**Lucene 编程**

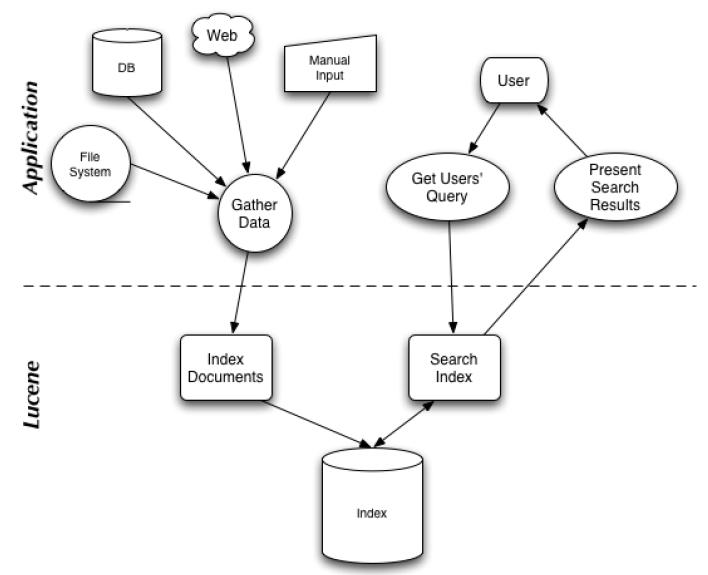
**Lucene 简介**

Lucene 是一个基于 Java 的全文信息检索工具包，它不是一个完整的搜索应用程序，而是为你的应用程序提供索引和搜索功能。Lucene 目前是 Apache Jakarta 家族中的一个开源项目。也是目前最为流行的基于 Java 开源全文检索工具包。

目前已经有很多应用程序的搜索功能是基于 Lucene 的，比如 Eclipse 的帮助系统的搜索功能。Lucene 能够为文本类型的数据建立索引，所以你只要能把你要索引的数据格式转化的文本的，Lucene 就能对你的文档进行索引和搜索。比如你要对一些 HTML 文档，PDF 文档进行索引的话你就首先需要把 HTML 文档和 PDF 文档转化成文本格式的，然后将转化后的内容交给 Lucene 进行索引，然后把创建好的索引文件保存到磁盘或者内存中，最后根据用户输入的查询条件在索引文件上进行查询。不指定要索引的文档的格式也使 Lucene 能够几乎适用于所有的搜索应用程序。

图 1 表示了搜索应用程序和 Lucene 之间的关系，也反映了利用 Lucene 构建搜索应用程序的流程：

**图 1. 搜索应用程序和 Lucene 之间的关系**



**索引和搜索**

索引是现代搜索引擎的核心，建立索引的过程就是把源数据处理成非常方便查询的索引文件的过程。为什么索引这么重要呢，试想你现在要在大量的文档中搜索含有某个关键词的文档，那么如果不建立索引的话你就需要把这些文档顺序的读入内存，然后检查这个文章中是不是含有要查找的关键词，这样的话就会耗费非常多的时间，想想搜索引擎可是在毫秒级的时间内查找出要搜索的结果的。这就是由于建立了索引的原因，你可以把索引想象成这样一种数据结构，他能够使你快速的随机访问存储在索引中的关键词，进而找到该关键词所关联的文档。Lucene 采用的是一种称为反向索引（inverted index）的机制。反向索引就是说我们维护了一个词 / 短语表，对于这个表中的每个词 / 短语，都有一个链表描述了有哪些文档包含了这个词 / 短语。这样在用户输入查询条件的时候，就能非常快的得到搜索结果。我们将在本系列文章的第二部分详细介绍 Lucene 的索引机制，由于 Lucene 提供了简单易用的 API，所以即使读者刚开始对全文本进行索引的机制并不太了解，也可以非常容易的使用 Lucene 对你的文档实现索引。

对文档建立好索引后，就可以在这些索引上面进行搜索了。搜索引擎首先会对搜索的关键词进行解析，然后再在建立好的索引上面进行查找，最终返回和用户输入的关键词相关联的文档。

**Lucene 软件包分析**

Lucene 软件包的发布形式是一个 JAR 文件，下面我们分析一下这个 JAR 文件里面的主要的 JAVA 包，使读者对之有个初步的了解。

Package: org.apache.lucene.document

这个包提供了一些为封装要索引的文档所需要的类，比如 Document, Field。这样，每一个文档最终被封装成了一个 Document 对象。

Package: org.apache.lucene.analysis

这个包主要功能是对文档进行分词，因为文档在建立索引之前必须要进行分词，所以这个包的作用可以看成是为建立索引做准备工作。

Package: org.apache.lucene.index

这个包提供了一些类来协助创建索引以及对创建好的索引进行更新。这里面有两个基础的类：IndexWriter 和 IndexReader，其中 IndexWriter 是用来创建索引并添加文档到索引中的，IndexReader 是用来删除索引中的文档的。

Package: org.apache.lucene.search

这个包提供了对在建立好的索引上进行搜索所需要的类。比如 IndexSearcher 和 Hits, IndexSearcher 定义了在指定的索引上进行搜索的方法，Hits 用来保存搜索得到的结果。

**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 给出了对某个目录下的文本文件建立索引的源代码。

**清单 1. 对文本文件建立索引**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | 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));                 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。

**Hits**

Hits 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 给出了完成搜索功能所需要的代码。

**清单 2 ：在建立好的索引上进行搜索**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | 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。