1. 什么是lucene

它是apache组织的一个全文检索引擎工具包

2. 作用:

可以使用lucene构建像百度,谷歌这样的全文检索引擎系统

3. lucene的原理:

全文检索算法(倒排索引表算法):

把需要检索的源文件的内容读取出来, 进行切分词, 将一个一个的词做成索引, 搜索的

时候可以先搜索索引,通过索引找文档, 那么这个过程就叫做全文检索

切分词: 将一句话切分成一个一个的词, 去掉空格,标点符号, 停用词(的,地, 得, a, an等)

将大写字母转成小写字母

优点: 速度快, 不会随着数据量的增大而降低查询效率

缺点: 目录会占用额外的磁盘空间

field域类型的思想问题:

a. 是否分词: 分词的目的是为了索引

是: 需要创建索引, 并且分词后有意义

例如: 名称, 描述

否: 不需要进行索引 或者是 分词后无意义的域就不进行分词

例如: 主键id, 身份证号

b. 是否索引: 索引的目的是为了查找,搜索

是: 需要根据这个域进行查询那么就需要索引

例如: 名称, 描述.....

否: 不需要根据这个域进行查询那么就不需要进行索引

例如: 图片

c. 是否存储: 存储的目的是为了显示出来

是否存储和是否索引无关, 索引影响的Index, 是否存储影响的Document

(我们查询的时候是从索引表中获取document的信息,也就是说有索引就能查询,再通过索引表的结果去找document,但是document有没有是是否存储管的,存储了你就能取出想应的信息,没存储那么你能查询但是结果却是null)

是: 名称, 价格, id等

否: 描述, 描述内容中含有大量图片文字, 这样如果存储会占用大量的磁盘空间, 不划算.

如果不存储,在需要的时候可以根据id写一条sql语句来进行查询数据库, 可以把

描述给取出来

在lucene中数字类型是需要分词的, 这个是lucene底层封装的算法的要求, 我们不可以修改,

所以定死了必须分词

索引和文档的维护(更新操作):

对索引和文档的更新操作是, 根据条件先进行查找, 如果找到了就删除,

让后将更新的内容添加到索引和文档的最后的位置

booleanQuery中:

注意: 如果查询的时候条件使用单个MUST\_NOT 或者多个条件中都是MUST\_NOT是无效的

相关度排序

在创建索引的时候会为每个域分配权重.

当查询的时候lucene会自动根据查询的关键字进行计算, 和关键字最匹配的排在最上面,

和关键字不太匹配的逐渐排在后面.

Lunece的基础语法

1. 基础的查询语法：

域名:关键词

Name:java

1. 匹配所有文档：

\*:\*

1. 范围查询语法：

域:[最小值 TO 最大值]

size:[0 TO 1000]

注意：TO要大写

什么是solr

solr是apache组织的一个全文检索引擎系统, 放到tomcat下可以独立运行, 因为它就是一个系统(底层是lunece)

solr是干什么用的

在企业中我们可以使用solr来构建全文检索服务, 将solr部署到tomcat下它就可以独立运行,

它对外通过http的形式提供全文检索服务(索引和文档的增删改查)

solr和lucene有什么区别?

lucene是全文检索引擎工具包,就是一堆jar包, 我们可以用它来开发自己的全文检索引擎系统,solr相当于lucene的成品.

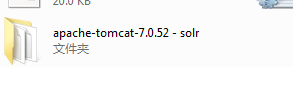
solr底层是用lucene来开发的, solr是一个现成的全文检索引擎系统, 放入tomcat下就可以独立运行了.底层是lucene.

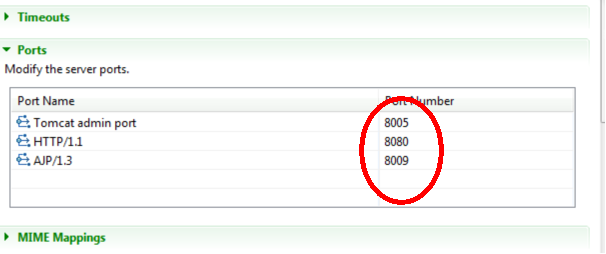
solrHome是solr的家, solrHome中可以有多个solrCore, 一个solrCore就是一个solr的实例.

实例和实例之间是相互隔离的, 每个实例都会对外单独提供索引和文档的增删改查服务

solr部署步骤:

a) 复制一台新的tomcat, 保证这个tomcat是干净的(新解压一个就好了或者在clean干净,看起是否干净的方法,点击bin目录下的startup.bat看他是否能跑,能跑就是干净的)需要修改一个端口号,2个tomcat的端口会有冲突(excplise想关联的好改)



b) 修改eclipse中tomcat的端口, 防止同时运行两个tomcat, 端口冲突

c) 将solr/example/webapps下的solr.war复制到tomcat/webapps下

d) 启动tomcat, 目的是为了解压, 启动成功后关闭tomcat, 将solr.war删除(一 定要删除在关闭状态下删除)

e) 复制solr/example/lib/ext下的所有放入, tomcat/webapps/solr/WEb-INF/lib下

f) 复制solr/example的solr文件夹到硬盘的根目录,并改名为solrHome

g) 将solrHome的路径放入到tomcat/webapps/solr/WEB-INF/web.xml中

<env-entry>

<env-entry-name>solr/home</env-entry-name>

<env-entry-value>E:\solrhome</env-entry-value>

<env-entry-type>java.lang.String</env-entry-type>

</env-entry>

h) 运行tomcat,部署成功

solrHome是solr的家, solrHome中可以有多个solrCore(刚才solrhome文件夹中的collection1就是一个实例), 一个solrCore就是一个solr的实例.

实例和实例之间是相互隔离的, 每个实例都会对外单独提供索引和文档的增删改查服务

注意:

1. solr中域要先定义后使用,如果没有定义就使用会报错

ERROR: [doc=002] unknown field 'xxxxxxxxxxadsadsfsadfsafdsad'"

2. solr中必须有主键域id, 如果没有会报错

"Document is missing mandatory uniqueKey field: id",

3. solr中没有单独的修改方法, 都是根据id进行查找,如果找到了将原有的内容删掉, 将

新内容添加进去, 如果没有找到, 这直接添加

solr界面根据id删除:

<delete>

<id>01</id>

</delete>

<commit/>

solr界面根据查询删除:

<delete>

<query>\*:\*</query>

</delete>

<commit/>

面试题:

& 和&& 区别?

&位运算符

&&逻辑运算符

进制分类: 十进制, 二进制, 八进制, 十六进制

位与&运算 相当于and 有假为假 1真 0假

10010000

11111111

10010000

Solr

我们在堆solr的使用的时候,通过solrj和和solr的操作界面进行交互,也就是通过程序控制浏览器,来达到伪操作的目的.

使用步骤;

创建solr连接.

SolrServer solrServer=**new** HttpSolrServer("http://localhost:8080/solr");

//创建document

SolrInputDocument document=**new** SolrInputDocument();

//doc填充数据

document.addField("id", "003");

document.addField("title", "我用鳄鱼上青铜");

document1.addField("id", "永刚是亲禽兽色");

document1.addField("title", "永刚是什么?禽兽");

//添加提交

solrServer.add(document);

solrServer.commit();

总结;solrj其实就是solr操作界面的jia包,通过它可以达到操作界面的目的.

Solr的修改和添加类似,先在索引中查找,找到后删除原有的doc,然后再添加后有的. 达到修改的目的.

删除;

solrServer.deleteById("id值")

solrServer.deleteByQuery(Query)

域:值;删除全部是 \*:\*

查询

//创建服务对象

//创建查询条件solrquery

//初始化query对象

//开始查询,返回返回结果response

//response上获取结果集

SolrServer solrServer=**new** HttpSolrServer("http://localhost:8080/solr");

//创建solrquery

SolrQuery query=**new** SolrQuery();

//初始化query对象

query.setQuery("\*:\*");

query.setQuery("id:003");

//调用查询方法,爬取response对象上的results

QueryResponse response = solrServer.query(query);

SolrDocumentList results = response.getResults();

**for** (SolrDocument solrDocument : results) {

System.***out***.println(solrDocument.get("id")+"\_\_\_\_\_\_\_\_"+solrDocument.get("title"));

}

注意事项;

Solr的域要先定义再使用.

solr中必须有id域.

Field域配置;

普通域

域名 ---name

类型----type(这里指定类型的根本目的是为了让其有对应的分词配置所以可以直接写分词器)

是否索引indexed

是否存储stored

是否必须有 required(没陪就报错)

是否多值---multivalued

<field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" />

动态域;dynamicField(模糊匹配域名,灵活性)

<dynamicField name="\*\_is" type="int" indexed="true" stored="true" multiValued="true"/>

主键域必须有没有会报错保存的时候需要仅仅有一个:<uniqueKey>

复制域(查询用的,和jsp中的pageContext有点像);copyField

<copyField source="cat" dest="text"/>

<copyField source="name" dest="text"/>

<copyField source="manu" dest="text"/>

<copyField source="features" dest="text"/>

<copyField source="includes" dest="text"/>

<copyField source="manu" dest="manu\_exact"/>

域的类型(先定义后使用)

<fieldType name="int" class="solr.TrieIntField" precisionStep="0" positionIncrementGap="0"/>

中文分词器配置=====

中文分词器是给solr使用的,儿子在tomcat的solr文件夹中

a) 将ik的jar包 放入tomcat/webapps/solr/WEB-INF/lib下

b) 在tomcat/webapps/solr/WEB-INF创建classes目录

c) 将ik的核心配置文件, 扩展词典, 停用词典复制到 /webapps/solr/WEB-INF/classes下

e) 在solrHome/collection1/conf/schema.xml中配置域的ik类型

Solr在后台写的时候.

1. 创建服务solrserver
2. 创建查询条件solrquery
3. 条件填充
   1. 设置默认查询域set(“df”,”域field”)
   2. Setquery(“查询字段”)开启高亮,添加高亮字段add,设置头和尾
   3. (注意在程序中如果前段没有给查询字段要设置成”\*:\*”),可以理解成主查询条件必须有
   4. 过滤条件(这些条件是在主查询的基础上才有的,注意过滤的含义)可以有多个
   5. 设置排序,start和rows
   6. 开始查得到response,获取results<Document>和高亮map<id,map<高亮字段名字,list<String>>>