课程大纲

1、query string search

2、query DSL

3、query filter

4、full-text search

5、phrase search

6、highlight search

---------------------------------------------------------------------------------------------------------------------------------

把英文翻译成中文，让我觉得很别扭，term，词项

# query string search

搜索全部商品：GET /ecommerce/product/\_search

took：耗费了几毫秒

timed\_out：是否超时，这里是没有

\_shards：数据拆成了5个分片，所以对于搜索请求，会打到所有的primary shard（或者是它的某个replica shard也可以）

hits.total：查询结果的数量，3个document

hits.max\_score：score的含义，就是document对于一个search的相关度的匹配分数，越相关，就越匹配，分数也高

hits.hits：包含了匹配搜索的document的详细数据

{

"took": 2,

"timed\_out": false,

"\_shards": {

"total": 5,

"successful": 5,

"failed": 0

},

"hits": {

"total": 3,

"max\_score": 1,

"hits": [

{

"\_index": "ecommerce",

"\_type": "product",

"\_id": "2",

"\_score": 1,

"\_source": {

"name": "jiajieshi yagao",

"desc": "youxiao fangzhu",

"price": 25,

"producer": "jiajieshi producer",

"tags": [

"fangzhu"

]

}

},

{

"\_index": "ecommerce",

"\_type": "product",

"\_id": "1",

"\_score": 1,

"\_source": {

"name": "gaolujie yagao",

"desc": "gaoxiao meibai",

"price": 30,

"producer": "gaolujie producer",

"tags": [

"meibai",

"fangzhu"

]

}

},

{

"\_index": "ecommerce",

"\_type": "product",

"\_id": "3",

"\_score": 1,

"\_source": {

"name": "zhonghua yagao",

"desc": "caoben zhiwu",

"price": 40,

"producer": "zhonghua producer",

"tags": [

"qingxin"

]

}

}

]

}

}

query string search的由来，因为search参数都是以http请求的query string来附带的

搜索商品名称中包含yagao的商品，而且按照售价降序排序：GET /ecommerce/product/\_search?q=name:yagao&sort=price:desc

适用于临时的在命令行使用一些工具，比如curl，快速的发出请求，来检索想要的信息；但是如果查询请求很复杂，是很难去构建的

在生产环境中，几乎很少使用query string search

---------------------------------------------------------------------------------------------------------------------------------

# query DSL

DSL：Domain Specified Language，特定领域的语言

http request body：**请求体，可以用json的格式来构建查询语法，比较方便，可以构建各种复杂的语法，比query string search肯定强大多了**

**查询所有的商品**

GET /ecommerce/product/\_search

{

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

}

查询名称包含yagao的商品，同时按照价格降序排序

GET /ecommerce/product/\_search

{

"query" : {

"match" : {

"name" : "yagao"

}

},

"sort": [

{ "price": "desc" }

]

}

分页查询商品，总共3条商品，假设每页就显示1条商品，现在显示第2页，所以就查出来第2个商品

GET /ecommerce/product/\_search

{

"query": { "match\_all": {} },

"from": 1,

"size": 1

}

指定要查询出来商品的名称和价格就可以

GET /ecommerce/product/\_search

{

"query": { "match\_all": {} },

"\_source": ["name", "price"]

}

更加适合生产环境的使用，可以构建复杂的查询

---------------------------------------------------------------------------------------------------------------------------------

# query filter

搜索商品名称包含yagao，而且售价大于25元的商品

GET /ecommerce/product/\_search

{

"query" : {

"bool" : {

"must" : {

"match" : {

"name" : "yagao"

}

},

"filter" : {

"range" : {

"price" : { "gt" : 25 }

}

}

}

}

}

---------------------------------------------------------------------------------------------------------------------------------

# full-text search（全文检索）

GET /ecommerce/product/\_search

{

"query" : {

"match" : {

"producer" : "yagao producer"

}

}

}

尽量，无论是学什么技术，比如说你当初 学java，学linux，学shell，学javascript，学hadoop。。。。一定自己动手，特别是**手工敲各种命令和代码，切记切记，减少复制粘贴的操作。只有自己动手手工敲，学习效果才最好。**

producer这个字段，会先被拆解，建立倒排索引

special 4

yagao 4

producer 1,2,3,4

gaolujie 1

zhognhua 3

jiajieshi 2

yagao producer ---> yagao和producer

{

"took": 4,

"timed\_out": false,

"\_shards": {

"total": 5,

"successful": 5,

"failed": 0

},

"hits": {

"total": 4,

"max\_score": 0.70293105,

"hits": [

{

"\_index": "ecommerce",

"\_type": "product",

"\_id": "4",

"\_score": 0.70293105,

"\_source": {

"name": "special yagao",

"desc": "special meibai",

"price": 50,

"producer": "special yagao producer",

"tags": [

"meibai"

]

}

},

{

"\_index": "ecommerce",

"\_type": "product",

"\_id": "1",

"\_score": 0.25811607,

"\_source": {

"name": "gaolujie yagao",

"desc": "gaoxiao meibai",

"price": 30,

"producer": "gaolujie producer",

"tags": [

"meibai",

"fangzhu"

]

}

},

{

"\_index": "ecommerce",

"\_type": "product",

"\_id": "3",

"\_score": 0.25811607,

"\_source": {

"name": "zhonghua yagao",

"desc": "caoben zhiwu",

"price": 40,

"producer": "zhonghua producer",

"tags": [

"qingxin"

]

}

},

{

"\_index": "ecommerce",

"\_type": "product",

"\_id": "2",

"\_score": 0.1805489,

"\_source": {

"name": "jiajieshi yagao",

"desc": "youxiao fangzhu",

"price": 25,

"producer": "jiajieshi producer",

"tags": [

"fangzhu"

]

}

}

]

}

}

---------------------------------------------------------------------------------------------------------------------------------

# phrase search（短语搜索）

跟全文检索相对应，相反，全文检索会将输入的搜索串拆解开来，去倒排索引里面去一一匹配，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，才可以算匹配，才能作为结果返回**

GET /ecommerce/product/\_search

{

"query" : {

"match\_phrase" : {

"producer" : "yagao producer"

}

}

}

{

"took": 11,

"timed\_out": false,

"\_shards": {

"total": 5,

"successful": 5,

"failed": 0

},

"hits": {

"total": 1,

"max\_score": 0.70293105,

"hits": [

{

"\_index": "ecommerce",

"\_type": "product",

"\_id": "4",

"\_score": 0.70293105,

"\_source": {

"name": "special yagao",

"desc": "special meibai",

"price": 50,

"producer": "special yagao producer",

"tags": [

"meibai"

]

}

}

]

}

}

---------------------------------------------------------------------------------------------------------------------------------

# highlight search（高亮搜索结果）

GET /ecommerce/product/\_search

{

"query" : {

"match" : {

"producer" : "producer"

}

},

"highlight": {

"fields" : {

"producer" : {}

}

}

}