# Installing Lucene in a Windows Local Machine

## Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.

## Step 1: Download Lucene for Windows

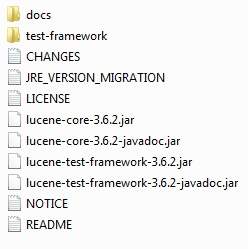
Following are the simple steps to download and install the framework on your machine:

<http://archive.apache.org/dist/lucene/java/>.

* Make a choice whether you want to install Lucene on Windows, or Unix and then proceed to the next step to download .zip file for windows and .tz file for Unix.
* Download the suitable version (3.6.2 used for this tutorial) of Lucene framework binaries from <http://archive.apache.org/dist/lucene/java/3.6.2/>

## Step 2: Unzip the File to a Folder

* For **lucene-3.6.2.zip**, and when you unzip the downloaded file it will give you directory structure inside C:\lucene-3.6.2 as follows.



# Lucene (Fuzzy Search) - Java Integration

The below sections explains how to use Lucene Fuzzy Search in a Java application.

## Prerequisites

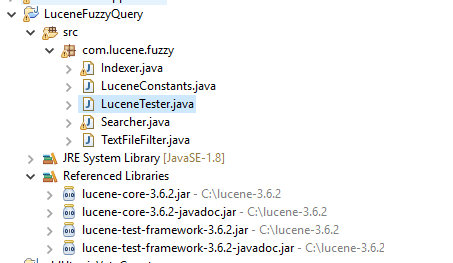
The prerequisites needed are:

* Locally installed Lucene
* Java IDE like Eclipse.
* Lucene APIs: Found inside the Lucene installation folder itself.

## Create a new Java Project

Create a new Java project in Eclipse

Add the below libraries in the Java Build Path. The Lucene jar files are found in the Lucene unzipped folder. In this case, C:\lucene-3.6.2



## Java Code for *LuceneConstants* Class

package com.lucene.fuzzy;

public class LuceneConstants {

public static final String CONTENTS="contents";

public static final String FILE\_NAME="filename";

public static final String FILE\_PATH="filepath";

public static final int MAX\_SEARCH = 10;

}

## Java Code for *TextFileFilter* Class

package com.lucene.fuzzy;

import java.io.File;

import java.io.FileFilter;

public class TextFileFilter implements FileFilter {

@Override

public boolean accept(File pathname) {

return pathname.getName().toLowerCase().endsWith(".txt");

}

}

## Java Code for *Indexer* Class

Create a new class in the project and use the below source code:

**package** com.lucene.fuzzy;

**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();

}

}

**Java Code for *Searcher* Class**

package com.lucene.fuzzy;

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 TopDocs search(Query query) throws IOException, ParseException{

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();

}

}

**Java Code for *LuceneTester* Class (Main)**

**package** com.lucene.fuzzy;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

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

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

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

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

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

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

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

**import** com.lucene.fuzzy.Indexer;

**import** com.lucene.fuzzy.TextFileFilter;

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

String indexDir = "C:\\lucene-3.6.2\\ProductIndex";

String dataDir = "C:\\lucene-3.6.2\\ProductData";

Indexer indexer;

Searcher searcher;

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

LuceneTester tester;

**try** {

tester = **new** LuceneTester();

tester.createIndex();

//Search text here

BufferedReader br;

String choice = "";

System.***out***.println("\*\*\*\*\* Lucene Index, Search Tester \*\*\*\*\*\*");

System.***out***.println("Enter the name to search:");

br = **new** BufferedReader(**new** InputStreamReader(System.***in***));

**try** {

choice = br.readLine();

} **catch** (IOException e) {

e.printStackTrace();

}

**try** {

br.close();

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

//Optional for phrases (multi words)

String[] searcharray = choice.split(" ");

**for** (**int** i = 0; i < searcharray.length; i++)

{

System.***out***.println("Search results for word " + (i+1) + ": " + searcharray[i]);

tester.searchUsingFuzzyQuery(searcharray[i]);

}

//Uncomment below line for one word queries

//tester.searchUsingFuzzyQuery(choice);

} **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** searchUsingFuzzyQuery(String searchQuery)

**throws** IOException, ParseException{

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

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

//create a term to search file name

//Term term = new Term(LuceneConstants.FILE\_NAME, searchQuery);

Term term = **new** Term(LuceneConstants.***CONTENTS***, searchQuery);

//create the term query object

Query query = **new** FuzzyQuery(term);

//do the search

TopDocs hits = searcher.search(query);

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

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

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

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

Document doc = searcher.getDocument(scoreDoc);

System.***out***.print("Score: "+ scoreDoc.score + " ");

System.***out***.println("File: "+ doc.get(LuceneConstants.***FILE\_PATH***));

}

searcher.close();

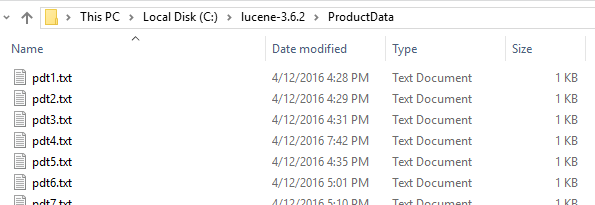
}

}

**Create Document Folder**

Create a new folder **C:\lucene-3.6.2\ProductData**

Create a number of .txt files with text inputs. We have used product names and descriptions from Amazon.com



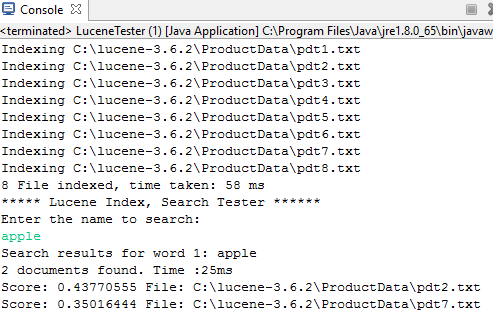
**Create Index Folder**

Create a new empty folder **C:\lucene-3.6.2\ProductIndex**

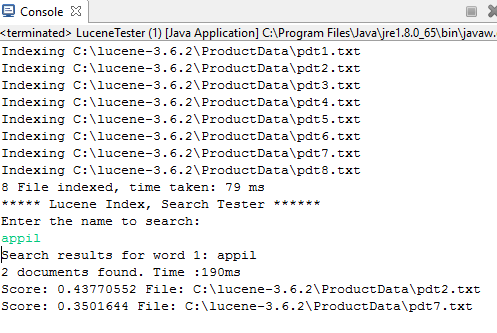
Lucene will take care of creation of indexes during runtime in this location.

### Results:

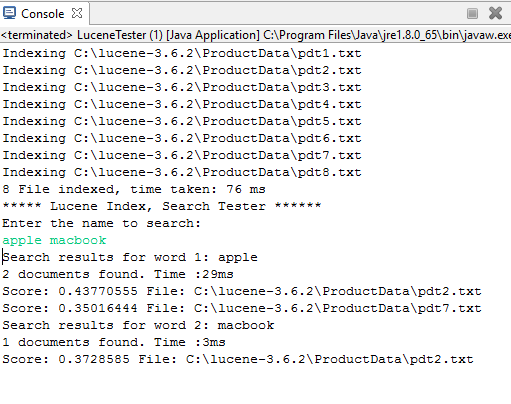
Single word: (Non fuzzy)



Single word: (fuzzy)



Phrase (Non fuzzy)

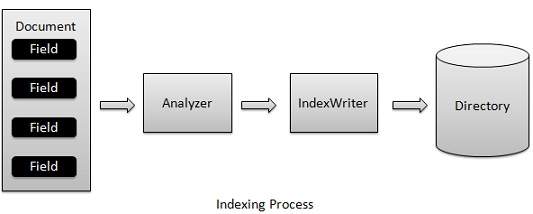


Phrase (Fuzzy)

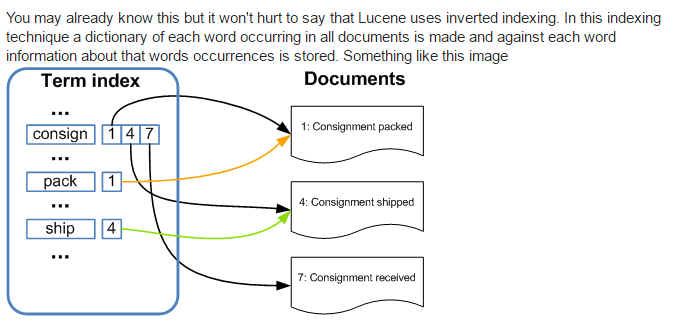
### 

## How does Lucene work?

Indexing process is one of the core functionality provided by Lucene. Following diagram illustrates the indexing process and use of classes. IndexWriter is the most important and core component of the indexing process.



We add Document(s) containing Field(s) to IndexWriter which analyzes the Document(s) using the Analyzer and then creates/open/edit indexes as required and store/update them in a Directory. IndexWriter is used to update or create indexes. It is not used to read indexes.



#### Additional Reading Suggested:

<http://www.tutorialspoint.com/lucene/lucene_first_application.htm>

<http://www.tutorialspoint.com/lucene/lucene_fuzzyquery.htm>