



链滴

Rest 风格说明 (ES 系列——关于索引的操作)

作者: [yscxy](#)

原文链接: <https://ld246.com/article/1605012749976>

来源网站: 链滴

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



Rest风格说明

它是一种架构风格，而不是标准，只是提供了一组设计原则和约束条件。主要用于客户端和服务端交互的软件，基于这个风格设计的软件可以更简洁，更有层次，更易于实现缓存机制等等。

基本命令说明

method	url地址	描述
PUT	localhost:9200/索引名称/类型名称/文档id	创建文档 (指定文档id)
POST	localhost:9200/索引名称/类型名称	创建文档 (随机文档id)
POST	localhost:9200/索引名称/类型名称/文档id/_update	修改文档
DELETE	localhost:9200/索引名称/类型名称/文档id	删除文档
GET	localhost:9200/索引名称/类型名称/文档id	查询文档通过文档id
POST	localhost:9200/索引名称/类型名称/_search	查询所有数据

安装elasticsearch-head

[elasticsearch-head](#)，可以直接下压缩包，也可以通过 git clone。

输入命令，等待下载完成：

**

```
git clone git://github.com/mobz/elasticsearch-head.git
```

安装 grunt-cli

```
npm install -g grunt-cli
```

安装 grunt

elasticsearch-head 下载完成后, 进入 elasticsearch-head 文件夹, 执行命令:

```
npm install grunt --save
```

安装依赖的 npm 包

```
npm install
```

修改启动文件

所有依赖包安装成功后, 修改 elasticsearch-head 目录下的 Gruntfile.js 文件, 在 options 属性内加 hostname, 设置为 0.0.0.0。

```
connect: {
  server: {
    options: {
      hostname: '0.0.0.0',
      port: 9100,
      base: '.',
      keepalive: true
    }
  }
}
```

修改 Elasticsearch 配置文件 config/elasticsearch.yml

在配置文件最后增加两个配置项, 这样 elasticsearch-head 插件才可以访问 Elasticsearch。

```
http.cors.enabled: true
http.cors.allow-origin: "*"

```

启动 elasticsearch-head

在 elasticsearch-head 目录下, 执行命令:

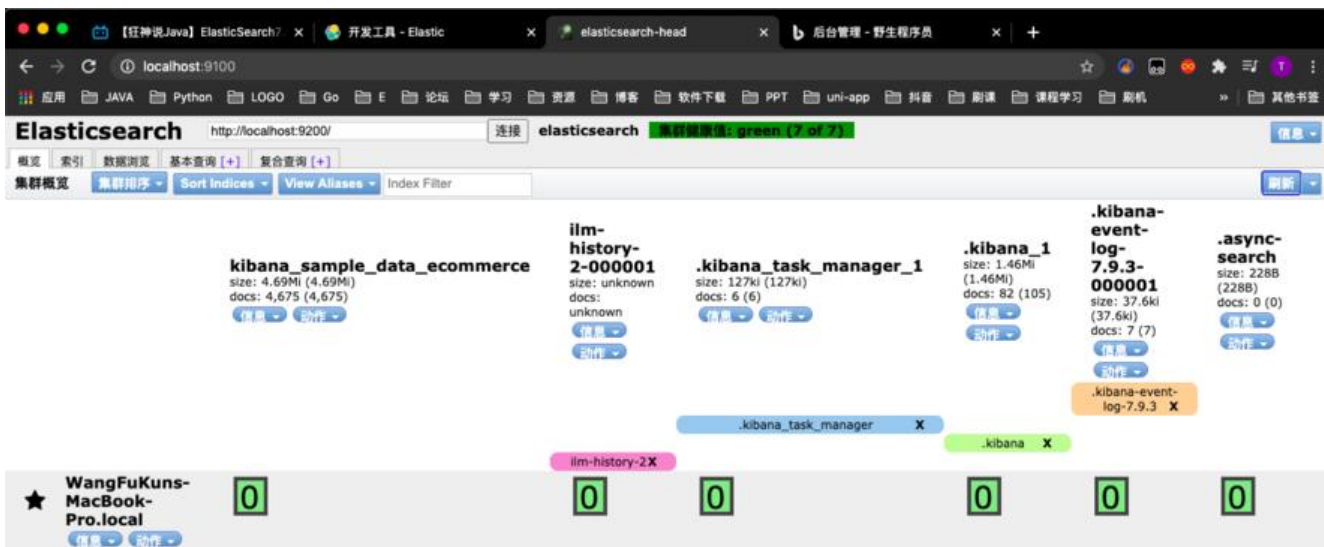
```
grunt server
```

输出如下内容表示启动成功:

```
Running "connect:server" (connect) task
Waiting forever...
Started connect web server on http://localhost:9100

```

访问 <http://localhost:9100> 地址, 就可以看到当前 Elasticsearch 集群信息。



基础测试

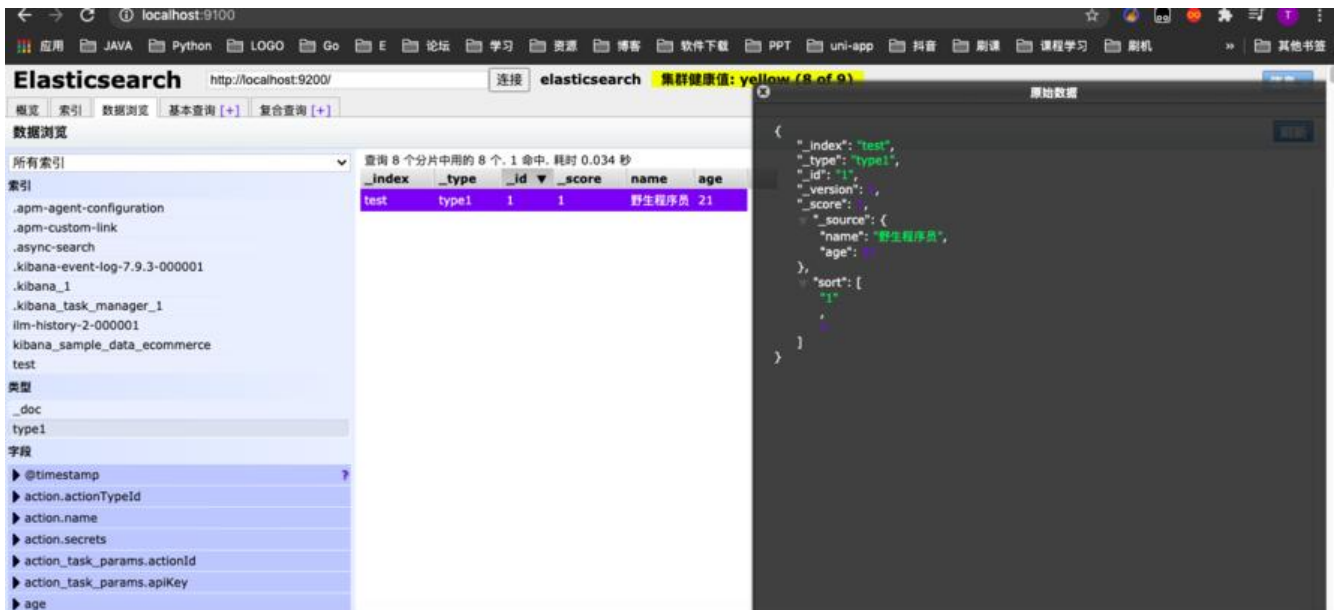
1. 创建一个索引

PUT/索引名/类型名/文档id

{ 请求体 }



在elasticsearch-head里面我们就可以看索引的状态了!



这样我们就完成了自动添加索引，不仅仅是命令可以添加索引，当然http请求也可以，当然我们也可将他当作数据库进行学习！

2. name字段的数据类型

- 字符串类型
[text](#)、[keyword](#)
- 数值类型
[long](#)、[integer](#)、[short](#)、[byte](#)、[double](#)、[float](#)、[half float](#)、[scaled float](#)
- 日期类型
[date](#)
- 布尔值类型
[boolean](#)
- 二进制类型
[binary](#)
- 等等.....

指定字段规则

```
PUT /test2
{
  "mappings": {
    "properties": {
      "name": {
        "type": "text"
      },
      "age": {
        "type": "integer"
      },
      "birthday": {
        "type": "date"
      }
    }
  }
}
```

200 - OK 327 ms

```
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "test2"
}
```

获得这个规则，可以通过get请求获取索引的信息

GET /test2

200 - OK 26 ms

```
1- {
2-   "test2" : {
3-     "aliases" : { },
4-     "mappings" : {
5-       "properties" : {
6-         "age" : {
7-           "type" : "integer"
8-         },
9-         "birthday" : {
10-          "type" : "date"
11-        },
12-         "name" : {
13-          "type" : "text"
14-        }
15-       }
16-     },
17-     "settings" : {
18-       "index" : {
19-         "creation_date" : "1605010535277",
20-         "number_of_shards" : "1",
21-         "number_of_replicas" : "1",
22-         "uuid" : "DL4G900STDemv9nm8FJ83A",
```

查看默认信息,从重新创建一个索引

PUT /test3/_doc/1

```
{
  "name": "野生程序员",
  "age": 21,
  "birt": "1997-10-09"
}
```

```

1- {
2  "_index" : "test3",
3  "_type" : "_doc",
4  "_id" : "1",
5  "_version" : 1,
6  "result" : "created",
7-  "_shards" : {
8    "total" : 2,
9    "successful" : 1,
10   "failed" : 0
11- },
12  "_seq_no" : 0,
13  "_primary_term" : 1
14- }
15

```

The screenshot shows the ElasticSearch Kibana interface. The top navigation bar includes 'Elasticsearch', 'http://localhost:9200/', '连接', 'elasticsearch', and '集群健康值: yellow (10)'. The main content area is divided into several sections:

- 数据浏览 (Data Explorer):** Shows a table with columns: `_index`, `_type`, `_id`, `_score`, `name`, `age`, `birt`. A single row is displayed: `test3`, `_doc`, `1`, `1`, `野生程序员`, `21`, `1997-10-09`.
- 索引 (Index):** Lists various indices, with `test3` selected.
- 类型 (Type):** Shows the type `_doc`.
- 字段 (Fields):** Lists fields including `@timestamp`, `action.actionTypeId`, `action.name`, and `action.secrets`.
- 历史记录 (History):** Shows a list of operations:


```

1 PUT /test3/_doc/1
2 {
3   "name": "野生程序员",
4   "age": 21,
5   "birt": "1997-10-09"
6 }
7 GET test3

```
- 原始数据 (Raw Data):** Displays the JSON document:


```

{
  "_index": "test3",
  "_type": "_doc",
  "_id": "1",
  "_version": 1,
  "_score": 1,
  "_source": {
    "name": "野生程序员",
    "age": 21,
    "birt": "1997-10-09"
  }
}

```
- 映射配置 (Mapping):** Shows the mapping configuration for the `test3` index:


```

1- {
2-   "test3": {
3-     "aliases": { },
4-     "mappings": {
5-       "properties": {
6-         "age": {
7-           "type": "long"
8-         },
9-         "birt": {
10-          "type": "date"
11-        },
12-         "name": {
13-          "type": "text",
14-          "fields": {
15-            "keyword": {
16-              "type": "keyword",
17-              "ignore_above": 256
18-            }
19-          }
20-        }
21-      }
22-    }
23-  }

```

也就是说，自己的文档没有设置数据类型，框架会自动识别给我们设置数据类型

获取索引情况 (拓展)

GET `_cat/health` 获取健康情况

GET `_cat/indices` 获取版本信息等等

历史记录 设置 帮助		200 - OK 28 ms										
1	PUT /test3/_doc/1	1	health	status	index	uuid	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
2	{	2	green	open	.kibana-event-log-7.9.3-000001	4h10li8JTFSZQ4YziGzBYQ	1	0	8	0	43kb	43kb
3	"name"	3	yellow	open	test2	DL4G9005TDemv9nm8FJ83A	1	1	0	0	208b	208b
4	:"野生程序员"	4	yellow	open	test3	3sg9uyOMTgqmE7Y0-Iu8Hw	1	1	1	0	4.3kb	4.3kb
5	"age":21,	5	yellow	open	test	V4E7FGuqTMiDAV7Bhu-yog	1	1	1	0	4.1kb	4.1kb
6	"birt":"1997-10-09"	6	green	open	.apm-custom-link	b00sYZ-HQagMJjrkabx9ZA	1	0	0	0	208b	208b
7	}	7	green	open	.kibana_task_manager_1	-ilyadiUTaaFnEyo-23zng	1	0	6	50	126.1kb	126.1kb
8	GET _cat/test3	8	green	open	.kibana_sample_data_ecommerce	8fKa3mkCRCi3tpop0ox0Uw	1	0	4675	0	4.6mb	4.6mb
9	GET _cat/headers	9	green	open	.apm-agent-configuration	dRB00V6bTGWxwG65fxcbgA	1	0	0	0	208b	208b
10	GET _cat/indices	10	green	open	.async-search	tWgXlx2MQmuk46Q6p43e5A	1	0	0	0	3.4kb	3.4kb
11		11	green	open	.kibana_1	El876r0dQvedY6Llbrzktg	1	0	149	4	2.9mb	2.9mb
12		12										

修改索引

修改用PUT既可以

```
PUT /test3/_doc/1
{
  "name":"野生程序员修改版本",
  "age":21,
  "birt":"1997-10-09"
}
GET /test3/_doc/1
```

记录 设置 帮助		200 - OK 55 ms	
1	PUT /test3/_doc/1	1	{
2	{	2	"_index" : "test3",
3	"name":"野生程序员修改版本",	3	"_type" : "_doc",
4	"age":21,	4	"_id" : "1",
5	"birt":"1997-10-09"	5	"_version" : 2,
6	}	6	"_seq_no" : 1,
7	GET /test3/_doc/1	7	"_primary_term" : 1,
8		8	"found" : true,
9		9	"_source" : {
10		10	"name" : "野生程序员修改版本",
11		11	"age" : 21,
12		12	"birt" : "1997-10-09"
13		13	}
14		14	}
15		15	

当然这个已经已经不宜用了，我们最新的方法是用POST请求，下面是示例

```
POST /test3/_doc/1/_update
{
  "doc":{"
    "name":"亡命之徒"
  }}
GET /test3/_doc/1
```


历史记录 设置 帮助

200 - OK 53 ms

```
1 PUT /test3/_doc/1
2- {
3  "name": "野生程序员修改版本",
4  "age": 21,
5  "birt": "1997-10-09"
6- }
7 GET /test3/_doc/1
8
9 POST /test3/_doc/1/_update
10- {
11-   "doc": {
12-     "name": "亡命之徒"
13-   }
14- }
15 GET /test3/_doc/1
```

```
1 #! Deprecation: [types removal] Specifying types in document update requests is deprecated, use
the endpoint /{index}/_update/{id} instead.
2- {
3  "_index": "test3",
4  "_type": "_doc",
5  "_id": "1",
6  "_version": 3,
7  "result": "updated",
8-  "_shards": {
9    "total": 2,
10   "successful": 1,
11   "failed": 0
12- },
13  "_seq_no": 2,
14  "_primary_term": 1
15- }
16
```

历史记录 设置 帮助

200 - OK 22 ms

```
1 PUT /test3/_doc/1
2- {
3  "name": "野生程序员修改版本",
4  "age": 21,
5  "birt": "1997-10-09"
6- }
7 GET /test3/_doc/1
8
9 POST /test3/_doc/1/_update
10- {
11-   "doc": {
12-     "name": "亡命之徒"
13-   }
14- }
15 GET /test3/_doc/1
```

```
1- {
2  "_index": "test3",
3  "_type": "_doc",
4  "_id": "1",
5  "_version": 3,
6  "_seq_no": 2,
7  "_primary_term": 1,
8  "found": true,
9-  "_source": {
10-    "name": "亡命之徒",
11-    "age": 21,
12-    "birt": "1997-10-09"
13-  }
14- }
15
```

删除索引

DELETE test #这是直接删除一个库

历史记录 设置 帮助

200 - OK 100 ms

```
1 DELETE test
2
```

```
1- {
2  "acknowledged": true
3- }
4
```