





qz_baozi

码龄6年

暂无认证

6

35万+

52万+

4928



等级

原创

周排名

总排名

访问

130

1

4

0

6

积分

粉丝

获赞

评论

收藏





私信

关注

搜博主文章

Q

热门文章

将redis设置为系统服务

719

Java 文件、文件夹权限修改

453

Elasticsearch Query DSL 整理总结 (一) —— Query DSL 概要, MatchAllQuery, 全文查询简述

339

MongoDB 3.2版本常用代码全整理(1) - 增删改

278

MongoDB 3.2版本常用代码全整理(3) - 地理空间索引

255

分类专栏

	架构	1篇
	Java7并发编程学习笔记	4篇
	mongodb	2篇
	redis	1篇
	杂项	1篇
	网络	1篇

最新文章

mysql的默认隔离级别

Flutter自学旅程

新浪微博技术架构分析 2010

2020年 2篇

2019年 10篇

2018年 1篇

2017年 8篇

目录

Elasticsearch Java Rest Client API 整理...

引言

Search APIs

Search API

Search Request

可选参数

使用 SearchSourceBuilder

构建查询条件

指定排序

高亮请求

聚合请求

建议请求 Requesting Suggestions

对请求和聚合分析

Elasticsearch Java Rest Client API 整理总结 (二) —— SearchAPI

转载 qz_baozi 2019-07-04 16:20:06 87 收藏 1

分类专栏: Elasticsearch 文章标签: Elasticsearch

作者: ReyCG

出处: ReyCG 的博客 — <https://www.cnblogs.com/reycg-blog/>

Elasticsearch Java Rest Client API 整理总结 (二) —— SearchAPI

目录

引言

Search APIs

Search API

Search Request

可选参数

使用 SearchSourceBuilder

构建查询条件

指定排序

高亮请求

聚合请求

建议请求 Requesting Suggestions

对请求和聚合分析

同步执行

异步执行

查询响应 SearchResponse

Search API 查询关系

结语

系列文章列表

引言

在上一篇 中主要介绍了 Document API, 本节中讲解 [search API](#)

Search APIs

Java High Level REST Client 支持下面的 Search API:

- [Search API](#)
- [Search Scroll API](#)
- [Clear Scroll API](#)
- [Multi-Search API](#)
- [Ranking Evaluation API](#)

Search API

Search Request

`searchRequest` 用来完成和搜索文档, 聚合, 建议等相关的任何操作同时也提供了各种方式来完成对查询结果的高亮操作。

最基本的查询操作如下

```
1 SearchRequest searchRequest = new SearchRequest();
2 SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
3 searchSourceBuilder.query(QueryBuilders.matchAllQuery()); // 添加 match_all 查询
  searchRequest.source(searchSourceBuilder); // 将 SearchSourceBuilder 添加到 SeachReq
```

可选参数

```
1 SearchRequest searchRequest = new SearchRequest("posts"); // 设置搜索的 index
  searchRequest.types("doc"); // 设置搜索的 type
```

除了配置 `index` 和 `type` 外, 还有一些其他的可选参数

```
1 searchRequest.routing("routing"); // 设置 routing 参数
  searchRequest.preference("_local"); // 配置搜索时偏爱使用本地分片, 默认是使用随机分片
```

什么是 routing 参数?

当索引一个文档的时候, 文档会被存储在一个主分片上。在存储时一般都会有多个主分片。Elasticsearch 如何知道一个文档应该放置在哪个分片呢? 这个过程是根据下面的这个公式来决定的:

```
1 shard = hash(routing) % number_of_primary_shards
```

- `routing` 是一个可变值, 默认是文档的 `_id` ,也可以设置成一个自定义的值
- `number_of_primary_shards` 是主分片数量

所有的文档 API 都接受一个叫做 `routing` 的路由参数, 通过这个参数我们可以自定义文档到分片的映射。一个自定义的路由参数可以用来确保所有相关的文档——例如所有属于同一个用户的文档——都被存储到同一个分片中。

使用 SearchSourceBuilder

对搜索行为的配置可以使用 `SearchSourceBuilder` 来完成, 来看一个实例

```
1 SearchSourceBuilder sourceBuilder = new SearchSourceBuilder(); // 默认配置
2 sourceBuilder.query(QueryBuilders.termQuery("user", "kimchy")); // 设置搜索, 可以是任何
3 sourceBuilder.from(0); // 起始 index
4 sourceBuilder.size(5); // 大小 size
  sourceBuilder.timeout(new TimeValue(60, TimeUnit.SECONDS)); // 设置搜索的超时时间
```

设置完成后, 就可以添加到 `SearchRequest` 中。

```
1 SearchRequest searchRequest = new SearchRequest();
  searchRequest.source(sourceBuilder);
```

构建查询条件

查询请求是通过使用 `QueryBuilder` 对象来完成的, 并且支持 Query DSL。

DSL (domain-specific language) 领域特定语言, 是指专注于某个应用程序领域的计算机语言。
— 百度百科

可以使用构造函数来创建 `QueryBuilder`

```
MatchQueryBuilder matchQueryBuilder = new MatchQueryBuilder("user", "kimchy");
```



举报

```
1 matchQueryBuilder.fuzziness(Fuzziness.AUTO); // 模糊查询
2 matchQueryBuilder.prefixLength(3); // 前缀查询的长度
  matchQueryBuilder.maxExpansions(10); // max expansion 选项，用来控制模糊查询
```

也可以使用 **QueryBuilders** 工具类来创建 **QueryBuilder** 对象。这个类提供了函数式编程风格的各种方法来快速创建 **QueryBuilder** 对象。

```
1 QueryBuilder matchQueryBuilder = QueryBuilders.matchQuery("user", "kimchy")
2                                     .fuzziness(Fuzziness.AUTO)
3                                     .prefixLength(3)
                                     .maxExpansions(10);
```

fuzzy-matching 拼写错误时的匹配：
好的全文检索不应该是完全相同的限定逻辑，相反，可以扩大范围来包括可能的匹配，从而根据相关性得分将更好的匹配放在前面。
例如，搜索 **quick brown fox** 时会匹配一个包含 **fast brown foxes** 的文档

不论什么方式创建的 **QueryBuilder**，最后都需要添加到 **SearchSourceBuilder** 中

```
searchSourceBuilder.query(matchQueryBuilder);
```

构建查询 文档中提供了一个丰富的查询列表，里面包含各种查询对应的 **QueryBuilder** 对象以及 **QueryBuilder** helper 方法，大家可以去参考。

关于构建查询的内容会在下篇文章中讲解，敬请期待。

指定排序

SearchSourceBuilder 允许添加一个或多个 **SortBuilder** 实例。这里包含 4 种特殊的实现, (**Field-**，**Score-**，**GeoDistance-** 和 **ScriptSortBuilder**)

```
1 sourceBuilder.sort(new ScoreSortBuilder().order(SortOrder.DESC)); // 根据分数 _score
  sourceBuilder.sort(new FieldSortBuilder("_uid").order(SortOrder.ASC)); // 根据 id 排序
```

过滤数据源

默认情况下，查询请求会返回文档的内容 **_source**，当然我们也可以配置它。例如，禁止对 **_source** 的获取

```
sourceBuilder.fetchSource(false);
```

也可以使用通配符模式以更细的粒度包含或排除特定的字段：

```
1 String[] includeFields = new String[] {"title", "user", "innerObject.*"};
2 String[] excludeFields = new String[] {"_type"};
  sourceBuilder.fetchSource(includeFields, excludeFields);
```

高亮请求

可以通过在 **SearchSourceBuilder** 上设置 **HighlightBuilder** 完成对结果的高亮，而且可以配置不同的字段具有不同的高亮行为。

```
1 SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
2 HighlightBuilder highlightBuilder = new HighlightBuilder();
3 HighlightBuilder.Field highlightTitle =
4     new HighlightBuilder.Field("title"); // title 字段高亮
5 highlightTitle.highlighterType("unified"); // 配置高亮类型
6 highlightBuilder.field(highlightTitle); // 添加到 builder
7 HighlightBuilder.Field highlightUser = new HighlightBuilder.Field("user");
8 highlightBuilder.field(highlightUser);
  searchSourceBuilder.highlighter(highlightBuilder);
```

聚合请求

要实现聚合请求分两步

1. 创建合适的 **AggregationBuilder**
2. 作为参数配置在 **SearchSourceBuilder** 上

```
1 SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
2 TermsAggregationBuilder aggregation = AggregationBuilders.terms("by_company")
3     .field("company.keyword");
4 aggregation.subAggregation(AggregationBuilders.avg("average_age")
5     .field("age"));
  searchSourceBuilder.aggregation(aggregation);
```

建议请求 Requesting Suggestions

SuggestionBuilder 实现类是由 **SuggestBuilders** 工厂类来创建的。

```
1 SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
2 SuggestionBuilder termSuggestionBuilder =
3     SuggestBuilders.termSuggestion("user").text("kimchy");
4 SuggestBuilder suggestBuilder = new SuggestBuilder();
5 suggestBuilder.addSuggestion("suggest_user", termSuggestionBuilder);
  searchSourceBuilder.suggest(suggestBuilder);
```

对请求和聚合分析

分析 API 可用来对一个特定的查询操作中的请求和聚合进行分析，此时要将 **SearchSourceBuilder** 的 profile 标志位设置为 true

```
1 SearchSourceBuilder searchSourceBuilder = new SearchSourceBuilder();
  searchSourceBuilder.profile(true);
```

只要 **SearchRequest** 执行完成，对应的 **SearchResponse** 响应中就会包含 **分析结果**

同步执行

同步执行是阻塞式的，只有结果返回后才能继续执行。

```
SearchResponse searchResponse = client.search(searchRequest);
```

异步执行

异步执行使用的是 **listener** 对结果进行处理。

```
1 ActionListener<SearchResponse> listener = new ActionListener<SearchResponse>() {
2     @Override
3     public void onResponse(SearchResponse searchResponse) {
4         // 查询成功
5     }
6
7     1 <span class="hljs-meta">@Override</span>
8     2 <span class="hljs-function"><span class="hljs-keyword">public</span></span> <span class="hljs-keyword">void</span></span> <span class="hljs-keyword">onResponse</span><span class="hljs-punctuation">(</span>SearchResponse searchResponse)<span class="hljs-punctuation">(</span> {
9     3     <span class="hljs-keyword">public</span></span> <span class="hljs-keyword">void</span></span> <span class="hljs-keyword">onResponse</span><span class="hljs-punctuation">(</span>SearchResponse searchResponse)<span class="hljs-punctuation">(</span> {
10    4     // 查询成功
11    }
12 }
```

};

查询响应 SearchResponse

查询执行完成后，会返回 **SearchResponse** 对象，并在对象中包含查询执行的细节和符合条件的文档集合。



- 请求本身的信息，如 HTTP 状态码，执行时间，或者请求是否超时

```
1 | RestStatus status = searchResponse.status(); // HTTP 状态码
2 | TimeValue took = searchResponse.getTook(); // 查询占用的时间
3 | Boolean terminatedEarly = searchResponse.isTerminatedEarly(); // 是否由于 SearchSourceTimeout
   | boolean timedOut = searchResponse.isTimedOut(); // 是否超时
```

- 查询影响的分片数量的统计信息，成功和失败的分片

```
1 | int totalShards = searchResponse.getTotalShards();
2 | int successfulShards = searchResponse.getSuccessfulShards();
3 | int failedShards = searchResponse.getFailedShards();
4 | for (ShardSearchFailure failure : searchResponse.getShardFailures()) {
5 |     // failures should be handled here
   }
```

检索 SearchHits

要访问返回的文档，首先要在响应中获取其中的 SearchHits

```
SearchHits hits = searchResponse.getHits();
```

SearchHits 中包含了所有命中的全局信息，如查询命中的数量或者最大分值：

```
1 | long totalHits = hits.getTotalHits();
   | float maxScore = hits.getMaxScore();
```

查询的结果嵌套在 SearchHits 中，可以通过遍历循环获取

```
1 | SearchHit[] searchHits = hits.getHits();
2 | for (SearchHit hit : searchHits) {
3 |     // do something with the SearchHit
   }
```

SearchHit 提供了如 index , type , docId 和每个命中查询的分数

```
1 | String index = hit.getIndex();
2 | String type = hit.getType();
3 | String id = hit.getId();
   | float score = hit.getScore();
```

而且，还可以获取到文档的源数据，以 JSON-String 形式或者 key-value map 对的形式。在 map 中，字段可以是普通类型，或者是列表类型，嵌套对象。

```
1 | String sourceAsString = hit.getSourceAsString();
2 | Map<String, Object> sourceAsMap = hit.getSourceAsMap();
3 | String documentTitle = (String) sourceAsMap.get("title");
4 | List<Object> users = (List<Object>) sourceAsMap.get("user");
5 | Map<String, Object> innerObject =
   | (Map<String, Object>) sourceAsMap.get("innerObject");
```

Search API 查询关系

上面的 QueryBuilder , SearchSourceBuilder 和 SearchRequest 之间都是嵌套关系，为此我专门整理了一个关系图，以便更清楚的确认它们之间的关系。感兴趣的同学可用此图与前面的 API 进行对应，以加深理解。

![search API 关系图]

(https://imgconvert.csdnimg.cn/aHR0cHM6Ly9pbWFnZXMuY25ibG9ncy5jb20vY25ibG9nc19jb20vcmlV5Y2ctYmxvYzY8xMzQwNTczL29jcmVzdF9oaWdoX2NsaWVudF9zZWZyY2hfcmlV5YXRpb24ucG5n)

结语

本篇包含了 Java High level Rest Client 的 SearchAPI 部分，获取高亮，聚合，分析的结果并没有在本文涉及，需要的同学可参考官方文档，下篇会包含查询构建，敬请期待~

系列文章列表

- Elasticsearch Java Rest Client API 整理总结 (一)——Document API
- Elasticsearch Java Rest Client API 整理总结 (二) —— SearchAPI
- Elasticsearch Java Rest Client API 整理总结 (三)——Building Queries

Elasticsearch Java Rest Client API 整理总结 (三)——Building Queries dianqiulai2465的博客 37
目录 上篇回顾 Building Queries 匹配所有的查询 全文查询 Full Text Queries 什么是全文查询？ Match 全文查询 API 列表 ...

Elasticsearch Java Rest Client API 整理总结 (一)——Document API qz baozi 210
转载地址 目录 引言 概述 High REST Client 起步 兼容性 Java Doc 地址 Maven 配置 依赖 初始化 文档 API Index API ...

优质评论可以帮助作者获得更高权重

抢沙发

评论

相关推荐

- ...Elasticsearch Java Rest Client API 整理总结 (二... 6-6
Java High Level REST Client 支持下面的 Search API: Search API Search Request searchRequest 用来完成和搜索文档,聚...
- ...Elasticsearch Java Rest Client API 整理总结 (二... 4-18
文章标签: elasticsearch java likequery 版权 目录 引言 在上一篇 中主要整理了 Document API,本节中主要讲解 search API ...
- ElasticSearch 应用开发 (八) Java High Level Rest Client——Java API 腊八粥 3714
1.概述 Java High Level REST Client Elasticsearch 官方高级客户端： 基于低级客户端， 提供特定的方法的API， 并处理请...
- ElasticSearch 应用开发 (七) Java Low Level Rest Client——Java API 腊八粥 1862
1.概述 Java restclient分为两种:Java Low Level Rest Client(本节介绍重点)、 Java HighLevel Rest Client (下节介绍)。 Ja...
- Elasticsearch Java Rest Client API 整理总结 (三... 6-14
子曰,温故而知新,可以为师也。学习的过程就是不断的回顾,总结,总结,再总结。首先,一起来回顾下上篇 search API中的内容...
- Elasticsearch Java Rest Client API 整理总结 (一)——Document... 6-14
注意,本 API 指南只针对 elasticsearch 6.3 版本。概述 Rest client 分成两部分: Java Low Level REST Client 官方低级别 es ...
- 十三、ElasticSearch——ES Client API之一 (Java High Level REST Client) m0_38143867的博客 1865
1 ES Client 简介 ES是一个服务，采用C/S结构 REST API， 端口 9200 Transport 连接 端口 9300 ES提供了多种编程语言...
- 十四、ElasticSearch——ES Client API之二 (Java Client) m0_38143867的博客 176
1Java Client 简介 java client 使用 TransportClient， 各种操作本质上都是异步的(可以用 listener， 或返回 Future)。 注意...
- Elasticsearch Java Rest Client API 整理总结 (三... 6-7
Elasticsearch Java Rest Client API 整理总结 (三)——Building Queries Building Queries 匹配所有的查询 全文查询 Full Text...
- Elasticsearch Java Rest Client API 整理总结 (一) 6-4
Elasticsearch Java Rest Client API 整理总结 (一) http://www.likecs.com/default/index/show?id=39549 转载于:https://www...
- Elasticsearch——RestHighLevelClient qln_weilong的博客 284
介绍 java rest client有两个实现类，分别是RestClient和RestHighLevelClient。前者是一个低级客户端，通过Http与elasticse...
- ElasticSearch 7.6.x 版本 2020最新版 JavaRest api Mr_kidBK的博客 3562
点个关注吧，球球啦！ 持续更新中..... 前言 周末闲来无事，到官网学习 (fanyi) 了最新版版本的ES JavaRest API。 E...
- Java High Level REST Client 中文API (仅供参考) 含江君 4384
1、初始化 兼容性 Java High Level REST Client需要Java 1.8，并依赖于Elasticsearch核心项目， 客户端版本与客户端开发...
- 实时搜索引擎Elasticsearch (5) ——Java API的使用 HinyLover的专栏 3万+
前一篇有关ES的文章介绍了使用Rest方式调用ES的聚合API。本文介绍ES的Java API调用。
- SpringBoot:Java High Level REST Client 搜索 API qq_42338293的博客 161
Springboot整合最新版elasticsearch参考之前的文章： SpringBoot： 整合ElasticSearch 7.2.0 Search API SearchRequest用...

- Elasticsearch Java API 很全的整理

HuaZi_Myth的博客 199

Elasticsearch 的API 分为 REST Client API (http请求形式) 以及 transportClient API两种。相比来说transportClient API效...
- Java小项目--小型图书管理系统（含完整代码及工具）-附件资源

03-05

Java小项目--小型图书管理系统（含完整代码及工具）-附件资源
- 程序员的数学：微积分

09-28

本课程介绍程序员必备的数学基础内容，在取材上侧重人工智能、数据分析等热门领域
- CSDN开发者助手，常用网站自动整合，多种工具一键调用

CSDN开发者助手由CSDN官方开发，集成一键呼出搜索、万能快捷工具、个性化新标签页和官方免广告四大功能。帮助您...
- u校园考试答案直接查看脚本, u校园自动答题, 自动填写答案, 调用隐藏接口, 100%出答案, 仅供研究...

06-21

u校园考试答案直接查看脚本, u校园自动答题, 自动填写答案, 调用隐藏接口, 100%出答案, 仅供研究使用
- 英特尔® OpenVINO™ 工具套件初级课程

02-11

<p> 欢迎参加英特尔® OpenVINO™ 工具套件初级课程！ 本课程面向零基础学员，将从AI的基本概念开始，介绍人工智能与...
- 图书管理系统（Java + Mysql）我的第一个完全自己做的实训项目

01-04

图书管理系统 Java + MySQL 完整实训代码，MVC三层架构组织，包含所有用到的图片资源以及数据库文件，大三上学期...
- 为了追学姐，用python把她的照片做成了拼图游戏，她看了...

lexsaints 6万+

pygame开发小游戏，附录完整代码。【建议收藏】

©2020 CSDN 皮肤主题: 大白 设计师:CSDN官方博客 返回首页

关于我们 招贤纳士 广告服务 开发助手 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00
公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心 网络110报警服务 中国互联网举报中心 家长监护 Chrome商店下载 ©1999-2021北京创新乐知网络技术有限公司 版权与免责声明 版权申诉 出版物许可证 营业执照



举报