Information Retrieval

Internal Assignment

1. Lucene

Lucene is a simple yet powerful Java-based **Search** library. It can be used in any application to add search capability to it.

Lucene is an open-source project.

It is scalable.

This high-performance library is used to index and search virtually any kind of text.

Lucene library provides the core operations which are required by any search application. Indexing and Searching.

## Lucene's Role in Search Application

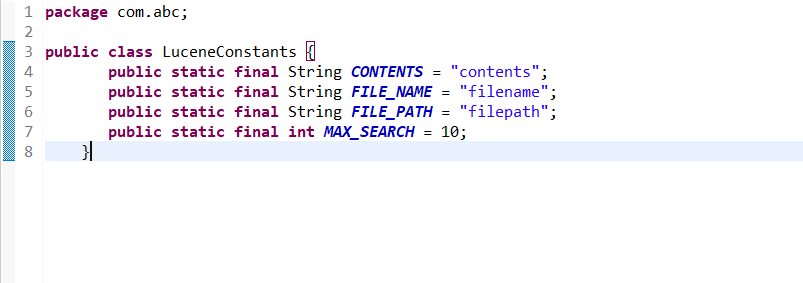
Lucene plays role in steps 2 to step 7 mentioned above and provides classes to do the required operations.

In a nutshell, Lucene is the heart of any search application and provides vital operations pertaining to indexing and searching.

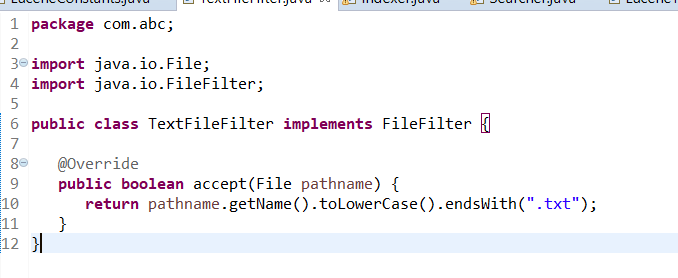
Acquiring contents and displaying the results is left for the application part to handle.

Implementation of Lucene using Eclipse

### LuceneConstants.java



### TextFileFilter.java



### Indexer.java

**package** com.abc;

**import** java.io.File;

**import** java.io.FileFilter;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** org.apache.lucene.analysis.standard.StandardAnalyzer;

**import** org.apache.lucene.document.Document;

**import** org.apache.lucene.document.Field;

**import** org.apache.lucene.index.CorruptIndexException;

**import** org.apache.lucene.index.IndexWriter;

**import** org.apache.lucene.store.Directory;

**import** org.apache.lucene.store.FSDirectory;

**import** org.apache.lucene.util.Version;

**public** **class** Indexer {

**private** IndexWriter writer;

**public** Indexer(String indexDirectoryPath) **throws** IOException {

//this directory will contain the indexes

Directory indexDirectory =

FSDirectory.*open*(**new** File(indexDirectoryPath));

//create the indexer

writer = **new** ~~IndexWriter~~(indexDirectory,

**new** StandardAnalyzer(Version.***LUCENE\_36***),**true**,

IndexWriter.~~MaxFieldLength~~.~~UNLIMITED~~);

}

**public** **void** close() **throws** CorruptIndexException, IOException {

writer.close();

}

**private** Document getDocument(File file) **throws** IOException {

Document document = **new** Document();

//index file contents

Field contentField = **new** Field(LuceneConstants.***CONTENTS***, **new** FileReader(file));

//index file name

Field fileNameField = **new** Field(LuceneConstants.***FILE\_NAME***,

file.getName(),Field.Store.***YES***,Field.Index.***NOT\_ANALYZED***);

//index file path

Field filePathField = **new** Field(LuceneConstants.***FILE\_PATH***,

file.getCanonicalPath(),Field.Store.***YES***,Field.Index.***NOT\_ANALYZED***);

document.add(contentField);

document.add(fileNameField);

document.add(filePathField);

**return** document;

}

**private** **void** indexFile(File file) **throws** IOException {

System.***out***.println("Indexing "+file.getCanonicalPath());

Document document = getDocument(file);

writer.addDocument(document);

}

**public** **int** createIndex(String dataDirPath, FileFilter filter)

**throws** IOException {

//get all files in the data directory

File[] files = **new** File(dataDirPath).listFiles();

**for** (File file : files) {

**if**(!file.isDirectory()

&& !file.isHidden()

&& file.exists()

&& file.canRead()

&& filter.accept(file)

){

indexFile(file);

}

}

**return** writer.numDocs();

}

}

### Searcher.java

**package** com.abc;

**import** java.io.File;

**import** java.io.IOException;

**import** org.apache.lucene.analysis.standard.StandardAnalyzer;

**import** org.apache.lucene.document.Document;

**import** org.apache.lucene.index.CorruptIndexException;

**import** org.apache.lucene.queryParser.ParseException;

**import** org.apache.lucene.queryParser.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.apache.lucene.util.Version;

**public** **class** Searcher {

IndexSearcher indexSearcher;

QueryParser queryParser;

Query query;

**public** Searcher(String indexDirectoryPath)

**throws** IOException {

Directory indexDirectory =

FSDirectory.*open*(**new** File(indexDirectoryPath));

indexSearcher = **new** ~~IndexSearcher~~(indexDirectory);

queryParser = **new** QueryParser(Version.***LUCENE\_36***,

LuceneConstants.***CONTENTS***,

**new** StandardAnalyzer(Version.***LUCENE\_36***));

}

**public** TopDocs search( String searchQuery)

**throws** IOException, ParseException {

query = queryParser.parse(searchQuery);

**return** indexSearcher.search(query, LuceneConstants.***MAX\_SEARCH***);

}

**public** Document getDocument(ScoreDoc scoreDoc)

**throws** CorruptIndexException, IOException {

**return** indexSearcher.doc(scoreDoc.doc);

}

**public** **void** close() **throws** IOException {

indexSearcher.close();

}

}

### LuceneTester.java

**package** com.abc;

**import** java.io.IOException;

**import** org.apache.lucene.document.Document;

**import** org.apache.lucene.queryParser.ParseException;

**import** org.apache.lucene.search.ScoreDoc;

**import** org.apache.lucene.search.TopDocs;

**public** **class** LuceneTester {

String indexDir = "D:\\Lucene\\Index";

String dataDir = "D:\\Lucene\\Data";

Indexer indexer;

Searcher searcher;

**public** **static** **void** main(String[] args) {

LuceneTester tester;

**try** {

tester = **new** LuceneTester();

tester.createIndex();

tester.search("Mohan");

} **catch** (IOException e) {

e.printStackTrace();

} **catch** (ParseException e) {

e.printStackTrace();

}

}

**private** **void** createIndex() **throws** IOException {

indexer = **new** Indexer(indexDir);

**int** numIndexed;

**long** startTime = System.*currentTimeMillis*();

numIndexed = indexer.createIndex(dataDir, **new** TextFileFilter());

**long** endTime = System.*currentTimeMillis*();

indexer.close();

System.***out***.println(numIndexed+" File indexed, time taken: "

+(endTime-startTime)+" ms");

}

**private** **void** search(String searchQuery) **throws** IOException, ParseException {

searcher = **new** Searcher(indexDir);

**long** startTime = System.*currentTimeMillis*();

TopDocs hits = searcher.search(searchQuery);

**long** endTime = System.*currentTimeMillis*();

System.***out***.println(hits.totalHits +

" documents found. Time :" + (endTime - startTime));

**for**(ScoreDoc scoreDoc : hits.scoreDocs) {

Document doc = searcher.getDocument(scoreDoc);

System.***out***.println("File: "

+ doc.get(LuceneConstants.***FILE\_PATH***));

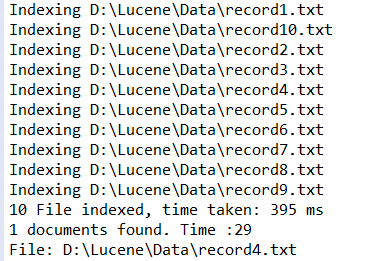
}

searcher.close();

}

}

Output:



Conclusion:

* Lucene is the library which helps to search data through keywords using a searching mechanism.
* In the above example data is in the form of text files.
* Total 10 text files are created to demonstrate lucene search mechanism.
* Keyword used to search is “Mohan” which was found in 4th text file by the lucene library.
* Lucene can be integrated in any of the application as it is platform independent.