Rest接口添加

http的put方式

PUT jiechao

{

“settings”:{

“index”:{

“number\_of\_shards”:5,

“number\_of\_replicase”:1

}

}

}

Es的文档，索引的CURE操作（增删改查）

索引的初始化，索引类似关系数据库中的数据库，可以理解为创建数据库。

1. PUT 索引的名称
2. Index里面 number\_of\_shards分片的数量（不能改），number\_of\_replicase副本的数量(可以改)

获取setting信息：

GET jiechao/\_settings

GET \_all/\_settings

GET .kibana,jiechao/\_settings

GET \_settings

更新 ：

PUT jiechao/\_settings

{

“number\_of\_replicase”:3

}

获取所有索引信息

GET \_all

GET jiechao

如何保存文章到索引中来（数据库中保存文章还要建表，然后存入数据库中，在es中不需要建表，在nosql中不需要）

保存文档：

PUT jiechao/job/1(索引/table/id)

{

“title”:”elasticsearch搜索引擎招聘”,

“city”:”重庆”,

“salary\_max”:30000,

“location”:{

“aera”:”江北区”,

“street”:”观音桥步行街工商银行”

}

“publish\_time”:”2017-5-22”,

“comments”:30

}

PUT jiechao/job/

{

“title”:”大数据开发”,

“city”:”重庆”,

“salary\_max”:50000,

“location”:{

“aera”:”渝中区”,

“street”:”轨道三号线”

}

“publish\_time”:”2017-5-22”,

“comments”:30

}

通过rest接口获取：

GET jiechao/job/1

获取某些字段

GET jiechao/job/1?\_source=title,city

获取所有字段

GET jiechao/job/1?\_source

修改文章：

PUT jiechao/job/1(索引/table/id)

{

“title”:”elasticsearch搜索引擎招聘”,

“salary\_max”:30000,

“location”:{

“aera”:”江北区”,

“street”:”观音桥步行街工商银行”

}

“publish\_time”:”2017-5-22”,

“comments”:30

}

第二种方式：

POST jiechao/job/1/\_update

{

“doc”:{

“comments”:20

}

}

删除：

删除文档

DELETE jiechao/job/1

删除type

DELETE jiechao/job

不支持

DELETE jiechao

Elasticsearch批量操作：

\_mget，通过之前GET方法获取某一个document的时候，当数量过多时，效率特别低，因为每次建立一个http连接，开销特别大，http每次建立都会建立三次握手协议。

GET \_mget

{

“docs”:[

{“\_index”:”testdb”,

“\_type”:”job1”,

“\_id”:1

},

{“\_index”:”testdb”,

“\_type”:”job2”,

“\_id”:2

}

]

}

更简单的方式：

GET testdb/\_mget

{

“docs”:[

{“\_type”:”job1”,

“\_id”:1

},

{“\_type”:”job2”,

“\_id”:2

}

]

}

如果type一样：

GET testdb/job1/\_mget

{

“docs”:[

{

“\_id”:1

},

{

“\_id”:2

}

]

}

简写：

GET testdb/job1/\_mget

{

“ids”:[1,2]

}

Bulk批量操作：批量导入可以合并多个操作，比如index,delete,update,create等等，也可以帮助从一个索引导入到另一个索引：

{“index”:{“\_index”:”test”,”\_type”:”type1”,”\_id”:1}}

{“field1”:”value1”}

POST \_bulk操作完成

{“index”:{“\_index”:”test”,”\_type”:”type1”,”\_id”:”1”}}

{“field1”:”value1”}

{“delete”:{“\_index”:”test”,”\_type”:”type1”,”\_id”:”3”}}

{“create”:{“\_index”:”test”,”\_type”:”type1”,”\_id”:”2”}}

{“field2”:”value2”}

{“update”:{“\_id”:”2”,”\_type”:”type2”,”\_index”:”index1”}}

{“doc”:{“field2”:”value2”}}

# 映射

创建索引时可以预先定义字段的类型以及相关属性

Elasticsearch会根据JSON源数据的基础类型猜想你想要的字段映射，将输入的数据转变成可搜索的索引项。Mapping就是我们自己定义的字段的数据类型，同时告诉Elasticsearch如何索引数据以及是否可以被搜索。作用是让索引建立更加细致和完善

|  |  |
| --- | --- |
| **内置类型** |  |
| **String类型** | text,keyword,string类型在es弃用 |
| **数字类型** | Long,integer,short,byte,double,float |
| **日期类型** | date |
| **Bool类型** | boolean |
| **Binary类型（二进制）** | Binary |
| **复杂类型** | object,nested |
| **Geo类型(地理位置)** | geo-point,geo-shape |
| **专业类型** | ip，competion(搜索建议) |

|  |  |  |
| --- | --- | --- |
| **属性** | **描述** | **适合类型** |
| **store** | 值为yes表示存储，为no表示不存储，默认为no | all |
| **index** | yes表示分析，no表示不分析，默认值为true | string |
| **null\_value** | 如果字段为空，可以设置一个默认值，比如”NA” | all |
| **analyzer** | 可以设置索引和搜素时用的分析器，默认使用的是standard分析器 | all |
| **Include\_in\_all** | 默认es为每个文档定义一个特殊域\_all，它的作用是让每个字段被搜索到，如果不想被某个字段搜索到，设置为false | all |
| **format** | 时间格式字符串模式 | date |

实例：

PUT jiechao

{

“mapping”:{

“job”:{

“properties”:{

“title”:{

“type”:”text”

},

“salary\_max”:{

“type”:”integer”

},

“city”:{

“type”:”keyword”

},

“company”:{

“properties”:{

“name”:{

“type”:”text”

},

“company\_addr”:{

“type”:”text”

},

“employee\_count”:{

“type”:”integer”

}

}

},

“Publish\_date”:{

“type”:”date”,

“format”:”yyyy-MM-dd”

}

“comments”:{

“type”:”integer”

}

}

}

}

}

放入数据：

PUT jiechao/job/2

{

“title”:”elasticsearch搜索引擎的开发招聘”,

“salary\_max”:50000,

“city”:”重庆”,

“company”:{

“name”:”重庆说故事有限公司”,

“company\_addr”:”渝中区磁器口”,

“employee\_count”:30

},

“publish\_date”:”2017-5-22”,

“coments”30

}

Elaticsearch查询

Elasticsearch是功能非常强大的搜索引擎，使用它的目的就是为了快速的查询到需要的数据。

查询类型：

基本查询：使用elasticsearch内置查询条件进行查询

组合查询：把多个查询组合在一起进行复合查询

过滤：查询同时，通过filter条件在不影响打分的条件下筛选数据

实例：

添加映射

PUT jiechao

{

“mapping”:{

“job”:{

“properties”:{

“title”:{

“store”:true,

“type”:”text”,

“analyzer”:”ik\_max\_word”

},

“company\_name”:{

“store”:true,

“type”:”keyword”

},

“desc”:{

“type”:”text”

},

“comments”:{

“type”:”integer”

},

“add\_time”:{

“type”:”date”,

“format”:”yyyy-MM-dd”

}

}

` }

}

}

放置四个数据：

POST jiechao/job/

{

“title”:”elasticsearch中文搜索引擎”,

“company\_name”:”湖南科技大学有限公司”,

“desc”:”elasticsearch是基于restful的全文搜索引擎”,

“comments”:30’

“add\_time”:”2017-5-22”

}

POST jiechao/job/

{

“title”:”python 爬虫开发工程师”,

“company\_name”:”上海大数据开发有限公司”,

“desc”:”熟悉python，数据结构，精通数据库”,

“comments”:30’

“add\_time”:”2017-5-22”

}

查询：

match查询(用到最多)

GET lagou/\_search

{

“query”:{

“match”:{

“title”:”python网站”

}

}

}

Ik分词：将python网站分为”python”,”网站”

term查询

GET jiechao/\_search

{

“query”:{

“term”:{

“company\_name”:”百度科技大学”

}

}

}

不会对company\_name字段分词，像关键词一样

terms查询

GET jiechao/\_search

{

“query”:{

“terms”:{

“title”:[“工程师”,”python”,”系统”]

}

}

}

[“工程师”,”python”,”系统”]中任意一个满足就行

控制查询的返回数量(可以做分页)

GET jiechao/\_search

{

“query”:{

“match”:{

“title”:”python”

}

},

“from”:1,

“size”:3

}

“from”:1 从第1个开始,取3个。

match\_all 查询

GET jiechao/\_search

{

“query”:{

“match\_all”:{}

}

}

match\_phrase查询

短语查询

GET /jiechao/\_search

{

“query”:{

“match\_phrase”:{

“title”:{

“query”:”python工程师”,

“slop”:3

}

}

}

}

“query”:”python系统”,必须同时满足”python”和”系统”

“slop”:3 是”python”和”系统”的距离 。如果小于8就无法搜索出来。，大于6就能搜索出来

multi\_match查询

可以指定多个字段

GET jiechao/\_search

{

“query”:{

“multi\_match”:{

“query”:”python”,

“field”:[“title^3”,”desc”]

}

}

}

Title^3设置title的权重比较高，优先搜索

指定返回字段

GET jiechao/\_search

{

“stored\_fields”:[“title”,”company\_name”],

“query”:{

“match”{

“title”:”python”

}

}

}

Strored\_field必须在映射中字段设置stored为true才能返回，否则会出错

通过sort把结果排序：

GET jiechao/\_search

{

“query”:{

“match\_all”:{}

},

“sort”:[{

“comments”:{

“order”:”asc”

}

}

]

}

对评论数排序，”order”:”asc”表示升序 排列，”order”:”desc”表示降序排列

查询范围

range查询

GET jiechao/\_search

{

“query”:{

“range”:{

“comments”:{

“gte”:10,

“lte”:20,

“boost”:2.0

}

}

}

}

对comments字段 范围排序 gte：大于等于，lte：小于等于，gt大于，lt小于，boost权重

GET jiechao/\_search

{

“query”:{

“range”’:{

“addtime”:{

“gte”:”2017-5-22”,

“lte”:”now”

}

}

}

}

大于等于2017-5-22,小于等于当前时间

wildcard查询，模糊查询

GET jiechao/\_search

{

“query”:{

“wildcard”:

{“title”:{“value”:”pyth\*n”,”boost”:2.0}}

}

}

查询title字段里，设置value值，”pyth\*n”匹配字符串通配符，有点像正则表达式，”python”,pythan”,….等等都能被查询到!

elasticsearch的bool组合查询

es 5.X以前的filtered已经被bool替换

用bool 查询包括must should must\_not folter这四种来完成:

Bool:{

“filter”:[],

“must”:[],

“should”:[],

“must\_not”:{},

}

测试：

POST jiechao/testjob/\_bulk

{ “index”:{“\_id”:1}}

{“salary”:10 ,”title”:”Python”}

{“index”:{“\_id”:2}}

{“salary”:20,”title”:”Scrapy”}

{“index”:{“\_id”:3}}

{“salary”:20,”title”:”Django”}

{“index”:{“\_id”:4}}

{“salary”:40,”title”:”Elasticsearch”}

简单的过滤查询：

对应的sql语句:

Select \* from testjob where comments=20

查询薪资为20k的工作

GET jiechao/testjob/\_search

{

“query”:{

“must”{

“match\_all”:{}

}

“filter”:{

“term”:{

“salary”:20

}

}

}

}

}

也可以查询多个值：

GET jiechao/testjob/\_search

{

“query”:{

“must”{

“match\_all”:{}

}

“filter”:{

“terms”:{

“salary”:[10,20]

}

}

}

}

}

Select \* from testjob where title=”Python”

GET jiechao/testjob/\_search

{

“query”:{

“must”{

“match\_all”:{}

}

“filter”:{

“term”:{

“title”:”Python”

}

}

}

}

}

? 为什么查询不出来! “Python”在入库的时候全部转换成小写，查询的时候“Python”就无法查询出来，如果用match查询或者”python”用小写就能成功查询！

查看分析器解析的结果：

GET \_analyze

{

“analyzer”:”ik\_max\_word”,

“text”:”Python网络开发工程师”

}

分词结果为：”Python”,”网络”,”络”,”开发”,“发 “，“工程师”,”工程”,”师”

GET \_analyze

{

“analyzer”:”ik\_smart”,

“text”:”Python网络开发工程师”

}

分词结果为: ”Python”,”网络”,”开发”, “工程师”

Bool过滤查询，可以做组合过滤查询

Select \* from testjob where (salary=20 OR title=Python) and (salary != 30)

数据库这句表示为查询薪资等于20K或者工作为python的工作，排除价格为20k的工作，用es实现：

GET jiechao/testjob/\_search

{

“query”:{

“bool”:{

“should”:[

{“term”:{“salary”:20}}

{“term”:{“title”:”python”}}

]

“must\_not”:{

“term”:{“price”:30}

}

}

}

}

嵌套查询:

select \* from testjob where title=”python” or (title=”elasticsearch” AND salary=30)

GET jiechao/testjob/\_search

{

“query”:{

“bool”:{

“should”:[

{“term”:{“title”:”python”}},

{“bool”:{

“must”:[

{“term”:{“title”:”elasticsearch”}},

{“term”:{“salary”:30}}

]

}

}

]

}

}

}

过滤空和非空数据：

POST jiechao/testjob2/\_bulk

{“index”:{“\_id”:”1”}}

{“tags”:[“search”]}

{“index”:{“\_id”:”2”}}

{“tags”:[search”,”python”]}

{“index”:{“\_id”:”3”}}

{“other\_field”:[“some data”]}

{“index”:{“\_id”:”4”}}

{“tags”:null}

{“index”:{“\_id”:”5”}}

{“tags”:[“search”,null]}

处理null空值的方法：

Select tags from testjob2 where tags is not NULL

GET jiechao/testjob2/\_search

{

“query”:{

“bool”:{

“filter”:{

“exists”:{

“field”:”tags”

}

}

}

}

}

Exists关键词的field 是固定的。

查询不存在的：

GET jiechao/testjob2/\_search

{

“query”:{

“bool”:{

“must\_not”:{

“exists”:{

“field”:”tags”

}

}

}

}

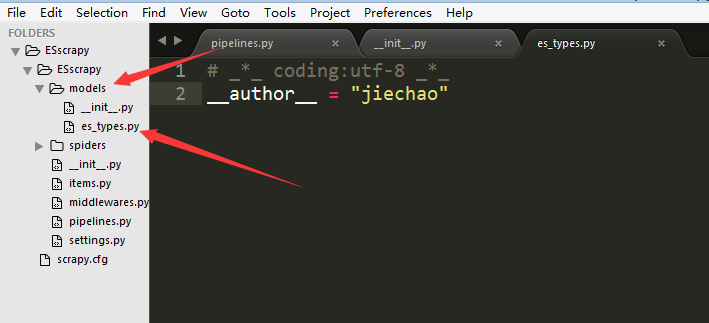
}

# Scrapy抓取的数据爬入es数据中

Elasticsearch的python数据接口：elsticsearch-dsl

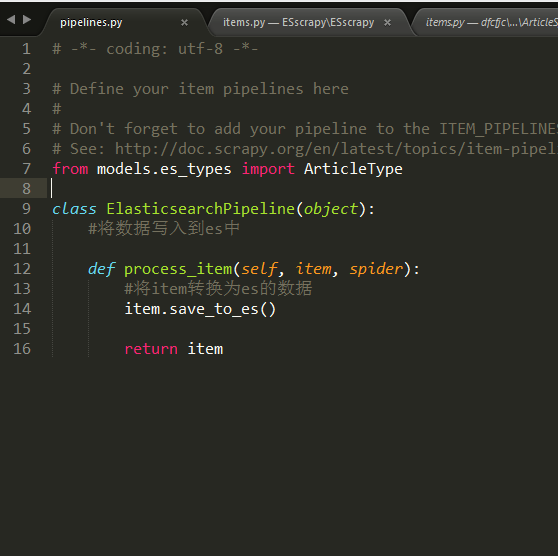
安装方法：pip install elasticsearch-dsl

在spiders的同级目录下建立一个models文件夹，并建立一个es\_types.py文件：

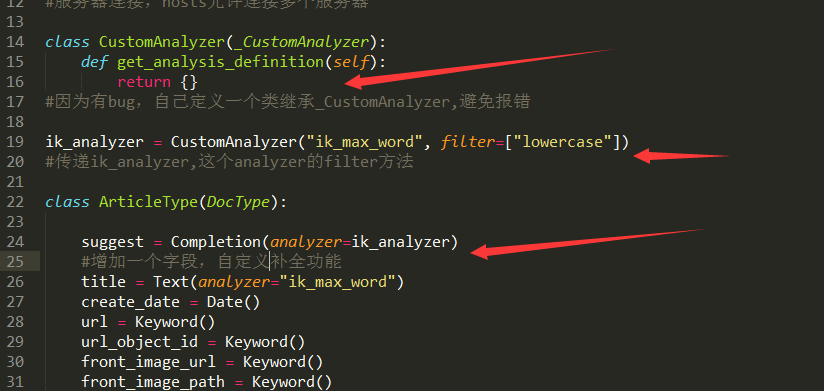








搜索引擎自动补全功能：



es根据文档生成了映射mapping：

PUT music

{

“mappings”:{

“song”:{

“properties”:{

“suggest”:{

“type”:”completion”

},

“title”:{

“type”:”keyword”

}

}

}

}

}

将爬去的数据生成suggest值：

PUT music/song/1?refresh

{

“suggest”:{

“input”:[“Nevermind”,”Nirana”],

“weight”: 34

}

}

Weight：权重

PUT music/song/1?refresh

{

“suggest”:[

{

“input”:”Nevermind”,

“weight”:10

},

{

“input”:”Nirana”,

“weight”:3

}

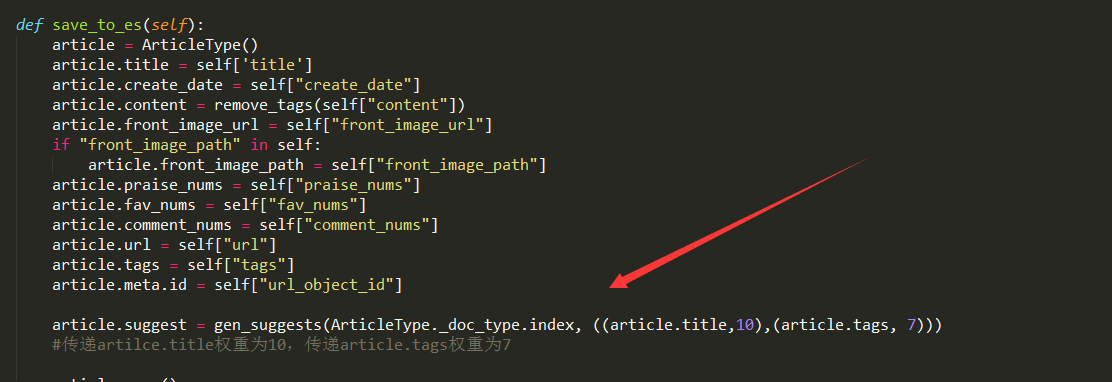
]

}

在item.py里定义一个全局函数：



调用这个全局函数



# Django搭建搜索网站

安装Django:

pip 命令安装方法:

pip install Django

源码安装方法:

下载源码包：<https://www.djangoproject.com/download/>

输入以下命令并安装：

tar xzvf Django-X.Y.tar.gz # 解压下载包

cd Django-X.Y # 进入 Django 目录

python setup.py install # 执行安装命令

django创建第一个项目：

django-admin.py startproject HelloWorld

创建完成后我们可以查看下项目的目录结构：

$ cd HelloWorld/

$ tree

.

|-- HelloWorld

| |-- \_\_init\_\_.py

| |-- settings.py

| |-- urls.py

| `-- wsgi.py

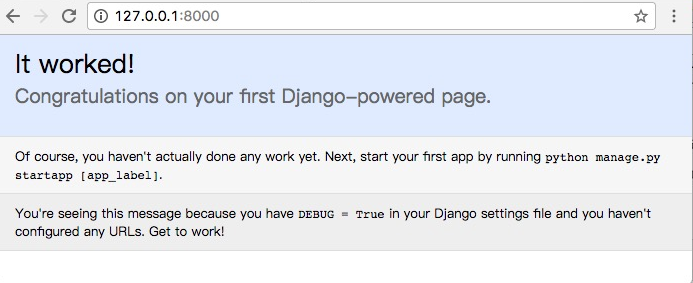
`-- manage.py

接下来我们进入 HelloWorld 目录输入以下命令，启动服务器：

python manage.py runserver 0.0.0.0:8000

0.0.0.0 让其它电脑可连接到开发服务器，8000 为端口号。如果不说明，那么端口号默认为 8000。

在浏览器输入你服务器的ip及端口号，如果正常启动，输出结果如下：



### 创建 APP：

Django规定，如果要使用模型，必须要创建一个app。我们使用以下命令创建一个 TestModel 的 app:

python manage.py startapp search