Springboot+es全文检索

<https://blog.csdn.net/weixin_37281289/article/details/101483434>

网盘下载地址

**查看所有的索引(索引就像是数据库一样) :**

postman方法get请求。<http://localhost:9200/_all>

kibana方法 get.       GET \_all即可查询(不加[http://localhost:9200/](http://localhost:9200/_all)是因为kibana默认以及连接了es)

**创建索引:**

**postman用PUT请求 [http://localhost:9200/test意思就是创建一个test索引](http://localhost:9200/test%E6%84%8F%E6%80%9D%E5%B0%B1%E6%98%AF%E5%88%9B%E5%BB%BA%E4%B8%80%E4%B8%AAtest%E7%B4%A2%E5%BC%95)**

**kibana方法**

**删除索引:**

**Postman   DELETE方法请求。   <http://localhost:9200/test>**

**kibana方法**

**为所有添加数据(相当于表):**

**Postman PUT。<http://localhost:9200/person/_doc/1>**

在raw中输入数据{

"first\_name" : "John",

"last\_name":"Smith",

"age":25,

"about":"I love go to rock climbing",

"interests":["sports","music"]

}

即可创建一条数据在\_doc中

**kibana方法**

**查询数据:**

**postman get方法   [http://localhost:9200/person/\_doc/1根据id查询](http://localhost:9200/person/_doc/1%E6%A0%B9%E6%8D%AEid%E6%9F%A5%E8%AF%A2)**

**postman get方法 根据其他数据查询。[http://localhost:9200/person/\_doc/\_search?q=first\_name:zhang1根据姓名查询](http://localhost:9200/person/_doc/_search?q=first_name:zhang1%E6%A0%B9%E6%8D%AE%E5%A7%93%E5%90%8D%E6%9F%A5%E8%AF%A2)**

[http://localhost:9200/person/\_doc/\_search?q=age:25根据年龄查询](http://localhost:9200/person/_doc/_search?q=age:25%E6%A0%B9%E6%8D%AE%E5%B9%B4%E9%BE%84%E6%9F%A5%E8%AF%A2)

**kibana方法**

GET /person/\_doc/1根据id

GET /person/\_doc/\_search?q=age:25根据age

**POST /person/\_search**

**{**

**"query":{**

**"bool": {**

**"should": [**

**{**

**"match": {**

**"last\_name": "Smith"**

**}**

**}**

**]**

**}**

**}**

**}根据姓名等用post就需要这么写**

**第二种多个条件用should时代表的是or的意思**

**POST /person/\_search**

**{**

**"query":{**

**"bool": {**

**"should": [**

**{**

**"match": {**

**"last\_name": "Smith"**

**}**

**}**

**,**

**{**

**"match": {**

**"about": "rock"**

**}**

**}**

**]**

**}**

**}**

**}**

**第三种多个条件下用must代表and**

**POST /person/\_search**

**{**

**"query":{**

**"bool": {**

**"must": [**

**{**

**"match": {**

**"last\_name": "Smith"**

**}**

**}**

**,**

**{**

**"match": {**

**"about": "rock"**

**}**

**}**

**]**

**}**

**}**

**}**

**为所有修改数据(相当于表) ：**

**PostmanPUT。<http://localhost:9200/person/_doc/1>**

在raw中输入数据{

"first\_name" : "John1”,

"last\_name":"Smith",

"age":25,

"about":"I love go to rock climbing",

"interests":["sports","music"]

}

修改时会判断如果没有此数据那么就是新增操作，如果有此数据那么就是修改操作

**kibana方法**

**分词器**

**es内置分词器**

**standard分词器**

POST \_analyze

{

  "analyzer": "standard",

  "text": "我是中国人"

}

英文可以但是中文分词效果不佳

**IK分词器**

下载地址 <https://github.com/medcl/elasticsearch-analysis-ik>

[https://github.com/medcl/elasticsearch-analysis-ik/releases找到对应的版本和es版本一致](https://github.com/medcl/elasticsearch-analysis-ik/releases%E6%89%BE%E5%88%B0%E5%AF%B9%E5%BA%94%E7%9A%84%E7%89%88%E6%9C%AC%E5%92%8Ces%E7%89%88%E6%9C%AC%E4%B8%80%E8%87%B4)

下载完成后需要去到es安装目录中的plugins目录中新建一个文件夹，将下载后的ik包解压后把里面的所有东西都复制到新建的文件夹中即可，之后需要重启es

第一种方式

POST \_analyze

{

  "analyzer": "ik\_smart",

  "text": "我是中国人"

}

第二种将词分为更多的部分

POST \_analyze

{

  "analyzer": "ik\_max\_word",

  "text": "我是中国人"

}

ik/config/main.dic中包括很多的词

如果一个新的词可以在此文件中进行添加，比如我是慕课网，那么就在文件最下面写

我 我是 慕课网 等分开表示

**连接springboot需要注意springboot版本，6.3.2的es在2.2.0的springboot上可以运行高版本springboot会报错**

需要修改es的

yml文件

将注释打开my-application

然后重启es

springboot结合

Application.properties文件中的配置

*#es配置***spring.data.elasticsearch.cluster-nodes**=**127.0.0.1:9300  
  
spring.data.elasticsearch.cluster-name**=**my-application**

pojo对象

**package** com.example.springbootelasticsearch.espojo;  
  
**import** lombok.Data;  
**import** org.springframework.data.annotation.Id;  
**import** org.springframework.data.elasticsearch.annotations.DateFormat;  
**import** org.springframework.data.elasticsearch.annotations.Document;  
**import** org.springframework.data.elasticsearch.annotations.Field;  
**import** org.springframework.data.elasticsearch.annotations.FieldType;  
  
**import** javax.persistence.Column;  
**import** java.util.Date;  
@Data  
*//相当于表*@Document(indexName = **"blog"**,type = **"doc"**,useServerConfiguration = **true**,createIndex = **false**)  
**public class** ESBlog {  
 @Id  
 **private** Integer **id**;  
 *//添加需要分词的字段用 @Field(type = FieldType.Text,analyzer = "ik\_max\_word")* @Field(type = FieldType.***Text***,analyzer = **"ik\_max\_word"**)  
 **private** String **title**;  
 @Field(type = FieldType.***Text***,analyzer = **"ik\_max\_word"**)  
 **private** String **author**;  
 @Field(type = FieldType.***Text***,analyzer = **"ik\_max\_word"**)  
 **private** String **content**;  
 @Field(type = FieldType.***Date***,format = DateFormat.***custom***,pattern = **"yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch\_millis"**)  
 **private** Date **createTime**;  
 @Field(type = FieldType.***Date***,format = DateFormat.***custom***,pattern = **"yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch\_millis"**)  
 **private** Date **updateTime**;  
}

DAO类继承ElasticsearchRepository用来连接es使用es提供的dao接口

**package** com.example.springbootelasticsearch.esdao;  
  
**import** com.example.springbootelasticsearch.espojo.ESBlog;  
**import** org.springframework.data.elasticsearch.repository.ElasticsearchRepository;  
  
**public interface** EsBlogDao **extends** ElasticsearchRepository<ESBlog,Integer> {  
}

controller中类似于写sql语句

BoolQueryBuilder builder= QueryBuilders.*boolQuery*();

//should是或者的关系  
builder.should(QueryBuilders.*matchPhraseQuery*(**"title"**,blogvo.getKeyword()));  
builder.should(QueryBuilders.*matchPhraseQuery*(**"content"**,blogvo.getKeyword()));

//迭代器可以用Page<ESBlog>来表示  
Page<ESBlog> search= (Page<ESBlog>) **esBlogDao**.search(builder);

//得到满足条件的数据content   
List<ESBlog> content = search.getContent();  
map.put(**"list"**,content);

**logstash设置mysql连接**

**用logstash对mysql的条件是需要有id和时间的字段**

**1.打开文件引入mysql连接的jar**

**2.到config文件夹中新建mysql.conf文件**

**内容如下**

**input{**

**jdbc{**

**#jdbc驱动包位置**

**jdbc\_driver\_library =>"/Users/zhangxianzeng/Downloads/elasticsearchquanbuwenjian/logstash-6.3.2/mysql-connector-java-8.0.16.jar"**

**#要使用的驱动包类，有过开发经验的都应该知道这个**

**jdbc\_driver\_class =>"com.mysql.jdbc.Driver"**

**#mysql数据库连接信息**

**jdbc\_connection\_string =>"jdbc:mysql://127.0.0.1:3306/blog"**

**#mysql用户名**

**jdbc\_user =>"root"**

**#mysql密码**

**jdbc\_password =>"zhangxian11"**

**#定时任务，多久执行一次查询，默认是一分钟，如何想要没有延迟，可以使用schedule =>"\* \* \* \* \*"**

**schedule =>"\* \* \* \* \*"**

**#清空上一次的sql\_last\_value记录**

**clean\_run =>true**

**#你想要执行的语句**

**statement =>"SELECT \* FROM t\_blog WHERE update\_time > :sql\_last\_value AND update\_time < NOW() ORDER BY update\_time desc"**

**}**

**}**

**output{**

**elasticsearch{**

**#es host:port**

**hosts=>["127.0.0.1:9200"]**

**#索引**

**index=>"blog"**

**#\_id**

**document\_id=>"%{id}"**

**}**

**}**

启动命令bin目录下 **logstash -f ../config/mysql.conf**

**查看是否有数据**

**GET /blog/\_stats**

**查看是否有数据**

POST /blog/\_search

{

}

停止重启会出现问题j解决删除data中的所有的文件即可

**全文检索和搜索框自动补全总结**

两种方式

第一种：

如果全文检索和搜索补齐用同一个index，只要是需要全文补齐那么这个index就需要提前新建好，因为需要确定一些类型

新建语句如下

PUT /blog1

{

  "mappings": {

    "doc" : {

      "properties" : {

        "title" : {

          "type": "text",

          "analyzer": "ik\_max\_word",

          "fields": {

            "suggest" : {

              "type" : "completion",

              "analyzer": "ik\_max\_word"

            }

          }

        },

        "content": {

          "type": "text",

          "analyzer": "ik\_max\_word"

        }

      }

    }

  }

}

新建好后启动logstash就会实时进行mysql于es的导入，将数据导入到这个index中，全文检索时会出现问题，补齐搜索时不会有问题，全文检索到问题是输入一个字的数据会找不到与这个数据有关联的数据，最少输入两个字才行。而且也不能很好的进行分词综上还是用第二种比较好一些。

第二种方法：

新建两个名字不同但是数据相同的index,一个是全文检索时用的index,一个是搜索框的index这样两个就不会产生冲突，但是搜索框的index还是需要先在es中新建好，全文检索的不用先新建只需要根据mysql.conf中的index名称即可，但是mysql.conf需要加一个配置

input{

jdbc{

           #jdbc驱动包位置

           jdbc\_driver\_library =>"/Users/zhangxianzeng/Downloads/elasticsearchquanbuwenjian/logstash-6.3.2/mysql-connector-java-8.0.16.jar"

           #要使用的驱动包类，有过开发经验的都应该知道这个

           jdbc\_driver\_class =>"com.mysql.jdbc.Driver"

           #mysql数据库连接信息

           jdbc\_connection\_string =>"jdbc:mysql://127.0.0.1:3306/blog"

           #mysql用户名

           jdbc\_user =>"root"

           #mysql密码

           jdbc\_password =>"zhangxian11"

           #定时任务，多久执行一次查询，默认是一分钟，如何想要没有延迟，可以使用schedule =>"\* \* \* \* \*"

           schedule =>"\* \* \* \* \*"

           #清空上一次的sql\_last\_value记录

           clean\_run =>true

           #你想要执行的语句

           statement =>"SELECT \* FROM t\_blog WHERE update\_time > :sql\_last\_value AND update\_time < NOW() ORDER BY update\_time desc"

}

}

output{

elasticsearch{

#es host:port

hosts=>["127.0.0.1:9200"]

#索引

index=>"blog"

#\_id

document\_id=>"%{id}"

}

//加一个elasticsearch模块

  elasticsearch{

  #es host:port

  hosts=>["127.0.0.1:9200"]

#索引

index=>"blog1"

#\_id

document\_id=>"%{id}"

  }

}

这样在搜索时就不会出现一个字时找不到的问题。

其余多个表导入一个index或者多个表导入多个index的问题可以看其他的比较

**拼音分词器**

<https://github.com/medcl/elasticsearch-analysis-pinyin/releases>

可根据拼音得到数据

**[Elasticsearch拼音分词和IK分词的安装及使用](https://www.cnblogs.com/dashuaiguo/p/10014978.html)**

**一、Es插件配置及下载**

**1.IK分词器的下载安装**

**关于IK分词器的介绍不再多少，一言以蔽之，IK分词是目前使用非常广泛分词效果比较好的中文分词器。做ES开发的，中文分词十有八九使用的都是IK分词器。**

**下载地址:<https://github.com/medcl/elasticsearch-analysis-ik>**

**2.pinyin分词器的下载安装**

**可以在淘宝、京东的搜索框中输入pinyin就能查找到自己想要的结果，这就是拼音分词，拼音分词则是将中文分析成拼音格式，可以通过拼音分词分析出来的数据进行查找想要的结果。**

**下载地址：<https://github.com/medcl/elasticsearch-analysis-pinyin>**

**注：插件下载一定要和自己版本对应的Es版本一致，并且安装完插件后需重启Es，才能生效。**

**插件安装位置：（本人安装了三个插件，暂时先不介绍murmur3插件，可以暂时忽略）**

**插件配置成功，重启Es**

**二、拼音分词器和IK分词器的使用**

**1.IK中文分词器的使用**

**1.1 ik\_smart: 会做最粗粒度的拆分**

GET /\_analyze

{

  "text":"中华人民共和国国徽",

  "analyzer":"ik\_smart"

}

结果：

{

  "tokens": [

    {

      "token": "中华人民共和国",

      "start\_offset": 0,

      "end\_offset": 7,

      "type": "CN\_WORD",

      "position": 0

    },

    {

      "token": "国徽",

      "start\_offset": 7,

      "end\_offset": 9,

      "type": "CN\_WORD",

      "position": 1

    }

  ]

}

**1.2  ik\_max\_word: 会将文本做最细粒度的拆分**

GET /\_analyze

{

  "text": "中华人民共和国国徽",

  "analyzer": "ik\_max\_word"

}

结果：

{

  "tokens": [

    {

      "token": "中华人民共和国",

      "start\_offset": 0,

      "end\_offset": 7,

      "type": "CN\_WORD",

      "position": 0

    },

    {

      "token": "中华人民",

      "start\_offset": 0,

      "end\_offset": 4,

      "type": "CN\_WORD",

      "position": 1

    },

    {

      "token": "中华",

      "start\_offset": 0,

      "end\_offset": 2,

      "type": "CN\_WORD",

      "position": 2

    },

    {

      "token": "华人",

      "start\_offset": 1,

      "end\_offset": 3,

      "type": "CN\_WORD",

      "position": 3

    },

    {

      "token": "人民共和国",

      "start\_offset": 2,

      "end\_offset": 7,

      "type": "CN\_WORD",

      "position": 4

    },

    {

      "token": "人民",

      "start\_offset": 2,

      "end\_offset": 4,

      "type": "CN\_WORD",

      "position": 5

    },

    {

      "token": "共和国",

      "start\_offset": 4,

      "end\_offset": 7,

      "type": "CN\_WORD",

      "position": 6

    },

    {

      "token": "共和",

      "start\_offset": 4,

      "end\_offset": 6,

      "type": "CN\_WORD",

      "position": 7

    },

    {

      "token": "国",

      "start\_offset": 6,

      "end\_offset": 7,

      "type": "CN\_CHAR",

      "position": 8

    },

    {

      "token": "国徽",

      "start\_offset": 7,

      "end\_offset": 9,

      "type": "CN\_WORD",

      "position": 9

    }

  ]

}

**2.拼音分词器的使用**

GET /\_analyze

{

  "text":"刘德华",

  "analyzer": "pinyin"

}

结果:

{

  "tokens": [

    {

      "token": "liu",

      "start\_offset": 0,

      "end\_offset": 1,

      "type": "word",

      "position": 0

    },

    {

      "token": "ldh",

      "start\_offset": 0,

      "end\_offset": 3,

      "type": "word",

      "position": 0

    },

    {

      "token": "de",

      "start\_offset": 1,

      "end\_offset": 2,

      "type": "word",

      "position": 1

    },

    {

      "token": "hua",

      "start\_offset": 2,

      "end\_offset": 3,

      "type": "word",

      "position": 2

    }

  ]

}

**注：不管是拼音分词器还是IK分词器，当深入搜索一条数据是时，必须是通过分词器分析的数据，才能被搜索到，否则搜索不到**

**三、IK分词和拼音分词的组合使用**

**当我们创建索引时可以自定义分词器，通过指定映射去匹配自定义分词器**

PUT /my\_index

{

  "settings": {

        "analysis": {

            "analyzer": {

                "ik\_smart\_pinyin": {

                    "type": "custom",

                    "tokenizer": "ik\_smart",

                    "filter": ["my\_pinyin", "word\_delimiter"]

                },

                "ik\_max\_word\_pinyin": {

                    "type": "custom",

                    "tokenizer": "ik\_max\_word",

                    "filter": ["my\_pinyin", "word\_delimiter"]

                }

            },

            "filter": {

                "my\_pinyin": {

                    "type" : "pinyin",

                    "keep\_separate\_first\_letter" : true,

                    "keep\_full\_pinyin" : true,

                    "keep\_original" : true,

                    "limit\_first\_letter\_length" : 16,

                    "lowercase" : true,

                    "remove\_duplicated\_term" : true

                }

            }

        }

  }

}

**当我们建type时，需要在字段的analyzer属性填写自己的映射**

PUT /my\_index/my\_type/\_mapping

{

    "my\_type":{

      "properties": {

        "id":{

          "type": "integer"

        },

        "name":{

          "type": "text",

          "analyzer": "ik\_smart\_pinyin"

        }

      }

    }

}

**测试，让我们先添加几条数据**

POST /my\_index/my\_type/\_bulk

{ "index": { "\_id":1}}

{ "name": "张三"}

{ "index": { "\_id": 2}}

{ "name": "张四"}

{ "index": { "\_id": 3}}

{ "name": "李四"}

**IK分词查询**

GET /my\_index/my\_type/\_search

{

  "query": {

    "match": {

      "name": "李"

    }

  }

}

结果：

{

  "took": 3,

  "timed\_out": false,

  "\_shards": {

    "total": 5,

    "successful": 5,

    "skipped": 0,

    "failed": 0

  },

  "hits": {

    "total": 1,

    "max\_score": 0.47160998,

    "hits": [

      {

        "\_index": "my\_index",

        "\_type": "my\_type",

        "\_id": "3",

        "\_score": 0.47160998,

        "\_source": {

          "name": "李四"

        }

      }

    ]

  }

}

**拼音分词查询：**

GET /my\_index/my\_type/\_search

{

  "query": {

    "match": {

      "name": "zhang"

    }

  }

}

结果：

{

  "took": 1,

  "timed\_out": false,

  "\_shards": {

    "total": 5,

    "successful": 5,

    "skipped": 0,

    "failed": 0

  },

  "hits": {

    "total": 2,

    "max\_score": 0.3758317,

    "hits": [

      {

        "\_index": "my\_index",

        "\_type": "my\_type",

        "\_id": "2",

        "\_score": 0.3758317,

        "\_source": {

          "name": "张四"

        }

      },

      {

        "\_index": "my\_index",

        "\_type": "my\_type",

        "\_id": "1",

        "\_score": 0.3758317,

        "\_source": {

          "name": "张三"

        }

      }

    ]

  }

}

**同一个mysql的表导入到不同的index中**

input{

jdbc{

           #jdbc驱动包位置

           jdbc\_driver\_library =>"/Users/zhangxianzeng/Downloads/elasticsearchquanbuwenjian/logstash-6.3.2/mysql-connector-java-8.0.16.jar"

           #要使用的驱动包类，有过开发经验的都应该知道这个

           jdbc\_driver\_class =>"com.mysql.jdbc.Driver"

           #mysql数据库连接信息

           jdbc\_connection\_string =>"jdbc:mysql://127.0.0.1:3306/blog"

           #mysql用户名

           jdbc\_user =>"root"

           #mysql密码

           jdbc\_password =>"zhangxian11"

           #定时任务，多久执行一次查询，默认是一分钟，如何想要没有延迟，可以使用schedule =>"\* \* \* \* \*"

           schedule =>"\* \* \* \* \*"

           #清空上一次的sql\_last\_value记录

           clean\_run =>true

           #你想要执行的语句

           statement =>"SELECT \* FROM t\_blog WHERE update\_time > :sql\_last\_value AND update\_time < NOW() ORDER BY update\_time desc"

}

}

output{

elasticsearch{

#es host:port

hosts=>["127.0.0.1:9200"]

#索引

index=>"blog"

#\_id

document\_id=>"%{id}"

}

//新增一个elasticsearch

  elasticsearch{

  #es host:port

  hosts=>["127.0.0.1:9200"]

#索引

index=>"blog1"

#\_id

document\_id=>"%{id}"

  }

}

**全文检索和搜索框自动补全总结**

两种方式

第一种：

如果全文检索和搜索补齐用同一个index，只要是需要全文补齐那么这个index就需要提前新建好，因为需要确定一些类型

新建语句如下

PUT /blog1

{

  "mappings": {

    "doc" : {

      "properties" : {

        "title" : {

          "type": "text",

          "analyzer": "ik\_max\_word",

          "fields": {

            "suggest" : {

              "type" : "completion",

              "analyzer": "ik\_max\_word"

            }

          }

        },

        "content": {

          "type": "text",

          "analyzer": "ik\_max\_word"

        }

      }

    }

  }

}

新建好后启动logstash就会实时进行mysql于es的导入，将数据导入到这个index中，全文检索时会出现问题，补齐搜索时不会有问题，全文检索到问题是输入一个字的数据会找不到与这个数据有关联的数据，最少输入两个字才行。而且也不能很好的进行分词综上还是用第二种比较好一些。

第二种方法：

新建两个名字不同但是数据相同的index,一个是全文检索时用的index,一个是搜索框的index这样两个就不会产生冲突，但是搜索框的index还是需要先在es中新建好，全文检索的不用先新建只需要根据mysql.conf中的index名称即可，但是mysql.conf需要加一个配置

input{

jdbc{

           #jdbc驱动包位置

           jdbc\_driver\_library =>"/Users/zhangxianzeng/Downloads/elasticsearchquanbuwenjian/logstash-6.3.2/mysql-connector-java-8.0.16.jar"

           #要使用的驱动包类，有过开发经验的都应该知道这个

           jdbc\_driver\_class =>"com.mysql.jdbc.Driver"

           #mysql数据库连接信息

           jdbc\_connection\_string =>"jdbc:mysql://127.0.0.1:3306/blog"

           #mysql用户名

           jdbc\_user =>"root"

           #mysql密码

           jdbc\_password =>"zhangxian11"

           #定时任务，多久执行一次查询，默认是一分钟，如何想要没有延迟，可以使用schedule =>"\* \* \* \* \*"

           schedule =>"\* \* \* \* \*"

           #清空上一次的sql\_last\_value记录

           clean\_run =>true

           #你想要执行的语句

           statement =>"SELECT \* FROM t\_blog WHERE update\_time > :sql\_last\_value AND update\_time < NOW() ORDER BY update\_time desc"

}

}

output{

elasticsearch{

#es host:port

hosts=>["127.0.0.1:9200"]

#索引

index=>"blog"

#\_id

document\_id=>"%{id}"

}

//加一个elasticsearch模块

  elasticsearch{

  #es host:port

  hosts=>["127.0.0.1:9200"]

#索引

index=>"blog1"

#\_id

document\_id=>"%{id}"

  }

}

这样在搜索时就不会出现一个字时找不到的问题。

其余多个表导入一个index或者多个表导入多个index的问题可以看其他的比较