# Lucene

**学习目标**

* 什么是全文检索，如何实现全文检索？
* Lucene实现全文检索的流程？
* 配置开发环境？
* 入门程序？
* 分析器的分析过程
* 索引库维护？
* 索引库查询？

# 1 全文检索是什么？

## 1.1 数据分类？

我们生活中的数据总体分为两种：结构化数据和非结构化数据。

**结构化数据：**指具有固定格式或有限长度的数据，如数据库，元数据等。

**非结构化数据：**指不定长或无固定格式的数据，如邮件，word文档等磁盘上的文件

## 1.2结构化数据搜索？

常见的结构化数据也就是数据库中的数据。在数据库中搜索很容易实现，通常都是使用sql语句进行查询，而且能很快的得到查询结果。

为什么数据库搜索很容易？

因为数据库中的数据存储是有规律的，有行有列而且数据格式、数据长度都是固定的。

## 1.3非结构化数据查询方法？

1. **顺序扫描法(Serial Scanning)**

所谓顺序扫描，比如要找内容包含某一个字符串的文件，就是一个文档一个文档的看，对于每一个文档，从头看到尾，如果此文档包含此字符串，则此文档为我们要找的文件，接着看下一个文件，直到扫描完所有的文件。如利用windows的搜索也可以搜索文件内容，只是相当的慢。

1. **全文检索(Full-text Search)**

将非结构化数据中的一部分信息提取出来，重新组织，使其变得有一定结构，然后对此有一定结构的数据进行搜索，从而达到搜索相对较快的目的。这部分从非结构化数据中提取出的然后重新组织的信息，我们称之**索引**。

例如：字典。字典的拼音表和部首检字表就相当于字典的索引，对每一个字的解释是非结构化的，如果字典没有音节表和部首检字表，在茫茫辞海中找一个字只能顺序扫描。然而字的某些信息可以提取出来进行结构化处理，比如读音，就比较结构化，分声母和韵母，分别只有几种可以一一列举，于是将读音拿出来按一定的顺序排列，每一项读音都指向此字的详细解释的页数。我们搜索时按结构化的拼音搜到读音，然后按其指向的页数，便可找到我们的非结构化数据——也即对字的解释。

**这种先建立索引，再对索引进行搜索的过程就叫全文检索(Full-text Search)。**

虽然创建索引的过程也是非常耗时的，但是索引一旦创建就可以多次使用，全文检索主要处理的是查询，所以耗时间创建索引是值得的。

## 1.4如何实现全文检索？

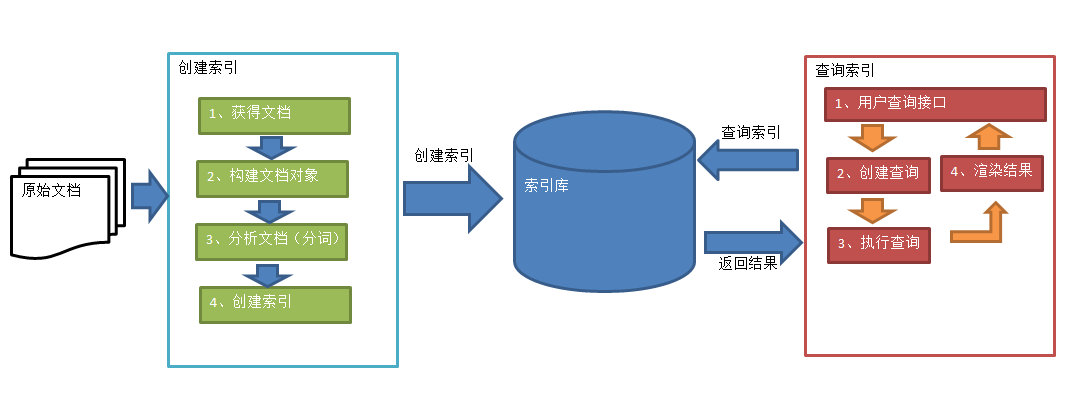
可以使用Lucene实现全文检索。Lucene是apache下的一个开放源代码的全文检索引擎工具包。提供了完整的查询引擎和索引引擎，部分文本分析引擎。Lucene的目的是为软件开发人员提供一个简单易用的工具包，以方便的在目标系统中实现全文检索的功能。

## 1.5全文检索的应用场景？

对于数据量大、数据结构不固定的数据可采用全文检索方式搜索，比如百度、Google等搜索引擎、论坛站内搜索、电商网站站内搜索等。

# 2 Lucene实现全文检索的流程

## 2.1索引和搜索流程图？



1、绿色表示索引过程，对要搜索的原始内容进行索引构建一个索引库，索引过程包括：

确定原始内容即要搜索的内容🡪采集文档🡪创建文档🡪分析文档🡪索引文档

2、红色表示搜索过程，从索引库中搜索内容，搜索过程包括：

用户通过搜索界面🡪创建查询🡪执行搜索，从索引库搜索🡪渲染搜索结果

## 2.2创建索引

对文档索引的过程，将用户要搜索的文档内容进行索引，索引存储在索引库（index）中。

这里我们要搜索的文档是磁盘上的文本文件，根据案例描述：凡是文件名或文件内容包括关键字的文件都要找出来，这里要对文件名和文件内容创建索引。

### 2.2.1获得原始文档

**原始文档**是指要索引和搜索的内容。原始内容包括互联网上的网页、数据库中的数据、磁盘上的文件等。

本案例中的原始内容就是磁盘上的文件，如下图：

E:\development\lucene\02.参考资料\searchsource



从互联网上、数据库、文件系统中等获取需要搜索的原始信息，这个过程就是信息采集，信息采集的目的是为了对原始内容进行索引。

在Internet上采集信息的软件通常称为爬虫或蜘蛛，也称为网络机器人，爬虫访问互联网上的每一个网页，将获取到的网页内容存储起来。

本案例我们要获取磁盘上文件的内容，可以通过文件流来读取文本文件的内容，对于pdf、doc、xls等文件可通过第三方提供的解析工具读取文件内容，比如Apache POI读取doc和xls的文件内容。

### 2.2.2创建文档对象

获取原始内容的目的是为了索引，在索引前需要将原始内容创建成文档（Document），文档中包括一个一个的域（Field），域中存储内容。

这里我们可以将磁盘上的一个文件当成一个document，Document中包括一些Field（file\_name文件名称、file\_path文件路径、file\_size文件大小、file\_content文件内容），如下图：

Document（文档）

Field（域）

Name：file\_name（文件名称）

Value：springmvc.txt

Field（域）

Name：file\_path（文件路径）

Value：e:/…/XXXXXX.txt

Field（域）

Name：file\_content（文件内容）

Value：36312

Field（域）

Name：file\_size（文件大小）

Value：346543

其它Field.。。。。。。。。。。。。

### 2.2.3分析文档

将原始内容创建为包含域（Field）的文档（document），需要再对域中的内容进行分析，分析的过程是经过对原始文档提取单词、将字母转为小写、去除标点符号、去除停用词等过程生成最终的语汇单元，可以将语汇单元理解为一个一个的单词。

比如下边的文档经过分析如下：

原文档内容：

Lucene is a Java full-text search engine. Lucene is not a complete

application, but rather a code library and API that can easily be used

to add search capabilities to applications.

分析后得到的语汇单元：

lucene、java、full、search、engine。。。。

每个单词叫做一个Term（对象，包含两部分），不同的域中拆分出来的相同的单词是不同的term。term中包含两部分一部分是文档的域名，另一部分是单词的内容。

例如：文件名中包含apache和文件内容中包含的apache是不同的term。

### 2.2.4创建索引

对所有文档分析得出的语汇单元进行索引，索引的目的是为了搜索，最终要实现只搜索被索引的语汇单元从而找到Document（文档）。

索引库

索引库

|  |
| --- |
| Filename:spring |
| Content:framework |
| Content:provides |
| Content:comprehensive |
| Content:rogramming |
| Content:configuration |
| Filename:Model |
| Content:spring |

Document1

Document2

Document3

Document4

注意：创建索引是对语汇单元索引，通过词语找文档，这种索引的结构叫**倒排索引结构**。

传统方法是根据文件找到该文件的内容，在文件内容中匹配搜索关键字，这种方法是顺序扫描方法，数据量大、搜索慢。

**倒排索引结构**是根据内容（词语）找文档，如下图：



**倒排索引结构也叫反向索引结构，包括索引和文档两部分，索引即词汇表，它的规模较小，而文档集合较大。**

## 2.3 查询索引

查询索引也是搜索的过程。搜索就是用户输入关键字，从索引（index）中进行搜索的过程。根据关键字搜索索引，根据索引找到对应的文档，从而找到要搜索的内容（这里指磁盘上的文件）。

### 2.3.1用户查询接口

(全文检索系统提供用户搜索的界面供用户提交搜索的关键字，搜索完成展示搜索结果。

比如：



Lucene不提供制作用户搜索界面的功能，需要根据自己的需求开发搜索界面。

### 2**.3.2**创建查询

(用户输入查询关键字执行搜索之前需要先构建一个查询对象，查询对象中可以指定查询要搜索的Field文档域、查询关键字等，查询对象会生成具体的查询语法，

例如：

语法 “fileName:lucene”表示要搜索Field域的内容为“lucene”的文档

### 2**.3.3**执行查询

搜索索引过程：

根据查询语法在倒排索引词典表中分别找出对应搜索词的索引，从而找到索引所链接的文档链表。

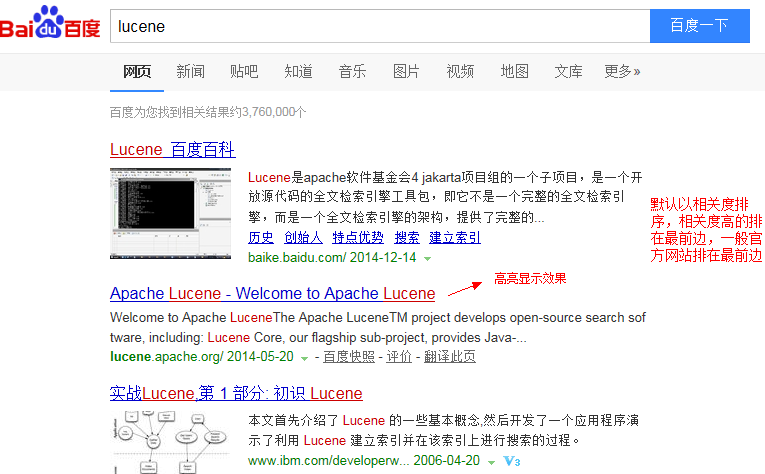
比如搜索语法为“fileName:lucene”表示搜索出fileName域中包含Lucene的文档。

搜索过程就是在索引上查找域为fileName，并且关键字为Lucene的term，并根据term找到文档id列表。



### 2**.3.4**渲染结果

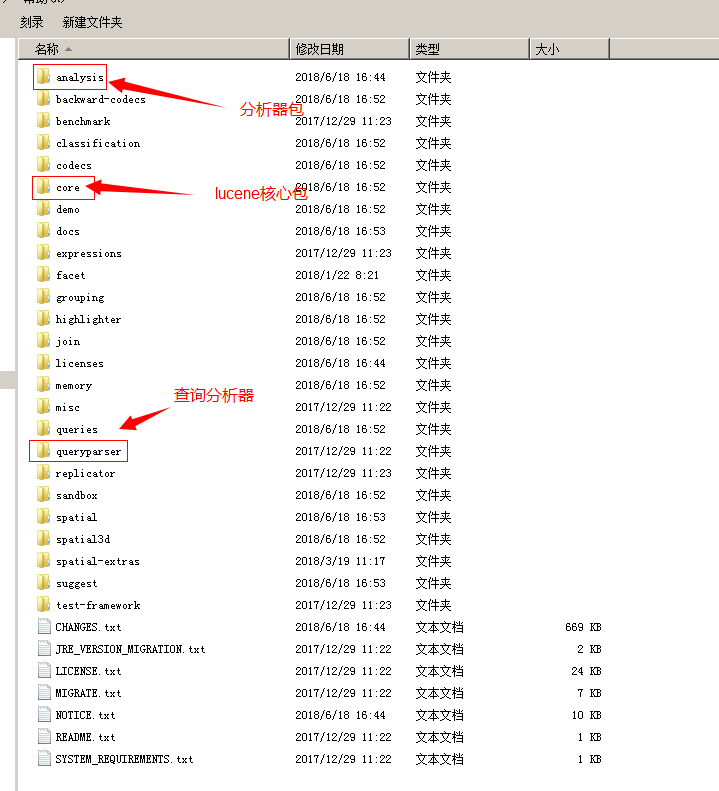
以一个友好的界面将查询结果展示给用户，用户根据搜索结果找自己想要的信息，为了帮助用户很快找到自己的结果，提供了很多展示的效果，比如搜索结果中将关键字高亮显示，百度提供的快照等。



# 3 配置开发环境

## 3.1 下载

Lucene是开发全文检索功能的工具包，从官方网站下载lucene-7.4.0，并解压。



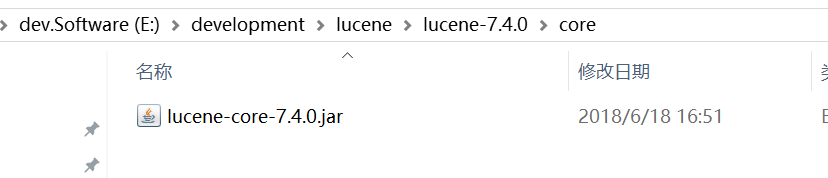
官方网站：<http://lucene.apache.org/>

版本：lucene-7.4.0

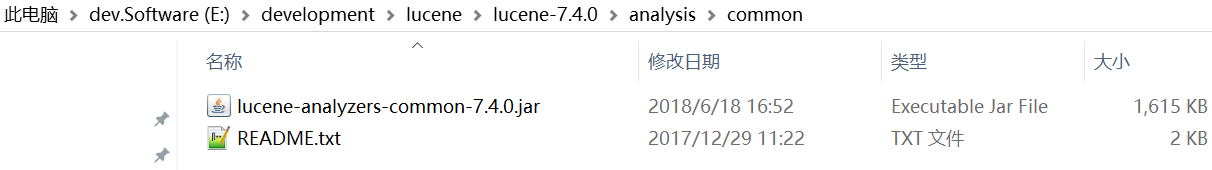
Jdk要求：1.8以上

## 3.2 使用的jar包

lucene-core-7.4.0.jar



lucene-analyzers-common-7.4.0.jar



commons-io.jar

# 4 入门程序

## 4.1 需求

实现一个文件的搜索功能，通过关键字搜索文件，凡是文件名或文件内容包括关键字的文件都需要找出来。还可以根据中文词语进行查询，并且需要支持多个条件查询。

本案例中的原始内容就是磁盘上的文件，如下图：



## 4.2 创建索引

### 4.2.1实现步骤

1、创建一个Director对象， 指定索引库保存的位置。

2、基于Directory对象创建一个 IndexWriter对象

3、读取磁盘上的文件，对应每个文件创建一个文档对象。

4、向文档对象中添加域

5、把文档对象写入索引库

6、关闭indexwriter对象

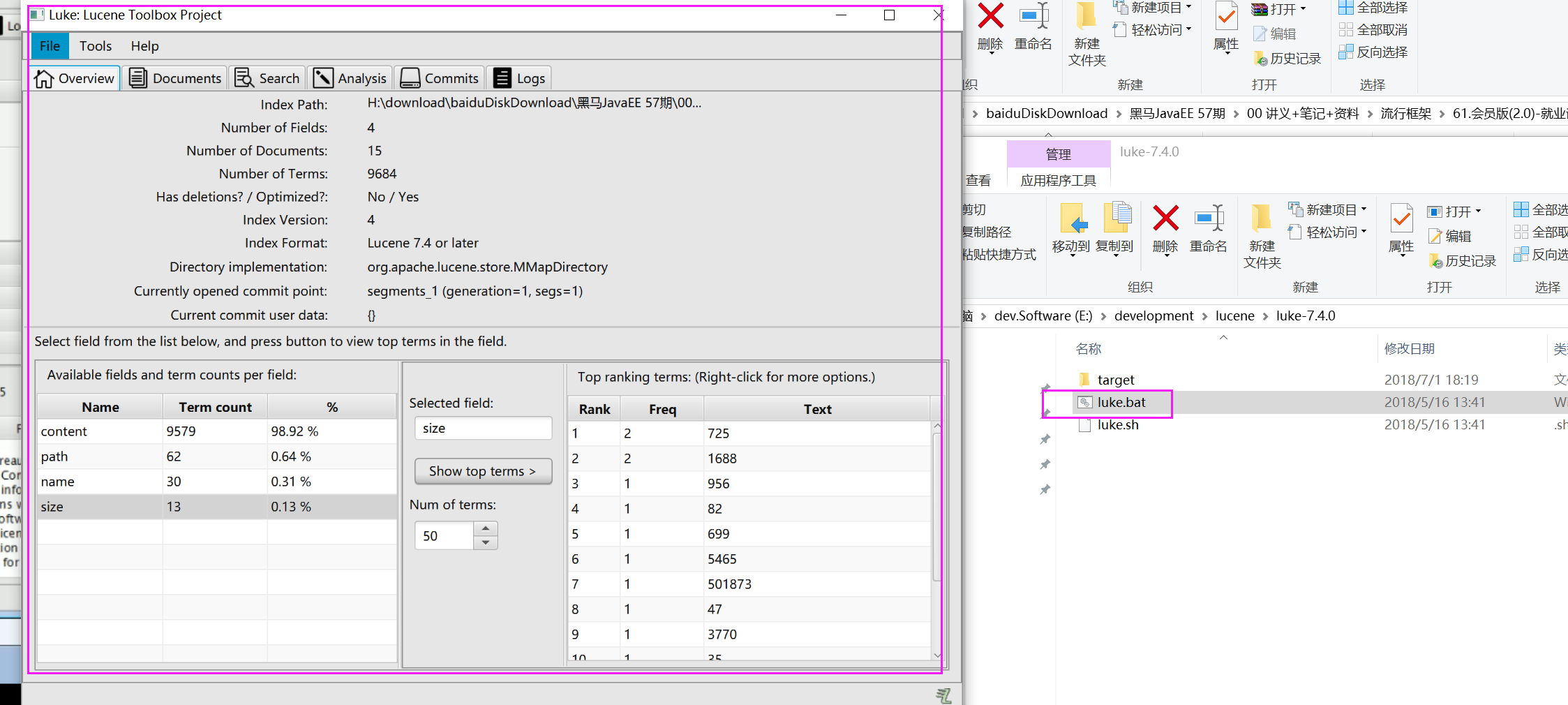
### 4.2.2代码实现

public class LuceneFirst {  
  
 @Test  
 public void createIndex() throws Exception{  
 *//1、创建一个Director对象， 指定索引库保存的位置。* Directory directory = new RAMDirectory();*//保存在内存  
 //保存在磁盘* Directory open = FSDirectory.*open*(new File("H:\\download\\baiduDiskDownload\\黑马JavaEE 57期\\00 讲义+笔记+资料\\流行框架\\61.会员版(2.0)-就业课(2.0)-Lucene\\index").toPath());  
 *//2、基于Directory对象创建一个 IndexWriter对象* IndexWriter indexWriter = new IndexWriter(open, new IndexWriterConfig());  
 *//3、读取磁盘上的文件，对应每个文件创建一个文档对象。* File dir = new File("H:\\download\\baiduDiskDownload\\黑马JavaEE 57期\\00 讲义+笔记+资料\\流行框架\\61.会员版(2.0)-就业课(2.0)-Lucene\\lucene\\02.参考资料\\searchsource");  
 File[] files = dir.listFiles();  
 for (File f : files) {  
 *//文件名、文件路径、文件内容、文件大小* String fileName = f.getName();  
 String filePath = f.getPath();  
 String fileContent = FileUtils.*readFileToString*(f, "UTF-8");  
 long fileSize = FileUtils.*sizeOf*(f);  
  
 *//创建Field : 参数1是域的名称，参数2是域的内容，参数3表示是否存储* Field fieldName = new TextField("name", fileName, Field.Store.*YES*);  
 Field fieldPath = new TextField("path", filePath, Field.Store.*YES*);  
 Field fieldContent = new TextField("content", fileContent, Field.Store.*YES*);  
 Field fieldSize = new TextField("size", fileSize+"", Field.Store.*YES*);  
  
 *//创建文档对象  
 //4、向文档对象中添加域* Document document = new Document();  
 document.add(fieldName);  
 document.add(fieldPath);  
 document.add(fieldContent);  
 document.add(fieldSize);  
  
 *//5、把文档对象写入索引库* indexWriter.addDocument(document);  
 }  
 *// 6、关闭indexwriter对象* indexWriter.close();  
 }  
}

**执行性后：**



### 4.2.3使用Luke工具查看索引文件



## 4.3 查询索引

### 4.3.1实现步骤

1、创建一个Director对象， 指定索引库的位置

2、创建一一个IndexReader对象

3、创建一个IndexSearcher对象，构造方法中的参数indexReader对象

4、创建一个Query对象，TermQuery

5、执行查询，得到一个TopDocs对象

6、取查询结果的总记录数

7、取文档列表

8、打印文档中的内容

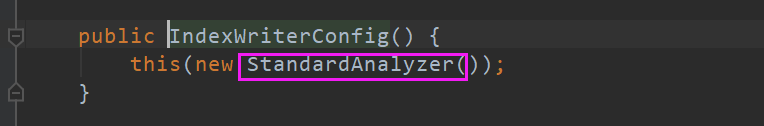
9、关闭IndexReader对象

### 4.3.2代码实现

public void searchIndex() throws Exception{  
 *//1、创建一个Director对象， 指定索引库的位置* Directory open = FSDirectory.*open*(new File("H:\\download\\baiduDiskDownload\\黑马JavaEE 57期\\00 讲义+笔记+资料\\流行框架\\61.会员版(2.0)-就业课(2.0)-Lucene\\index").toPath());  
 *//2、创建一一个IndexReader对象* IndexReader indexReader = DirectoryReader.*open*(open);  
 *//3、创建一个IndexSearcher对象，构造方法中的参数indexReader对象* IndexSearcher indexSearcher = new IndexSearcher(indexReader);  
 *//4、创建一个Query对象，TermQuery* Query query = new TermQuery(new Term("content","spring"));  
 *//5、执行查询，得到一个TopDocs对象 参数1：查询对象， 参数2：返回的最大记录数* TopDocs topDocs = indexSearcher.search(query, 10);  
 *//6、取查询结果的总记录数* long totalHits = topDocs.totalHits;  
 System.*out*.println("查询到的总记录数："+totalHits);  
 *//7、取文档列表* ScoreDoc[] scoreDocs = topDocs.scoreDocs;  
 *//8、打印文档中的内容* for (ScoreDoc scoreDoc : scoreDocs) {  
 *//取文档id* int docId = scoreDoc.doc;  
 *//根据文档id获取文档* Document document = indexSearcher.doc(docId);  
 System.*out*.println(document.get("name"));  
 System.*out*.println(document.get("path"));  
 System.*out*.println(document.get("size"));  
*// System.out.println(document.get("content"));* System.*out*.println("-----------------------寂寞的分隔说---------------------");  
 }  
 *//9、关闭IndexReader对象* indexReader.close();  
 }

# 5 分析器

默认使用标准分析器



## 5.1 分析器分词效果

使用Analyzer对象的tokenStream方法返回一-个TokenStream对象。词对象中包含了最终分词结果。

实现步骤:

1)创建一个Analyzer对象，StandardAnalyzer对 象

2)使用分析器对象的tokenStream方法获得一个TokenStream对 象

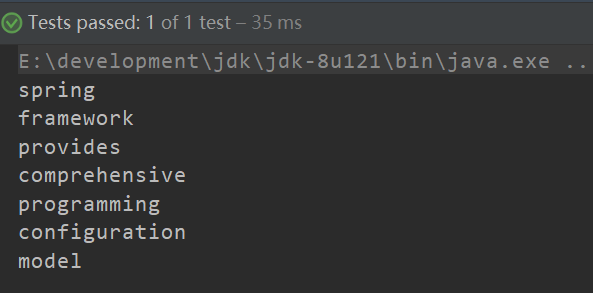
3)向TokenStream对象中设置- 一个引用，相当于数- -个指针

4)调用TokenStream对象的rest方法。如果不调用抛异常

5)使用while循环遄历TokenStream对象

6)关闭TokenStream对象

@Test  
public void testTokenStream() throws Exception{  
 *//1)创建一个Analyzer对象，StandardAnalyzer对象* Analyzer analyzer = new StandardAnalyzer();  
 *//2)使用分析器对象的tokenStream方法获得一个TokenStream对象 参数1：域名称，要分析的文本* TokenStream tokenStream = analyzer.tokenStream("", "The Spring Framework provides a comprehensive programming and configuration model.");  
 *//3)向TokenStream对象中设置一个引用，相当于数一个指针* CharTermAttribute charTermAttribute = tokenStream.addAttribute(CharTermAttribute.class);  
 *//4)调用TokenStream对象的rest方法。如果不调用抛异常* tokenStream.reset();  
 *//5)使用while循环遄历TokenStream对象* while (tokenStream.incrementToken()){  
 System.*out*.println(charTermAttribute.toString());  
 }  
 *//6)关闭TokenStream对象* tokenStream.close();  
}



分析中文的话，会把每一个字分出来，没有组合的；只能一个字一个字的查

## 5.2 中文分析器

### 5.2.1 Lucene自带中文分词器

### 5.2.2 IKAnalyzer

IKAnalyzer是一个开源的，基于java语言开发的轻量级的中文分词工具包。从2006年12月推出1.0版开始，IKAnalyzer 已经推出了3个大版本。

linliangyi2007 (林良益) 是一位资深的Java开发者和优秀开源开发者, JavaEye非常荣幸的采访了他。

使用方法：

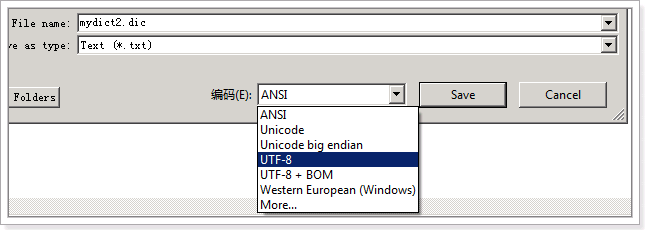
第一步：把jar包添加到工程中

第二步：把配置文件和扩展词典和停用词词典添加到classpath下

注意：hotword.dic和ext\_stopword.dic文件的格式为UTF-8，注意是无BOM 的UTF-8 编码。

也就是说禁止使用windows记事本编辑扩展词典文件

使用EditPlus.exe保存为无BOM 的UTF-8 编码格式，如下图：



把IKAnalyzer的jar包添加到工程中

2)把配置文件和扩展词典添加到工程的classpath'下

注意:扩展词典严禁使用windows记事本编轻保证扩展词典的编码格式是utf-8

扩展词典:添加一些新词

停用词词典:无意义的词或者是敏感词汇

@Test  
 public void testTokenStream() throws Exception{  
 *//1)创建一个Analyzer对象，StandardAnalyzer对象  
// Analyzer analyzer = new StandardAnalyzer();* Analyzer analyzer = new IKAnalyzer();  
 *//2)使用分析器对象的tokenStream方法获得一个TokenStream对象 参数1：域名称，要分析的文本  
// TokenStream tokenStream = analyzer.tokenStream("", "The Spring Framework provides a comprehensive programming and configuration model.");* TokenStream tokenStream = analyzer.tokenStream("", "2012年2月8日 - 每个网站的必须提供的基本功能之一,用Lucene构造一个“索引-查询”的应用是常见的java解决方案,目前由linliangyi2007创立的IK Analyzer是最好的Lu");  
 *//3)向TokenStream对象中设置一个引用，相当于数一个指针* CharTermAttribute charTermAttribute = tokenStream.addAttribute(CharTermAttribute.class);  
 *//4)调用TokenStream对象的rest方法。如果不调用抛异常* tokenStream.reset();  
 *//5)使用while循环遄历TokenStream对象* while (tokenStream.incrementToken()){  
 System.*out*.println(charTermAttribute.toString());  
 }  
 *//6)关闭TokenStream对象* tokenStream.close();  
 }

## 5.3 使用自定义分析器

*// IndexWriter indexWriter = new IndexWriter(open, new IndexWriterConfig());* IndexWriter indexWriter = new IndexWriter(open, new IndexWriterConfig(new IKAnalyzer()));

# 6 索引库的维护

## 6.1 索引库的添加

### 6.1.1 Field域的属性

**是否分析**：是否对域的内容进行分词处理。前提是我们要对域的内容进行查询。

**是否索引**：将Field分析后的词或整个Field值进行索引，只有索引方可搜索到。

比如：商品名称、商品简介分析后进行索引，订单号、身份证号不用分析但也要索引，这些将来都要作为查询条件。

**是否存储**：将Field值存储在文档中，存储在文档中的Field才可以从Document中获取

比如：商品名称、订单号，凡是将来要从Document中获取的Field都要存储。

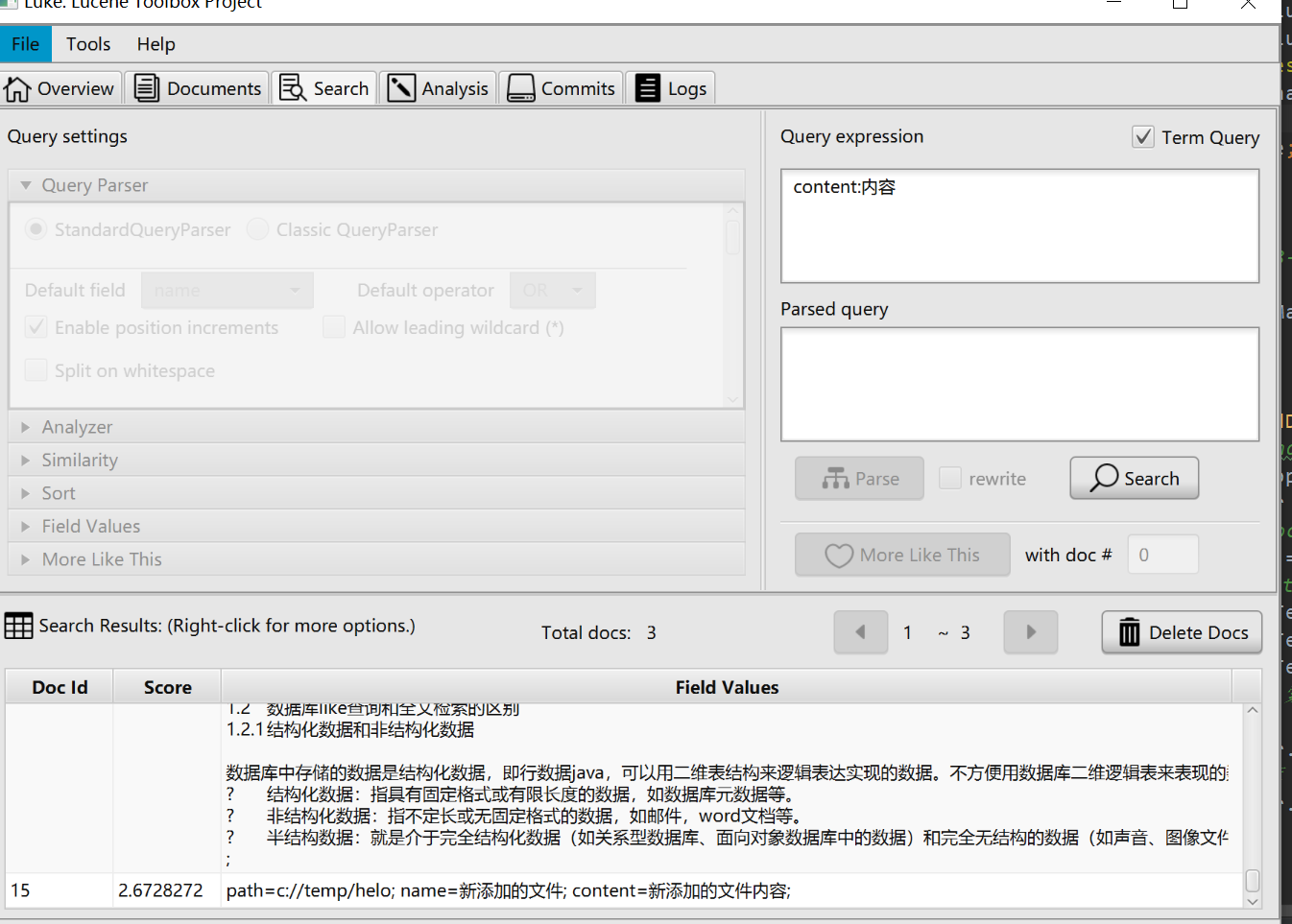
**是否存储的标准：是否要将内容展示给用户**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field类 | 数据类型 | Analyzed  是否分析 | Indexed  是否索引 | Stored  是否存储 | 说明 |
| StringField(FieldName, FieldValue,Store.YES)) | 字符串 | N | Y | Y或N | 这个Field用来构建一个字符串Field，但是不会进行分析，会将整个串存储在索引中，比如(订单号,姓名等)  是否存储在文档中用Store.YES或Store.NO决定 |
| LongPoint(String name, long... point) | Long型 | Y | Y | N | 可以使用LongPoint、IntPoint等类型存储数值类型的数据。让数值类型可以进行索引。但是不能存储数据，如果想存储数据还需要使用StoredField。 |
| StoredField(FieldName, FieldValue) | 重载方法，支持多种类型 | N | N | Y | 这个Field用来构建不同类型Field  不分析，不索引，但要Field存储在文档中 |
| TextField(FieldName, FieldValue, Store.NO)  或  TextField(FieldName, reader) | 字符串  或  流 | Y | Y | Y或N | 如果是一个Reader, lucene猜测内容比较多,会采用Unstored的策略. |

### 6.1.2 添加文档代码实现

*//创建Field : 参数1是域的名称，参数2是域的内容，参数3表示是否存储* Field fieldName = new TextField("name", fileName, Field.Store.*YES*);  
*// Field fieldPath = new TextField("path", filePath, Field.Store.YES);* Field fieldPath = new StoredField("path", filePath);  
 Field fieldContent = new TextField("content", fileContent, Field.Store.*YES*);  
*// Field fieldSize = new TextField("size", fileSize+"", Field.Store.YES);* Field fieldSizeValue = new LongPoint("size", fileSize);*//不存储* Field fieldSizeStore = new StoredField("size", fileSize);*//只存储*

@Test  
public void addDocument() throws Exception{  
 *//创建一个Indexwriter对象，需要使用IKAnalyzer作为分析器* Directory open = FSDirectory.*open*(new File("H:\\download\\baiduDiskDownload\\黑马JavaEE 57期\\00 讲义+笔记+资料\\流行框架\\61.会员版(2.0)-就业课(2.0)-Lucene\\index").toPath());  
 IndexWriter indexWriter = new IndexWriter(open, new IndexWriterConfig(new IKAnalyzer()));  
 *//创建一个Document 对象* Document d = new Document();  
 *//向document对象中添加域* d.add(new TextField("name","新添加的文件", Field.Store.*YES*));  
 d.add(new TextField("content","新添加的文件内容", Field.Store.*YES*));  
 d.add(new TextField("path","c://temp/helo", Field.Store.*YES*));  
 *//把文档写入索引库* indexWriter.addDocument(d);  
 *//关闭索引库* indexWriter.close();  
}



## 6.2 索引库删除

### 6.2.1删除全部

@Test  
public void deletAllDocument() throws Exception{  
 *//创建一个Indexwriter对象，需要使用IKAnalyzer作为分析器* Directory open = FSDirectory.*open*(new File("H:\\download\\baiduDiskDownload\\黑马JavaEE 57期\\00 讲义+笔记+资料\\流行框架\\61.会员版(2.0)-就业课(2.0)-Lucene\\index").toPath());  
 IndexWriter indexWriter = new IndexWriter(open, new IndexWriterConfig(new IKAnalyzer()));  
   
 indexWriter.deleteAll();  
 *//关闭索引库* indexWriter.close();  
}

### 6.2.2指定查询条件删除

@Test  
public void deletDocumentByQuery() throws Exception{  
 *//创建一个Indexwriter对象，需要使用IKAnalyzer作为分析器* Directory open = FSDirectory.*open*(new File("H:\\download\\baiduDiskDownload\\黑马JavaEE 57期\\00 讲义+笔记+资料\\流行框架\\61.会员版(2.0)-就业课(2.0)-Lucene\\index").toPath());  
 IndexWriter indexWriter = new IndexWriter(open, new IndexWriterConfig(new IKAnalyzer()));  
  
 indexWriter.deleteDocuments(new Term("name","apache"));  
 *//关闭索引库* indexWriter.close();  
}

## 6.3 索引库的修改

修改的原理：先删除再添加

@Test  
public void updateDocument() throws Exception{  
 *//创建一个Indexwriter对象，需要使用IKAnalyzer作为分析器* Directory open = FSDirectory.*open*(new File("H:\\download\\baiduDiskDownload\\黑马JavaEE 57期\\00 讲义+笔记+资料\\流行框架\\61.会员版(2.0)-就业课(2.0)-Lucene\\index").toPath());  
 IndexWriter indexWriter = new IndexWriter(open, new IndexWriterConfig(new IKAnalyzer()));  
  
 *//创建一个Document 对象* Document d = new Document();  
 *//向document对象中添加域* d.add(new TextField("name","新更新的文件", Field.Store.*YES*));  
 d.add(new TextField("content","新更新的文件内容", Field.Store.*YES*));  
 d.add(new TextField("path","c://temp/helo", Field.Store.*YES*));  
  
 indexWriter.updateDocument(new Term("name","spring"),d);  
 *//关闭索引库* indexWriter.close();  
}

# 7 Lucene 索引库查询

## 7.1 TermQuery

参照6.。。

## 7.2RangeQuery数值范围查询

public class SearchIndex {  
  
 private IndexReader indexReader;  
 private IndexSearcher indexSearcher;  
  
 @Before  
 public void init() throws Exception{  
 Directory open = FSDirectory.*open*(new File("H:\\download\\baiduDiskDownload\\黑马JavaEE 57期\\00 讲义+笔记+资料\\流行框架\\61.会员版(2.0)-就业课(2.0)-Lucene\\index").toPath());  
 *//2、创建一一个IndexReader对象* indexReader = DirectoryReader.*open*(open);  
 indexSearcher = new IndexSearcher(indexReader);  
 }  
  
  
 @Test  
 public void testRangeQuery() throws Exception{  
 Query query = LongPoint.*newRangeQuery*("size", 0L, 100L);  
 this.printResult(query);  
 }  
  
  
  
 private void printResult(Query query)throws Exception{  
 TopDocs topDocs = indexSearcher.search(query, 10);  
 System.*out*.println("总记录数哦："+topDocs.totalHits);  
 ScoreDoc[] scoreDocs = topDocs.scoreDocs;  
 for (ScoreDoc scoreDoc : scoreDocs) {  
 *//取文档id* int docId = scoreDoc.doc;  
 *//根据文档id获取文档* Document document = indexSearcher.doc(docId);  
 System.*out*.println(document.get("name"));  
 System.*out*.println(document.get("path"));  
 System.*out*.println(document.get("size"));  
*// System.out.println(document.get("content"));* System.*out*.println("-----------------------寂寞的分隔说---------------------");  
 }  
 indexReader.close();  
 }  
}

## 7.3 使用queryparse查询

可以对要查询的内容先分词。然后基于分词的结果进行查询。

添加jar包

lucene-queryparser-7.4.0.jar

private void printResult(Query query)throws Exception{  
 TopDocs topDocs = indexSearcher.search(query, 10);  
 System.*out*.println("总记录数哦："+topDocs.totalHits);  
 ScoreDoc[] scoreDocs = topDocs.scoreDocs;  
 for (ScoreDoc scoreDoc : scoreDocs) {  
 *//取文档id* int docId = scoreDoc.doc;  
 *//根据文档id获取文档* Document document = indexSearcher.doc(docId);  
 System.*out*.println(document.get("name"));  
 System.*out*.println(document.get("path"));  
 System.*out*.println(document.get("size"));  
*// System.out.println(document.get("content"));* System.*out*.println("-----------------------寂寞的分隔说---------------------");  
 }  
 indexReader.close();  
 }  
  
  
 @Test  
 public void testQuerParser() throws Exception{  
 *//1)创建一个QuerParser对象，参数1：默认搜索域，参数2：分析器对象* QueryParser queryParser = new QueryParser("name",new IKAnalyzer());  
 Query query = queryParser.parse("lucene是一个Java开发的全文检索工具包\n");  
 printResult(query);  
 }