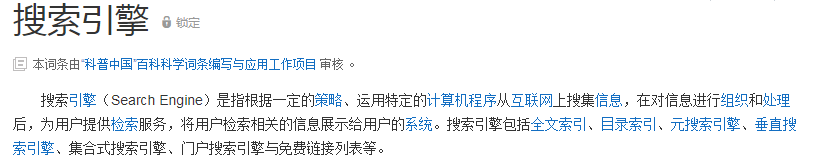
Luence课程教案

学习目标：

* 了解搜索引擎技术
* 掌握Lucene和Solr的关系
* 掌握Lucene索引的创建
* 掌握Lucene索引的查询
* 掌握Lucene中文分词器的使用
* 掌握Lucene索引的更新
* 掌握Lucene索引的删除
* 掌握Lucene关键词高亮
* 掌握Lucene分页
* 掌握Lucene加权得分处理
* 掌握Lucene排序操作

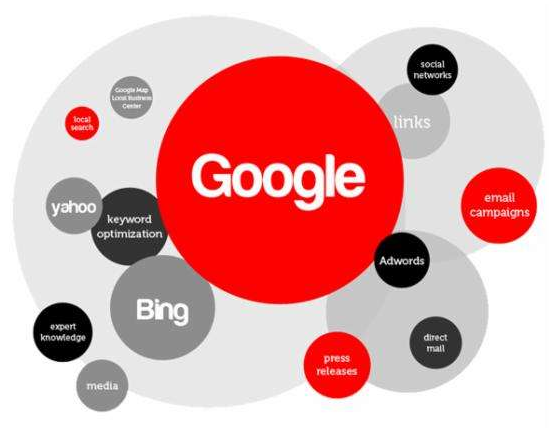
# 了解搜索技术

## 搜索引擎



一个搜索引擎由搜索器 、索引器 、检索器 和用户接口 四个部分组成。搜索器的功能是在互联网 中漫游，发现和搜集信息。索引器的功能是理解搜索器所搜索的信息，从中抽取出索引项，用于表示文档 以及生成文档库的索引表。检索器的功能是根据用户的查询在索引库中快速检出文档，进行文档与查询的相关度评价，对将要输出的结果进行排序，并实现某种用户相关性反馈机制。用户接口的作用是输入用户查询、显示查询结果、提供用户相关性反馈机制。

常见的搜索引擎有：

## 搜索引擎发展史

Google：1998年10月之前，Google只是美国斯坦福大学的一个小项目。1995年博士生Larry Page开始学习搜索引擎设计，于1997年9月15日注册了google.com 的域名。

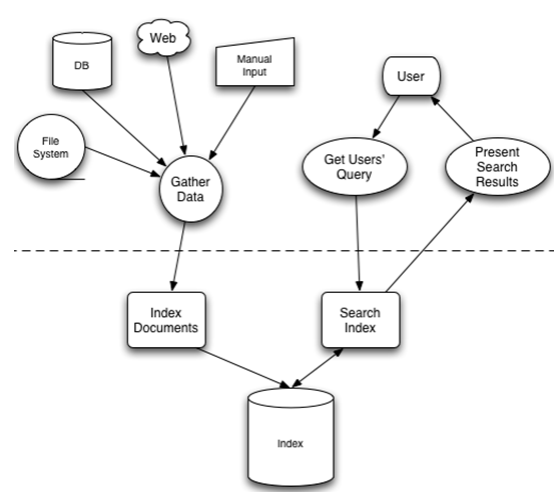
百度：2000年1月，两位北大校友，李彦宏与好友徐勇在北京中关村创立了百度公司。

雅虎：1994年4月，斯坦福（Stanford）大学的两名博士生，美籍华人杨致远和美国人David Filo共同创办了超级目录索引（Yahoo），并成功地使搜索引擎的概念深入人心。从此搜索引擎进入了高速发展时期。

新浪：1998年12月1日，四通利方信息技术有限公司和华渊资讯公司宣布合并，成立新浪网公司并推出同名的中文网站。其搜索引擎技术的合作对象是百度公司。

搜狐：1998年2月，爱特信公司创办了“搜狐”大型中文网络系统。搜狐站点的内容大量采用了人工选择和分类，并提供“分类查询”和“关键词”两种方式检索。其搜索引擎技术的合作对象是百度公司。

## 搜索引擎的原理



### 爬行

搜索引擎是通过一种特定规律的软件跟踪网页的链接，从一个链接爬到另外一个链接，像蜘蛛在蜘蛛网上爬行一样，所以被称为“蜘蛛”也被称为“机器人”。搜索引擎蜘蛛的爬行是被输入了一定的规则的，它需要遵从一些命令或文件的内容。

### 抓取存储

搜索引擎是通过蜘蛛跟踪链接爬行到网页，并将爬行的数据存入原始页面数据库。其中的页面数据与用户浏览器得到的HTML是完全一样的。搜索引擎蜘蛛在抓取页面时，也做一定的重复内容检测，一旦遇到权重很低的网站上有大量抄袭、采集或者复制的内容，很可能就不再爬行。

### 预处理

搜索引擎将蜘蛛抓取回来的页面，进行各种步骤的预处理，包括：提取文字，中文分词，去停止词，消除噪音（搜索引擎需要识别并消除这些噪声，比如版权声明文字、导航条、广告等……），正向索引，倒排索引，链接关系计算，特殊文件处理。

除了HTML 文件外，搜索引擎通常还能抓取和索引以文字为基础的多种文件类型，如 PDF、Word、WPS、XLS、PPT、TXT 文件等。我们在搜索结果中也经常会看到这些文件类型。 但搜索引擎还不能处理图片、视频、Flash 这类非文字内容，也不能执行脚本和程序。

### 排名

用户在搜索框输入关键词后，排名程序调用索引库数据，计算排名显示给用户，排名过程与用户直接互动的。但是，由于搜索引擎的数据量庞大，虽然能达到每日都有小的更新，但是一般情况搜索引擎的排名规则都是根据日、周、月阶段性不同幅度的更新。

## 搜索技术的应用场景

搜索引擎广泛应用在大型综合搜索网站，如百度，谷歌等，也应用于系统的站内搜索。

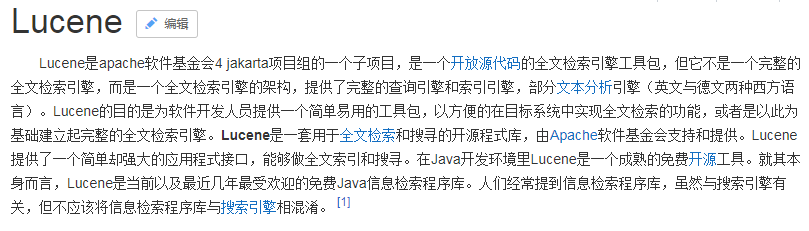
## 实现搜索技术的方式

实现搜索技术可以从数据库本身出发，使用数据库的搜索命令来进行搜索，但是这种搜索的方式往往面临在数据量很大的情况下，模糊搜索不一定走索引，因此效率就会很低。

初次之外就是使用Lucene搜索技术，解决在海量数据的情况下，利用倒排索引技术，实现快速的搜索、打分、排序等功能。

# Lucene概述

## 2.1 什么是Lucene



作为一个开放源代码项目，Lucene从问世之后，引发了开放源代码社群的巨大反响，程序员们不仅使用它构建具体的全文检索应用，而且将之集成到各种系统软件中去，以及构建Web应用，甚至某些商业软件也采用了Lucene作为其内部全文检索子系统的核心。apache软件基金会的网站使用了Lucene作为全文检索的引擎，IBM的开源软件eclipse的2.1版本中也采用了Lucene作为帮助子系统的全文索引引擎，相应的IBM的商业软件Web Sphere中也采用了Lucene。Lucene以其开放源代码的特性、优异的索引结构、良好的系统架构获得了越来越多的应用。

Lucene是一个高性能、可伸缩的信息搜索(IR)库。它可以为你的应用程序添加索引和搜索能力。Lucene是用java实现的、成熟的开源项目，是著名的Apache Jakarta大家庭的一员，并且基于Apache软件许可 [ASF, License]。同样，Lucene是当前非常流行的、免费的Java信息搜索(IR)库。

## 2.2 全文检索和倒排索引

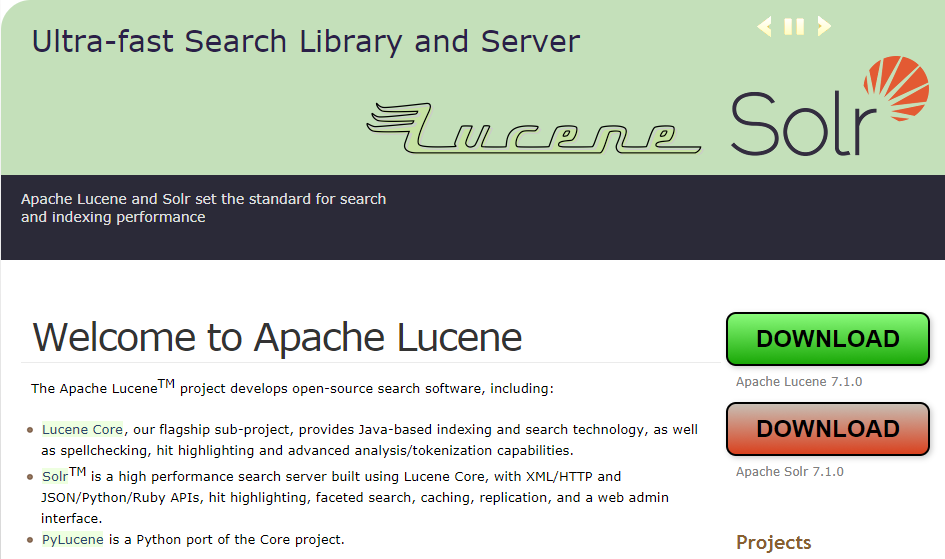
计算机程序一个文档一个文档的扫描，对于每一个文档，从头看到尾，对每一词建立一个索引，指明该词在文章中出现的次数和位置，当用户查找数据时，索引程序就根据事先建立的索引进行查找，并将查找结果反馈给用户的检索方式。

## 2.3 Lucene与Solr的关系

Lucene是一套实现了全文检索的底层API，提供对于全文检索的基础支持，而Solr是全文检索引擎的一个实现产品，是一个企业级搜索应用服务器。

## 2.4 Lucene的下载

Lucene是Apache旗下的顶级项目，我们可以直接访问其官网进行下载和使用。



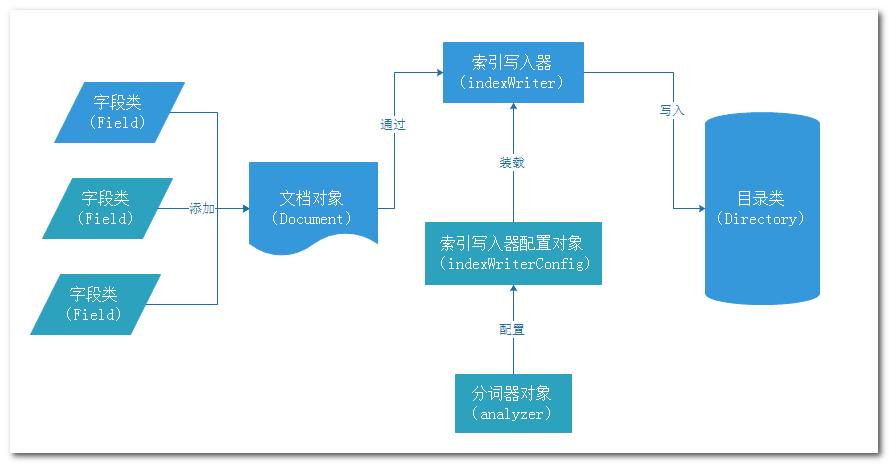
 目前最新的版本是7.x系列，但是大多数企业中依旧使用4.x版本，比较稳定。本次课程我们使用4.10.2版本。

# Lucene的基本使用

使用Lucene的API来实现对索引的增（创建索引）、删（删除索引）、改（修改索引）、查（搜索数据）。

## 3.1 创建索引

### 3.1.1 创建索引的流程



### 3.1.2 添加依赖

使用Lucene需要添加Lucene的依赖。

|  |
| --- |
| lucene核心库 lucene-core  查询解析器 lucene-queryparser  默认分词器 lucene-analyzers-common  IK分词器 ikanalyzer  高亮显示 lucene-highlighter |

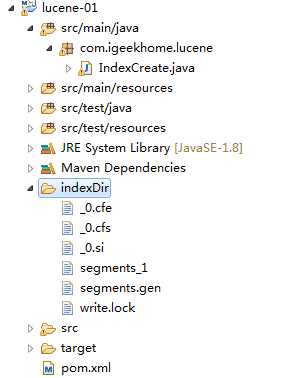
Maven工程中的依赖添加如下：

|  |
| --- |
| <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>  <groupId>com.igeek.lucene</groupId>  <artifactId>lucene-01</artifactId>  <version>0.0.1-SNAPSHOT</version>  <dependencies>  <!-- Junit单元测试 -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.12</version>  </dependency>  <!-- lucene核心库 -->  <dependency>  <groupId>org.apache.lucene</groupId>  <artifactId>lucene-core</artifactId>  <version>4.10.2</version>  </dependency>  <!-- Lucene的查询解析器 -->  <dependency>  <groupId>org.apache.lucene</groupId>  <artifactId>lucene-queryparser</artifactId>  <version>4.10.2</version>  </dependency>  <!-- lucene的默认分词器库 -->  <dependency>  <groupId>org.apache.lucene</groupId>  <artifactId>lucene-analyzers-common</artifactId>  <version>4.10.2</version>  </dependency>  <!-- lucene的高亮显示 -->  <dependency>  <groupId>org.apache.lucene</groupId>  <artifactId>lucene-highlighter</artifactId>  <version>4.10.2</version>  </dependency>  <!-- IK分词器 -->  <dependency>  <groupId>com.janeluo</groupId>  <artifactId>ikanalyzer</artifactId>  <version>2012\_u6</version>  </dependency>  </dependencies>  </project> |

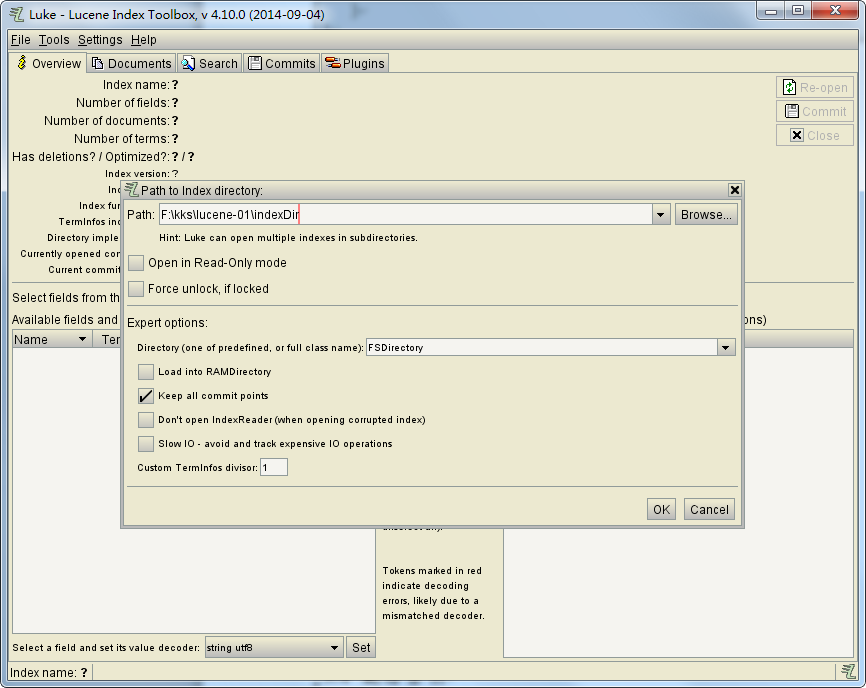
### 3.1.3 代码实现

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.analysis.Analyzer;  **import** org.apache.lucene.analysis.standard.StandardAnalyzer;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.document.Field.Store;  **import** org.apache.lucene.document.StringField;  **import** org.apache.lucene.document.TextField;  **import** org.apache.lucene.index.IndexWriter;  **import** org.apache.lucene.index.IndexWriterConfig;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.apache.lucene.util.Version;  **public** **class** IndexCreate {  **public** **static** **void** main(String[] args) **throws** Exception {  // 创建文档对象  Document document = **new** Document();  // 创建并添加字段信息  document.add(**new** StringField("id", "1", Store.***YES***));  // 添加字段  document.add(**new** TextField("title", "中国工博会上演“人工智能总动员”", Store.***YES***));  // 创建索引目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 创建分词器对象  Analyzer analyzer = **new** StandardAnalyzer();  // 创建配置对象  IndexWriterConfig conf = **new** IndexWriterConfig(Version.***LATEST***, analyzer);  // 创建索引的写出工具类  IndexWriter indexWriter = **new** IndexWriter(directory, conf);  // 添加文档  indexWriter.addDocument(document);  // 提交  indexWriter.commit();  // 关闭  indexWriter.close();  }  } |

程序执行后在工程中生产索引文件，如下图。

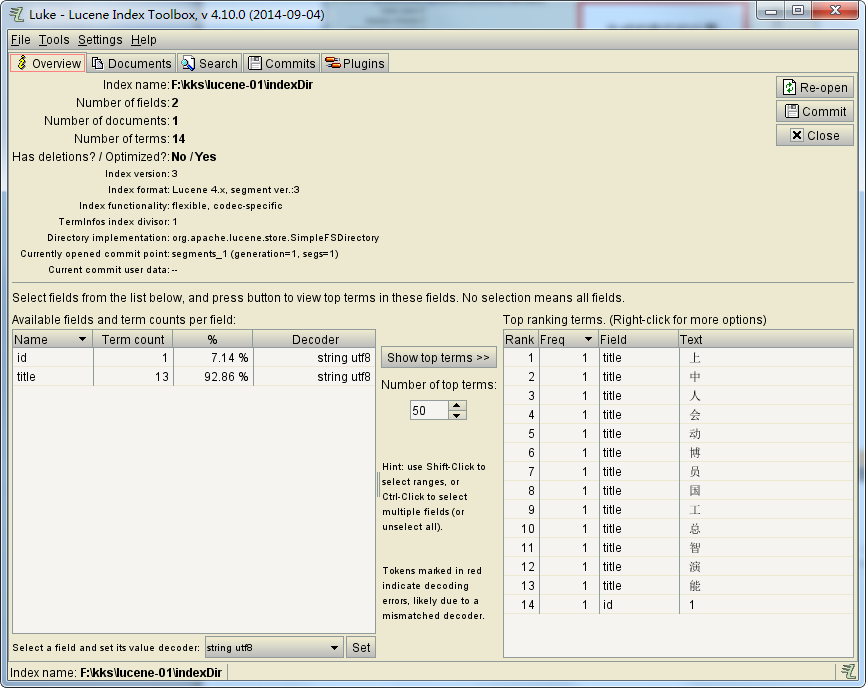


所以创建成功之后，可以使用工具来查看已经创建的索引。



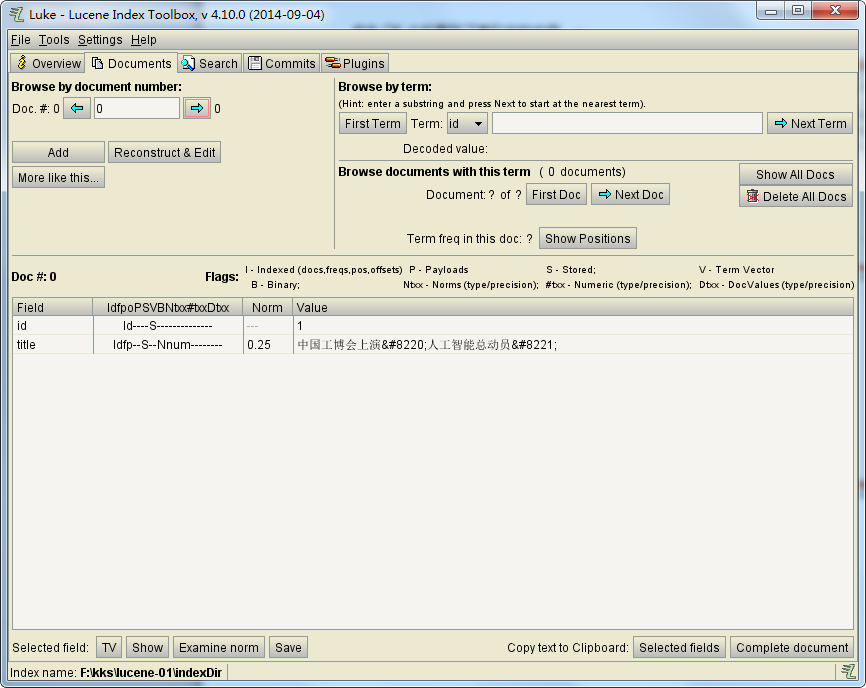
生成的索引的位置

点击OK之后看到了索引中的内容。



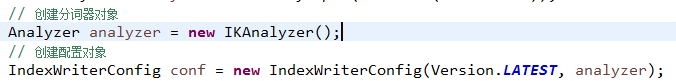
分词的信息

字段的信息

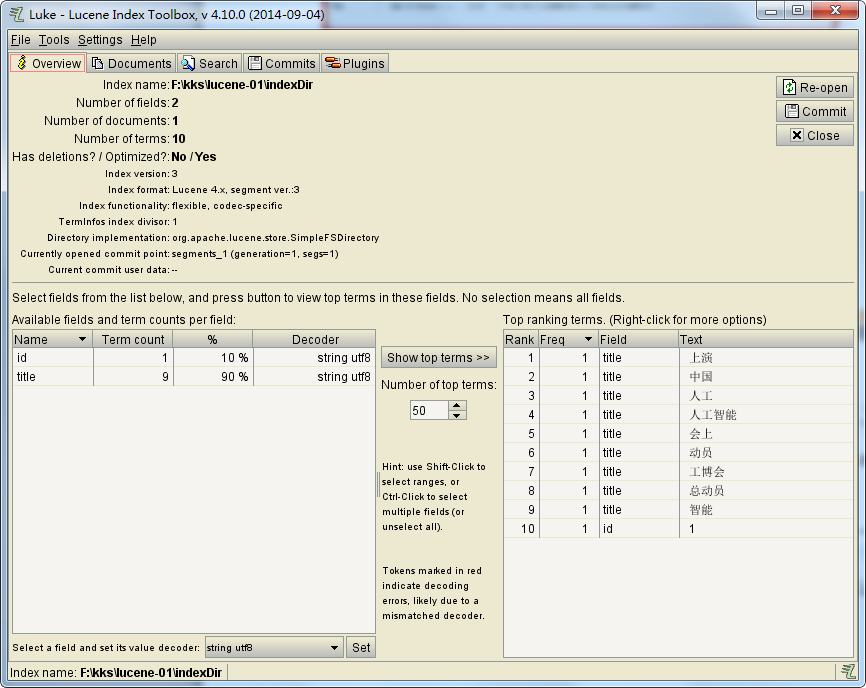


存储文档的信息

从以上可以看出，使用标准分词器，对于中文的分词处理存在问题，因此，我们可以使用IK分词器。



使用IK分词器后，对于中文的分词支持是不错的。



## 3.2 查询索引

### 3.2.1 基本查询

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.index.DirectoryReader;  **import** org.apache.lucene.index.IndexReader;  **import** org.apache.lucene.queryparser.classic.QueryParser;  **import** org.apache.lucene.search.IndexSearcher;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.search.ScoreDoc;  **import** org.apache.lucene.search.TopDocs;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.wltea.analyzer.lucene.IKAnalyzer;  **public** **class** IndexSearch {  **public** **static** **void** main(String[] args) **throws** Exception {  // 索引目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 索引读取工具  IndexReader reader = DirectoryReader.*open*(directory);  // 索引搜索工具  IndexSearcher searcher = **new** IndexSearcher(reader);  // 创建查询解析器  QueryParser parser = **new** QueryParser("title", **new** IKAnalyzer());  // 创建查询对象  Query query = parser.parse("人工智能");  // 搜索数据  TopDocs topDocs = searcher.search(query, 10);  // 获取总条数  System.***out***.println("本次搜索共找到" + topDocs.totalHits + "条数据");  // 获取得分文档对象  ScoreDoc[] scoreDocs = topDocs.scoreDocs;  **for** (ScoreDoc scoreDoc : scoreDocs) {  // 取出文档编号  **int** docID = scoreDoc.doc;  // 根据编号去找文档  Document doc = reader.document(docID);  System.***out***.println("id: " + doc.get("id"));  System.***out***.println("title: " + doc.get("title")); }  }  } |

### 3.2.2 Term查询

Term(词条)是搜索的最小单位，不可再分词，值必须是字符串。

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.index.DirectoryReader;  **import** org.apache.lucene.index.IndexReader;  **import** org.apache.lucene.index.Term;  **import** org.apache.lucene.queryparser.classic.QueryParser;  **import** org.apache.lucene.search.IndexSearcher;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.search.ScoreDoc;  **import** org.apache.lucene.search.TermQuery;  **import** org.apache.lucene.search.TopDocs;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.wltea.analyzer.lucene.IKAnalyzer;  **public** **class** IndexSearch {  **public** **static** **void** main(String[] args) **throws** Exception {  // 索引目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 索引读取工具  IndexReader reader = DirectoryReader.*open*(directory);  // 索引搜索工具  IndexSearcher searcher = **new** IndexSearcher(reader);  // 创建查询对象  Query query = **new** TermQuery(**new** Term("title", "人工"));  // 搜索数据  TopDocs topDocs = searcher.search(query, 10);  // 获取总条数  System.***out***.println("本次搜索共找到" + topDocs.totalHits + "条数据");  // 获取得分文档对象  ScoreDoc[] scoreDocs = topDocs.scoreDocs;  **for** (ScoreDoc scoreDoc : scoreDocs) {  // 取出文档编号  **int** docID = scoreDoc.doc;  // 根据编号去找文档  Document doc = reader.document(docID);  System.***out***.println("id: " + doc.get("id"));  System.***out***.println("title: " + doc.get("title"));  }  }  } |

### 3.2.3通配符查询

WildcardQuery可以进行测试通配符查询，?可以代表任意一个字符，\*可以任意多个任意字符。

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.index.DirectoryReader;  **import** org.apache.lucene.index.IndexReader;  **import** org.apache.lucene.index.Term;  **import** org.apache.lucene.queryparser.classic.QueryParser;  **import** org.apache.lucene.search.IndexSearcher;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.search.ScoreDoc;  **import** org.apache.lucene.search.TermQuery;  **import** org.apache.lucene.search.TopDocs;  **import** org.apache.lucene.search.WildcardQuery;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.wltea.analyzer.lucene.IKAnalyzer;  **public** **class** IndexSearch {  **public** **static** **void** main(String[] args) **throws** Exception {  // 索引目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 索引读取工具  IndexReader reader = DirectoryReader.*open*(directory);  // 索引搜索工具  IndexSearcher searcher = **new** IndexSearcher(reader);  // 创建查询对象  Query query = **new** WildcardQuery(**new** Term("title", "\*智能\*"));  // 搜索数据  TopDocs topDocs = searcher.search(query, 10);  // 获取总条数  System.***out***.println("本次搜索共找到" + topDocs.totalHits + "条数据");  // 获取得分文档对象  ScoreDoc[] scoreDocs = topDocs.scoreDocs;  **for** (ScoreDoc scoreDoc : scoreDocs) {  // 取出文档编号  **int** docID = scoreDoc.doc;  // 根据编号去找文档  Document doc = reader.document(docID);  System.***out***.println("id: " + doc.get("id"));  System.***out***.println("title: " + doc.get("title"));  }  }  } |

### 3.2.4模糊查询

FuzzyQuery可以进行模糊查询，创建模糊查询对象:允许用户输错。但是要求错误的最大编辑距离不能超过2。

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.index.DirectoryReader;  **import** org.apache.lucene.index.IndexReader;  **import** org.apache.lucene.index.Term;  **import** org.apache.lucene.search.FuzzyQuery;  **import** org.apache.lucene.search.IndexSearcher;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.search.ScoreDoc;  **import** org.apache.lucene.search.TopDocs;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **public** **class** IndexSearch {  **public** **static** **void** main(String[] args) **throws** Exception {  // 索引目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 索引读取工具  IndexReader reader = DirectoryReader.*open*(directory);  // 索引搜索工具  IndexSearcher searcher = **new** IndexSearcher(reader);  // 创建查询对象  Query query = **new** FuzzyQuery(**new** Term("title","智商"),1);  // 搜索数据  TopDocs topDocs = searcher.search(query, 10);  // 获取总条数  System.***out***.println("本次搜索共找到" + topDocs.totalHits + "条数据");  // 获取得分文档对象  ScoreDoc[] scoreDocs = topDocs.scoreDocs;  **for** (ScoreDoc scoreDoc : scoreDocs) {  // 取出文档编号  **int** docID = scoreDoc.doc;  // 根据编号去找文档  Document doc = reader.document(docID);  System.***out***.println("id: " + doc.get("id"));  System.***out***.println("title: " + doc.get("title"));  }  }  } |

### 3.2.5数值范围查询

数值范围查询使用NumericRangeQuery，可以用来对非String类型的ID进行精确的查找。

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.index.DirectoryReader;  **import** org.apache.lucene.index.IndexReader;  **import** org.apache.lucene.search.IndexSearcher;  **import** org.apache.lucene.search.NumericRangeQuery;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.search.ScoreDoc;  **import** org.apache.lucene.search.TopDocs;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **public** **class** IndexSearch {  **public** **static** **void** main(String[] args) **throws** Exception {  // 索引目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 索引读取工具  IndexReader reader = DirectoryReader.*open*(directory);  // 索引搜索工具  IndexSearcher searcher = **new** IndexSearcher(reader);  // 创建查询对象  Query query = NumericRangeQuery.*newIntRange*("id", 1, 2, **true**, **true**);  // 搜索数据  TopDocs topDocs = searcher.search(query, 10);  // 获取总条数  System.***out***.println("本次搜索共找到" + topDocs.totalHits + "条数据");  // 获取得分文档对象  ScoreDoc[] scoreDocs = topDocs.scoreDocs;  **for** (ScoreDoc scoreDoc : scoreDocs) {  // 取出文档编号  **int** docID = scoreDoc.doc;  // 根据编号去找文档  Document doc = reader.document(docID);  System.***out***.println("id: " + doc.get("id"));  System.***out***.println("title: " + doc.get("title"));  }  }  } |

### 3.2.6组合查询

布尔查询本身没有查询条件，可以把其它查询通过逻辑运算进行组合，Occur.MUST表示交集，Occur.SHOULD表示并集，Occur.MUST\_NOT表示非。

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.index.DirectoryReader;  **import** org.apache.lucene.index.IndexReader;  **import** org.apache.lucene.search.BooleanClause.Occur;  **import** org.apache.lucene.search.BooleanQuery;  **import** org.apache.lucene.search.IndexSearcher;  **import** org.apache.lucene.search.NumericRangeQuery;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.search.ScoreDoc;  **import** org.apache.lucene.search.TopDocs;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **public** **class** IndexSearch {  **public** **static** **void** main(String[] args) **throws** Exception {  // 索引目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 索引读取工具  IndexReader reader = DirectoryReader.*open*(directory);  // 索引搜索工具  IndexSearcher searcher = **new** IndexSearcher(reader);  // 创建查询对象  Query query1 = NumericRangeQuery.*newIntRange*("id", 1, 3, **true**, **true**);  Query query2 = NumericRangeQuery.*newIntRange*("id", 2, 4, **true**, **true**);  // 创建布尔查询的对象  BooleanQuery query = **new** BooleanQuery();  // 组合其它查询  query.add(query1, Occur.***MUST\_NOT***);  query.add(query2, Occur.***SHOULD***);  // 搜索数据  TopDocs topDocs = searcher.search(query, 10);  // 获取总条数  System.***out***.println("本次搜索共找到" + topDocs.totalHits + "条数据");  // 获取得分文档对象  ScoreDoc[] scoreDocs = topDocs.scoreDocs;  **for** (ScoreDoc scoreDoc : scoreDocs) {  // 取出文档编号  **int** docID = scoreDoc.doc;  // 根据编号去找文档  Document doc = reader.document(docID);  System.***out***.println("id: " + doc.get("id"));  System.***out***.println("title: " + doc.get("title"));  }  }  } |

## 3.3 修改索引

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.document.Field.Store;  **import** org.apache.lucene.document.StringField;  **import** org.apache.lucene.document.TextField;  **import** org.apache.lucene.index.IndexWriter;  **import** org.apache.lucene.index.IndexWriterConfig;  **import** org.apache.lucene.index.Term;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.apache.lucene.util.Version;  **import** org.wltea.analyzer.lucene.IKAnalyzer;  **public** **class** IndexUpdate {  **public** **static** **void** main(String[] args) **throws** Exception {  //创建目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  //创建配置对象  IndexWriterConfig conf = **new** IndexWriterConfig(Version.***LATEST***, **new** IKAnalyzer());  //创建索引写出工具  IndexWriter writer = **new** IndexWriter(directory, conf);  //创建新的文档数据  Document doc = **new** Document();  doc.add(**new** StringField("id", "1", Store.***YES***));  doc.add(**new** TextField("title", "美媒称中国科技创新拥有秘密武器：战略性和创新性思维", Store.***YES***));  //修改索引  writer.updateDocument(**new** Term("id", "1"), doc);  //提交  writer.commit();  //关闭  writer.close();  }  } |

## 3.4 删除索引

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.index.IndexWriter;  **import** org.apache.lucene.index.IndexWriterConfig;  **import** org.apache.lucene.index.Term;  **import** org.apache.lucene.search.NumericRangeQuery;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.apache.lucene.util.Version;  **import** org.wltea.analyzer.lucene.IKAnalyzer;  **public** **class** IndexDelete {  **public** **static** **void** main(String[] args) **throws** Exception {  // 创建目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 创建配置对象  IndexWriterConfig conf = **new** IndexWriterConfig(Version.***LATEST***, **new** IKAnalyzer());  // 创建索引写出工具  IndexWriter writer = **new** IndexWriter(directory, conf);  // 根据词条进行删除  writer.deleteDocuments(**new** Term("id", "1"));  // 根据query对象删除  Query query = NumericRangeQuery.*newIntRange*("id", 2, 2, **true**, **true**);  writer.deleteDocuments(query);  // 删除所有  writer.deleteAll();  // 提交  writer.commit();  // 关闭  writer.close();  }  } |

# Lucene的高级使用

## 4.1高亮显示

高亮显示的主要实现原理在于，为所有的关键字添加一个HTML标签，通过该标签来设置高亮。

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.index.DirectoryReader;  **import** org.apache.lucene.index.IndexReader;  **import** org.apache.lucene.queryparser.classic.QueryParser;  **import** org.apache.lucene.search.IndexSearcher;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.search.ScoreDoc;  **import** org.apache.lucene.search.TopDocs;  **import** org.apache.lucene.search.highlight.Formatter;  **import** org.apache.lucene.search.highlight.Highlighter;  **import** org.apache.lucene.search.highlight.QueryScorer;  **import** org.apache.lucene.search.highlight.Scorer;  **import** org.apache.lucene.search.highlight.SimpleHTMLFormatter;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.wltea.analyzer.lucene.IKAnalyzer;  **public** **class** IndexHighlighter {  **public** **static** **void** main(String[] args) **throws** Exception {  // 目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 创建读取工具  IndexReader reader = DirectoryReader.*open*(directory);  // 创建搜索工具  IndexSearcher searcher = **new** IndexSearcher(reader);  QueryParser parser = **new** QueryParser("title", **new** IKAnalyzer());  Query query = parser.parse("人工智能");  // 格式化器  Formatter formatter = **new** SimpleHTMLFormatter("<em>", "</em>");  Scorer scorer = **new** QueryScorer(query);  // 准备高亮工具  Highlighter highlighter = **new** Highlighter(formatter, scorer);  // 搜索  TopDocs topDocs = searcher.search(query, 10);  System.***out***.println("本次搜索共" + topDocs.totalHits + "条数据");  ScoreDoc[] scoreDocs = topDocs.scoreDocs;  **for** (ScoreDoc scoreDoc : scoreDocs) {  // 获取文档编号  **int** docID = scoreDoc.doc;  Document doc = reader.document(docID);  System.***out***.println("id: " + doc.get("id"));  String title = doc.get("title");  // 处理查询结果  String hTitle = highlighter.getBestFragment(**new** IKAnalyzer(), "title", title);  System.***out***.println("title: " + hTitle);  }  }  } |

## 4.2排序

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.index.DirectoryReader;  **import** org.apache.lucene.index.IndexReader;  **import** org.apache.lucene.queryparser.classic.QueryParser;  **import** org.apache.lucene.search.IndexSearcher;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.search.ScoreDoc;  **import** org.apache.lucene.search.Sort;  **import** org.apache.lucene.search.SortField;  **import** org.apache.lucene.search.SortField.Type;  **import** org.apache.lucene.search.TopDocs;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.wltea.analyzer.lucene.IKAnalyzer;  **public** **class** IndexSort {  **public** **static** **void** main(String[] args) **throws** Exception {  // 目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 创建读取工具  IndexReader reader = DirectoryReader.*open*(directory);  // 创建搜索工具  IndexSearcher searcher = **new** IndexSearcher(reader);  QueryParser parser = **new** QueryParser("title", **new** IKAnalyzer());  Query query = parser.parse("人工智能");  // 创建排序对象，false升序，true降序  Sort sort = **new** Sort(**new** SortField("id", Type.***INT***, **true**));  // 搜索  TopDocs topDocs = searcher.search(query, 10, sort);  System.***out***.println("本次搜索共" + topDocs.totalHits + "条数据");  ScoreDoc[] scoreDocs = topDocs.scoreDocs;  **for** (ScoreDoc scoreDoc : scoreDocs) {  // 获取文档编号  **int** docID = scoreDoc.doc;  Document doc = reader.document(docID);  System.***out***.println("id: " + doc.get("id"));  System.***out***.println("title: " + doc.get("title"));  }  }  } |

## 4.3分页

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.index.DirectoryReader;  **import** org.apache.lucene.index.IndexReader;  **import** org.apache.lucene.queryparser.classic.QueryParser;  **import** org.apache.lucene.search.IndexSearcher;  **import** org.apache.lucene.search.Query;  **import** org.apache.lucene.search.ScoreDoc;  **import** org.apache.lucene.search.Sort;  **import** org.apache.lucene.search.SortField;  **import** org.apache.lucene.search.SortField.Type;  **import** org.apache.lucene.search.TopDocs;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.wltea.analyzer.lucene.IKAnalyzer;  **public** **class** IndexPageQuery {  **public** **static** **void** main(String[] args) **throws** Exception {  //每页条数  **int** pageSize = 1;  //当前页码  **int** pageNum = 2;  //当前页的起始条数  **int** start = (pageNum - 1) \* pageSize;  //当前页的结束条数  **int** end = start + pageSize;    // 目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  // 创建读取工具  IndexReader reader = DirectoryReader.*open*(directory);  // 创建搜索工具  IndexSearcher searcher = **new** IndexSearcher(reader);    QueryParser parser = **new** QueryParser("title", **new** IKAnalyzer());  Query query = parser.parse("人工智能");    // 创建排序对象  Sort sort = **new** Sort(**new** SortField("id", Type.***INT***, **false**));  // 搜索数据  TopDocs topDocs = searcher.search(query, end,sort);  System.***out***.println("本次搜索共" + topDocs.totalHits + "条数据");    ScoreDoc[] scoreDocs = topDocs.scoreDocs;  **for** (**int** i = start; i < end; i++) {  ScoreDoc scoreDoc = scoreDocs[i];  // 获取文档编号  **int** docID = scoreDoc.doc;  Document doc = reader.document(docID);  System.***out***.println("id: " + doc.get("id"));  System.***out***.println("title: " + doc.get("title"));  }  }  } |

## 4.4加权算法

 Lucene会对搜索结果打分，用来表示文档数据与词条关联性的强弱，得分越高，表示查询的匹配度就越高，排名就越靠前。

|  |
| --- |
| **package** com.igeekhome.lucene;  **import** java.io.File;  **import** org.apache.lucene.analysis.Analyzer;  **import** org.apache.lucene.document.Document;  **import** org.apache.lucene.document.Field.Store;  **import** org.apache.lucene.document.IntField;  **import** org.apache.lucene.document.TextField;  **import** org.apache.lucene.index.IndexWriter;  **import** org.apache.lucene.index.IndexWriterConfig;  **import** org.apache.lucene.store.Directory;  **import** org.apache.lucene.store.FSDirectory;  **import** org.apache.lucene.util.Version;  **import** org.wltea.analyzer.lucene.IKAnalyzer;  **public** **class** IndexCreate {  **public** **static** **void** main(String[] args) **throws** Exception {  //创建文档对象  Document document = **new** Document();  //创建并添加字段信息  document.add(**new** IntField("id", 3, Store.***YES***));    //创建字段  TextField textField = **new** TextField("title", "韩资企业在渝达222家 深耕汽车研发制造、人工智能等领域", Store.***YES***);  //设置加权  textField.setBoost(2.0f);  //添加字段  document.add(textField);  //创建索引目录对象  Directory directory = FSDirectory.*open*(**new** File("indexDir"));  //创建分词器对象  Analyzer analyzer = **new** IKAnalyzer();  //创建配置对象  IndexWriterConfig conf = **new** IndexWriterConfig(Version.***LATEST***, analyzer);  //创建索引的写出工具类  IndexWriter indexWriter = **new** IndexWriter(directory, conf);  //添加文档  indexWriter.addDocument(document);  //提交  indexWriter.commit();  indexWriter.close();  }  } |