搜索引擎选择 Lucene 或 Elasticsearch 或 Solr

> Elasticsearch Solr两者比较

https://segmentfault.com/a/1190000008684910
https://www.cnblogs.com/chowmin/articles/4629220.html

> 工具

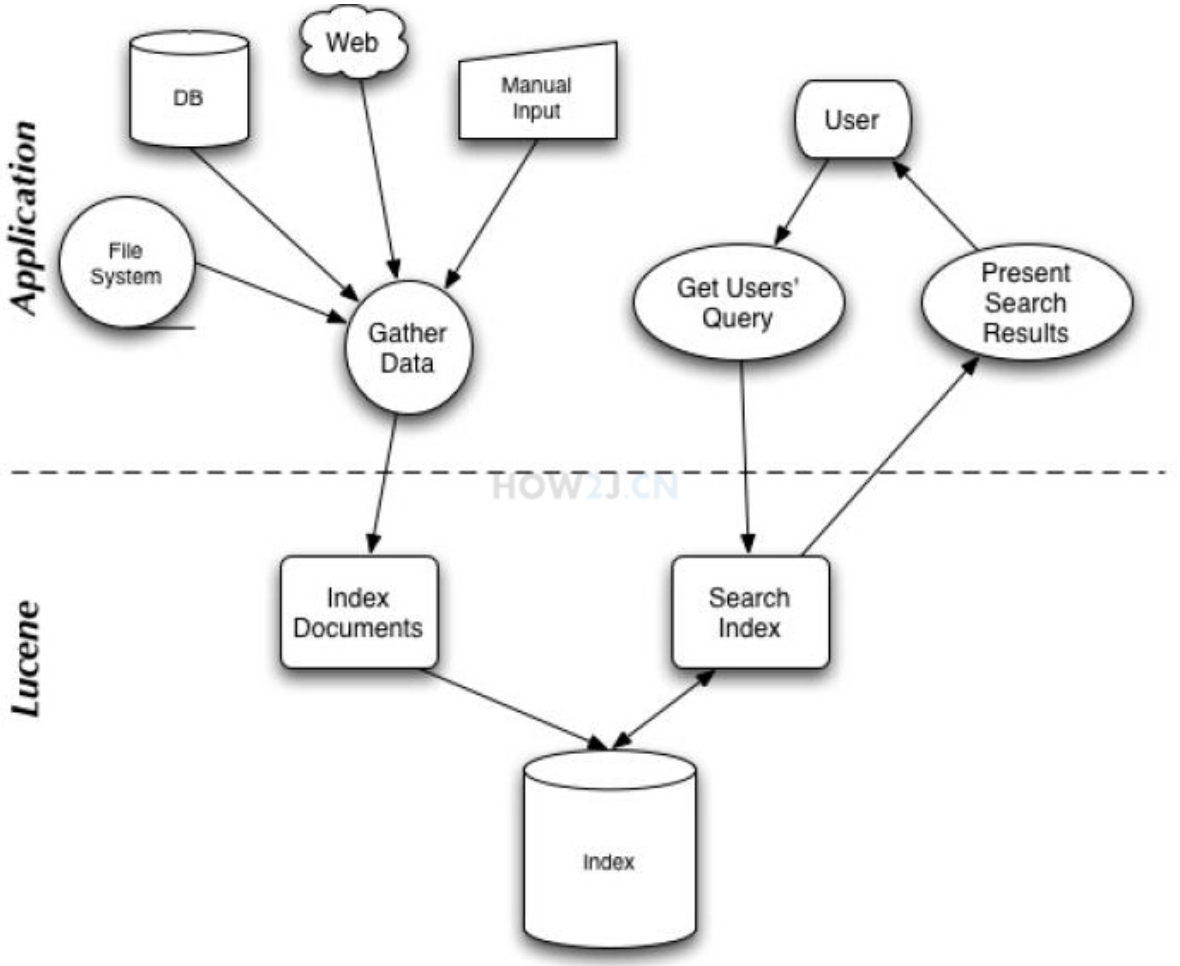
mysql语句在线转换elasticsearch查询语句

> Lucene的使用

lucene的官网

思路:

1. 首先搜集数据
   数据可以是文件系统,数据库,网络上,手工输入的,或者像本例直接写在内存上的.
2. 通过数据创建索引
3. 用户输入关键字
4. 通过关键字创建查询器
5. 根据查询器到索引里获取数据
6. 然后把查询结果展示在用户面前



分词器的概念
分词器指的是搜索引擎如何使用关键字进行匹配,如入门中的关键字: 护眼带光源.
如果使用like,那么%护眼带光源%,匹配出来的结果就是要么全匹配,要不都不匹配.
而使用分词器,就会把这个关键字分为 护眼, 带, 光源 3个关键字, 这样就可以找到不同相关程序的结果了.

IKAnalyzer这个分词器很久都没有维护了,也不支持Lucene7. IKAnalyzer这个是修改之后的.

之后通过对demo的运行有了一个初步的理解.
搜索引擎技术相当于将数据进行相关词条的比对,得出适配度,返回查找的数据,以及关键字.
这有点让我想起看得MOOC的编译原理中语法分析.

> lucene索引删除和更新

索引建立好了之后,还是需要维护的,比如新增,删除和维护. 新增就是建立索引的过程,这里就不表了. 索引里的数据,其实就是一个一个的Document对象,那么本文就是介绍如何删除和更新这些Document对象.

删除索引和更新索引

```
//删除id=51173的数据
IndexWriterConfig config = new IndexWriterConfig(analyzer);
IndexWriter indexWriter = new IndexWriter(index, config);
indexWriter.deleteDocuments(new Term("id", "51173"));
indexWriter.commit();
indexWriter.close();
```

更新索引

```
// 更新索引
IndexWriterConfig config = new IndexWriterConfig(analyzer);
IndexWriter indexWriter = new IndexWriter(index, config);
Document doc = new Document();
doc.add(new TextField("id", "51173", Field.Store.YES));
doc.add(new TextField("name", "神鞭，鞭没了，神还在", Field.Store.YES));
doc.add(new TextField("category", "道具", Field.Store.YES));
doc.add(new TextField("price", "998", Field.Store.YES));
doc.add(new TextField("place", "南海群岛", Field.Store.YES));
doc.add(new TextField("code", "888888", Field.Store.YES));
indexWriter.updateDocument(new Term("id", "51173"), doc );
indexWriter.commit();
indexWriter.close();
```

更新索引后,再用鞭查询,得到的结果是查出了更新之后的数据.

> Solr

什么是Solr?
前面学习了Lucene,现在开始学习Solr.
以连接数据库为类比: Lucene就相当于JDBC,是基本的用法.
Solr就相当于MyBatis,方便开发人员配置, 访问和调用.
而且Solr被做成了webapp形式,以tomcat的应用的方式启动,提供了可视化的配置界面

启动服务器
切换到solr目录\bin下,执行

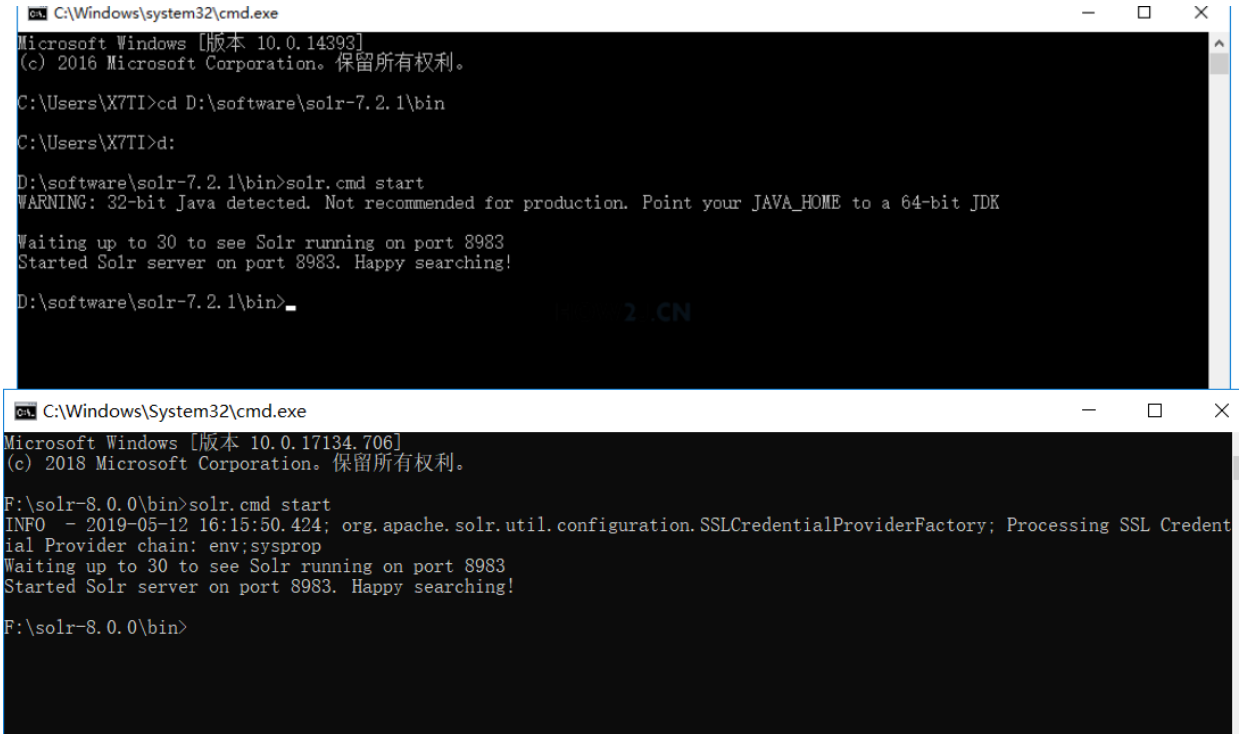
```
solr.cmd start
```

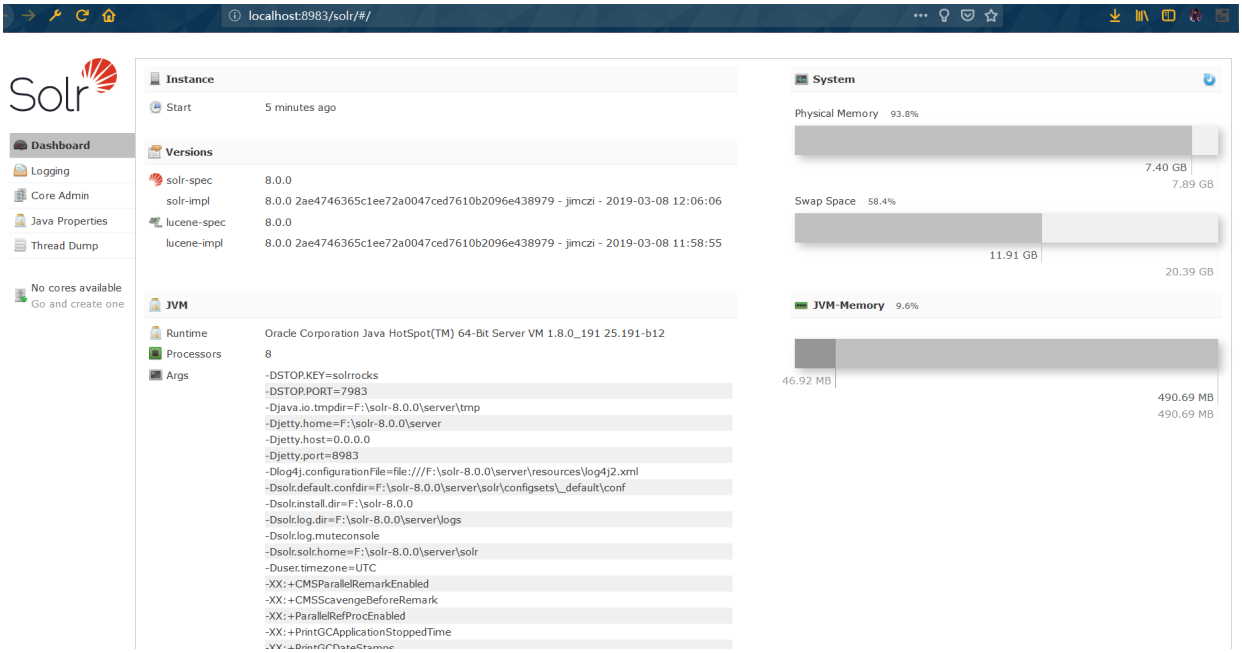
如此就启动了服务器, 会占用端口8983, 倘若端口被占用,会启动失败.

使用solr时,直接进入solr的bin文件夹,从而执行命令

```
solr.cmd start
```

如此就启动了服务器,会占用端口8983.



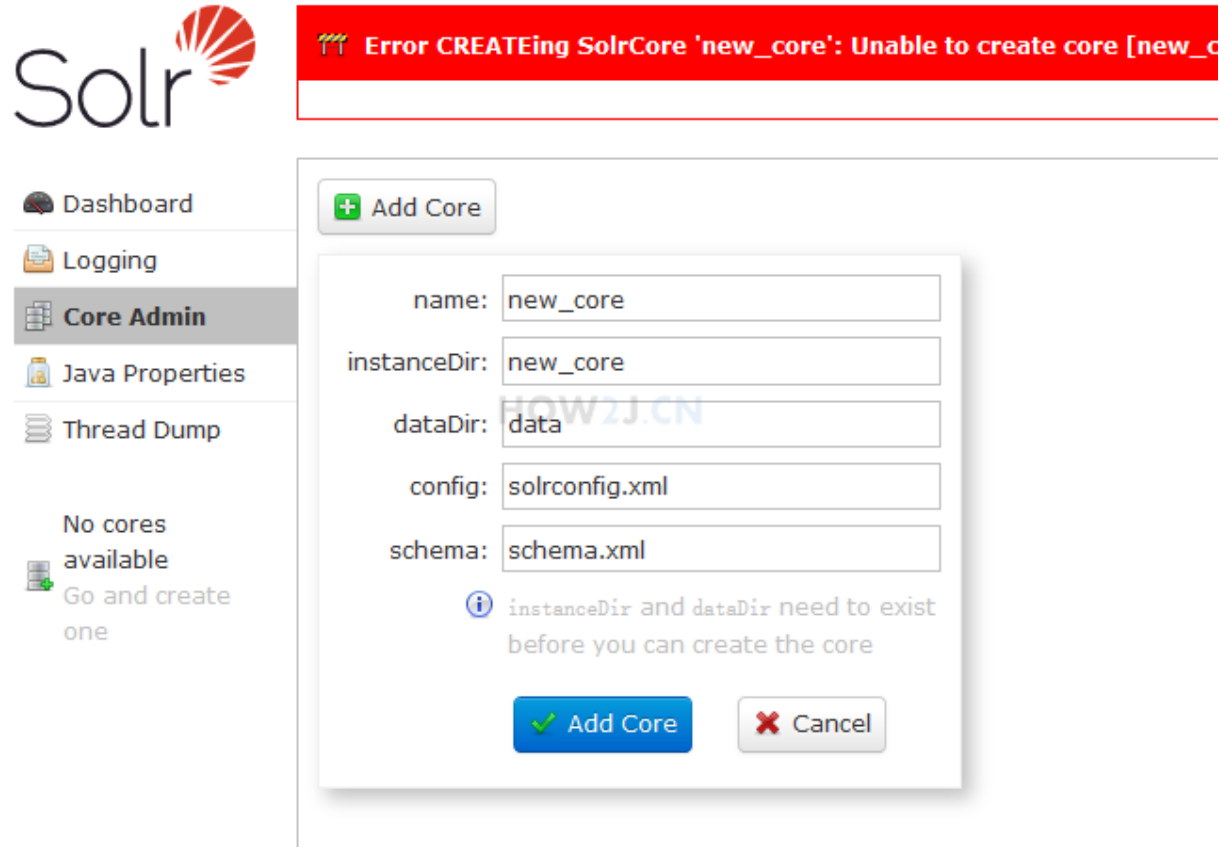


访问服务器
输入地址:
http://localhost:8983/solr/#/

创建Core

如果说Solr相当于一个数据库的话,那么Core就相当于一张表

不要通过图形界面创建Core
如图所示,通过图形界面创建Core会失败,应该使用命令行方式创建Core

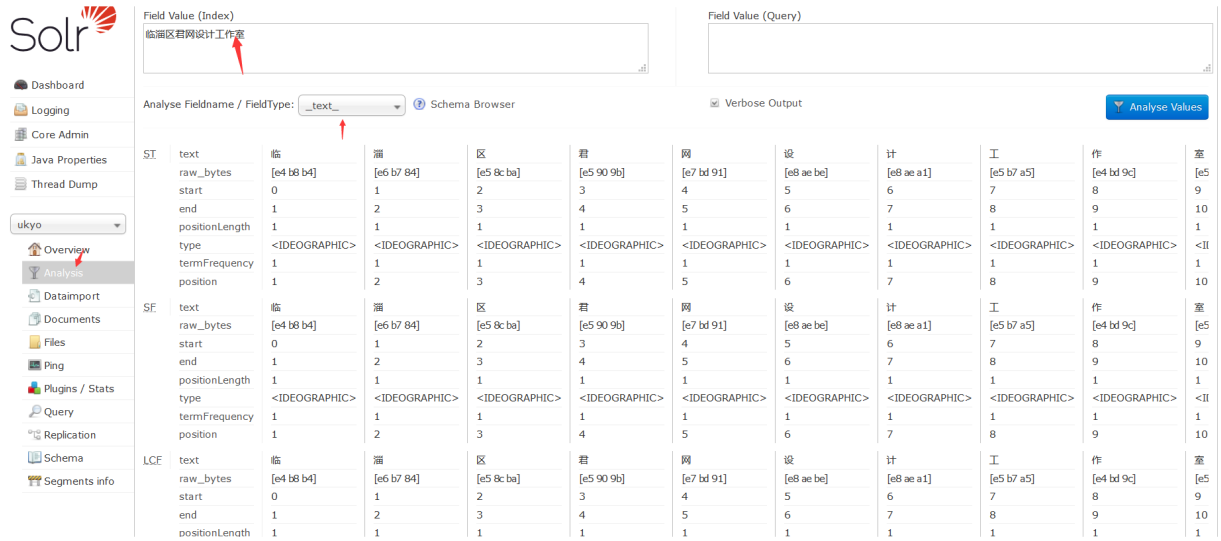


命令行方式创建Core
如图所示就是创建了Core

```
F:\solr-8.0.0\bin>solr.cmd create -c ukyo
WARNING: Using _default configset with data driven schema functionality. NOT RECOMMEN
         To turn off: bin\solr config -c ukyo -p 8983 -action set-user-property -prope
 false
INFO  - 2019-05-12 16:25:33.838; org.apache.solr.util.configuration.SSLCredentialProvi
ial Provider chain: env;sysprop

Created new core 'ukyo'

F:\solr-8.0.0\bin>_
```

删除new_core
如果点击了步骤 不要通过图形界面创建Core 里的图形界面里的Add Core,那么就会一直有错误提醒,那么按照如下方式删除new_core就不会再有错误提醒了.

```
F:\solr-8.0.0\bin>solr.cmd delete -c new_core
INFO  - 2019-05-12 16:30:07.314; org.apache.solr.util.configuration.SSLCredentialProviderFactory; Processing SSL Credent
ial Provider chain: env;sysprop

Deleting core 'new_core' using command:
http://localhost:8983/solr/admin/cores?action=UNLOAD&core=new_core&deleteIndex=true&deleteDataDir=true&deleteInstanceDir
=true
```



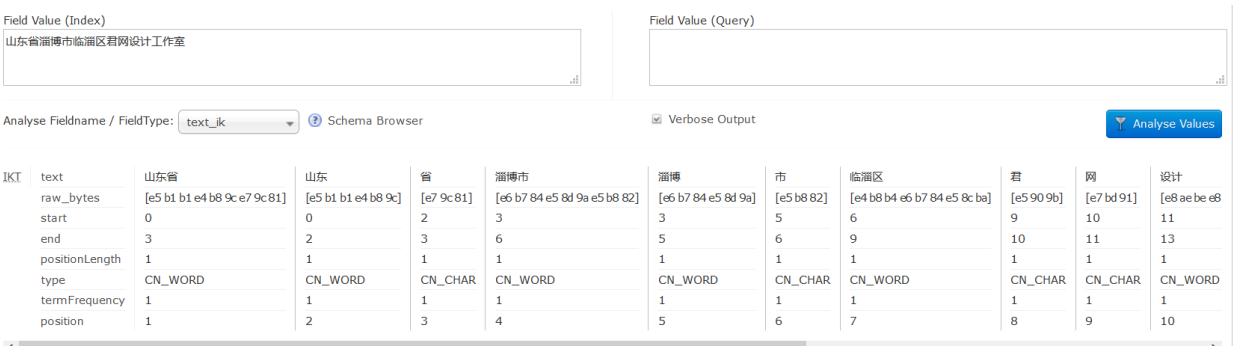
默认情况下是没有中文分词的,如上图,得到的是按照每个字的分词结果

配置中文分词.

重启Solr

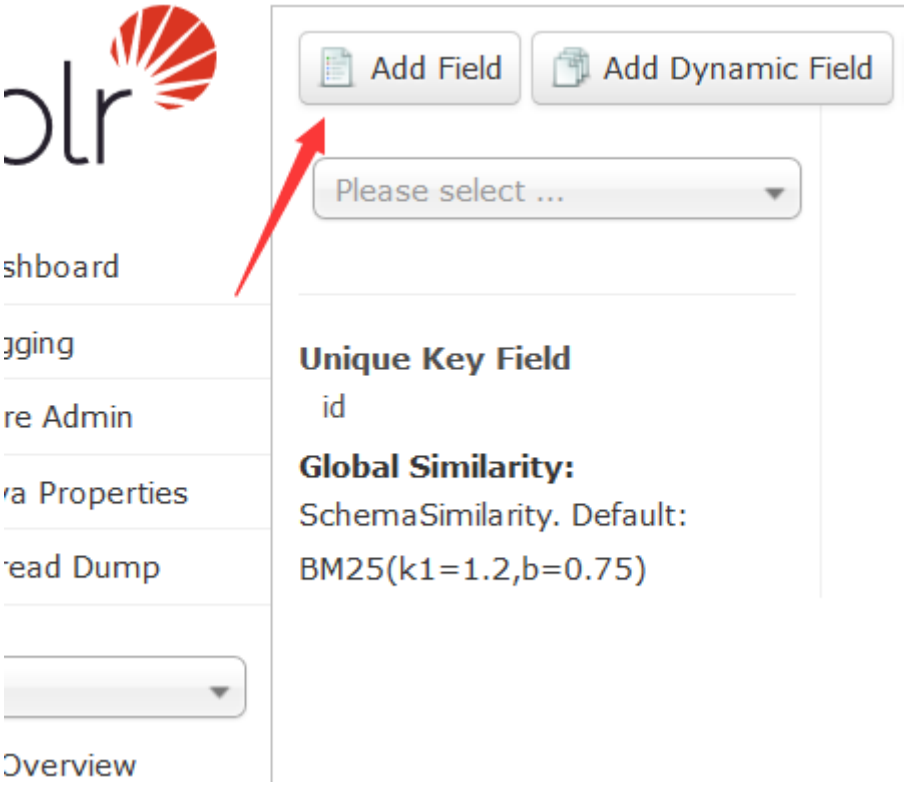
```
solr.cmd stop -all
solr.cmd start
```

这里再次进行分词,而FieldType选择新配置的text_ik



设置字段

字段概念
创建Core中的Core就相当于表,那么接下来就要呀为这个表设置字段,用于存放数据.



http://localhost:8983/solr/#/ukyo/schema 已经添加了字段了,接下来创建索引

创建索引
http://how2j.cn/k/search-engine/search-engine-create-index/1682.html

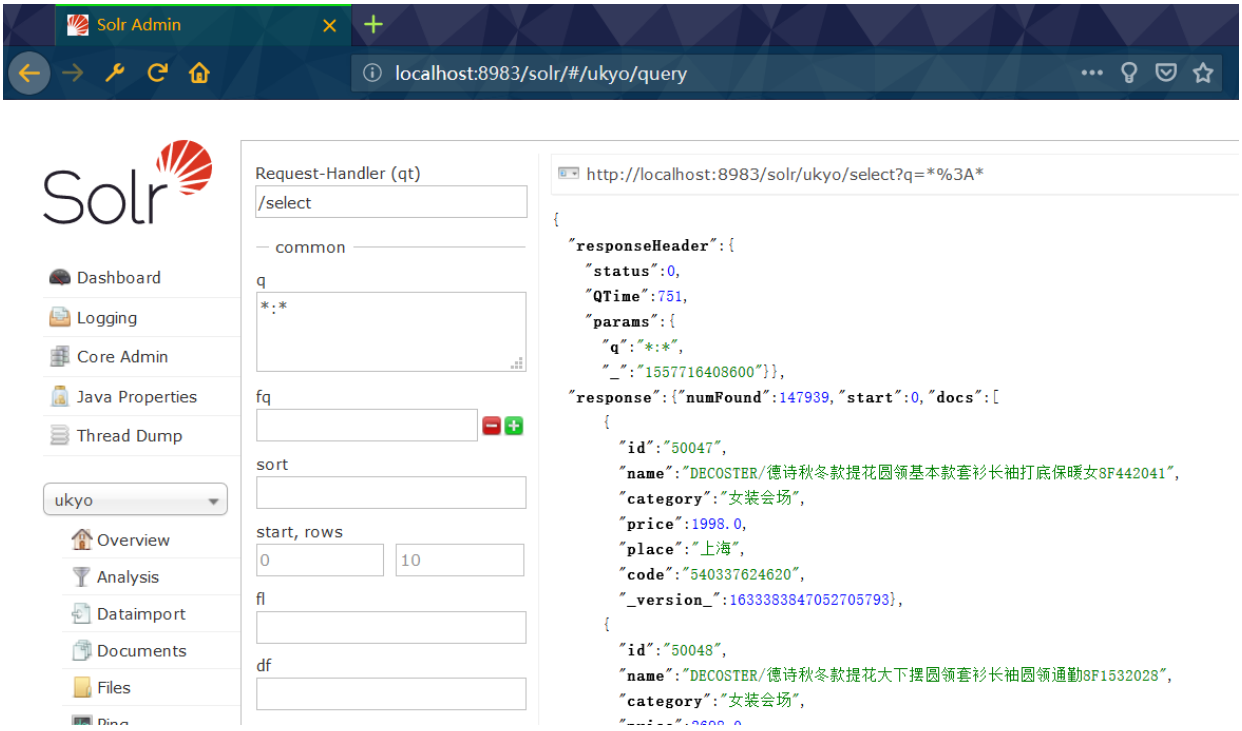
如何创建索引

Solr提供了一种方式向其中增加索引的界面,但是,不太方便,也和实际工作环境不相符合.
实际工作环境一般都是从数据库里读取数据,然后加入到索引的, 很少会通过界面添加索引,
因为这样维护更新删除也不方便,尤其是数据量比较大的时候.
那么,看看如何通过程序把数据加入到Solr索引里.

SolrJ:

Solr支持通过各种各样的语言 (如 php, JavaScript, c#) 把数据加入到索引里, 这里使用第三方工具SolrJ , 使用Java语言来把数据加入到索引里.

使用SolrJ操作Solr

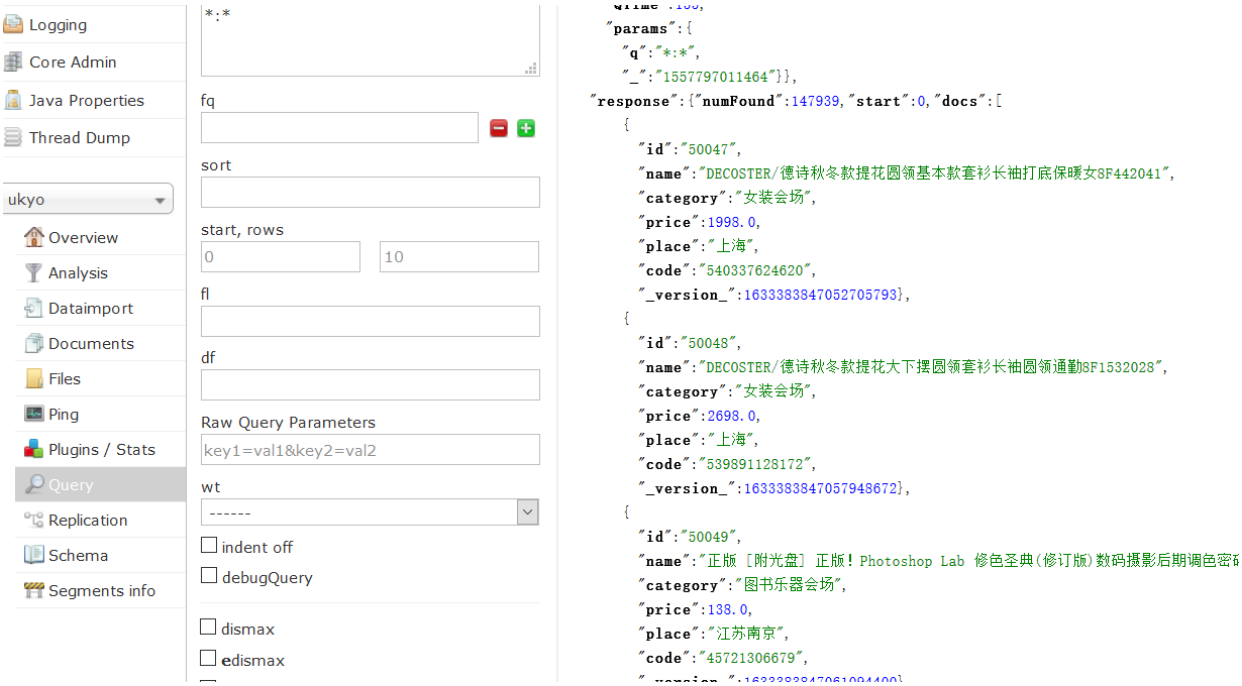


solr4j demo 已推送到github的solr4j

本来应该先把这14万条记录保存进数据库,然后再从数据库中取出来,考虑到14万条数据从数据库里读取的时间消耗,就改成直接从文件里读取出来,然后转换为泛型是Product的集合的形式,相当于从数据库里读取出来了,不过会快很多.

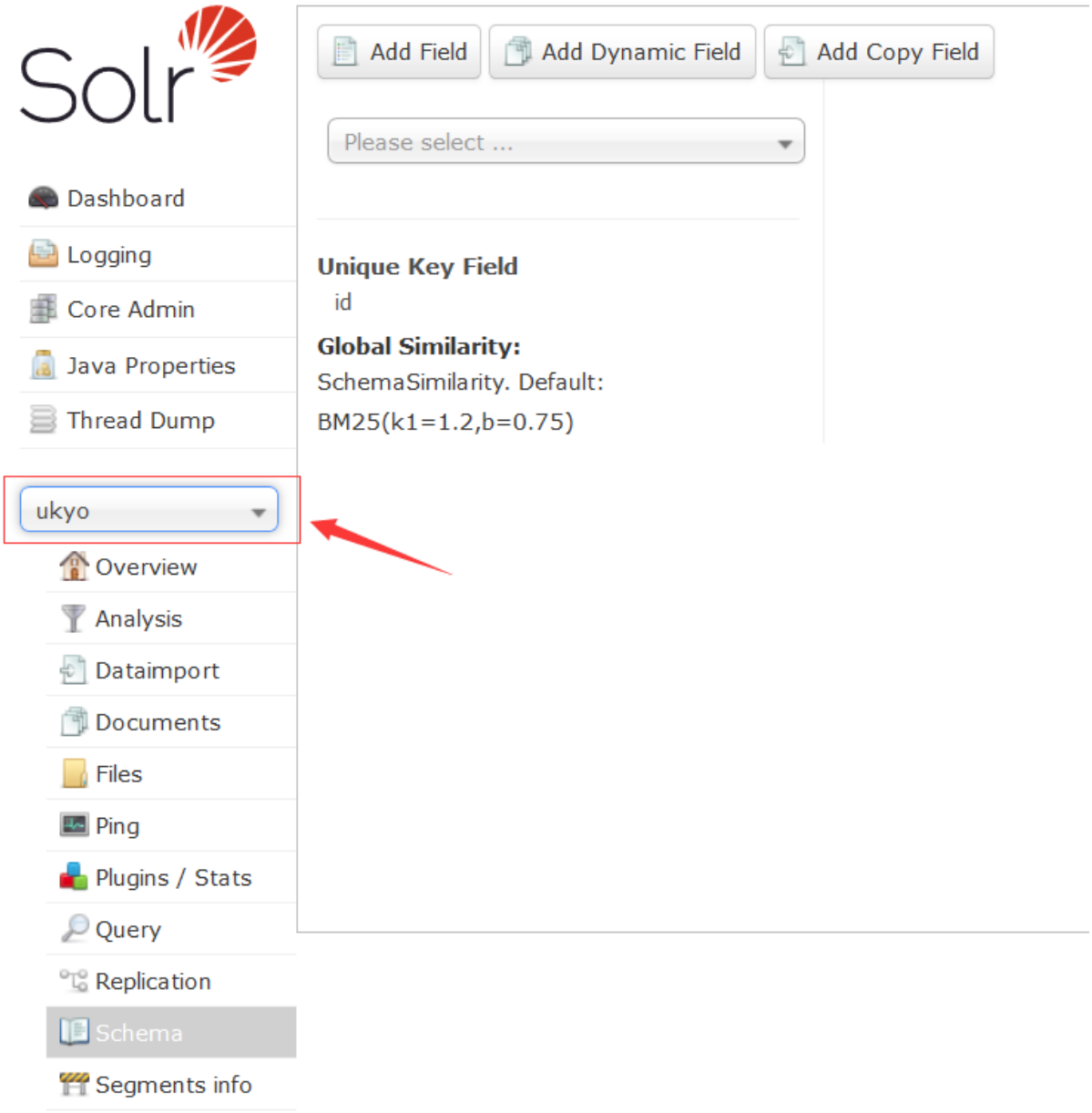
http://how2j.cn/k/search-engine/search-engine-create-index/1682.html 步骤6之后

打开链接看效果.



solr分页查询

分页时调整了分页的逻辑,但会有重复的id的数据出来,还没调整完成.



https://github.com/deadzq/solr4j-page1.git

solr高亮显示

https://github.com/deadzq/solr4j-light.git

solr更新删除索引

修改之前查询一次
修改之后查询一次
删除之后查询一次
之后再更改回来.
https://github.com/deadzq/solr4j-update-delete.git

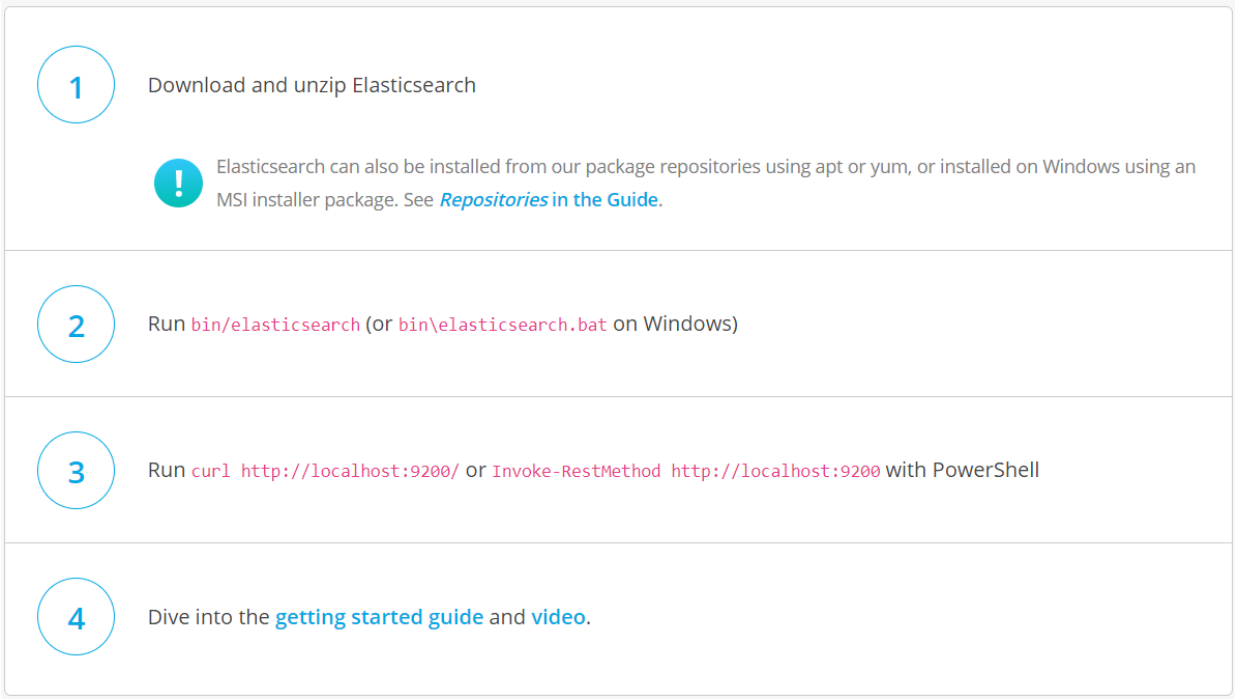
以上就是Solr的入门级用法,更进一步的学习,请进入Solr官网学习:
https://lucene.apache.org/solr/

lucene和solr的测试demo包
链接：https://pan.baidu.com/s/1NsEXZypi92pKtAblNUBOkQ
提取码：a64o
复制这段内容后打开百度网盘手机App，操作更方便哦
lucene和solr的基本告一段落.重点看下面的ElasticSearch~

> 什么是ElasticSearch

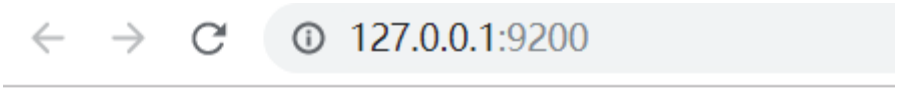
和Solr一样,ElasticSearch也是基于Lucene进行了封装,提供了更为便利的访问和调用.
官网
https://www.elastic.co/cn/
Elastic中文社区
http://www.elasticsearch.cn/

到官网下载压缩包,之后解压.



上图是操作步骤,

1. 介绍




```
{
  "name" : "N_IM2Ey",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "5Sdm-3BkRAmvpVjfne83YA",
  "version" : {
    "number" : "6.2.2",
    "build_hash" : "10b1edd",
    "build_date" : "2018-02-16T19:01:30.685723Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```


里面给出了服务器的基本信息，如版本号是6.2.2, 底层用的lucene版本号是7.2.1 等等
注： 不同浏览器对于json代码的默认渲染方式是不一样的，截图看到的是火狐的渲染方式，其他浏览器可能是下载一个文件，里面呢内容是这些，不过都表示启动成功了。

elasticsearch工具-Kibana
Kibana 是在ElasticSearch 有了相当多的数据之后，进行分析这些数据用的工具。 但是现在还么有数据呀，为什么就要介绍这个工具呢？ 因为Kibana 里面有一个叫做 Dev Tools的，可以很方便地以 Restful 风格向 ElasticSearch 服务器提交请求，接下来的部分学习，都会使用Kibana 里的这个Dev Tools 来讲解，简单又方便
https://www.cnblogs.com/cjsblog/p/9476813.html 详细

Kibana同样是解压后切换到bin下,执行脚本启动
kibana.bat

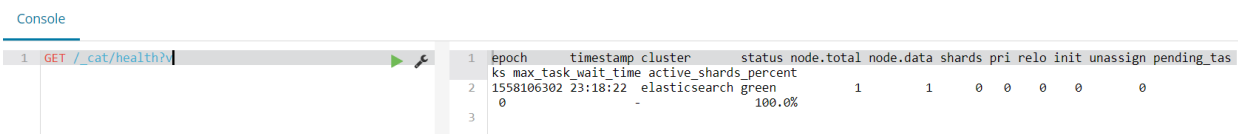
验证启动

```
http://localhost:5601/app/kibana#/dev_tools/console?_g=()
```

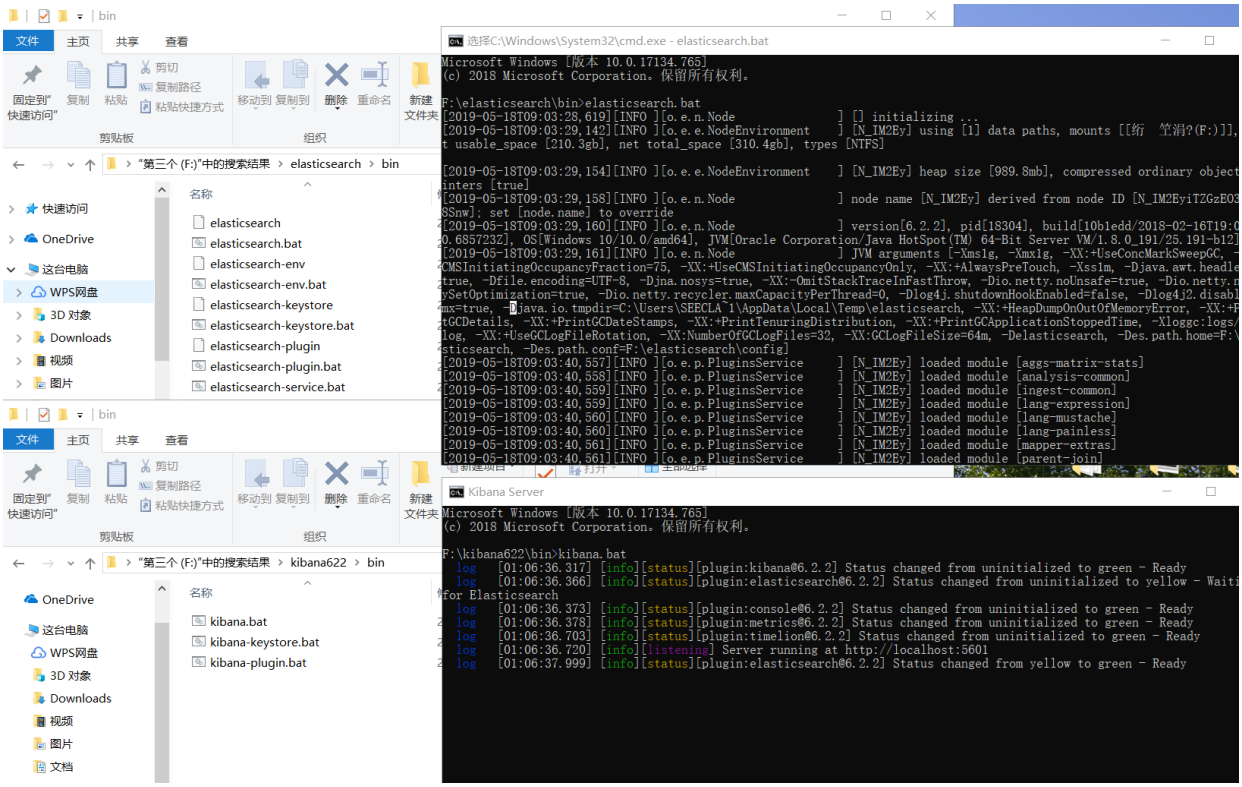
在控制输入

```
GET /_cat/health?v
```

运行测试
然后点击绿色箭头进行运行，就可以看到右侧出现查询结果
GET /_cat/health?v 这个命令用来查看服务器状态（健康度），green 表示一切OK



启动kibana时必须先要开启elasticsearch



elasticsearch-kibana-索引管理

索引概念

索引相当于就是一个数据库服务器上的某个数据库,所以索引也可以看成是Elastic Search里的某个数据库

Restful风格

要进行管理索引的工作: 管理无非就是增删改查,即CRUD.

在使用Restful风格之前,进行索引管理需要这样的访问地址: add,delete,update,get 等不同的访问地址来表示不同的业务请求.

但是使用Restful风格,就通过提交不同的method来表示CRUD:

PUT 表示增加

GET 表示获取

DELETE 表示删除

POST 表示更新

Method是http请求里的一个属性,常用的属性值是post和get.

增加索引

在kibana控制台输入如下命令: (索引的增加是通过kibana)

打开kibana控制台:

```
http://localhost:5601/app/kibana#/dev_tools/console?_g=()
```

运行如下命令:

```
PUT /ukyo?pretty
```

我这次返回的是这样,504

```
{
  "statusCode": 504,
  "error": "Gateway Time-out",
  "message": "Client request timeout"
}
```

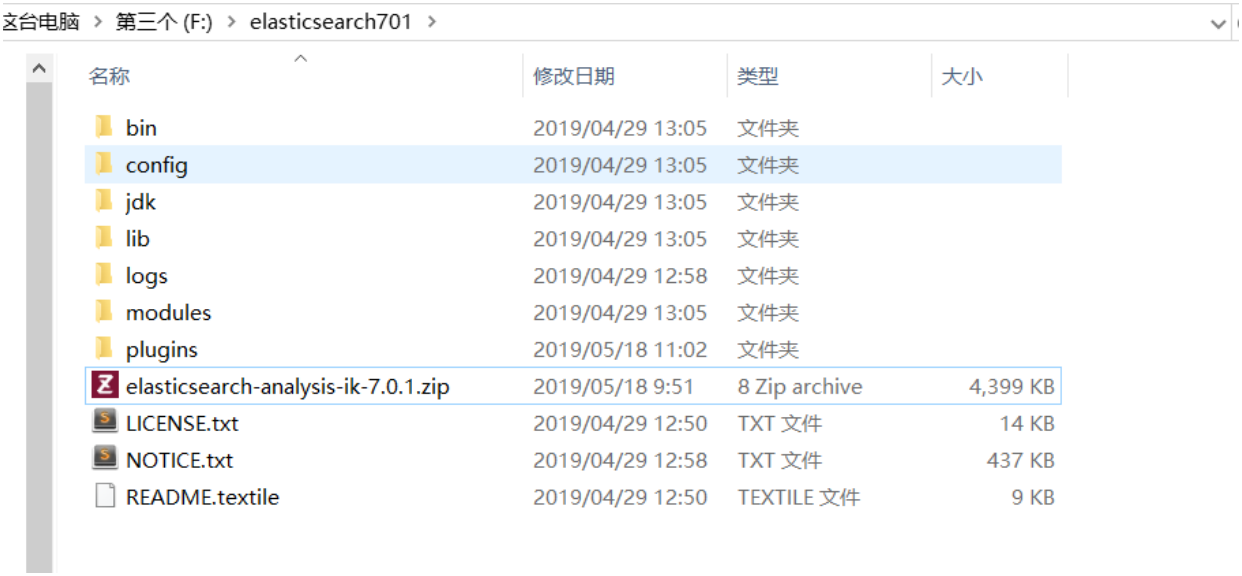
这里的问题之后解决一下了.

elasticsearch中文分词器

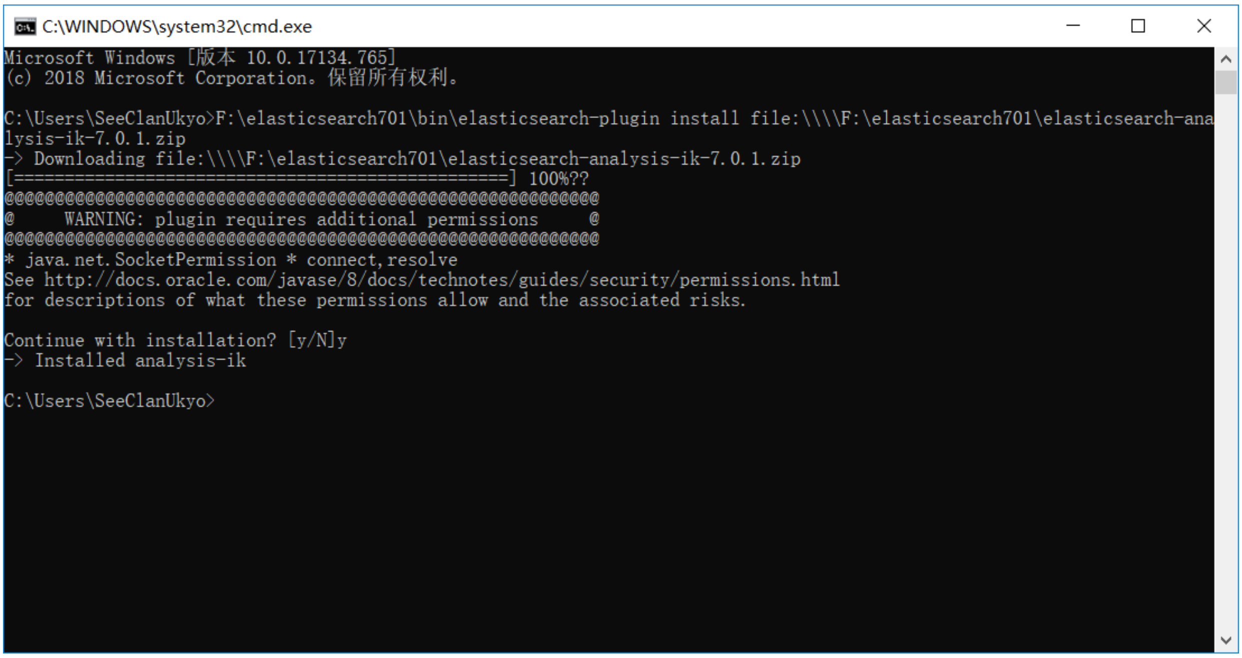
分词器指的是搜索引擎如何使用关键字进行匹配,如入门中的关键字: 护眼带光源.
如果使用like,那么 %护眼带光源% ,匹配出来的结果就是要么全匹配,要不就不匹配.
而使用分词器,就会把这个关键字分为护眼, 带, 光源 3个关键字, 这样就可以找到不同相关程度的结果了.

安装elasticsearch中文分词器

对应elasticsearch版本 安装对应版本的分词器



如图,进入文件夹,放入中文分词器的zip包.

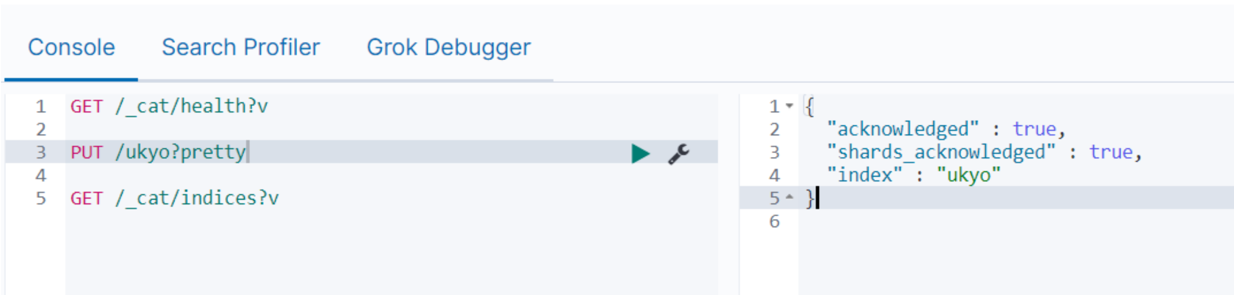



```
C:\Users\SeeClanUkyo>F:\elasticsearch701\bin\elasticsearch-plugin install
 file:\\\\F:\elasticsearch701\elasticsearch-analysis-ik-7.0.1.zip
```

执行以上命令,如图中所示即安装完成.

安装插件后要重启,否则无法生效.
重启很简单,就是黑窗口x掉,然后重新运行cmd: elasticsearch.bat

在使用了新的es版本后,加入索引问题得到了解决



查看索引

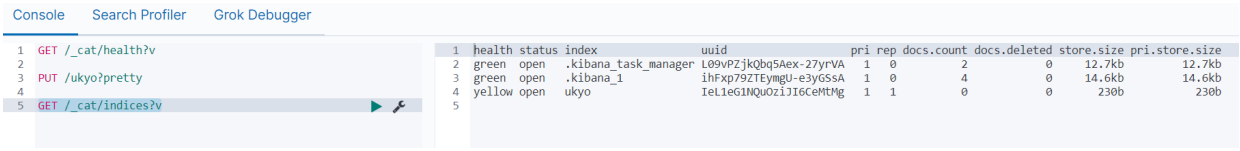
```
GET /_cat/indices?v
```

索引相当于就是一个数据库服务器上的某个数据库,所以索引也可以看成是Elastic Search里的某个数据库

索引相当于就是一个数据库服务器上的某个数据库,所以索引也可以看成是Elastic Search里的某个数据库

索引相当于就是一个数据库服务器上的某个数据库,所以索引也可以看成是Elastic Search里的某个数据库

执行下图命令,可以看到新建立的索引.



删除索引

```
DELETE /ukyo?pretty
```

删除成功返回:

```
{
  "acknowledged" : true
}
```

删除失败返回:

```
{
  "error" : {
    "root_cause" : [
      {
        "type" : "index_not_found_exception",
        "reason" : "no such index [ukyo]",
        "resource.type" : "index_or_alias",
        "resource.id" : "ukyo",
        "index_uuid" : "_na_",
        "index" : "ukyo"
      }
    ],
    "type" : "index_not_found_exception",
    "reason" : "no such index [ukyo]",
    "resource.type" : "index_or_alias",
    "resource.id" : "ukyo",
    "index_uuid" : "_na_",
    "index" : "ukyo"
  },
  "status" : 404
}
```

继续ES的中文分词器

在kibana的console中执行

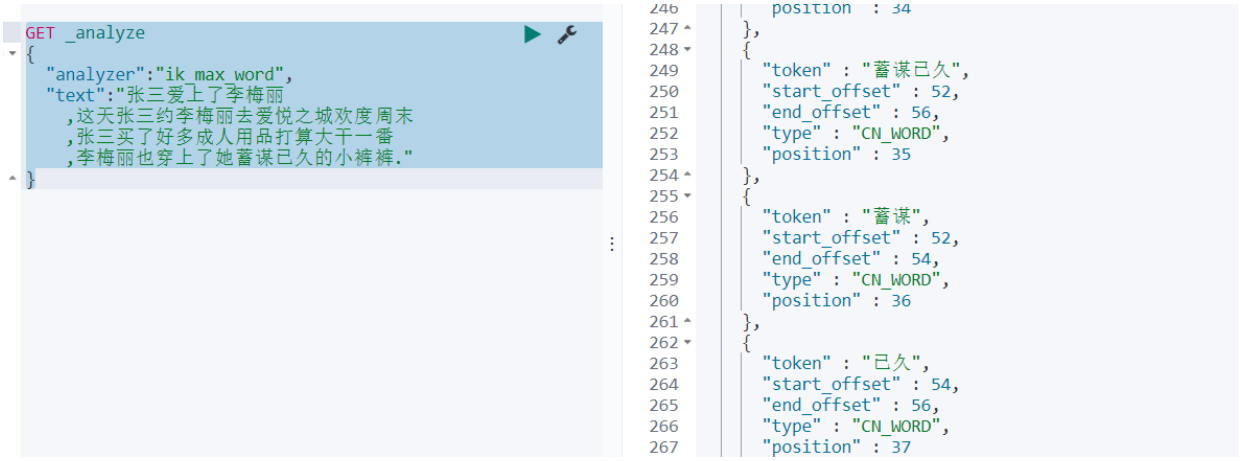
```
GET _analyze
{
  "analyzer":"ik_max_word",
  "text":"护眼带光源"
}
```

返回:

```
{
  "tokens" : [
    {
      "token" : "护眼",
      "start_offset" : 0,
      "end_offset" : 2,
      "type" : "CN_WORD",
      "position" : 0
    },
    {
      "token" : "带",
      "start_offset" : 2,
      "end_offset" : 3,
      "type" : "CN_CHAR",
      "position" : 1
    },
    {
      "token" : "光源",
      "start_offset" : 3,
      "end_offset" : 5,
      "type" : "CN_WORD",
```

```
        "position" : 2
      }
    ]
  }
```

总体的分词效果还算可以:



```
GET _analyze
{
  "analyzer":"ik_max_word",
  "text":"张三爱上了李梅丽
    ,这天张三约李梅丽去爱悦之城欢度周末
    ,张三买了好多成人用品打算大干一番
    ,李梅丽也穿上了她蓄谋已久的小裤裤."
}
```

```
    "position" : 34
  },
  {
    "token" : "蓄谋已久",
    "start_offset" : 52,
    "end_offset" : 56,
    "type" : "CN_WORD",
    "position" : 35
  },
  {
    "token" : "蓄谋",
    "start_offset" : 52,
    "end_offset" : 54,
    "type" : "CN_WORD",
    "position" : 36
  },
  {
    "token" : "已久",
    "start_offset" : 54,
    "end_offset" : 56,
    "type" : "CN_WORD",
    "position" : 37
```

---

elasticsearch Kibana - 文档管理

步骤1. 增加文档
步骤2. 获取文档
步骤3. 修改文档1
步骤4. 修改文档2
步骤5. 删除文档

---

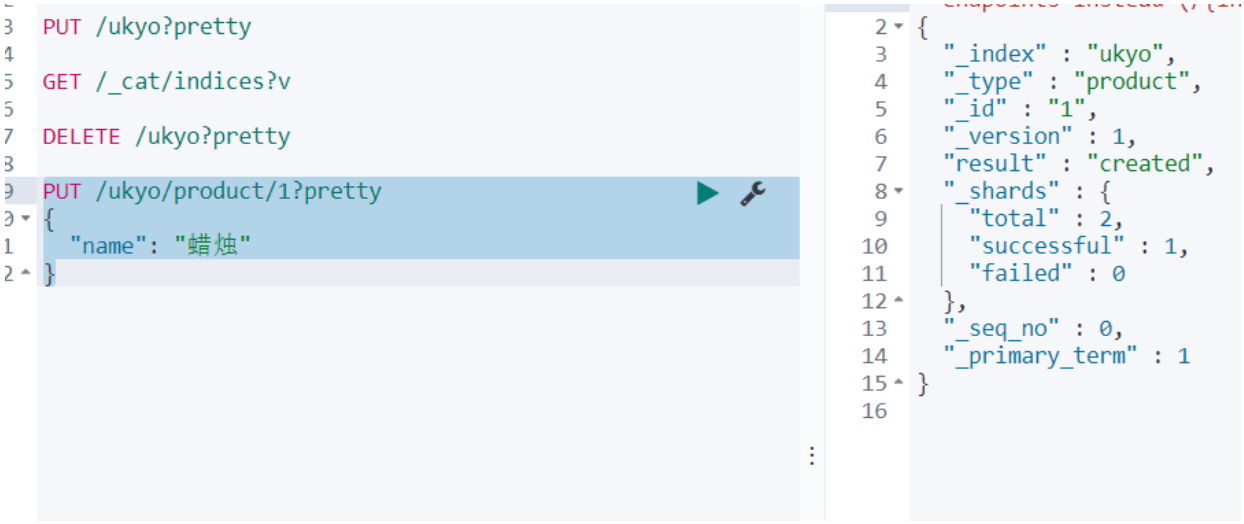
1. 增加文档

还是kibana的控制台

```
http://localhost:5601/app/kibana#/dev_tools/console?_g=()
```

运行如下命令:

```
PUT /ukyo/product/1?pretty
{
  "name": "蜡烛"
}
```



如图所示,增加成功.
注意!!!: 其中的product在elastic search里是type的概念,相当于数据库里的表,这里就相当于向product表里插入了一条数据.

2.获取文档

```
#获取文档
GET /ukyo/product/1?pretty
```

正常返回:

```
{
  "_index": "ukyo",
  "_type": "product",
  "_id": "1",
  "_version": 1,
  "found": true,
  "_source": {
    "name": "蜡烛"
  }
}
```

_index 表示哪个索引
_type 表示哪个表
_id 主键
_version 版本
found 数据存在
_source 数据内容

```
GET /_cat/health?v

#新增索引ukyo
PUT /ukyo?pretty

#获取索引
GET /_cat/indices?v

#删除索引ukyo
DELETE /ukyo?pretty

#中文语法分析,需要安装中文语法分析插件ik...
GET _analyze
{
  "analyz[redacted]": "ik_max_word"
}

#增加文档
#其中的product在elastic search里是type的概念,相当于数据库里的表,这里就相当于向product表里插入了一条数据
PUT /ukyo/product/1?pretty
{
  "name": "蜡烛"
}

#获取文档
GET /ukyo/product/1?pretty
```

```
requests is deprecated, use the /{ind
instead.
{
  "_index" : "ukyo",
  "_type" : "product",
  "_id" : "1",
  "_version" : 1,
  "_seq_no" : 0,
  "_primary_term" : 1,
  "found" : true,
  "_source" : {
    "name" : "蜡烛"
  }
}
```

来让我再次插入一条文档.查看是否id为2.
获取时拼接2获取.

```
}

#获取文档
GET /ukyo/product/1?pretty

#再次添加文档,查看id是否指定
PUT /ukyo/product/2?pretty
{
  "name": "口香糖"
}

#获取文档 id:2 OK
GET /ukyo/product/2?pretty


#再次添加文档,查看id是否指定123
PUT /ukyo/product/123?pretty
{
  "name": "中国博物馆"
}

#获取文档 id:123 OK
GET /ukyo/product/123?pretty
```

```
#! Deprecation: [types removal]
  requests is deprecated, us
{
  "_index" : "ukyo",
  "_type" : "product",
  "_id" : "2",
  "_version" : 1,
  "_seq_no" : 1,
  "_primary_term" : 1,
  "found" : true,
  "_source" : {
    "name" : "口香糖"
  }
}
```

来再次插入一条文档,id为123的.
获取时直接使用123获取也是可以的.

```
#获取文档
GET /ukyo/product/1?pretty

#再次添加文档,查看id是否指定
PUT /ukyo/product/2?pretty
{
  "name": "口香糖"
}

#获取文档 id:2 OK
GET /ukyo/product/2?pretty


#再次添加文档,查看id是否指定123
PUT /ukyo/product/123?pretty
{
  "name": "中国博物馆"
}

#获取文档 id:123 OK
GET /ukyo/product/123?pretty
```

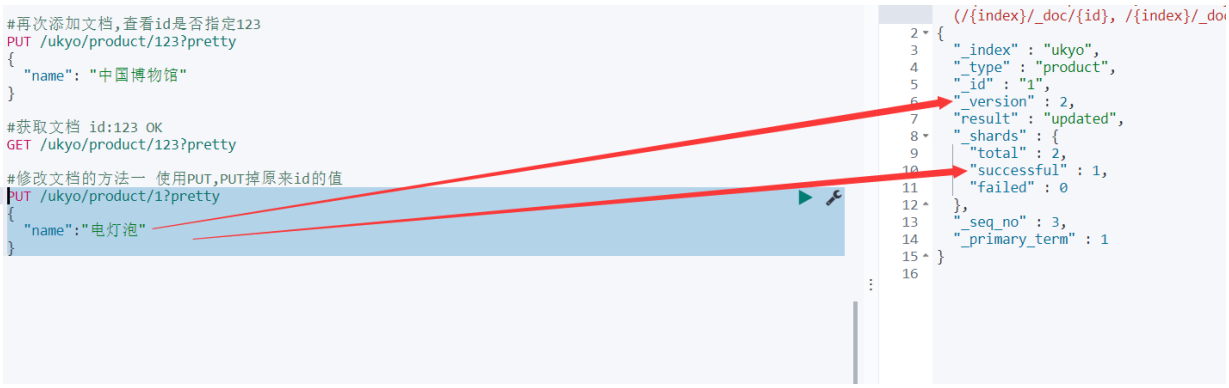
```
{
  "_index" : "ukyo",
  "_type" : "product",
  "_id" : "123",
  "_version" : 1,
  "_seq_no" : 2,
  "_primary_term" : 1,
  "found" : true,
  "_source" : {
    "name" : "中国博物馆"
  }
}
```

3.修改文档method1

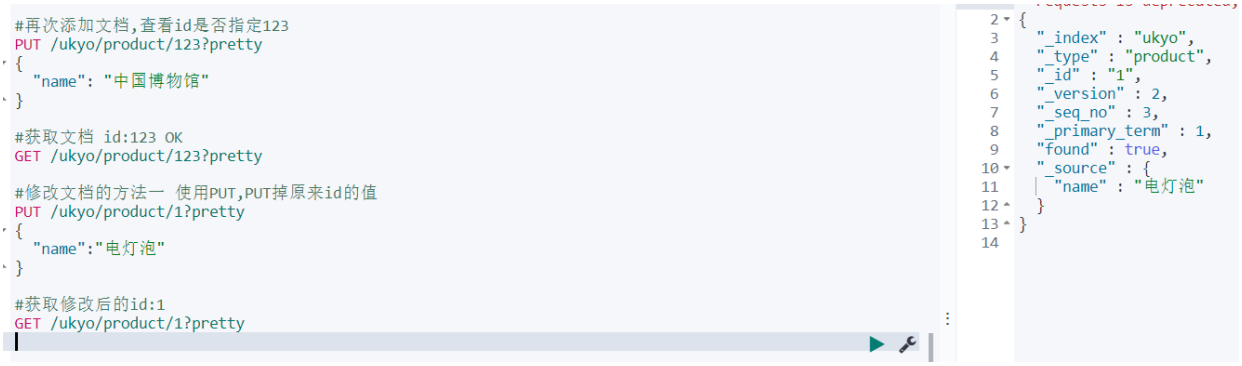
修改两种方式,第一种还是用PUT,PUT本来用来做增加的,但是当输入的id已经存在的时候,就自动变成修改动能了.

```
PUT /ukyo/product/1?pretty
{
    "name":"电灯泡"
}
```

更改显示:



查看后显示:



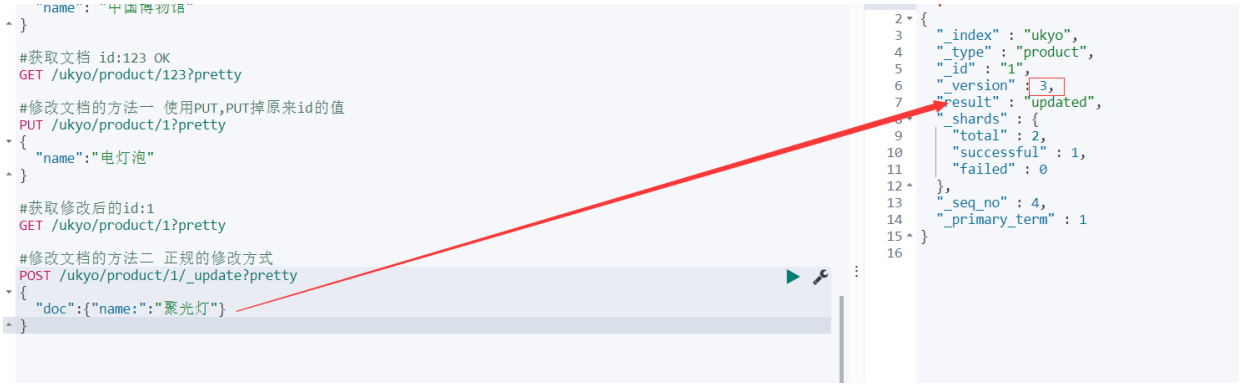
再次获取看看是否已经更改.

4.修改文档method2

修改的第二种方式,使用POST,这才是正规的修改,其实效果和第一种方法是一样的.

```
POST /ukyo/product/1/_update?pretty
{
    "doc":{"name:":"聚光灯"}
}
```

更改显示:



注意上图,在doc的json中name不小心打错了,多了个:号,继而结果成为了这样:

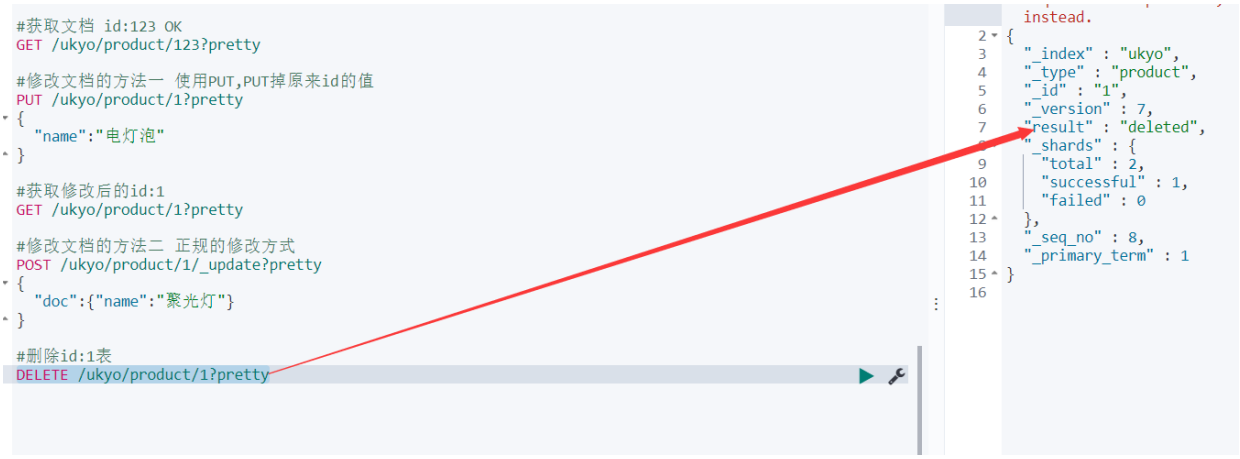
经过删除,再次POST更改后:

再次查看.
注意!!!: 留意其中的版本已经是3了,第一次创建的时候是1,第一次修改时是2,再次修改是3.(包括删除了这个数据,再次使用同id,依旧会递增_id.

5.删除文档

执行删除:

```
DELETE /ukyo/product/1?pretty
```



再次查看id:1
结果:

```
{
  "_index" : "ukyo",
  "_type" : "product",
  "_id" : "1",
  "found" : false
}
```

found为false,说明没有找到.即DELETE命令执行成功.

> **elasticsearch Kibana - 批量导入**

1.批量导入两条数据
还是kibana的控制台

```
http://localhost:5601/app/kibana#/dev_tools/console?_g=()
```

运行如下命令:

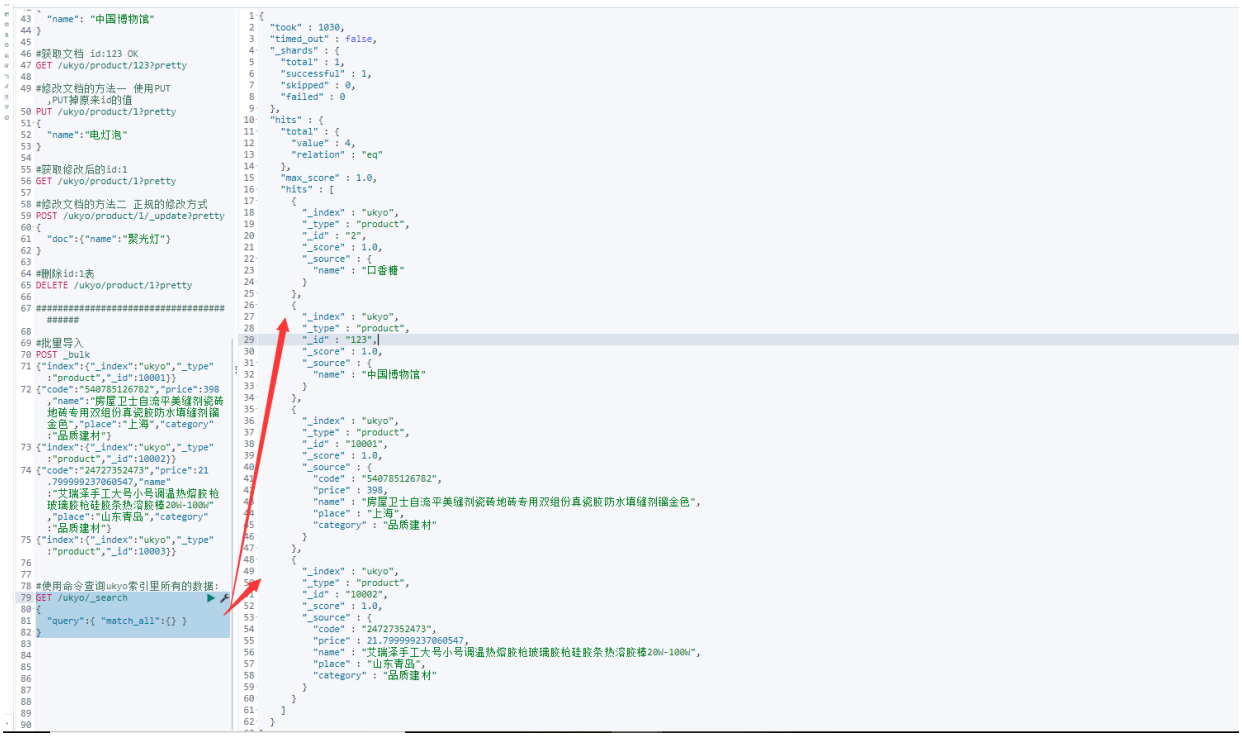
```
POST _bulk
{"index":{"_index":"ukyo","_type":"product","_id":10001}}
{"code":"540785126782","price":398,"name":"房屋卫士自流平美缝剂瓷砖地砖专用双组份真瓷胶防水填缝剂镏金色","place":"上海","category":"品质建材"}
{"index":{"_index":"ukyo","_type":"product","_id":10002}}
{"code":"24727352473","price":21.799999237060547,"name":"艾瑞泽手工大号小号调温热熔胶枪玻璃胶枪硅胶条热溶胶棒20W-100W","place":"山东青岛","category":"品质建材"}
{"index":{"_index":"ukyo","_type":"product","_id":10003}}
```

注意: 其中的product在elastic search里是type的概念,相当于数据库里的表,这里就相当于向product表里插入了一条数据.
成功返回:

```
{
  "took" : 427,
  "errors" : false,
  "items" : [
    {
      "index" : {
        "_index" : "ukyo",
        "_type" : "product",
        "_id" : "10001",
        "_version" : 1,
        "result" : "created",
```

```
            "_shards" : {
              "total" : 2,
              "successful" : 1,
              "failed" : 0
            },
            "_seq_no" : 9,
            "_primary_term" : 1,
            "status" : 201
          }
        },
        {
          "index" : {
            "_index" : "ukyo",
            "_type" : "product",
            "_id" : "10002",
            "_version" : 1,
            "result" : "created",
            "_shards" : {
              "total" : 2,
              "successful" : 1,
              "failed" : 0
            },
            "_seq_no" : 10,
            "_primary_term" : 1,
            "status" : 201
          }
        }
    ]
}
```

2.验证插入的数据



3.局限性
使用这种方式能够插入的上限较小,在前面的Lucene和Solr教程里,都有14万条的数据要插入,用这种方式就插不进去了.
那么就会使用curl.exe来插入.

> elasticsearch curl - 批量导入

curl.exe
curl是一个工具,可以模拟浏览器向服务器提交数据.
需要把curl.exe和要执行导入的数据放到同一个目录下.

上一步撤销重做了,因为电脑系统重装,东西没有找到
访问地址http://localhost:9200/还是发现

```
{
  "name" : "LAPTOP-AFF8817V",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "h4HJ6f0GT4WtfyQ8AYQnNA",
  "version" : {
    "number" : "7.0.1",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "e4efcb5",
    "build_date" : "2019-04-29T12:56:03.145736Z",
    "build_snapshot" : false,
    "lucene_version" : "8.0.0",
    "minimum_wire_compatibility_version" : "6.7.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

es运行成功,再次运行kibana

curl批量导入成功

```
hards":{"total":2,"successful":1,"failed":0},"_seq_no":147933,"_primary_term":4,"status":201}},{"index":
,"_type":"product","_id":"157924","_version":1,"result":"created","forced_refresh":true,"_shards":{"tota
":1,"failed":0},"_seq_no":147934,"_primary_term":4,"status":201}},{"index":{"_index":"ukyo","_type":"pro
925","_version":1,"result":"created","forced_refresh":true,"_shards":{"total":2,"successful":1,"failed":
935,"_primary_term":4,"status":201}},{"index":{"_index":"ukyo","_type":"product","_id":"157926","_versio
reated","forced_refresh":true,"_shards":{"total":2,"successful":1,"failed":0},"_seq_no":147936,"_primary
":201}},{"index":{"_index":"ukyo","_type":"product","_id":"157927","_version":1,"result":"created","forc
,"_shards":{"total":2,"successful":1,"failed":0},"_seq_no":147937,"_primary_term":4,"status":201}},{"ind
kyo","_type":"product","_id":"157928","_version":1,"result":"created","forced_refresh":true,"_shards":{"
sful":1,"failed":0},"_seq_no":147938,"_primary_term":4,"status":201}},{"index":{"_index":"ukyo","_type":
"157929","_version":1,"result":"created","forced_refresh":true,"_shards":{"total":2,"successful":1,"fail
:147939,"_primary_term":4,"status":201}},{"index":{"_index":"ukyo","_type":"product","_id":"157930","_ve
":"created","forced_refresh":true,"_shards":{"total":2,"successful":1,"failed":0},"_seq_no":147940,"_pri
atus":201}},{"index":{"_index":"ukyo","_type":"product","_id":"157931","_version":1,"result":"created","
true,"_shards":{"total":2,"successful":1,"failed":0},"_seq_no":147941,"_primary_term":4,"status":201}},{
":"ukyo","_type":"product","_id":"157932","_version":1,"result":"created","forced_refresh":true,"_shards
ccessful":1,"failed":0},"_seq_no":147942,"_primary_term":4,"status":201}},{"index":{"_index":"ukyo","_ty
id":"157933","_version":1,"result":"created","forced_refresh":true,"_shards":{"total":2,"successful":1,"
_no":147943,"_primary_term":4,"status":201}},{"index":{"_index":"ukyo","_type":"product","_id":"157934",
sult":"created","forced_refresh":true,"_shards":{"total":2,"successful":1,"failed":0},"_seq_no":147944,"
,"status":201}},{"index":{"_index":"ukyo","_type":"product","_id":"157935","_version":1,"result":"create
sh":true,"_shards":{"total":2,"successful":1,"failed":0},"_seq_no":147945,"_primary_term":4,"status":201
ndex":"ukyo","_type":"product","_id":"157936","_version":1,"result":"created","forced_refresh":true,"_sh
,"successful":1,"failed":0},"_seq_no":147946,"_primary_term":4,"status":201}},{"index":{"_index":"ukyo",
","_id":"157937","_version":1,"result":"created","forced_refresh":true,"_shards":{"total":2,"successful"
_seq_no":147947,"_primary_term":4,"status":201}},{"index":{"_index":"ukyo","_type":"product","_id":"1579
,"result":"created","forced_refresh":true,"_shards":{"total":2,"successful":1,"failed":0},"_seq_no":1479
m":4,"status":201}},{"index":{"_index":"ukyo","_type":"product","_id":"157939","_version":1,"result":"cr
efresh":true,"_shards":{"total":2,"successful":1,"failed":0},"_seq_no":147949,"_primary_term":4,"status"
```


再次查看

还是不行,条数不够,可能是重装时破坏了数据,环境等.
这里重安装kibana,再看.

重装了es和kibana,首先还需要给es安装中文分词插件.

```
F:\>c:

C:\Users\ukyo>F:\elasticsearch-7.0.1\bin\elasticsearch-plugin install file:\\\\F:\elasticsearch-7.0.1\elasticsearch-analysis-ik-7.0.1.zip
-> Downloading file:\\\\F:\elasticsearch-7.0.1\elasticsearch-analysis-ik-7.0.1.zip
[=================================================] 100%??
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@     WARNING: plugin requires additional permissions     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* java.net.SocketPermission * connect,resolve
See http://docs.oracle.com/javase/8/docs/technotes/guides/security/permissions.html
for descriptions of what these permissions allow and the associated risks.

Continue with installation? [y/N]y
-> Installed analysis-ik

C:\Users\ukyo>_
```

安装好后,照样要重启es.

这里我先把elasticsearch目录bin放入到了环境变量,包括kibana的bin目录也是.
在cmd启动时:

```
C:\Users\ukyo>elasticsearch
```

照样,启动kibana也是直接输入kibana即可.顺序还是先es再kibana否则kibana是会报错的.

之后还是看5601端口

```
http://localhost:5601/app/kibana#/dev_tools/console?_g=()
```

继续执行批量导入
curl.exe和products.json还是在桌面目录中.

命令:进入curl.exe的目录中

```
curl -H "Content-Type: application/json" -XPOST "localhost:9200/ukyo/prod
uct/_bulk?refresh" --data-binary "@products.json"
```

批量导入还是不行,debug一下

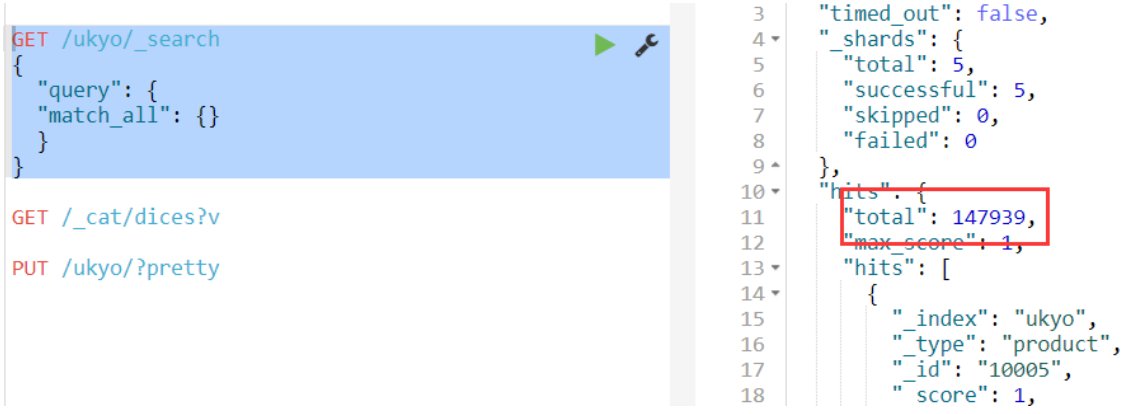
首先我删除了名为ukyo的索引,再次生成.

虽然拿小数据json(截取的json),做了测试,但是最终效果不是很好.

然后试了下查询方式,居然能查到总数据的最后一条,这可能是es和kibana的版本问题.


```
GET /ukyo/product/_search
{
  "query": {

    "match": {
      "_id": "157939"
    }
  }
}
```

再又解压es6.2.2版本和kibana6.2.2版本后,导入数据查询total没问题了.



> ELASTICSEARCH - KIBANA 查询操作 包括倒排序 条件查询 分页查询

查询所有

```
GET /ukyo/_search
{
    "query":{"match_all":{}}
}
```

id倒排序

命令:

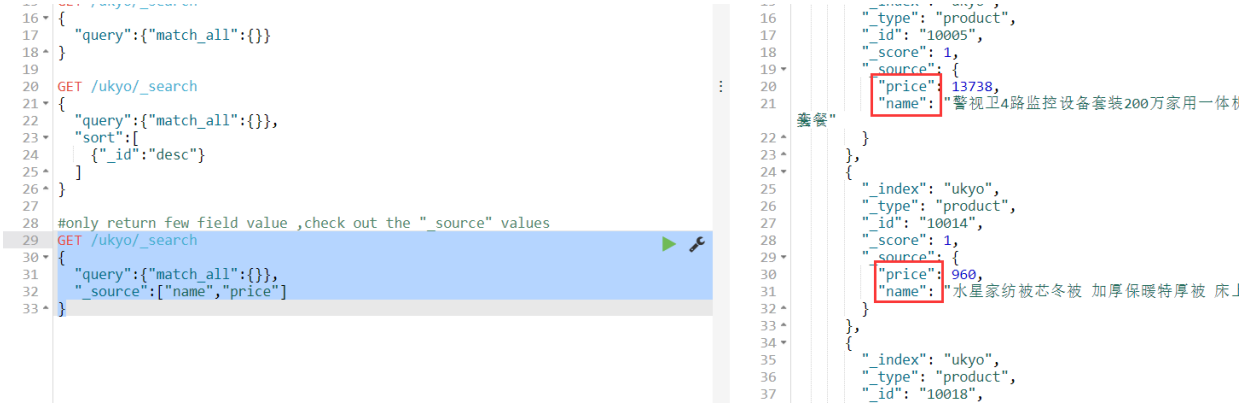
```
GET /ukyo/_search
{
    "query":{"match_all":{}},
    "sort":[
        {"_id":"desc"}
    ]
}
```

查出来与教程一样.

只返回部分字段

使用命令:

```
GET /ukyo/_search
{
    "query":{"match_all":{}},
    "_source":["name","price"]
}
```



条件查询

```
GET /ukyo/_search
{
    "query":{"match":{"name":"时尚连衣裙"}}
}
```

分页查询

```
GET /ukyo/_search
{
    "query":{"match_all":{}},
    "from":1,
    "size":3,
    "sort":{"_id":{"order":"desc"}}
}
```

> ELASTICSEARCH - KIBANA 聚合操作

统计数据

```
GET /ukyo/_search
{
    "size": 0,
    "aggs":{
        "group_by_place":{
            "terms":{
                "field":"place.keyword",
                "size": 3
            }
        }
    }
}
```

上面的这个语法相当于sql语句:

```
select count(*),place from product group by place limit 0,3
```

返回如图所示显示统计了3条数据.

> elasticsearch JavaAPI

以上内容都是使用kibana这个工具实现的. 可是实际开发中,肯定会通过变成语言来实现, 那么接下来的教程会讲解如何通过 Java API 实现对 ElasticSearch 的管理.

ELASTICSEARCH - JAVAAPI 索引管理
试了下使用该程序是可以生成索引的.

ELASTICSEARCH - JAVAAPI 文档管理

https://github.com/deadzq/es4j-doc

```
        //准备数据
        Product product = new Product();
        product.setId(1);
        product.setName("product 1");

        //增加文档
        addDocument(product);

        //获取文档
        getDocument(1);

        //修改数据
        product.setName("producct 2");
        //修改文档
        updateDocument(product);
        //获取文档
        getDocument(1);

        //删除文档
```

```
        deleteDocument(1);
        //获取文档
        getDocument(1);
```

文档管理也是能操作到对象数据的.

ELASTICSEARCH JAVAAPI - 批量操作

https://github.com/deadzq/es4j-batch-insert

批量插入操作.

ELASTICSEARCH JAVAAPI - 查询

https://github.com/deadzq/es4j-search

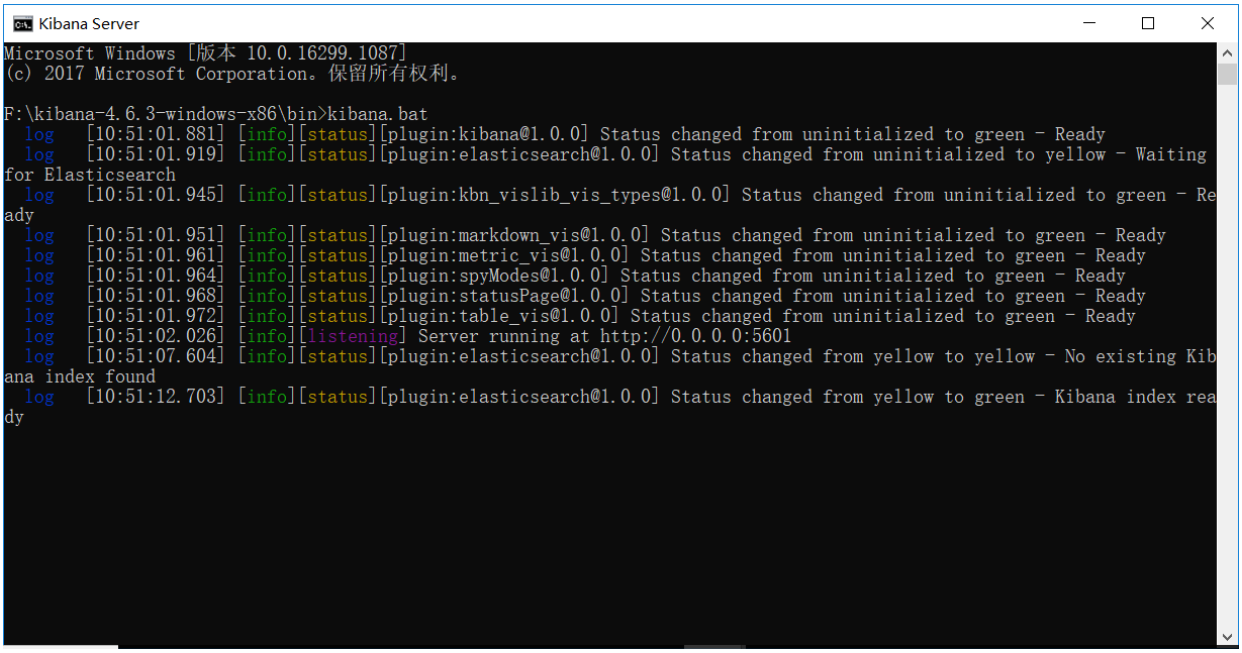
> springboot融合elasticsearch

版本问题
springboot有一个spring data组件,可以用来连接各种数数据源. 用来连接elasticsearch的是spring-data-elasticsearch.
但是,截至到2018-9-17, spring-data-elasticsearch更新比较慢,其最高版本无法支持教程里的elasticsearch 6.x 版本.
为了支持6.x版本,需要用到奇怪的transportclient来连接,这就不是spring-data系列里的内容了.
考虑到将来,spring data总会支持最新版本的elasticsearch的. 所以我们还是使用 spring-data 来进行链接. 只是, elasticsearch的版本, 我们换成了2.4.2 .kibana版本,也换成了能够连接elasticsearch 2.4.2 的4.6.3版本.

启动elasticsearch2.4.2
照之前启动方法启动
,但是kibana4.x.x版本启动失败

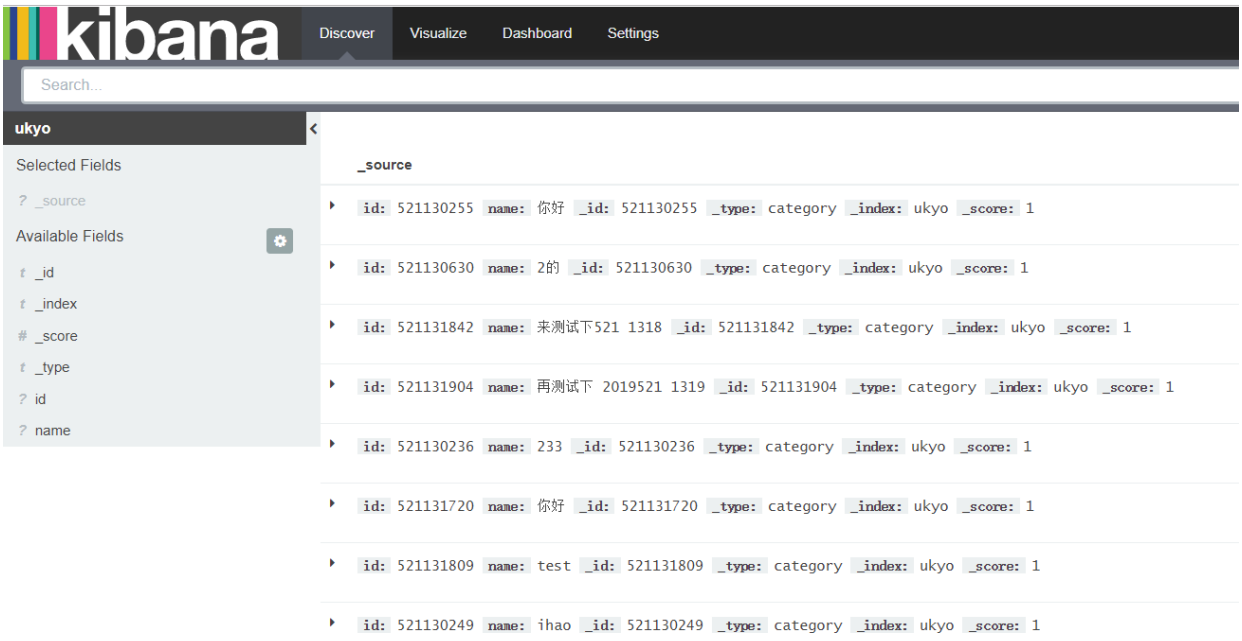
之后在github查找相关demo.

现在记录下以往版本链接
https://www.elastic.co/downloads/past-releases/kibana-4-6-3
看上面的链接,最后的kibana-4-6-3 如果改为elasticsearch-4-6-3照样能下载es的老版本



更换了es版本和相应的kibana版本(463),启动项目后无错.

调节好了jsp的问题后,
可以通过web页面向es,kibana添加索引:



https://github.com/deadzq/es-demo 上传的demo