# 使用DSL处理搜索结果

我们已经能够利用DSL语法从海量数据中检索出你想要的数据，但数据查询之后往往还需要对搜索的结果进行处理：排序、分页、高亮。

所以接下来我们就来学习ES当中的搜索结果处理！主要就是排序、分页、高亮！

## 排序

ES支持对搜索结果排序，即ES默认是有排序的，即默认是根据相关度算分(\_score)来排序：分值越高排名越靠前，但是有的时候我们希望按照别的东西去做排序，比如价格、日期等，所以**ES是允许你去指定排序字段的，一旦自己制定了排序字段，ES就会放弃默认的打分，即可以在搜索结果看到\_score为null了，这样，查询效率也会有一定的提升。**ES支持可以排序的字段类型有：keyword类型、数值类型、地理坐标类型、日期类型等。其实数值类型和日期类型是用得最多的。

因为排序字段的数据类型不一样，所以DSL语法也会有一定的差异：其中keyword、数值、日期这三种排序的语法其本上是一样的，如下：

### keyword、数值、日期这三种常规字段的通用DSL语法：

**GET /索引库名/\_search**

**{**

**“query”: {**

**“match\_all”: {}**

**},**

**“sort”: [**

**{**

**“FIELD”: “des” // 排序字段和排序方式ASC,DESC**

**}**

**]**

**}**

可以看到多了一个和“query”同级的“sort”。“query”是我们上一节所讲的，上一章所讲的所有搜索DSL都是写在“query”内部的！而我们这一章讲的是搜索结果处理，所以是和“query”是同级的关系！

可以看到“sort”是数组，意味着将来你可以指定多个排序字段，就像我们MySQL里面，MySQL中的order by不是可以指定多个吗！示例指定了一个，你还可以继续指定，如果多个排序字段的话，那么是先按照第一个字段排序，第一个字段相等再按第二个字段排序，这一点跟MySQL也是一样的！

### geo地理坐标字段排序通用DSL语法

地理坐标类型排序的DSL就相对复杂了一丢丢：

**GET /索引库名/\_search**

**{**

**“query”: {**

**“match\_all”: {}**

**},**

**“sort”: [**

**{**

**“\_geo\_distance”: {**

**“FIELD”: “维度, 经度”,**

**“order”: “asc”,**

**“unit”: “km”**

**}**

**}**

**]**

**}**

### 排序案例一：

需求：对酒店数据按照用户评价降序排序，评价相同的按照价格升序排序

解读；要排序的在自动断就两个嘛：评价 & 价格

#### DSL语句如下：注意，socre写法可以简化，详情件price！

**GET /hotel/\_search**

**{**

**"query": {**

**"match\_all": {}**

**},**

**"sort": [**

**{**

**"score": {**

**"order": "desc"**

**}**

**},**

**{**

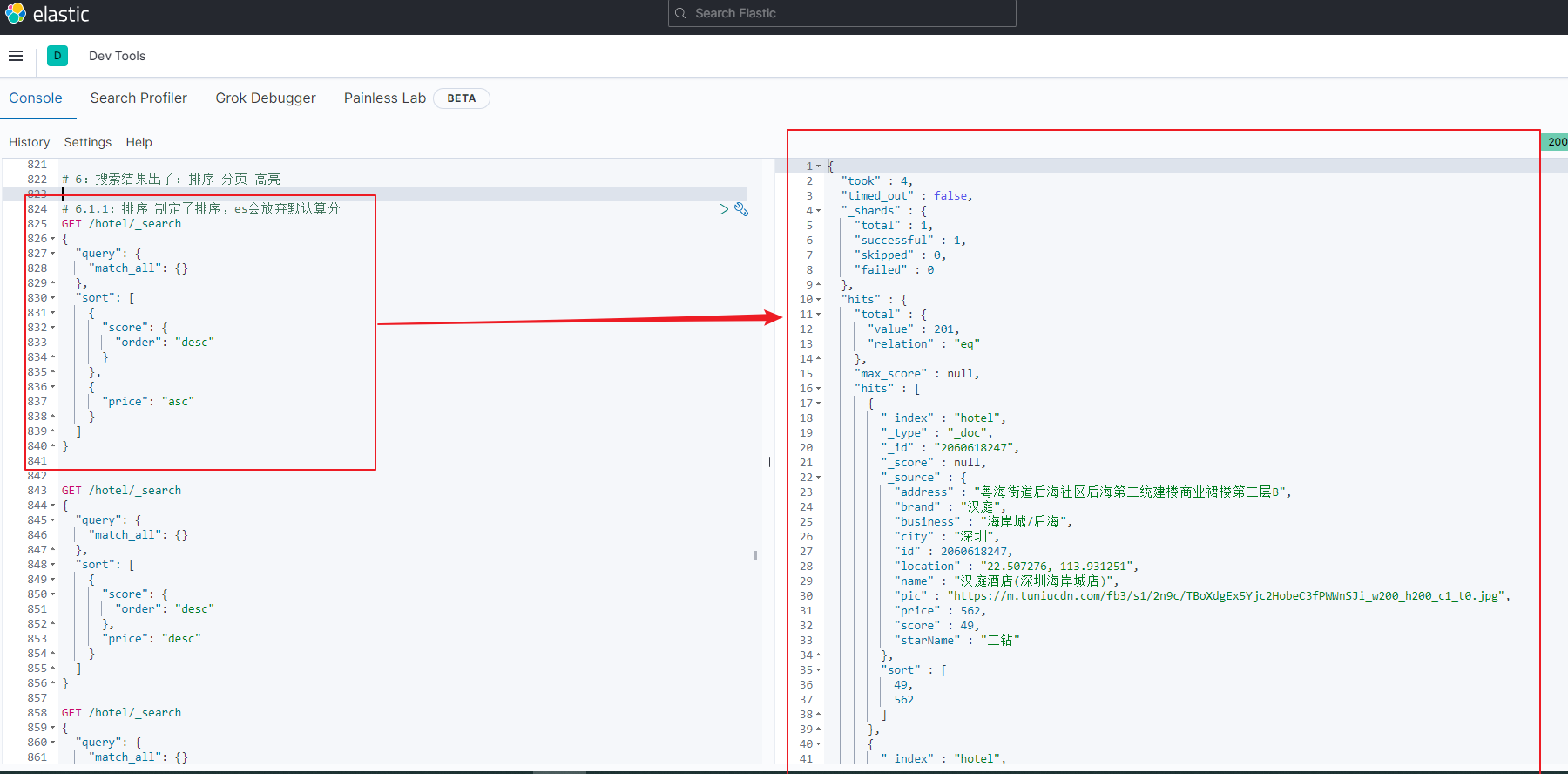
**"price": "asc"**

**}**

**]**

**}**

#### DevTools控制台查看搜索结果：可以看到排第一的酒店评分相当高，居然有4.9分呢！



### 排序案例二

需求：实现对酒店数据按照到你的位置坐标的距离升序排序。



因此你先要知道你的坐标啊，以下是一个工具网站，可以获取地理坐标：

<https://lbs.amap.com/demo/jsapi-v2/example/map/click-to-get-lnglat/>

#### DSL语句如下：

地理坐标可以按以下方式写，当然也可以按照通用方法演示的那样写！

**GET /hotel/\_search**

**{**

**"query": {**

**"match\_all": {}**

**},**

**"sort": [**

**{**

**"\_geo\_distance": {**

**"location": {**

**"lat": 31.034661,**

**"lon": 121.612282**

**},**

**"order": "asc",**

**"unit": "km"**

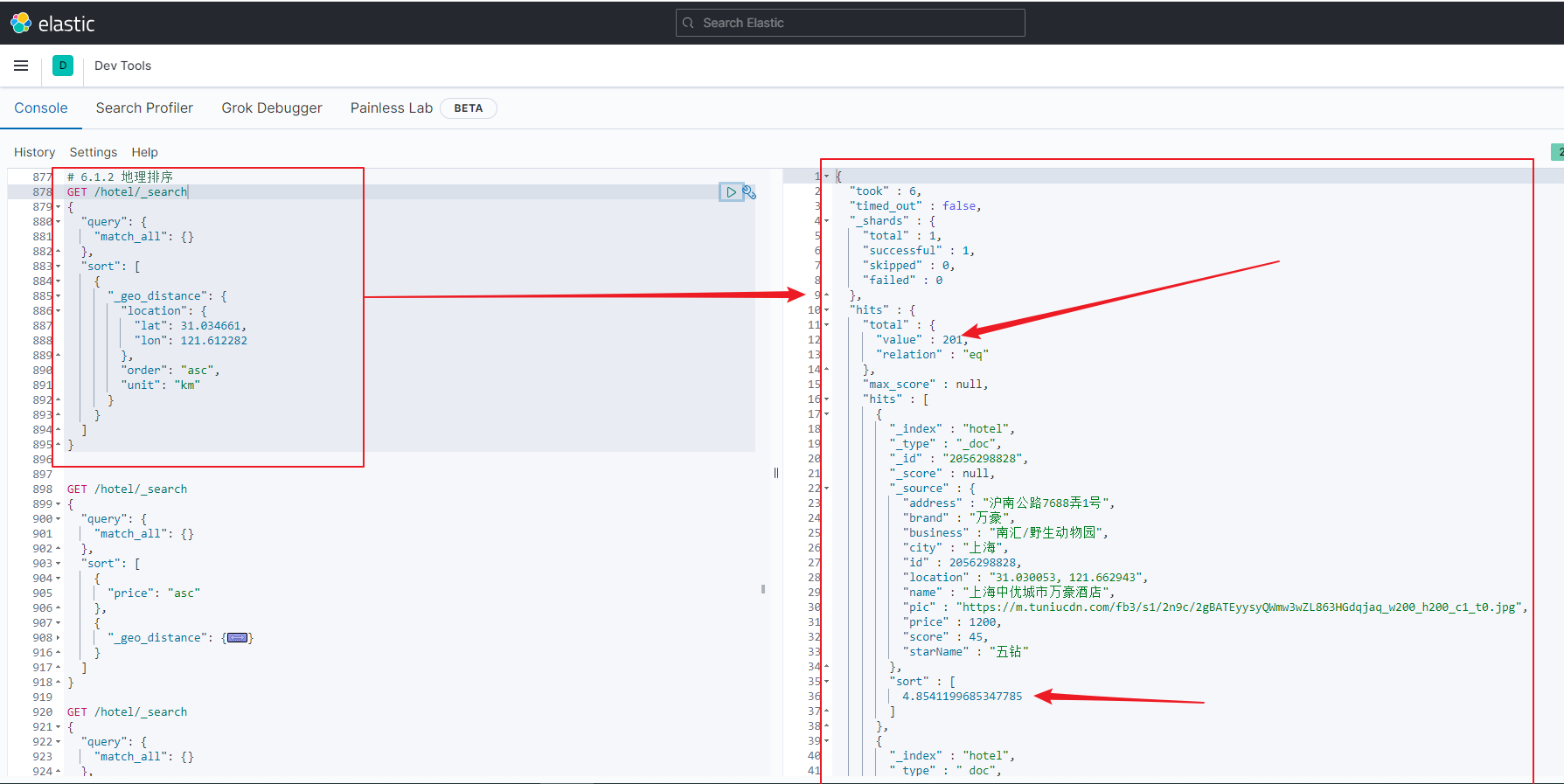
**}**

**}**

**]**

**}**

#### DevTools控制台搜索并查看结果，可以看到，搜索结果给出了sort字段，就是你排序的值，可以看到第一条数据的酒店距离你4.8公里左右！



### 排序案例总结

需要注意的是，根据案例一与案例二的测试截图，可以看到一旦做了排序，文档的相关新打分就没有意义了，就不会去给文档做相关性算分，ES就会放弃去算分，自然搜索结果中的\_score就没有值了。

## 分页

之前搜索的时候，我们就发现了，ElasticSearch不管我们搜索到的结果有多少条，默认情况下只返回top10的数据，这其实就是因为**ES的底层有一个默认的分页参数限制了返回的结果数量。也就是说如果你要修改或者要看到更多的分页结果，就需要去修改这些默认的分页参数了**。而如果要查询更多数据就需要修改分页参数了。

ES中通过修改Form、size参数来控制要返回的分页结果。

### 分页DSL语句示例

**GET /索引库名/\_search**

**{**

**“query”: {**

**“match\_all”: {}**

**},**

**“form”: 990, // 分页开始的位置，默认为0**

**“size”: 10, // 期望获取的文档总数**

**“sort”: [**

**{“price”: “asc”}**

**]**

**}**

如上，修改from和size的值就能控制要返回的分页结果！你可能会问问什么要加排序，傻瓜，加了排序不是更容易看出分页的效果吗！

#### DSL语句如下：返回前20条数据

**GET /hotel/\_search**

**{**

**"query": {**

**"match\_all": {}**

**},**

**"from": 0,**

**"size": 20,**

**"sort": [**

**{**

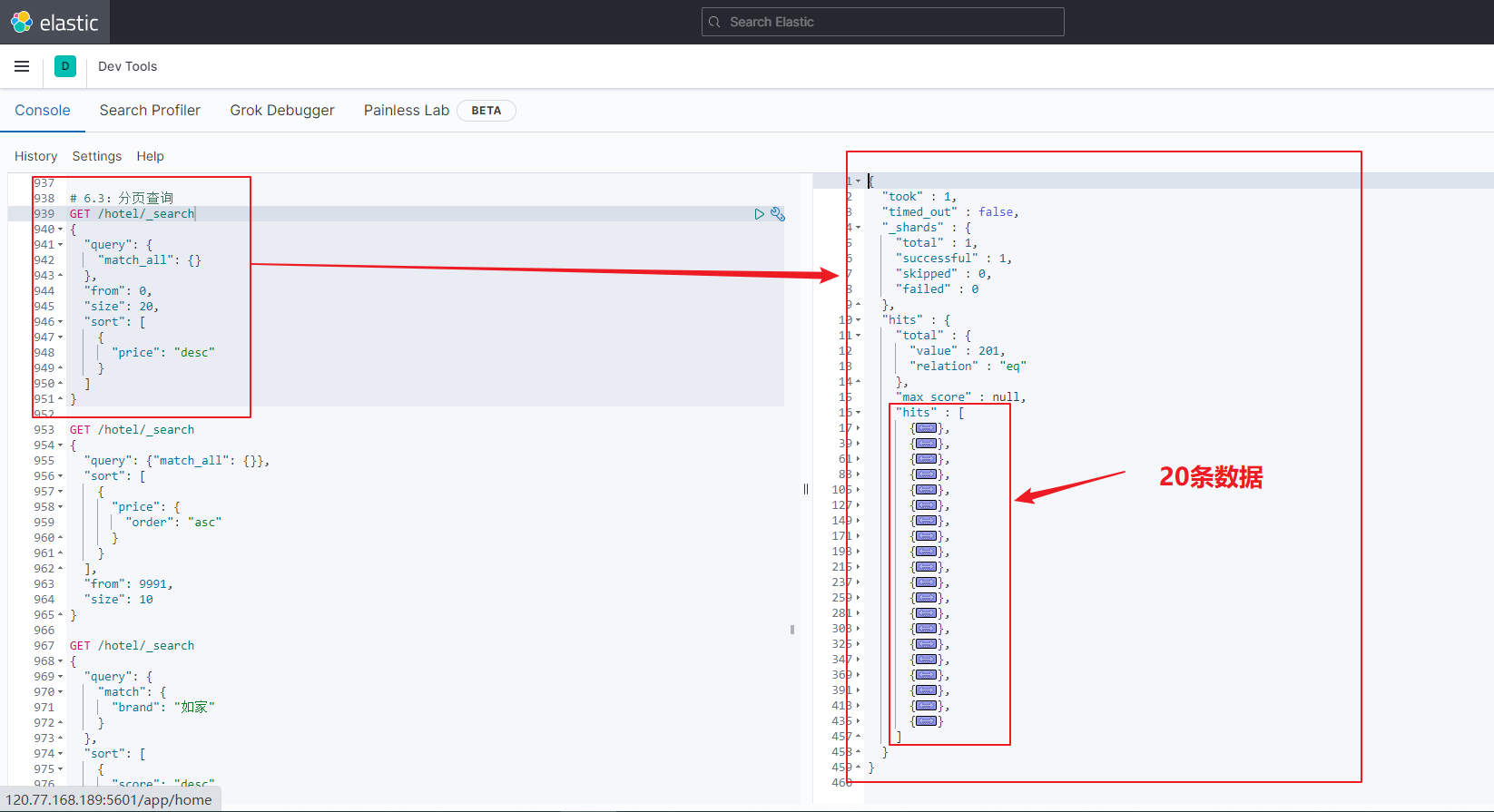
**"price": "desc"**

**}**

**]**

**}**

#### DevTools控制台查看搜索结果



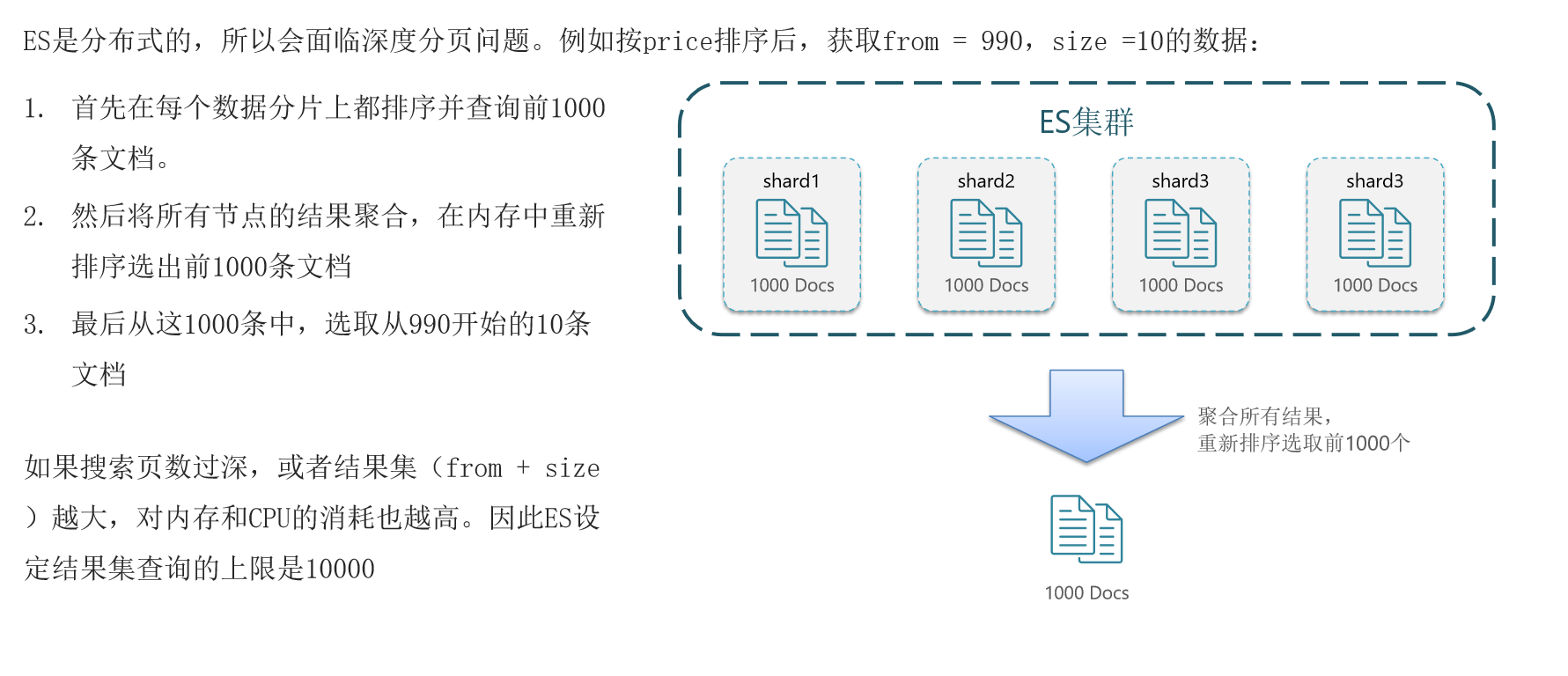
### 深度分页问题

ES是分布式的，所以会面临深度分页问题！

上面是不是分页就学完了呢？并不是！ES分页虽然和MySQL很像，都是指定from和size就行了，非常简单，但是ES和MySQL的底层原理差别是很大的。

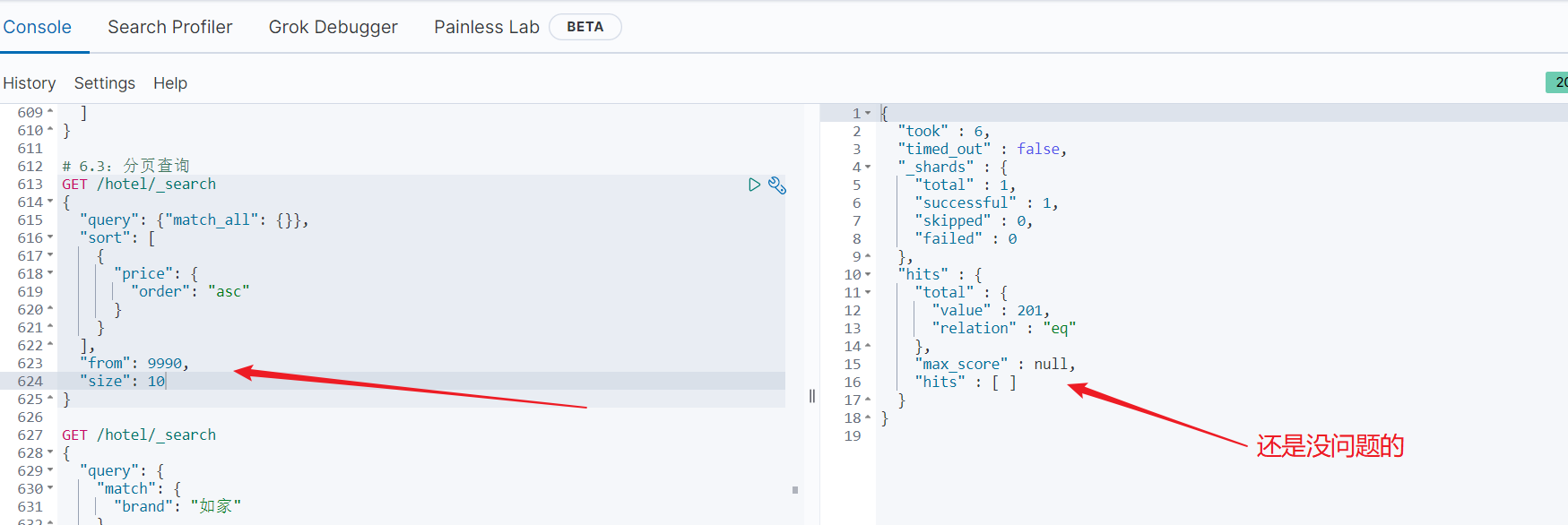
ES因为底层采用的是倒排索引，倒排索引结构其实是不利于做分页的，所以**ES采用的是逻辑上的分页，比方说，你如果要查从990到1000这十条数据，对ES来说只能是查出从0到1000的数据，然后去截990到1000的这一部分，这是由倒排索引的数据结构决定的，没有办法。**

那你可能会说这么做也没什么问题啊！是的，这么做在单点查询的时候没什么问题！但是我们做的是集群呢？我们在实际生产环境下，为了让ES能存储更多的数据，一定会做ES的集群，ES天生就支持集群，而一旦做了ES集群，ES就会对数据进行拆分，放到不同的机器上，拆分出的每一份我们叫分片，也就是说每一片的数据不一样，那现在如果要查询前1000条数据，那么就要在每一个分片中查询出前1000条数据，然后将所有节点的结果聚合，在内存中重新排序选取前1000个。

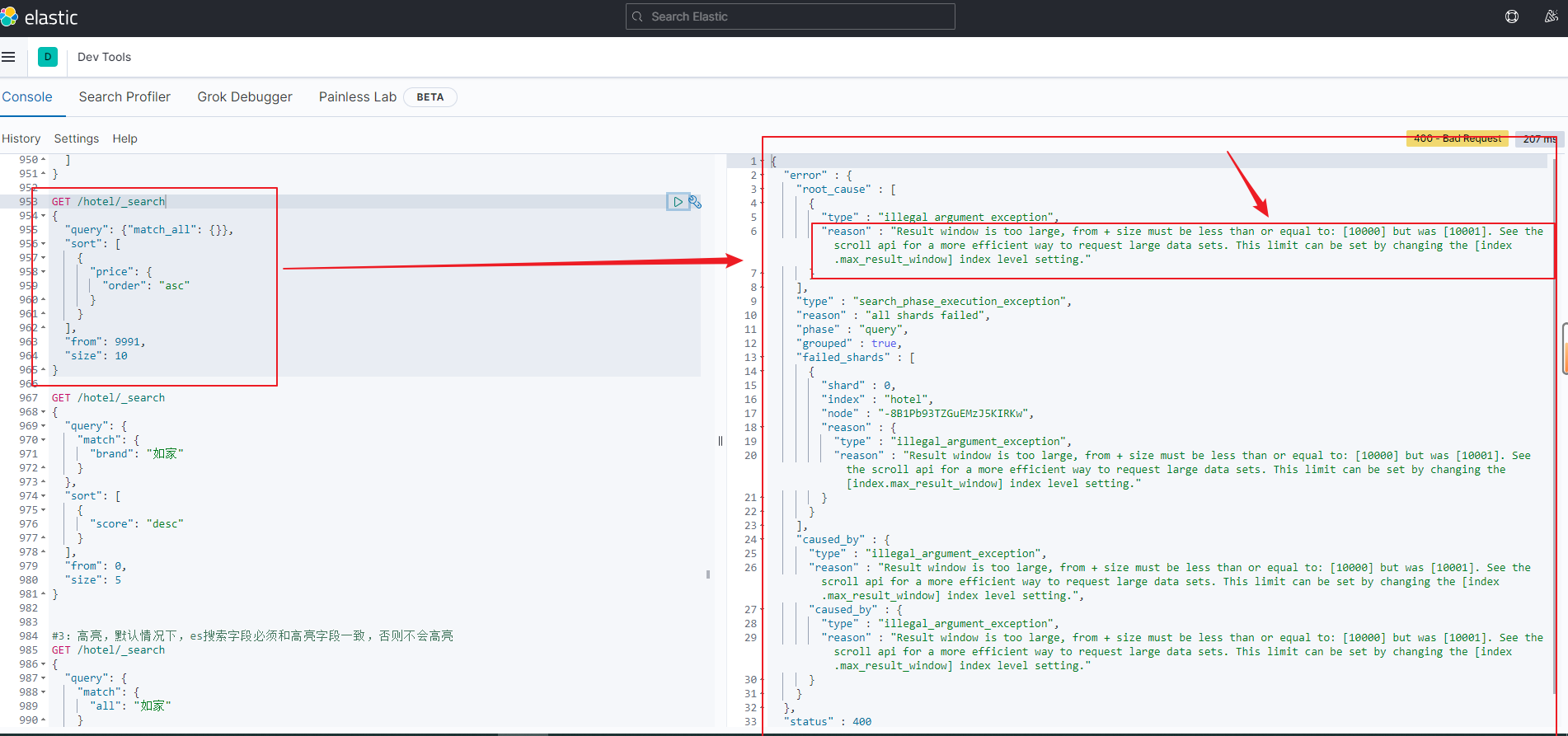


乍一看好像没什么问题，那是因为在你潜意识中的ES分片数量不多，但是真实企业开发中，比如百度的ES分片分分钟成千上万台，比如有5000台ES，如果每台1000条，意味着我要在每一台上截100条数据，然后1000X5000，就是总共五百万条数据，就是要把五百万条数据在内存中重新做排序截取前1000条，这样问题就大了，内存的消耗就很大了，所以说如果搜索的页数过深、或者结果集（form + size）越大，堆内存和CPU的消耗也越高。因此ES设定结果集查询的上限是10000，即from+size之和不能超过一万。

不信你看，10000以内还是没问题的：



超过1万就报错了：



### 深度分页问题解决方案

那我就想查超过10000条数据怎么办？

其实事实上实际中会从业务层面禁止你查10000条数据，不信你可以看百度、淘宝、京东他们的搜索分页都是有限制的，以百度为例，百度搜索结果的分页最多只能到第70多页，每页大概10条，也就是说最多只能查几百条数据，京东也一样，京东最多只能让你查100页。所以面临深度分页的问题，你要从根源上杜绝，就是在业务上不让你查那么多。

但是我真的有深度分页的需求要怎么办？ES也给出了深度分页的解决方案，针对深度分页，ES提供了两种解决方案：

**Search after：**分页时需要排序，原理是从上一次排序值开始，查询下一页数据。官方推荐使用的方式；这一种方式有个缺点，因为你是知道了上一次查询最后一条数据的排序值，然后往后查，也就是说翻页只能往后翻，不能往前翻。

**Scroll：**原理是将排序数据形成快照，保存在内存。官方已经不推荐使用。

### 分页总结

#### from + size：

优点：支持随机翻页

缺点：深度分页问题，默认查询上限（from + size）是10000

场景：百度、京东、谷歌、淘宝这样的随机翻页搜索

#### after search：

优点：没有查询上限（单次查询的size不超过10000）

缺点：只能向后逐页查询，不支持随机翻页

场景：没有随机翻页需求的搜索，例如手机向下滚动翻页

#### scroll：

优点：没有查询上限（单次查询的size不超过10000）

缺点：会有额外内存消耗，并且搜索结果是非实时的

场景：海量数据的获取和迁移。从ES7.1开始不推荐，建议用 after search方案。

## 高亮

### 高亮显示原理

接下来学习ElasticSearch对于搜索结果的高亮处理，什么是高亮？**就是在搜索结果中把搜索关键字突出显示。高亮其实就是给关键字（用户搜什么）加上标签，而后写上样式。**

比如我在百度中搜索java，你会发现搜索结果中所有的java都变成了红色，而其他字体都是正常的黑色，这就是高亮显示；



问题是，这个标签是谁加上去的，要知道，ES在搜索时，默认情况下结果是没有带标签的，都是正常的文本。显然也不是前端加的，因为搜索结果是会变化的，搜索条件也是会变化的，今天搜java，明天搜python，后天搜c++，搜索的内容不一样，将来需要高亮的部分也不一样，前端不可能提前预知、提前写好。即搜索条件是动态的，结果也是动态的，结果中需要高亮的部分也是动态的，即应该是返回搜索结果的人来完成高亮处理，完成高亮标签的添加：即服务端，即ElasticSearch。ES在返回搜索结果的时候，应该给这些关键字加上标签，这样一来，前端只需要干一件事，就是写好CSS样式，渲染页面时自然就高亮了。

所以，高亮显示原理是这样的：

1：ES服务端将搜索结果中的关键字用标签标记起来；

2：前端在页面中给标签添加CSS样式

### 问题：ElasticSearch怎么怎么样给关键字加上标签呢？

这个动作不需要你去做，你只需要告诉ES你要给那个字段加标签，加什么标签就行了，ES就会自动帮你完成加标签的动作，DSL语法如下：

**GET /索引库名/\_search**

**{**

**“query”: {**

**“match”: {**

**“FIELD”: “TEXT”**

**}**

**},**

**“highlight”: {**

**“fields”: { // 指定要高亮的字段**

**“FIELD”: {**

**“pre\_tags”: “<em>”, // 用来标记高亮字段的前置标签**

**“post\_tags”: “</em>” // 用来标记高亮字段的前置标签**

**}**

**}**

**}**

**}**

可以看到：

Query：还是简单的match全文检索查询，注意了，要做搜素高亮显示，搜索类型一定不能使用match\_all或者事其他类型的查询，一定要使用带关键字的查询，即match查询或者term查询；

Highlight：就是高亮处理，可以看到高亮highlight和query同级；

Fields：中指定高亮的字段，fields是数组，证明将来可以高亮的字段不止一个！为什么呢？你想啊，比如我们将来去百度搜索java的时候，网页里的结果有好几部分，一是标题，二是网页介绍。。。所以说你将来可以指定多个字段去做高亮！

pre\_tags/post\_tags：默认就是<em></em>，所以可以不写

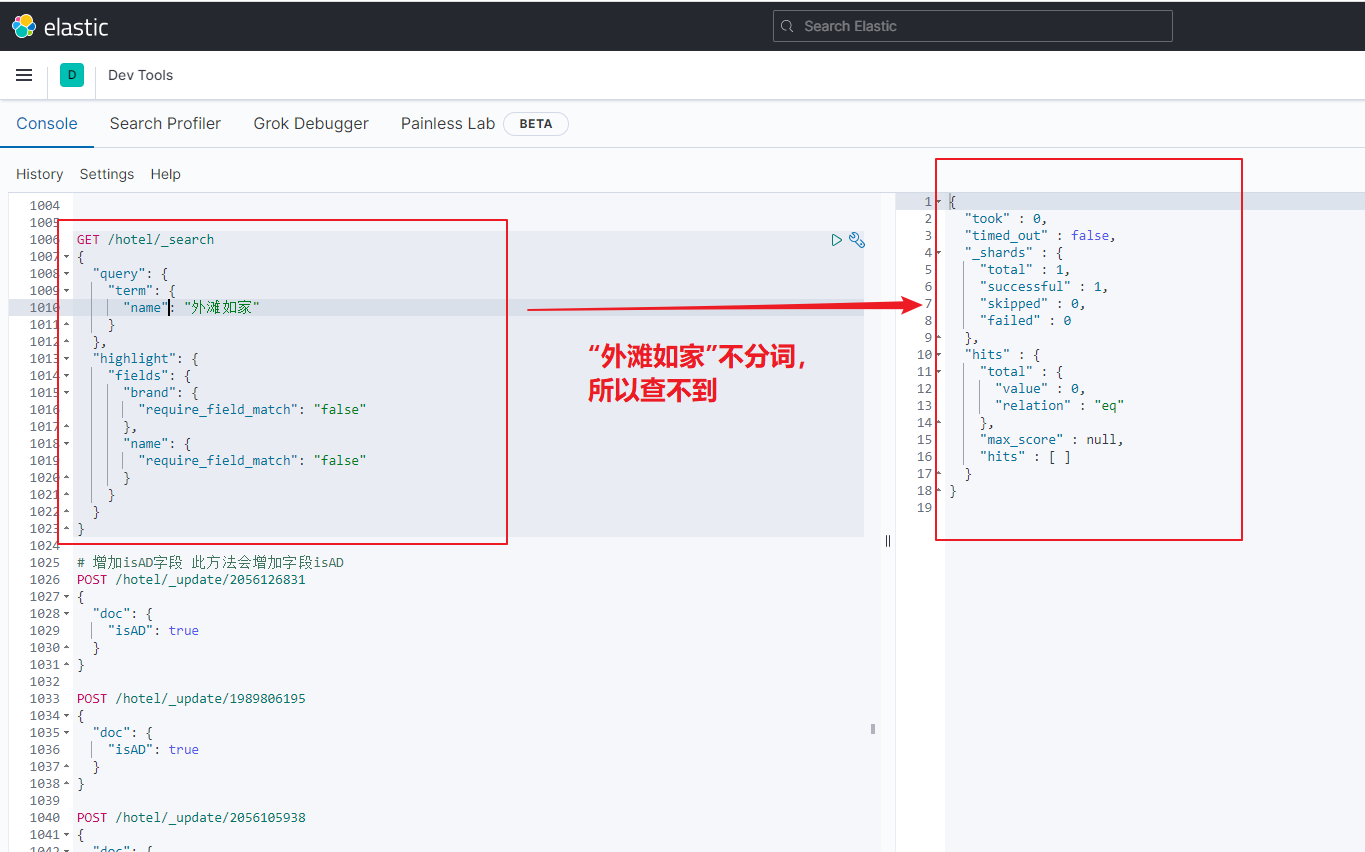
### 注意点（很重要很重要很重要）

1：query是按照搜索条件搜索出内容，highlights是对搜索内容的对应字段加高亮！可以将query和highlights分看看待！看成是两阶段！

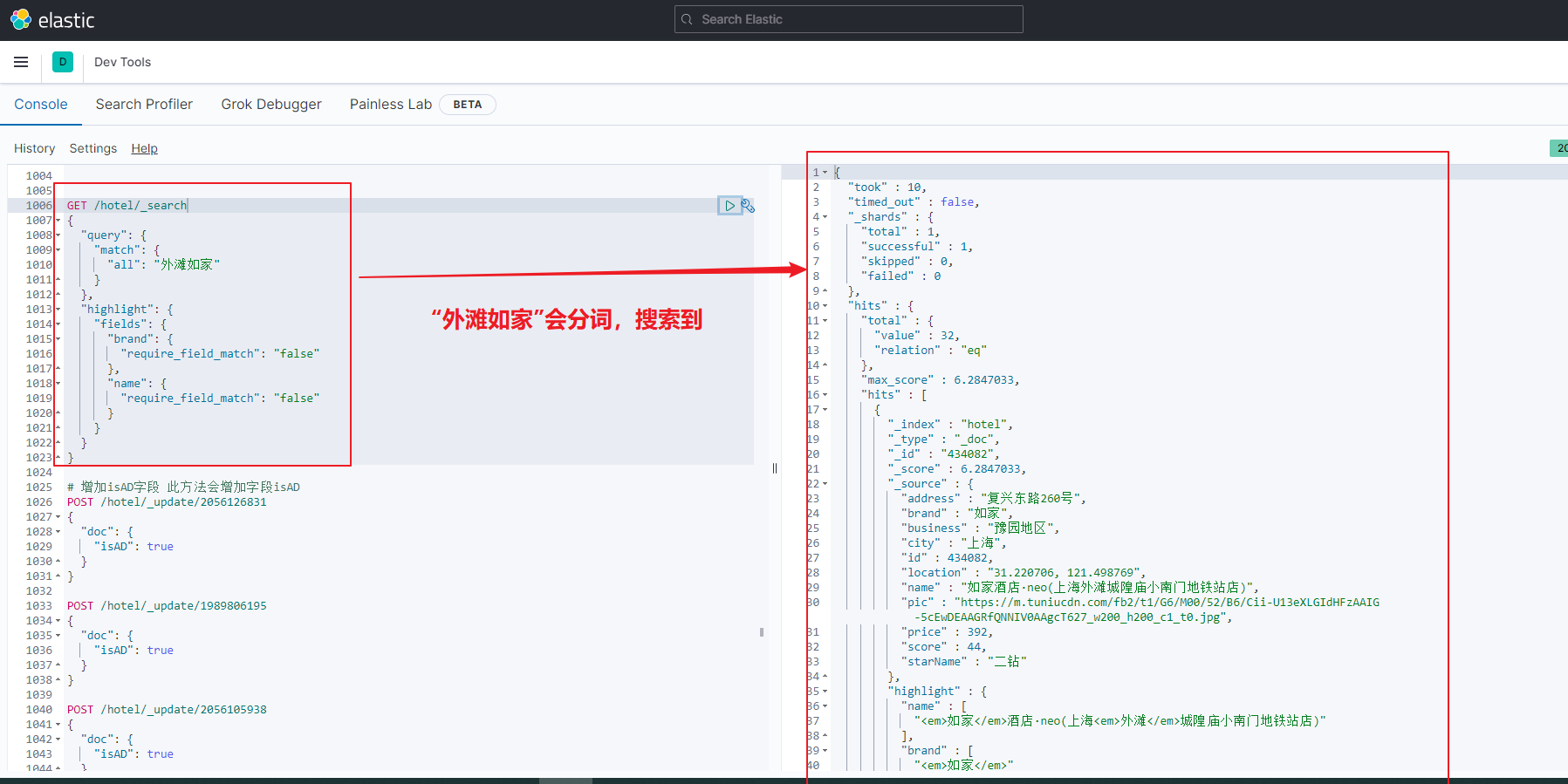
搜索类型一定不能使用match\_all或者事其他类型的查询，一定要使用带关键字的查询，即match查询或者term查询，当然建议使用match全文检索去搜，因为term是精确查询，term不分词，match会分词，因此match匹配到的文档更多！

2：DSL搜索语句中，query里面的搜索类型，比如：

term表示的是就对输入的内容不分词，输入内容作为一个整体到对应字段的倒排索引表中查询；只要倒排索引表中包含输入的内容，就匹配成功，比如倒排索引表中的是“上海浦东如家酒店”，而你输入的是“如家”，那么因为“上海浦东如家酒店”包含“如家”，所以“上海浦东如家酒店”对应的文档也能匹配到，注意是“包含”，不是“等于”。



match一般对应的是text文本类型字段，表示对输入的内容按照索引库中对应字段的分词器进行分词，得到一个个词条，然后这些词条再到对应字段的倒排索引表中查询。只要倒排索引表中包含分词得出的词条，就匹配成功，比如倒排索引表中的是“上海浦东如家酒店”，而你分此后有个词条是“如家”，那么因为“上海浦东如家酒店”包含“如家”，所以“上海浦东如家酒店”对应的文档也能匹配到，注意是“包含”，不是“等于”。



### 案例：演示DSL实现高亮查询

#### DSL语句如下：

**GET /hotel/\_search**

**{**

**"query": {**

**"match": {**

**"all": "外滩如家"**

**}**

**},**

**"highlight": {**

**"fields": {**

**"brand": {**

**"require\_field\_match": "false"**

**},**

**"name": {**

**"require\_field\_match": "false"**

**}**

**}**

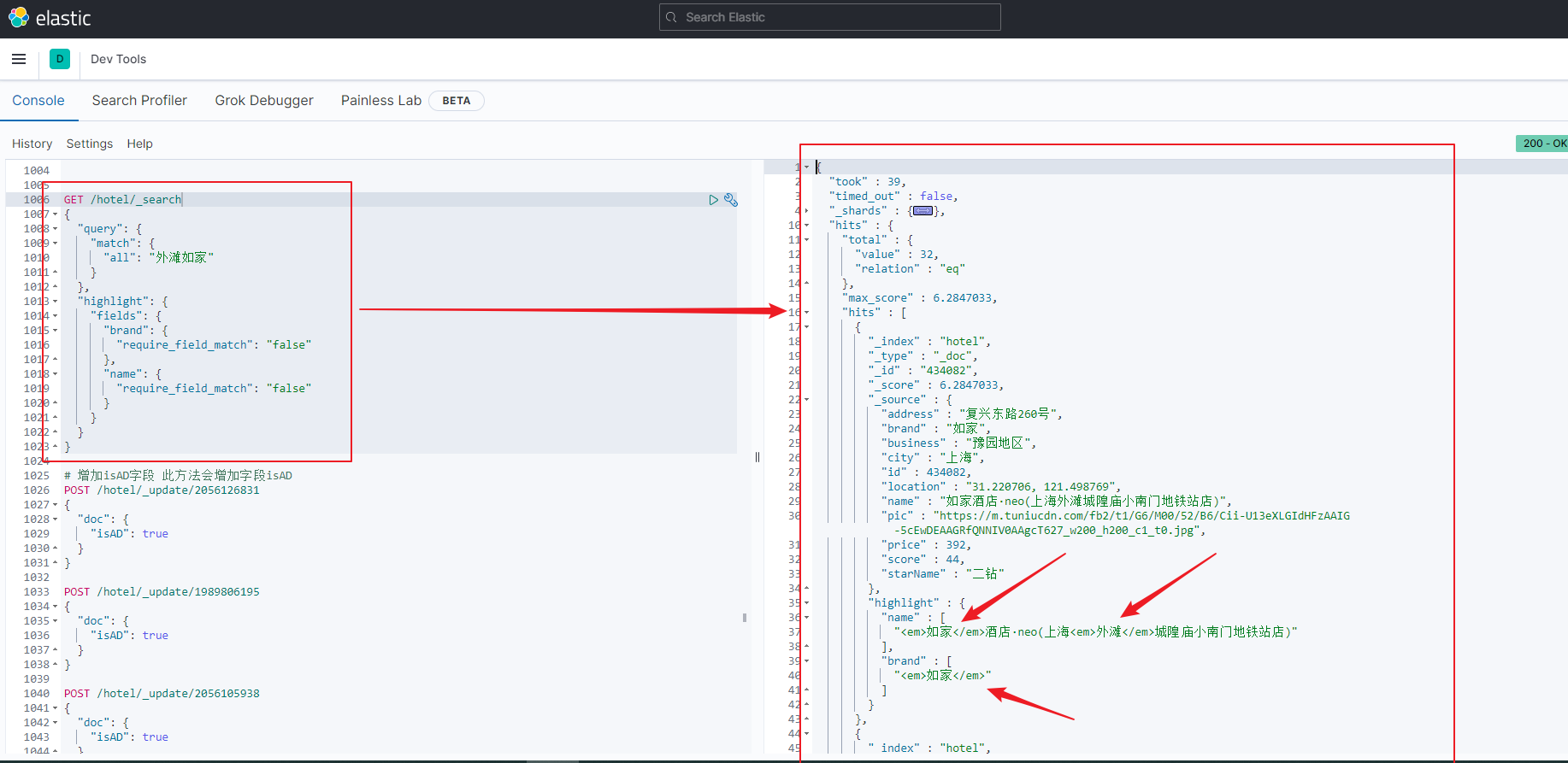
**}**

**}**

注意，默认情况下，es搜索字段必须和高亮字段一致，否则不会高亮，此时搜索字段是all，高亮字段是name，此时默认无法高亮，如果要高亮，写上配置： "require\_field\_match": "false"，意思就是不需要字段匹配。

#### DevTools控制台查看搜索结果的高亮内容：

\_source是原始文档内容，高亮结果在下面的highlight中返回了！高亮已经实现，将来你要干的事就是拿highlight高亮的内容把\_source中不高亮的对应字段替换掉！



### 总结：搜索结果处理整体语法

