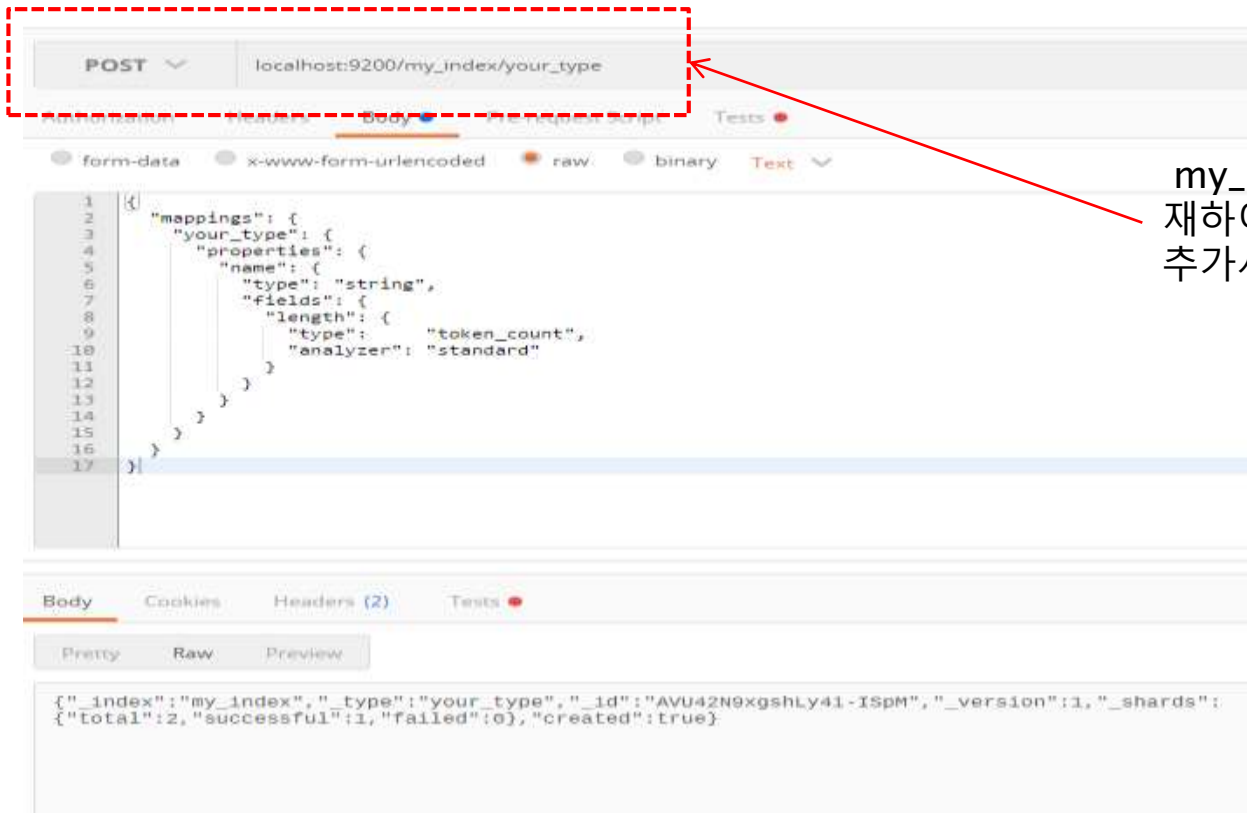
```
curl XGET localhost:9200/dahl/moon
{ "dahl": {
  "mappings": {
    "moon": {
      "properties": {
        "name": {
          "type": "string"
        }
      }
    }
  }
}
```



추가 타입 생성

추가 Type

Index에 추가 type에 대한 Mapping 생성시 오류가 발생하면 post 메소드로 처리



my_index 내에 my_type이 존재하여 put 메소드 your_type 추가시 오류



Document 생성

Document 생성

Mapping에 맞도록 데이터를 정의하고 id 값을 1을 정의하고 다큐먼트를 생성

```
curl XPUT localhost:9200/dahl/moon/1 -d '{
  "name": "dahlmoon"
}'

{ "_index": "dahl", "_type": "moon", "_id": "1", "_version": 1, "_shards": { "total": 2,
"successful": 1, "failed": 0 }, "created": true}
```


Document 생성 : op_type

PUT 메소드로 신규 생성할 경우 op_type을 이용해서 신규가 아닐 경우 에러 처리

```
curl XPUT localhost:9200/dahl/moon/3?op_type=create -d '{
  "name": "dahl moon"
}'

{"_index":"dahl","_type":"moon","_id":"3","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"created":true}

# 기존에 존재할 경우 신규가 안되고 에러 처리
curl XPUT localhost:9200/dahl/moon/3?op_type=create -d '{
  "name": "dahl moon"
}'

{"error":{"root_cause":[{"type":"document_already_exists_exception","reason":"[moon][3]: document already exists","shard":"4","index":"dahl"}],"type":"document_already_exists_exception","reason":"[moon][3]: document already exists","shard":"4","index":"dahl"},"status":409}
```

Document 생성 : _create

PUT 메소드로 신규 생성할 경우 _create를 이용해서 신규가 아닐 경우 에러 처리

```
curl XPUT localhost:9200/dahl/moon/4/_create -d '{
  "name": "dahl yong moon"
}'

{"_index":"dahl","_type":"moon","_id":"4","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"created":true}

# 기존에 존재할 경우 신규가 안되고 에러 처리
curl XPUT localhost:9200/dahl/moon/4/_create -d '{
  "name": "dahl yong moon"
}'

{"error":{"root_cause":[{"type":"document_already_exists_exception","reason":"[moon][4]: document already exists","shard":"2","index":"dahl"}],"type":"document_already_exists_exception","reason":"[moon][4]: document already exists","shard":"2","index":"dahl"},"status":409}
```



Id 지정없이 다큐먼트 생성

Id 지정없이 다크먼트 생성

_id에 임의의 id 값이 생성되어 갱신됨

```
curl XPOST localhost:9200/_website/blog/ -d '{
  "title": "My second blog entry",
  "text": "Still trying this out...",
  "date": "2014/01/01"
}'

{ "_index": "website", "_type": "blog", "_id": "AVUjaeueQly_M2v0EEWe", "_version": 1,
  "_shards": { "total": 2, "successful": 1, "failed": 0 }, "created": true}
```



데이터 조화

Document 생성 결과 조회

인덱스, 타입 그리고 id로 document 생성 결과를 조회

```
curl XGET localhost:9200/dahl/moon/1
```

```
{ "_index": "dahl", "_type": "moon", "_id": "1", "_version": 1, "found": true, "_source":  
  { "name": "dahlmoon" } }
```

Document 만 조회(_source)

실제 다큐먼트 내용만 조회가 필요한 경우
_source 필드에서 데이터만 추출

```
curl XGET localhost:9200/dahl/moon/1
```

```
{ "_index": "dahl", "_type": "moon", "_id": "1", "_version": 1, "found": true, "_source":  
{ "name": "dahlmoon" }}
```

```
curl XGET localhost:9200/dahl/moon/1 /_source
```

```
{ "name": "dahlmoon"}
```

특정 필드만 조회

실제 다큐먼트 내용중에 특정 필드만 조회가 필요한 경우 ?_source=필드명으로 데이터만 추출

```
curl XGET localhost:9200/dahl/moon/1
```

```
{ "_index": "dahl", "_type": "moon", "_id": "1", "_version": 1, "found": true, "_source":  
{ "name": "dahlmoon" } }
```

```
curl XGET localhost:9200/dahl/moon/1 ?_source= name
```

```
{ "_index": "dahl", "_type": "moon", "_id": "1", "_version": 1, "found": true, "_source": { "name": "dahlmoon" } }
```




다큐먼트 존재여부 확인

다큐먼트 존재 여부 조회 -1

존재하지 id로 조회할 경우 404 not found 에러 발생

#Head 메소드로 124번 id가 존재하는지 점검

```
C:\search>curl -XHEAD localhost:9200/books/book/124
```

Warning: Setting custom HTTP method to HEAD with -X/--request may not work the

Warning: way you want. Consider using -I/--head instead.

#-i 를 주고 점검

```
C:\search>curl -i -XHEAD localhost:9200/books/book/124
```

Warning: Setting custom HTTP method to HEAD with -X/--request may not work the

Warning: way you want. Consider using -I/--head instead.

HTTP/1.1 404 Not Found

Content-Type: text/plain; charset=UTF-8

Content-Length: 0

다큐먼트 존재 여부 조회 -2

-I 로 조회시 결과값으로 json을 문서가 옴

#-I로 조회시 elasticsearch 조회된 body 값을 리턴함

```
C:\search>curl -i localhost:9200/books/book/124
HTTP/1.1 404 Not Found
Content-Type: application/json; charset=UTF-8
Content-Length: 59
{"_index":"books","_type":"book","_id":"124","found":false}
```

#--head 조회시 body 결과가 없음

```
C:\search>curl --head localhost:9200/books/book/124
HTTP/1.1 404 Not Found
Content-Type: text/plain; charset=UTF-8
Content-Length: 0
```



전체 갱신

생성된 다큐먼트 전체 갱신

동일한 id에 put처리하면 전체 내용이 갱신되어 버림. created 값이 false는 기존 것을 갱신했다는 표시

```
curl XPUT localhost:9200/website/blog/123 -d '{
  "title": "My first blog entry",
  "text": "I am starting to get the hang of this dahl!!!!!!...",
  "date": "2014/01/02",
  "author": "dahl"
}'

{"_index":"website","_type":"blog","_id":"123","_version":2,"_shards":{"total":2,"successful":1,"failed":0},"created":false}
```



삭제

다큐먼트 삭제

Delete 메소드를 정의하고 다크먼트 id를 부여해서 삭제

```
curl XDELETE localhost:9200/dahl/moon/4
```

```
{"found":true,"_index":"dahl","_type":"moon","_id":"4","_version":2,"_shards":{"total":2,"successful":1,"failed":0}}
```

#삭제 이후에 id로 다시 생성할 경우 버전번호가 올라감
#데이터만 삭제했지 mapping 정보는 가지고 있음

```
curl XPOST localhost:9200/dahl/moon/4 -d '{  
  "name" : "dahl yong moon "  
}'
```

```
{"_index":"dahl","_type":"moon","_id":"4","_version":3,"_shards":{"total":2,"successful":1,"failed":0},"created":true}
```

index 삭제

Delete 메소드로 index 삭제하면 기존 mapping 정보까지 전부 삭제됨

```
curl XDELETE localhost:9200/dahl
```

```
{"acknowledged":true}
```

#index 삭제 이후에는 mapping 정보를 가지고 있지 않아

#생성시 버전번호가 다시 1 부터 시작됨

```
curl XPOST localhost:9200/dahl/moon1/1 -d '
```

```
{  
  "name" : "dahl yong moon "  
}
```

```
{"_index":"dahl","_type":"moon1","_id":"1","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"created":true}
```


BULK 처리

Moon Yong Joon



Bulk 구조

데이터 지정 방식

Bulk 처리를 위해 action과 request를 정의해서 처리

```
{ action: { metadata }}\n{ request body }\n{ action: { metadata }}\n{ request body }
```

- Metadata에는 Mapping 정보 (_index, _type 등 내장정보)를 지정
- Request body는 실제 다큐먼트에 저장 정보나 update를 위한 정보

```
{ "delete": { "_index": "website", "_type": "blog", "_id": "123" } }\n{ "create": { "_index": "website", "_type": "blog", "_id": "123" } }\n{ "title": "My first blog post" }\n{ "index": { "_index": "website", "_type": "blog" } }\n{ "title": "My second blog post" }\n{ "update": { "_index": "website", "_type": "blog", "_id": "123", "_retry_on_conflict": 3 } }\n{ "doc" : { "title" : "My updated blog post" } }
```

Action

Bulk 처리 시 실제 Method의 행위를 지시

| Action | Request body | 설명 |
|--------|--------------|--------------------------------|
| index | 필수 | 데이터 입력 명령, 기존 다큐먼트가 존재하면 갱신 처리 |
| create | 필수 | 데이터 입력 명령, 기존 다큐먼트가 존재하면 에러 처리 |
| update | 필수 | 데이터에 대한 일부 갱신 처리 |
| delete | N/A | 데이터 삭제 처리 |



Bulk 실행

실행 방식

Bulk 처리를 위해 action과 request를 정의해서 처리

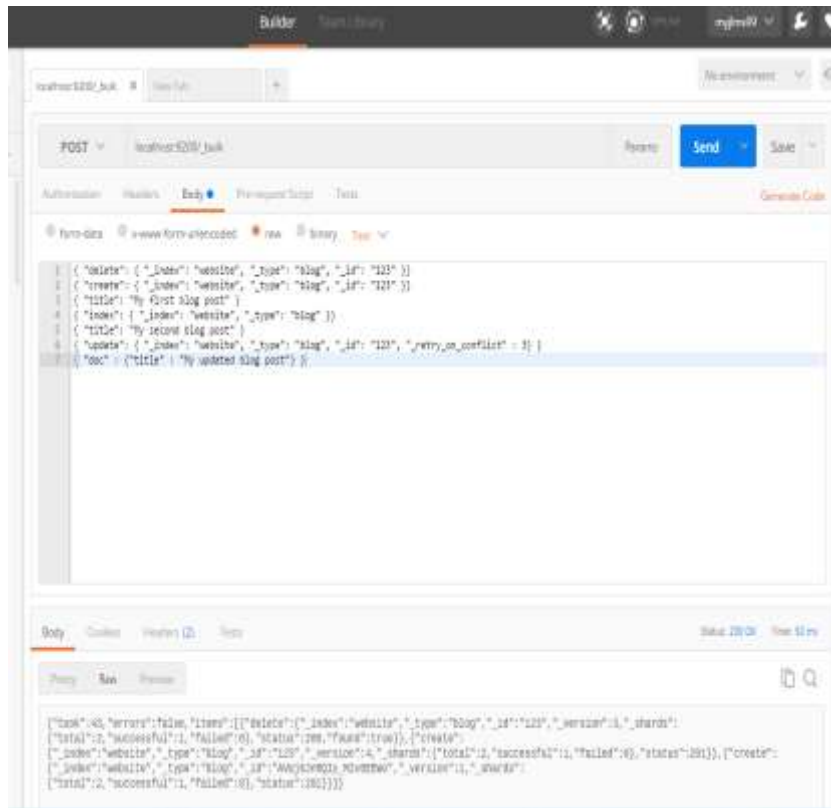
```
curl -XPOST host:port/{index}/{type}/_bulk -d '{데이터}'  
또는 curl -XPOST host:port/{index}/{type}/_bulk --data-binary @{파일명}
```

```
curl -XPOST host:port/{index}/_bulk -d '{데이터}'  
또는 curl -XPOST host:port/{index}/_bulk --data-binary @{파일명}
```

```
curl -XPOST host:port/_bulk -d '{데이터}'  
또는 curl -XPOST host:port/_bulk --data-binary @{파일명}
```

실행

Bulk로 데이터를 추가, 갱신, 삭제 처리



Request body

```
{ "delete": { "_index": "website", "_type": "blog", "_id": "123" } }
{ "create": { "_index": "website", "_type": "blog", "_id": "123" } }
{ "title": "My first blog post" }
{ "index": { "_index": "website", "_type": "blog" } }
{ "title": "My second blog post" }
{ "update": { "_index": "website", "_type": "blog", "_id": "123",
  "_retry_on_conflict": 3 } }
{ "doc" : { "title" : "My updated blog post" } }
```

처리 결과

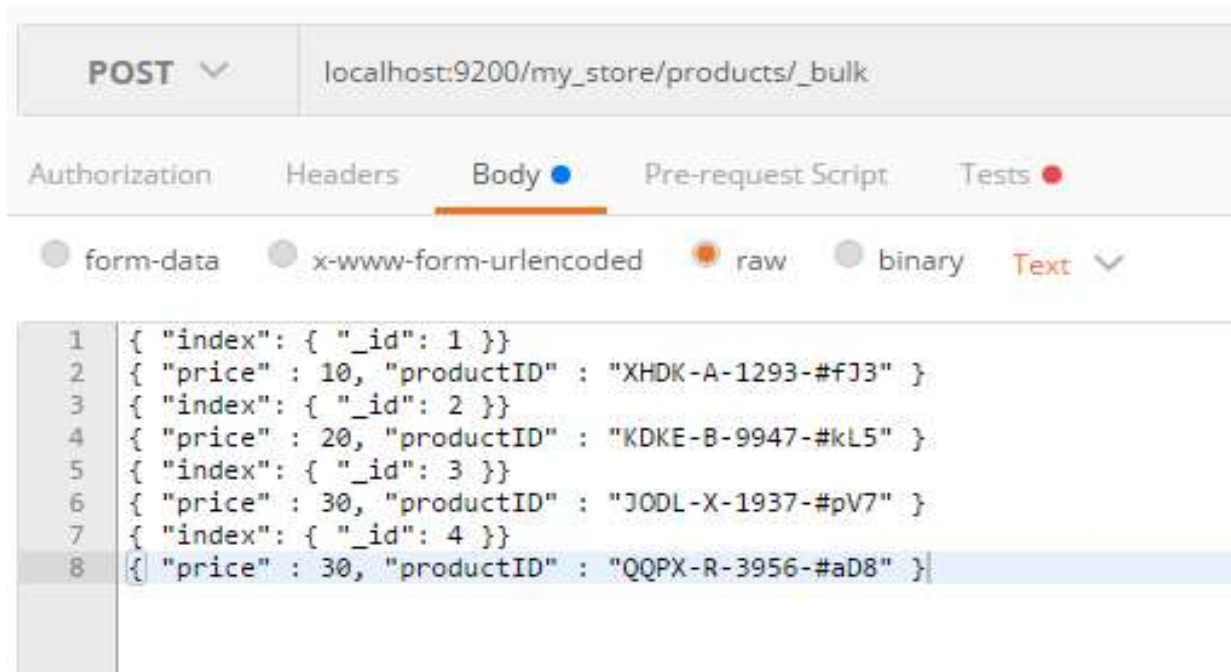
```
{ "took": 43, "errors": false, "items": [ { "delete": { "_index": "website",
  "_type": "blog", "_id": "123", "_version": 3, "_shards": { "total": 2,
  "successful": 1, "failed": 0 }, "status": 200, "found": true }, { "create": {
  "_index": "website", "_type": "blog", "_id": "123", "_version": 4,
  "_shards": { "total": 2, "successful": 1, "failed": 0 }, "status": 201 },
  { "index": { "_index": "website", "_type": "blog", "_id": "AVUj6JnRQly_M2v0EEWo",
  "_version": 1, "_shards": { "total": 2, "successful": 1, "failed": 0 },
  "status": 201 } } ] }
```



Bulk 실행(_id만 입력)

실행

localhost:9200/my_store/products/_bulk ,
index, type을 지정하고 내부 파일에는 _id만 부여



실행 결과

3개 모두 index에 등록됨

Pretty Raw Preview



```
{"took":497,"errors":false,"items":[{"index":{"_index":"my_store","_type":"products","_id":"1","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}}, {"index":{"_index":"my_store","_type":"products","_id":"2","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}}, {"index":{"_index":"my_store","_type":"products","_id":"3","_version":1,"_shards":{"total":2,"successful":1,"failed":0},"status":201}}]}
```

MAPPINGS

Moon Yong Joon



Mapping 기본

Mapping 이란

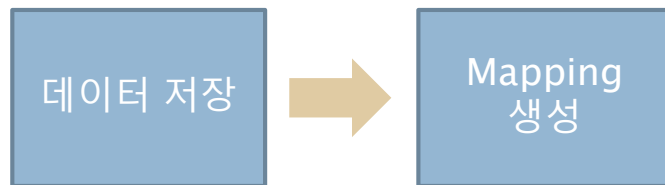
데이터의 저장형태와 검색엔진에서 접근하고 처리하기 위한 명세

```
{ "mappings":  
  { "<타입명>":  
    { "<내장필드명>":  
      { ... <필드 내용> ... }  
      .....  
    }  
  }  
}
```

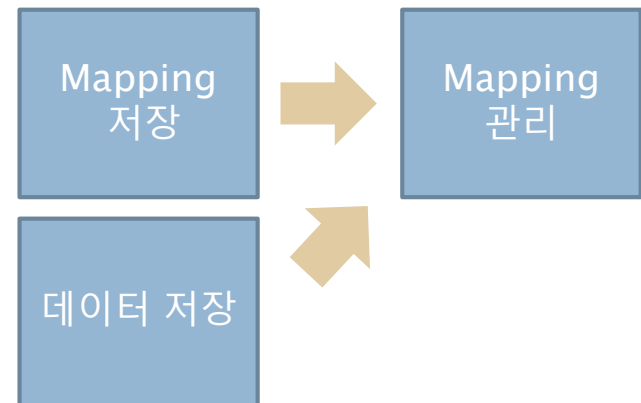
Mapping 특징

필드 구성을 별도로 하지 않아도 동적 매핑에 의해 자동으로 데이터 타입과 문서가 저장
동적 템플릿을 통해 개별 타입에 대한 유연한 관리

동적 매핑 생성



정적 매핑 생성



Mapping 생성

Elasticsearch에 데이터를 입력하면 인덱스가 자동 생성되지만 아래처럼 매핑을 입력한 후에 데이터를 저장할 수 있음

```
PUT <host>/<Index명> -d '{
  "mappings": {
    "<타입명>": {
      "<내장필드명>": {
        ... <필드 내용> ...
      }
    }
  }
}'
```

타입 매핑의 예 1

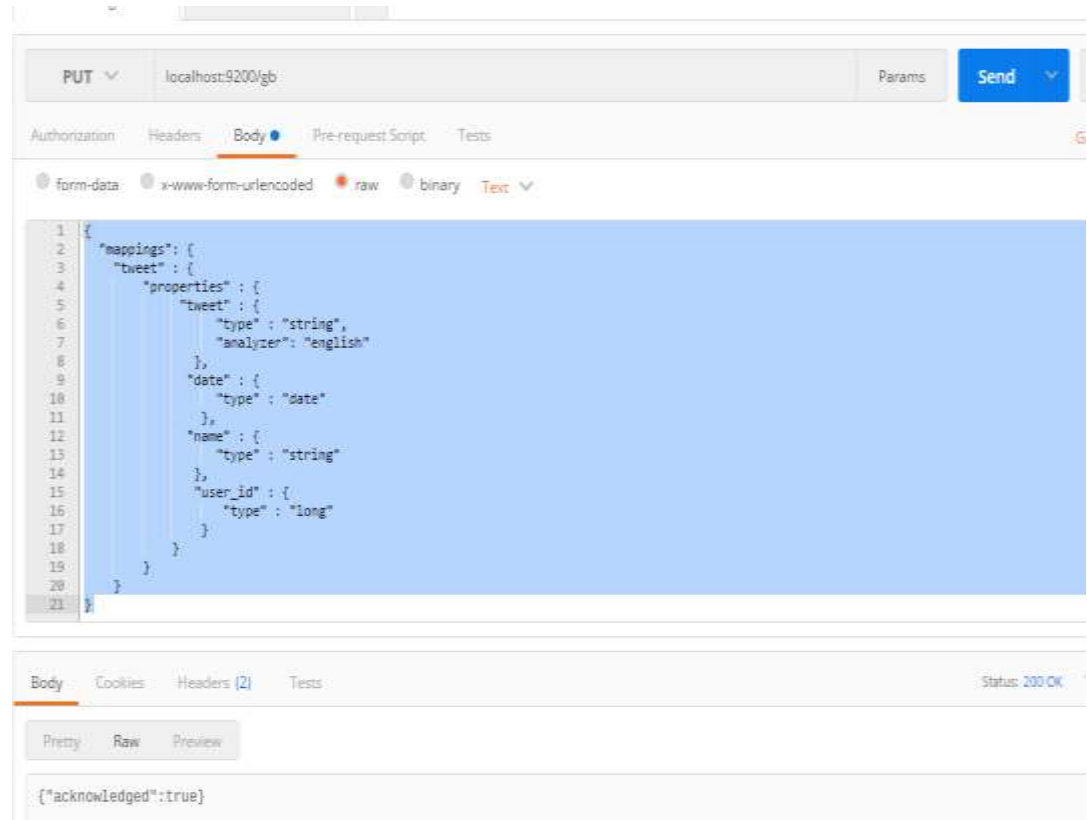
type에 대한 사용자 정의에 대한 매핑은 타입 내의 properties 내에 정의 됨

```
{ "books": {  
  "mappings": {  
    "book": {  
      "properties": {  
        "author": { "type": "string" },  
        "category": { "type": "string" },  
        "date": { "type": "date",  
                  "format": "strict_date_optional_time||epoch_millis" },  
        "pages": { "type": "long" },  
        "title": { "type": "string" }  
      }  
    }  
  }  
}
```


타입 매핑의 예2

타입을 생성하기 위해 매핑을 정의한 후에 put 메소드를 이용해서 생성

```
{
  "mappings": {
    "tweet" : {
      "properties" : {
        "tweet" : {
          "type" : "string",
          "analyzer": "english"
        },
        "date" : {
          "type" : "date"
        },
        "name" : {
          "type" : "string"
        },
        "user_id" : {
          "type" : "long"
        }
      }
    }
  }
}
```





Mapping 조회

Mapping 조회

Type에 대해 생성된 매핑을 조회하기 위해
_mapping API를 이용해서 검색

```
GET <host>/<Index명>/_mapping?pretty
```

```
GET <host>/<index명>/<타입명>/_mapping?pretty
```

Mapping 조회 : index 전체

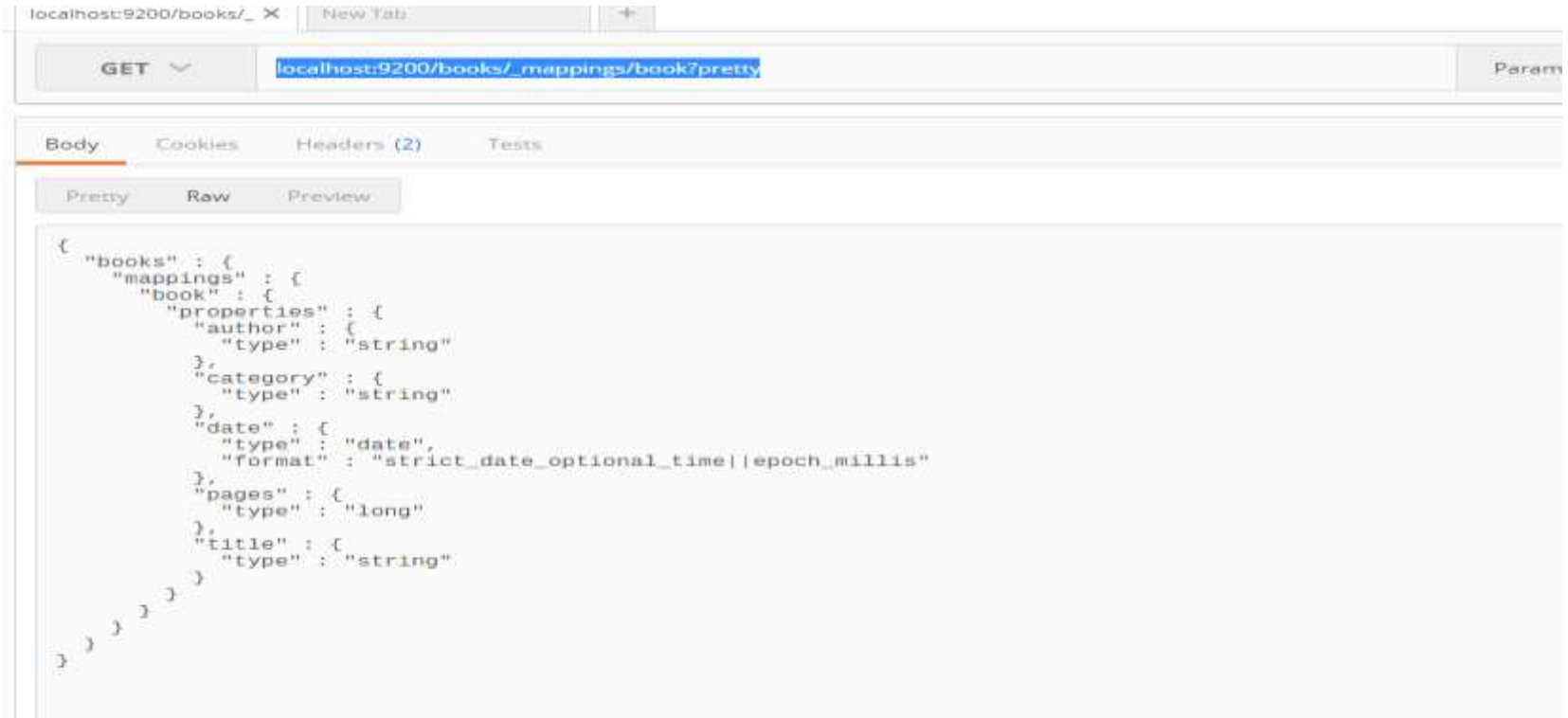
localhost:9200/books/_mappings?pretty이용
해서 검색



```
{
  "books" : {
    "mappings" : {
      "itbook" : {
        "properties" : {
          "author" : {
            "type" : "string"
          },
          "pages" : {
            "type" : "long"
          },
          "price" : {
            "type" : "long"
          },
          "title" : {
            "type" : "string"
          }
        }
      }
    }
  },
  "book" : {
    "properties" : {
      "author" : {
        "type" : "string"
      },
      "category" : {
        "type" : "string"
      },
      "date" : {
        "type" : "date",
        "format" : "strict_date_optional_time||epoch_millis"
      },
      "pages" : {
        "type" : "long"
      }
    }
  }
}
```

Mapping 조회 : 특정 타입만

localhost:9200/books/_mappings/book?pretty
이용해서 검색



```
{
  "books" : {
    "mappings" : {
      "book" : {
        "properties" : {
          "author" : {
            "type" : "string"
          },
          "category" : {
            "type" : "string"
          },
          "date" : {
            "type" : "date",
            "format" : "strict_date_optional_time||epoch_millis"
          },
          "pages" : {
            "type" : "long"
          },
          "title" : {
            "type" : "string"
          }
        }
      }
    }
  }
}
```

매핑에 따른 값 조회

type에 mapping에 맞게 값이 조회

```
curl -XGET localhost:9200/books/book/11?pretty
```

```
{ "_index": "books",  
  "_type": "book",  
  "_id": "11",  
  "_version": 1,  
  "found": true,  
  "_source": {  
    "title": "Elasticsearch Guide",  
    "author": [ "lee" ],  
    "date": "2014-05-01",  
    "pages": 300,  
    "category": "ICT"  
  }  
}
```



Mapping 추가

Mapping 추가

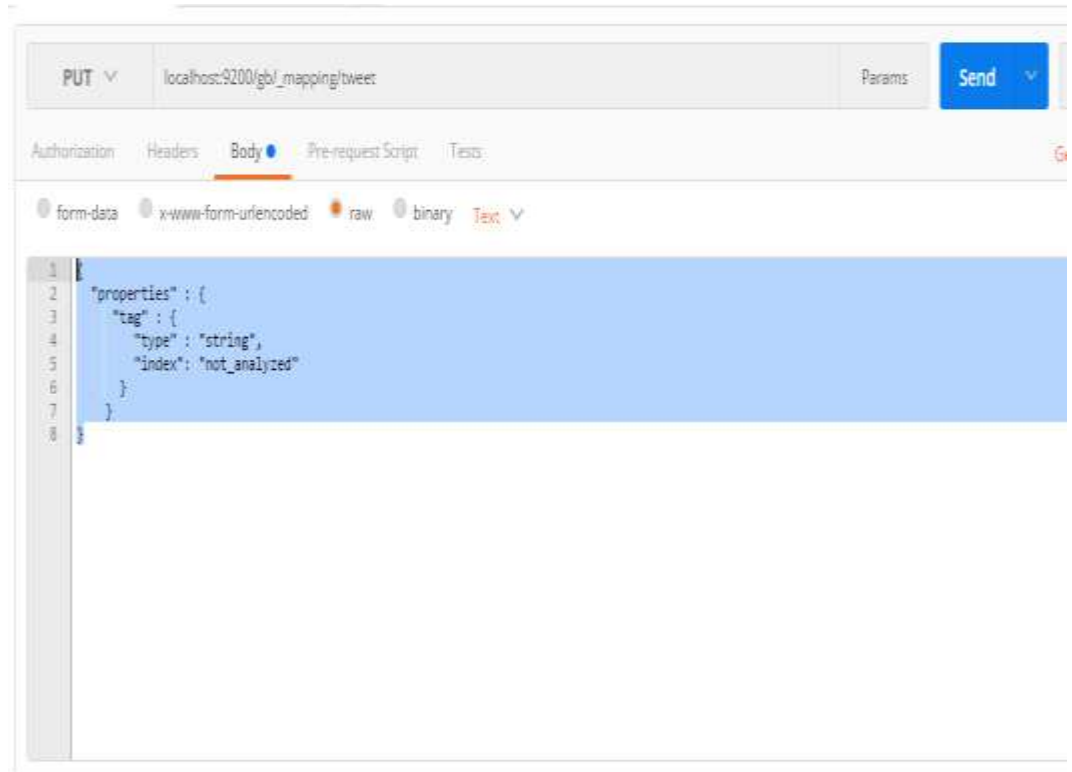
Type에 대해 생성된 매핑에 대해 추가하기도
_mapping API를 이용해서 데이터를 추가

```
GET <host>/<Index명>/_mapping/<타입명> -d '{
  "<타입명>" : {
    "properties" : {
      <매핑내용>
    }
  }
}
```


Mapping 추가 예시 -1

기존에 정의된 타입의 매핑정보에 속성을 추가해서 갱신하기

```
{
  "properties" : {
    "tag" : {
      "type" : "string",
      "index" : "not_analyzed"
    }
  }
}
```



META FIELDS

Moon Yong Joon



Identity meta-fields

_index와 _type

내장필드 _index와 _type 에는 실제 인덱스와 타입에 대한 이름이 세팅

```
{ "_index": "books",  
  "_type": "book",  
  "_id": "11",  
  "_version": 1,  
  "found": true,  
  "_source": {  
    "title": "Elasticsearch Guide",  
    "author": [ "lee" ],  
    "date": "2014-05-01",  
    "pages": 300,  
    "category": "ICT"  
  }  
}
```



Document source meta-fields

_source

입력한 원본 데이터는 _source 필드에 저장됨

```
{ "_index": "books",  
  "_type": "book",  
  
  "_id": "11",  
  "_version": 1,  
  "found": true,  
  "_source": {  
    "title": "Elasticsearch Guide",  
    "author": [ "lee" ],  
    "date": "2014-05-01",  
    "pages": 300,  
    "category": "ICT"  
  }  
}
```

_source 에 데이터 미저장

데이터 저장이 필요하지 않을 경우 _source 내부의 enabled의 값을 false로 세팅해서 처리

```
Curl - XPUT <host>/<index> -d '{
  "mappings" : {
    "타입명" : {
      "_source" : {
        "enabled" : false
      }
    }
  }
}
```

_source 에 특정데이터 명기 저장

특정 데이터만 저장이 필요한 경우 includes 속성에 실제 저장될 필드명만 명기해서 처리

```
Curl - XPUT <host>/<index> -d '{
  "mappings": {
    "타입명": {
      "_source": {
        "includes": ["필드명", "필드명", "필드명"]
      }
    }
  }
}
```


_source 에 특정데이터 임의 저장

특정 데이터에 대해 임의로 저장할 경우 **excludes** 속성을 이용해서 필드명이 특정 문자와 *를 이용해 지정

```
Curl -XPUT <host>/<index> -d '{
  "mappings": {
    "타입명": {
      "_source": {
        "excludes": ["필드명의 특정문자*"]
      }
    }
  }
}
```



Indexing meta-fields

_all

다큐먼트 내의 검색할 대상 필드를 지정 가능

```
Curl -XPUT <host>/<index> -d '{
  "mappings": {
    "타입명": {
      "_all": {"enabled": true}
      "properties": {
        "필드명1": {
          "include_in_all": true ,
          "type": "string"
        }
        "필드명2": {
          "include_in_all": false ,
          "type": "string"
        }
      }
    }
  }
}
```

_timestamp

다큐먼트 내의 timestamp에 대한 값을 저장하기 위해서는 명기적으로 지정해야 함

```
Curl -XPUT <host>/<index> -d '{
  "mappings": {
    "타입명": {
      "_timestamp": {
        "enabled": true
        "store": true
      }
    }
  }
}
```

_ttl(time to live)

입력된 데이터가 실제 index내에서 유지되는 시간을 설정할 때 사용 - d(일), m(분), h(시), s(초), ms(1 / 1000초), w(주)

```
Curl - XPUT <host> /<index> -d '{
```

```
{
  "mappings" : {
    "타입명" : {
      "_ttl" : {
        "enabled" : true
        "default" : "2d"
      }
    }
  }
}
```

_analyzer

다큐먼트 내의 특정필드에 저장된 값을 이용해
서 다크먼트가 색인되어 분석할 때 사용

```
Curl -XPUT <host>/<index> -d '{
  "mappings": {
    "타입명": {
      "_analyzer": {"path": "필드명"}
      "properties": {
        "필드명1": {
          "type": "string"
        }
        "필드명2": {
          "type": "string"
        }
      }
    }
  }
}
```

URI 검색



_search API

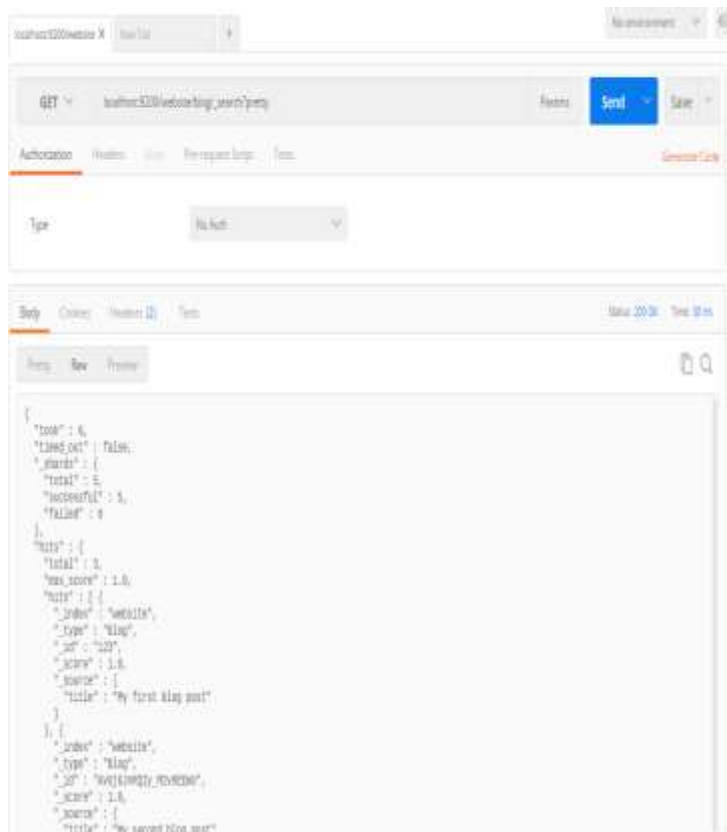
_search 검색 방법

_search를 이용해서 다양한 것을 검색할 수 있음

| | |
|--|-------------------------------------|
| <code>/_search</code> | # 모든 인덱스 내의 모든 타입을 검색 |
| <code>/gb/_search</code> | # gb 인덱스 내의 모든 타입을 검색 |
| <code>/gb,us/_search</code> | # gb,us 인덱스 내의 모든 타입을 검색 |
| <code>/g*,u*/_search</code> | # g*,u*로 시작하는 인덱스 내의 모든 타입을 검색 |
| <code>/gb/user/_search</code> | # 인덱스 gb, 타입 user 내의 모든 다큐먼트 조회 |
| <code>/gb,us/user,tweet/_search</code> | # gb, us 인덱스 내의 user, tweet 타입 내 조회 |
| <code>/_all/user,tweet/_search</code> | # 모든 인덱스 내의 user와 tweet 타입 내 조회 |

_search 검색

특정 type 내의 다큐먼트를 조회



Get

`localhost:9200/website/blog/_search?pretty`

처리 결과

```
{
  "took": 6,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 3,
    "max_score": 1.0,
    "hits": [
      {
        "_index": "website",
        "_type": "blog",
        "_id": "123",
        "_score": 1.0,
        "_source": {
          "title": "My first blog post"
        }
      },
      {
        "_index": "website",
        "_type": "blog",
        "_id": "456789101234567890",
        "_score": 1.0,
        "_source": {
          "title": "My second blog post"
        }
      },
      {
        "_index": "website",
        "_type": "blog",
        "_id": "123",
        "_score": 1.0,
        "_source": {
          "title": "My first blog post"
        }
      }
    ]
  }
}
```



페이지

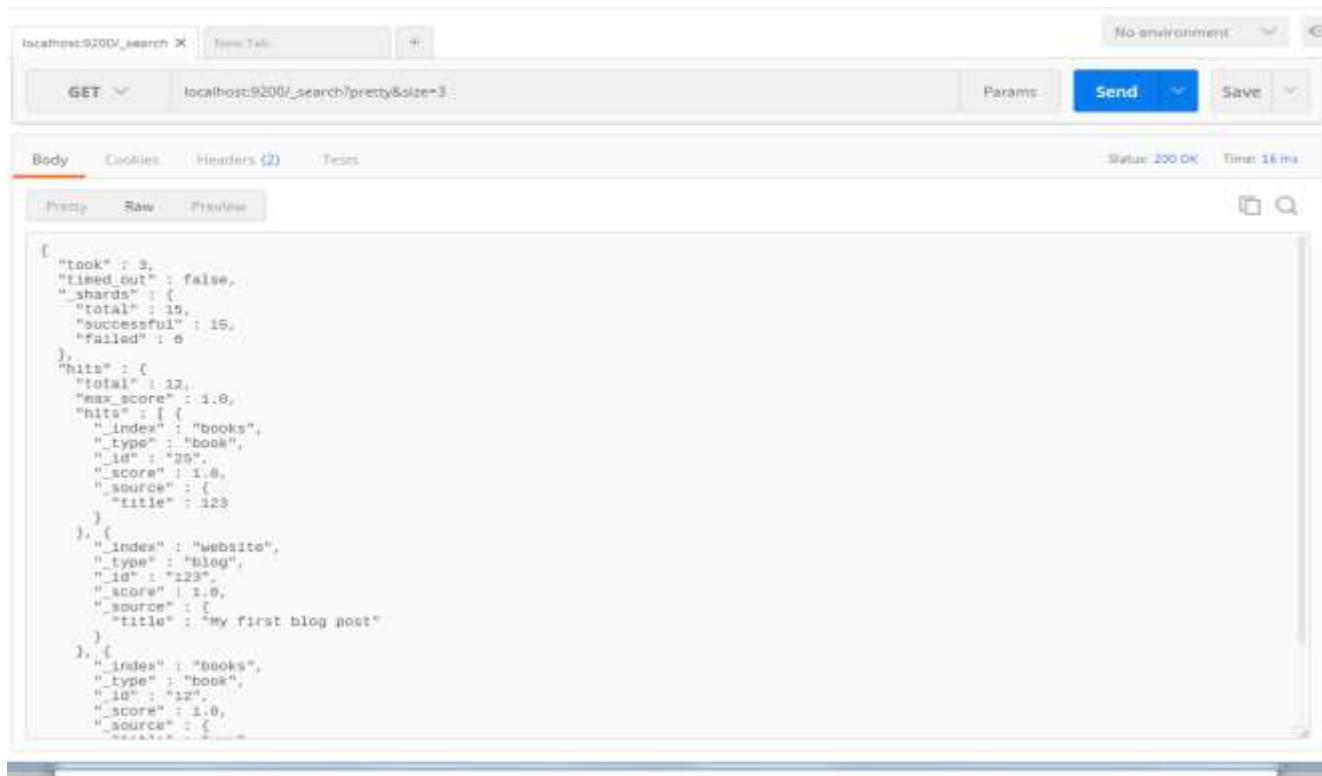
페이징 처리

_search한 결과를 페이지처리하기 위해 size와 from을 이용해서 페이징 처리

```
GET /_search?size=5  
GET /_search?size=5&from=5  
GET /_search?size=5&from=10
```

페이징 처리 예시

_search에 size=3 으로 주면 실제 12개에서 3개만 출력됨





Query string

Query string 처리

_search?q= 값 또는 필드명:질의어를 하나 또는 여러 개를 입력하고 검색

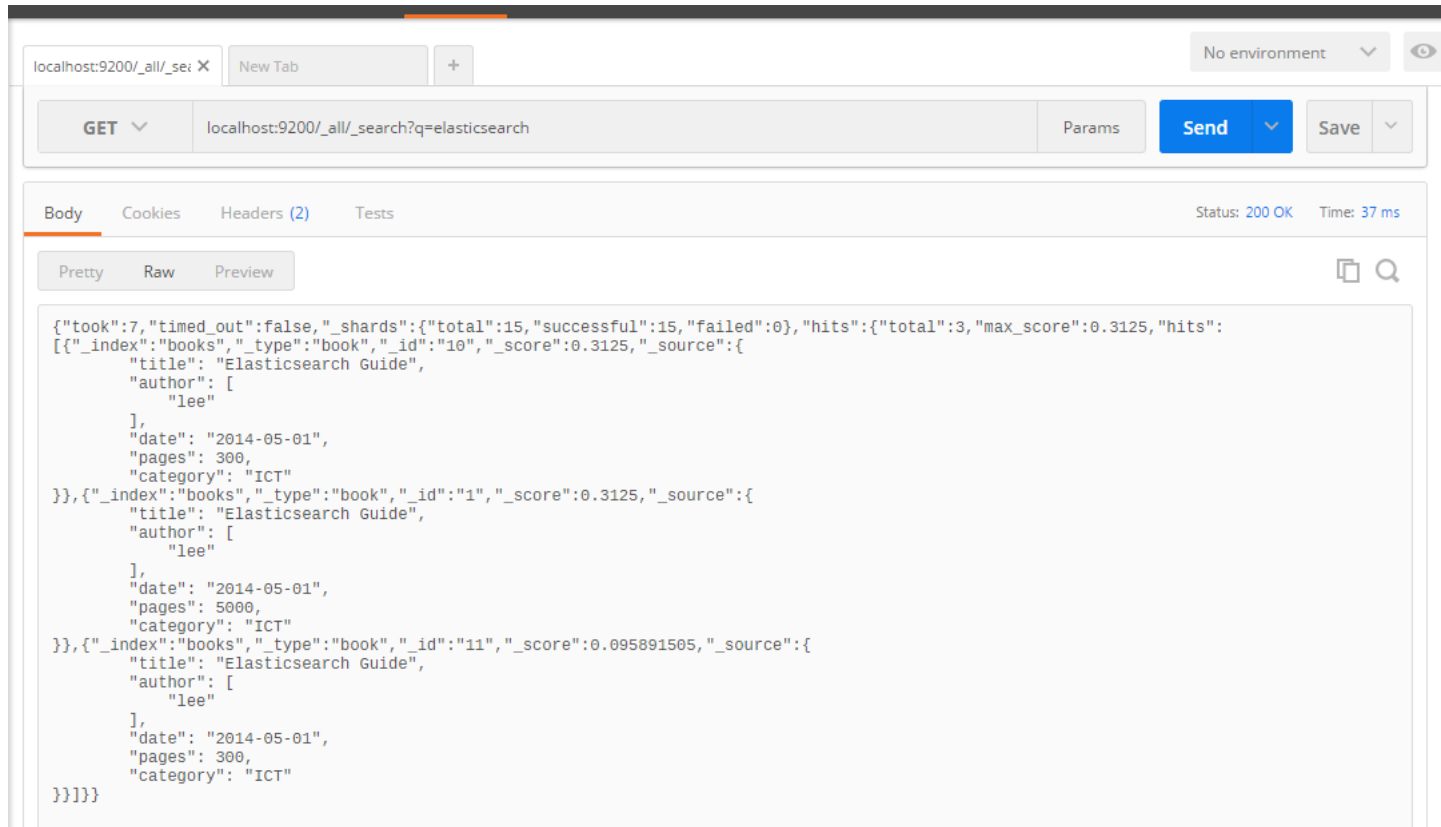
```
GET localhost:9200/_all/_search?q=질의어
```

```
GET localhost:9200/_all/_search?q=필드명:질의어
```

```
GET localhost:9200/_all/_search?q=필드명:질의어&필드명:질의어
```

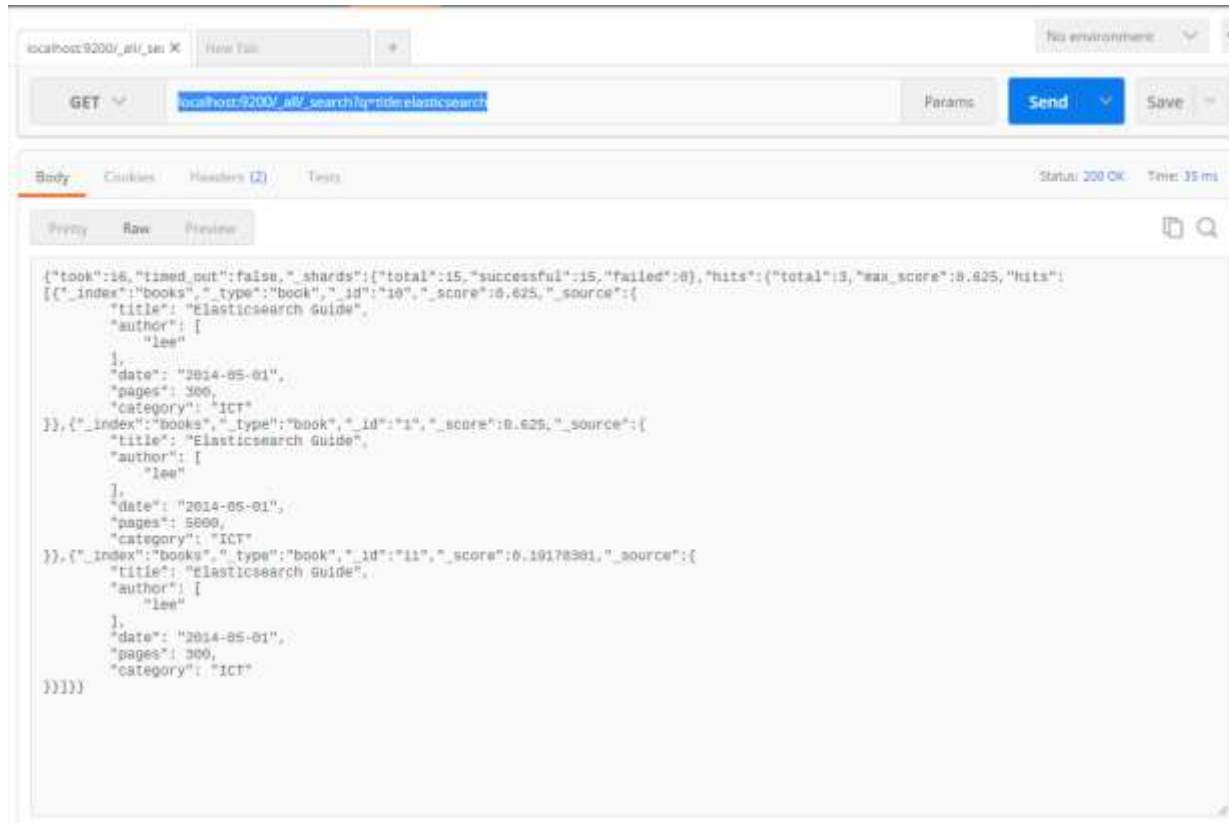
Query string 처리 : 질의어

localhost:9200/_all/_search?q=elasticsearch
를 지정하고 검색



Query string 처리 : 필드명:값

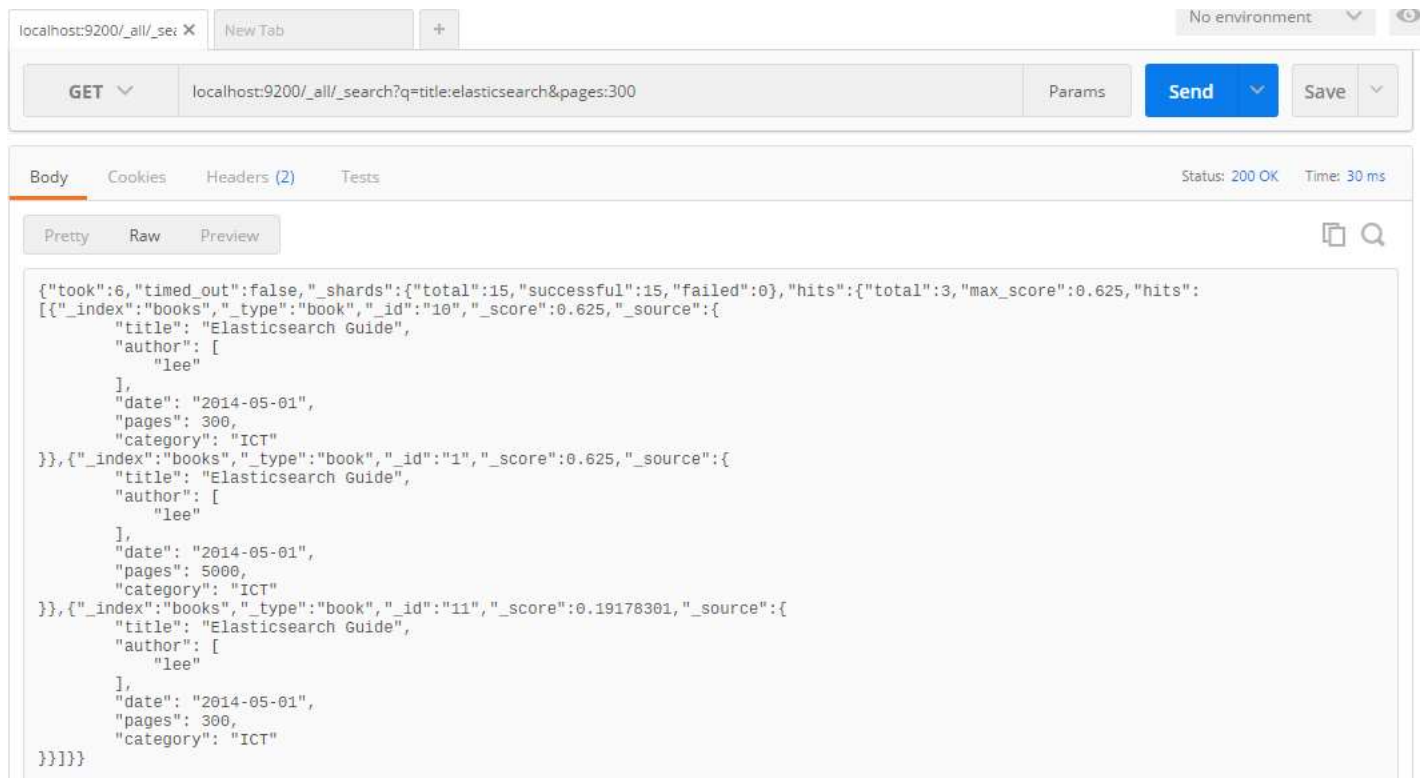
localhost:9200/_all/_search?q=title:elasticsearch 를 지정하고 검색



```
{
  "took": 16,
  "timed_out": false,
  "_shards": {
    "total": 15,
    "successful": 15,
    "failed": 0
  },
  "hits": {
    "total": 3,
    "max_score": 0.825,
    "hits": [
      {
        "_index": "books",
        "_type": "book",
        "_id": "10",
        "_score": 0.825,
        "_source": {
          "title": "Elasticsearch Guide",
          "author": [
            "lee"
          ],
          "date": "2014-05-01",
          "pages": 300,
          "category": "ICT"
        }
      },
      {
        "_index": "books",
        "_type": "book",
        "_id": "1",
        "_score": 0.825,
        "_source": {
          "title": "Elasticsearch Guide",
          "author": [
            "lee"
          ],
          "date": "2014-05-01",
          "pages": 5000,
          "category": "ICT"
        }
      },
      {
        "_index": "books",
        "_type": "book",
        "_id": "11",
        "_score": 0.19178301,
        "_source": {
          "title": "Elasticsearch Guide",
          "author": [
            "lee"
          ],
          "date": "2014-05-01",
          "pages": 300,
          "category": "ICT"
        }
      }
    ]
  }
}
```

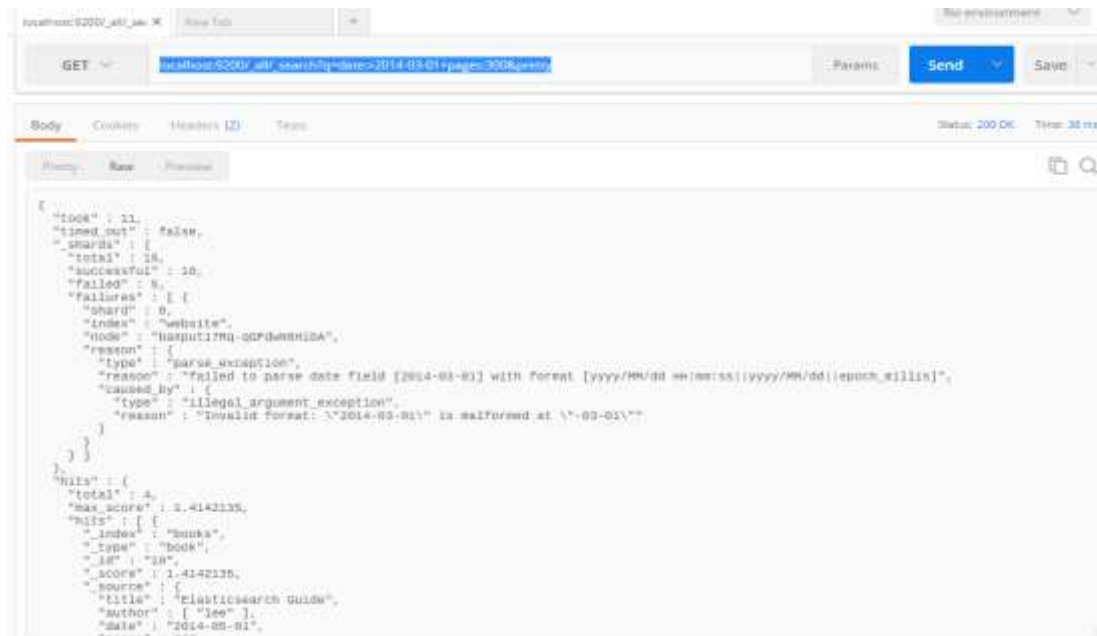
Query string 처리 : 여러 필드(&)

localhost:9200/_all/_search?q=title:elasticsearch
arch&pages:300를 지정하고 검색



Query string 처리 : 여러 필드(+)

localhost:9200/_all/_search?q=date:>2014-03-01+pages:300&pretty로 검색시 2개 필드 조건에 해당되는 것을 전부 검색



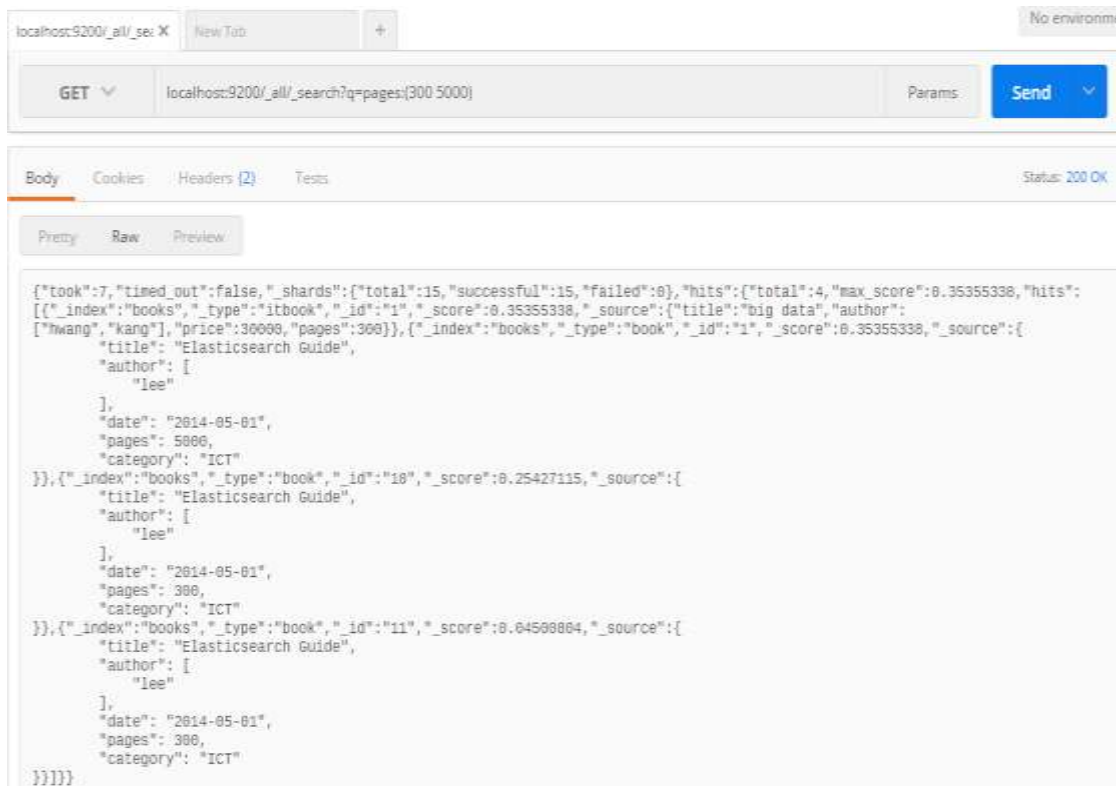
date:>2014-03-01&pages:300
는 3개이나

date:>2014-03-01+pages:300
는 양쪽 조건에 맞는 총 4개

Query string 처리 : contain

localhost:9200/_all/_search?q=pages:(300 5000)를 지정하고 검색

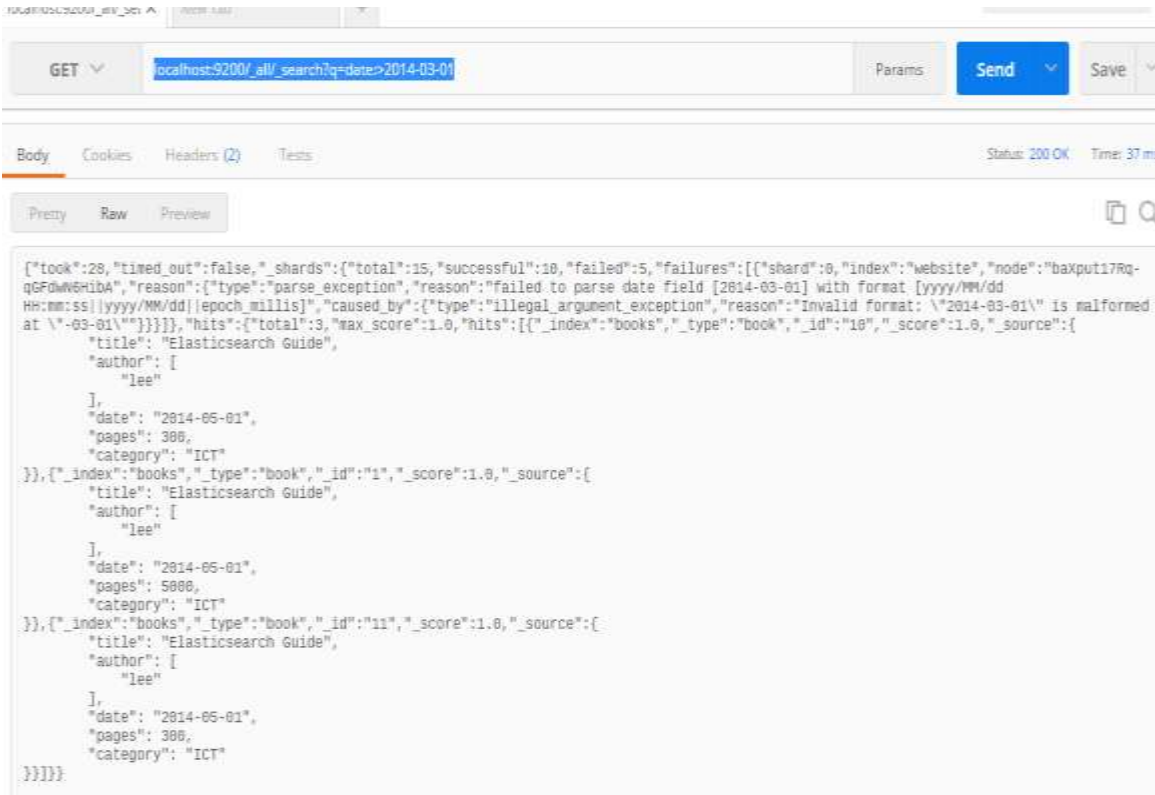
필드값을 괄호
표시하고 값들
을 blank로 구
분



```
{
  "took": 7,
  "timed_out": false,
  "_shards": {
    "total": 15,
    "successful": 15,
    "failed": 0
  },
  "hits": {
    "total": 4,
    "max_score": 0.35355338,
    "hits": [
      {
        "_index": "books",
        "_type": "itbook",
        "_id": "1",
        "_score": 0.35355338,
        "_source": {
          "title": "big data",
          "author": [
            "hwang",
            "kang"
          ],
          "price": 30000,
          "pages": 300
        }
      },
      {
        "_index": "books",
        "_type": "book",
        "_id": "1",
        "_score": 0.35355338,
        "_source": {
          "title": "Elasticsearch Guide",
          "author": [
            "lee"
          ],
          "date": "2014-05-01",
          "pages": 5000,
          "category": "ICT"
        }
      },
      {
        "_index": "books",
        "_type": "book",
        "_id": "10",
        "_score": 0.25427115,
        "_source": {
          "title": "Elasticsearch Guide",
          "author": [
            "lee"
          ],
          "date": "2014-05-01",
          "pages": 300,
          "category": "ICT"
        }
      },
      {
        "_index": "books",
        "_type": "book",
        "_id": "11",
        "_score": 0.64500004,
        "_source": {
          "title": "Elasticsearch Guide",
          "author": [
            "lee"
          ],
          "date": "2014-05-01",
          "pages": 300,
          "category": "ICT"
        }
      }
    ]
  }
}
```

Query string 처리 : 부등식

localhost:9200/_all/_search?q=date:>2014-03-01를 지정하고 검색



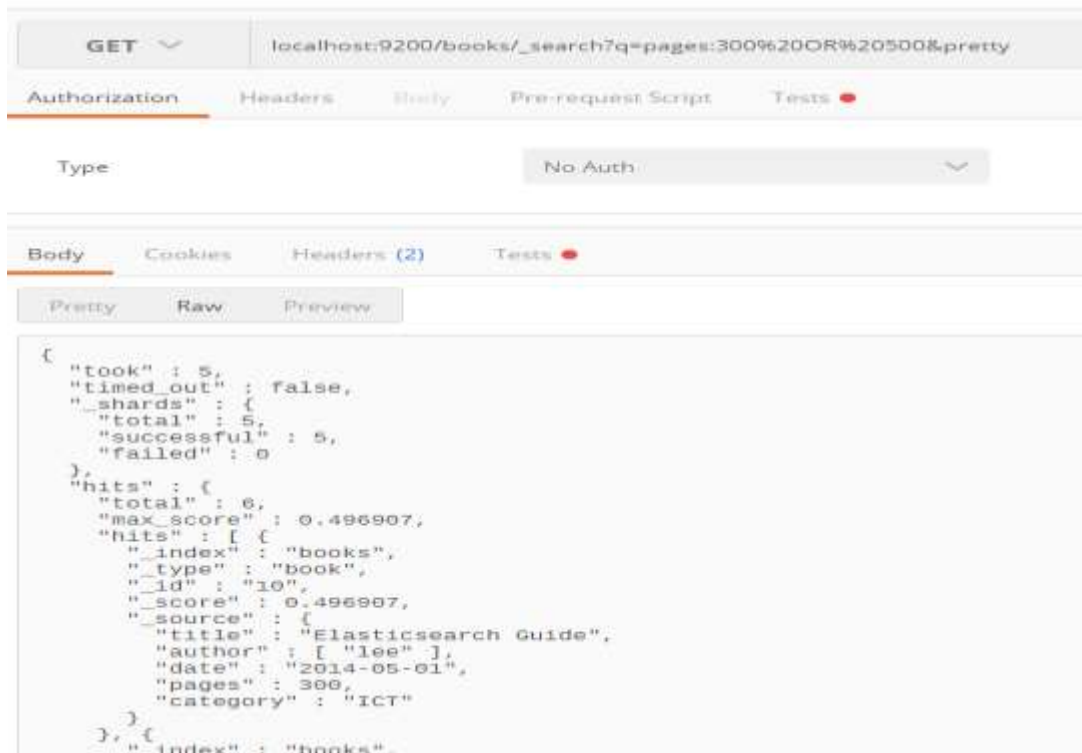
```
GET localhost:9200/_all/_search?q=date:>2014-03-01

{"took":28,"timed_out":false,"_shards":{"total":15,"successful":10,"failed":5,"failures":[{"shard":0,"index":"website","node":"baxput17Rq-q6FdwN6hibA","reason":{"type":"parse_exception","reason":"failed to parse date field [2014-03-01] with format [yyyy/MM/dd HH:mm:ss|yyyy/MM/dd|epoch_millis]","caused_by":{"type":"illegal_argument_exception","reason":"Invalid format: '\\2014-03-01\\' is malformed at '\\-03-01\\'"}}}]}, "hits":{"total":3,"max_score":1.0,"hits":[{"_index":"books","_type":"book","_id":"10","_score":1.0,"_source":{"title":"Elasticsearch Guide","author":{"name":"lee"},"date":"2014-05-01","pages":300,"category":"ICT"}}, {"_index":"books","_type":"book","_id":"1","_score":1.0,"_source":{"title":"Elasticsearch Guide","author":{"name":"lee"},"date":"2014-05-01","pages":5000,"category":"ICT"}}, {"_index":"books","_type":"book","_id":"11","_score":1.0,"_source":{"title":"Elasticsearch Guide","author":{"name":"lee"},"date":"2014-05-01","pages":300,"category":"ICT"}}]}}
```

필드값앞에 부
등식을 표시해
서 조회

Query string 처리 : AND/OR

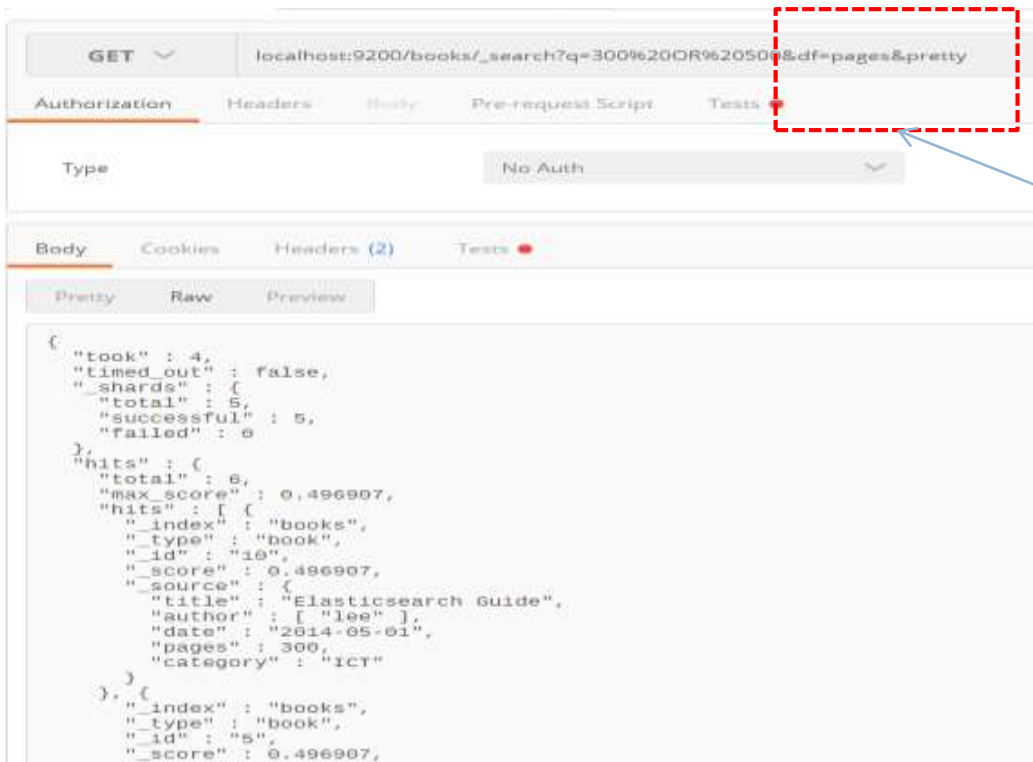
Pages 질의어 300 or 500을 가진 것을 검색
- 공백(%20)과 함께 표시



```
{
  "took": 5,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 6,
    "max_score": 0.496907,
    "hits": [ {
      "_index": "books",
      "_type": "book",
      "_id": "10",
      "_score": 0.496907,
      "_source": {
        "title": "Elasticsearch Guide",
        "author": [ "lee" ],
        "date": "2014-05-01",
        "pages": 300,
        "category": "ICT"
      }
    }, {
      "_index": "books",
```

Query string 처리 : df

필드명 대신 df(default field)를 매개변수로 사용



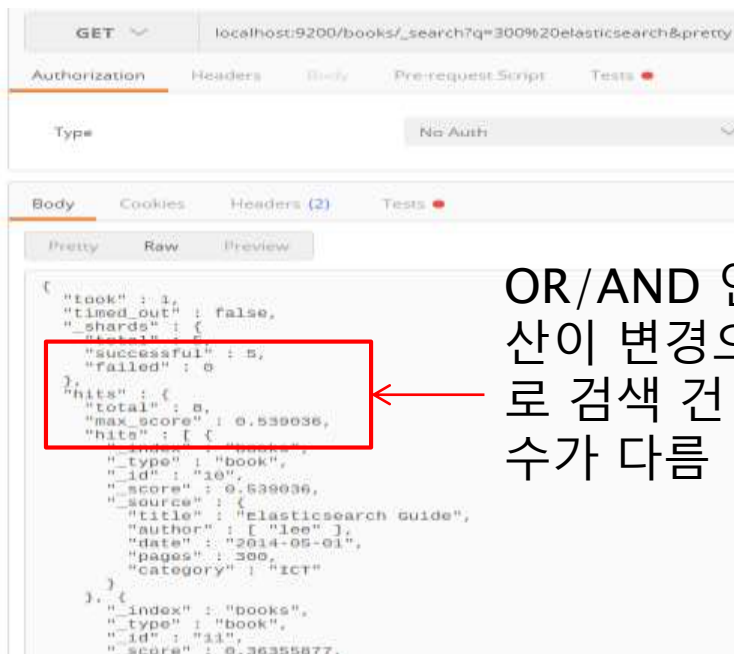
q=질의어만 표시하
고 실질적인 필드를
df로 정의해서 별도
로 지정

Query string 처리 : default op

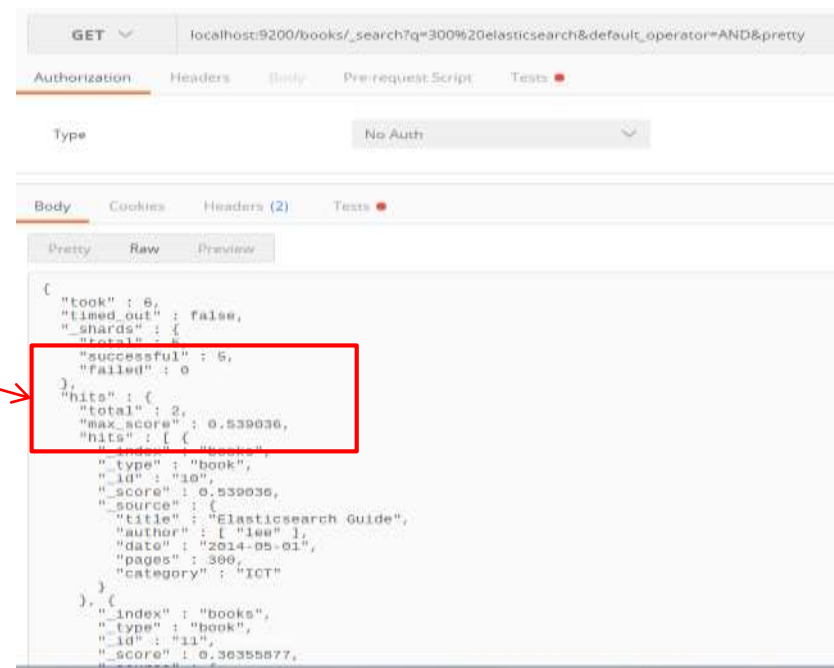
질의어와 질의어 사이에 AND, OR를 직접 입력도 가능하지만 default operator를 이용

질의어 사이 blank(OR)

default_operator로 변경



OR/AND 연산이 변경으로 검색 건수가 다름



QUERY DSL



_search API

_search 검색 방법

_search를 이용해서 다양한 것을 검색할 수 있음

| | |
|--|-------------------------------------|
| <code>/_search</code> | # 모든 인덱스 내의 모든 타입을 검색 |
| <code>/gb/_search</code> | # gb 인덱스 내의 모든 타입을 검색 |
| <code>/gb,us/_search</code> | # gb,us 인덱스 내의 모든 타입을 검색 |
| <code>/g*,u*/_search</code> | # g*,u*로 시작하는 인덱스 내의 모든 타입을 검색 |
| <code>/gb/user/_search</code> | # 인덱스 gb, 타입 user 내의 모든 다큐먼트 조회 |
| <code>/gb,us/user,tweet/_search</code> | # gb, us 인덱스 내의 user, tweet 타입 내 조회 |
| <code>/_all/user,tweet/_search</code> | # 모든 인덱스 내의 user와 tweet 타입 내 조회 |



Query DSL 이란

Query DSL

Query DSL을 별도로 정의해서 검색

```
GET /_search
{
  "query": YOUR_QUERY_HERE
}

curl XGET localhost:9200/_search -d '
{
  "query" : {
    "match" : {
      "title" : "elasticsearch"
    }
  }
}'
```

Query DSL 구조 : leaf 구문

Query 를 정의 시 하나의 query 구문을 사용해 검색

```
{
  QUERY_NAME: {
    FIELD_NAME: {
      ARGUMENT: VALUE,
      ARGUMENT: VALUE,...
    }
  }
}
```

```
{
  "query": {
    "match": {
      "tweet": "elasticsearch"
    }
  }
}
```

Query DSL 구조 : compound구문

Query 를 정의 시 다종의 query 구문을 사용해
검색

```
{
  "query" : {
    "bool" : {
      "must" : { "match" : { "title" : "elasticsearch" } },
      "must_not" : { "match" : { "name" : "lee" } },
      "should" : { "match" : { "pages" : 300 } }
    }
  }
}
```

Queries and Filters

Query와 Filter로 분리해서 검색을 정의할 수 있으면 특성에 맞도록 질의 구문을 작성해야 함

Filter 는 필드들의 값을 평가할 경우 Yes/No로 인지 되는 쿼리
→ 다양하게 사용되며 결과가 캐싱되면 응답속도가 빠름

Query는 단답형이 아닌 질의 스타일로 인지하는 쿼리
→ 텍스트 질의나 점수에 대한 질의에 사용

QUERY DSL : FILTER



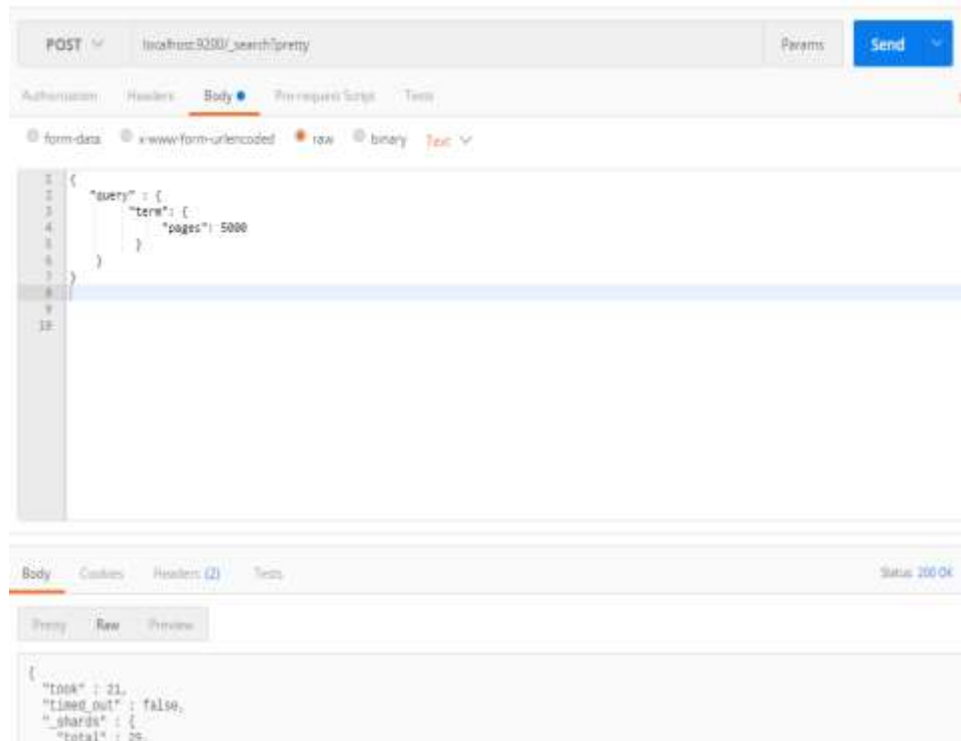
term

Term 필터

정확한 값 즉 일치되는 경우만 검색 (numbers, dates, Booleans, or not_analyzed exact-value string fields)

토큰을 Term이라고 하면
term 내의 필드 정의된 값을
토큰과 비교해서 처리

```
{
  "filter" : {
    "term": {
      "pages": 5000
    }
  }
}
```



Term 필터: 조회가 안되는 경우

Term filter 처리시 prefix 값으로 조회시 실제 조회가 되지 않음. 이런 경우 prefix query를 사용해야 함

```
{
  "filter" : {
    "term" : {
      "title" : "ela"
    }
  }
}
```

```
{
  "took" : 8,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 0,
    "max_score" : null,
    "hits" : [ ]
  }
}
```

Term 필터: 멀티 값처리

price 필드의 값을 array로 처리하면 멀티 값을 검색

```
POST localhost:9200/my_store/products/_search?pretty

{
  "query": {
    "filtered": {
      "filter": {
        "terms": {
          "price": [20, 30]
        }
      }
    }
  }
}
```

```
{
  "took": 1,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 2,
    "max_score": 1.0,
    "hits": [
      {
        "_index": "my_store",
        "_type": "products",
        "_id": "2",
        "_score": 1.0,
        "_source": {
          "price": 20,
          "productID": "KDKE-B-9947-#KL5"
        }
      },
      {
        "_index": "my_store",
        "_type": "products",
        "_id": "3",
        "_score": 1.0,
        "_source": {
          "price": 30,
          "productID": "JODL-X-1937-#pV7"
        }
      }
    ]
  }
}
```



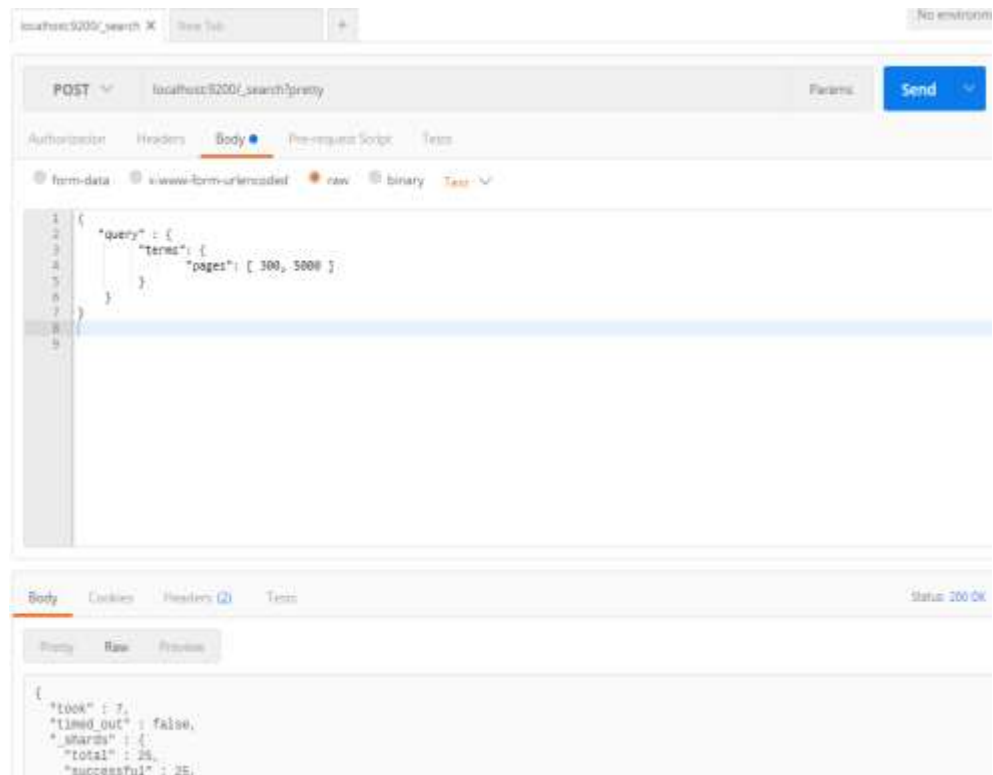
terms

Terms 필터

Term 필터와 유사하지만 필드안에 여러 개의 값을 동시에 검색할 경우 사용

2개의 토큰 값을 동시에
검색하려면 array로 입력
해서 처리

```
{
  "filter" : {
    "terms": {
      "pages": [ 300, 5000 ]
    }
  }
}
```





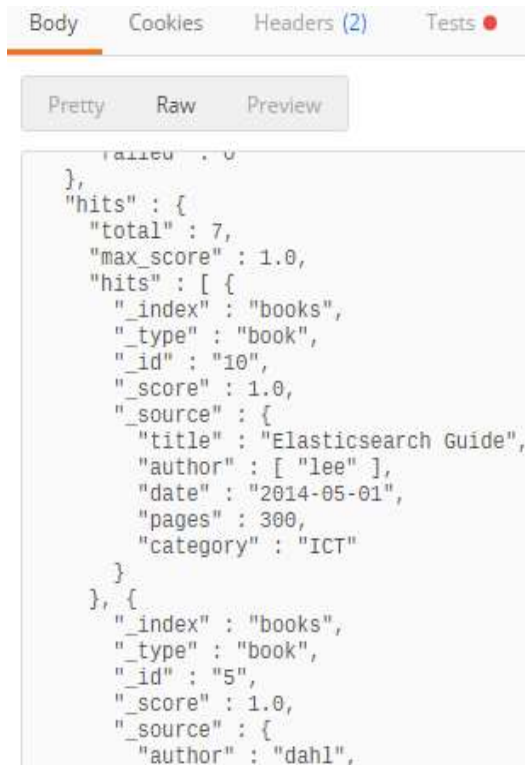
range

range Filter

특정 범위에 해당된 필드들이 다큐먼트를 검색

gt : Greater than
gte : Greater than or equal to
lt : Less than
lte : Less than or equal to

```
{
  "filter": {
    "range": {
      "pages": {
        "gte": 300,
        "lte": 5000
      }
    }
  }
}
```



```
{
  "hits": {
    "total": 7,
    "max_score": 1.0,
    "hits": [ {
      "_index": "books",
      "_type": "book",
      "_id": "10",
      "_score": 1.0,
      "_source": {
        "title": "Elasticsearch Guide",
        "author": [ "lee" ],
        "date": "2014-05-01",
        "pages": 300,
        "category": "ICT"
      }
    }, {
      "_index": "books",
      "_type": "book",
      "_id": "5",
      "_score": 1.0,
      "_source": {
        "author": "dahl",
```



and, or, not

not Filter

먼저 처리된 필터를 다시 처리하므로 다른 필터들과 달리 캐시되지 않는다

```
{
  "filter" : {
    "not" : {
      "range" : {
        "pages" : {
          "gte" : 500,
          "lte" : 5000
        }
      }
    }
  }
}
```

Pretty Raw Preview

```
{
  "max_score" : 1.0,
  "hits" : [ {
    "_index" : "books",
    "_type" : "book",
    "_id" : "25",
    "_score" : 1.0,
    "_source" : {
      "title" : 123
    }
  }, {
    "_index" : "books",
    "_type" : "book",
    "_id" : "12",
    "_score" : 1.0,
    "_source" : {
      "title" : "xxx"
    }
  }, {
    "_index" : "books",
    "_type" : "book",
    "_id" : "10",
    "_score" : 1.0,
    "_source" : {
      "title" : "Elasticsearch Guide",
      "author" : [ "lee" ],
      "date" : "2014-05-01",
      "pages" : 300,
      "category" : "ICT"
    }
  }
]
```

and/or Filter

두개 field를 가진 필터 처리

```
{
  "filter": {
    "and": [
      {
        "range": {
          "pages": {
            "gte": 300,
            "lte": 5000
          }
        }
      },
      {
        "term": { "title": "elasticsearch" }
      }
    ]
  }
}
```

```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "10",
      "_score" : 1.0,
      "_source" : {
        "title" : "Elasticsearch Guide",
        "author" : [ "lee" ],
        "date" : "2014-05-01",
        "pages" : 300,
        "category" : "ICT"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "1",
      "_score" : 1.0,
      "_source" : {
        "title" : "Elasticsearch Guide",
        "author" : [ "lee" ],
        "date" : "2014-05-01",
        "pages" : 5000,
        "category" : "ICT"
      }
    }
  ]
}
```



bool

bool filter

내부 질의로 다른 쿼리를 포함해서 사용하는 검색

```
{
  "filter" : {
    "bool" : {
      "must" : {
        "term" : {"title": "big"}
      },
      "must_not" : {
        "term" : { "title" : "elasticsearch"}
      },
      "should" : {
        "term" : {"plot" : "elasticsearch"}
      }
    }
  }
}
```

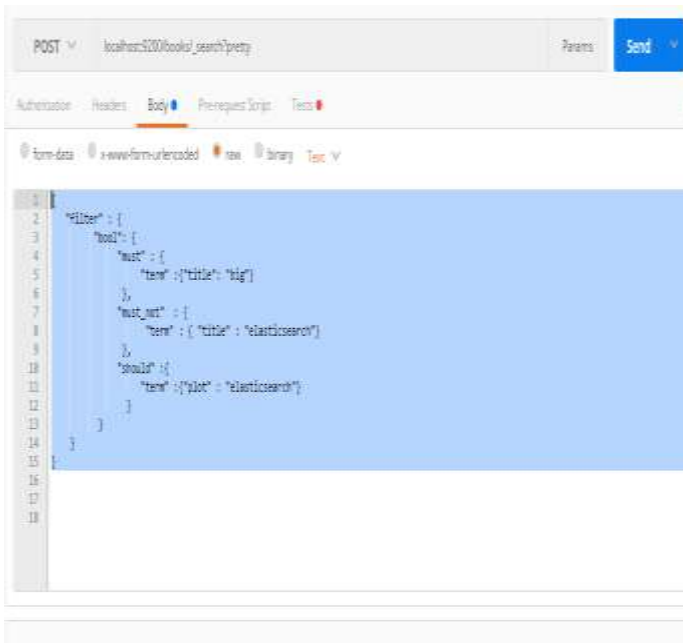
must
매칭 필수, AND 조건.

must_not
매칭 불가 NOT 조건

should
반드시 해당될 필요는 없지만 해당된다면
더 높은 스코어를 가지는 조건, OR 조건

bool filter 처리 예시

title에 big이 있고 plot에 elasticsearch가 있으면서 title에는 절대 elasticsearch가 없는 경우만 검색



```
{
  "took" : 4,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "5",
      "_score" : 1.0,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : "big picture",
        "plot" : "elasticsearch"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "4",
      "_score" : 1.0,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : "big picture",
        "plot" : "elasticsearch"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "3",
      "_score" : 1.0,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : "big picture",
        "plot" : "elasticsearch"
      }
    }
  ]
}
```

QUERY DSL : QUERY

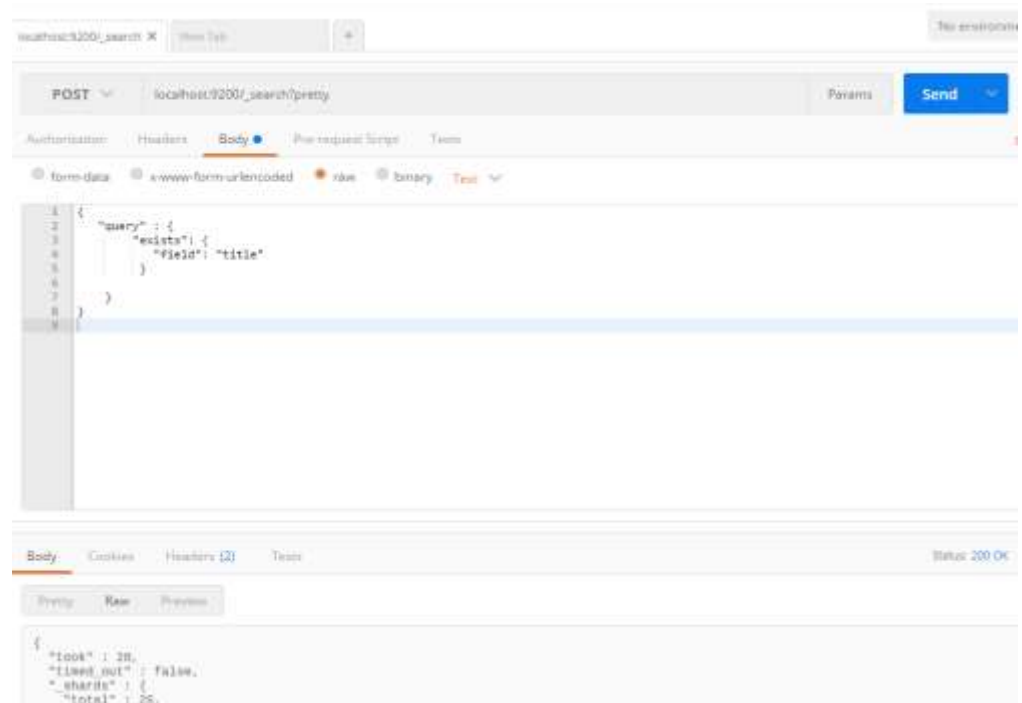


Exist

exists and missing Filters

다큐먼트 내에 존재하는 필드가 있는 여부 확인

```
{  
  "query" : {  
    "exists": {  
      "field": "title"  
    }  
  }  
}
```





Match

Match Query : 필드매핑

형태소 분석을 거친 뒤 분석된 질의문으로 검색



```
POST localhost:9200/books/_search?pretty

{
  "query": {
    "match": {
      "title": "elasticsearch"
    }
  }
}
```

```
{
  "took" : 6,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : 0.8784157,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "10",
      "_score" : 0.8784157,
      "_source" : {
        "title" : "Elasticsearch Guide",
        "author" : [ "lee" ],
        "date" : "2014-05-01",
        "pages" : 300,
        "category" : "ICT"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "1",
      "_score" : 0.625,
      "_source" : {
        "title" : "Elasticsearch Guide",
        "author" : [ "lee" ],
        "date" : "2014-05-01",
        "pages" : 5000,
        "category" : "ICT"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "11",
      "_score" : 0.625,
      "_source" : {
        "title" : "Elasticsearch Guide",
        "author" : [ "lee" ]
      }
    }
  ]
}
```

Match Query : 세부질의(and)

각 필드에 세부 query와 operator를 정의해서 상세 검색

필드 내부 값을 세부 query를 지정해서 사용 가능 Operator이 기본값은 or로 처리

```
POST localhost:9200/books/_search?pretty

{
  "query": {
    "match": {
      "title": {
        "query": "big data",
        "operator": "and"
      }
    }
  }
}
```

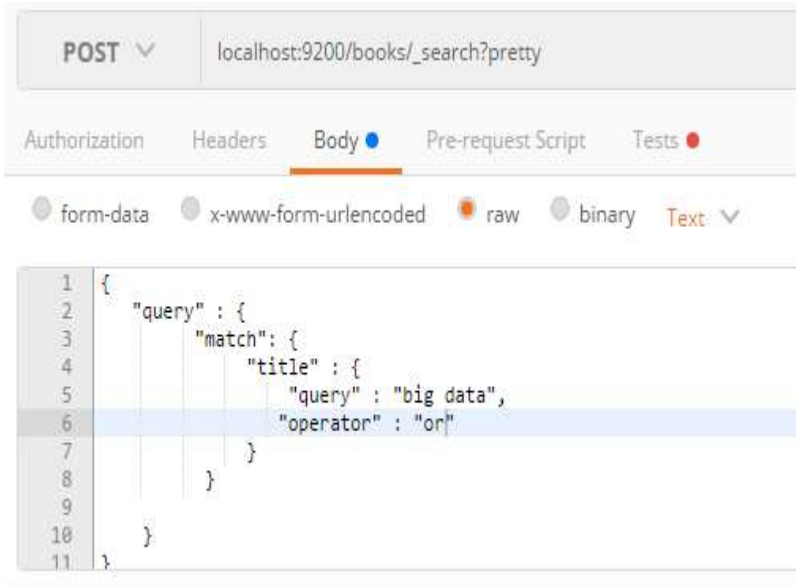
```
Pretty Raw Preview

{
  "took": 3,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 0.8838835,
    "hits": [ {
      "_index": "books",
      "_type": "itbook",
      "_id": "1",
      "_score": 0.8838835,
      "_source": {
        "title": "big data",
        "author": [ "hwang", "kang" ],
        "price": 30000,
        "pages": 300
      }
    } ]
  }
}
```

Match Query : 세부질의(or)

각 필드에 세부 query와 operator를 정의해서 상세 검색

필드 내부 값을 세부 query를 지정해서 사용 가능 Operator이 기본값은 or로 처리




```
{
  "took" : 9,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 4,
    "max_score" : 0.8838835,
    "hits" : [ {
      "_index" : "books",
      "_type" : "itbook",
      "_id" : "1",
      "_score" : 0.8838835,
      "_source" : {
        "title" : "big data",
        "author" : [ "hwang", "kang" ],
        "price" : 30000,
        "pages" : 300
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "5",
      "_score" : 0.24439743,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : "big picture",
        "plot" : "elasticsearch"
      }
    }
  ]
}
```

Match Query : phrase

내부 검색을 하나의 문구로 처리하기 위해 type
을 phrase 지정 후 검색

하나의 구문으로 인식되게 처리됨



```
1 {
2   "query": {
3     "match": {
4       "title": {
5         "query": "big data",
6         "type": "phrase"
7       }
8     }
9   }
10 }
11 }
```

```
{
  "took": 3,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 1.25,
    "hits": [ {
      "_index": "books",
      "_type": "itbook",
      "_id": "1",
      "_score": 1.25,
      "_source": {
        "title": "big data",
        "author": [ "hwang", "kang" ],
        "price": 30000,
        "pages": 300
      }
    } ]
  }
}
```

Match Query : minimum_should_match

필드 내의 문구가 특정 비율 만큼 맞을 경우 검색

```
{
  "query": {
    "match": {
      "title": {
        "query": "quick brown dog",
        "minimum_should_match": "75%"
      }
    }
  }
}
```

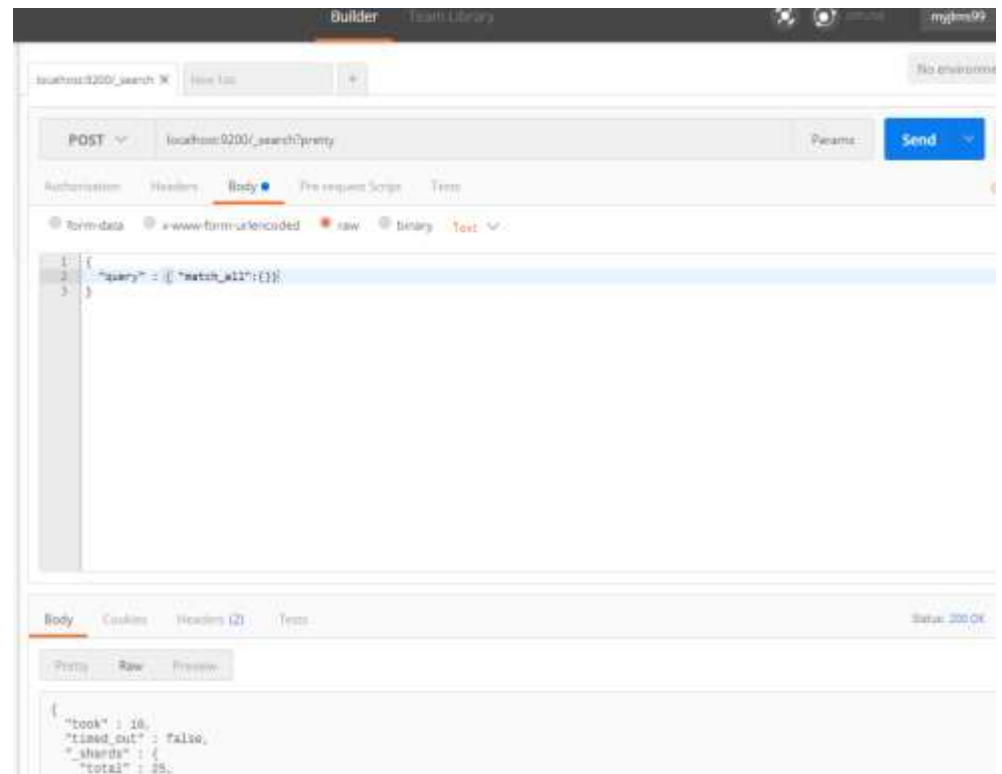



match_all

match_all Query

모든 다큐먼트를 전부 조회

```
{  
  "query" : {  
    "match_all": {}  
  }  
}
```





multi_match

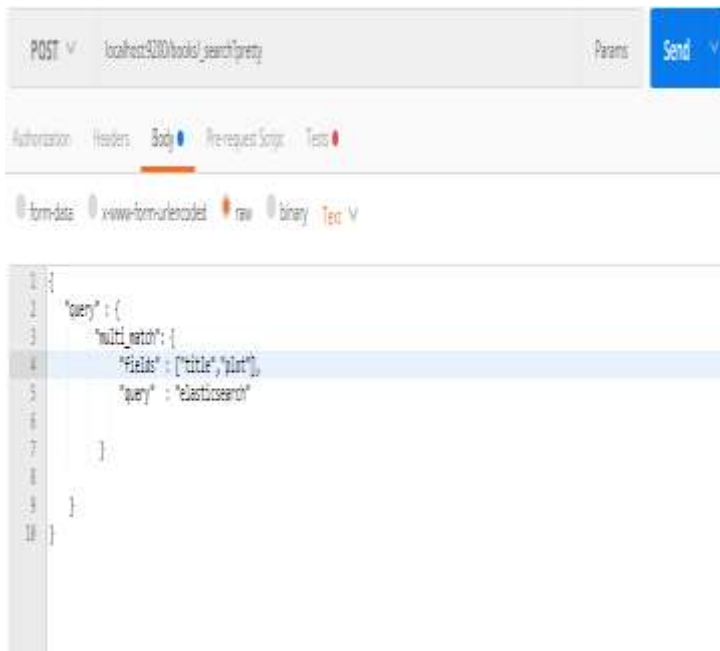
multi_match Query

다수의 필드에 동일한 값을 검색할 경우 사용

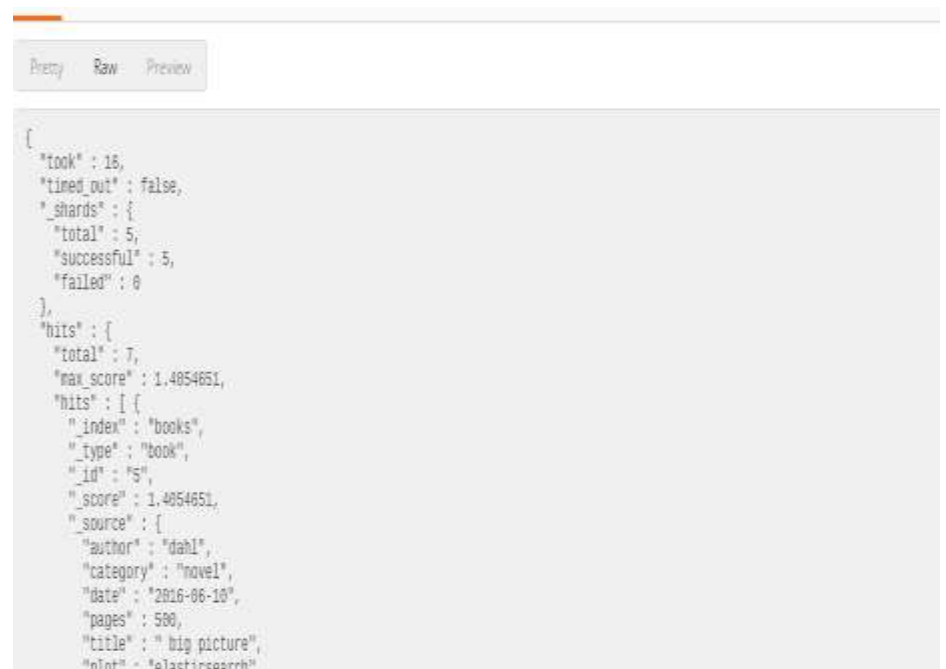
```
{
  "query": {
    "multi_match": {
      "fields": ["title", "plot"],
      "query": "elasticsearch"
    }
  }
}
```

multi_match Query 처리예시

Fields에 다수 필드를 넣고 query에 값을 넣어 조회



```
POST localhost:9200/books/_search?pretty
{
  "query": {
    "multi_match": {
      "fields": ["title", "plot"],
      "query": "elasticsearch"
    }
  }
}
```



```
{
  "took": 18,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 1.4054651,
    "hits": [ {
      "_index": "books",
      "_type": "book",
      "_id": "5",
      "_score": 1.4054651,
      "_source": {
        "author": "dahl",
        "category": "novel",
        "date": "2016-06-10",
        "pages": 500,
        "title": "big picture",
        "plot": "elasticsearch"
      }
    } ]
  }
}
```



bool

bool query

내부 질의로 다른 쿼리를 포함해서 사용하는 검색

```
{
  "query" : {
    "bool" : {
      "must" : {
        "term" : {"title": "big"}
      },
      "must_not" : {
        "term" : { "title" : "elasticsearch"}
      },
      "should" : {
        "term" : {"plot" : "elasticsearch"}
      }
    }
  }
}
```

must

매칭 필수, AND 조건.

must_not

매칭 불가 NOT 조건

should

반드시 해당될 필요는 없지만 해당된다면
더 높은 스코어를 가지는 조건, OR 조건

bool query : match(or) → bool

Match 쿼리 내의 2개 term을 bool 내의 should 속성 내의 2개 쿼리로 표현 가능

```
{  
  "match": { "title": "brown fox"}  
}
```

```
{  
  "bool": {  
    "should": [  
      { "term": { "title": "brown" } },  
      { "term": { "title": "fox" } }  
    ]  
  }  
}
```


bool query : match(and) → bool

Match 쿼리 내의 2개 term과 and 처리시 을
bool 내의 must속성 내의 2개 쿼리로 표현 가능

```
{
  "match": {
    "title": {
      "query": "brown fox",
      "operator": "and"
    }
  }
}
```


```
{
  "bool": {
    "must": [
      { "term": { "title": "brown" } },
      { "term": { "title": "fox" } }
    ]
  }
}
```

bool query : match(%) → bool

Match 쿼리 내의 3개 term과 % 처리시 을 bool
내의 should속성 내의 3개 쿼리와 minimum
match를 사용해서 표현 가능

```
{
  "match": {
    "title": {
      "query": "quick brown fox",
      "minimum_should_match": "75%"
    }
  }
}
```

```
{
  "bool": {
    "should": [
      { "term": { "title": "brown" } },
      { "term": { "title": "fox" } },
      { "term": { "title": "quick" } }
    ],
    "minimum_should_match": 2
  }
}
```



Query_string

query_string query 1

Query string 처리를 query 구문 내에서 “<필드명>:값”을 문자열 처리후 검색: big 이 들어간 문서가 전부 조회

```
{
  "query" : {
    "query_string": {
      "query" : "title: big"
    }
  }
}
```

```
failed : 0
},
"hits" : {
  "total" : 4,
  "max_score" : 0.8784157,
  "hits" : [ {
    "_index" : "books",
    "_type" : "book",
    "_id" : "5",
    "_score" : 0.8784157,
    "_source" : {
      "author" : "dahl",
      "category" : "novel",
      "date" : "2016-06-10",
      "pages" : 500,
      "title" : " big picture",
      "plot" : "elasticsearch"
    }
  }, {
    "_index" : "books",
    "_type" : "itbook",
    "_id" : "1",
    "_score" : 0.625,
    "_source" : {
```

query_string query 2

query_string 처리를 query에는 검색 문자열, 별도의 필드와 오퍼레이터를 주고 검색: big data가 들어가 있는 다큐먼트만 조회

```
{
  "query" : {
    "query_string" : {
      "query" : "big data",
      "default_field" : "title",
      "default_operator" : "and"
    }
  }
}
```

```
{
  "took" : 3,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 0.8838835,
    "hits" : [ {
      "_index" : "books",
      "_type" : "itbook",
      "_id" : "1",
      "_score" : 0.8838835,
      "_source" : {
        "title" : "big data",
        "author" : [ "hwang", "kang" ],
        "price" : 30000,
        "pages" : 300
      }
    } ]
  }
}
```

query_string query 3

query_string 처리를 query에는 검색 문자열, 별도의 필드와 오퍼레이터를 주고 검색: big 또는 data가 들어가 있는 다큐먼트를 포함해서 조회

```
{
  "query" : {
    "query_string" : {
      "query" : "big data",
      "default_field" : "title",
      "default_operator" : " ||| "
    }
  }
}
```

```
}, {
  "hits" : {
    "total" : 4,
    "max_score" : 0.8838835,
    "hits" : [ {
      "_index" : "books",
      "_type" : "itbook",
      "_id" : "1",
      "_score" : 0.8838835,
      "_source" : {
        "title" : "big data",
        "author" : [ "hwang", "kang" ],
        "price" : 30000,
        "pages" : 300
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "5",
      "_score" : 0.24439743,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : "big picture",
        "plot" : "elasticsearch"
      }
    }
  ]
}
```



prefix

query_string query 1

Term filter처럼 형태소 분석이 적용되지 않지만
접두어로만 검색이 된다

```
{
  "query" : {
    "prefix": {
      "title" : "ela"
    }
  }
}
```

Pretty Raw Preview

```
{
  "took" : 12,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "10",
      "_score" : 1.0,
      "_source" : {
        "title" : "Elasticsearch Guide",
        "author" : [ "lee" ],
        "date" : "2014-05-01",
        "pages" : 300,
        "category" : "ICT"
      }
    }
  ], {
    "_index" : "books",
    "_type" : "book",
```



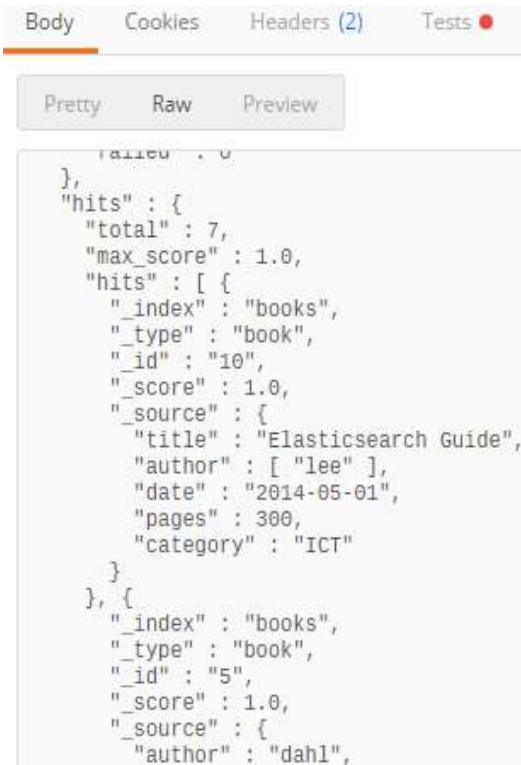

range

range query

특정 범위에 해당된 필드들이 다큐먼트를 검색

gt : Greater than
gte : Greater than or equal to
lt : Less than
lte : Less than or equal to

```
{
  "query": {
    "range": {
      "pages": {
        "gte": 300,
        "lte": 5000
      }
    }
  }
}
```



Body Cookies Headers (2) Tests ●

Pretty Raw Preview

```
{
  "hits": {
    "total": 7,
    "max_score": 1.0,
    "hits": [ {
      "_index": "books",
      "_type": "book",
      "_id": "10",
      "_score": 1.0,
      "_source": {
        "title": "Elasticsearch Guide",
        "author": [ "lee" ],
        "date": "2014-05-01",
        "pages": 300,
        "category": "ICT"
      }
    }, {
      "_index": "books",
      "_type": "book",
      "_id": "5",
      "_score": 1.0,
      "_source": {
        "author": "dahl",

```



fuzzy

fuzzy query : 문자열 처리

레벤슈타인 거리 알고리즘을 기반으로 유사 단어의 검색을 지원 bag을 입력시 big을 검색

```
{
  "query": {
    "fuzzy": {
      "title": "bag"
    }
  }
}
```

```
Pretty Raw Preview
{
  "hits": {
    "total": 4,
    "max_score": 0.8784157,
    "hits": [ {
      "_index": "books",
      "_type": "book",
      "_id": "5",
      "_score": 0.8784157,
      "_source": {
        "author": "dahl",
        "category": "novel",
        "date": "2016-06-10",
        "pages": 500,
        "title": "big picture",
        "plot": "elasticsearch"
      }
    }, {
      "_index": "books",
      "_type": "itbook",
      "_id": "1",
      "_score": 0.625,
      "_source": {
        "title": "big data",
        "author": [ "hwang", "kang" ],
        "price": 30000,
        "pages": 300
      }
    }
  ]
}
```

fuzzy query : 범위처리

특정 값을 value에 정의하고 fuzziness는 +/- 범위(200)까지이 검색

```
{
  "query": {
    "fuzzy": {
      "pages": {
        "value": 500,
        "fuzziness": 200
      }
    }
  }
}
```

```
"hits" : {
  "total" : 6,
  "max_score" : 1.0,
  "hits" : [ {
    "_index" : "books",
    "_type" : "book",
    "_id" : "10",
    "_score" : 1.0,
    "_source" : {
      "title" : "Elasticsearch Guide",
      "author" : [ "lee" ],
      "date" : "2014-05-01",
      "pages" : 300,
      "category" : "ICT"
    }
  }, {
    "_index" : "books",
    "_type" : "book",
    "_id" : "5",
    "_score" : 1.0,
    "_source" : {
      "author" : "dahl",
      "category" : "novel",
      "date" : "2016-06-10",
      "pages" : 500,
      "title" : "big picture",
      "plot" : "elasticsearch"
    }
  }, {
    "_index" : "books",
    "_type" : "book",
    "_id" : "4",
    "_score" : 1.0,
    "_source" : {
      "author" : "dahl".
    }
  }
]
```



nested

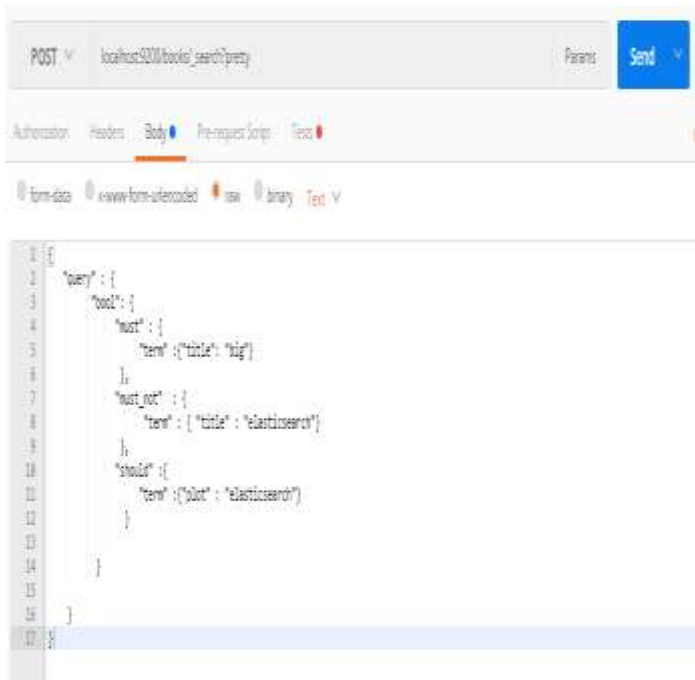
bool query : 쿼리 조합하기

Bool 질의 내부에 match 질의를 사용해서 쿼리를 조합해서 질의하기

```
{
  "query": {
    "bool": {
      "must": { "match": { "title": "quick" } },
      "must_not": { "match": { "title": "lazy" } },
      "should": [
        { "match": { "title": "brown" } },
        { "match": { "title": "dog" } }
      ]
    }
  }
}
```

bool query처리예시

title에 big이 있고 plot에 elasticsearch가 있으면서 title에는 절대 elasticsearch가 없는 경우만 검색



```
POST localhost:5200/_search/prety
{
  "query": {
    "bool": {
      "must": {
        "term": { "title": "big" }
      },
      "must_not": {
        "term": { "title": "elasticsearch" }
      },
      "should": {
        "term": { "plot": "elasticsearch" }
      }
    }
  }
}
```



```
{
  "took": 5,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 4,
    "max_score": 1.6149478,
    "hits": [
      {
        "_index": "books",
        "_type": "book",
        "_id": "5",
        "score": 1.6149478,
        "_source": {
          "author": "dahl",
          "category": "novel",
          "date": "2016-06-10",
          "pages": 500,
          "title": "big picture",
          "plot": "elasticsearch"
        }
      },
      {
        "_index": "books",
        "_type": "book",
        "_id": "4",
        "score": 1.6149478,
        "_source": {
          "author": "dahl",
          "category": "novel",
          "date": "2016-06-10",
          "pages": 500,
          "title": "big picture",
          "plot": "elasticsearch"
        }
      }
    ]
  }
}
```


중첩(nested) 검색

중첩 객체 내부 필드를 이용해서 검색



```
1 {
2   "query": {
3     "nested": {
4       "path": "user",
5       "query": {
6         "bool": {
7           "must": [
8             { "match": { "user.first": "Alice" } }
9           ]
10        }
11      }
12    }
13  }
14 }
```



```
{
  "took" : 14,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.4054651,
    "hits" : [ {
      "_index" : "my_index",
      "_type" : "my_type",
      "_id" : "1",
      "_score" : 1.4054651,
      "_source" : {
        "group" : "fans",
        "user" : [ {
          "first" : "John",
          "last" : "Smith"
        }, {
          "first" : "Alice",
          "last" : "White"
        } ]
      }
    } ]
  }
}
```

중첩(nested) 검색- 조건1

객체단위로 검색하므로 다른 경우는 결과가 없음

```
POST localhost:9200/my_index/_search?pretty
{
  "query": {
    "nested": {
      "path": "user",
      "query": {
        "bool": {
          "must": [
            { "match": { "user.first": "Alice" } },
            { "match": { "user.last": "Smith" } }
          ]
        }
      }
    }
  }
}
```

```
Pretty Raw Preview
{
  "took" : 14,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 0,
    "max_score" : null,
    "hits" : [ ]
  }
}
```

중첩(nested) 검색- 조건 2

객체단위(Alice White)로 검색하므로 다큐먼트가 검색

```
POST localhost:9200/my_index/_search?pretty

Authorization Headers Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary Text

1 {
2   "query": {
3     "nested": {
4       "path": "user",
5       "query": {
6         "bool": {
7           "must": [
8             { "match": { "user.first": "Alice" } },
9             { "match": { "user.last": "White" } }
10          ]
11        }
12      }
13    }
14  }
15 }
16
17
18
19
```

```
Pretty Raw Preview

{
  "took" : 6,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.987628,
    "hits" : [ {
      "_index" : "my_index",
      "_type" : "my_type",
      "_id" : "1",
      "_score" : 1.987628,
      "_source" : {
        "group" : "fans",
        "user" : [ {
          "first" : "John",
          "last" : "Smith"
        }, {
          "first" : "Alice",
          "last" : "White"
        } ]
      }
    } ]
  }
}
```

중첩(nested) 검색- inner처리

Inner_hits에 대한 정보를 별도로 출력이 가능함

```
POST localhost:9200/my_index/_search?pretty

{
  "query": {
    "nested": {
      "path": "user",
      "query": {
        "bool": {
          "must": [
            { "match": { "user.first": "Alice" } },
            { "match": { "user.last": "White" } }
          ]
        }
      },
      "inner_hits": {
        "highlight": {
          "fields": {
            "user.first": {}
          }
        }
      }
    }
  }
}
```

```
{
  "_id": "1",
  "_score": 1.987628,
  "_source": {
    "group": "fans",
    "user": [
      {
        "first": "John",
        "last": "Smith"
      },
      {
        "first": "Alice",
        "last": "White"
      }
    ]
  },
  "inner_hits": {
    "user": {
      "hits": {
        "total": 1,
        "max_score": 1.987628,
        "hits": [
          {
            "_index": "my_index",
            "_type": "my_type",
            "_id": "1",
            "_nested": {
              "field": "user",
              "offset": 1
            },
            "_score": 1.987628,
            "_source": {
              "first": "Alice",
              "last": "White"
            },
            "highlight": {
              "user.first": [ "<em>Alice</em>" ]
            }
          }
        ]
      }
    }
  }
}
```



boosting

질의시 boost 정하기

Match되는 필드에 대해 가중치(boost)를 부여

```
{
  "query": {
    "bool": {
      "must": {
        "match": {
          "content": {
            "query": "full text search",
            "operator": "and"
          }
        }
      },
      "should": [
        { "match": {
          "content": {
            "query": "Elasticsearch",
            "boost": 3
          }
        } },
        { "match": {
          "content": {
            "query": "Lucene",
            "boost": 2
          }
        } }
      ]
    }
  }
}
```

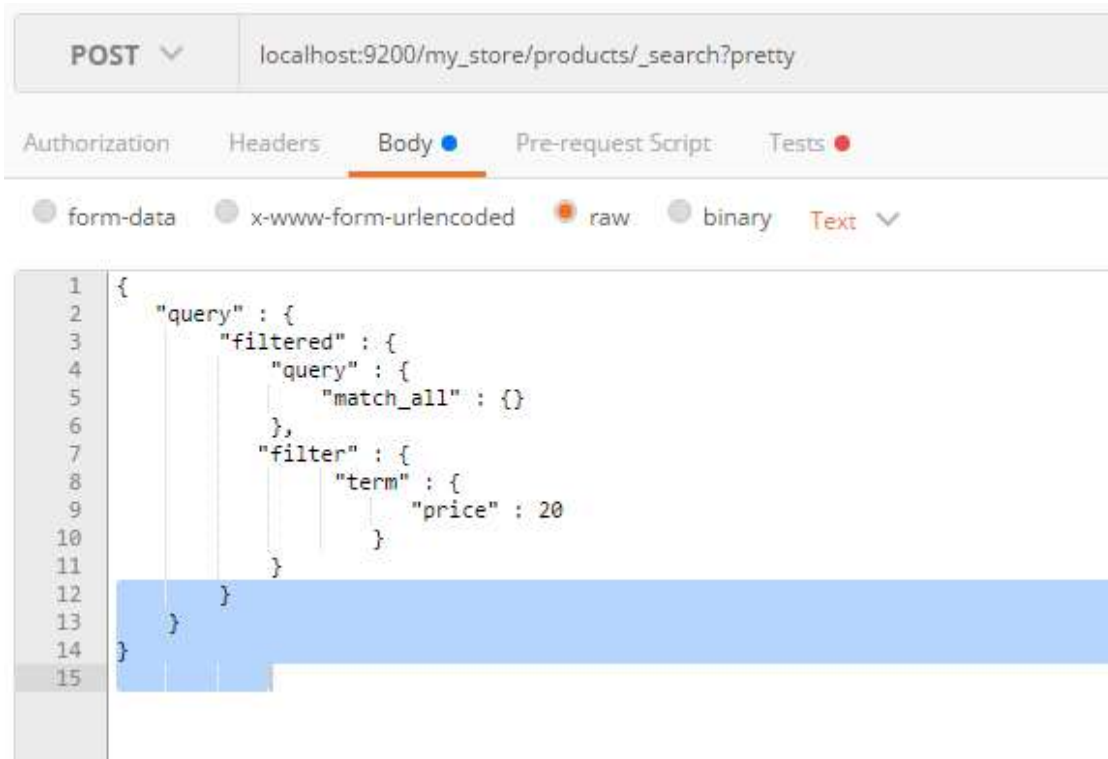
QUERY DSL : FILTERD



filetered

filtered

Query와 filter를 전부 사용하고 싶을 경우 사용



```
1 {
2   "query" : {
3     "filtered" : {
4       "query" : {
5         "match_all" : {}
6       },
7       "filter" : {
8         "term" : {
9           "price" : 20
10        }
11      }
12    }
13  }
14 }
15 }
```

Query가 실행되고
다음에 filter가 실행

Filtered 처리 결과

저장된 3개의 다큐먼트 중에 price가 20인 값만
필터링 처리

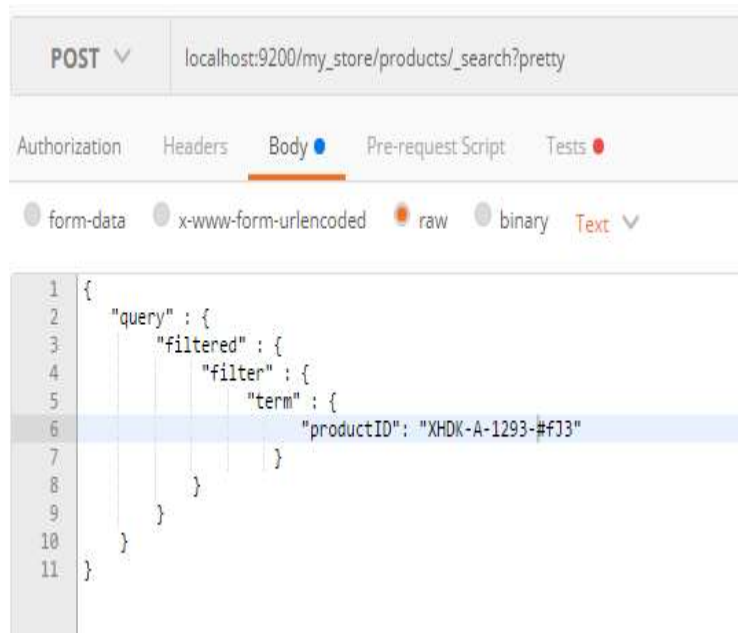
```
Pretty Raw Preview
{
  "took" : 6,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 1,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "my_store",
      "_type" : "products",
      "_id" : "2",
      "_score" : 1.0,
      "_source" : {
        "price" : 20,
        "productID" : "KDKE-B-9947-#KL5"
      }
    } ]
  }
}
```



Filetered:term 처리

Filtered: term의 값을 텍스트

term filter를 사용할 경우 텍스트 인식시 주의 사항:
문장이 인덱싱되므로 값으로 처리시 결과가 없음



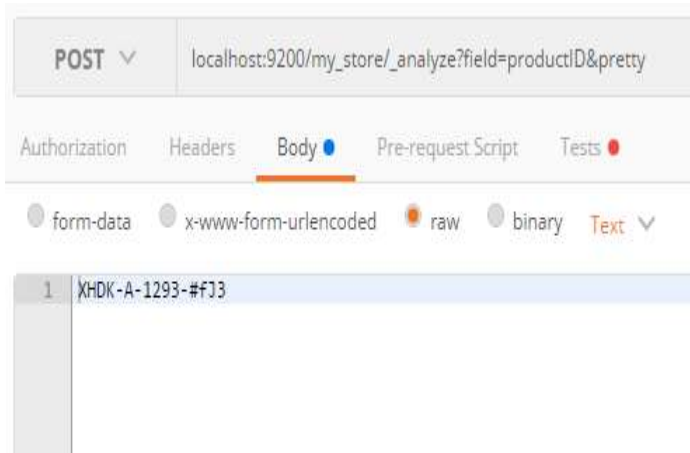
```
1 {
2   "query": {
3     "filtered": {
4       "filter": {
5         "term": {
6           "productID": "XHDK-A-1293-#fj3"
7         }
8       }
9     }
10  }
11 }
```



```
{
  "took" : 2,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 0,
    "max_score" : null,
    "hits" : [ ]
  }
}
```

Filtered: 텍스트값 analyzed

productID내의 텍스트를 분석해보면 토큰단위로 분리되어 있어 term filter를 이용해서 처리하지 못함



```
{
  "tokens" : [ {
    "token" : "xhdk",
    "start_offset" : 0,
    "end_offset" : 4,
    "type" : "<ALPHANUM>",
    "position" : 0
  }, {
    "token" : "a",
    "start_offset" : 5,
    "end_offset" : 6,
    "type" : "<ALPHANUM>",
    "position" : 1
  }, {
    "token" : "1293",
    "start_offset" : 7,
    "end_offset" : 11,
    "type" : "<NUM>",
    "position" : 2
  }, {
    "token" : "fj3",
    "start_offset" : 13,
    "end_offset" : 16,
    "type" : "<ALPHANUM>",
    "position" : 3
  } ]
}
```

Filtered: 해결방법

term filter 처리하기 위해서는 분석이 안되도록 정의한 후 저장해야 함

```
{
  "mappings" : {
    "products" : {
      "properties" : {
        "productID" : {
          "type" : "string",
          "index" : "not_analyzed"
        }
      }
    }
  }
}
```



Filetered:bool 처리

Filtered : bool

price=10 or price=20 이고 price=30이 아닌
것을 검색

```
POST localhost:9200/my_store/products/_search?pretty

{
  "query": {
    "filtered": {
      "filter": {
        "bool": {
          "should": [
            { "term": { "price": 10 } },
            { "term": { "price": 20 } }
          ],
          "must_not": {
            "term": { "price": 30 }
          }
        }
      }
    }
  }
}
```

```
{
  "took" : 6,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "my_store",
      "_type" : "products",
      "_id" : "2",
      "_score" : 1.0,
      "_source" : {
        "price" : 20,
        "productID" : "KDKE-B-9947-#kL5"
      }
    }, {
      "_index" : "my_store",
      "_type" : "products",
      "_id" : "1",
      "_score" : 1.0,
      "_source" : {
        "price" : 10,
        "productID" : "XHDK-A-1293-#fJ3"
      }
    } ]
  }
}
```




Filetered:nested bool 처리

Filtered : nested bool

price=10 or price=30 인 것을 검색

```
POST localhost:9200/my_store/products/_search?pretty
Authorization Headers Body Pre-request Script Tests
form-data x-www-form-urlencoded raw binary Text
1 {
2   "query": {
3     "filtered": {
4       "filter": {
5         "bool": {
6           "should": [
7             {
8               "term": {"price": 10}
9             },
10            {
11              "bool": {
12                "must": [
13                  {
14                    "term": {"price": 30}
15                  }
16                ]
17              }
18            }
19          ]
20        }
21      }
22    }
23  }
24 }
25 }
```

```
{
  "took": 1,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "failed": 0
  },
  "hits": {
    "total": 2,
    "max_score": 1.0,
    "hits": [ {
      "_index": "my_store",
      "_type": "products",
      "_id": "1",
      "_score": 1.0,
      "_source": {
        "price": 10,
        "productID": "XHDK-A-1293-#fJ3"
      }
    }, {
      "_index": "my_store",
      "_type": "products",
      "_id": "3",
      "_score": 1.0,
      "_source": {
        "price": 30,
        "productID": "JODL-X-1937-#pV7"
      }
    }
  ]
}
```



Filetered: Equals Exactly

Filtered : Equals Exactly

다큐먼트에 들어있는 값을 bool 내의 must에 term을 연속적으로 표시해서 AND 처리를 통해 정확성 값을 검색

```
POST /my_index/my_type/_search
{
  "query": {
    "filtered": {
      "filter": {
        "bool": {
          "must": [
            { "term": { "tags": "search" } },
            { "term": { "tag_count": 1 } }
          ]
        }
      }
    }
  }
}
```

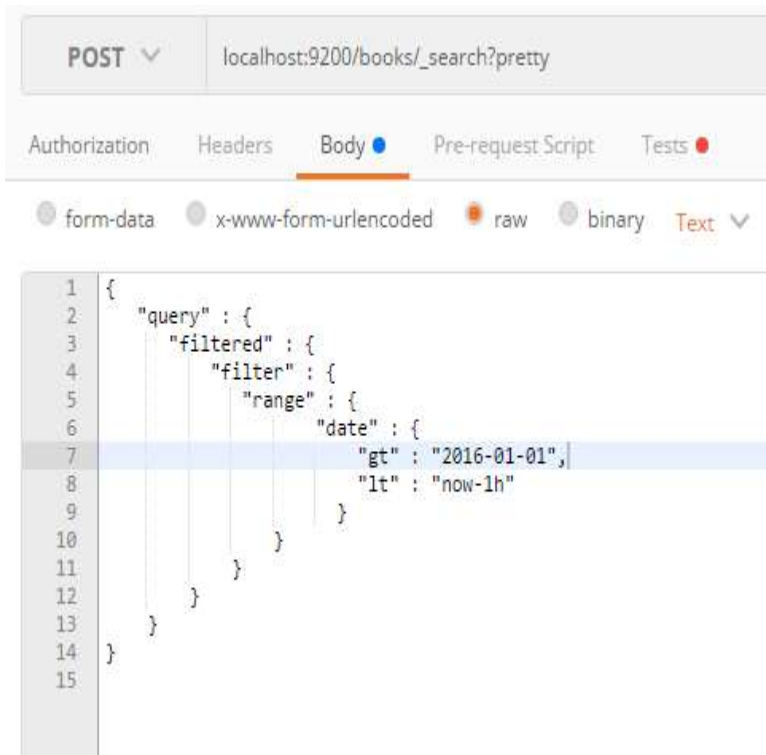
tags & tag_count가
만족하는 것을 처리함



Filetered: date range

Filtered : date range

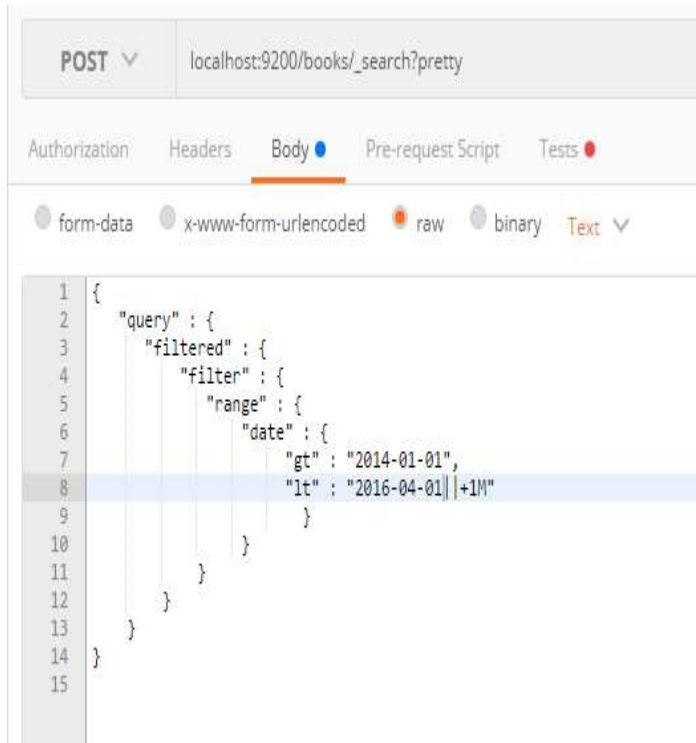
now - 1h 보다 현재보다 한시간 전을 계산하고
이것보다 작은 date 필드 처리



```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "5",
      "_score" : 1.0,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : " big picture",
        "plot" : "elasticsearch"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "4",
      "_score" : 1.0,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : " big picture",
        "plot" : "elasticsearch"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "3",
      "_score" : 1.0,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : " big picture",
        "plot" : "elasticsearch"
      }
    }
  ]
}
```

Filtered : date operator

|| + 1M 은 한달을 가산해서 처리



```
{
  "took" : 9,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "10",
      "_score" : 1.0,
      "_source" : {
        "title" : "Elasticsearch Guide",
        "author" : [ "lee" ],
        "date" : "2014-05-01",
        "pages" : 300,
        "category" : "ICT"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "1",
      "_score" : 1.0,
      "_source" : {
        "title" : "Elasticsearch Guide",
        "author" : [ "lee" ],
        "date" : "2014-05-01",
        "pages" : 5000,
        "category" : "ICT"
      }
    }
  ]
}
```



Filetered: string range

Filtered : string range

필드의 값에 대한 알파벳 순서에 따라 처리

```
POST localhost:9200/books/_search?pretty
Authorization Headers Body Pre-request Script Tests
form-data x-www-form-urlencoded raw binary Te

1 {
2   "query": {
3     "filtered": {
4       "filter": {
5         "range": {
6           "title": {
7             "gte": "a",
8             "lt": "c"
9           }
10        }
11      }
12    }
13  }
14 }
15 }
```

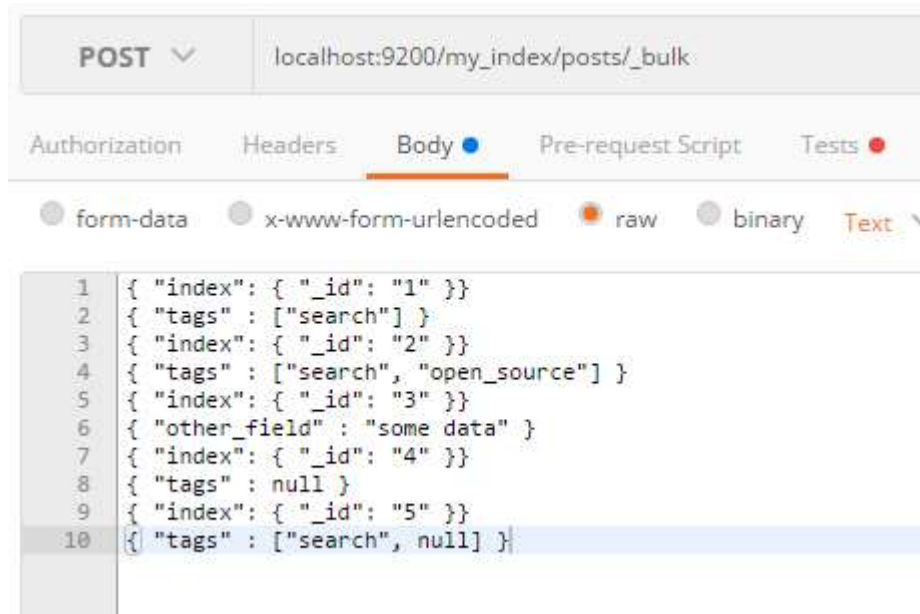
```
{
  "took" : 8,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 4,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "5",
      "_score" : 1.0,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : " big picture",
        "plot" : "elasticsearch"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "4",
      "_score" : 1.0,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : " big picture",
        "plot" : "elasticsearch"
      }
    }
  ]
}
```



Filetered:existing

Filtered : 새로운 다큐먼트 생성

null 값을 가진 다큐먼트 생성

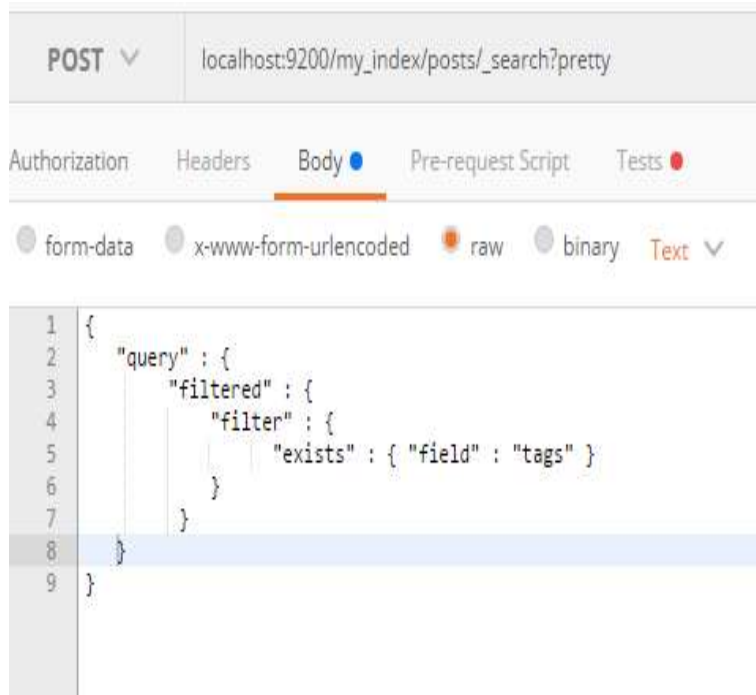


The screenshot shows a REST client interface with a POST request to `localhost:9200/my_index/posts/_bulk`. The 'Body' tab is selected, and the 'Text' format is chosen. The request body contains 10 document snippets, each with an index and tags. The 10th snippet has a null tag.

```
1 { "index": { "_id": "1" }}
2 { "tags" : ["search"] }
3 { "index": { "_id": "2" }}
4 { "tags" : ["search", "open_source"] }
5 { "index": { "_id": "3" }}
6 { "other_field" : "some data" }
7 { "index": { "_id": "4" }}
8 { "tags" : null }
9 { "index": { "_id": "5" }}
10 { "tags" : ["search", null] }
```

Filtered :exist

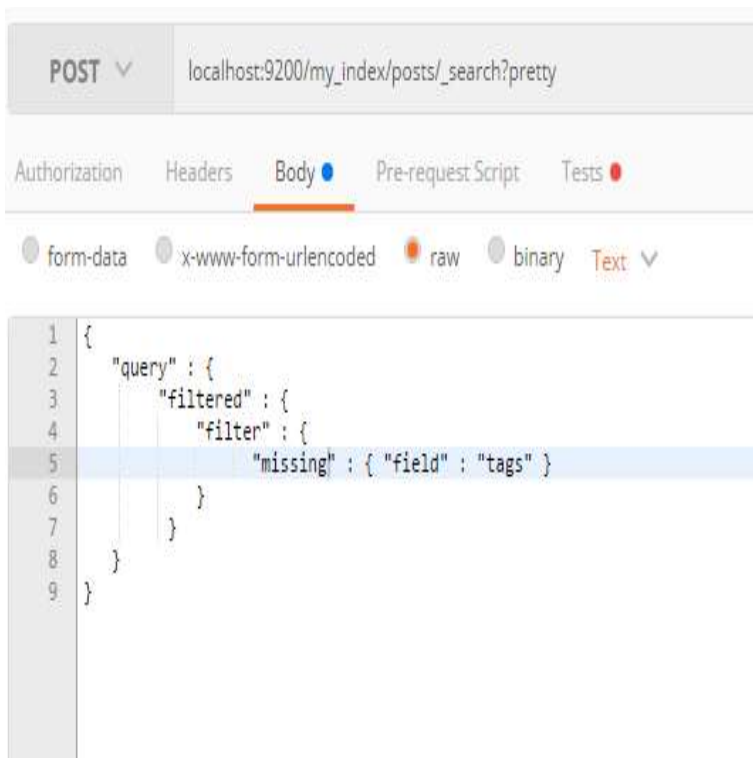
field 존재 여부를 조회



```
{
  "took" : 9,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "my_index",
      "_type" : "posts",
      "_id" : "2",
      "_score" : 1.0,
      "_source" : {
        "tags" : [ "search", "open_source" ]
      }
    }, {
      "_index" : "my_index",
      "_type" : "posts",
      "_id" : "1",
      "_score" : 1.0,
      "_source" : {
        "tags" : [ "search" ]
      }
    } ]
  }
}
```

Filtered : missing

field 가 없거나 null 값 여부를 조회



```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 2,
    "max_score" : 1.0,
    "hits" : [ {
      "_index" : "my_index",
      "_type" : "posts",
      "_id" : "4",
      "_score" : 1.0,
      "_source" : {
        "tags" : null
      }
    }, {
      "_index" : "my_index",
      "_type" : "posts",
      "_id" : "3",
      "_score" : 1.0,
      "_source" : {
        "other_field" : "some data"
      }
    }
  ]
}
```

Filtered : object field

object 필드는 object명.속성명으로 정의하고
검색

```
{
  "bool": {
    "should": [
      { "exists": { "field": { "name.first" }}}},
      { "exists": { "field": { "name.last" }}}
    ]
  }
}
```

QUERY DSL : _VALIDATE



validate

_validate API

_validate API 를 이용해서 query 점검
localhost:9200/books/_validate/query



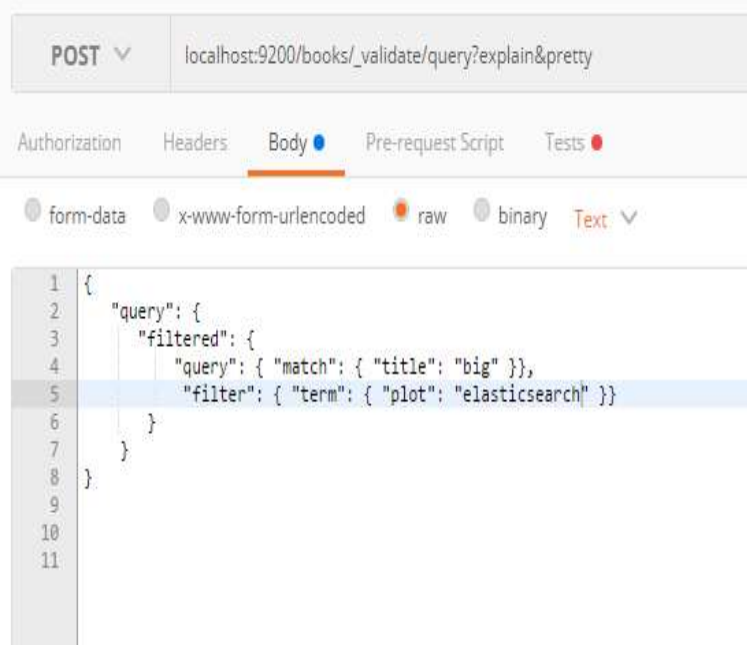
```
1 {
2   "query": {
3     "filtered": {
4       "query": { "match": { "title": "big" } },
5       "filter": { "term": { "plot": "elasticsearch" } }
6     }
7   }
8 }
9
10
11
```



```
Pretty Raw Preview
{"valid":true,"_shards":{"total":1,"successful":1,"failed":0}}
```

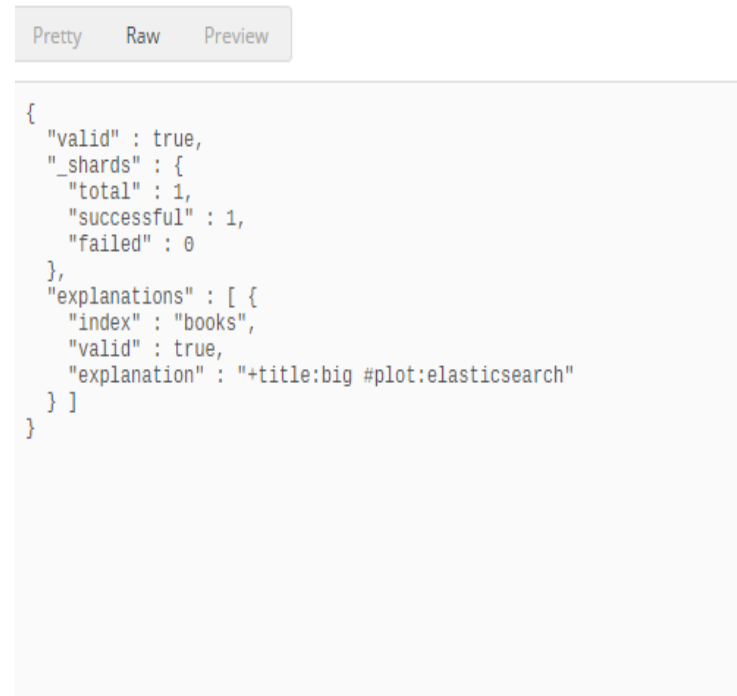
_validate API : explain

_validate API 를 이용해서 query 시 explain을 사용하면 점검한 항목에 대해서도 명확히 표시
localhost:9200/books/_validate/query?explain&pretty



```
POST localhost:9200/books/_validate/query?explain&pretty

{
  "query": {
    "filtered": {
      "query": { "match": { "title": "big" } },
      "filter": { "term": { "plot": "elasticsearch" } }
    }
  }
}
```

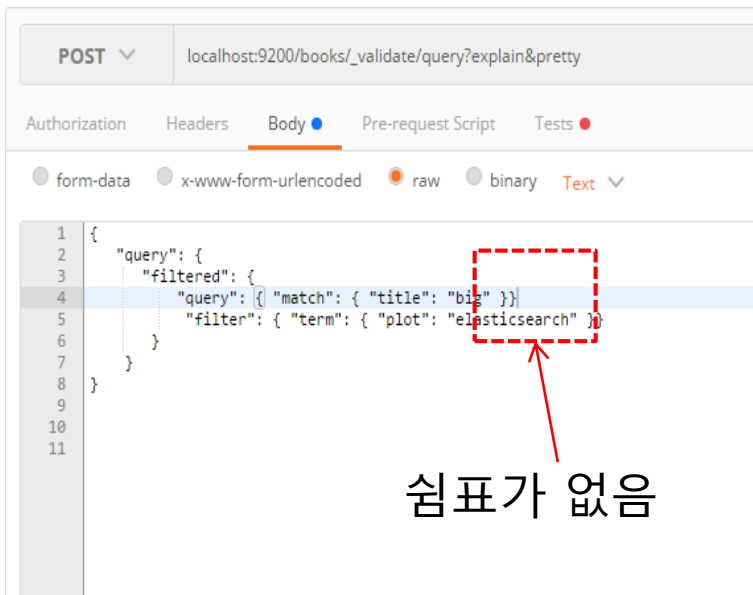


```
Pretty Raw Preview

{
  "valid" : true,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "explanations" : [ {
    "index" : "books",
    "valid" : true,
    "explanation" : "+title:big #plot:elasticsearch"
  } ]
}
```

_validate API : error 발생

_validate API 를 이용할 경우 error 발생시 명확히 explain 부분에 표시됨



SORT

Moon Yong Joon

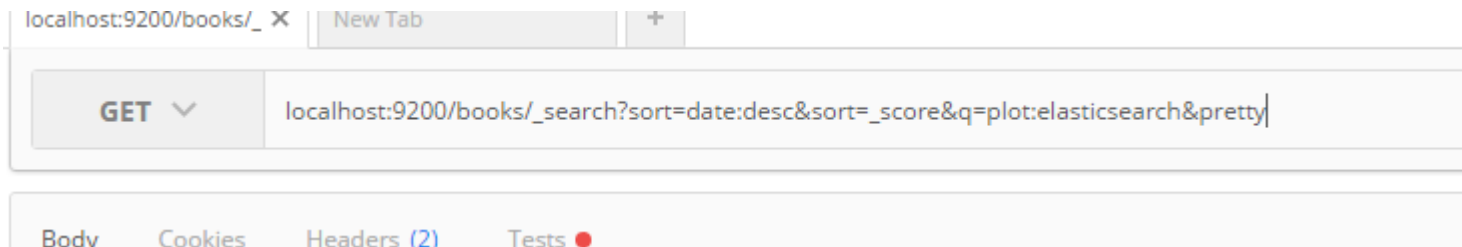


URI sort

sort

URI에 sort 다음에 sorting 할 속성과 order를 정한 후 스트링쿼리를 작성

localhost:9200/books/_search?sort=date:desc&sort=_score&q=plot:elasticsearch&pretty



Sort 결과

URI에 sort 사용한 결과

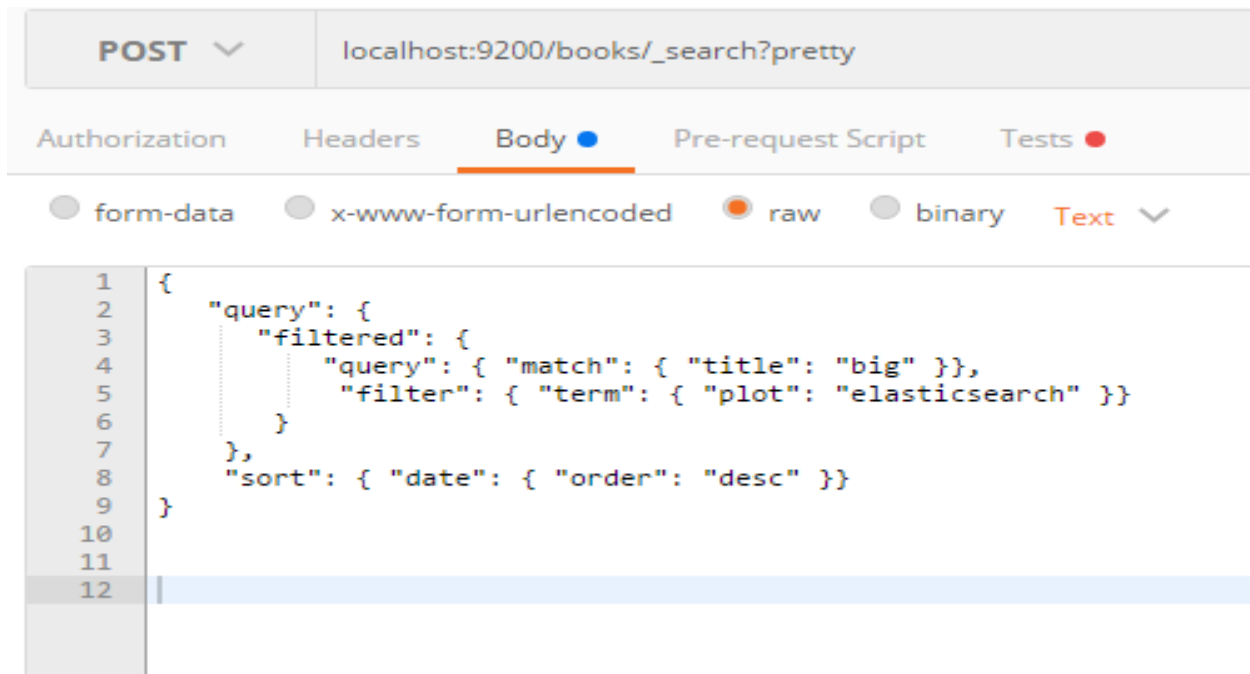
```
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 4,
    "max_score" : 1.4054651,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "5",
      "_score" : 1.4054651,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : "big picture",
        "plot" : "elasticsearch"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "3",
      "_score" : 1.0,
      "_source" : {
        "plot" : "elasticsearch"
      }
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "4"
```



QueryDSL sort

sort

Sort 내의 필드명과 순서(desc/asc)에 대해 정의



```
1 {
2   "query": {
3     "filtered": {
4       "query": { "match": { "title": "big" } },
5       "filter": { "term": { "plot": "elasticsearch" } }
6     }
7   },
8   "sort": { "date": { "order": "desc" } }
9 }
10
11
12
```

Sort 결과

Sort 는 `_score` 대신 `sort` 속성 내의 값을 기준으로 분류

```
{
  "took" : 69,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : null,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "5",
      "_score" : null,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : "big picture",
        "plot" : "elasticsearch"
      },
      "sort" : [ 1465516800000 ]
    },
    {
      "_index" : "books",
      "_type" : "book",
      "_id" : "4",
      "_score" : null,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : "big picture"
      }
    }
  ]
}
```

`_score`를 사용하지 않음

Sort 속성에 `date` 값이 `mili` 초 단위로 계산

Sort : multi field

Sort 내의 멀티 필드명과 순서(desc/asc)에 대해 정의: `_score`로 sort하므로 점수가 산출됨

```
1 {
2   "query": {
3     "filtered": {
4       "query": { "match": { "title": "big" } },
5       "filter": { "term": { "plot": "elasticsearch" } }
6     }
7   },
8   "sort": [ { "date": { "order": "asc" } },
9             { "_score": { "order": "desc" } }
10  ]
11 }
12
13
14
```

Sort : multi field 결과

Sort 내의 멀티 필드명과 순서(desc/asc)에 대해 정의

```
{
  "took" : 9,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "failed" : 0
  },
  "hits" : {
    "total" : 3,
    "max_score" : null,
    "hits" : [ {
      "_index" : "books",
      "_type" : "book",
      "_id" : "5",
      "_score" : 0.8784157,
      "_source" : {
        "author" : "dahl",
        "category" : "novel",
        "date" : "2016-06-10",
        "pages" : 500,
        "title" : "big picture",
        "plot" : "elasticsearch"
      },
      "sort" : [ 1465516800000, 0.8784157 ]
    }, {
      "_index" : "books",
      "_type" : "book",
      "_id" : "4",
      "_score" : 0.37158427,
```

_score를 사용하
므로 점수 산출

date, _score 2개 기
준으로 sorting