创建zookeeper

创建KAFKA

run -d --name kafka --publish 9092:9092 --link zookeeper –

-env KAFKA\_ZOOKEEPER\_CONNECT=zookeeper:2181

--env KAFKA\_ADVERTISED\_HOST\_NAME=192.168.1.120 --env KAFKA\_ADVERTISED\_PORT=9092 -t wurstmeister/kafka:latest

docker run -d --name esnode1 -p 8200:8200 -t docker.elastic.co/elasticsearch/elasticsearch

:6.2.4

Elasticsearch 创建索引unassigned shards(未分配碎片)

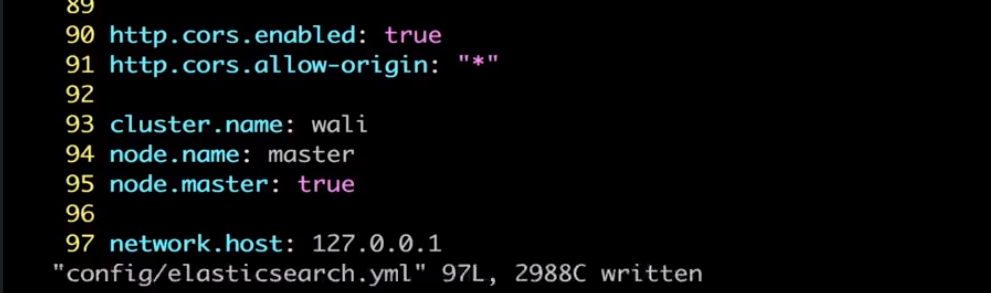
1. INDEX\_CREATED：由于创建索引的API导致未分配。  
   2）CLUSTER\_RECOVERED ：由于完全集群恢复导致未分配。  
   3）INDEX\_REOPENED ：由于打开open或关闭close一个索引导致未分配。  
   4）DANGLING\_INDEX\_IMPORTED ：由于导入dangling索引的结果导致未分配。  
   5）NEW\_INDEX\_RESTORED ：由于恢复到新索引导致未分配。  
   6）EXISTING\_INDEX\_RESTORED ：由于恢复到已关闭的索引导致未分配。  
   7）REPLICA\_ADDED：由于显式添加副本分片导致未分配。  
   8）ALLOCATION\_FAILED ：由于分片分配失败导致未分配。  
   9）NODE\_LEFT ：由于承载该分片的节点离开集群导致未分配。  
   10）REINITIALIZED ：由于当分片从开始移动到初始化时导致未分配（例如，使用影子shadow副本分片）。  
   11）REROUTE\_CANCELLED ：作为显式取消重新路由命令的结果取消分配。  
   12）REALLOCATED\_REPLICA ：确定更好的副本位置被标定使用，导致现有的副本分配被取消，出现未分配。

Docker 查看启动日志

docker logs -f -t esnode1

Elacticsearch 集群配置

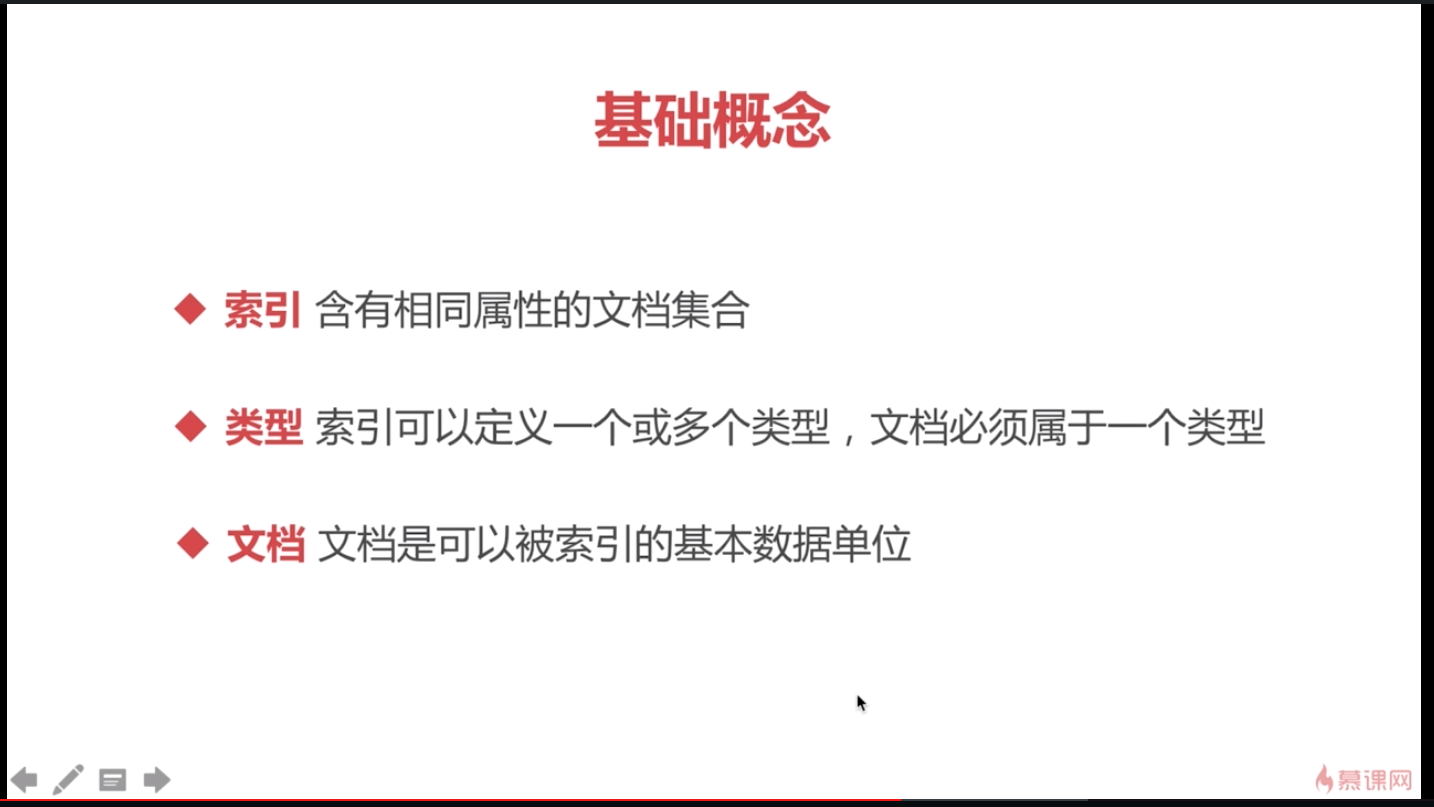
Master:



Slaver 配置



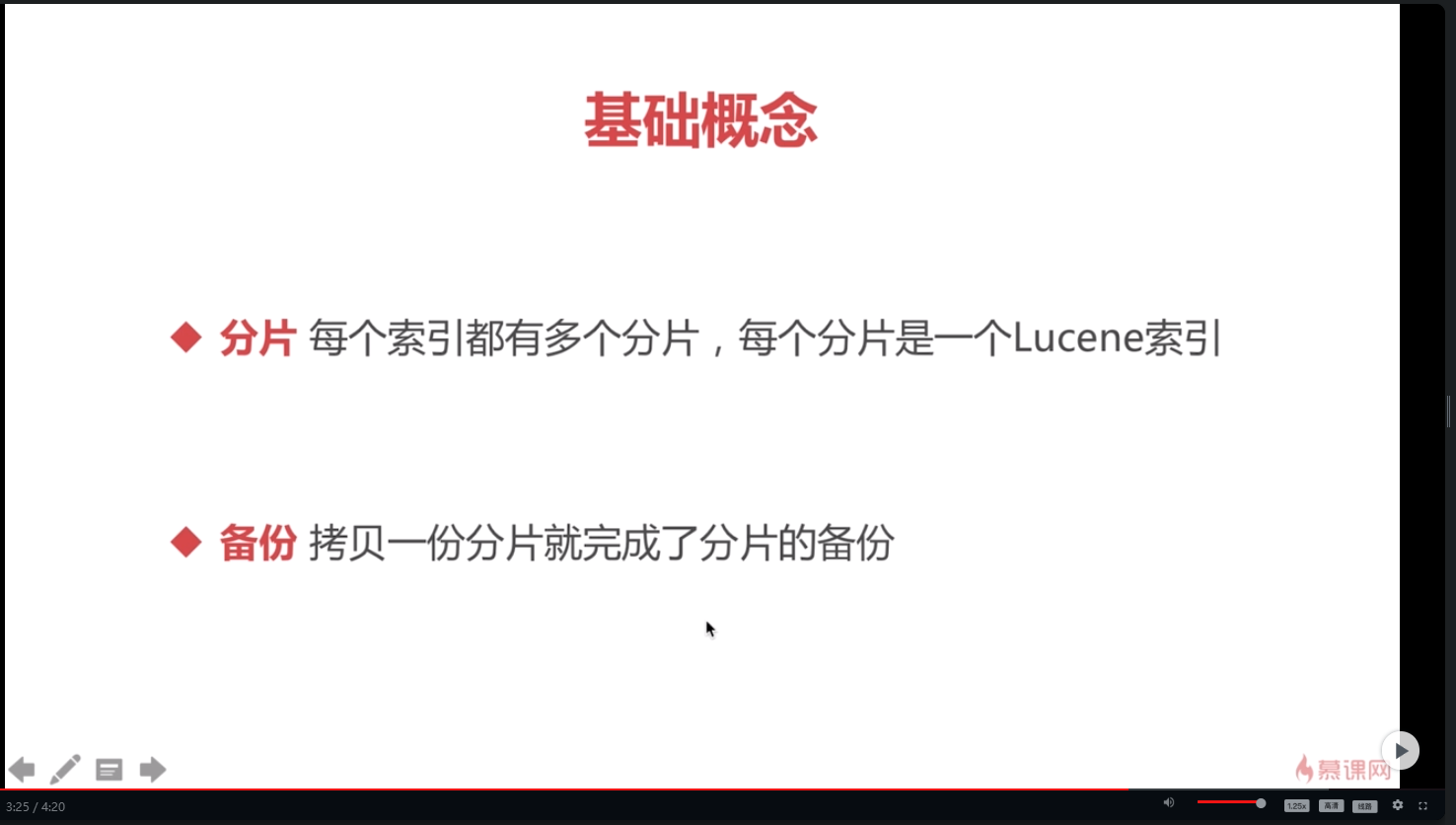
Elasticseach 基本概念

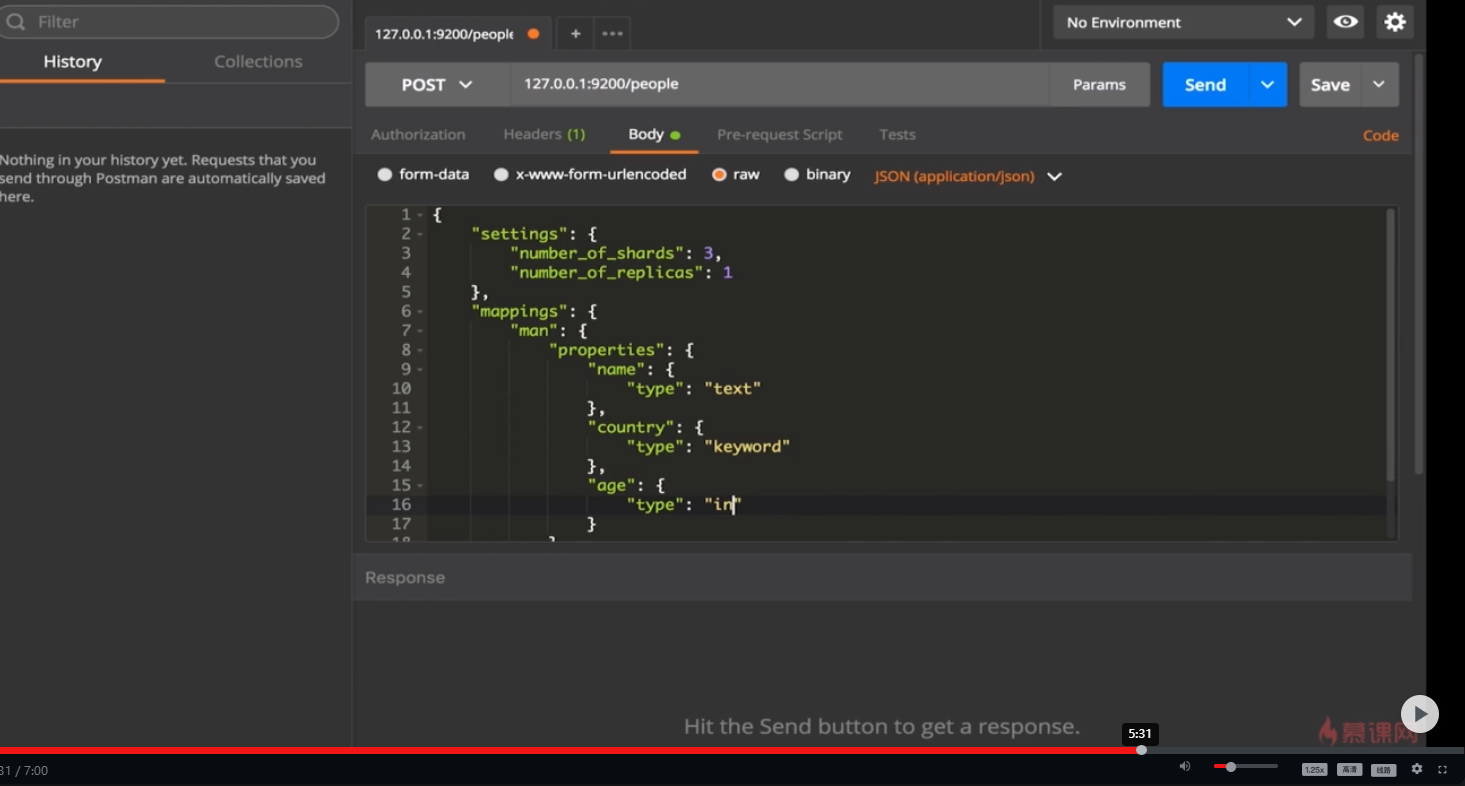


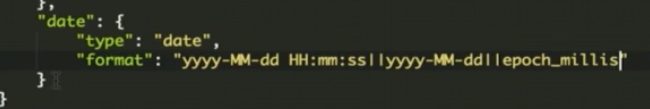
索引 相当于数据库里面的 database

类型 相当于数据库里面的 table

文档 相当于数据库里面的 一条记录





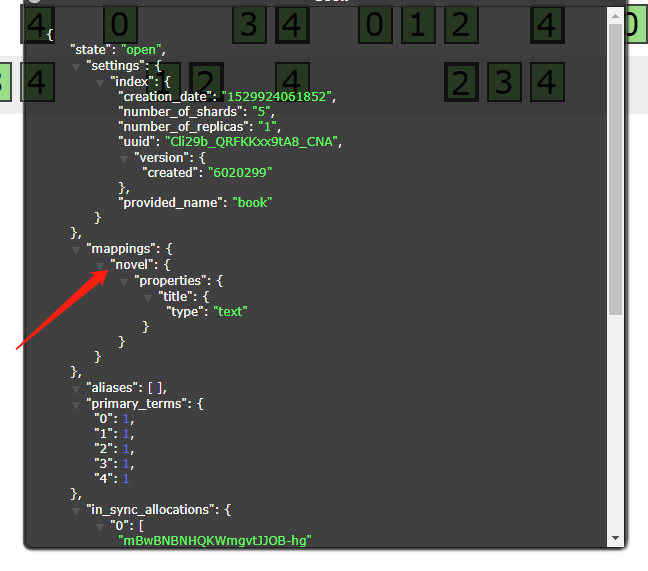


时间戳定义方式

创建people索引 ,有点像java创建类 number\_of\_shards 分片数 number\_of\_replicas 备份数量。

细框是粗框的索引分片备份

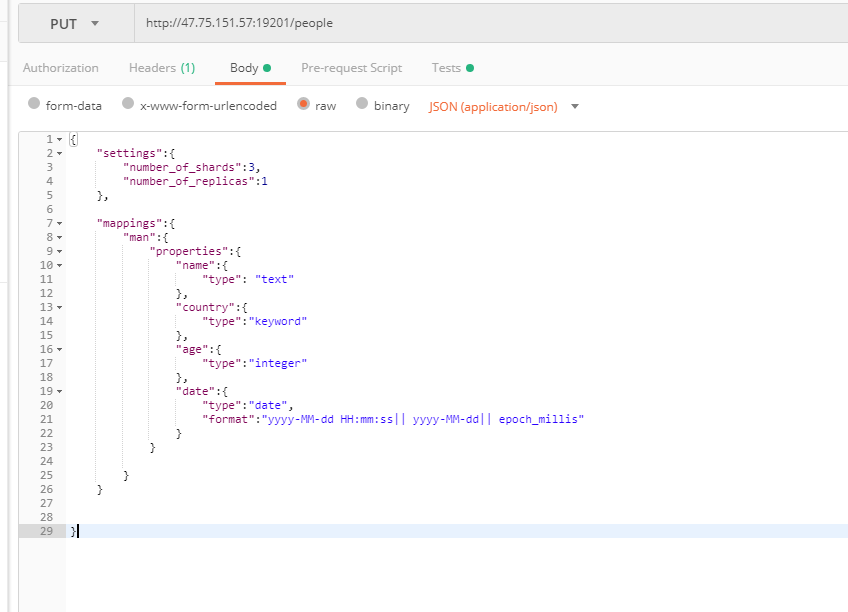
二：索引是结构化还是非结构化



Mappings 里面提供结构映射，mappings为空就是非结构化

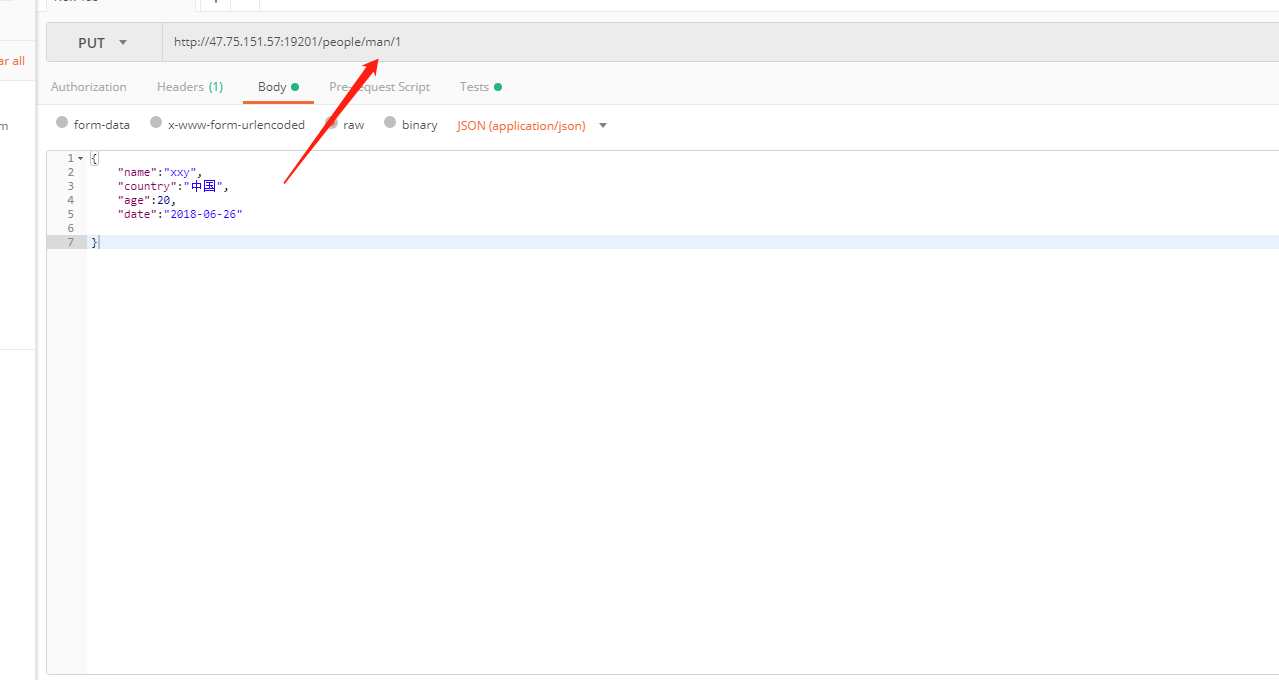
三、索引创建：

1.事例 创建一个people索引

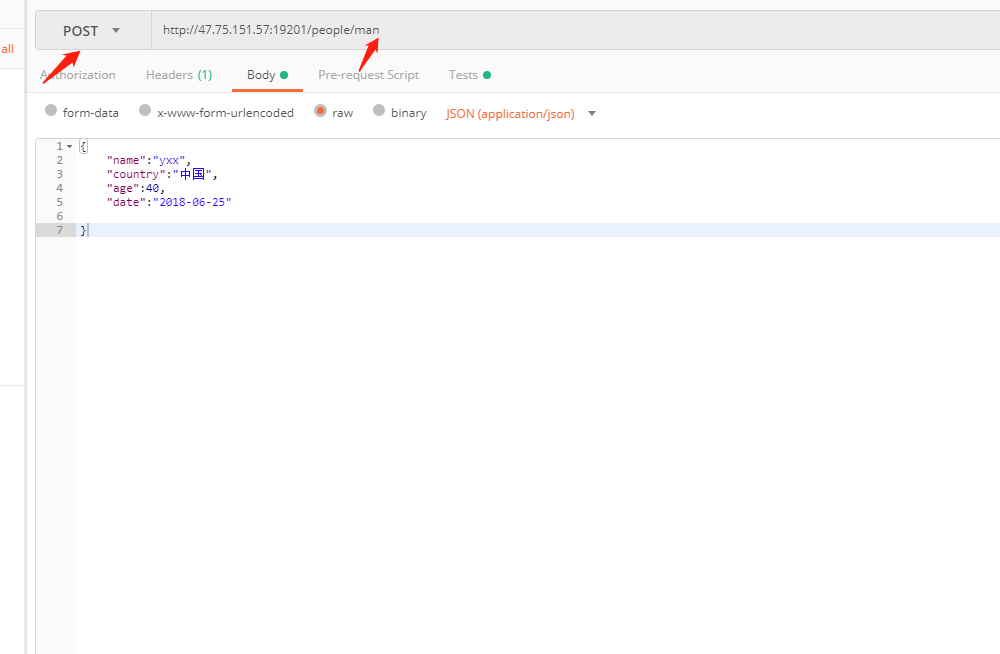


2 数据插入

1. 指定文档的ID插入

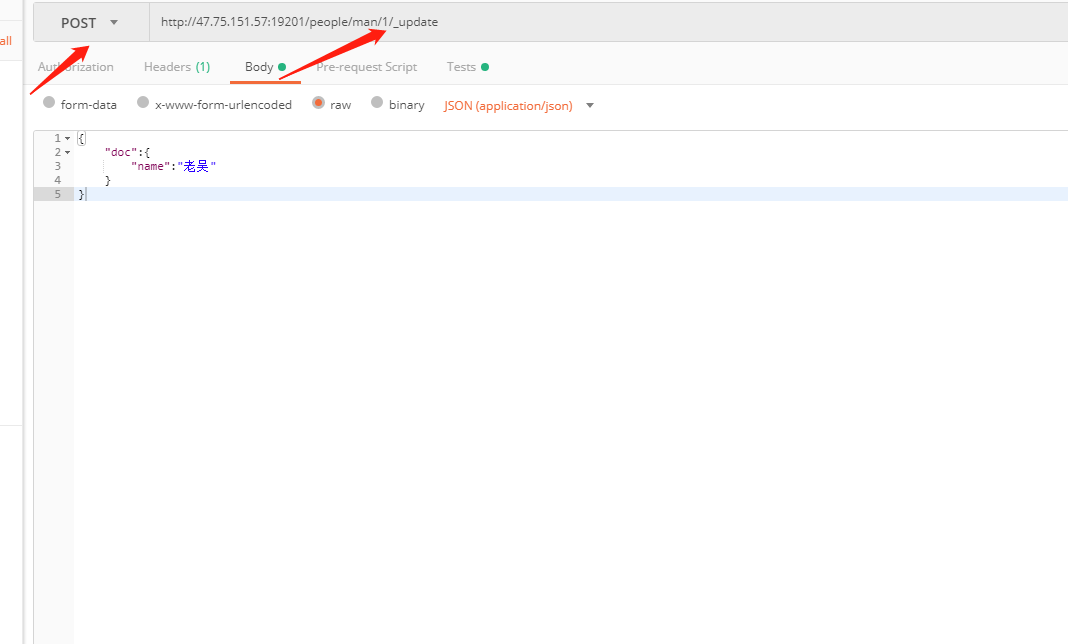


1. 自动生成文档ID

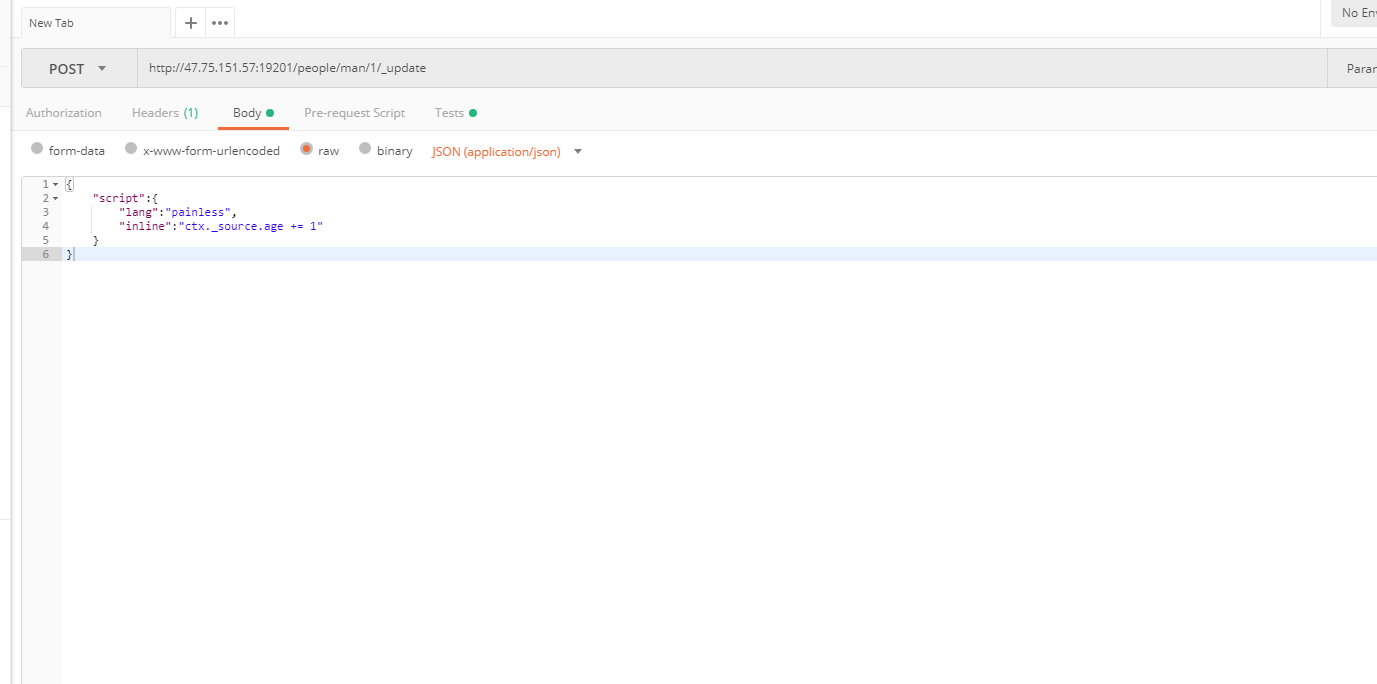


3 修改文档数据

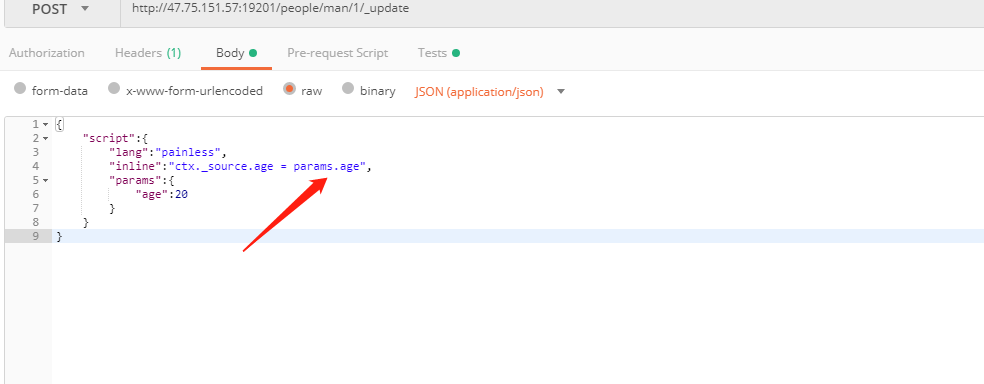
1. 直接根据ID修改



2》脚本形式



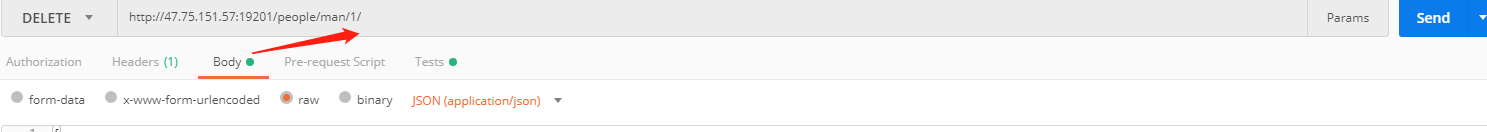
ES支持很多脚本语言 painless es内置的语言 脚本操作age+1



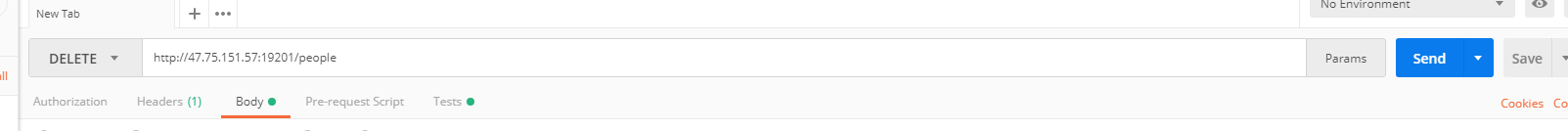
让age等于params里的值

4.删除操作

1》删除文档 根据ID删除

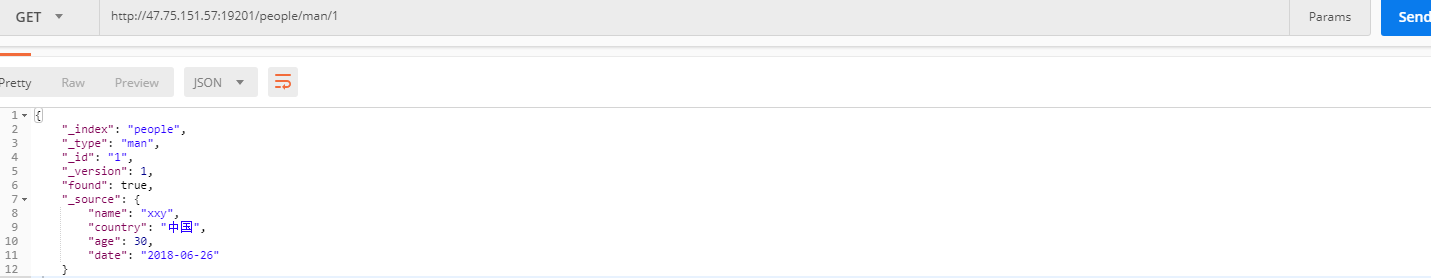


2》删除索引

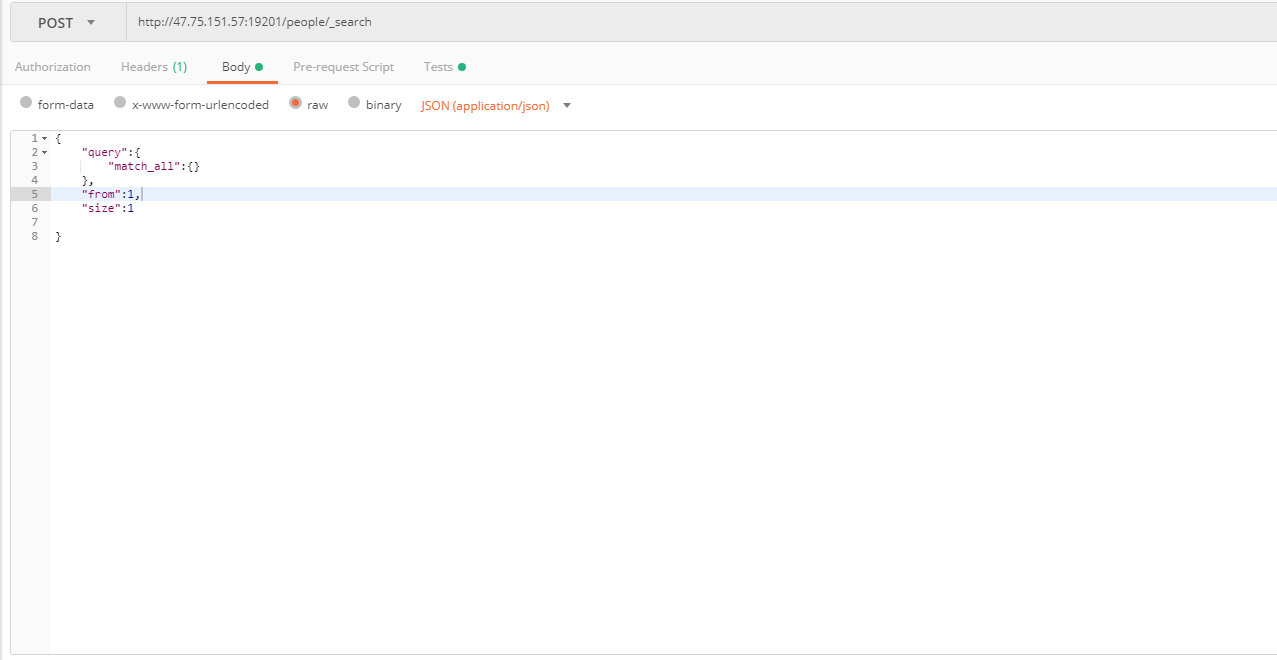


5.ES常用查询语法

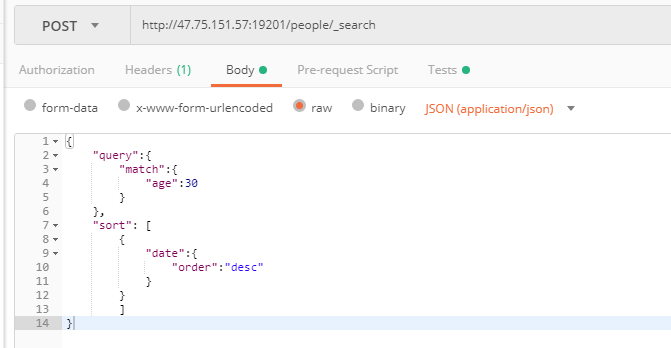
1.简单查询



2.条件查询

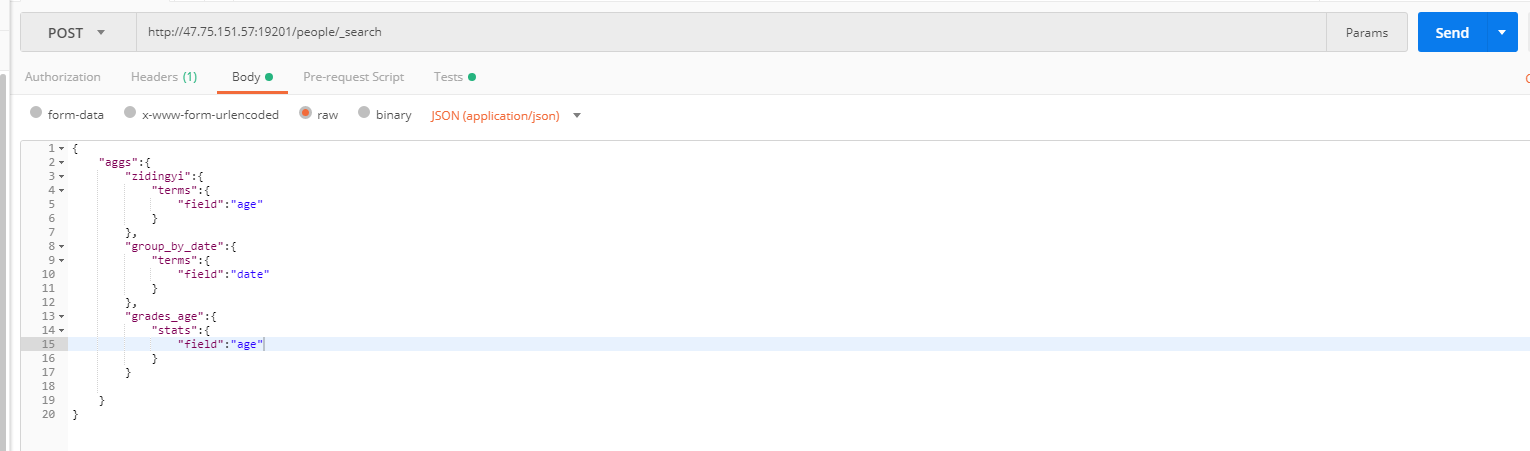


1：查询以query开始 match\_all查询全部 默认返回10条数据 from 表示从哪里返回，size表示 返回的数据的多少

2. 

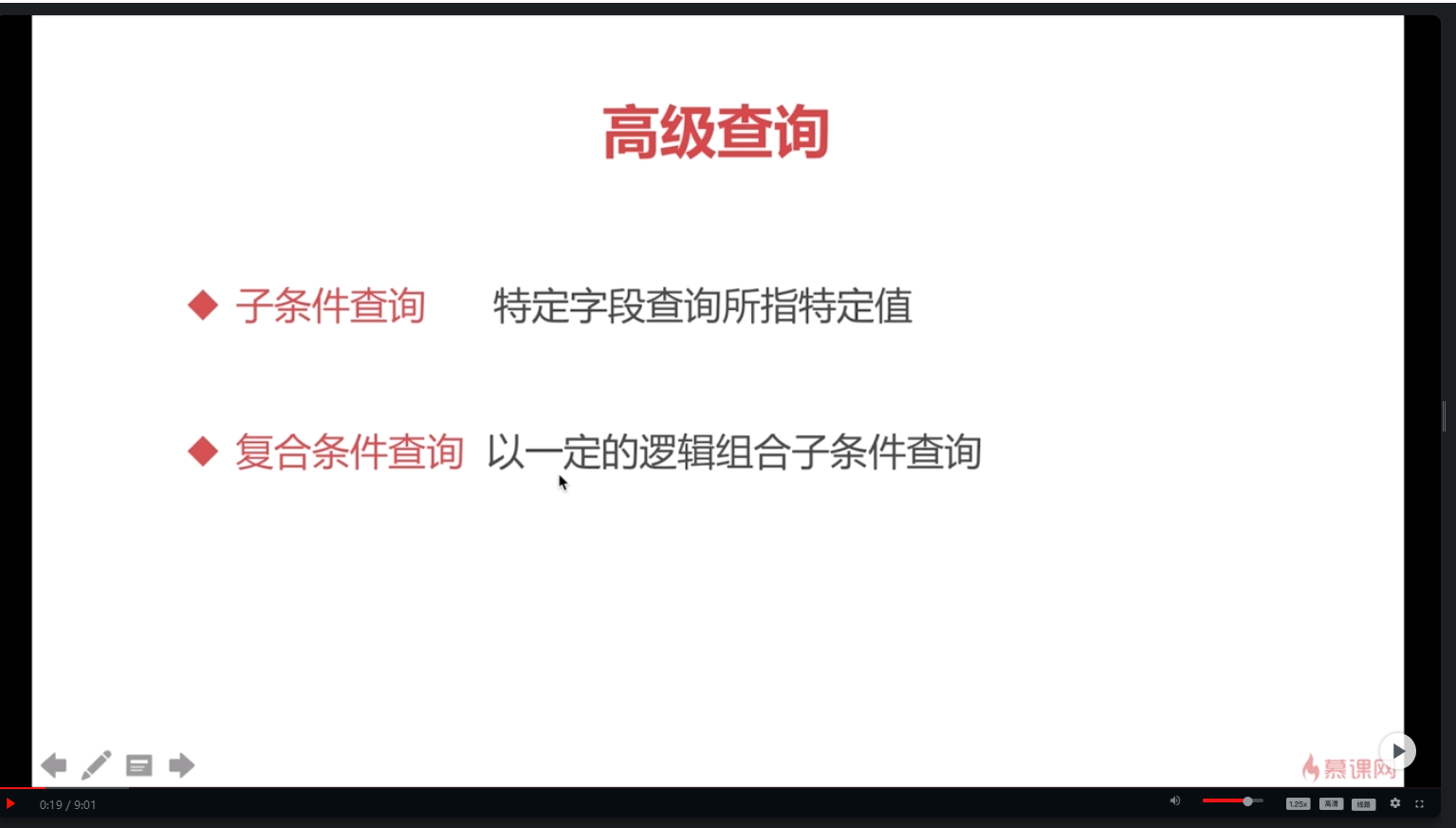
条件查询 使用match 添加查询条件 sort添加排序 根据时间降序排列

3.聚合查询

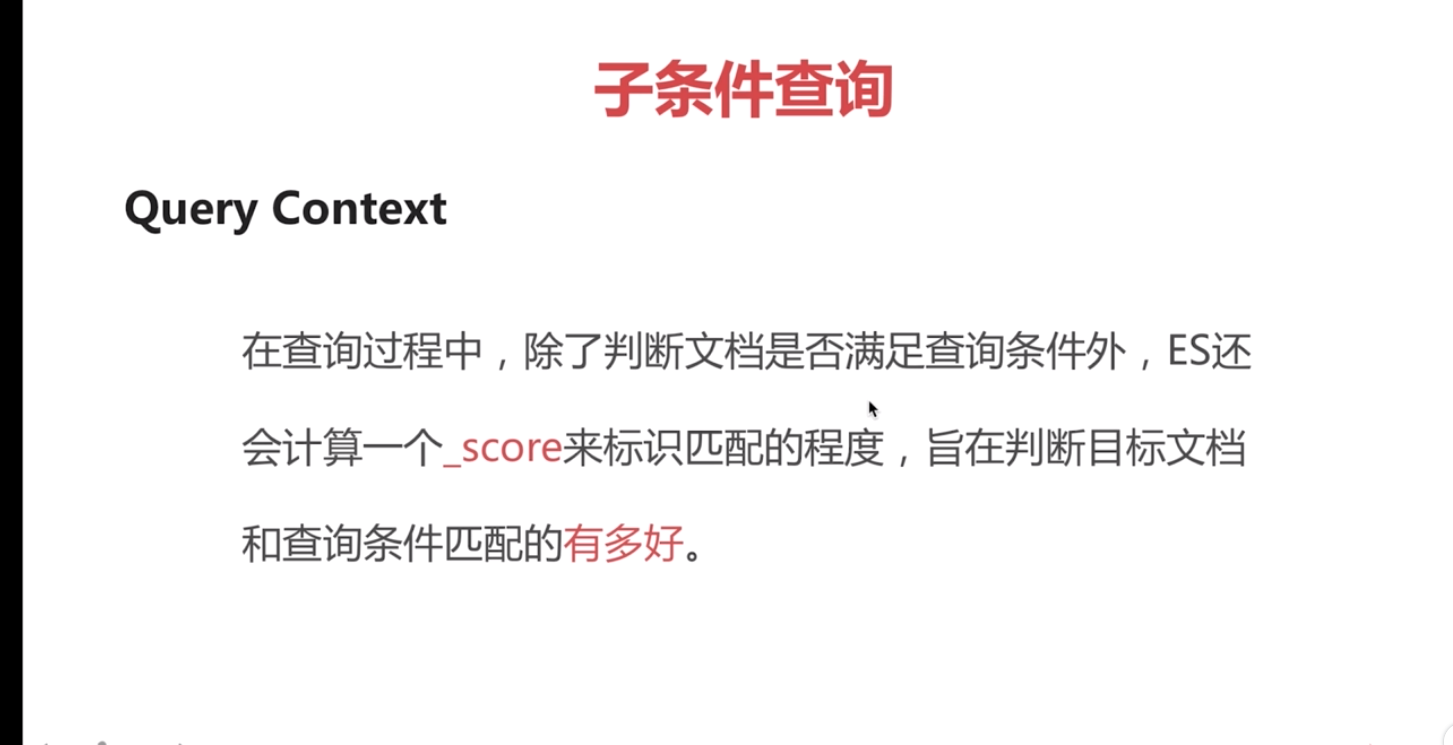


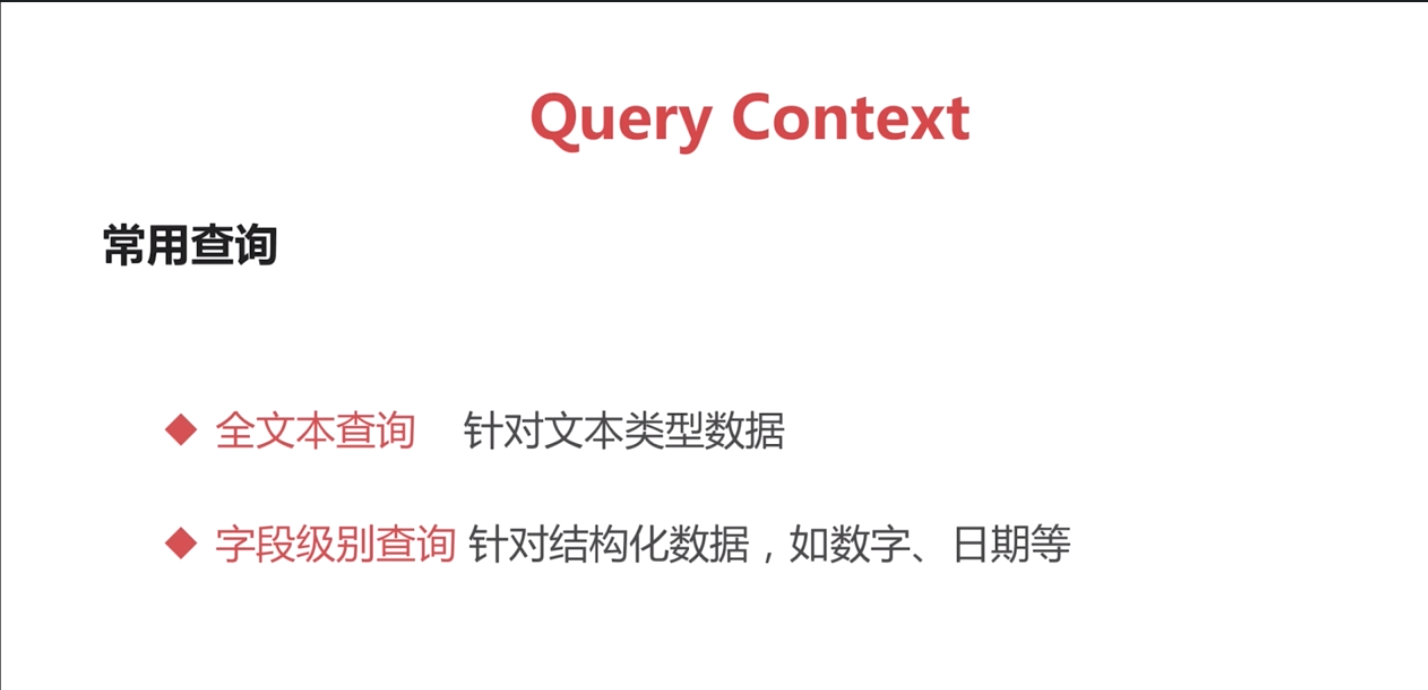
聚合查询关键字 aggs ，自定义聚合名字 terms(具体项) 关键字 下根据字段聚合分类

Stats 关键字下的 根据age 做计算如最大值、最小值、平均值等

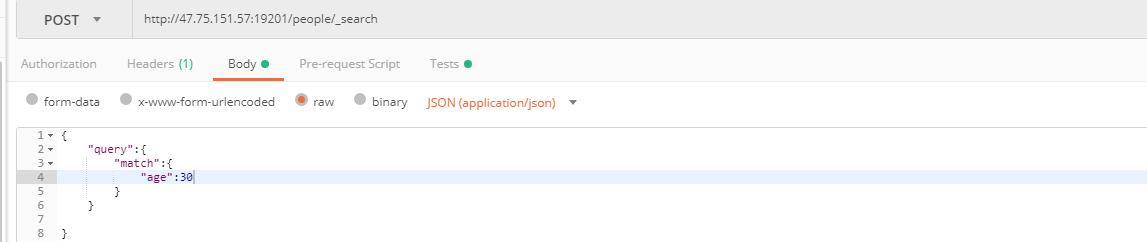


3》高级查询

1. 子条件查询  
   



1. 全文本查询 模糊查询



Match 模糊匹配 会匹配多个词语

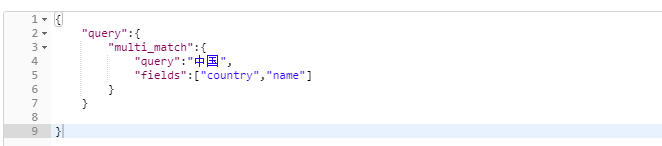
2习语匹配

关键字 match\_pharse

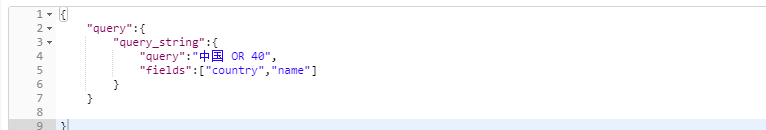
会匹配整个

3 多个字段匹配

关键字 multi\_match

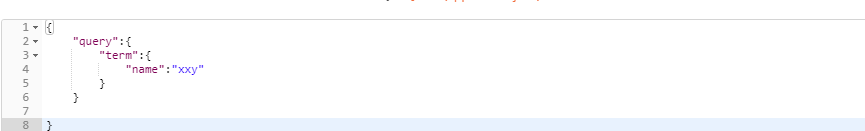


从两个 county和name 字段里面查询匹配 “中国”



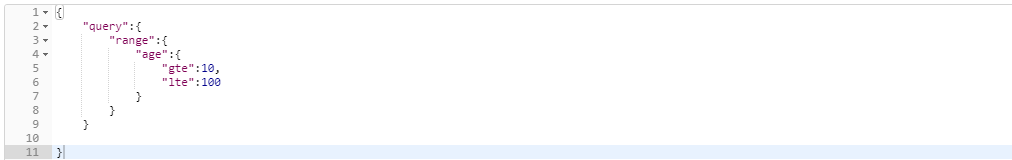
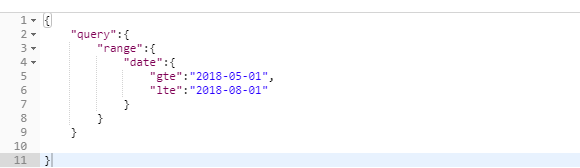
查询 string 查询是中国或者是 40 的STRING 从 country或者name里面查询

2》结构化查询

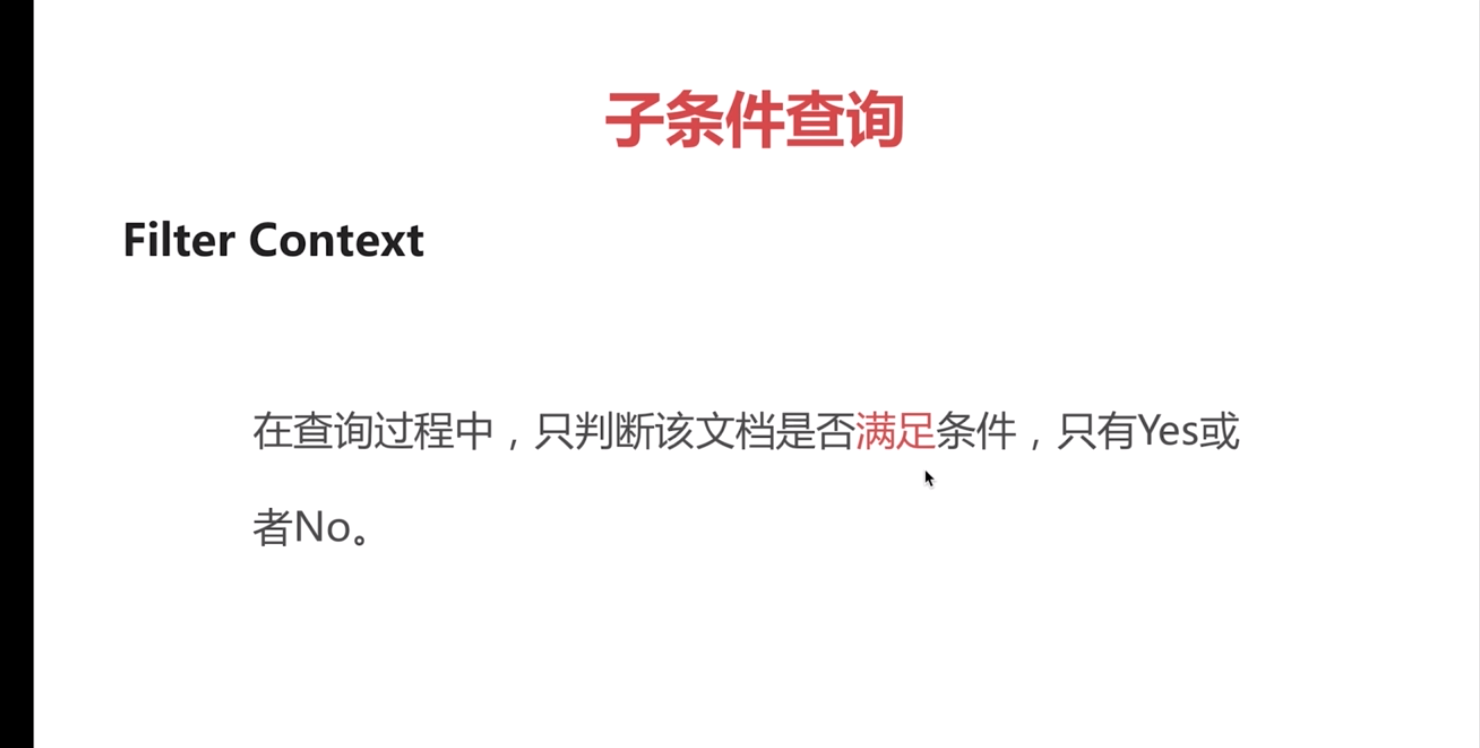


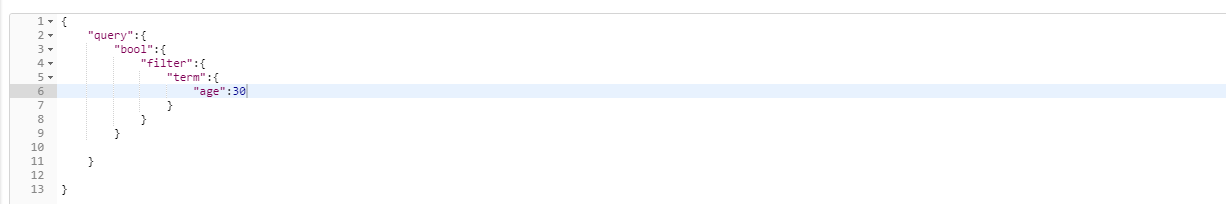
Term 具体项

Range 范围

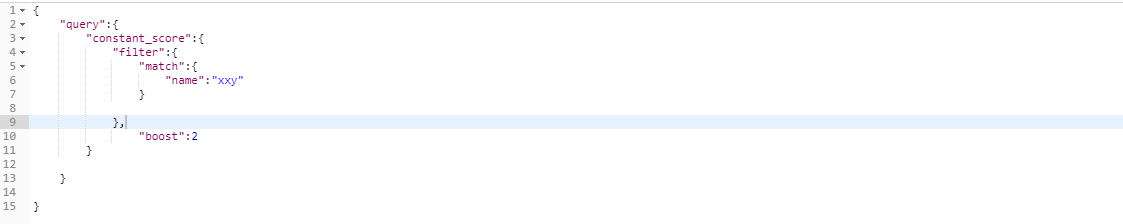


查询年龄10到100的数据也可以对日期进行比较



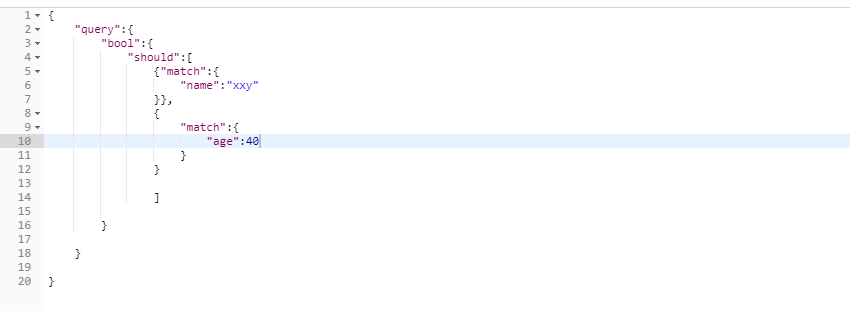


1. 复合条件查询
2. 固定分数查询



Boost 设置分数

2



Should关键字有或者的意思

3．

Must关键字表示必须同时满足条件

Must\_NOT 关键字表示必须不是

中文分词索引优化

最近开始学习SpringCloud，想给项目里的模块单独打包，方便多次部署。可是在网上找了很多的解决方案，都没有操作成功的。最后查看[官网的文档](http://link.zhihu.com/?target=https%3A//docs.spring.io/spring-boot/docs/current/reference/html/build-tool-plugins-maven-plugin.html)才找到解决方案，顺便把成功的打包方式贴出来吧。

首先，修改pom.xml文件

**<?xml version="1.0" encoding="UTF-8"?>**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

*<!-- ... -->*

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<version>2.0.3.RELEASE</version>

<executions>

<execution>

<goals>

<goal>repackage</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

这里主要是添加了版本和<executions>标签。

然后，使用maven命令打包：

mvn clean package

打包完成后，就可以直接用java -jar module.jar运行了

还有一点要注意的是，Spring Boot 2以上的版本需要用到JDK8以上，低版本打包会出错。

#GIt练习项目

这个项目用于数据据系统设计小组成员练习git 请不要上传其它东西。

0.打开[这个项目](https://git.oschina.net/xring/osgit)，猛戳「Fork」按钮  
1.把 Fork 到你账号下的项目 Clone 到本地：git clone [https://git.oschina.net/你的帐号/osgit.git](https://git.oschina.net/%E4%BD%A0%E7%9A%84%E5%B8%90%E5%8F%B7/osgit.git)  
2.在项目目录下，创建一个新分支来工作，比如新分支名叫 dev，则：git branch dev  
3.切换到新分支：git checkout dev  
4.运行 git remote add upstream <https://git.oschina.net/xring/osgit> 把原始项目库添加为上游库，用于同步最新内容  
5.在目录中进行一些修改  
6.提交你的工作：git add “这里写你修改过的内容”，然后 git commit -m '工作描述’。在这些过程中，你都可以用 git status 查看状态  
7.运行 git remote update 更新  
8.运行 git fetch upstream master 拉取上游库的更新到本地  
9.运行 git rebase upstream/master 将上游库的更新合并到你的 dev 分支  
10.运行 git push origin dev:master 将你的提交 push 到你的库中  
11.登录 Github，在你 Fork 的项目页有个 「Pull Request」按钮，点击它，填写一些说明，然后提交

重复步骤 5～11   
  
请大家多多练习