# ElasticSearch简介

ElasticSearch是一个基于Lucene的搜索服务器。它提供了一个分布式多用户能力的全文搜索引擎，基于RESTful web接口。Elasticsearch是用Java开发的，并作为Apache许可条款下的开放源码发布，是当前流行的企业级搜索引擎。设计用于[云计算](https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97" \t "_blank)中，能够达到实时搜索，稳定，可靠，快速，安装使用方便。

我们建立一个网站或应用程序，并要添加搜索功能，但是想要完成搜索工作的创建是非常困难的。我们希望搜索解决方案要运行速度快，我们希望能有一个零配置和一个完全免费的搜索模式，我们希望能够简单地使用JSON通过HTTP来索引数据，我们希望我们的搜索服务器始终可用，我们希望能够从一台开始并扩展到数百台，我们要实时搜索，我们要简单的多租户，我们希望建立一个云的解决方案。因此我们利用Elasticsearch来解决所有这些问题及可能出现的更多其它问题。

官网链接：https://www.elastic.co/cn/products

# 一、Elasticsearch方法的使用及源代码：

## 1.Elasticsearch模块的安装与引用：

Python环境中，第一步需要安装相对应的elasticsearch模块，pip install elasticsearch，

然后在文件中引用from elasticsearch import Elasticsearch

## 2.Elasticsearch的连接

obj = ElasticSearchClass("59.110.41.175", "9200", "", "")

其中ElasticSearchClass里面是elasticsearch的的一些常用方法：

## 3.ElasticSearch的源代码：

class ElasticSearchClass(object):

def \_\_init\_\_(self, host, port, user, passwrod):

self.host = host

self.port = port

self.user = user

self.password = passwrod

self.connect()

def connect(self):

“””客户端的连接”””

self.es = Elasticsearch(hosts=[{'host': self.host, 'port': self.port}],

http\_auth=(self.user, self.password ))

def insertDocument(self, index, type, body, id=None):

'''

插入一条数据body到指定的index、指定的type下;可指定Id,若不指定,ES会自动生成

:param index: 待插入的index值

:param type: 待插入的type值

:param body: 待插入的数据 -> dict型

:param id: 自定义Id值

:return:

'''

return self.es.index(index=index, doc\_type=type, body=body, id=id)

def count(self, indexname):

"""

:param indexname:

:return: 统计index总数

"""

return self.conn.count(index=indexname)

def delete(self, indexname, doc\_type, id):

"""

:param indexname:

:param doc\_type:

:param id:

:return: 删除index中具体的一条

"""

self.es.delete(index=indexname, doc\_type=doc\_type, id=id)

def get(self, doc\_type, indexname, id):

return self.es.get(index=indexname,doc\_type=doc\_type, id=id)

def searchindex(self, index):

"""

查找所有index数据

"""

try:

return self.es.search(index=index)

except Exception as err:

print(err)

def searchDoc(self, index=None, type=None, body=None):

'''

查找index下所有符合条件的数据

:param index:

:param type:

:param body: 筛选语句,符合DSL语法格式

:return:

'''

return self.es.search(index=index, doc\_type=type, body=body)

def search(self,index,type,body,size=10,scroll='10s'):

"""

根据index，type查找数据，

其中size默认为十条数据，可以修改为其他数字，但是不能大于10000

"""

return self.es.search(index=index, doc\_type=type,body=body,size=size,scroll=scroll)

def scroll(self, scroll\_id, scroll):

"""

根据上一个查询方法，查询出来剩下所有相关数据

"""

return self.es.scroll(scroll\_id=scroll\_id, scroll=scroll)

# 二、Elasticsearch的基本操作：

## 1.asticsear的连接

obj = ElasticSearchClass("59.110.41.175", "9200", "", "")

连接elasticsearch客户端

## 2.数据的的插入

obj.insertDocument(index=”question”,type='text,id=9,body={"any":body,"timestamp":datetime.now()})

其中index和type是固定传入，id可以自己传入也可以系统生成，其中body数据为自己组合的数据

## 3.数据的删除

dd = obj.delete(index='question', type='text', id=7310)

数据删除时候是根据id进行删除，删除数据时候，index，type需要和之前传入时候的index，type保持一致

## 4. 数据的搜索

其中，搜索之后数据显示默认为十条数据

### 4.1、通过index搜索数据

res = obj.search(indexname=index)

### 4.2、通过body搜索数据

#### 4.2.1、全部匹配：

# 查询所有数据

body = {

"query":{

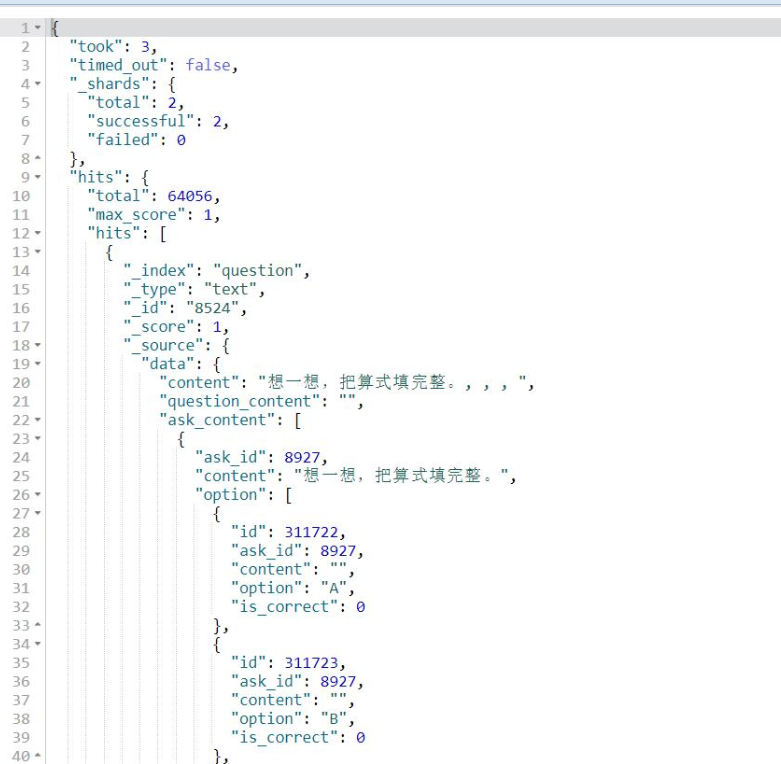
"match\_all":{}

}

}

response = obj.search(index="question",type="text",body=body)

返回的数据默认显示为十条数据，其中hits[“total”]为查询数量总数



其中Match\_all 默认匹配所有的数据

#### 4.2.2、广泛匹配某个字段

body = {

"query" : {

"match" : {

"data.content" : "一根铁丝"

}

}

}

Match默认匹配某个字段

response = obj.search(index="question",type="text",body=body)

返回结果：



#### 4.2.3、匹配多个字段

body = {

"query": {

"bool": {

"should": [

{ "match": { "data.content": "一根铁丝" }},

{ "match": { "data.question\_content": "一根铁丝" }},

{ "match": { "data.ask\_content.content": '一根铁丝' }}

],

}

}

}

Should或匹配可以匹配某个字段也可以匹配所有字段，其中至少有一个语句要匹配，与 OR 等价

response = obj.search(index="question",type="text",body=body,scroll='5s')

#### 4.2.4、匹配所有字段

body = {

"query": {

"bool": {

"must": [

{ "match": { "data.content": "李阿姨" }},

{ "match": { "data.question\_content": "李阿姨" }},

{ "match": { "data.ask\_content.content": '李阿姨' }}

],

}

}

}

Must必须匹配所有需要查询的字段

response = obj.search(index="question",type="text",body=body,scroll='5s')

返回结果



#### 4.2.5、短语匹配查询：

精确匹配一系列单词或者短语

body = {

"query" : {

"match\_phrase" : {

"data.content" : "一根铁丝"

}

}

}

response = obj.search(index="question",type="text",body=body,scroll='5s')

返回结果：



#### 4.2.6、高亮搜索：

许多应用都倾向于在每个搜索结果中 *高亮* 部分文本片段，以便让用户知道为何该文档符合查询条件。在 Elasticsearch 中检索出高亮片段也很容易。

再次执行前面的查询，并增加一个新的 highlight 参数：

Body = {

"query" : {

"match\_phrase" : {

"about" : "rock climbing"

}

},

"highlight": {

"fields" : {

"about" : {}

}

}

}

当执行该查询时，返回结果与之前一样，与此同时结果中还多了一个叫做 highlight 的部分。这个部分包含了 about 属性匹配的文本片段，并以 HTML 标签 <em></em> 封装：

{

...

"hits": {

"total": 1,

"max\_score": 0.23013961,

"hits": [

{

...

"\_score": 0.23013961,

"\_source": {

"data.content": "李阿姨"

},

"highlight": {

"about": [

"张阿姨和<em>李阿姨</em>"

]

}

}

]

}

}

## 5．数据的返回格式

{

...

"hits": {

"total": 1,

"max\_score": 0.23013961,

"hits": [

{

...

"\_score": 0.23013961,

"\_source": {

"字段名1": "XXX",

"字段名2": "XXX",

"字段名3": "XXX",

}

}

]

}

}

# 参考文献

官网参考网址：

<https://www.elastic.co/guide/cn/elasticsearch/guide/current/foreword_id.html>

官方文档

https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html

Elasticsearch——分页查询From&Size VS scroll

https://www.cnblogs.com/xing901022/archive/2016/03/16/5284902.html

Elasticsearch聚合 之 Terms

https://www.cnblogs.com/xing901022/p/4947436.html

elasticsearch 深分页问题以及解决方法

http://blog.csdn.net/wild46cat/article/details/64123353

elasticsearch-利用游标查询 'Scroll'来做分页查询

<http://blog.csdn.net/chuan442616909/article/details/55195024>

ElasticSearch 索引查询使用指南——详细版

https://blog.csdn.net/pilihaotian/article/details/52452014