# ElasticSearch学习笔记

## ElasticSearch简介

### 什么是ElasticSearch

ElasticSearch是基于Apache Lucene构建的搜索引擎

使用java编写，提供简单易用的Restful Api

容易横向扩展，支持PB级的结构化与非结构化数据处理

## ElasticSearch应用场景

海量数据分析引擎

站内搜索引擎

数据仓库

## 安装

预置环境  
JDK1.8

Node 8.9

Windows7

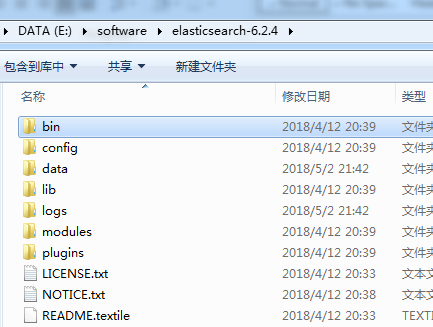
### 单实例安装

第一步：下载

下载地址：<http://www.elastic.co/downloads/elasticsearch>

第二步：解压

解压后目录结构如下：



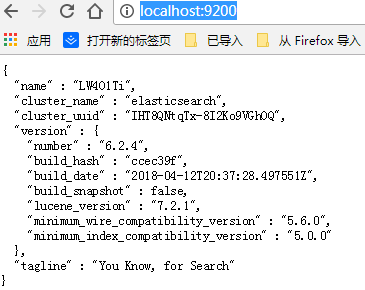
第三步：启动

双击bin目录下的elasticsearch.bat文件启动ElasticSearch

启动成功：如下图.



访问：<http://localhost:9200/> 出现如下画面表示安装成功。



### 插件安装

#### ElasticSearch-head

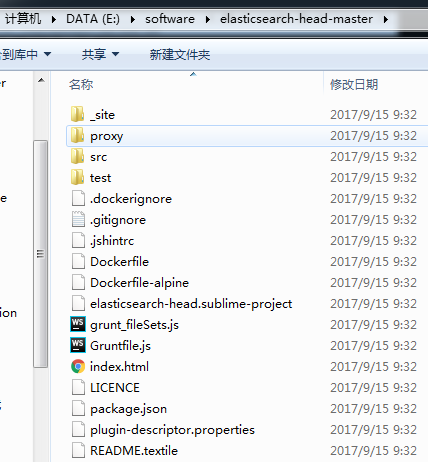
ElasticSearch-head插件是ElasticSearch集群的web前端

第一步：下载

下载地址：<https://github.com/mobz/elasticsearch-head>

第二步：解压

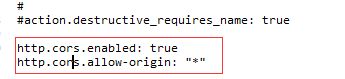
解压后目录结构如下：



第三步：ElasticSearch配置允许跨域

修改ElasticSearch 下的config目录下的elasticsearch.yml文件

配置允许跨域信息：



|  |
| --- |
| http.cors.enabled: true  http.cors.allow-origin: “\*” |

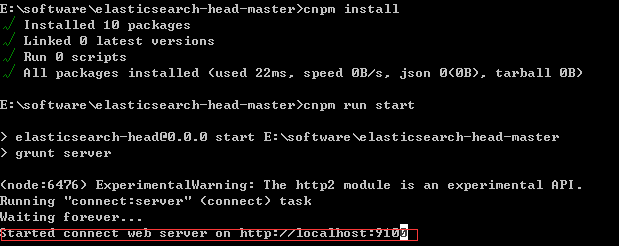
配置好后重启elasticsearch服务

第四步：启动插件

cnpm install

cnpm run start

启动成功：如下图.



访问：<http://localhost:9100> 出现如下画面表示安装成功

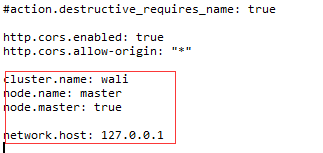


### 分布式安装

#### 配置master节点信息

修改ElasticSearch 下的config目录下的elasticsearch.yml文件

配置master节点信息：

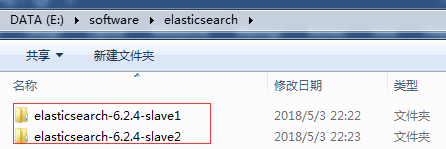


|  |
| --- |
| cluster.name: wali  node.name: master  node.master: true  network.host: 127.0.0.1 |

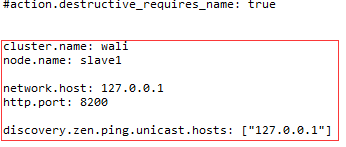
配置好之后，重新启动。

#### 配置随从节点信息

先将解压后的ElasticSearch拷贝两份放到指定目录，命名如下

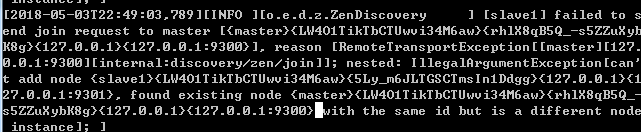


然后配置elasticsearch-6.2.4-slave1的config目录下的elasticsearch.yml文件

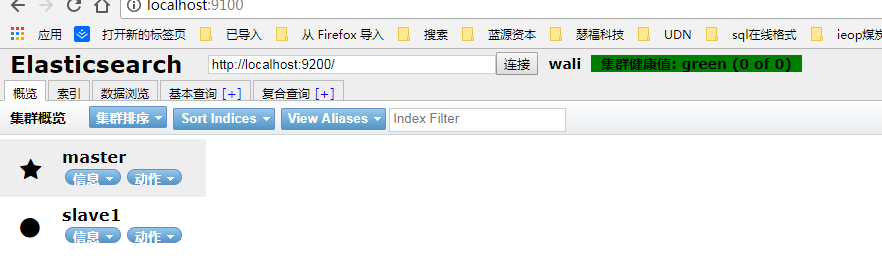


|  |
| --- |
| cluster.name: wali  node.name: slave1  network.host: 127.0.0.1  http.port: 8200  discovery.zen.ping.unicast.hosts: ["127.0.0.1"] |

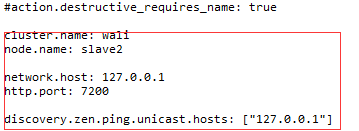
配置好之后启动slave1，如果出现如下错误信息，是直接拷贝的时候，把data文件夹一起给拷贝了，需要把data文件夹删除掉，因为里面包含了节点信息。



启动成功后，访问：<http://localhost:9100> 出现如下画面表示slave1配置成功

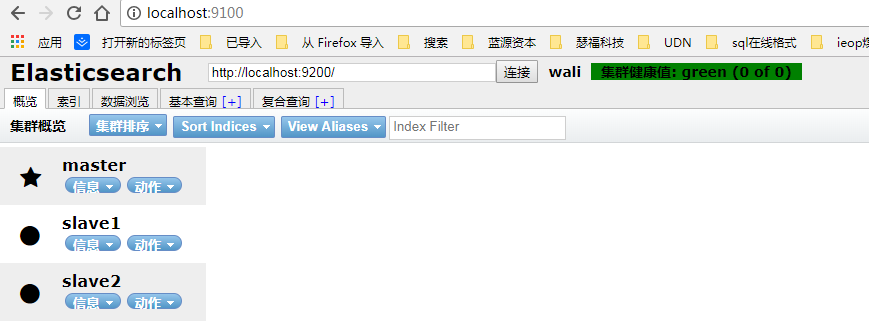


将slave2的配置文件修改如下：



|  |
| --- |
| cluster.name: wali  node.name: slave2  network.host: 127.0.0.1  http.port: 7200  discovery.zen.ping.unicast.hosts: ["127.0.0.1"] |

修改完成后，启动slave2, 启动成功后，访问：<http://localhost:9100> 出现如下画面表示slave1配置成功。



至此可以看到集群有三个节点了。

## 基础概念

### 索引

索引是含有相同属性文档的集合。索引相当于mysql数据库中数据库。

### 类型

索引可以定义一个或多个类型，文档必须属于一个类型。类型相当于mysql数据库中的表。

### 文档

文档是可以被索引的基本数据单位。文档相当于mysql数据库表中的一行记录。

### 分片

每个索引都有多个分片，每个分片是一个Lucene索引。

### 备份

拷贝一份分片就完成了分片的备份。

## 基本用法

ElasticSearch以Restful风格命名API

API基本格式

http://<ip>:<port>/<索引>/<类型>/<文档id>

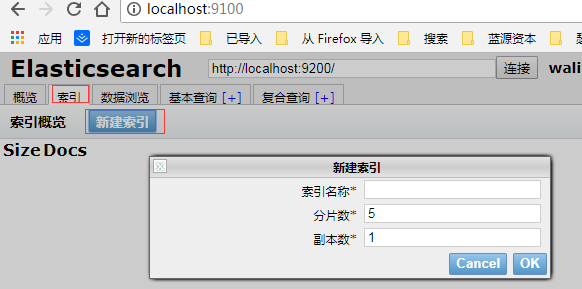
常用http动词 GET/PUT/POST/DELETE

### 创建索引

#### ElasticSearch-head插件创建索引

访问<http://localhost:9100/>

点击索引->新建索引->输入索引名称->点击ok 如下图：创建索引默认会创建5个分片，每个分片创建一个备份



新建成功后，在概览tab页可看到如下信息：



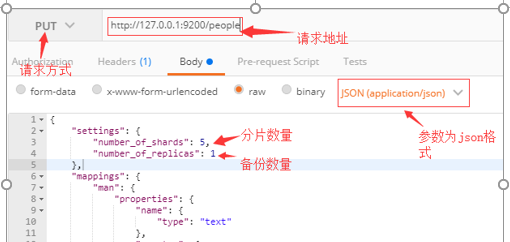
节点旁边的带数字的方块代表分片，其中带粗线边框的方块是主分片，与其数字相同的带细边框的方块是其备份。

#### Postman创建索引

下面使用Postman创建一个名为people的索引。

分片数量：5

备份数量：1



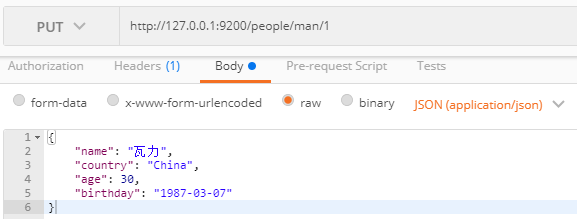
具体请求参数：

|  |
| --- |
| {  "settings": {  "number\_of\_shards": 5,  "number\_of\_replicas": 1  },  "mappings": {  "man": {  "properties": {  "name": {  "type": "text"  },  "country": {  "type": "keyword"  },  "age": {  "type": "integer"  },  "birthday": {  "type": "date",  "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch\_millis"  }  }  }  }  } |

### 新增

使用Postman在索引people的man类型下新增文档

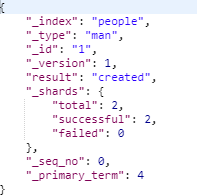
#### 新增并指定文档id



具体请求参数：

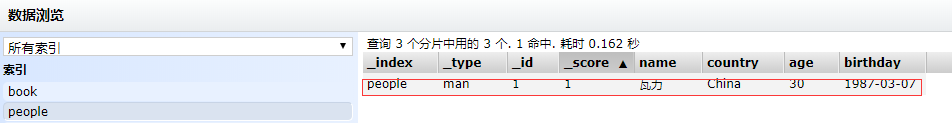
|  |
| --- |
| {  "name": "瓦力",  "country": "China",  "age": 30,  "birthday": "1987-03-07"  } |

出现如下信息表示信息成功



在elasticsearch-head插件中可看到索引people下docs数量变为1，如下图：



点击数据浏览tab页可看到新增的数据，如下图：

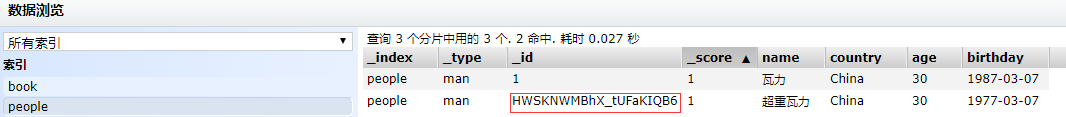
#### 新增自动生成文档id



具体请求参数：

|  |
| --- |
| {  "name": "超重瓦力",  "country": "China",  "age": 30,  "birthday": "1977-03-07"  } |

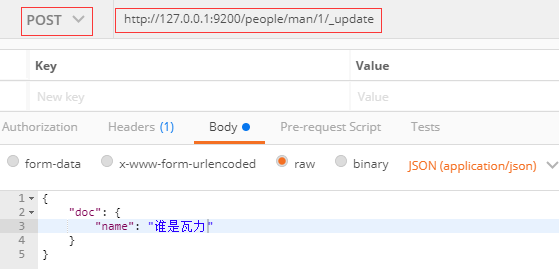
在elasticsearch-head插件中点击数据浏览tab页可看到新增的数据，如下图：



### 修改

#### 直接修改文档

使用Postman修改id为1的文档的name字段为”谁是瓦力”



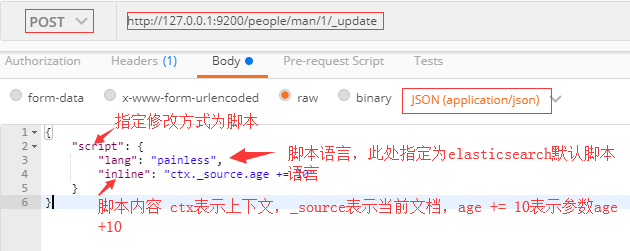
具体参数如下：

|  |
| --- |
| {  "doc": {  "name": "谁是瓦力"  }  } |

修改成功后，在elasticsearch-head插件中点击数据浏览tab页可看到修改后的数据，如下图：



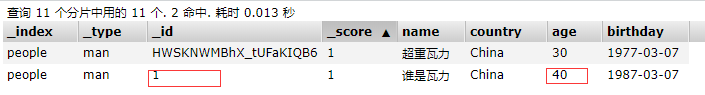
#### 脚本修改文档



具体参数如下：

|  |
| --- |
| {  "script": {  "lang": "painless",  "inline": "ctx.\_source.age += 10"  }  } |

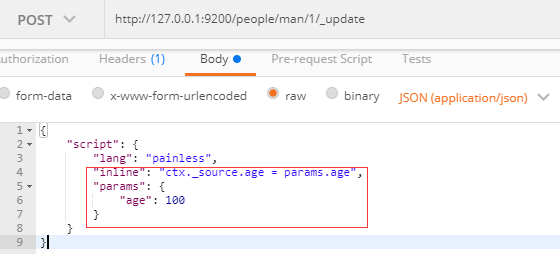
修改成功后，在elasticsearch-head插件中点击数据浏览tab页可看到修改后的数据，如下图：



除了上面这种在inline中传参的方式，还有另一种传参方式。

如将age修改为100

如下图：



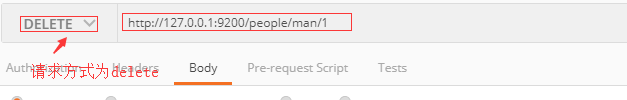
具体参数如下：

|  |
| --- |
| {  "script": {  "lang": "painless",  "inline": "ctx.\_source.age = params.age",  "params": {  "age": 100  }  }  } |

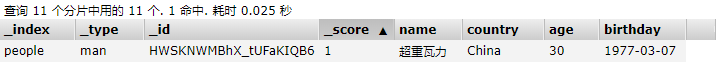
### 删除

#### 删除文档

使用Postman删除索引people中类型为man的id为1的文档



删除成功后，在elasticsearch-head插件中点击数据浏览tab页看到id为1的数据不见了，如下图：



#### 删除索引

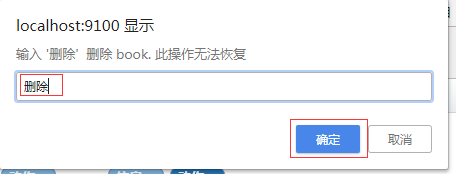
##### ElasticSearch-head插件删除索引

使用ElasticSearch-head插件删除名称为book的索引

如下图所示：点击动作，在弹出的列表中选择删除



在弹出框中输入删除，点击确定，如下图：



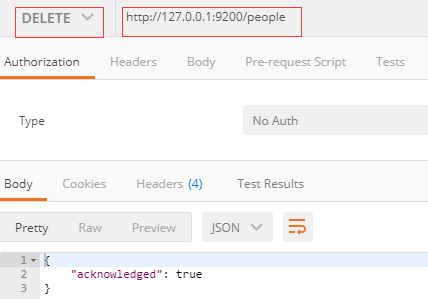
会弹出如下提示框，点击确定，删除成功，



删除成功后，该索引下的类型及类型下文档均会被删除。

##### Postman删除索引

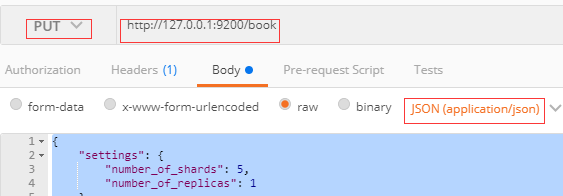
使用Postman删除名称为people的索引



删除索引是一个很危险的操作，应谨慎操作。

### 查询

在查询之前，先使用Postman创建一个名为book的索引。



具体参数如下：

|  |
| --- |
| {  "settings": {  "number\_of\_shards": 5,  "number\_of\_replicas": 1  },  "mappings": {  "novel": {  "properties": {  "word\_count": {  "type": "integer"  },  "author": {  "type": "keyword"  },  "title": {  "type": "text"  },  "publish\_date": {  "type": "date",  "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch\_millis"  }  }  }  }  } |

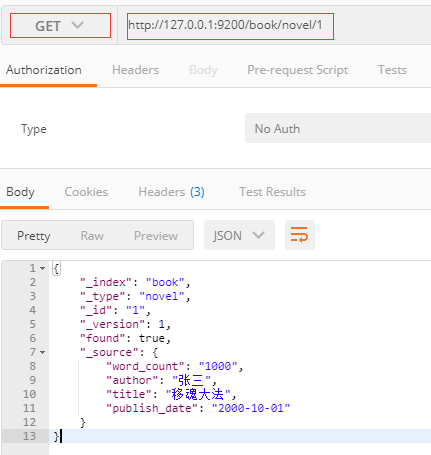
创建好索引之后，再新增如下文档信息。



#### 简单查询

##### 根据id查询

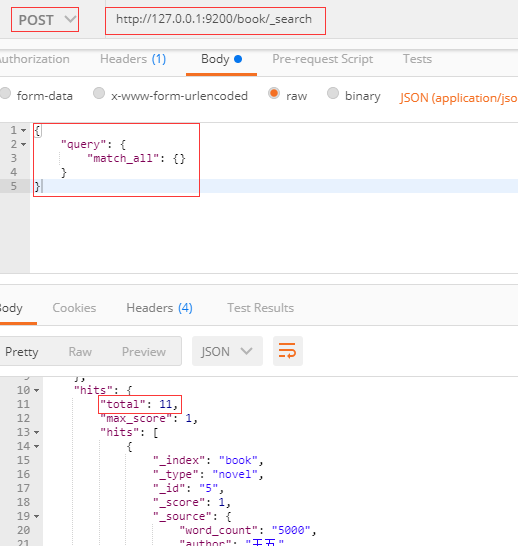
使用Postman查询book索引下类型为novel的id为1的文档



#### 条件查询

##### 查询全部

使用Postman查询book索引下全部文档

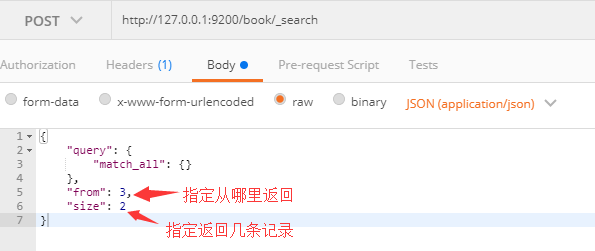


查询参数如下：

|  |
| --- |
| {  "query": {  "match\_all": {}  }  } |

从查询结果中可以看到一共有11条数据，默认返回了10条数据。

##### 指定从哪返回及返回记录数



查询参数如下：

|  |
| --- |
| {  "query": {  "match\_all": {}  },  "from": 3,  "size": 2  } |

##### 根据关键字查询

###### 按照标题查询



查询参数：

|  |
| --- |
| {  "query": {  "match": {  "title": "ElasticSearch"  }  }  } |

返回结果按照\_score字段默认倒序排列。

###### 自定义排序

根据标题查询，返回结果根据publish\_date字段倒序排列。



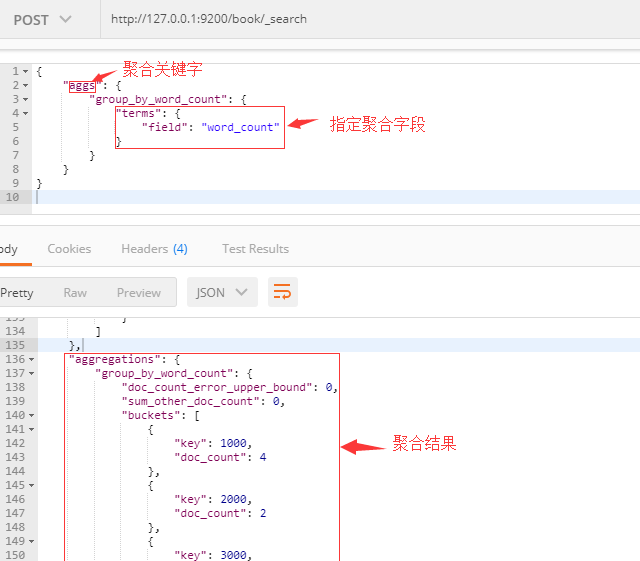
查询参数：

|  |
| --- |
| {  "query": {  "match": {  "title": "ElasticSearch"  }  },  "sort": {  "publish\_date": [{  "order": "desc"  }]  }  } |

#### 聚合查询

##### 单个字段分组聚合

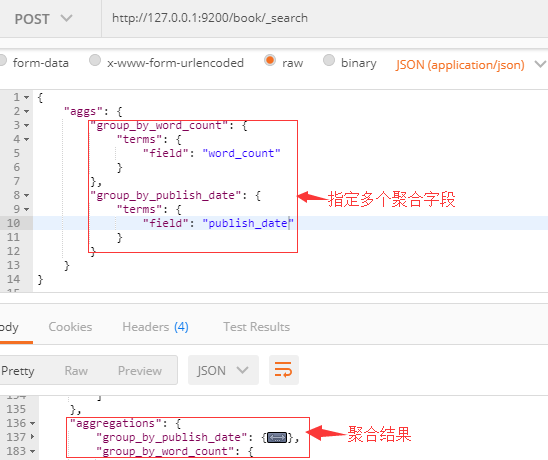
按word\_count字段聚合查询



参数：

|  |
| --- |
| {  "aggs": {  "group\_by\_word\_count": {  "terms": {  "field": "word\_count"  }  }  }  } |

##### 多个字段分组聚合



参数：

|  |
| --- |
| {  "aggs": {  "group\_by\_word\_count": {  "terms": {  "field": "word\_count"  }  },  "group\_by\_publish\_date": {  "terms": {  "field": "publish\_date"  }  }  }  } |

## 高级查询

### 子条件查询

特定字段查询所指特定值

#### Query Context

在查询过程中，处理判断文档是否满足查询条件外，ElasticSearch还回计算一个\_score来标识匹配的程度，旨在判断目标文档和查询条件匹配的有多好。

##### 常用查询

###### 文本查询

文本查询针对文本类型数据

模糊查询

关键字：match

例子：查询author包含”瓦力”的文档



具体参数：

|  |
| --- |
| {  "query": {  "match": {  "author": "瓦力"  }  }  } |

短语匹配

关键字：match\_phrase

例子：查询title包含”ElasticSearch入门”短语的文档



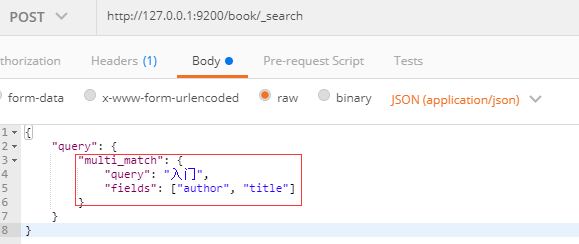
具体参数：

|  |
| --- |
| {  "query": {  "match\_phrase": {  "title": "ElasticSearch入门"  }  }  } |

多个字段查询

关键字：multi\_match

例如：查询title或作者名称为包含“瓦力”的文档



具体参数：

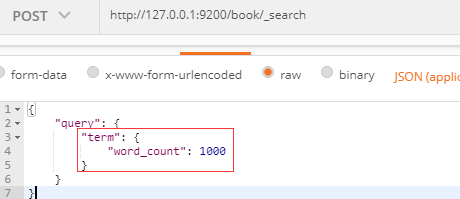
|  |
| --- |
| {  "query": {  "term": {  "word\_count": 1000  }  }  } |

###### 字段级别查询

查询单个字段

关键字：term

例子：查询word\_count为1000的文档



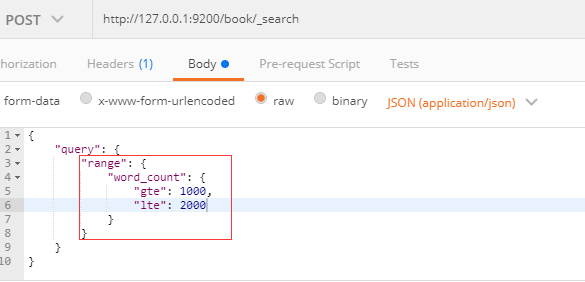
具体参数：

|  |
| --- |
| {  "query": {  "range": {  "word\_count": {  "gte": 1000,  "lte": 2000  }  }  }  } |

范围查询

关键字：range

例子：查询1000<=word\_count<=2000的文档



#### Filter Context

在查询过程中，只判断该文档是否满足条件，只有Yes或者No。

关键字：filter

例子：查询word\_count为1000的文档



具体参数：

|  |
| --- |
| {  "query": {  "bool": {  "filter": {  "term": {  "word\_count": 1000  }  }  }  }  } |

### 复合条件查询

以一定逻辑组合子条件查询

## 实战演练

## 总结

资料：

<https://www.imooc.com/learn/889>

<https://blog.csdn.net/laoyang360/article/details/52244917>