# RestAPI操作索引库

前面已经学习了如何使用DSL语句去操作ES，不管是索引库操作、还是文档操作。不过，我们作为一名java程序员，肯定需要使用java代码去操作ES，要想实现java代码去操作ES，就必须使用ES官方提供的RestClient来实现。

## 什么是RestClient

事实上，RestClient是ES官方提供了各种不同语言的客户端，客户端的作用就是用来帮助程序员组装DSL语句，通过http请求发送给ES，这样不就能操作ES了吗！之前我们操作ES是通过在DevTools控制台受些DSL语句实现，现在有了RestClient，就不用手写了，由RestClient客户端帮我们去弄了，你只需要把请求发送给ES服务器即可！

这些客户端的本质就是组装DSL语句，通过HTTP请求发送给ES，官方文档地址为：

<https://www.elastic.co/guide/en/elasticsearch/client/index.html>



可以看到，ES提供了各种语言的客户端，第一个就是JAVA语言客户端，而JAVA客户端又包含两部分，一是Java Low Level Rest Client，另一个是Java High Level Rest Client，最早出现的是Low Level，后来才有了High Level，High Level是基于Low Level做了进一步的封装，用起来更加方便快捷，所以我们学习使用Java High Level Rest Client！

## 使用RestClient

根据资料提供的酒店数据创建索引库，我们通过hotel案例来学习RestClient，学习使用JavaRestClient实现创建、删除索引库，判断索引库是否存在！

基本步骤如下：

1：导入课前资料Demo；

2：分析数据结构，定义mapping属性；

3：初始化JavaRestClient；

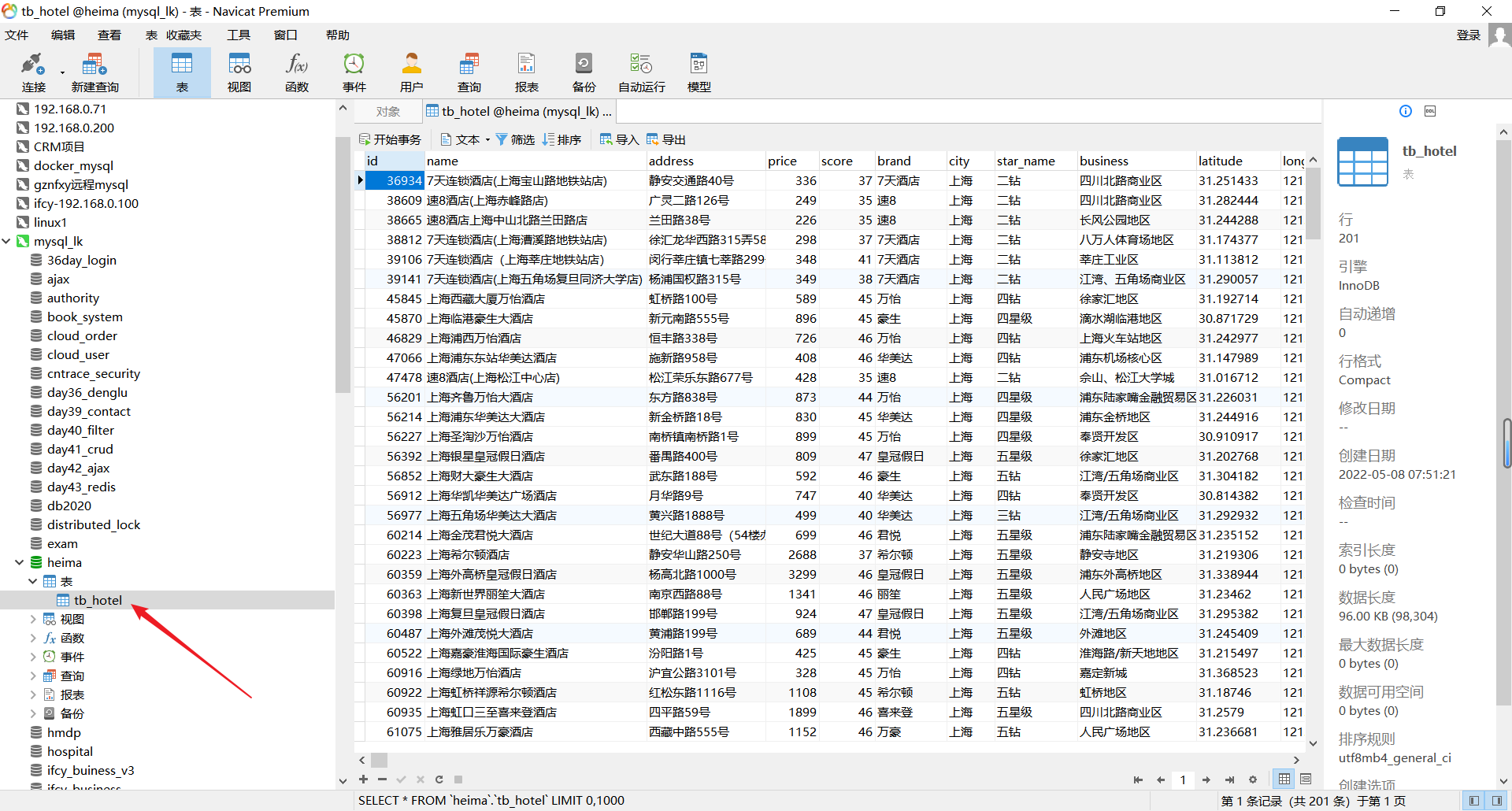
4：利用JavaRestClient创建索引库；

5：利用JavaRestClient删除索引库；

6：利用JavaRestClient

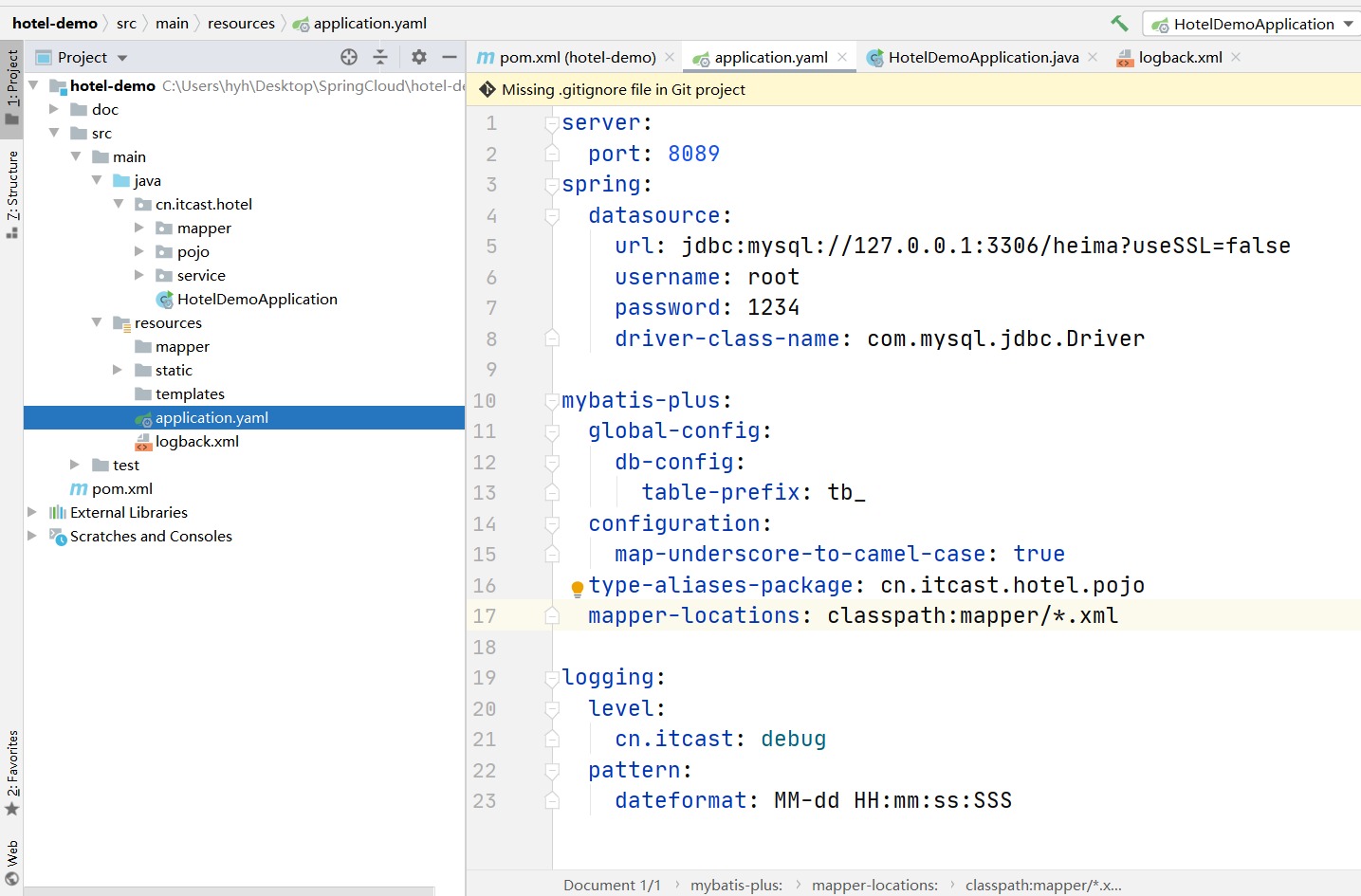
## 步骤

### 1：导入tb\_hotel.sql文件运行，创建heima数据库



### 2：导入项目hotel-demo





### 3：编写mapping映射（说白了就是编写mapping创建索引库呗）

在刚刚已经导入了酒店hotel项目的相关是数据，接下来基于酒店的数据结构去分析一下，尝试编写一下hotel的mapping映射，我们知道mapping主要考虑以下的问题：

1：字段名叫什么?

2：字段数据类型是什么?

3：字段是否参与搜索?

4：字段是否分词？

5：字段如果分词，分词器是什么？

其中字段名和数据类型一目了然，我们可以基于数据库的表结构得出，字段名一目了然，数据类型则可以把数据库中的数据类型转为ES中对应的数据类型；

至于是否搜索以及是否分词这两个就比较特殊了，这两者是与业务强相关的，比如当前是酒店业务，那么酒店名称一定是要参与搜索的字段，所以酒店名这个字段的index应该是true，即参与搜索，分词显然也是要分词的，因为酒店名比较长；为了提高搜索概率，分词器可以选用ik\_max\_word，综上，是与业务相关的，要根据业务场景综合考虑。

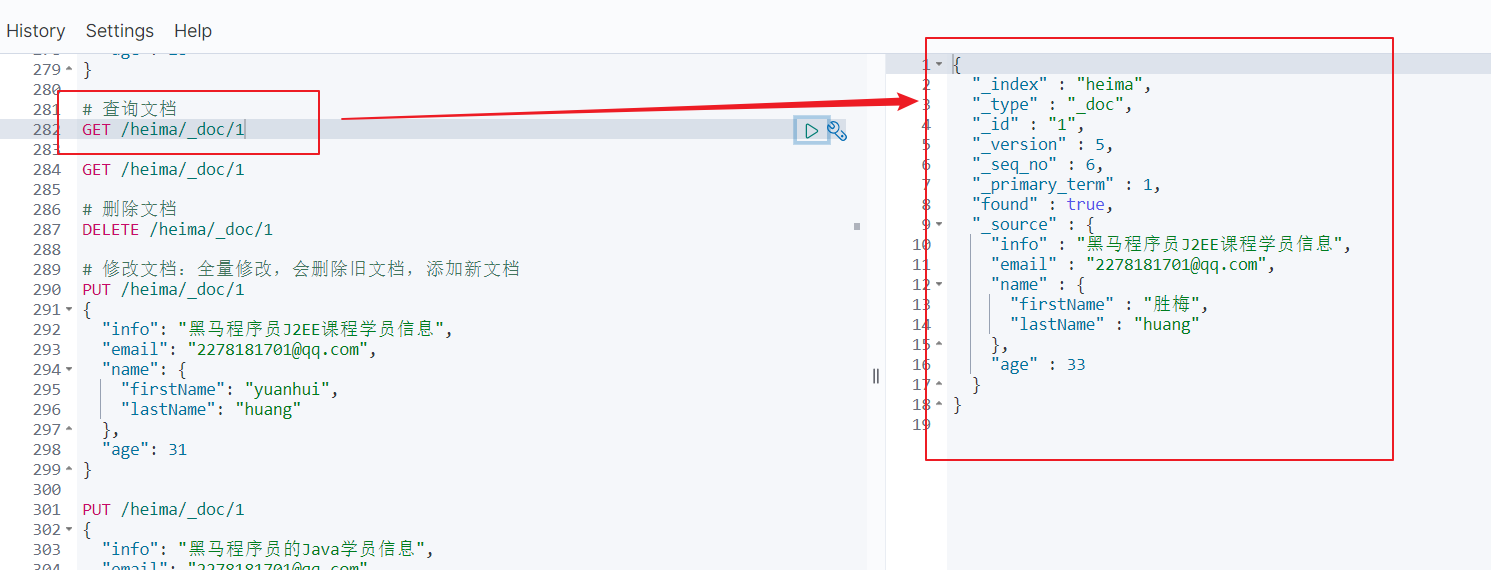
例如，根据下图中的tb\_hotel表，我们去分析下面的字段，去编写映射mapping。



映射分析过程如下：

#### id字段

首先id在MySQL的数据类型是bigint，对应Java的数据类型是long，但是在ES中不能是long，而应该是字符串，请看下面，图中查询是的id明明是数字1，但是在ES中却变成了一个字符串”1”，也就是说在ES中id这个字段比较特殊，都是字符串类型！



又因为id是一个整体不可分割，不分词，所以是keyword，因为索引库id这个字段比较特殊，是字符串类型。当然了id肯定要参与将来的CRUD增删改查，所以id的index值走默认true就行了；

#### name字段

name酒店名称显然一定是要参与搜索的字段，所以字段的index默认就好，应该是true，即参与搜索，name酒店名称数据类型先让你是字符串text，分词显然也是要分词的，分词器为了提高搜索概率，可以选用ik\_max\_word;

#### address

address这个字段，虽然和name字段很相似，都是内容很长的字符串，但是虽然长得很像，但是你要基于业务去判断分析，因为一般不会根据地址去搜酒店！所以address显然是不需要搜索的，不需要搜索就没有分词的必要，那么type就应该选择为keyword，同时也不需要分词其啊！又因为不搜索，那么index的值不能为默认true，应该为false；

#### price & score

price价格、score评分显然是要参与搜索的，数据类型显然是integer;

#### band & city & startName & business

brand品牌、city城市、startName星级、business商圈显然都是要参与搜索的，所以index默认值true，同时不需要分词，所以类型是keyword。

#### latitude和longtime

latitude和longtime经纬度地理坐标，代表的是酒店的坐标，既不是float也不是double，因为地理坐标在ES中比较特殊，有特殊的表现方式，有两种表现方式：

**geo\_point：**代表的是地图上的一个点，由维度（latitude）和经度（longtime）确定地图上的一个点。例如”32.8752345, 120.2981576”；

**get\_shape：**代表的地图上的一个区域，有多个geo\_point组成复杂几何图形，代表地图上的一个区域。例如一条直线：”LINESTRING (-77.03653 38.897676, -77.009051 38.889939)”。

显然酒店应该是地图上的一个点，即数据类型应该是geo\_point就可以了，由上看出值是由经度和维度拼在一起的字符串，虽然值字符串，但是类型是叫geo\_point，所以，可以创建一个叫location，类型是geo\_point的字段，由经度和维度拼在一起。

#### pic

pic是酒店图片，是一个url路径，显然不需要分词，即type为keyword，其次没有人会根据url去搜索，显然index为false，即既不分词也不搜索。

#### 问题

至此，我们就定义完酒店的所有字段了，但是，现在会发现有一个问题，即映射mapping中的name酒店名、brand品牌、business商圈、city城市、startName星级这些字段都要参与搜索，这些字段都要参与搜索，也就意味着将来用户输入关键字去搜的时候，可能要根据多个字段去搜，也就是你的查询条件不是一个值，而是多个值。比如：**广州珠江新城五星级丽兹卡尔顿酒店**

可以想一下，在用ES去做搜索的时候，你是根据一个字段搜效率高，还是根据多个字段搜效率高？你可以对比一下数据库你就知道了，显然是根据一个字段！但是现在的需求就是希望用户输入酒店名称能搜到、用户输入品牌能搜到、用户输入商圈也能搜到，就是要根据多个字段，同时又要搜索性能好，那该怎么办？

#### 字段拷贝copy\_to到符合属性

ES给我们提供了copy\_to属性来解决以上问题：问题就是我在搜索的时候想要根据多个字段去搜，但是又想搜索效率高！字段拷贝可以使用copy\_to属性将当前字段拷贝到指定字段，比方说此处想根据name酒店名、brand品牌、business商圈、city城市、startName星级这些字段去搜，那么就可以把这些字段都拷贝到一个字段：all字段里面，all字段的类型显然是text，因为是输入的搜索内容是可以多个字段，分词器因为要提高搜索命中率，显然可以用ik\_max\_word，此时，意味着这一个all字段就同时具备了name酒店名、brand品牌、business商圈、city城市、startName星级这些字段的值，也就是说实现了在一个字段里搜索到多个字段的内容，而且这种拷贝ES做了优化，并不是真的把文档拷贝进去了，而只是基于此做了倒排索引，所以在查的时候是看不到all这个字段的，好像不存在一样，但是搜却可以根据all字段去搜，这个all字段又叫复合字段。

#### Mapping示例如下：

**PUT /hotel**

**{**

**"mappings": {**

**"properties": {**

**"id": {**

**"type": "keyword"**

**},**

**"name": {**

**"type": "text",**

**"analyzer": "ik\_max\_word",**

**"copy\_to": "all"**

**},**

**"address": {**

**"type": "keyword",**

**"index": false**

**},**

**"price": {**

**"type": "integer"**

**},**

**"sorce": {**

**"type": "integer"**

**},**

**"brand": {**

**"type": "keyword",**

**"copy\_to": "all"**

**},**

**"city": {**

**"type": "keyword"**

**},**

**"starName": {**

**"type": "keyword"**

**},**

**"business": {**

**"type": "keyword",**

**"copy\_to": "all"**

**},**

**"location": {**

**"type": "geo\_point"**

**},**

**"pic": {**

**"type": "keyword",**

**"index": false**

**},**

**"all": {**

**"type": "text",**

**"analyzer": "ik\_max\_word"**

**}**

**}**

**}**

**}**

#### 总结

我们在定义mapping映射的时候，主要考虑的问题就是：

1；字段的名字？

2：字段的类型？

3：text类型字段是否分词，分词的话分词器是什么？

4：字段是否要索引？

5：如果想同时根据多个字段搜，可以用copy\_to？

6：地理坐标的特殊表单方式geo\_point和get\_shape

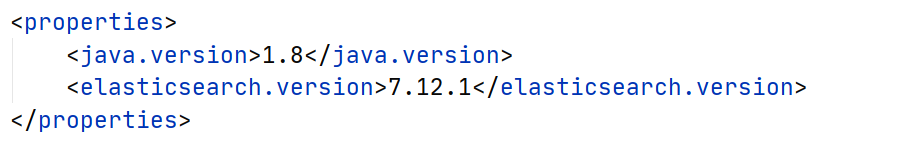
### 4：初始化JavaRestClient

#### 4.1：引入ES的Java Rest High Level Cilent依赖

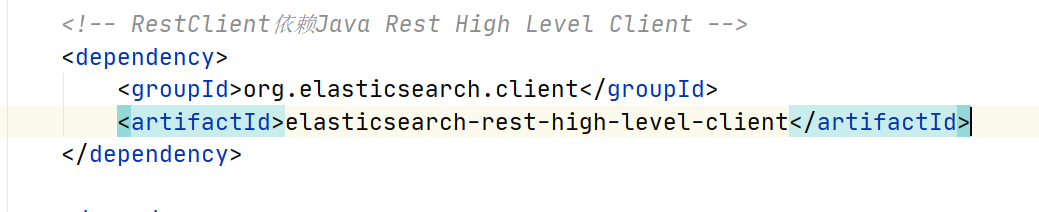


#### 4.2：因为SpringBoot默认的ES的版本是7.6.2，所以我们需要覆盖默认的ES版本，即必须和ES服务器的版本保持一致

这个项目是SpringBoot项目，SpringBoot项目的父工程中管理者所有常见依赖的版本，其中ElasticSearchde的版本就定义成了7.6.2，所以要覆盖SpringBoot父工程中指定的7.6.2的ES版本：

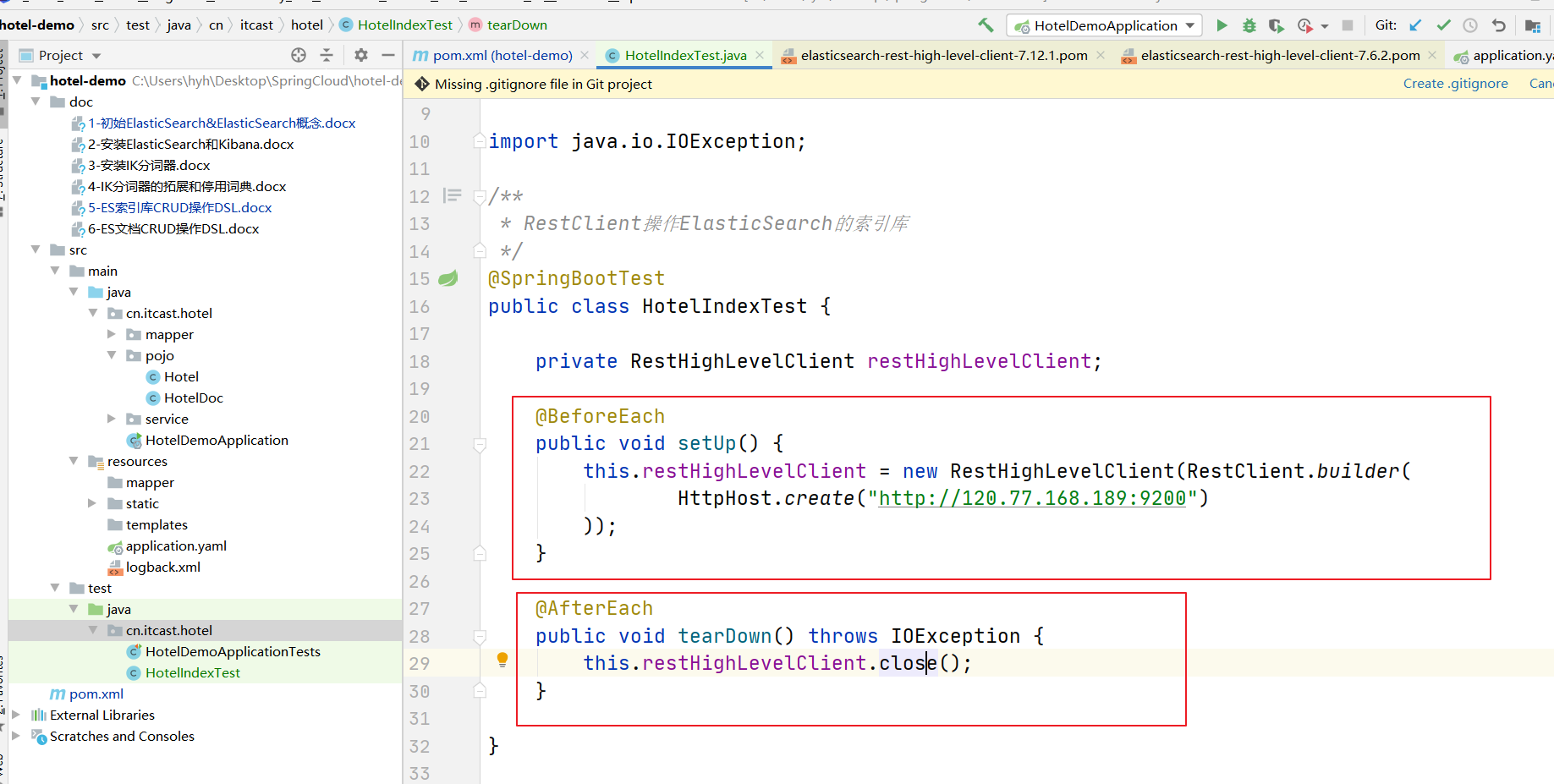


ES依赖版本就自然换成了7.12.1，在依赖中就不用指定版本了：



#### 4.3：初始化RestHighLevelClient

ES集群的话就写多个HttpHost.create()

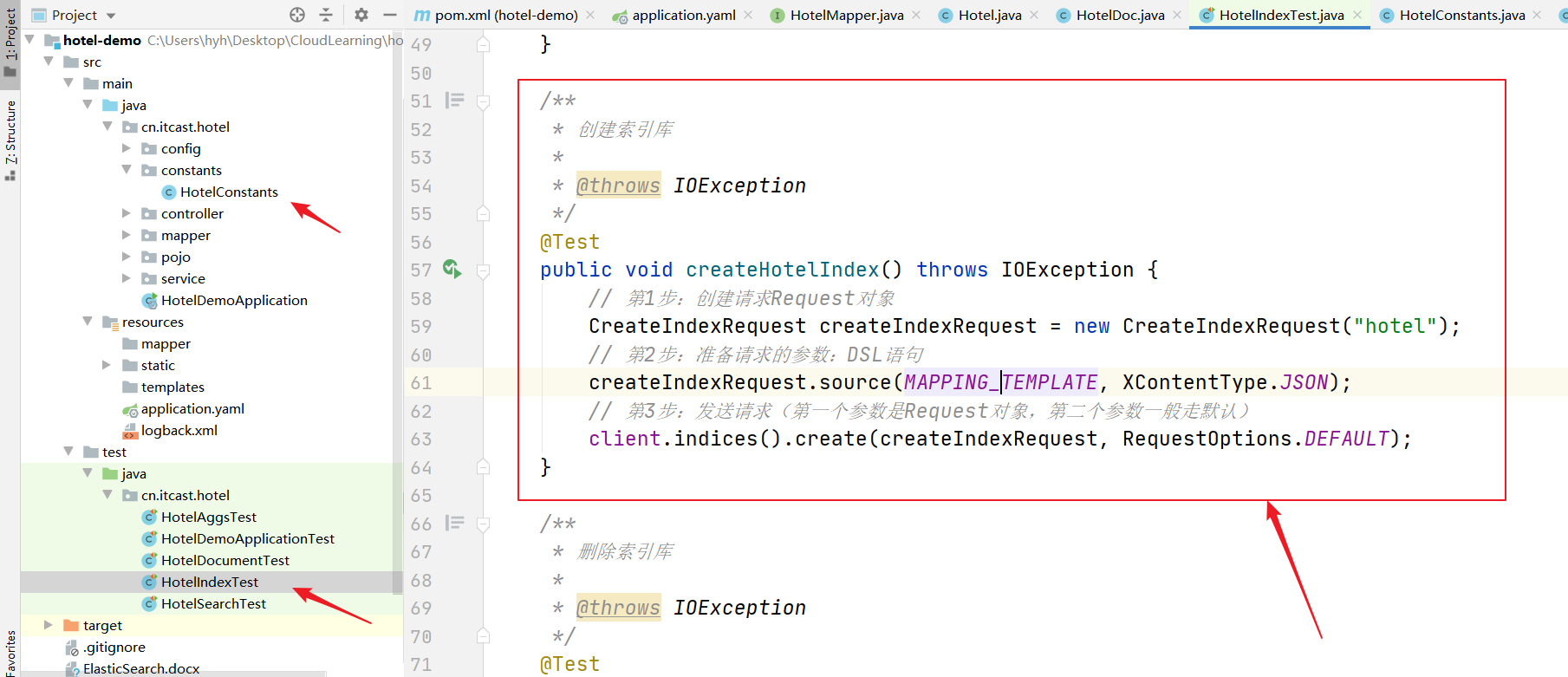


#### 4.4：测试客户端是否创建成功

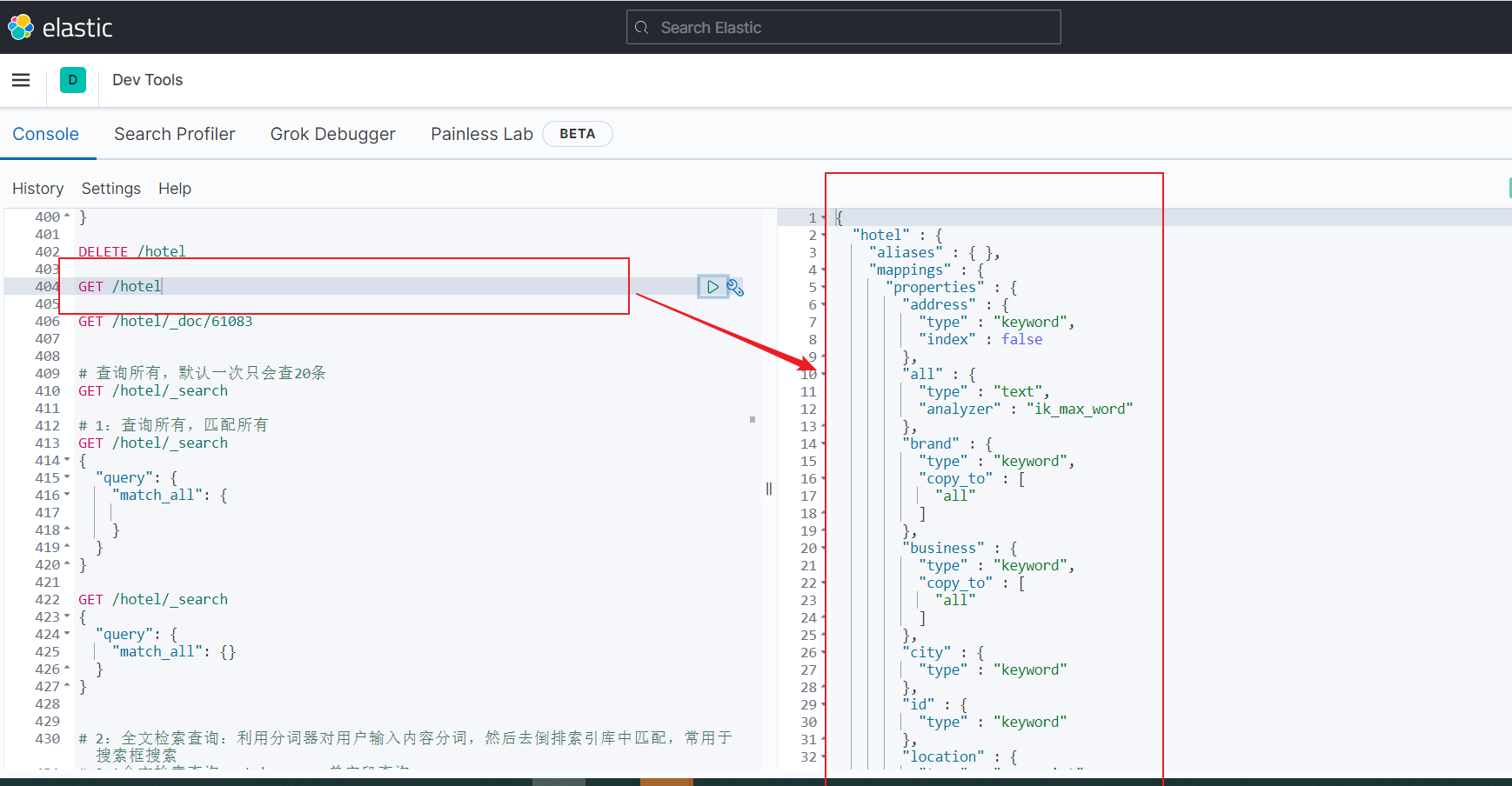


### 5：创建索引库

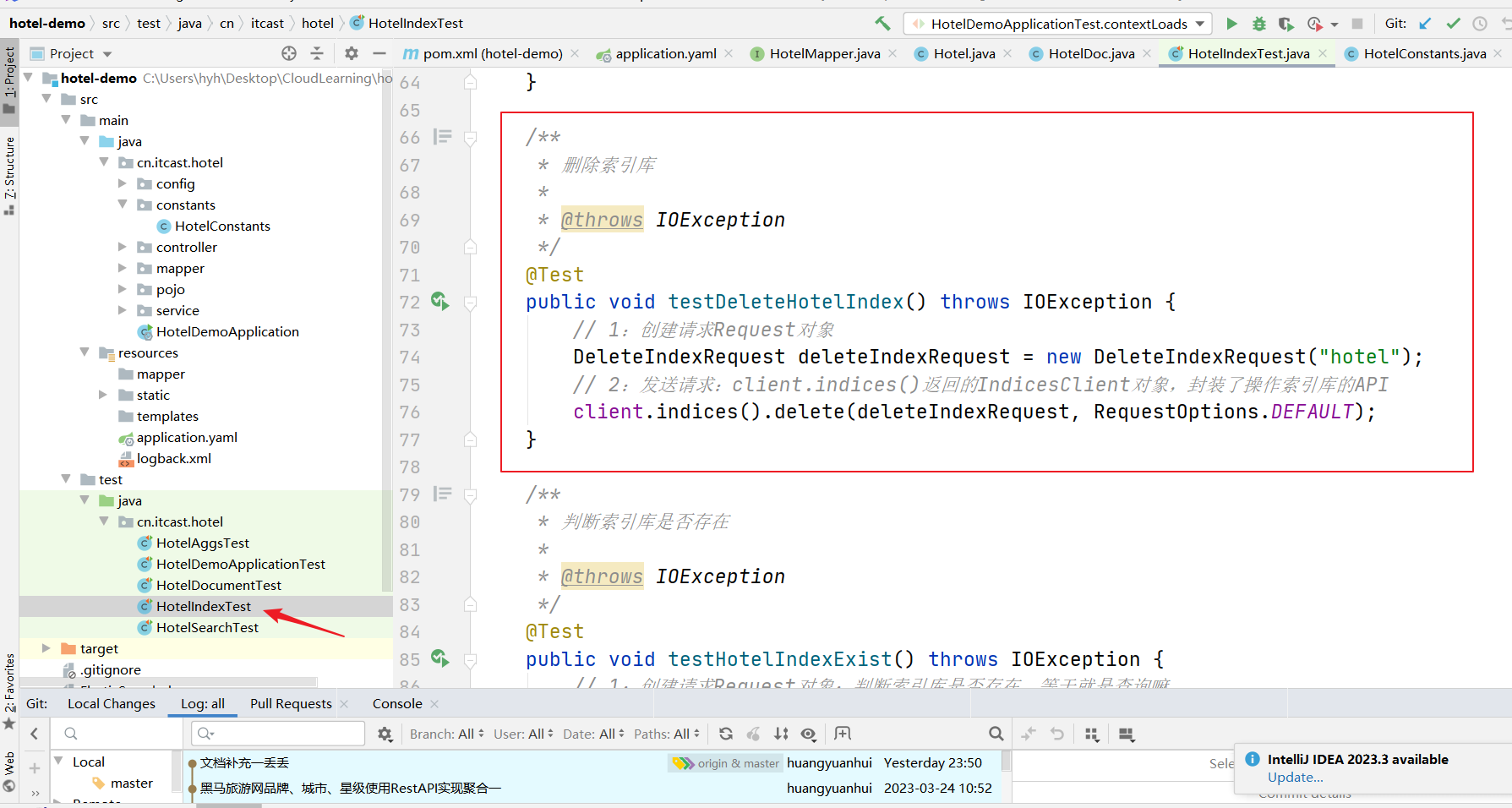
同学们，我们已经准备好了酒店索引库hotel的mapping映射，并且还完成了JavaRestClient的初始化，可以说是万事俱备只欠东风了，接下来，就看看如何利用JavaRestClient来去创建索引库：



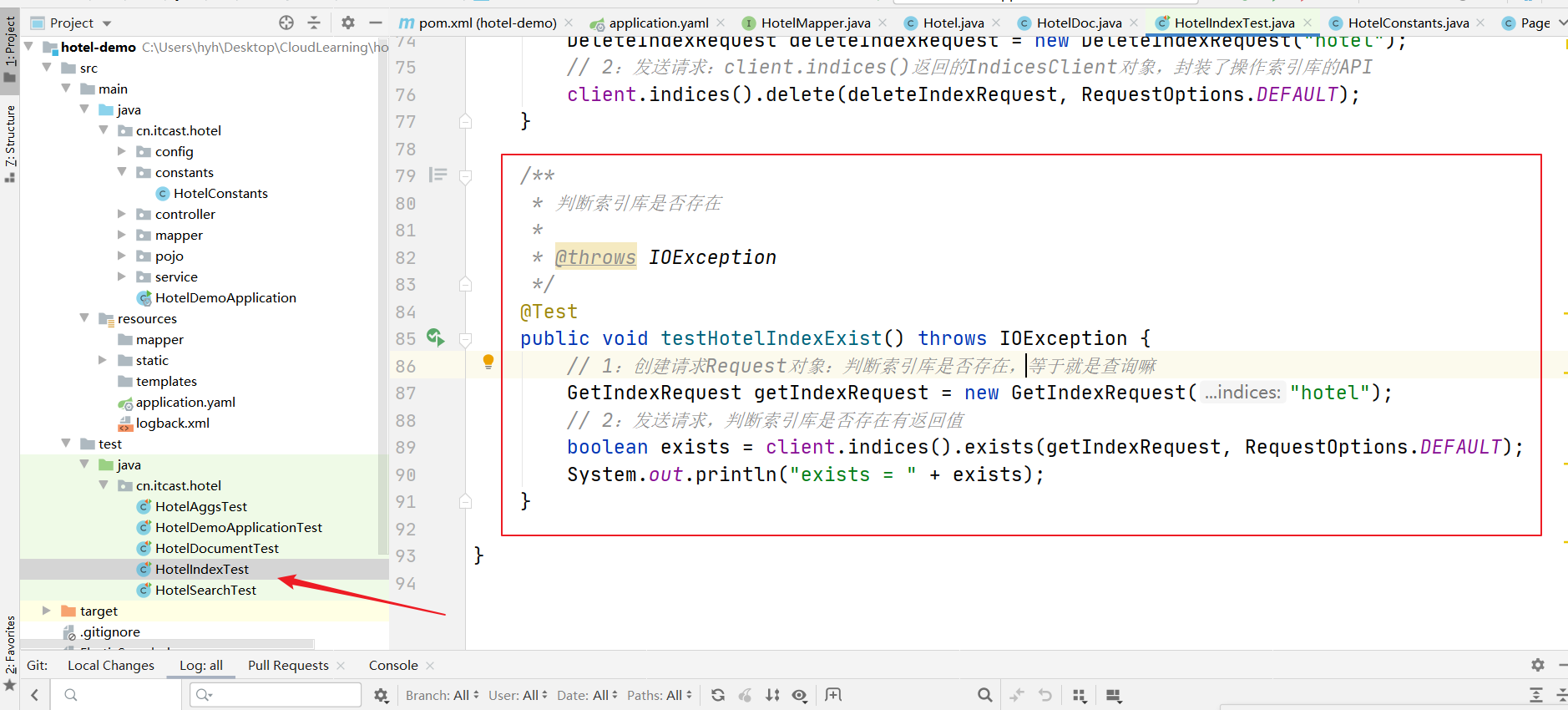
运行测试案例的创建索引库方法，到DevTools控制台查看索引库hotel是否创建成功，你看你看你看，查询到了，证明我们索引库的创建没问题！



### 6：索引库删除



### 7：判断索引库是否存在（就是查询索引库）



## 注意点

### 1：RestHighLevelClient是ES官方提供的操作ES的API；

### 2：所有操作索引库的api都在RestHighLevelClient对象的indices()方法中；

### 3：JAVARestClient是Rest风格的API，新增就是create，删除就是delete，是否存在就是exists，所以很容易学习

## 总结：

索引库操作的基本步骤：

1：初始化RestHighLevelClient；new RestHighLevelClient

2：创建XxxIndexRequest对象，XXX分别是CREATE、Get、Delete

3：准备DSL（当然了，只有CREATE时需要）

4：发送请求。调用ResetHighLevelClient#indices().xxx()方法，xxx是create、exists、delete