## 一个简单的搜索应用程序

假设我们的电脑的目录中含有很多文本文档，我们需要查找哪些文档含有某个关键 词。为了实现这种功能，我们首先利用 Lucene 对这个目录中的文档建立索引，然后在建立好的索引中搜索我们所要查找的文档。通过这个例子读者会对如何利用 Lucene 构建自己的搜索应用程序有个比较清楚的认识。

## 建立索引

为了对文档进行索引，Lucene 提供了五个基础的类，他们分别是 Document, Field, IndexWriter, Analyzer, Directory。下面我们分别介绍一下这五个类的用途：

**Document**

Document 是用来描述文档的，这里的文档可以指一个 HTML 页面，一封电子邮件，或者是一个文本文件。一个 Document 对象由多个 Field 对象组成的。可以把一个 Document 对象想象成数据库中的一个记录，而每个 Field 对象就是记录的一个字段。

**Field**

Field 对象是用来描述一个文档的某个属性的，比如一封电子邮件的标题和内容可以用两个 Field 对象分别描述。

**Analyzer**

在 一个文档被索引之前，首先需要对文档内容进行分词处理，这部分工作就是由 Analyzer 来做的。Analyzer 类是一个抽象类，它有多个实现。针对不同的语言和应用需要选择适合的 Analyzer。Analyzer 把分词后的内容交给 IndexWriter 来建立索引。

**IndexWriter**

IndexWriter 是 Lucene 用来创建索引的一个核心的类，他的作用是把一个个的 Document 对象加到索引中来。

**Directory**

这个类代表了 Lucene 的索引的存储的位置，这是一个抽象类，它目前有两个实现，第一个是 FSDirectory，它表示一个存储在文件系统中的索引的位置。第二个是 RAMDirectory，它表示一个存储在内存当中的索引的位置。

熟悉了建立索引所需要的这些类后，我们就开始对某个目录下面的文本文件建立索引了。

##### 清单 1. 对文本文件建立索引

package TestLucene;

import java.io.File;

import java.io.FileReader;

import java.io.Reader;

import java.util.Date;

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;

import org.apache.lucene.index.IndexWriter;

/\*\*

\* This class demonstrate the process of creating index with Lucene

\* for text files

\*/

public class TxtFileIndexer {

public static void main(String[] args) throws Exception{

//indexDir is the directory that hosts Lucene's index files

File indexDir = new File("D:\\luceneIndex");

//dataDir is the directory that hosts the text files that to be indexed

File dataDir = new File("D:\\luceneData");

Analyzer luceneAnalyzer = new StandardAnalyzer();

File[] dataFiles = dataDir.listFiles();

IndexWriter indexWriter = new IndexWriter(indexDir,luceneAnalyzer,true);

long startTime = new Date().getTime();

for(int i = 0; i < dataFiles.length; i++){

if(dataFiles[i].isFile() && dataFiles[i].getName().endsWith(".txt")){

System.out.println("Indexing file " + dataFiles[i].getCanonicalPath());

Document document = new Document();

Reader txtReader = new FileReader(dataFiles[i]);

document.add(Field.Text("path",dataFiles[i].getCanonicalPath()));

document.add(Field.Text("contents",txtReader));

document.add(Field.Text("tiatlw",txtReader));

indexWriter.addDocument(document);

}

}

indexWriter.optimize();

indexWriter.close();

long endTime = new Date().getTime();

System.out.println("It takes " + (endTime - startTime)

+ " milliseconds to create index for the files in directory "

+ dataDir.getPath());

}

}

在清单 1 中，我们注意到类 IndexWriter 的构造函数需要三个参数，第一个参数指定了所创建的索引要存放的位置，他可以是一个 File 对象，也可以是一个 FSDirectory 对象或者 RAMDirectory 对象。第二个参数指定了 Analyzer 类的一个实现，也就是指定这个索引是用哪个分词器对文挡内容进行分词。第三个参数是一个布尔型的变量，如果为 true 的话就代表创建一个新的索引，为 false 的话就代表在原来索引的基础上进行操作。接着程序遍历了目录下面的所有文本文档，并为每一个文本文档创建了一个 Document 对象。然后把文本文档的两个属性：路径和内容加入到了两个 Field 对象中，接着在把这两个 Field 对象加入到 Document 对象中，最后把这个文档用 IndexWriter 类的 add 方法加入到索引中去。这样我们便完成了索引的创建。接下来我们进入在建立好的索引上进行搜索的部分。

## 搜索文档

利用 Lucene 进行搜索就像建立索引一样也是非常方便的。在上面一部分中，我们已经为一个目录下的文本文档建立好了索引，现在我们就要在这个索引上进行搜索以找到包含某 个关键词或短语的文档。Lucene 提供了几个基础的类来完成这个过程，它们分别是呢 IndexSearcher, Term, Query, TermQuery, Hits. 下面我们分别介绍这几个类的功能。

**Query**

这是一个抽象类，他有多个实现，比如 TermQuery, BooleanQuery, PrefixQuery. 这个类的目的是把用户输入的查询字符串封装成 Lucene 能够识别的 Query。

**Term**

Term 是搜索的基本单位，一个 Term 对象有两个 String 类型的域组成。生成一个 Term 对象可以有如下一条语句来完成：Term term = new Term(“fieldName”,”queryWord”); 其中第一个参数代表了要在文档的哪一个 Field 上进行查找，第二个参数代表了要查询的关键词。

**TermQuery**

TermQuery 是抽象类 Query 的一个子类，它同时也是 Lucene 支持的最为基本的一个查询类。生成一个 TermQuery 对象由如下语句完成： TermQuery termQuery = new TermQuery(new Term(“fieldName”,”queryWord”)); 它的构造函数只接受一个参数，那就是一个 Term 对象。

**IndexSearcher**

IndexSearcher 是用来在建立好的索引上进行搜索的。它只能以只读的方式打开一个索引，所以可以有多个 IndexSearcher 的实例在一个索引上进行操作。

**TopDocs**

TopDocs是用来保存搜索的结果的。

介绍完这些搜索所必须的类之后，我们就开始在之前所建立的索引上进行搜索了。

##### 清单 2 ：在建立好的索引上进行搜索

package TestLucene;

import java.io.File;

import org.apache.lucene.document.Document;

import org.apache.lucene.index.Term;

import org.apache.lucene.search.Hits;

import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.search.TermQuery;

import org.apache.lucene.store.FSDirectory;

/\*\*

\* This class is used to demonstrate the

\* process of searching on an existing

\* Lucene index

\*

\*/

public class TxtFileSearcher {

public static void main(String[] args) throws Exception{

String queryStr = "lucene";

//This is the directory that hosts the Lucene index

File indexDir = new File("D:\\luceneIndex");

FSDirectory directory = FSDirectory.getDirectory(indexDir,false);

IndexSearcher searcher = new IndexSearcher(directory);

if(!indexDir.exists()){

System.out.println("The Lucene index is not exist");

return;

}

Term term = new Term("contents",queryStr.toLowerCase());

TermQuery luceneQuery = new TermQuery(term);

Hits hits = searcher.search(luceneQuery);

for(int i = 0; i < hits.length(); i++){

Document document = hits.doc(i);

System.out.println("File: " + document.get("path"));

}

}

}

在清单 2 中，类 IndexSearcher 的构造函数接受一个类型为 Directory 的对象，Directory 是一个抽象类，它目前有两个子类：FSDirctory 和 RAMDirectory. 我们的程序中传入了一个 FSDirctory 对象作为其参数，代表了一个存储在磁盘上的索引的位置。构造函数执行完成后，代表了这个 IndexSearcher 以只读的方式打开了一个索引。然后我们程序构造了一个 Term 对象，通过这个 Term 对象，我们指定了要在文档的内容中搜索包含关键词”lucene”的文档。接着利用这个 Term 对象构造出 TermQuery 对象并把这个 TermQuery 对象传入到 IndexSearcher 的 search 方法中进行查询，返回的结果保存在 Hits 对象中。最后我们用了一个循环语句把搜索到的文档的路径都打印了出来。 好了，我们的搜索应用程序已经开发完毕，怎么样，利用 Lucene 开发搜索应用程序是不是很简单。

## 总结

本文首先介绍了 Lucene 的一些基本概念，然后开发了一个应用程序演示了利用 Lucene 建立索引并在该索引上进行搜索的过程。希望本文能够为学习 Lucene 的读者提供帮助。

开发包：http://www.apache.org/dyn/closer.lua/lucene/java/6.2.1

<http://www.yiibai.com/lucene/lucene_indexwriter.html>

Field详解:http://blog.csdn.net/zhaoxiao2008/article/details/14180019

http://yingbin920.iteye.com/blog/1554852