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://:/<索引>/<类型>/<文档id>

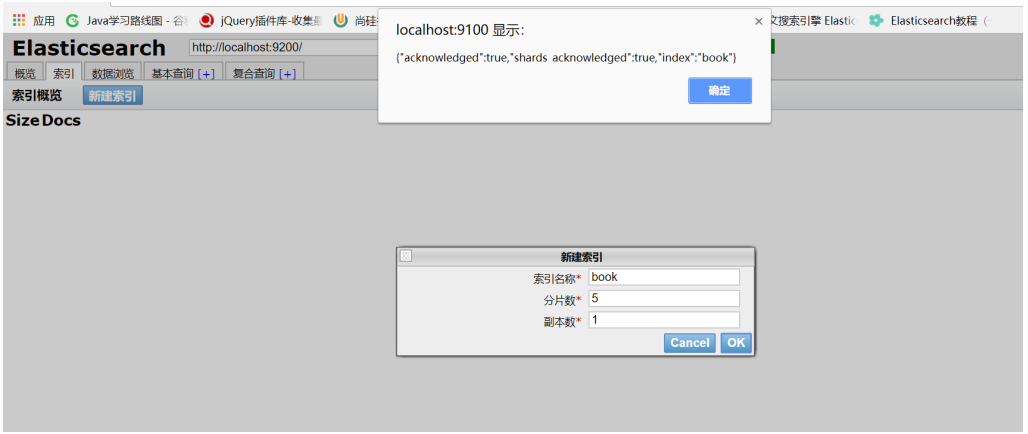
2、常用 HTTP 动词：

POST/GET/PUT/DELETE

2、创建索引

非结构化创建

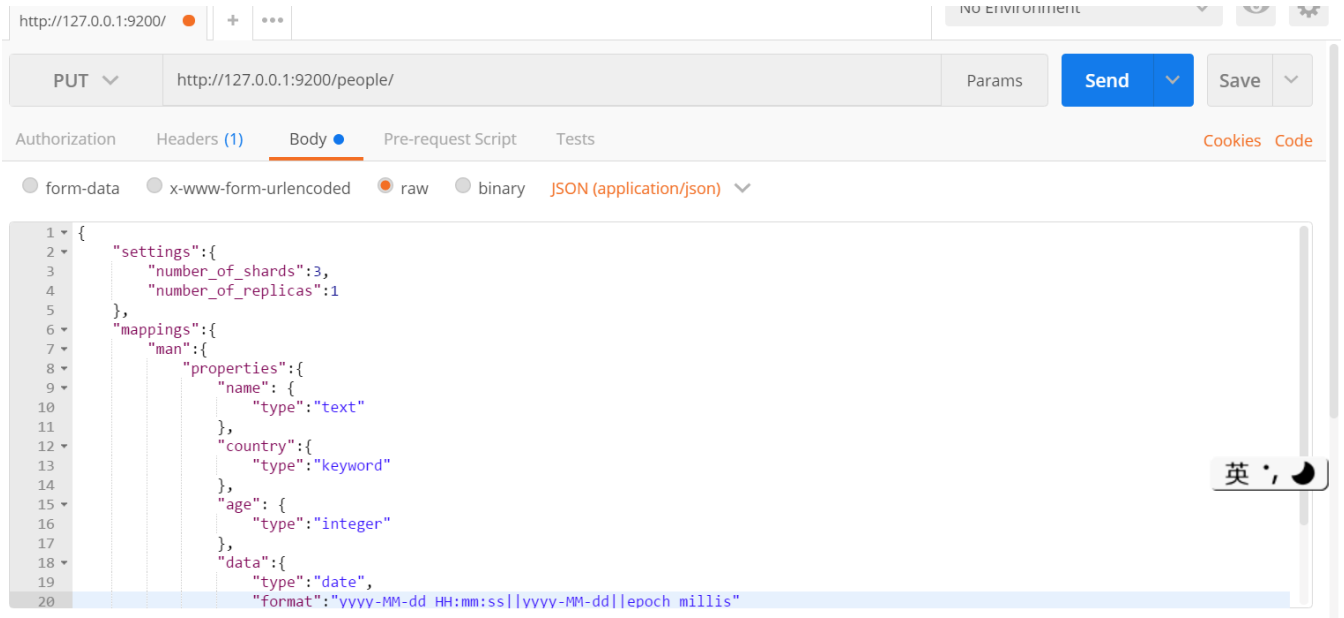
使用 head 插件创建（创建名称，是英文名，小写不能有中划线），创建结果如下图：



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）

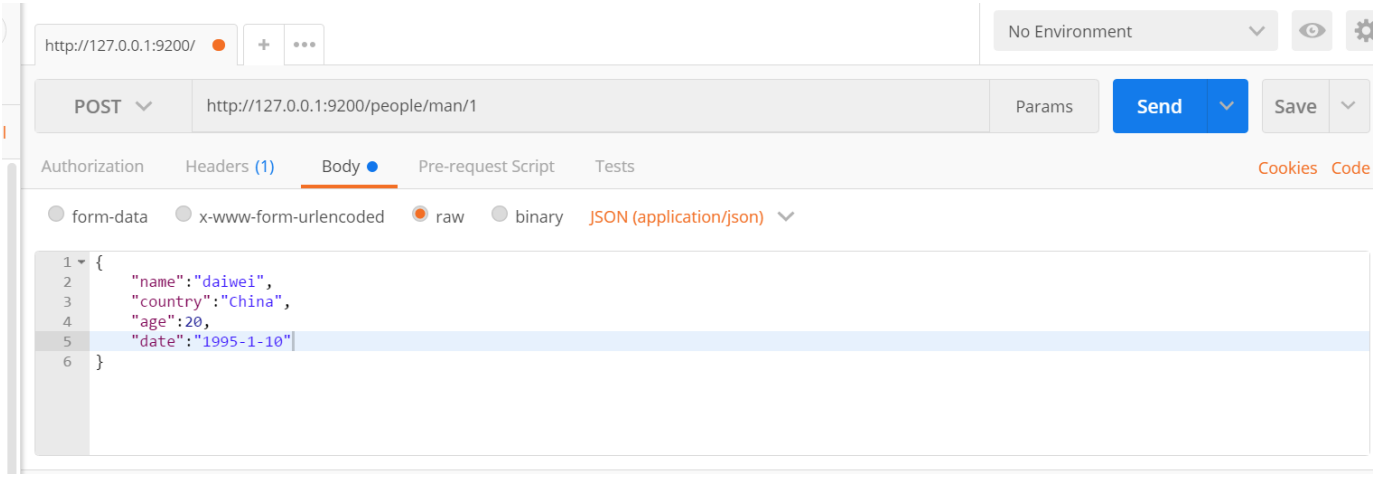
2、使用 Postman 进行创建



3、插入数据

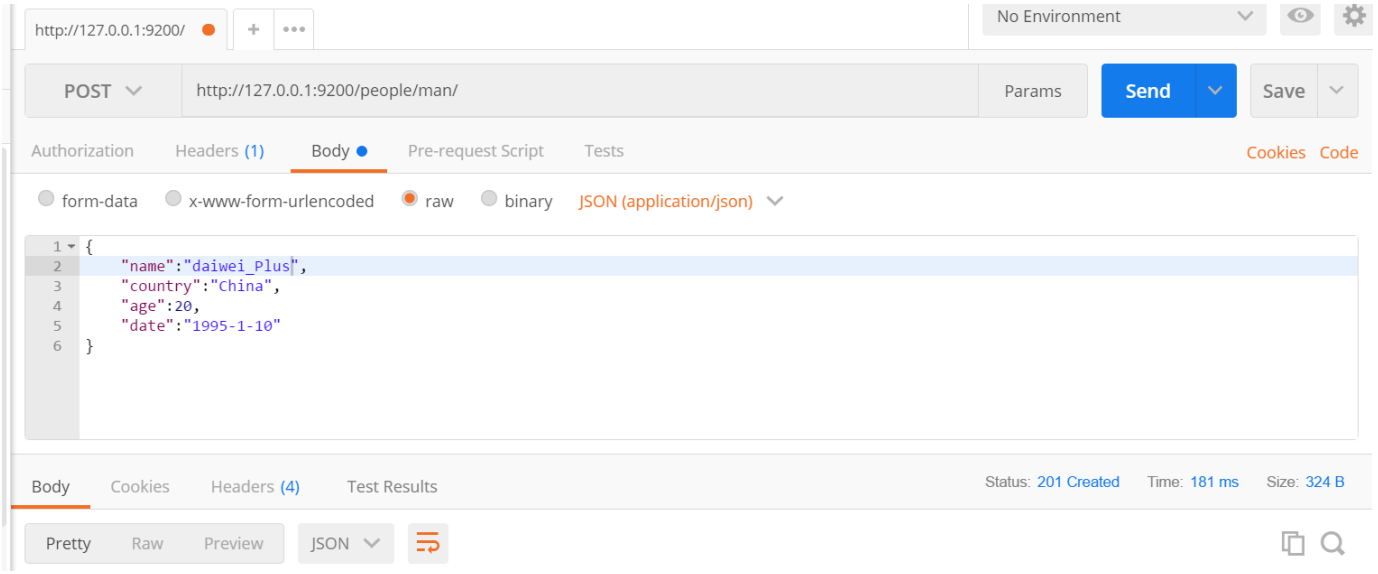
指定id插入

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

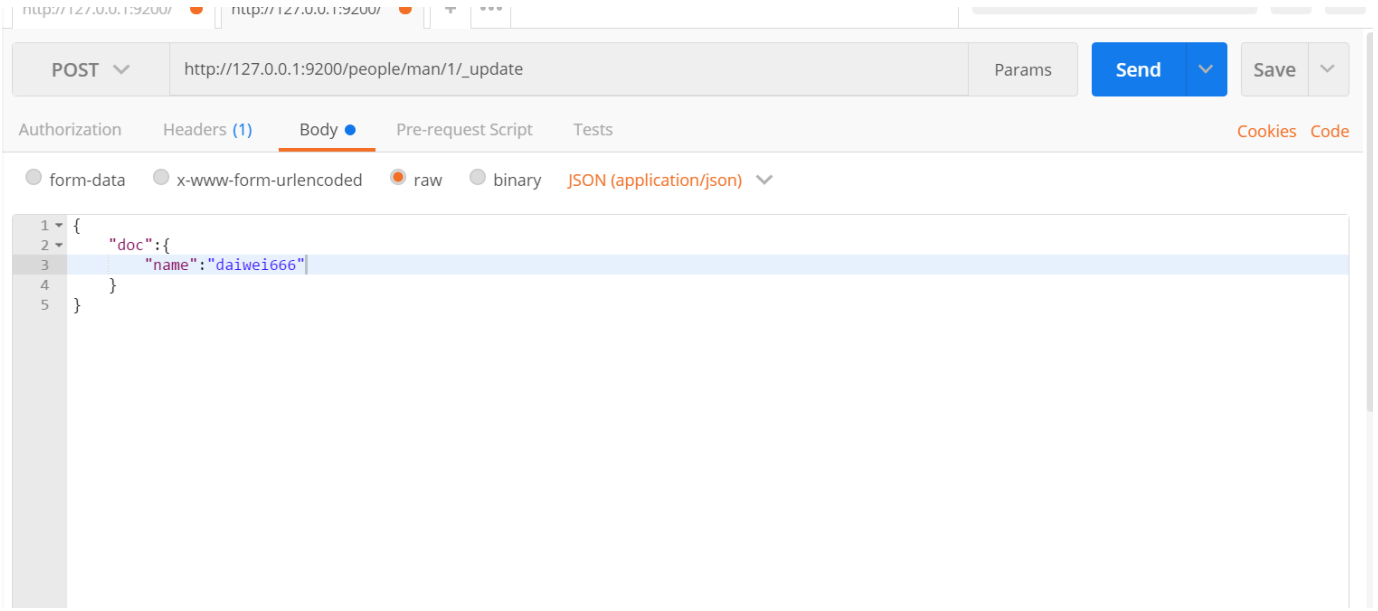
指定 id 插入，只需要在类型后面指定 id 即可，不指定 id 插入，让 es 自动生成



4、修改数据

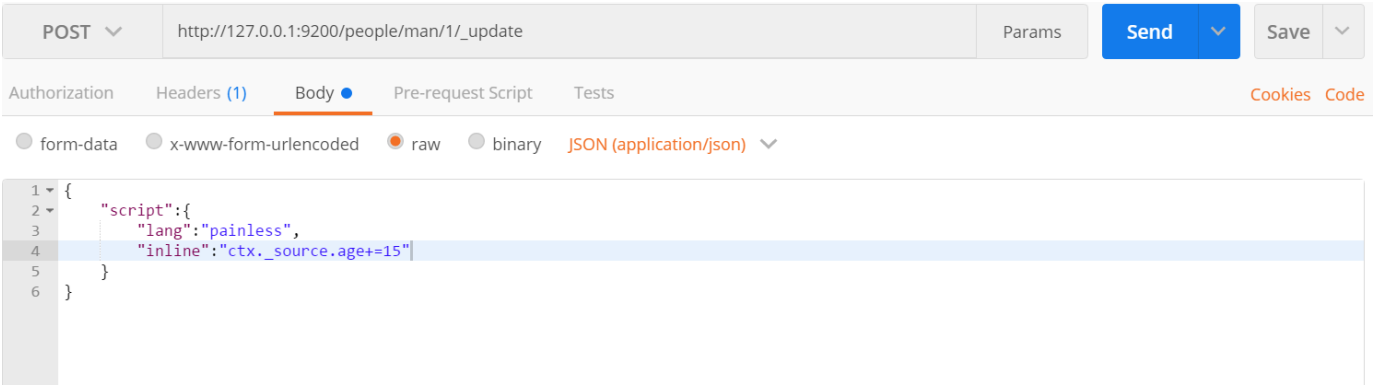
使用 doc 的方式修改数据

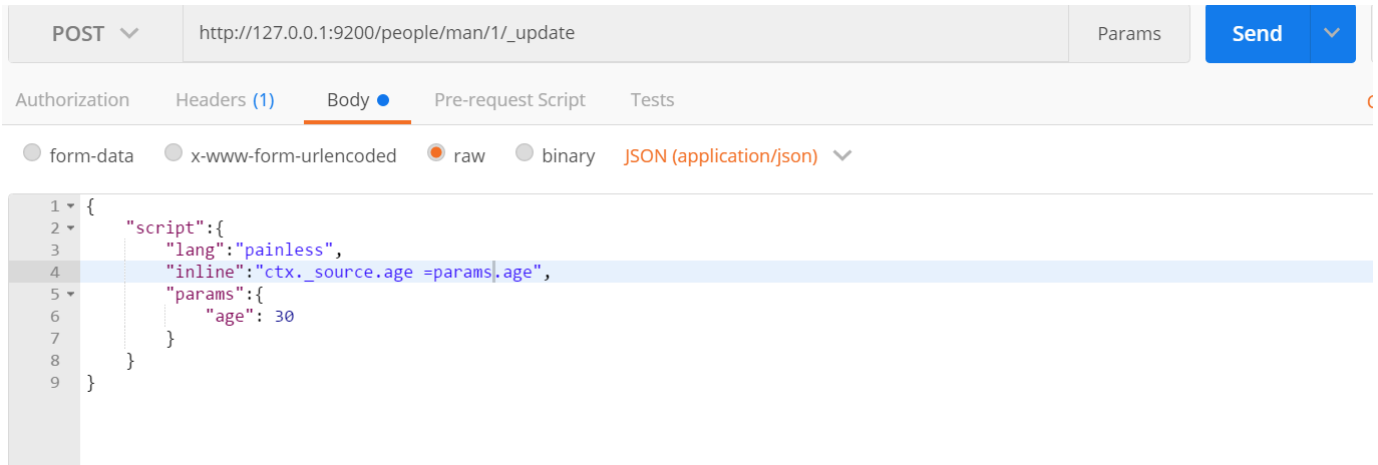
操作如下图：



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：





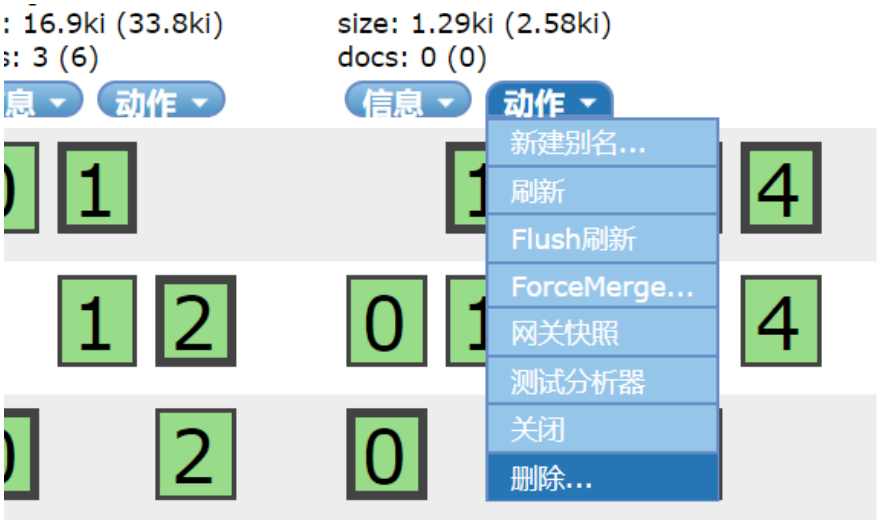
5、删除操作

删除一个document操作

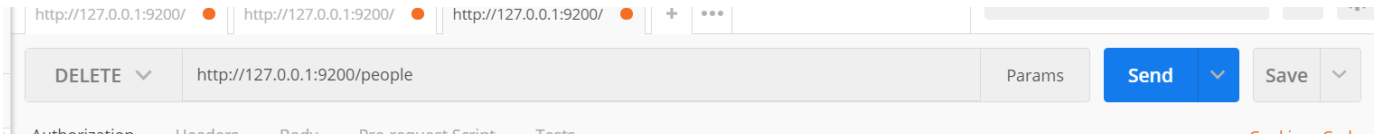
删除 id 为 1 的数据，操作如下：

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：

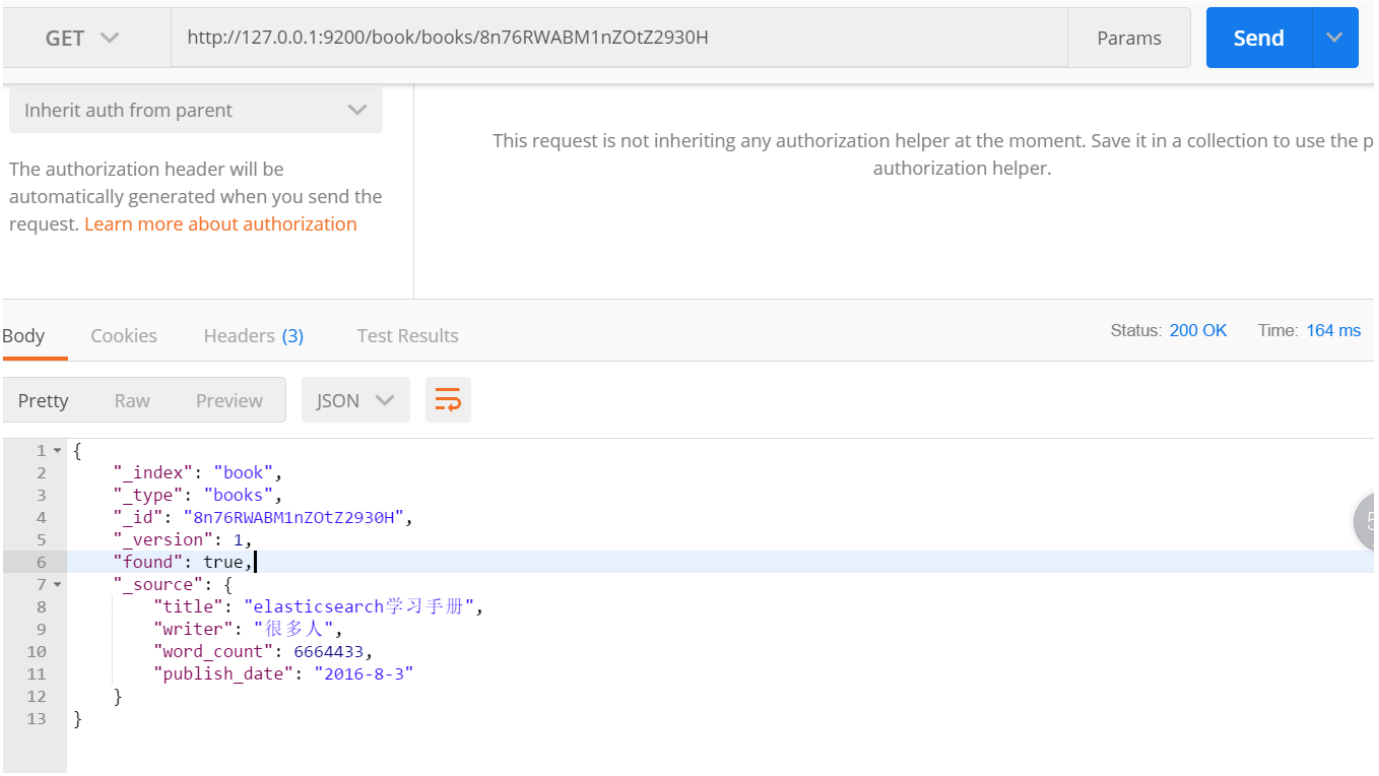
1、简单查询

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：

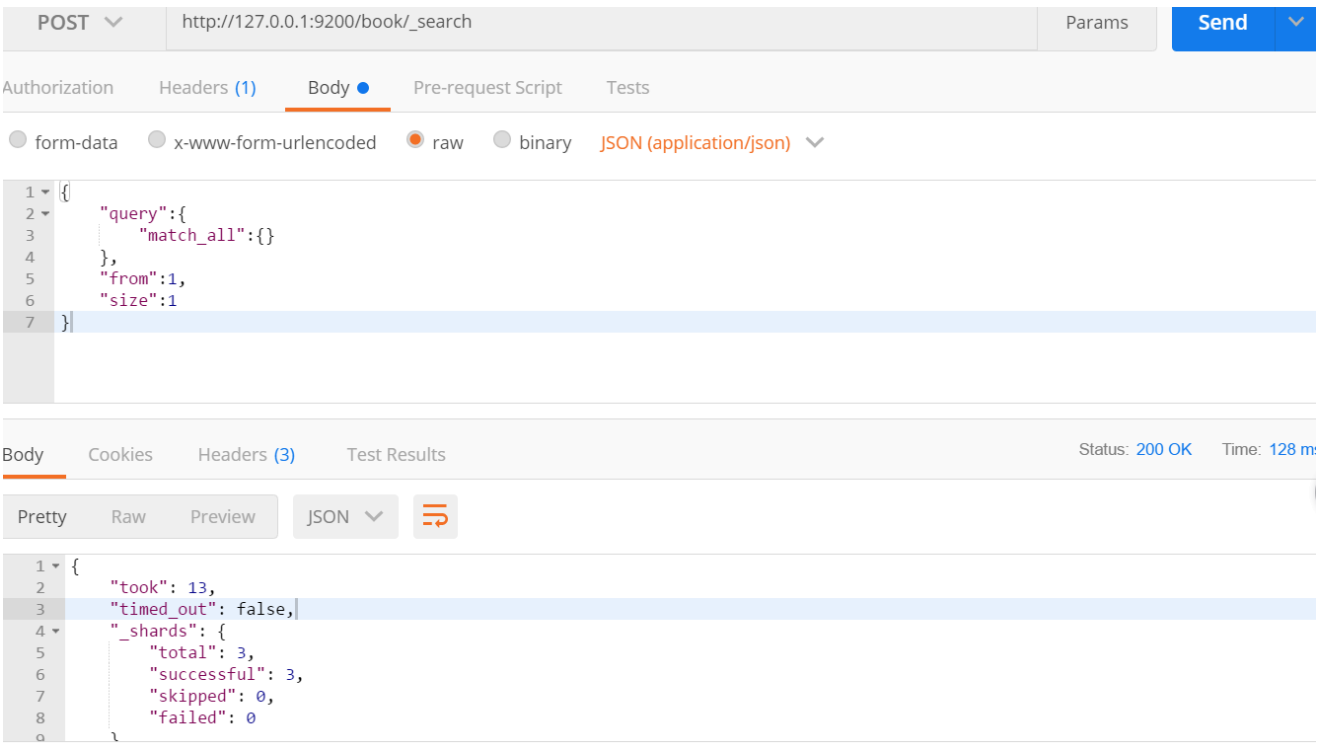
使用Postman 访问 api 接口，Get 127.0.0.1:9200/索引/类型/id



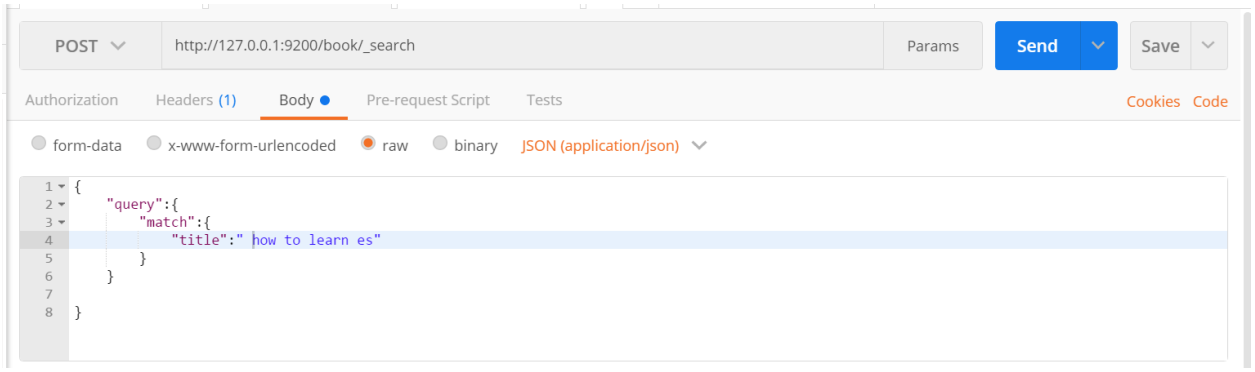
条件查询：

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：



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

```
GET ecommerce/product/_search
{
  "query": {
    "match": {
      "name": "pro"
    }
  },
  "sort": [
    {
      "price":"desc"
    }
  ]
}
```

7、高级查询操作

数据过滤（ query filter)

演示查询 json

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

```
        {
          "match": {
            "name": "xiaomi"
          }
        },
        {
          "match": {
            "desc": "qijian"
          }
        }
      ],
      "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
        "range": {
          "price": {
            "gte": 1000,
            "lte": 5000
          }
        }
      }
    }
  }
}
```

全文检索（full-text search）

演示查询 json

```
GET /ecommerce/product/_search
{
  "query": {
    "match": {
      "desc": "xiaomi qijian shouji"
    }
  }
}
```

返回结果

```
{
  "took": 49,
  "timed_out": false,
  "_shards": {
```

```
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 2,
    "max_score": 0.8630463,
    "hits": [
      {
        "_index": "ecommerce",
        "_type": "product",
        "_id": "1",
        "_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
        "_source": {
          "name": "xiaomi 7",
          "desc": "xiaomi shouji jiushi kuai, niandu qijian",
          "price": 2399,
          "tag": [
            "xiaomi",
            "shouji"
          ]
        }
      },
      {
        "_index": "ecommerce",
        "_type": "product",
        "_id": "3",
        "_score": 0.68324494,
        "_source": {
          "name": "huawei mate 10",
          "desc": "huawei shouji niandu niandu shangwu shouji qijian",
          "price": 4399,
          "tag": [
            "huawei",
            "shouji"
          ]
        }
      }
    ]
  }
}
```

短语搜索（ phrase search ）

根全文检索相对应， 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 要求是输入的搜索串 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

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

结果 json

```
{
  "took": 4,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 2,
    "max_score": 0.2876821,
    "hits": [
      {
        "_index": "ecommerce",
        "_type": "product",
        "_id": "2",
        "_score": 0.2876821,
        "_source": {
          "name": "xiaomi 6",          // 包含 xiaomi
          "desc": "xiaomi6 jiushi kuai!!",
          "price": 2399,
          "tag": [
```

```
            "xiaomi",
            "shouji"
          ]
        }
      },
      {
        "_index": "ecommerce",
        "_type": "product",
        "_id": "1",
        "_score": 0.2876821,
        "_source": {
          "name": "xiaomi 7",         // 包含 xiaomi
          "desc": "xiaomi shouji jiushi kuai, niandu qijian",
          "price": 2399,
          "tag": [
            "xiaomi",
            "shouji"
          ]
        }
      }
    ]
  }
}
```

高亮搜索结果（hightlight search）

演示查询 json

```
GET /ecommerce/product/_search
{
  "query": {
    "match": {
      "name": "xiaomi"
    }
  },
  "highlight": {          // 高亮 field
    "fields": {
      "name": {}
    }
  }
}
```

查询结果


```
{
  "took": 17,
  "timed_out": false,
  "_shards": {
    "total": 5,
    "successful": 5,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": 2,
    "max_score": 0.2876821,
    "hits": [
      {
        "_index": "ecommerce",
        "_type": "product",
        "_id": "2",
        "_score": 0.2876821,
        "_source": {
          "name": "xiaomi 6",
          "desc": "xiaomi6 jiushi kuai!!",
          "price": 2399,
          "tag": [
            "xiaomi",
            "shouji"
          ]
        },
        "highlight": {            // 返回 html 代码 ，可用代码后续处理
          "name": [
            "<em>xiaomi</em> 6"
          ]
        }
      },
      {
        "_index": "ecommerce",
        "_type": "product",
        "_id": "1",
        "_score": 0.2876821,
        "_source": {
          "name": "xiaomi 7",
          "desc": "xiaomi shouji jiushi kuai, niandu qijian",
          "price": 2399,
```

```
          "tag": [
            "xiaomi",
            "shouji"
          ]
        },
        "highlight": {
          "name": [
            "<em>xiaomi</em> 7"
          ]
        }
      }
    ]
  }
}
```