1. Linux安装

1.1. 下载&安装

1.1.1. 环境需求

- CentOs7
- 内存4G+

1.1.2. 下载

官方elasticsearch下载，下载elasticsearch，目前最新的稳定版本为 7.4.0 版本.

1.1.3. 安装

```
[root@localhost download]$ pwd
/data/download/

[root@localhost download]$ wget
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-7.4.0-linux-
x86_64.tar.gz

[root@localhost download]$ cd ../app/

[root@localhost app]$ mkdir elastic

[root@localhost app]$ useradd elastic -g dev

[root@localhost app]$ passwd elastic

[root@localhost app]$ chown -R elastic:dev elastic

[root@localhost app]$ su elastic

[elastic@localhost app]$ cd /elastic

[elastic@localhost elastic]$ cp ../../download/elasticsearch-7.4.0-linux-
x86_64.tar.gz .

[elastic@localhost elastic]$ tar -zxvf elasticsearch-7.4.0-linux-x86_64.tar.gz

[elastic@localhost elastic]$ mv elasticsearch-7.4.0/ .
```

1.1.4. 修改配置文件

路径config/elasticsearch.yml

```
-- 允许外部IP访问
network.host: 0.0.0.0

-- 把这个注释先放开
cluster.initial_master_nodes: ["node-1", "node-2"]
```

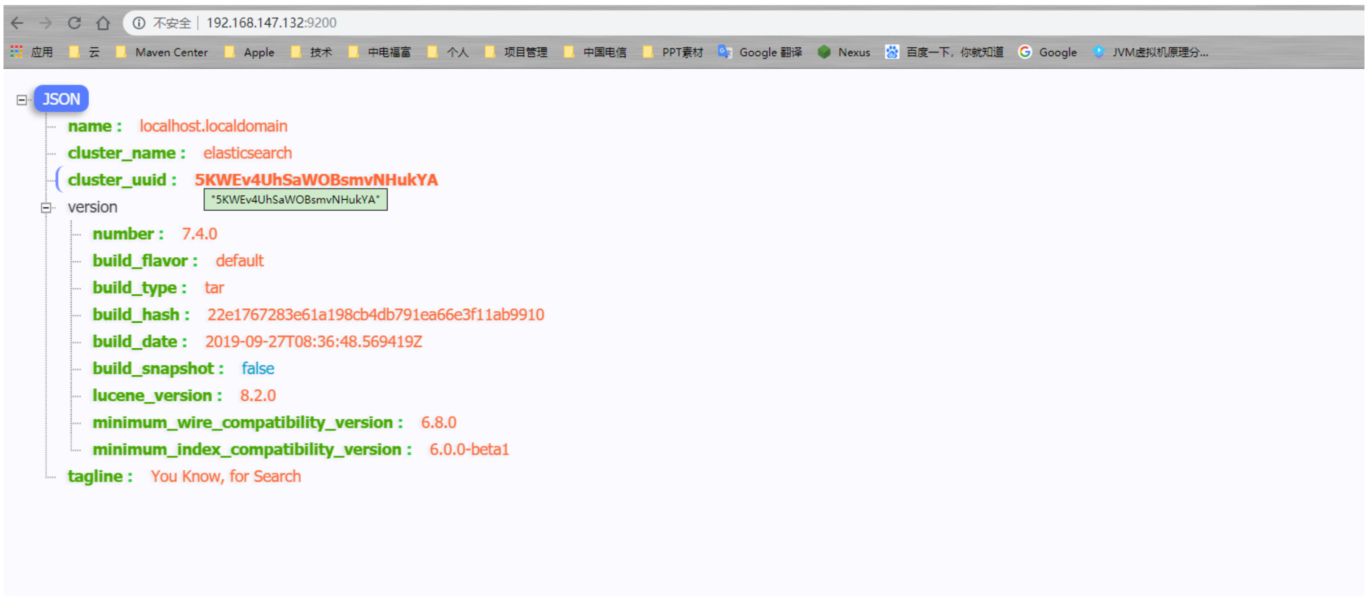
1.1.5. 启动&验证结果

- 启动

```
[elastic@localhost elastic]$ ./bin/elasticsearch
```

- 验证结果

Elastic会在默认9200端口运行，打开地址：http://192.168.147.132:9200/



```
name :   localhost.localdomain
cluster_name :   elasticsearch
cluster_uuid :   5KWEv4UhSaWOBsmvNHukYA
                 "5KWEv4UhSaWOBsmvNHukYA"
version
    number :   7.4.0
    build_flavor :   default
    build_type :   tar
    build_hash :   22e1767283e61a198cb4db791ea66e3f11ab9910
    build_date :   2019-09-27T08:36:48.569419Z
    build_snapshot :   false
    lucene_version :   8.2.0
    minimum_wire_compatibility_version :   6.8.0
    minimum_index_compatibility_version :   6.0.0-beta1
tagline :   You Know, for Search
```

1.2. 中文分词插件IK

1.2.1. 安装

ik插件地址： https://github.com/medcl/elasticsearch-analysis-ik，为了演示需要，这里选择wget方式。

- 下载

```
[root@localhost download]$ wget https://github.com/medcl/elasticsearch-analysis-
ik/releases/download/v7.4.0/elasticsearch-analysis-ik-7.4.0.zip
```

- 安装插件

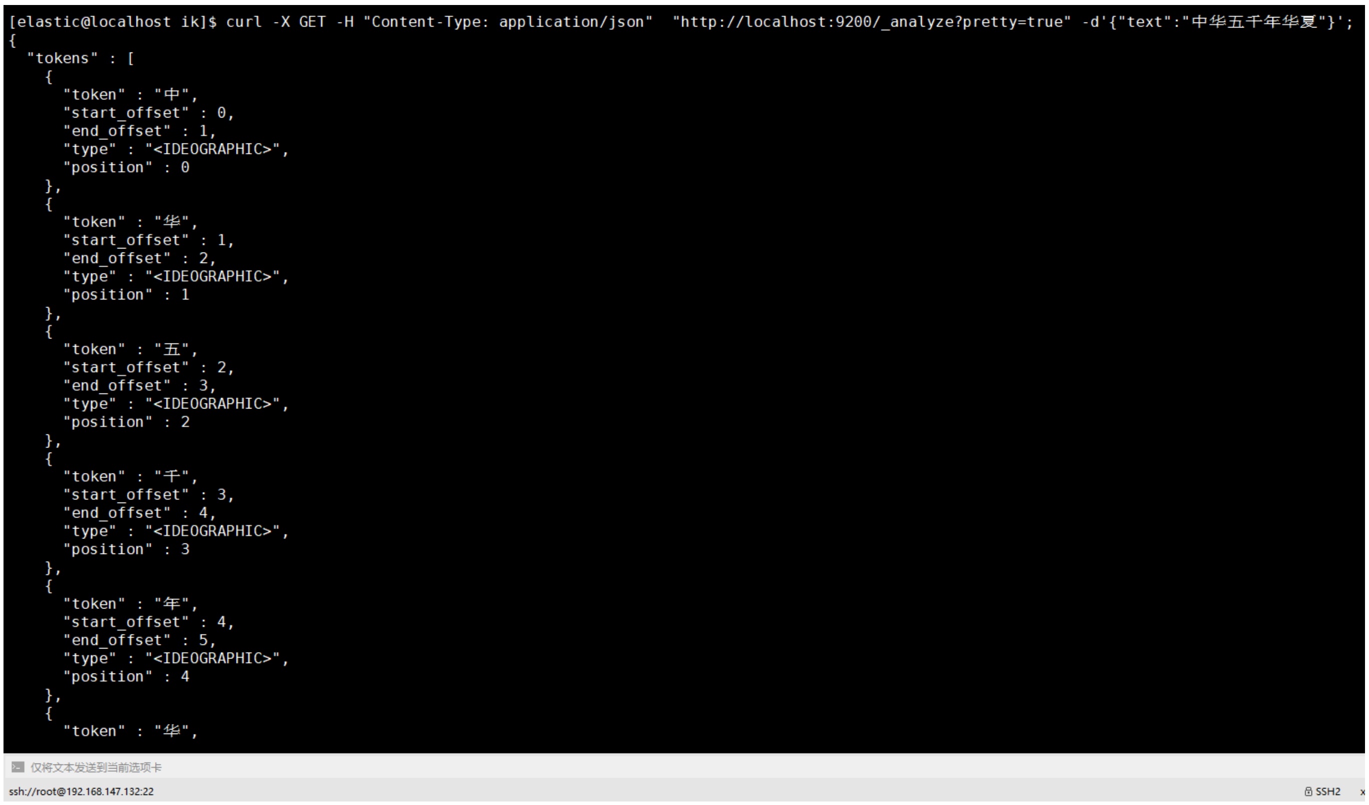
```
[elastic@localhost elastic]$ cd plugins

[elastic@localhost plugins]$ cd mkdir ik && cd ik

[elastic@localhost ik]$ cp ../../../download/elasticsearch-analysis-ik-7.4.0.zip .

[elastic@localhost ik]$ unzip elasticsearch-analysis-ik-7.4.0.zip
```

完成后重启es

- 验证分词器

使用crul命令，输入下面的URL地址，验证分词器是否成功。

```
[elastic@localhost elastic]$ curl -X GET -H "Content-Type: application/json"
"http://localhost:9200/_analyze?pretty=true" -d'{"text":"中华五千年华夏"}';
```



```
[elastic@localhost ik]$ curl -X GET -H "Content-Type: application/json"  "http://localhost:9200/_analyze?pretty=true" -d'{"text":"中华五千年华夏"}';
{
  "tokens" : [
    {
      "token" : "中",
      "start_offset" : 0,
      "end_offset" : 1,
      "type" : "<IDEOGRAPHIC>",
      "position" : 0
    },
    {
      "token" : "华",
      "start_offset" : 1,
      "end_offset" : 2,
      "type" : "<IDEOGRAPHIC>",
      "position" : 1
    },
    {
      "token" : "五",
      "start_offset" : 2,
      "end_offset" : 3,
      "type" : "<IDEOGRAPHIC>",
      "position" : 2
    },
    {
      "token" : "千",
      "start_offset" : 3,
      "end_offset" : 4,
      "type" : "<IDEOGRAPHIC>",
      "position" : 3
    },
    {
      "token" : "年",
      "start_offset" : 4,
      "end_offset" : 5,
      "type" : "<IDEOGRAPHIC>",
      "position" : 4
    },
    {
      "token" : "华",
```

1.2.2. ik_max_word和ik_smart

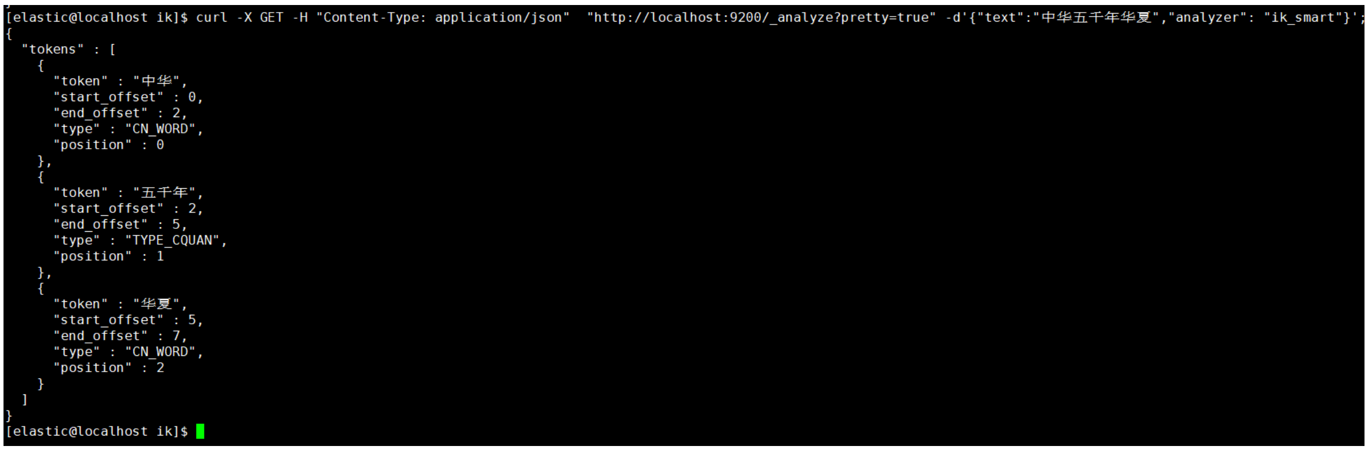
- **ik_max_word**: 将文本按最细粒度的组合来拆分，比如会将“中华五千年华夏”拆分为“五千年、五千、五千年华、华夏、千年华夏”，总之是可能的组合；
- **ik_smart**: 最粗粒度的拆分，比如会将“五千年华夏”拆分为“五千年、华夏”

不添加分词类别，Elastic对于汉字默认使用standard只是将汉字拆分成一个个的汉字，而我们ik则更加的智能，下面通过几个案例来说明。

1.2.2.1. ik_smart分词

在JSON格式中添加**analyzer**节点内容为**ik_smart**

```
[elastic@localhost elastic]$ curl -X GET -H "Content-Type: application/json"
"http://localhost:9200/_analyze?pretty=true" -d'{"text":"中华五千年华
夏","analyzer": "ik_smart"}';
```

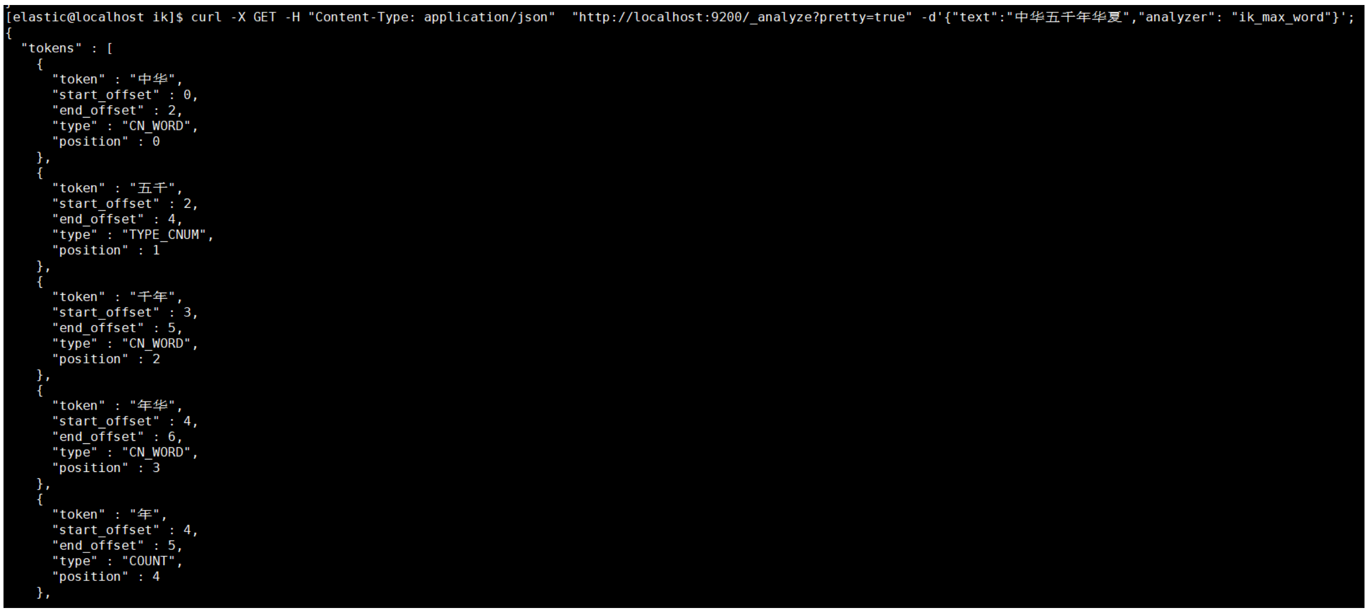


```
[elastic@localhost ik]$ curl -X GET -H "Content-Type: application/json"  "http://localhost:9200/_analyze?pretty=true" -d'{"text":"中华五千年华夏","analyzer": "ik_smart"}';
{
  "tokens" : [
    {
      "token" : "中华",
      "start_offset" : 0,
      "end_offset" : 2,
      "type" : "CN_WORD",
      "position" : 0
    },
    {
      "token" : "五千年",
      "start_offset" : 2,
      "end_offset" : 5,
      "type" : "TYPE_CQUAN",
      "position" : 1
    },
    {
      "token" : "华夏",
      "start_offset" : 5,
      "end_offset" : 7,
      "type" : "CN_WORD",
      "position" : 2
    }
  ]
}
[elastic@localhost ik]$
```

1.2.2.2. ik_max_word分词

在JSON格式中添加**analyzer**节点内容为**ik_max_word**

```
[elastic@localhost elastic]$ curl -X GET -H "Content-Type: application/json"
"http://localhost:9200/_analyze?pretty=true" -d'{"text":"中华五千年华
夏","analyzer": "ik_max_word"}';
```



```
[elastic@localhost ik]$ curl -X GET -H "Content-Type: application/json"  "http://localhost:9200/_analyze?pretty=true" -d'{"text":"中华五千年华夏","analyzer": "ik_max_word"}';
{
  "tokens" : [
    {
      "token" : "中华",
      "start_offset" : 0,
      "end_offset" : 2,
      "type" : "CN_WORD",
      "position" : 0
    },
    {
      "token" : "五千",
      "start_offset" : 2,
      "end_offset" : 4,
      "type" : "TYPE_CNUM",
      "position" : 1
    },
    {
      "token" : "千年",
      "start_offset" : 3,
      "end_offset" : 5,
      "type" : "CN_WORD",
      "position" : 2
    },
    {
      "token" : "年华",
      "start_offset" : 4,
      "end_offset" : 6,
      "type" : "CN_WORD",
      "position" : 3
    },
    {
      "token" : "年",
      "start_offset" : 4,
      "end_offset" : 5,
      "type" : "COUNT",
      "position" : 4
    },
```

1.3. 索引

1.3.1. 创建索引

由于在ElasticSearch 7.x之后就默认不在支持指定索引类型，所以在在elasticsearch7.x上执行：

```
{
    "settings" : {
        "index" : {
            "number_of_shards" : 3,
            "number_of_replicas" : 2
        }
    },
    "mappings" : {
        "twitter":{
            ......
        }
    }
```

执行结果则会出错：Root mapping definition has unsupported parameters（刚开始接触就踩了这个坑，折煞劳资好久）。如果在6.x上执行，则会正常执行。 出现这个的原因是，elasticsearch7默认不在支持指定索引类型，默认索引类型是_doc，如果想改变，则配置include_type_name: true 即可(这个没有测试，官方文档说的，无论是否可行，建议不要这么做，因为elasticsearch8后就不在提供该字段)。

https://www.elastic.co/guide/en/elasticsearch/reference/current/removal-of-types.html

1.3.1.1. 官方例子说明

```
curl -X PUT "localhost:9200/twitter" -H 'Content-Type: application/json' -d'
{
    "settings" : {
        "index" : {
            "number_of_shards" : 3,
            "number_of_replicas" : 2
        }
    }
}
'
```

- -d指定了你的参数，这里将这些参数放到了json文件中
- settings设置内容含义

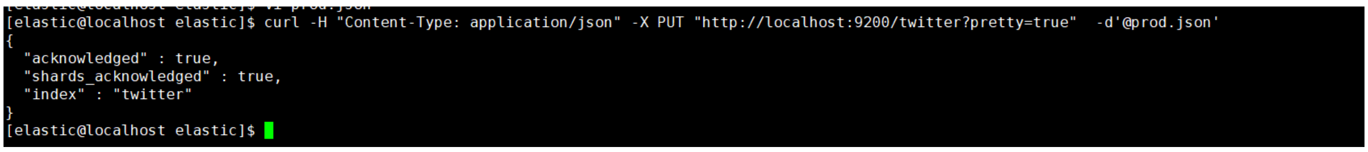
name	价格
number_of_shards	分片数
number_of_replicas	副本数
mappings	结构化数据设置 下面的一级属性 是自定义的类型
properties	类型的属性设置节点，下面都是属性

name	价格
epoch_millis	表示时间戳

1.3.1.2. 自定义索引

- 使用json文件创建索引 使用 -d‘@your jsonFile’指定你的json文件。下边我创建了一个索引名称为product（可自己定义）的索引。

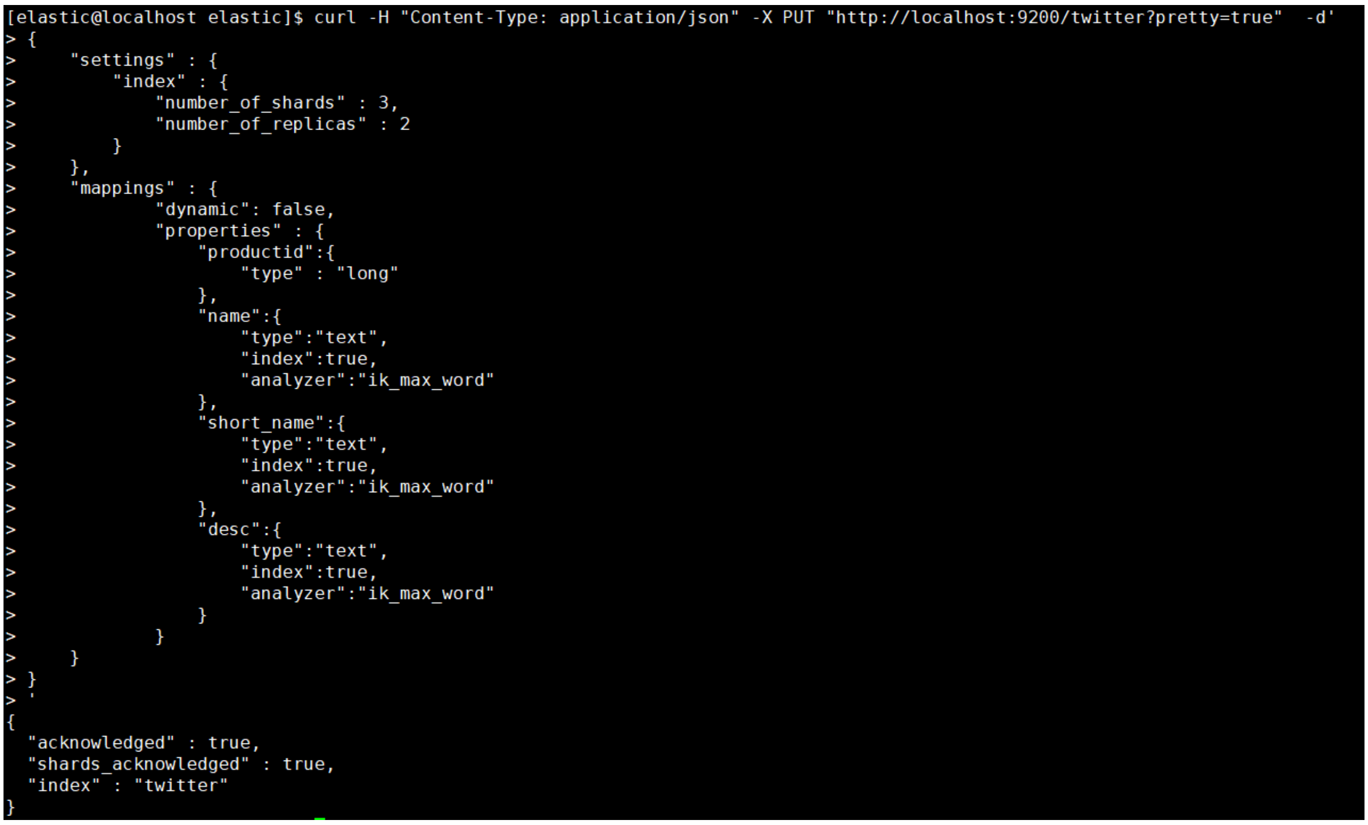
```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/twitter?pretty=true"  -d'@prod.json'
```



```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT "http://localhost:9200/twitter?pretty=true"  -d'@prod.json'
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "twitter"
}
[elastic@localhost elastic]$ 
```

- 参数形式创建索引

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/twitter?pretty=true"  -d'
{
    "settings" : {
        "index" : {
            "number_of_shards" : 3,
            "number_of_replicas" : 2
        }
    },
    "mappings" : {
            "dynamic": false,
            "properties" : {
                "productid":{
                    "type" : "long"
                },
                "name":{
                    "type":"text",
                    "index":true,
                    "analyzer":"ik_max_word"
                },
                "short_name":{
                    "type":"text",
                    "index":true,
                    "analyzer":"ik_max_word"
                },
                "desc":{
                    "type":"text",
                    "index":true,
                    "analyzer":"ik_max_word"
                }
            }
        }
    }
    '
```

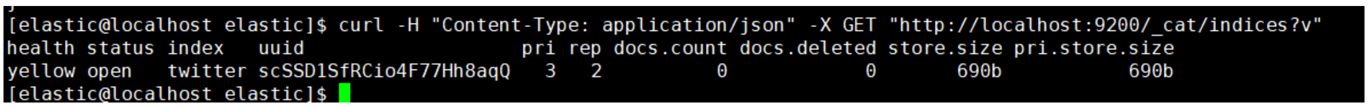


```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT "http://localhost:9200/twitter?pretty=true"  -d'
> {
>     "settings" : {
>         "index" : {
>             "number_of_shards" : 3,
>             "number_of_replicas" : 2
>         }
>     },
>     "mappings" : {
>             "dynamic": false,
>             "properties" : {
>                 "productid":{
>                     "type" : "long"
>                 },
>                 "name":{
>                     "type":"text",
>                     "index":true,
>                     "analyzer":"ik_max_word"
>                 },
>                 "short_name":{
>                     "type":"text",
>                     "index":true,
>                     "analyzer":"ik_max_word"
>                 },
>                 "desc":{
>                     "type":"text",
>                     "index":true,
>                     "analyzer":"ik_max_word"
>                 }
>             }
>     }
> }
> '
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "twitter"
}
```

1.3.2. 查看索引

1.3.2.1. 全部索引

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/_cat/indices?v"
health status index   uuid                   pri rep docs.count docs.deleted
store.size pri.store.size
yellow open   twitter scSSD1SfRCio4F77Hh8aqQ   3   2          0            0
690b           690b
```



```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET "http://localhost:9200/_cat/indices?v"
health status index   uuid                   pri rep docs.count docs.deleted store.size pri.store.size
yellow open   twitter scSSD1SfRCio4F77Hh8aqQ   3   2          0            0       690b           690b
[elastic@localhost elastic]$
```

1.3.2.2. 条件查询

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/twitter?pretty=true"
{
  "twitter" : {
    "aliases" : { },
    "mappings" : {
      "dynamic" : "false",
      "properties" : {
        "desc" : {
          "type" : "text",
          "analyzer" : "ik_max_word"
        },
        "name" : {
          "type" : "text",
          "analyzer" : "ik_max_word"
        },
        "productid" : {
          "type" : "long"
        },
        "short_name" : {
          "type" : "text",
          "analyzer" : "ik_max_word"
        }
      }
    },
    "settings" : {
      "index" : {
        "creation_date" : "1571153735610",
        "number_of_shards" : "3",
        "number_of_replicas" : "2",
        "uuid" : "scSSD1SfRCio4F77Hh8aqQ",
        "version" : {
          "created" : "7040099"
        },
        "provided_name" : "twitter"
      }
    }
  }
}
```

1.3.3. 查看索引分词器

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/twitter/_analyze?pretty=true" -d'
{
  "field": "text",
  "text": "秦皇汉武."
}
'
```

elastic

1.3.4. 修改索引

1.3.5. 删除索引

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X DELETE
"http://localhost:9200/twitter?pretty=true"
```

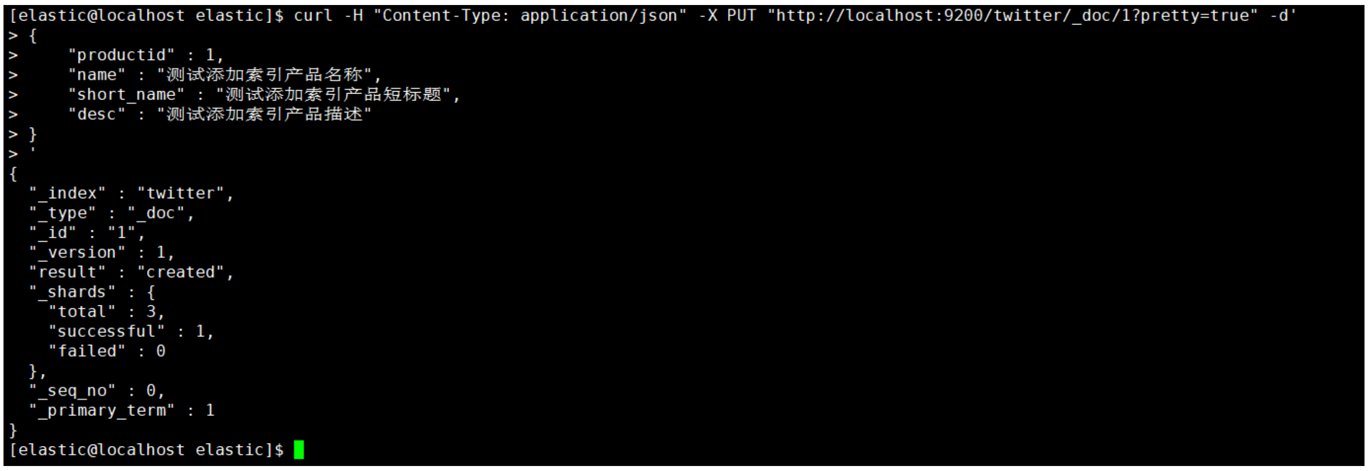
1.4. 如何数据管理

1.4.1. 添加数据

- 这里演示PUT方式为twitter索引添加数据，并且指定id，应当注意此处的默认类型为_doc，还有一种就是采用POST方式添加数据，并且自动生成主键，本文就不再演示，请自行查阅相关材料。

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/twitter/_doc/1?pretty=true" -d'
{
    "productid" : 1,
    "name" : "测试添加索引产品名称",
    "short_name" : "测试添加索引产品短标题",
    "desc" : "测试添加索引产品描述"
}
'
```

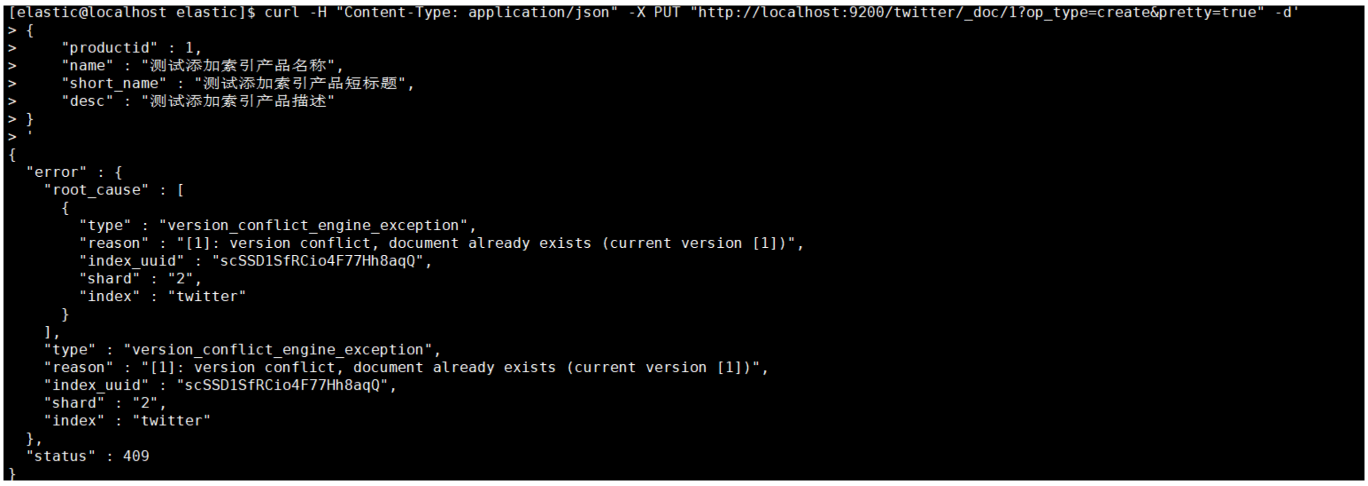
执行返回结果如图，则添加数据成功。



```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT "http://localhost:9200/twitter/_doc/1?pretty=true" -d'
> {
>     "productid" : 1,
>     "name" : "测试添加索引产品名称",
>     "short_name" : "测试添加索引产品短标题",
>     "desc" : "测试添加索引产品描述"
> }
> '
{
  "_index" : "twitter",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 3,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
[elastic@localhost elastic]$
```

- 指定id为1，还可以加上参数op_type=create，这样在创建重复id时会报错导致创建失败，否则会更新该id的属性值。

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/twitter/_doc/1?op_type=create&pretty=true" -d'
{
    "productid" : 1,
    "name" : "测试添加索引产品名称",
    "short_name" : "测试添加索引产品短标题",
    "desc" : "测试添加索引产品描述"
}
'
```

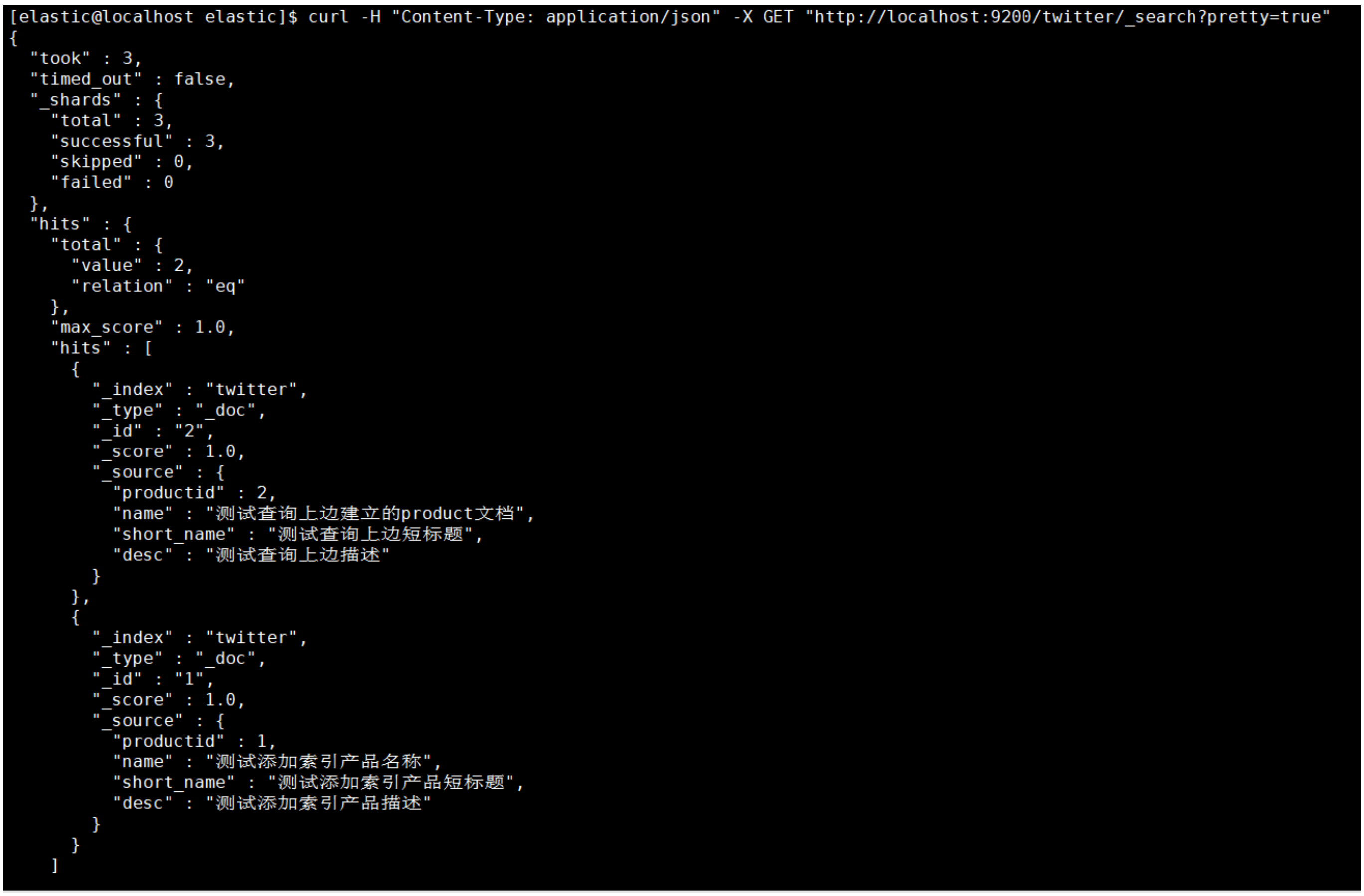


```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT "http://localhost:9200/twitter/_doc/1?op_type=create&pretty=true" -d'
> {
>     "productid" : 1,
>     "name" : "测试添加索引产品名称",
>     "short_name" : "测试添加索引产品短标题",
>     "desc" : "测试添加索引产品描述"
> }
> '
{
  "error" : {
    "root_cause" : [
      {
        "type" : "version_conflict_engine_exception",
        "reason" : "[1]: version conflict, document already exists (current version [1])",
        "index_uuid" : "scSSD1SfRCio4F77Hh8aqQ",
        "shard" : "2",
        "index" : "twitter"
      }
    ],
    "type" : "version_conflict_engine_exception",
    "reason" : "[1]: version conflict, document already exists (current version [1])",
    "index_uuid" : "scSSD1SfRCio4F77Hh8aqQ",
    "shard" : "2",
    "index" : "twitter"
  },
  "status" : 409
}
```

1.4.2. 基础查询

1.4.2.1. 查询所有

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/twitter/_search?pretty=true"
```



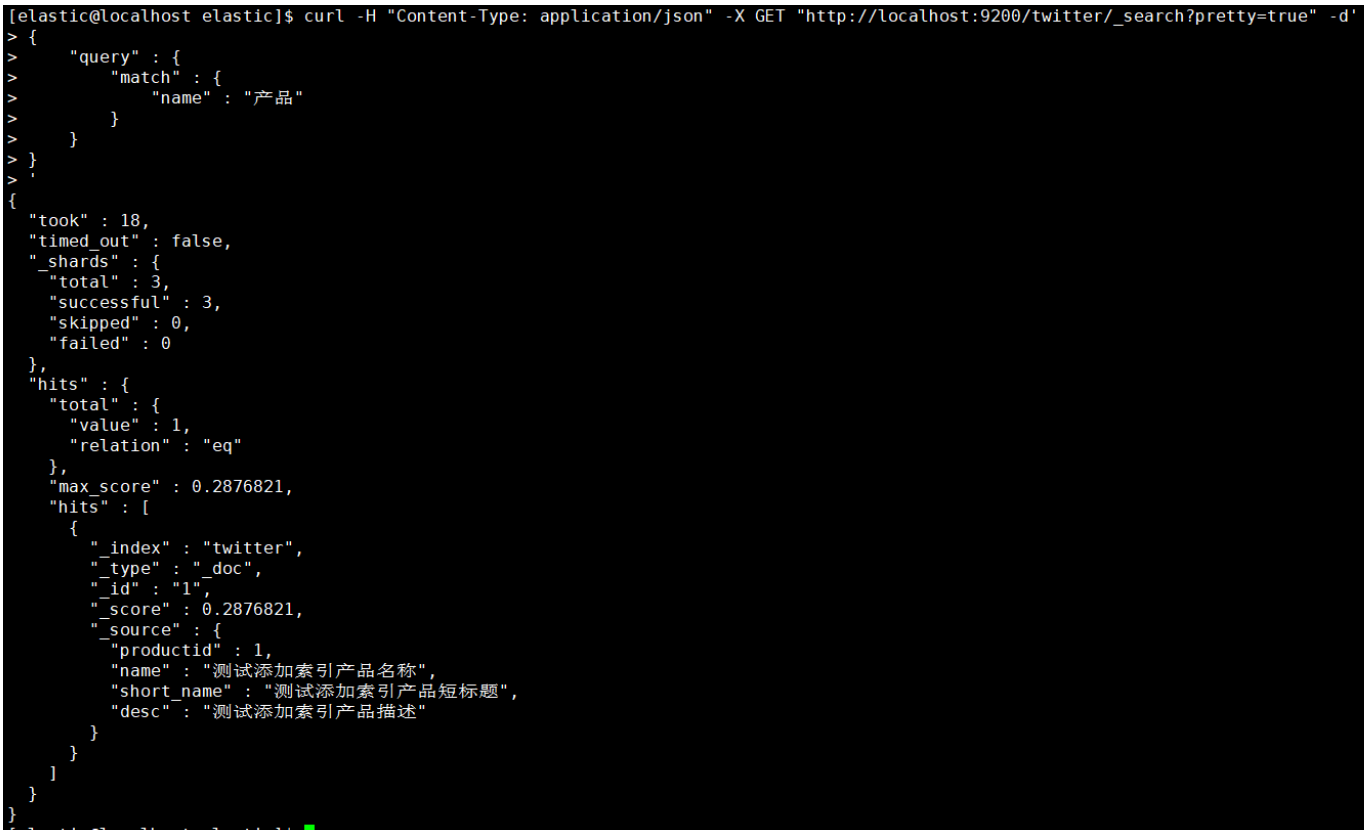
```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET "http://localhost:9200/twitter/_search?pretty=true"
{
  "took" : 3,
  "timed_out" : false,
  "_shards" : {
    "total" : 3,
    "successful" : 3,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 2,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "twitter",
        "_type" : "_doc",
        "_id" : "2",
        "_score" : 1.0,
        "_source" : {
          "productid" : 2,
          "name" : "测试查询上边建立的product文档",
          "short_name" : "测试查询上边短标题",
          "desc" : "测试查询上边描述"
        }
      },
      {
        "_index" : "twitter",
        "_type" : "_doc",
        "_id" : "1",
        "_score" : 1.0,
        "_source" : {
          "productid" : 1,
          "name" : "测试添加索引产品名称",
          "short_name" : "测试添加索引产品短标题",
          "desc" : "测试添加索引产品描述"
        }
      }
    ]
```

1.4.2.2. 条件查询

条件查询会涉及到精确词查询、匹配查询、多条件查询、聚合查询四种，分别为"term"、"match"、"multi_match"、"multi_match"。

- 按找数据的名称作为条件查询匹配

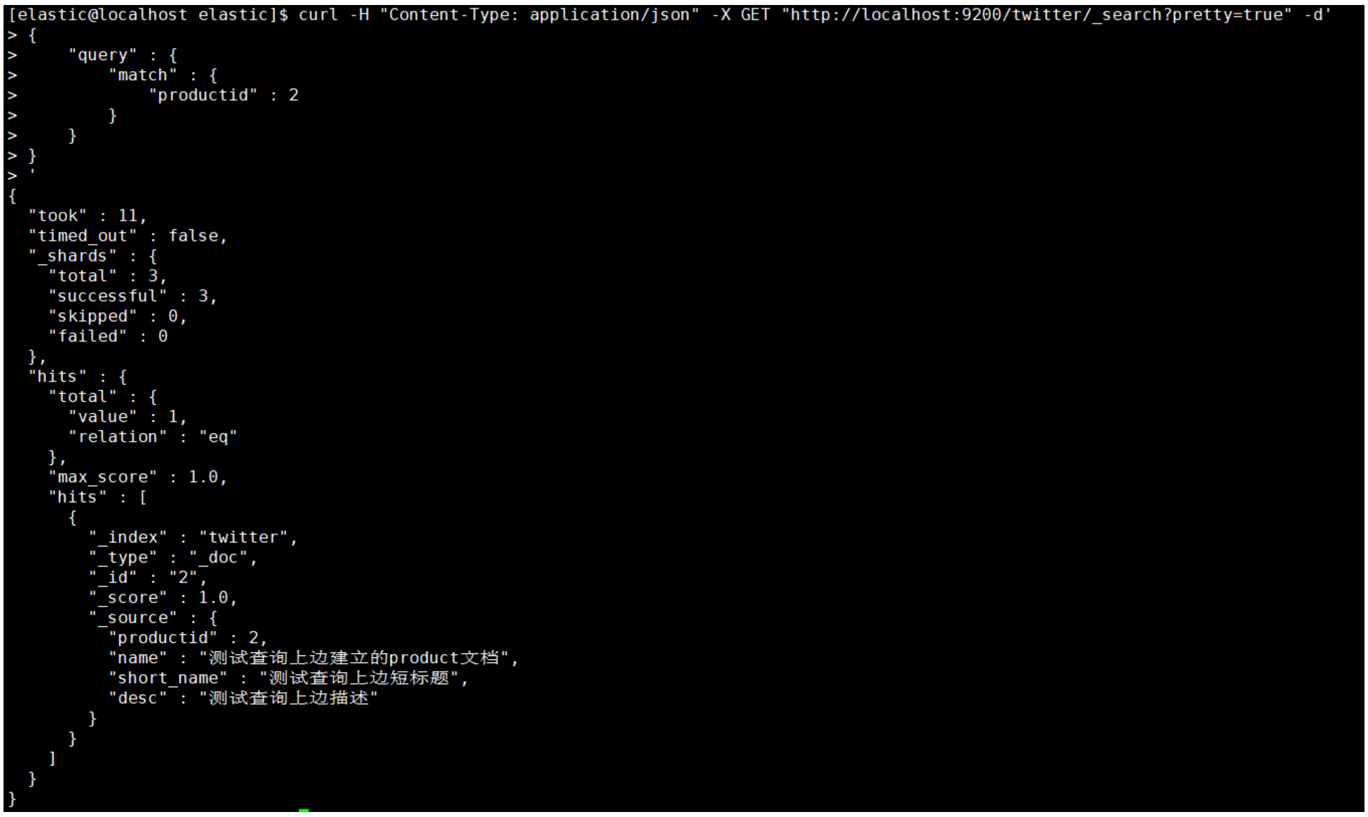
```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/twitter/_search?pretty=true" -d'
{
    "query" : {
        "match" : {
            "name" : "产品"
        }
    }
}
'
```



```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET "http://localhost:9200/twitter/_search?pretty=true" -d'
> {
>     "query" : {
>         "match" : {
>             "name" : "产品"
>         }
>     }
> }
> '
{
  "took" : 18,
  "timed_out" : false,
  "_shards" : {
    "total" : 3,
    "successful" : 3,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 0.2876821,
    "hits" : [
      {
        "_index" : "twitter",
        "_type" : "_doc",
        "_id" : "1",
        "_score" : 0.2876821,
        "_source" : {
          "productid" : 1,
          "name" : "测试添加索引产品名称",
          "short_name" : "测试添加索引产品短标题",
          "desc" : "测试添加索引产品描述"
        }
      }
    ]
  }
}
```

- 按找数据的标识作为条件查询匹配

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/twitter/_search?pretty=true" -d'
{
    "query" : {
        "match" : {
            "productid" : 100
        }
    }
}
'
```

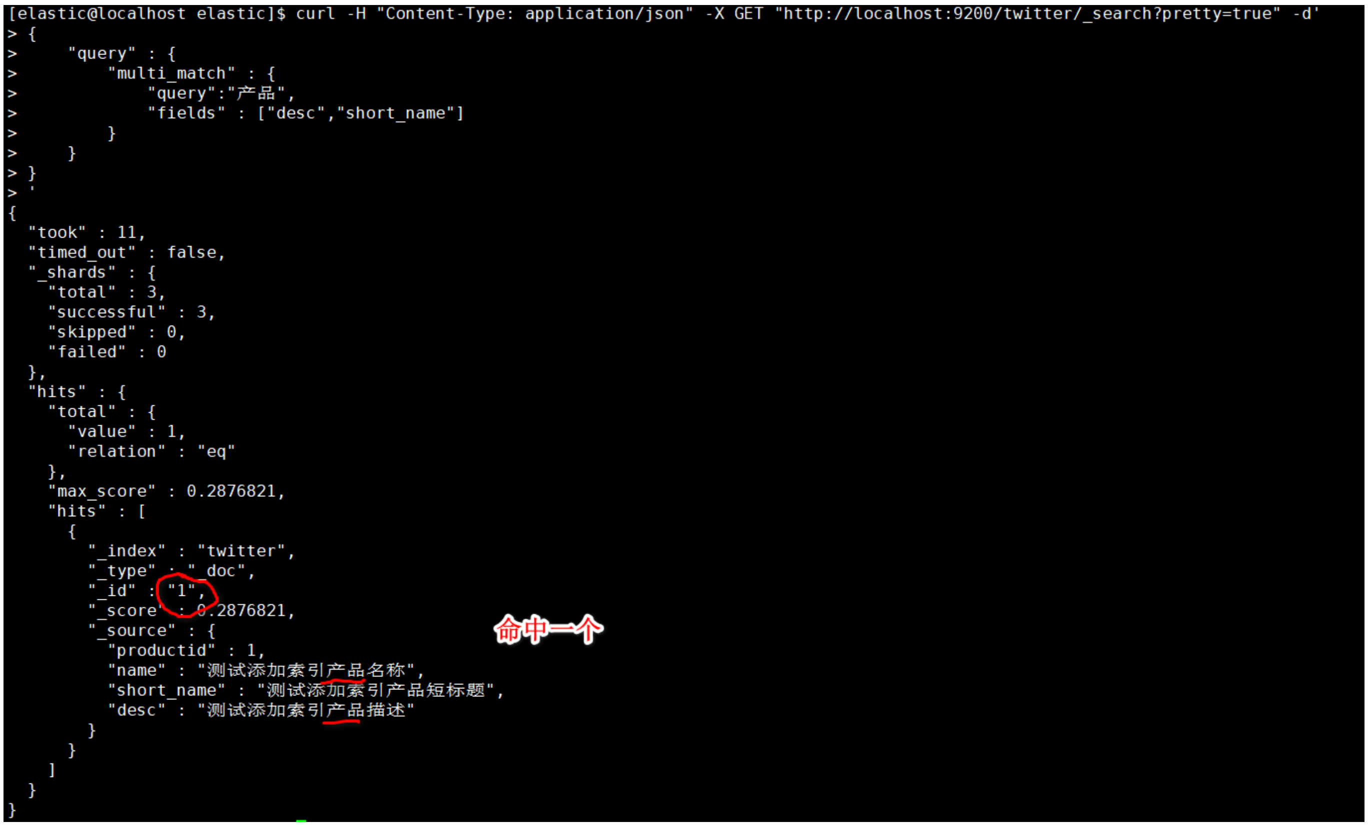


```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET "http://localhost:9200/twitter/_search?pretty=true" -d'
> {
>     "query" : {
>         "match" : {
>             "productid" : 2
>         }
>     }
> }
> '
{
  "took" : 11,
  "timed_out" : false,
  "_shards" : {
    "total" : 3,
    "successful" : 3,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "twitter",
        "_type" : "_doc",
        "_id" : "2",
        "_score" : 1.0,
        "_source" : {
          "productid" : 2,
          "name" : "测试查询上边建立的product文档",
          "short_name" : "测试查询上边短标题",
          "desc" : "测试查询上边描述"
        }
      }
    ]
  }
}
```

- 多条件匹配

选择匹配desc、short_name列作为多条件

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/twitter/_search?pretty=true" -d'
{
    "query" : {
        "multi_match" : {
            "query":"产品",
            "fields" : ["desc","short_name"]
        }
    }
}
'
```



```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET "http://localhost:9200/twitter/_search?pretty=true" -d'
> {
>     "query" : {
>         "multi_match" : {
>             "query":"产品",
>             "fields" : ["desc","short_name"]
>         }
>     }
> }
> '
{
  "took" : 11,
  "timed_out" : false,
  "_shards" : {
    "total" : 3,
    "successful" : 3,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 0.2876821,
    "hits" : [
      {
        "_index" : "twitter",
        "_type" : "_doc",
        "_id" : "1",
        "_score" : 0.2876821,
        "_source" : {
          "productid" : 1,
          "name" : "测试添加索引产品名称",
          "short_name" : "测试添加索引产品短标题",
          "desc" : "测试添加索引产品描述"
        }
      }
    ]
  }
}
```

- 当没有匹配任何数据适合则如下：

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/twitter/_search?pretty=true" -d'
> {
>     "query" : {
>         "match" : {
>             "productid" : 100
>         }
>     }
> }
> '
{
  "took" : 1,
  "timed_out" : false,
  "_shards" : {
    "total" : 3,
    "successful" : 3,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 0,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  }
}
```

1.4.3. 高级条件查询

1.4.3.1. 权重boost查询

指定一个boost值来控制每个查询子句的相对权重，该值默认为1。一个大于1的boost会增加该查询子句的相对权重。 索引映射定义的时候指定boost在elasticsearch5之后已经弃用。建议在查询的时候使用。

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/twitter/_search?pretty=true" -d'
{
    "query": {
        "match" : {
            "title": {
                "query": "quick brown fox",
                "boost": 2
            }
        }
    }
}
'
```

1.4.3.2. 过滤coerce查询

数据不总是我们想要的，由于在转换JSON body为真正JSON 的时候,整型数字5有可能会被写成字符串"5"或者浮点数5.0。coerce属性可以用来清除脏数据。 一般在以下场景中：

- 字符串会被强制转换为整数
- 浮点数被强制转换为整数

1.4.3.2.1. 创建索引

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/wongs?pretty=true"  -d'
{
    "settings" : {
        "index" : {
            "number_of_shards" : 3,
            "number_of_replicas" : 2
        }
    },
    "mappings" : {
            "properties" : {
                "col_1":{
                    "type" : "integer"
                },
                "col_2":{
                    "type":"integer",
                    "coerce": false
                }
            }
    }
}
'
```

1.4.3.2.2. 创建第一个数据

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/wongs/_doc/1?pretty=true" -d'
{
    "col_1" : "20"
}
'
```

结果为成功，说明col_1列数据没问题。

1.4.3.2.3. 创建第二个数据

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/wongs/_doc/1?pretty=true" -d'
> {
>     "col_2" : "20"
> }
> '
{
  "error" : {
    "root_cause" : [
      {
        "type" : "mapper_parsing_exception",
        "reason" : "failed to parse field [col_2] of type [integer] in document
with id '1'. Preview of field's value: '20'"
      }
    ],
    "type" : "mapper_parsing_exception",
    "reason" : "failed to parse field [col_2] of type [integer] in document with
id '1'. Preview of field's value: '20'",
    "caused_by" : {
      "type" : "illegal_argument_exception",
      "reason" : "Integer value passed as String"
    }
  },
  "status" : 400
}
```

由于不能被格式化，数据新增失败。

1.4.3.3. copy_to

copy_to允许你创造自定义超级字段_all. 也就是说，多字段的取值被复制到一个字段并且取值所有字段的取值组合, 并且可以当成一个单独的字段查询. 如，first_name和last_name可以合并为full_name字段。

1.4.3.3.1. 定义索引

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/idx_copy_to?pretty=true"  -d'
{
    "settings" : {
        "index" : {
            "number_of_shards" : 3,
            "number_of_replicas" : 2
        }
    },
    "mappings" : {
            "properties" : {
                "first_name":{
                    "type" : "text",
                    "copy_to": "full_name"
                },
                "last_name":{
                    "type":"text",
                    "copy_to": "full_name"
                },
                "full_name":{
                    "type": "text"
                }
            }
    }
}
'
```

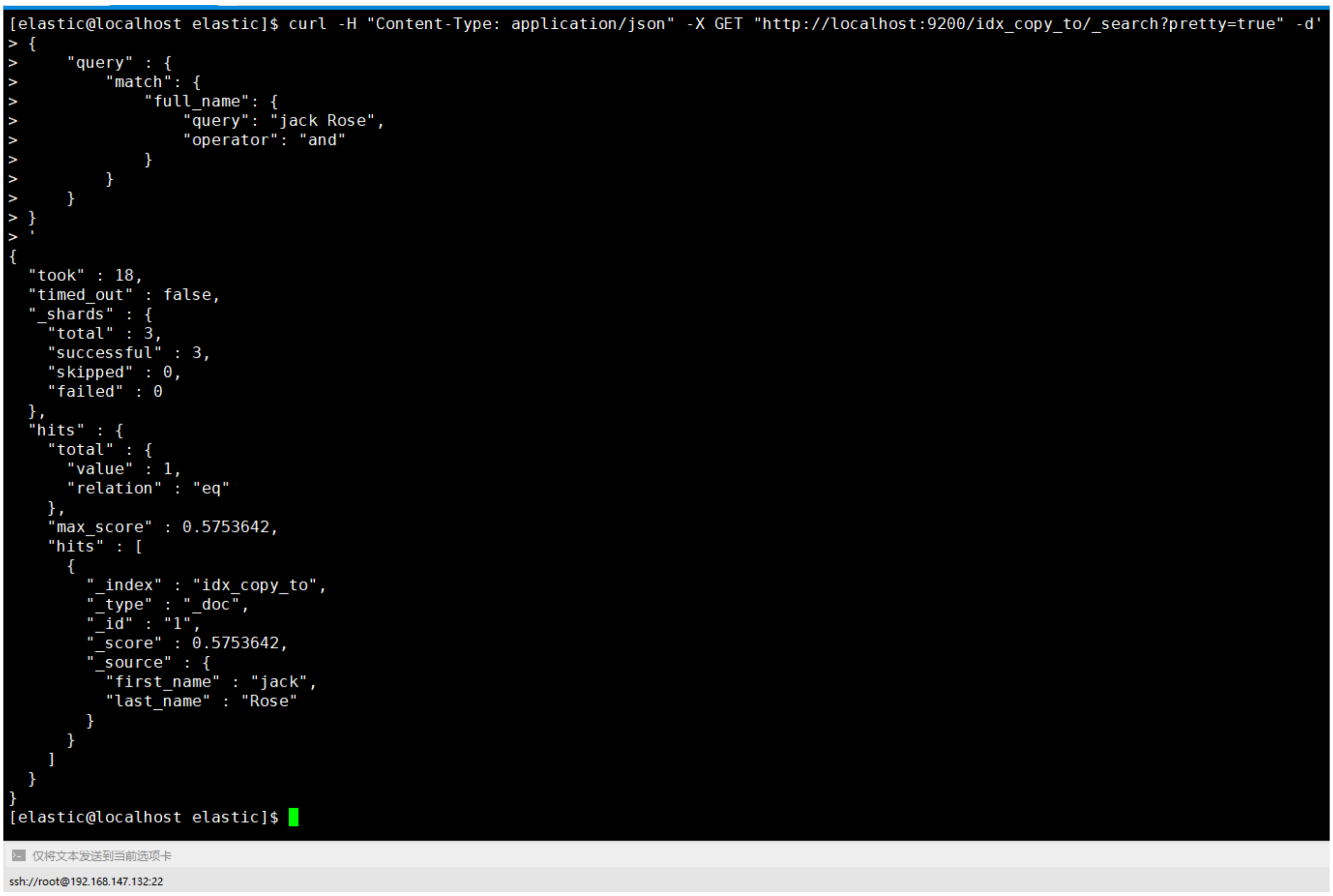
1.4.3.3.2. 新增数据

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/idx_copy_to/_doc/1?pretty=true" -d'
> {
>     "first_name" : "jack",
>     "last_name" : "Rose"
> }
> '
{
  "_index" : "idx_copy_to",
  "_type" : "_doc",
  "_id" : "1",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 3,
    "successful" : 1,
    "failed" : 0
  },
  "_seq_no" : 0,
  "_primary_term" : 1
}
```

1.4.3.3.3. 查询数据

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/idx_copy_to/_search?pretty=true" -d'
{
    "query" : {
        "match": {
            "full_name": {
                "query": "jack Rose",
                "operator": "and"
            }
        }
    }
}
'
```

从下图中得知first_name和 last_name字段取值都被复制到 full_name 字段。



```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET "http://localhost:9200/idx_copy_to/_search?pretty=true" -d'
> {
>     "query" : {
>         "match": {
>             "full_name": {
>                 "query": "jack Rose",
>                 "operator": "and"
>             }
>         }
>     }
> }
> '
{
  "took" : 18,
  "timed_out" : false,
  "_shards" : {
    "total" : 3,
    "successful" : 3,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 1,
      "relation" : "eq"
    },
    "max_score" : 0.5753642,
    "hits" : [
      {
        "_index" : "idx_copy_to",
        "_type" : "_doc",
        "_id" : "1",
        "_score" : 0.5753642,
        "_source" : {
          "first_name" : "jack",
          "last_name" : "Rose"
        }
      }
    ]
  }
}
[elastic@localhost elastic]$
```

1.4.3.4. doc_values

是为了加快排序、聚合操作，在建立倒排索引的时候，额外增加一个列式存储映射，是一个空间换时间的做法。默认是开启的，对于确定不需要聚合或者排序的字段可以关闭。

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/idx_doc_val?pretty=true"  -d'
{
    "settings" : {
        "index" : {
            "number_of_shards" : 3,
            "number_of_replicas" : 2
        }
    },
    "mappings" : {
            "properties" : {
                "first_name":{
                    "type" : "text"
                },
                "last_name":{
                    "type":"text",
                    "doc_values": false
                }
            }
    }
}
'
```

1.4.3.5. dynamic

默认情况下，字段可以自动添加到文档或者文档的内部对象，elasticsearc也会自动索引映射字段。

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X PUT
"http://localhost:9200/idx_dynamic?pretty=true"  -d'
{
    "settings" : {
        "index" : {
            "number_of_shards" : 3,
            "number_of_replicas" : 2
        }
    },
    "mappings" : {
            "properties" : {
                "first_name":{
                    "type" : "text"
                },
                "last_name":{
                    "type":"text",
                    "doc_values": false
                }
            }
    }
}
'
```

2. SpringBoot集成

以下三个版本一定排除掉，开始时候没注意发现这个坑，踩了好久

2.1. POM

```
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>elasticsearch-rest-high-level-client</artifactId>
    <version>7.4.0</version>
    <exclusions>
        <exclusion>
            <groupId>org.elasticsearch</groupId>
            <artifactId>elasticsearch</artifactId>
        </exclusion>
        <exclusion>
            <groupId>org.elasticsearch.client</groupId>
```

```
                <artifactId>elasticsearch-rest-client</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>org.elasticsearch.client</groupId>
        <artifactId>elasticsearch-rest-client</artifactId>
        <version>7.4.0</version>
    </dependency>
    <dependency>
        <groupId>org.elasticsearch</groupId>
        <artifactId>elasticsearch</artifactId>
        <version>7.4.0</version>
    </dependency>
```

2.2. yml配置

```
server:
  port: 9090
spring:
  datasource:
    name: mysql
    type: com.alibaba.druid.pool.DruidDataSource
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://127.0.0.1:3306/springboot?
useUnicode=true&characterEncoding=utf8&useSSL=false&serverTimezone=UTC
    username: root
    password: 123456
    druid:
      initial-size: 5
      min-idle: 5
      max-active: 20
      max-wait: 30000
      time-between-eviction-runs-millis: 60000
      min-evictable-idle-time-millis: 300000
      validation-query: select 1
      test-while-idle: true
      test-on-borrow: false
      test-on-return: false
      pool-prepared-statements: false
      max-pool-prepared-statement-per-connection-size: 20
      connectionProperties: druid.stat.mergeSql=true;druid.stat.slowSqlMillis=6000
es:
  host: 192.168.147.132
  port: 9200
  scheme: http

mybatis:
  mapperLocations: classpath:mapper/**/*.xml
```

这里定义

```
es:
  host: 192.168.147.132
  port: 9200
  scheme: http
```

2.3. 核心操作类

```
package xyz.wongs.weathertop.base.dao;

import com.alibaba.fastjson.JSON;
import lombok.extern.slf4j.Slf4j;
import org.elasticsearch.action.admin.indices.delete.DeleteIndexRequest;
import org.elasticsearch.action.bulk.BulkRequest;
import org.elasticsearch.action.delete.DeleteRequest;
import org.elasticsearch.action.index.IndexRequest;
import org.elasticsearch.action.search.SearchRequest;
import org.elasticsearch.action.search.SearchResponse;
import org.elasticsearch.action.support.IndicesOptions;
import org.elasticsearch.client.RequestOptions;
import org.elasticsearch.client.RestHighLevelClient;
import org.elasticsearch.client.indices.CreateIndexRequest;
import org.elasticsearch.client.indices.CreateIndexResponse;
import org.elasticsearch.client.indices.GetIndexRequest;
import org.elasticsearch.common.settings.Settings;
import org.elasticsearch.common.xcontent.XContentType;
import org.elasticsearch.index.query.QueryBuilder;
import org.elasticsearch.index.reindex.DeleteByQueryRequest;
import org.elasticsearch.search.SearchHit;
import org.elasticsearch.search.builder.SearchSourceBuilder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import sun.rmi.runtime.Log;
import xyz.wongs.weathertop.base.entiy.ElasticEntity;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

@Slf4j
@Component
public class BaseElasticDao {

    @Autowired
    RestHighLevelClient restHighLevelClient;

    /**
     * @author WCNGS@QQ.COM
     * @See
     * @date 2019/10/17 17:30
     * @param idxName   索引名称
     * @param idxSQL    索引描述
     * @return void
     * @throws
     * @since
     */
    public void createIndex(String idxName,String idxSQL){

        try {

            if (!this.indexExist(idxName)) {
                log.error(" idxName={} 已经存在,idxSql={}",idxName,idxSQL);
                return;
            }
            CreateIndexRequest request = new CreateIndexRequest(idxName);
            buildSetting(request);
            request.mapping(idxSQL, XContentType.JSON);
            CreateIndexResponse res =
restHighLevelClient.indices().create(request, RequestOptions.DEFAULT);
            if (!res.isAcknowledged()) {
                throw new RuntimeException("初始化失败");
            }
        } catch (Exception e) {
            e.printStackTrace();
            System.exit(0);
        }
    }

    /** 断某个index是否存在
     * @author WCNGS@QQ.COM
     * @See
     * @date 2019/10/17 17:27
     * @param idxName index名
     * @return boolean
     * @throws
     * @since
     */
    public boolean indexExist(String idxName) throws Exception {
        GetIndexRequest request = new GetIndexRequest(idxName);
        request.local(false);
        request.humanReadable(true);
        request.includeDefaults(false);

        request.indicesOptions(IndicesOptions.lenientExpandOpen());
        return restHighLevelClient.indices().exists(request,
RequestOptions.DEFAULT);
    }

    /** 设置分片
     * @author WCNGS@QQ.COM
     * @See
     * @date 2019/10/17 19:27
     * @param request
     * @return void
     * @throws
     * @since
     */
    public void buildSetting(CreateIndexRequest request){

        request.settings(Settings.builder().put("index.number_of_shards",3)
                .put("index.number_of_replicas",2));
    }
    /**
     * @author WCNGS@QQ.COM
     * @See
     * @date 2019/10/17 17:27
     * @param idxName index
     * @param entity    对象
     * @return void
     * @throws
     * @since
     */
    public void insertOrUpdateOne(String idxName, ElasticEntity entity) {

        IndexRequest request = new IndexRequest(idxName);
        request.id(entity.getId());
        request.source(JSON.toJSONString(entity.getData()), XContentType.JSON);
        try {
            restHighLevelClient.index(request, RequestOptions.DEFAULT);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }


    /** 批量插入数据
     * @author WCNGS@QQ.COM
     * @See
     * @date 2019/10/17 17:26
     * @param idxName index
     * @param list 带插入列表
     * @return void
     * @throws
     * @since
     */
    public void insertBatch(String idxName, List<ElasticEntity> list) {

        BulkRequest request = new BulkRequest();
        list.forEach(item -> request.add(new
IndexRequest(idxName).id(item.getId())
                .source(JSON.toJSONString(item.getData()), XContentType.JSON)));
        try {
            restHighLevelClient.bulk(request, RequestOptions.DEFAULT);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
    /** 批量删除
     * @author WCNGS@QQ.COM
     * @See
     * @date 2019/10/17 17:14
     * @param idxName index
     * @param idList    待删除列表
     * @return void
     * @throws
     * @since
     */
    public <T> void deleteBatch(String idxName, Collection<T> idList) {

        BulkRequest request = new BulkRequest();
        idList.forEach(item -> request.add(new DeleteRequest(idxName,
item.toString())));
        try {
            restHighLevelClient.bulk(request, RequestOptions.DEFAULT);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    /**
     * @author WCNGS@QQ.COM
     * @See
     * @date 2019/10/17 17:14
     * @param idxName index
     * @param builder   查询参数
     * @param c 结果类对象
     * @return java.util.List<T>
     * @throws
     * @since
     */
    public <T> List<T> search(String idxName, SearchSourceBuilder builder,
Class<T> c) {

        SearchRequest request = new SearchRequest(idxName);
        request.source(builder);
        try {
            SearchResponse response = restHighLevelClient.search(request,
RequestOptions.DEFAULT);
            SearchHit[] hits = response.getHits().getHits();
            List<T> res = new ArrayList<>(hits.length);
            for (SearchHit hit : hits) {
                res.add(JSON.parseObject(hit.getSourceAsString(), c));
            }
            return res;
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    /** 删除index
```

```
     * @author WCNGS@QQ.COM
     * @See
     * @date 2019/10/17 17:13
     * @param idxName
     * @return void
     * @throws
     * @since
     */
    public void deleteIndex(String idxName) {
        try {
            restHighLevelClient.indices().delete(new DeleteIndexRequest(idxName),
RequestOptions.DEFAULT);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * @author WCNGS@QQ.COM
     * @See
     * @date 2019/10/17 17:13
     * @param idxName
     * @param builder
     * @return void
     * @throws
     * @since
     */
    public void deleteByQuery(String idxName, QueryBuilder builder) {

        DeleteByQueryRequest request = new DeleteByQueryRequest(idxName);
        request.setQuery(builder);
        //设置批量操作数量,最大为10000
        request.setBatchSize(10000);
        request.setConflicts("proceed");
        try {
            restHighLevelClient.deleteByQuery(request, RequestOptions.DEFAULT);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

3. 实战

3.1. 创建索引

由于在**BaseElasticDao**类中**createIndex**方法，我在Controller层将索引名称和索引SQL封装过，详细见Github演示源码 中**xyz.wongs.weathertop.palant.vo.IdxVo**

演示索引

```
{
    "idxName": "idx_locat",
    "idxSql": {
        "dynamic": false,
        "properties": {
            "id": {
                "type": "long"
            },
            "flag": {
                "type": "text",
                "index": true
            },
            "localCode": {
                "type": "text",
                "index": true
            },
            "localName": {
                "type": "text",
                "index": true,
                "analyzer": "ik_max_word"
            },
            "lv": {
                "type": "long"
            },
            "supLocalCode": {
                "type": "text",
                "index": true
            },
            "url": {
                "type": "text",
                "index": true
            }
        }
    }
}
```

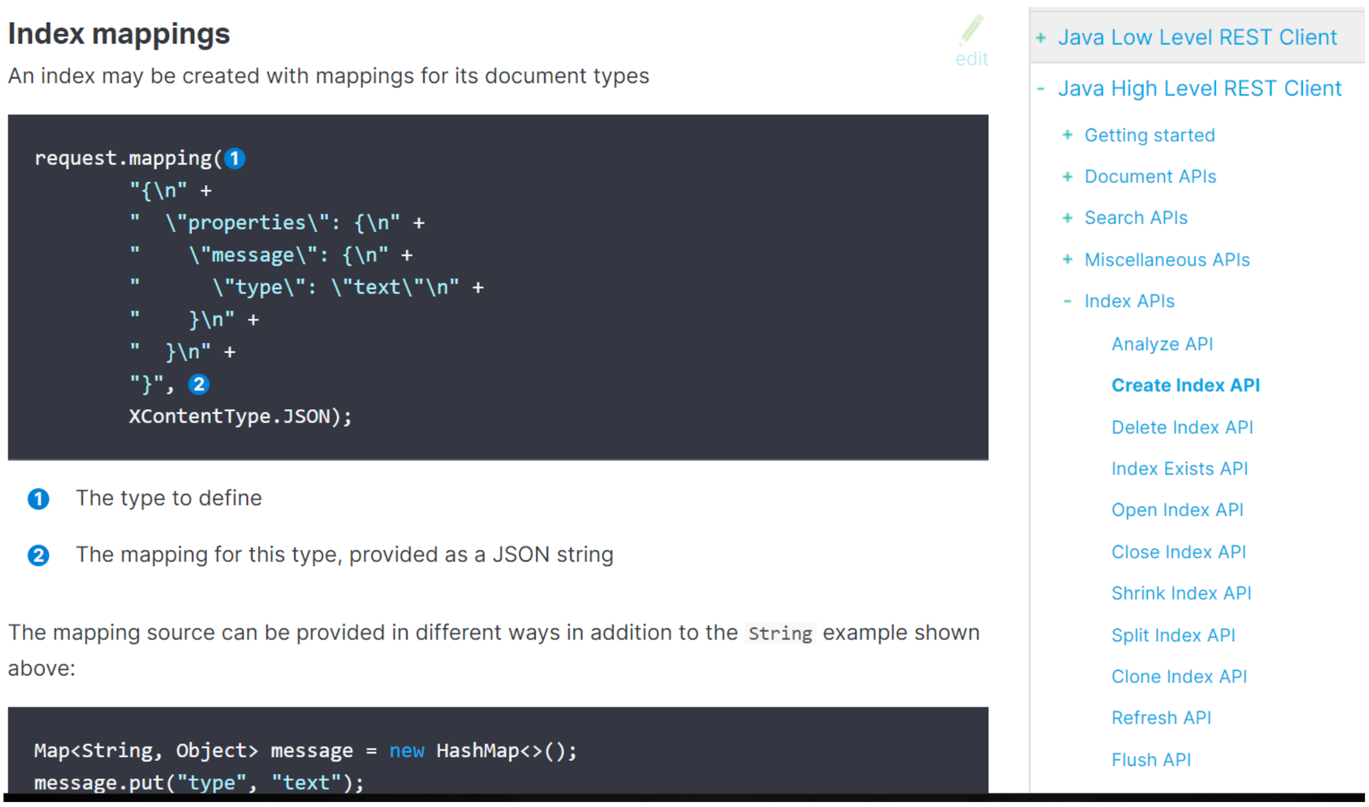
核心代码说明

- DAO层Elastic Search操作说明

```
public void createIndex(String idxName,String idxSQL){

    try {

        if (!this.indexExist(idxName)) {
            log.error(" idxName={} 已经存在,idxSql={}",idxName,idxSQL);
            return;
        }
```

```
            CreateIndexRequest request = new CreateIndexRequest(idxName);
            //1、创建Setting
            buildSetting(request);
            //2、指定索引的JSON
            request.mapping(idxSQL, XContentType.JSON);
            CreateIndexResponse res = restHighLevelClient.indices().create(request,
    RequestOptions.DEFAULT);
            //3、指定此类型的映射，以JSON字符串形式提供
            if (!res.isAcknowledged()) {
                throw new RuntimeException("初始化失败");
            }
        } catch (Exception e) {
            e.printStackTrace();
            System.exit(0);
        }
    }
```

如官网截图：



Index mappings

An index may be created with mappings for its document types

```
request.mapping( 1
        "{\n" +
        "  \"properties\": {\n" +
        "    \"message\": {\n" +
        "      \"type\": \"text\"\n" +
        "    }\n" +
        "  }\n" +
        "}", 2
        XContentType.JSON);
```

1 The type to define

2 The mapping for this type, provided as a JSON string

The mapping source can be provided in different ways in addition to the `String` example shown above:

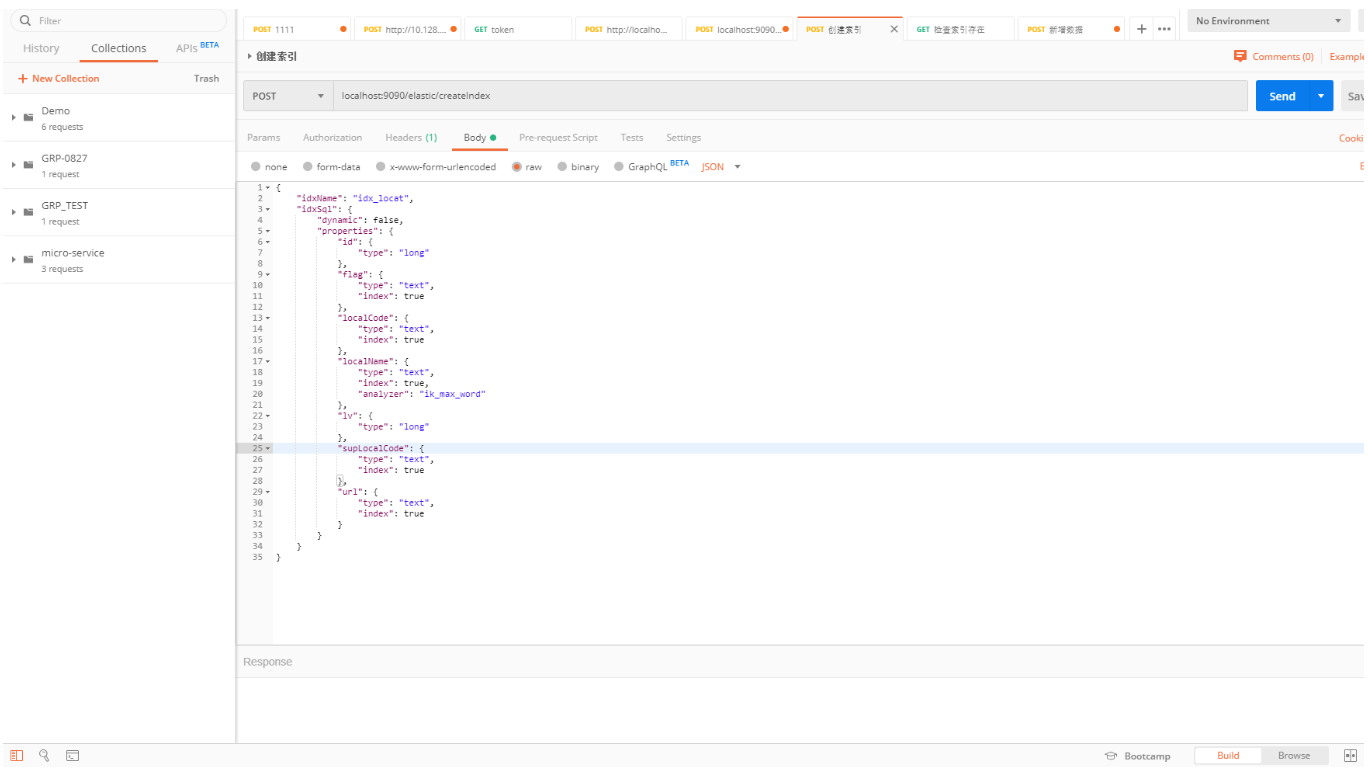
```
Map<String, Object> message = new HashMap<>();
message.put("type", "text");
```

- Controller Http API说明

```
@RequestMapping(value = "/createIndex",method = RequestMethod.POST)
public ResponseResult createIndex(@RequestBody IdxVo idxVo){
    ResponseResult response = new ResponseResult();
    String idxSql = JSONObject.toJSONString(idxVo.getIdxSql());
    log.warn(" idxName={}, idxSql={}",idxVo.getIdxName(),idxSql);
    baseElasticDao.createIndex(idxVo.getIdxName(),idxSql);
    return response;
}
```

- Postman调用**Controller**，发现创建索引成功。

```
[elastic@localhost elastic]$ curl -H "Content-Type: application/json" -X GET
"http://localhost:9200/_cat/indices?v"
health status index        uuid                   pri rep docs.count docs.deleted
store.size pri.store.size
yellow open   twitter      scSSD1SfRCio4F77Hh8aqQ   3   2          2            0
8.3kb          8.3kb
yellow open   idx_location _BJ_pOv0SkS4tv-EC3xDig   3   2          1            0
4kb            4kb
yellow open   wongs        uT13XiyjSW-VOS3GCqao8w   3   2          1            0
3.4kb          3.4kb
yellow open   idx_locat    Kr3wGU7JT_OUrRJkyFSGDw   3   2          3            0
13.2kb         13.2kb
yellow open   idx_copy_to  HouC9s6LSjiwrJtDicgY3Q   3   2          1            0
4kb            4kb
```



POST localhost:9090/elastic/createIndex

```
{
    "idxName": "idx_locat",
    "idxSql": {
        "dynamic": false,
        "properties": {
            "id": {
                "type": "long"
            },
            "flag": {
                "type": "text",
                "index": true
            },
            "localCode": {
                "type": "text",
                "index": true
            },
            "localName": {
                "type": "text",
                "index": true,
                "analyzer": "ik_max_word"
            },
            "lv": {
                "type": "long"
            },
            "supLocalCode": {
                "type": "text",
                "index": true
            },
            "url": {
                "type": "text",
                "index": true
            }
        }
    }
}
```

4. 源码

Github演示源码 ， 记得给Star