1,Restful API

API基本风格 http://<ip>:<port>/<索引>/<类型>/<文档id>

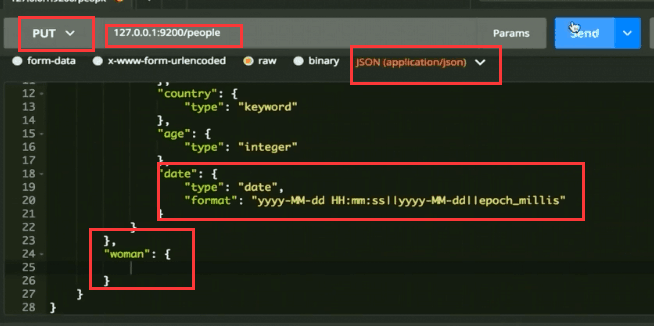
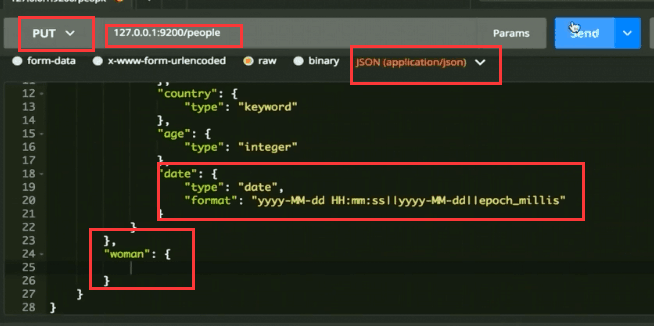
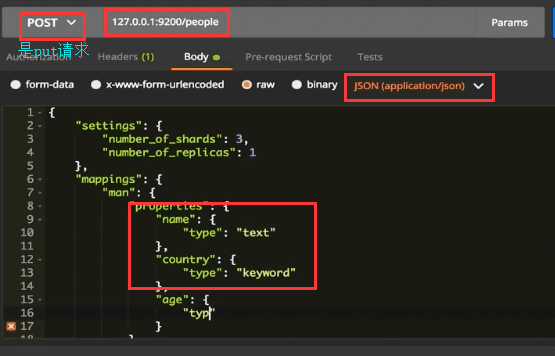
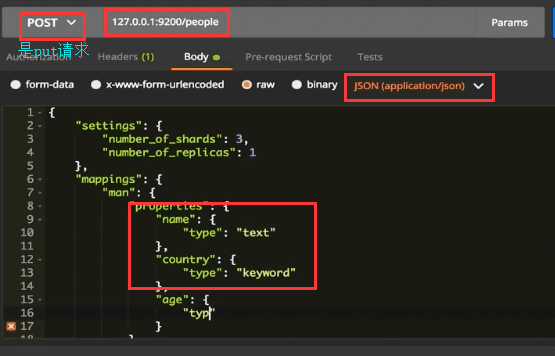
常用HTTP动词 GET/PUT/POST/DELETE

2,创建索引

可类比为数据库

创建方式可区分非结构化创建和结构化创建

 利用postmam 创建：



3，数据插入

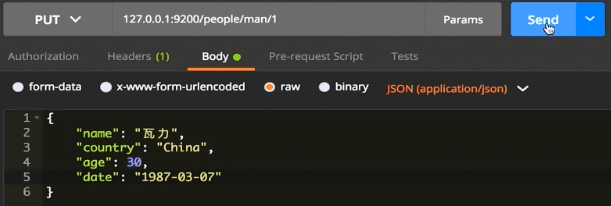
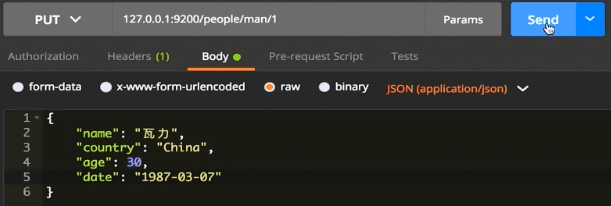
put 127.0.0.1:9200/people/man/1

people 是索引

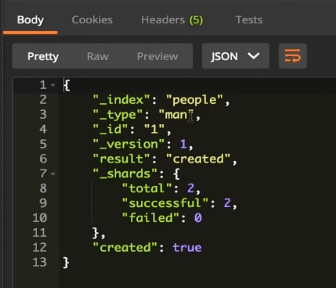
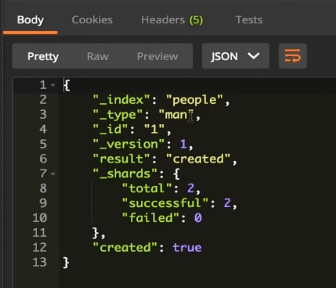
man 是类型

1 是 文档id

post 127.0.0.1:9200/people/man/   这个可以自动生成一个字符串id



响应值：



4，数据修改

post 127.0.0.1:9200/people/man/1/\_update

{

    "doc":{

        "name":"谁是瓦力"

    }

}

脚本修改：

{

    “script”:{

            "lang":"painless"  //指定脚本语言，es支持多种脚本语言，如python,painless是es内置的脚本语言

            "inline":"ctx.\_source.age+=10"//指定脚本内容，ctx代表es上下文,\_source代表es当前文档

    }

}

{

    “script”:{

            "lang":"painless" ,

            "inline":"ctx.\_source.age=params.age",//把参数配置在外部

            "params":{

                "age":10

            }

    }

}

5，数据删除

1，删除文档

DELETE  127.0.0.1:9200/people/man/1/  删除索引为1的文档

2，删除索引

DELETE  127.0.0.1:9200/people   删除people索引

6，数据查询

1，简单查询

GET 127.0.0.1:9200/book/novel/1

2，条件查询

POST 127.0.0.1:9200/book/\_search

//查询全部

request body :

            {

    "query":{

        "match\_all":{}

    }，

    "from":1

    "size":1

}

响应：hits 响应的数据 ，默认响应10条数据 ,可以使用from 和 size 指定

//关键字查询

{

    "query":{

        "match":{

            "title":"ElasticSearch"

        }

    },

    "sort":[ //指定排序规则

        {"publish\_data":{"order":"desc"}} //指定排序字段

    ]

}

响应：默认是使用\_score 这个字段排序，可以通过sort 字段排序字段，排序规则

3，聚合查询

{

    "aggs":{//聚合关键字

        "group\_by\_word\_count":{//给聚合条件起名字（自定义的，可以随便起）

            "terms":{//关键词

                "field":"word\_count" //指定聚合字段

            }

        }

    }

}

响应结果，文档按照字数统计分组统计

多条聚合查询

//多条聚合

{

    "aggs":{

        "group\_by\_word\_count":{

            "terms":{

                "field":"word\_count"

            }

        },

        "group\_by\_publish\_date":{

            "terms":{

                "field":"publish\_data"

            }

        }

    }

}

响应：有两组聚合信息响应

//聚合计算

{

    "aggs":{

        "grades\_word\_count":{

            "stats":{

                "field":"word\_count"

            }

        }

    }

}

响应结果是各种计算

计算可以直接指定计算对象 如将：stats 换为min 就直接返回最小值



**高级查询**

 子条件查询 ：特定字段查询所指特定值

        1，Query context

                在查询过程中，除了了判断文档是否满足查询条件外，ES还会计算一个\_score来标识匹配的程度，旨在判断目标文档和查询条件匹配的有多好

            全文本查询：  针对文本类型数据

           //模糊匹配

            {

                "query":{

                        "match":{

                        "author":"瓦力"

                        }

                }

            }

           //短语匹配 match\_phrase

        //多个字段模糊匹配

        {

                "query":{

                        "multi\_match":{

                                "query":"瓦力"

                                "fields":["author","title"]//要求作者和标题中均包含 瓦力

                          }

                    }

          }

            //语法查询  经常用于kibanna中，做数据搜索用，支持通配符，范围查询，正则表达式

            {

                "query":{

                    query\_string":{

                        "query":"瓦力"

                    }

              }

            {

                "query":{

                        query\_string":{

                                "query":"(瓦力)OR Python"//查询包含 瓦力和python的

                        }

              }

                {

                        "query":{

                                query\_string":{

                                        "query":" 瓦力 OR Python",//查询同时包含 瓦力和python的

                                        "fields":["title","author"]

                                }

                }

        字段级别查询：针对结构化数据，如数字，日期等

        {

                "query":{

                    "term":{

                        "author":"瓦力"

                        }

                   }

            }

    //范围查询

{

    "query":{

        "range":{

            "word\_count":{

                "gte":1000,

                "lte":2000//大于等于1000小于等于2000

            }

        }

    }

}

        2，Filter context

在查询过程中，只判断该文档是否满足条件，只有Yes或No

{

    "query":{

        "bool":{

            "filter":{

                "term":{

                    "word\_count":1000

                }

            }

        }

    }

}

复合条件查询： 以一定的逻辑组合子条件查询

常用查询：固定分数查询，布尔查询

//固定分数查询

{

    "query":{

        "constant\_score":{//固定的分数

            "filter":{

                "match":{

                    "title":"Elasticsearch"

                }

            },

            "boost":2//关键词指定分数

        }

    }

}

//布尔查询

{

    "query":{

        "bool":{

            "should":[// or

                {

                    "match":{

                        "author":"瓦力"

                    }

                },

                {

                    "match":{

                        "title":"Elasticsearch"

                    }

                }

            ]

        }

    }

}

{

    "query":{

        "bool":{

            "must":[ //and

                {

                    "match":{

                        "author":"瓦力"

                    }

                },

                {

                    "match":{

                        "title":"Elasticsearch"

                    }

                }

            ]

        }

    }

}

{

    "query":{

        "bool":{

            "must":[

                {

                    "match":{

                        "author":"瓦力"

                    }

                },

                {

                    "match":{

                        "title":"Elasticsearch"

                    }

                }

            ],

            "filter":[

                {

                    "term":{

                        "word\_count":1000

                    }

                }

            ]

        }

    }

}

must\_not //非