

# 1、ES api 储存结构

ES 是以 RestFul api 风格来命名自己的api的

1、api 基本格式：

<http://localhost:9200/<索引>/<类型>/<文档id>>

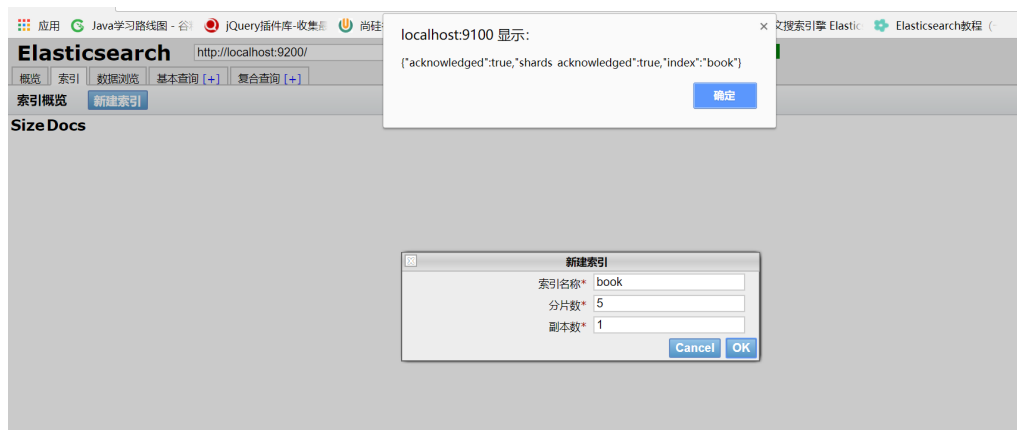
2、常用 HTTP 动词：

POST/GET/PUT/DELETE

## 2、创建索引

### 非结构化创建

使用 head 插件创建（创建名称，是英文名，小写不能有中划线），创建结果如下图：

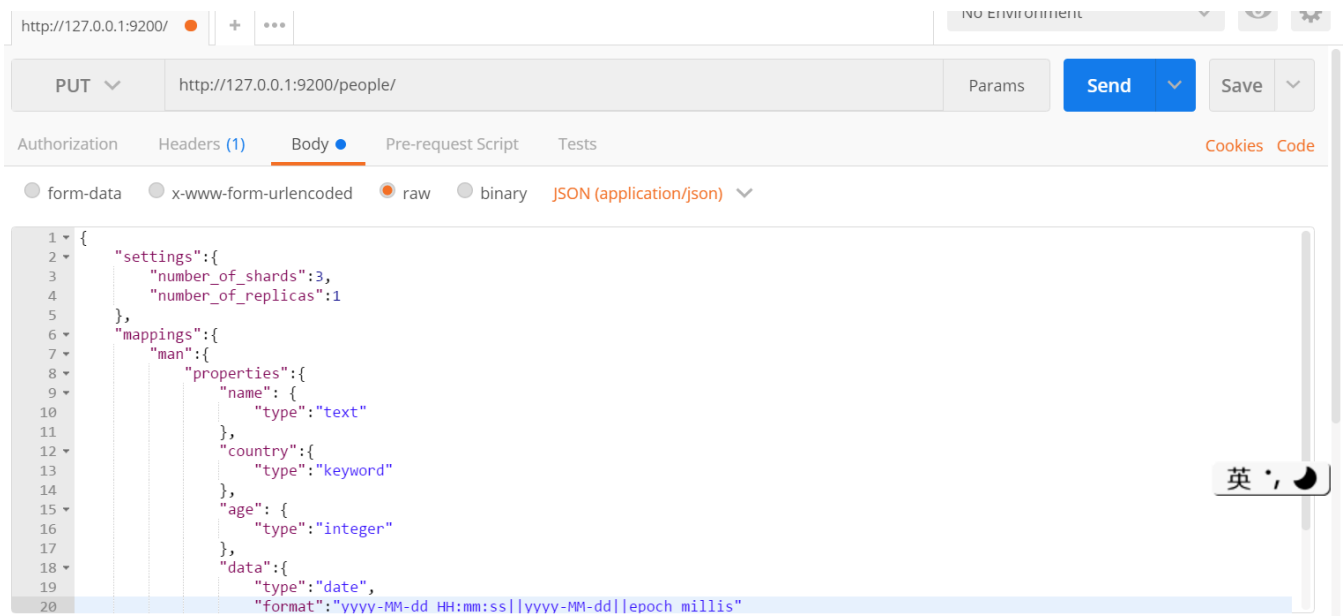


### 结构化创建

1、使用 head 插件进行创建（点击复合查询 --> 输入如下 json --> 提交）



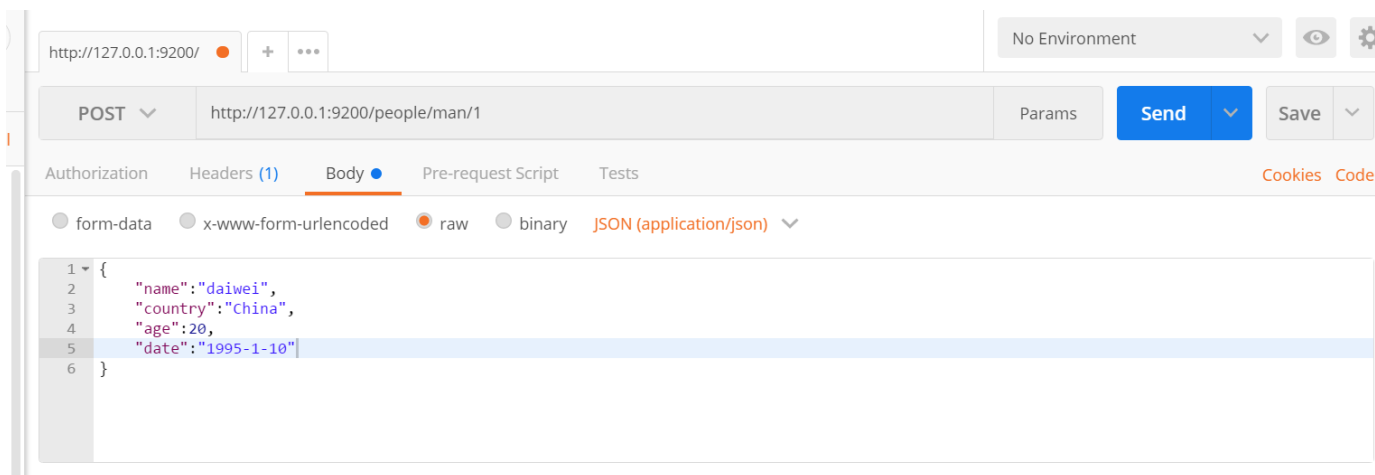
## 2、使用 Postman 进行创建



## 3、插入数据

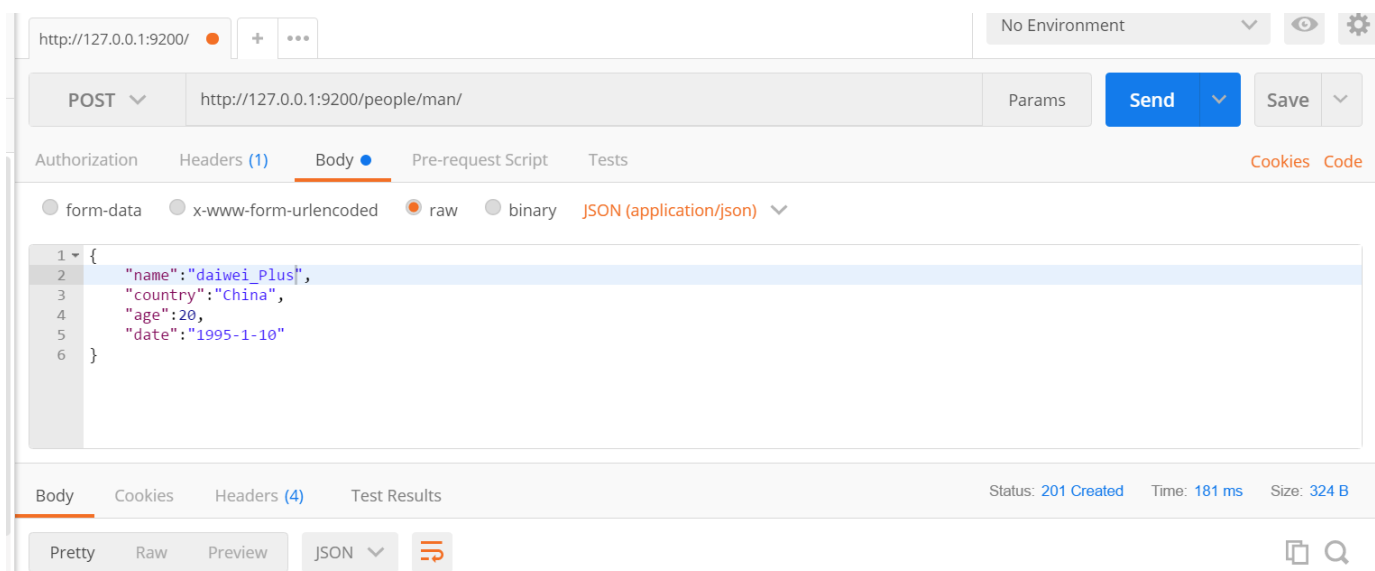
### 指定id插入

这里使用 Postman 进行插入，插入结构应该遵守创建索引的结构，插入如下图：



## 不指定id插入

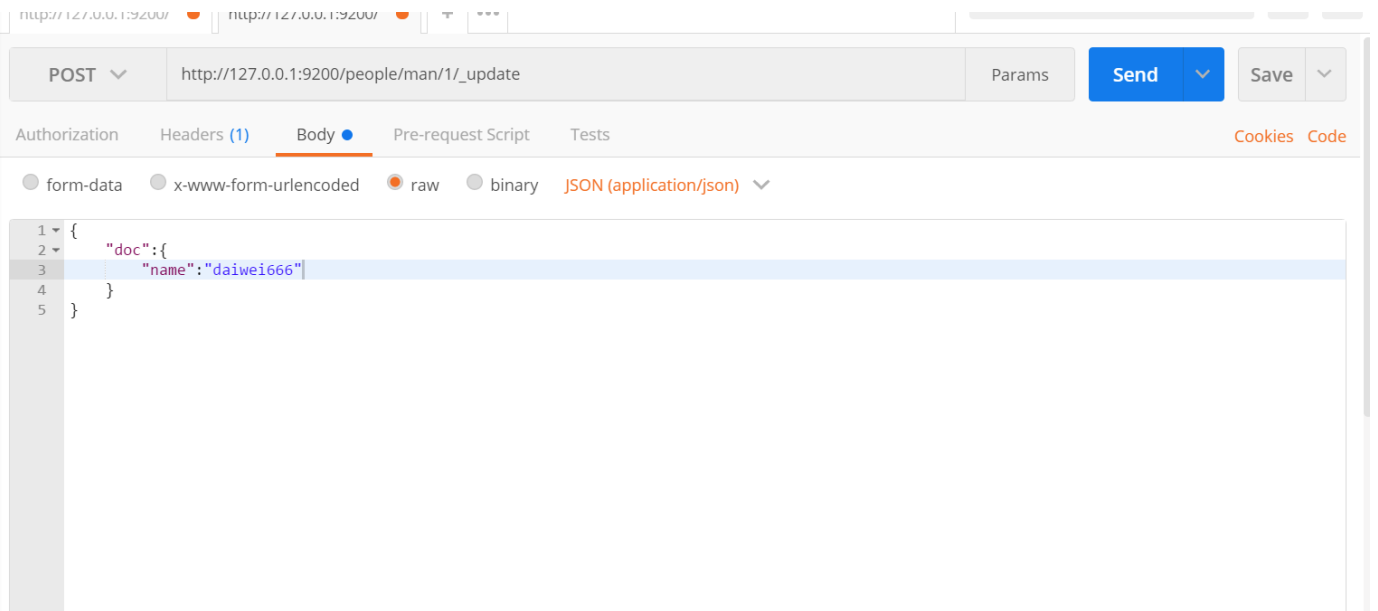
指定 id 插入，只需要在类型后面指定 id 即可，不指定 id 插入，让 es 自动生成



## 4、修改数据

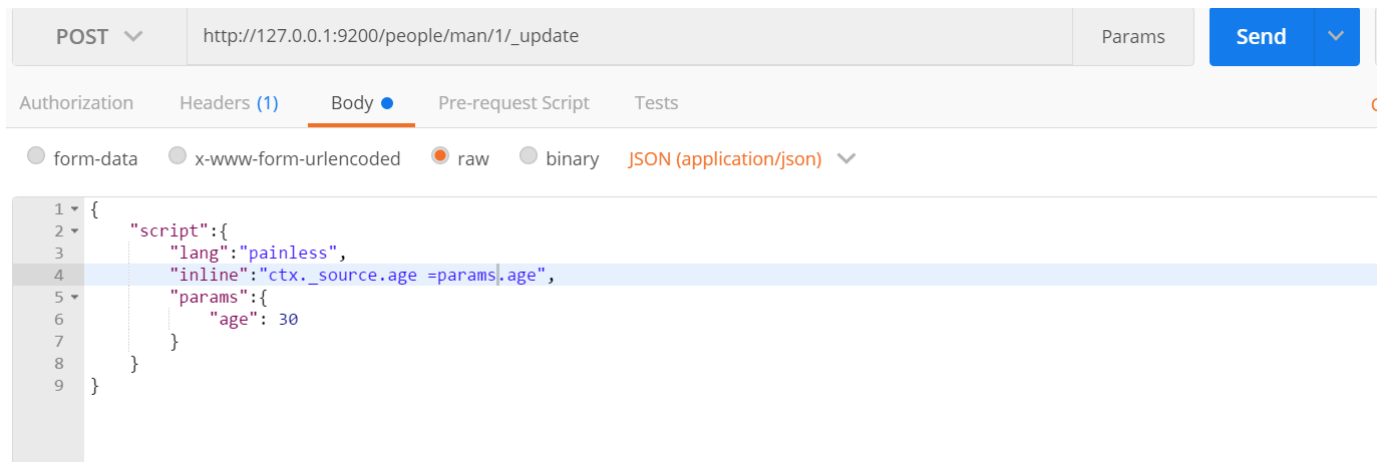
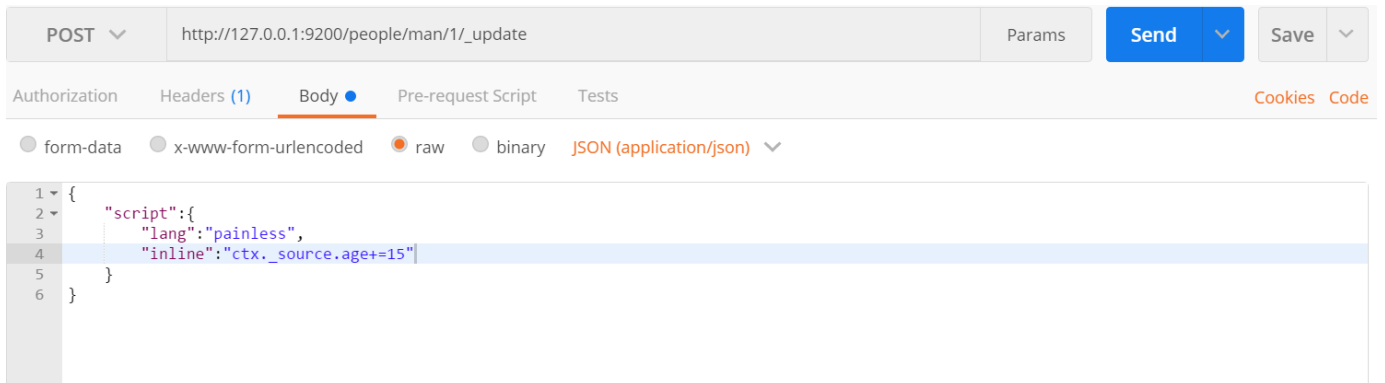
### 使用 doc 的方式修改数据

操作如下图：



## 使用脚本的方式修改数据

把 id =1 的数据 年龄增加 15，操作语言如下图：



## 5、删除操作

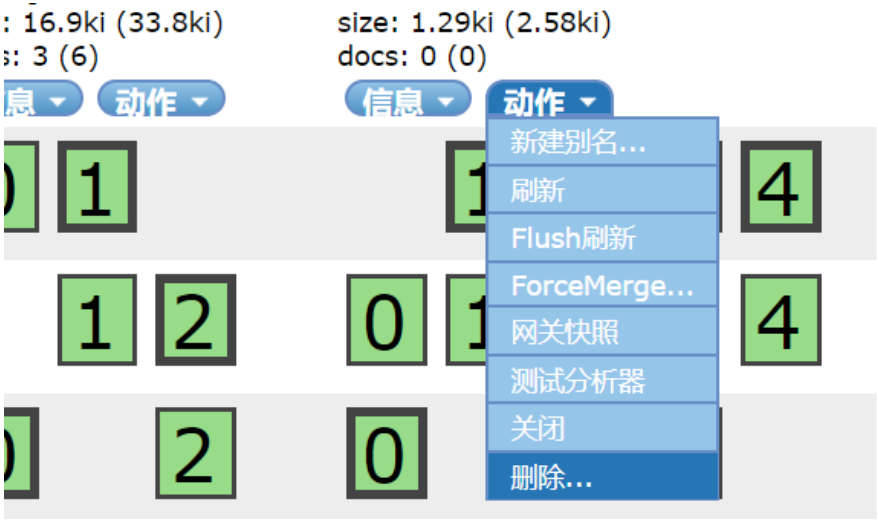
## 删除一个document操作

删除 id 为 1 的数据，操作如下：

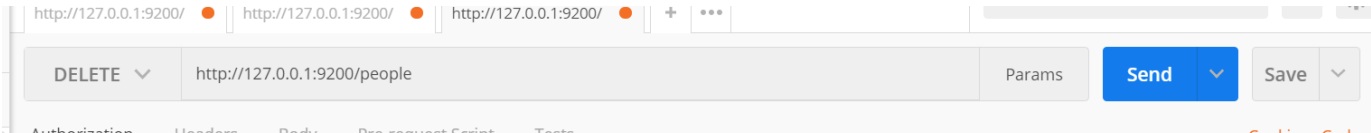


## 删除索引

1、使用 head 插件进行删除：



2、使用 Postman 进行删除：



# 6、查询操作

查询分类：

- 1、简单查询
- 2、条件查询

query DSL : Domain Specified Language 特定领域语言 ---> es 特有的查询语言

简单查询：

使用Postman 访问 api 接口，Get 127.0.0.1:9200/索引/类型/id

GET ▼ http://127.0.0.1:9200/book/books/8n76RWABM1nZ0tZ2930H Params Send ▼

Inherit auth from parent ▼

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the p authorization helper.

Body Cookies Headers (3) Test Results Status: 200 OK Time: 164 ms

Pretty Raw Preview JSON ▼ ≡

```
1 {
2   "_index": "book",
3   "_type": "books",
4   "_id": "8n76RWABM1nZ0tZ2930H",
5   "_version": 1,
6   "found": true,
7   "_source": {
8     "title": "elasticsearch学习手册",
9     "writer": "很多人",
10    "word_count": 6664433,
11    "publish_date": "2016-8-3"
12  }
13 }
```

## 条件查询：

1、使用 Postman api 访问 Post 127.0.0.1:9200/book/\_search 使用query关键字 类似  
**分页查询**  
(使用 from 关键字，设置从哪返回数据，使用 size 关键字 设置返回数据条数)，  
查询所有：

POST ▼ http://127.0.0.1:9200/book/\_search Params Send ▼

Authorization Headers (1) Body ● Pre-request Script Tests

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json) ▼

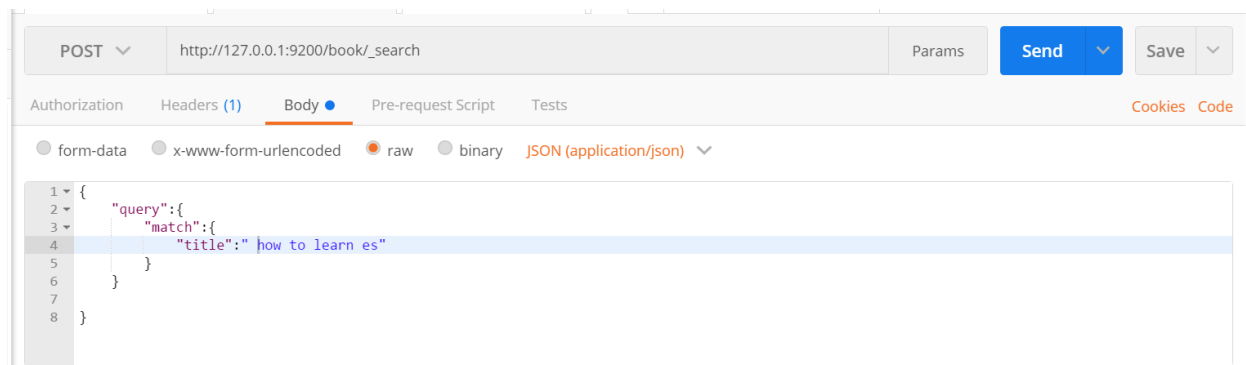
```
1 {
2   "query": {
3     "match_all": {}
4   },
5   "from": 1,
6   "size": 1
7 }
```

Body Cookies Headers (3) Test Results Status: 200 OK Time: 128 ms

Pretty Raw Preview JSON ▼ ≡

```
1 {
2   "took": 13,
3   "timed_out": false,
4   "_shards": {
5     "total": 3,
6     "successful": 3,
7     "skipped": 0,
8     "failed": 0
9   }
10 }
```

2、设置查询条件，使用 match 关键字进行查询(**不知道为什么只能精确查询，不能模糊查询**) (ps: 如果搜索字段时关键字即 类型是 keyword 的时候是无法进行模糊搜索的，只有不为keyword 类型才可以进行 模糊查询)



3、设置查询结果进行排序，使用 **sort** 进行对需要排序字段进行排序

```
1 GET ecommerce/product/_search
2 {
3   "query": {
4     "match": {
5       "name": "pro"
6     }
7   },
8   "sort": [
9     {
10      "price": "desc"
11    }
12  ]
13 }
```

## 7、高级查询操作

数据过滤 ( query filter)

演示查询 json

```
1 GET /ecommerce/product/_search
2 {
3   "query": {
4     "bool": {
5       "must": [           //must 下面是必须要匹配的查询条件
```

```

6      {
7          "match": {
8              "name": "xiaomi"
9          }
10     },
11     {
12         "match": {
13             "desc": "qijian"
14         }
15     }
16 ],
17 "filter": {      // filter 是要过滤的条件， 是要在查询出来的东西中过滤，所以要
和查询条件的 must 并列
18     "range": {
19         "price": {
20             "gte": 1000,
21             "lte": 5000
22         }
23     }
24 }
25 }
26 }
27 }

```

## 全文检索 ( full-text search )

### 演示查询 json

```

1 GET /ecommerce/product/_search
2 {
3     "query": {
4         "match": {
5             "desc": "xiaomi qijian shouji"
6         }
7     }
8 }

```

### 返回结果

```

1 {
2     "took": 49,
3     "timed_out": false,
4     "_shards": {

```



```
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9 },
10 "hits": {
11     "total": 2,
12     "max_score": 0.8630463,
13     "hits": [
14         {
15             "_index": "ecommerce",
16             "_type": "product",
17             "_id": "1",
18             "_score": 0.8630463,           // _score 相关度分数，相似度越高，分数越高，分
数越高排的越前
19             "_source": {
20                 "name": "xiaomi 7",
21                 "desc": "xiaomi shouji jiushi kuai, niandu qijian",
22                 "price": 2399,
23                 "tag": [
24                     "xiaomi",
25                     "shouji"
26                 ]
27             }
28         },
29         {
30             "_index": "ecommerce",
31             "_type": "product",
32             "_id": "3",
33             "_score": 0.68324494,
34             "_source": {
35                 "name": "huawei mate 10",
36                 "desc": "huawei shouji niandu niandu shangwu shouji qijian",
37                 "price": 4399,
38                 "tag": [
39                     "huawei",
40                     "shouji"
41                 ]
42             }
43         }
44     ]
45 }
46 }
```

## 短语搜索 ( phrase search )

跟全文检索相对应，相反，全文检索会将输入的字符串拆解开来，去倒排索引中一一匹配，只要能匹配上任意一个拆解后的单词，就可以作为结果返回。

phrase search 要求是输入搜索串 必须在指定的字段文本中，完全包含一模一样的才算匹配，才能作为结果返回。

### 演示查询 json

```
1 GET /ecommerce/product/_search
2 {
3   "query": {
4     "match_phrase": { // 搜索串 必须在指定的字段文本中，完全包含一模一样
5       "name": "xiaomi"
6     }
7   }
8 }
```

### 结果 json

```
1 {
2   "took": 4,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": 2,
12    "max_score": 0.2876821,
13    "hits": [
14      {
15        "_index": "ecommerce",
16        "_type": "product",
17        "_id": "2",
18        "_score": 0.2876821,
19        "_source": {
20          "name": "xiaomi 6", // 包含 xiaomi
21          "desc": "xiaomi6 jiushi kuai!!",
22          "price": 2399,
23          "tag": [
```

```

24         "xiaomi",
25         "shouji"
26     ]
27 }
28 },
29 {
30     "_index": "ecommerce",
31     "_type": "product",
32     "_id": "1",
33     "_score": 0.2876821,
34     "_source": {
35         "name": "xiaomi 7",           // 包含 xiaomi
36         "desc": "xiaomi shouji jiushi kuai, niandu qijian",
37         "price": 2399,
38         "tag": [
39             "xiaomi",
40             "shouji"
41         ]
42     }
43 }
44 ]
45 }
46 }

```

## 高亮搜索结果 (highlight search)

演示查询 json

```

1 GET /ecommerce/product/_search
2 {
3     "query": {
4         "match": {
5             "name": "xiaomi"
6         }
7     },
8     "highlight": {           // 高亮 field
9         "fields": {
10             "name": {}
11         }
12     }
13 }

```

## 查询结果

```
1 {
2   "took": 17,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": 2,
12    "max_score": 0.2876821,
13    "hits": [
14      {
15        "_index": "ecommerce",
16        "_type": "product",
17        "_id": "2",
18        "_score": 0.2876821,
19        "_source": {
20          "name": "xiaomi 6",
21          "desc": "xiaomi6 jiushi kuai!!",
22          "price": 2399,
23          "tag": [
24            "xiaomi",
25            "shouji"
26          ]
27        },
28        "highlight": {           // 返回 html 代码 ，可用代码后续处理
29          "name": [
30            "<em>xiaomi</em> 6"
31          ]
32        }
33      },
34      {
35        "_index": "ecommerce",
36        "_type": "product",
37        "_id": "1",
38        "_score": 0.2876821,
39        "_source": {
40          "name": "xiaomi 7",
41          "desc": "xiaomi shouji jiushi kuai, niandu qijian",
42          "price": 2399,
```

```
43         "tag": [  
44             "xiaomi",  
45             "shouji"  
46         ]  
47     },  
48     "highlight": {  
49         "name": [  
50             "<em>xiaomi</em> 7"  
51         ]  
52     }  
53 }  
54 ]  
55 }  
56 }
```

