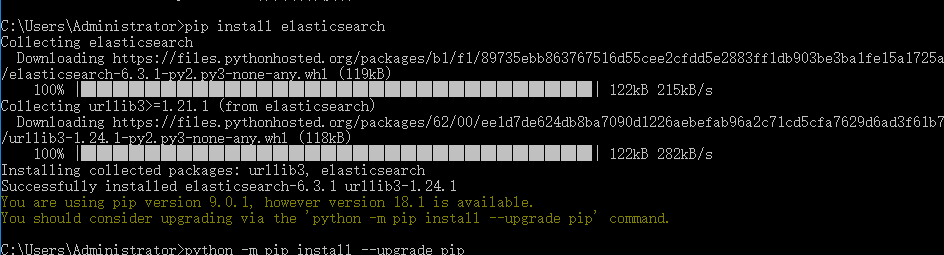
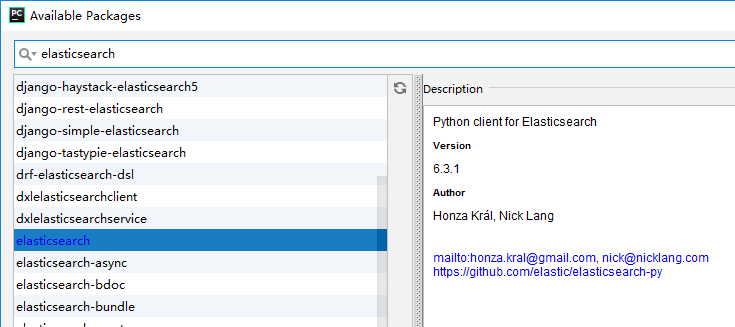
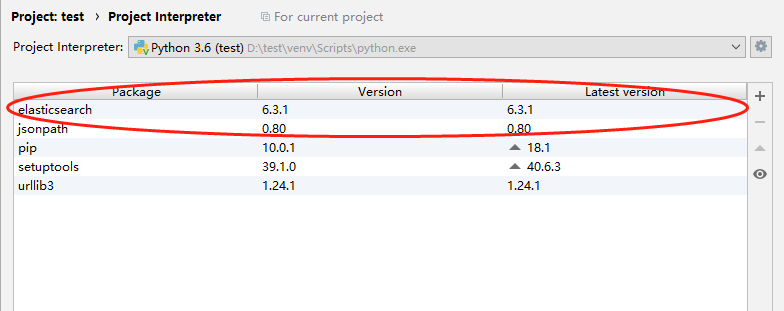
1. cmd下使用命令插入elasticsearch

**pip install elasticsearch**



在项目中引用elasticsearch





1. 创建索引

**from** elasticsearch **import** Elasticsearch  
  
  
es = Elasticsearch([**'127.0.0.1:9200'**])  
es.indices.create(index=**'test-index1'**, ignore=400)

**注意:可以连接多个节点**

**es = Elasticsearch(['192.168.1.1:9200',’192.168.1.2:9200’])**

**注意:ignore可以忽略异常**

**ignore可以忽略异常，其值是需要忽略的异常对应的返回码，常见的有400表示索引已存在，404表示索引没找到。**

然后在dev-tools中查看操作结果:

GET /test-index

结果如下:

{

"test-index" : {

"aliases" : { },

"mappings" : { },

"settings" : {

"index" : {

"creation\_date" : "1546505062936",

"number\_of\_shards" : "5",

"number\_of\_replicas" : "1",

"uuid" : "OytsBP\_qSL2RRD-sgpkY3g",

"version" : {

"created" : "6050499"

},

"provided\_name" : "test-index"

}

}

}

}

表示python可以操作elasticsearch了!

**注意:** <https://elasticsearch-py.readthedocs.io/>

1. 删除索引

es.indices.delete(index=**'test-index'**, ignore=[400, 404])

再次执行:

GET /test-index

结果如下:

{

"error" : {

"root\_cause" : [

{

"type" : "index\_not\_found\_exception",

"reason" : "no such index",

"resource.type" : "index\_or\_alias",

"resource.id" : "test-index",

"index\_uuid" : "\_na\_",

"index" : "test-index"

}

],

"type" : "index\_not\_found\_exception",

"reason" : "no such index",

"resource.type" : "index\_or\_alias",

"resource.id" : "test-index",

"index\_uuid" : "\_na\_",

"index" : "test-index"

},

"status" : 404

}

表示当前索引已经被删除!

1. 插入数据

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
*#其中index表示插入的索引,doc\_type表示数据类型,id表示主键,body表示要插入索引的内容*

*#注意:因为elasticsearch中的数据格式为json,所以python中提供的数据类型为dict*es.index(index=**"my-index"**, doc\_type=**"test-type"**, id=1, body={**"any"**: **"data01"**, **"timestamp"**: datetime.now()})

在dev-tools中操作

get /my-index/test-type/1

结果如下:

{

"\_index" : "my-index",

"\_type" : "test-type",

"\_id" : "1",

"\_version" : 1,

"found" : true,

"\_source" : {

"any" : "data01",

"timestamp" : "2019-01-03T16:54:35.945056"

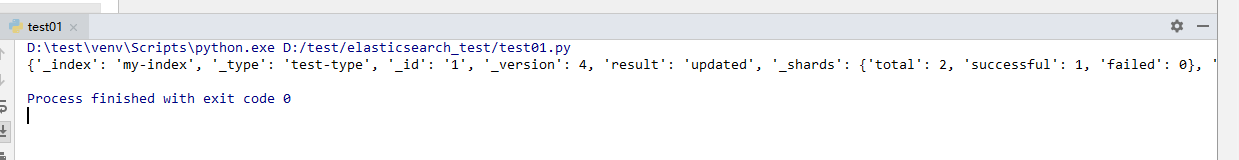
}

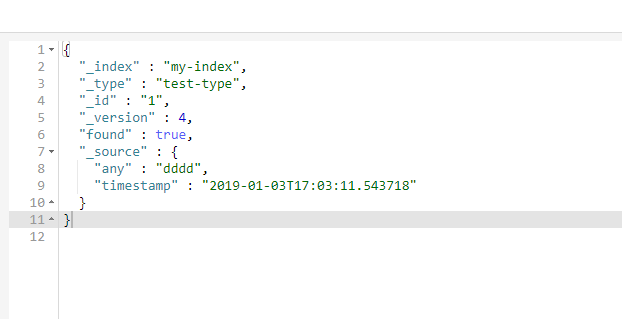
}

**重点:python中的所有操作都会提供返回值,返回值的数据同DevTools的操作结果,在插入数据的时候把代码改造下**

cc = es.index(index=**"my-index"**, doc\_type=**"test-type"**, id=1, body={**"any"**: **"dddd"**, **"timestamp"**: datetime.now()})  
  
print(cc)

**可以看到cc的内容如下:同DevTools中的get /my-index/test-type/1的查询结果,两者的内容一致.**



****

1. 查询数据

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
result = es.get(index=**"my-index"**,doc\_type=**"test-type"**,id=**"1"**)  
print(result)

查询结果同DevTools中的查询结果

----------------------------------------------------------------------------------

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body={  
 **"query"**:{  
 **"match\_all"**: {}  
 }  
}  
  
result2 = es.search(index=**"my-index"**,doc\_type=**"test-type"**,body=body)  
print(result2)

**注意:这里的body中的内容在DevTools中是这样写的**

get /my-index/test-type/\_search

{

"query":{

"match\_all": {}

}

}

**其中的body等同于DevTools中的内容**

**--------------------------------------------------------**

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body={  
 **"query"**: {  
 **"match"**: {  
 **"timestamp"**: **"2019-01-03"** }  
 }  
}  
  
result2 = es.search(index=**"my-index"**,doc\_type=**"test-type"**,body=body)  
print(result2)

**注意:这里的body为按条件查询的方式,match表示匹配方式,等同于DevTools中的**

GET /my-index/test-type/\_search

{

"query": {

"match": {

"timestamp": "2019-01-03"

}

}

}

1. 删除数据

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
result = es.delete(index=**"my-index"**,doc\_type=**"test-type"**,id=1)  
print(result)

删除的时候需要指定index(索引),doc\_type(文档类型),id(序号),等同于

DELETE /my-index/test-type/1

**注意:如果多次执行删除操作,会报错.需要通过捕获异常处理,改进过的代码如下:**

**报错的实质就是当前的id不存在**

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
**try**:  
 result = es.delete(index=**"my-index"**,doc\_type=**"test-type"**,id=1)  
**except**:  
 print(**"你看,删多了吧"**)

1. 按条件删除

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body = {  
 **"query"**:{  
 **"term"**: {  
 **"school.name"**: **"华山派"** }  
 }  
}  
  
result = es.delete\_by\_query(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

**注意:使用的是笑傲江湖的数据,删除数据以后再使用条件查询,发现数据已经不见了**

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"range"**: {  
 **"age"**: {  
 **"lte"**: 20*#删除小于等于20岁的人*  
 }  
 }  
 }  
}  
  
result = es.delete\_by\_query(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. 按条件更新(update\_by\_query)

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body = {  
 **"script"**: {  
 **"source"**: **"ctx.\_source['age']=21;ctx.\_source['name']='renyy'"** },  
 **"query"**: {  
 **"range"**: {  
 **"age"**: {  
 **"lte"**: 20  
 }  
 }  
 }  
}  
  
result = es.update\_by\_query(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

**重点:更新查询条件为年龄小于等于20岁的人,其中script中为脚本语法(有坑,详细内容还未查询),修改多项内容,内容与内容之间用”;”号间隔**

POST /person/\_update\_by\_query

{

"script":{

"source": "ctx.\_source['age']=21;ctx.\_source['name']='renyy'"

},

"query": {

"range": {

"age": {

"lte": 20

}

}

}

}

1. 批量新增

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body = [  
 {**"index"**:{}},  
 {**"name"**:**"张无忌"**,**"age"**:22,**"school"**:{**"name"**:**"武当派"**}},  
 {**"index"**:{}},  
 {**"name"**:**"张翠山"**,**"age"**:46,**"school"**:{**"name"**:**"武当派"**}}  
]  
  
result = es.bulk(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

对于从数据库中取出的数据list类型的操作

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
list = [  
 {**"name"**:**"宋远桥"**,**"age"**:45,**"school"**:{**"name"**:**"武当派"**}},  
 {**"name"**:**"阳顶天"**,**"age"**:55,**"school"**:{**"name"**:**"魔教"**}},  
]  
  
  
doc = []  
  
**for** person **in** list:  
 id = {**"index"**:{}}  
 doc.append(id)  
  
 body = {  
 **"name"**:person.get(**"name"**),  
 **"age"**:person.get(**"age"**),  
 **"school"**:{  
 **"name"**:person.get(**"school"**).get(**"name"**)  
 }  
 }  
 doc.append(body)  
  
result = es.bulk(index=**"person"**,doc\_type=**"daxia"**,body=doc)  
  
print(result)

1. 查询所有数据

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**)  
  
print(result)

或者指定body也可以

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body = {  
 **"query"**:{  
 **"match\_all"**:{}  
 }  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. term和terms

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body = {  
 **"query"**:{  
 **"term"**: {  
 **"name"**: {  
 **"value"**: **"张三丰"** }  
 }  
 }  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

---------------------------------

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body = {  
 **"query"**:{  
 **"terms"**: {  
 **"name"**: [  
 **"张三丰"**,  
 **"张无忌"** ]  
 }  
 }  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. match和multi\_match

match是匹配到就可以,multi\_match是查询一个关键字,在不同的属性里查找到

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body = {  
 **"query"**:{  
 **"match"**: {  
 **"name"**: **"张三丰"** }  
 }  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

------------------------------------

**from** elasticsearch **import** Elasticsearch  
**from** datetime **import** datetime  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"multi\_match"**: {  
 **"query"**: **"张三丰"**,#查询的内容为张三丰  
 **"fields"**: [**"name"**,**"school.name"**]#查询的位置为name和school.name属性中  
 }  
 }  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. ids根据多个id查询

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"ids"**: {  
 **"values"**: [1,2]  
 }  
 }  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. 复合查询bool

must:必须都满足条件

should:其中一个满足

must\_not:必须都不满足

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"bool"**: {  
 **"must"**: [  
 {**"term"**: {  
 **"name"**: {  
 **"value"**: **"张三丰"** }  
 }},  
 {**"term"**: {  
 **"age"**: {  
 **"value"**: 98  
 }  
 }}  
 ]  
 }  
 }  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

------------------------------------------------------------

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"bool"**: {  
 **"should"**: [  
 {**"term"**: {  
 **"name"**: {  
 **"value"**: **"张三丰"** }  
 }},  
 {**"term"**: {  
 **"age"**: {  
 **"value"**: 98  
 }  
 }}  
 ]  
 }  
 }  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

--------------------------------------------------------------

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"bool"**: {  
 **"must\_not"**: [  
 {**"term"**: {  
 **"name"**: {  
 **"value"**: **"张三丰"** }  
 }},  
 {**"term"**: {  
 **"age"**: {  
 **"value"**: 98  
 }  
 }}  
 ]  
 }  
 }  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. 分页查询

query是查询的条件部分

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"match\_all"**: {}#这里是查询的条件部分  
 },  
 **"from"**: 0,#从第几条开始查  
 **"size"**: 10#一次查多少条数据  
}  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. 范围查询加分页

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"range"**: {#设置区间条件  
 **"age"**: {  
 **"gte"**: 10,  
 **"lte"**: 100  
 }  
 }  
 },  
 **"from"**: 0,  
 **"size"**: 10  
}  
  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. 前缀查询

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"prefix"**: {  
 **"name"**: {  
 **"value"**: **"张"# 以”张”开头的内容** }  
 }  
 }  
}  
  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. 通配符查询

\*无忌:以无忌结尾的都要

张\*:以张开头的都要

张\*忌:张无忌,张有忌,张1234忌都是我的菜

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"wildcard"**: {#使用通配符  
 **"name"**: {  
 **"value"**: **"\*无忌"** }  
 }  
 }  
}  
  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. 排序

Sort是数组,也可以同时对多个字段进行排序,asc升序,desc降序

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"match\_all"**: {}  
 },  
 **"sort"**: [  
 {  
 **"age"**: {  
 **"order"**: **"desc"** }  
 }  
 ]  
}  
  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. 过滤查询

查询\_id属性:

result = es.search(index=**"person"**,doc\_type=**"daxia"**,filter\_path=[**"hits.hits.\_id"**])

查询\_source属性:  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,filter\_path=[**"hits.hits.\_source"**])

默认全查:  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,filter\_path=[**"hits.hits.\_\*"**])

查询当前数据总数

result = es.count(index=**"person"**,doc\_type=**"daxia"**)

1. 聚合函数

求最小值:min

求最大值:max

求平均值:avg

求和:sum

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"match\_all"**: {}  
 },  
 **"aggs"**: {#这里是聚合函数的意思  
 **"最小年龄"**: {#自定义的字段名称  
 **"min"**: {#求最小值  
 **"field"**: **"age"#要求那个字段的最小值** }  
 }  
 }  
}  
  
  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

--------------------------------------------

同时查询多条聚合函数

**from** elasticsearch **import** Elasticsearch  
  
es = Elasticsearch()  
  
body = {  
 **"query"**: {  
 **"match\_all"**: {}  
 },  
 **"aggs"**: {  
 **"最小年龄"**: {  
 **"sum"**: {  
 **"field"**: **"age"** }  
 },  
 **"平均年龄"**:{  
 **"avg"**: {  
 **"field"**: **"age"** }  
 }  
 }  
}  
  
  
  
result = es.search(index=**"person"**,doc\_type=**"daxia"**,body=body)  
  
print(result)

1. 以上内容是课件