



NLP and IR

Building your first Search Engine with Lucene

Aliaksei Severyn

University of Trento, Italy

April 06, 2012

Plan for the lab

- Introduction to Lucene Search Engine
- Lucene concepts
- Hands-on experience with indexing and searching
 - HelloWorld example
- Using search engine to retrieve answer passages for Question Answering system
 - Indexing and searching 180k of QA corpus

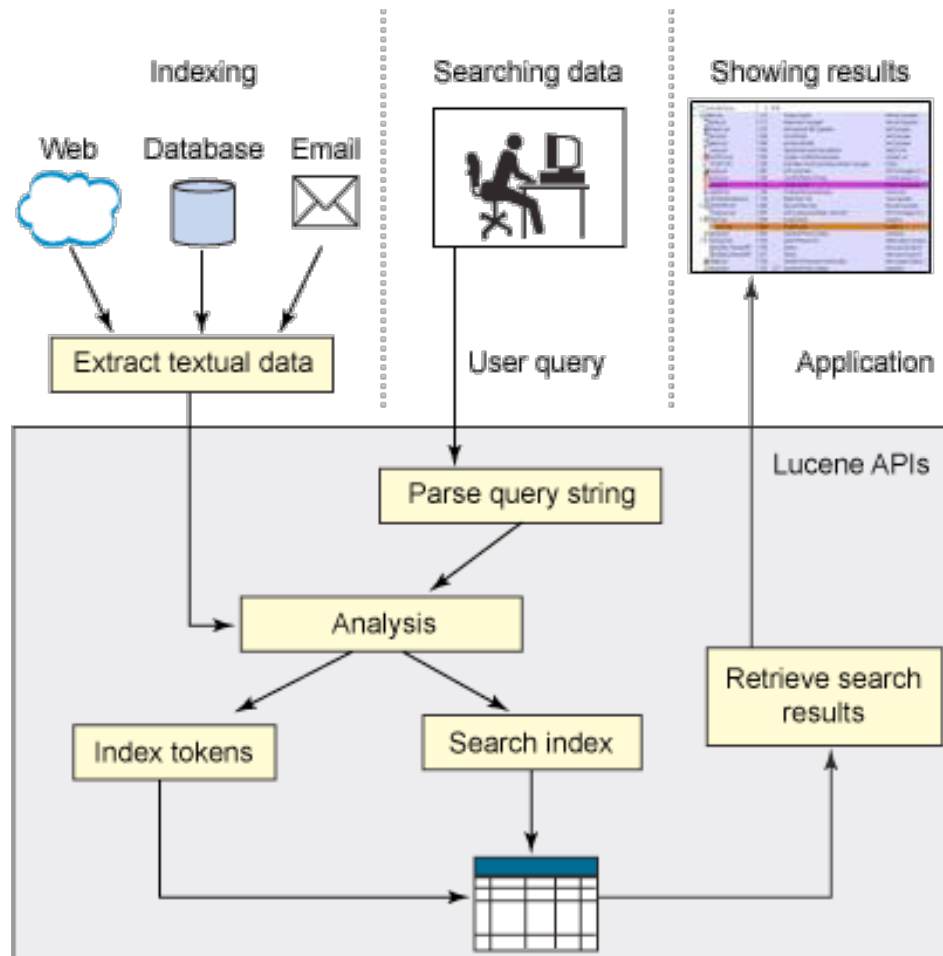
What is Lucene

- software library for search
- open source
- not a complete application
- set of java classes
- active user and developer communities
- widely used, e.g, IBM and Microsoft.

High level overview

- Lucene is a full-text search library
- Designed to add search to your application
- Maintains a full-text index.
- Searches the index and returns results ranked by either the relevance to the query (or by an arbitrary field such as a document's last modified date.)

Typical architecture of a Lucene search app



Our first HelloWorld app with Lucene

- Create an in-memory index
- Add a few documents
- Construct a query
- Search an index
- Display results

Setting up our first example

Download Lucene sources and binary from:

<http://www.apache.org/dist/lucene/java/3.5.0/>

Or download everything from :

<http://disi.unitn.it/~severyn/NLPIR.2012/lab01/intro.tar.gz>

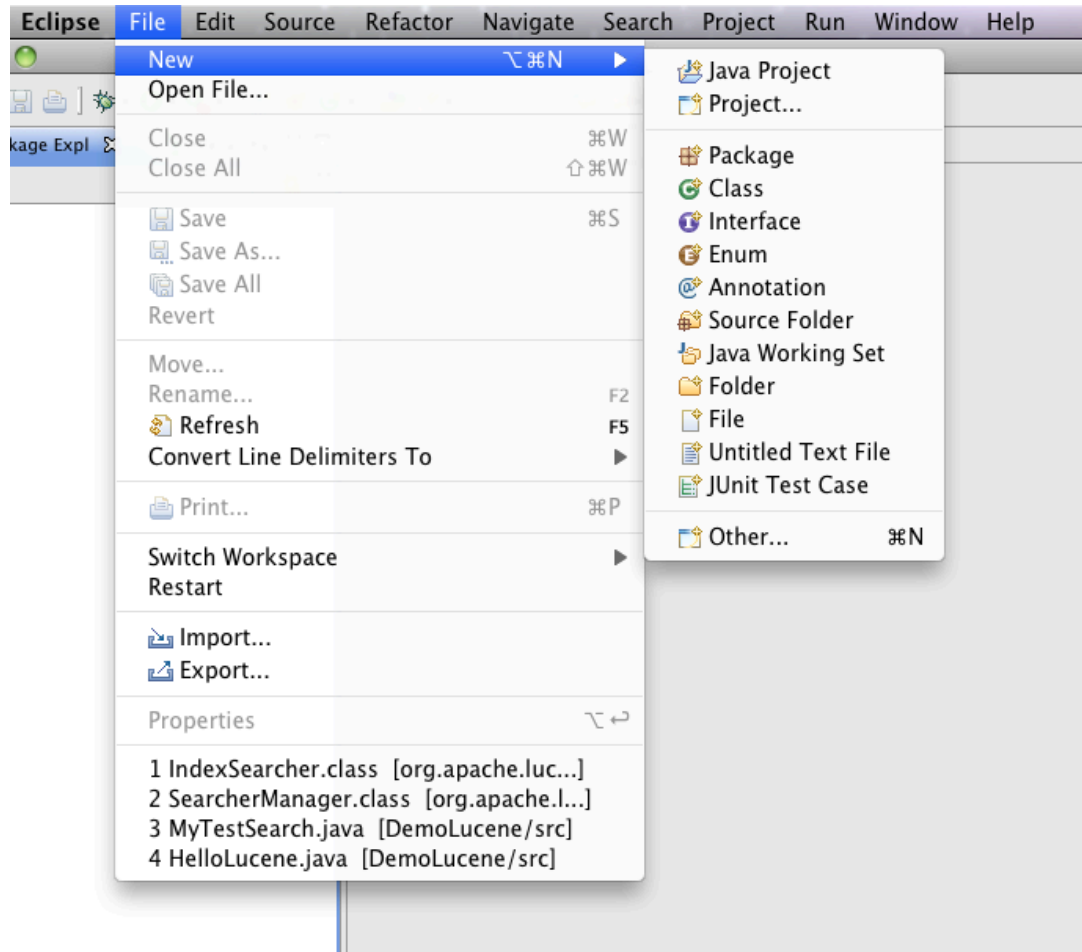
E.g. try the following in your terminal:

```
$ wget
  http://disi.unitn.it/~severyn/NLPIR.2012/lab01/intro.tar.gz
$ tar xvfz lab01.tar.gz
$ cd lab01
$ javac -cp .:lucene-core-3.5.0.jar HelloLucene.java
$ java HelloLucene
```

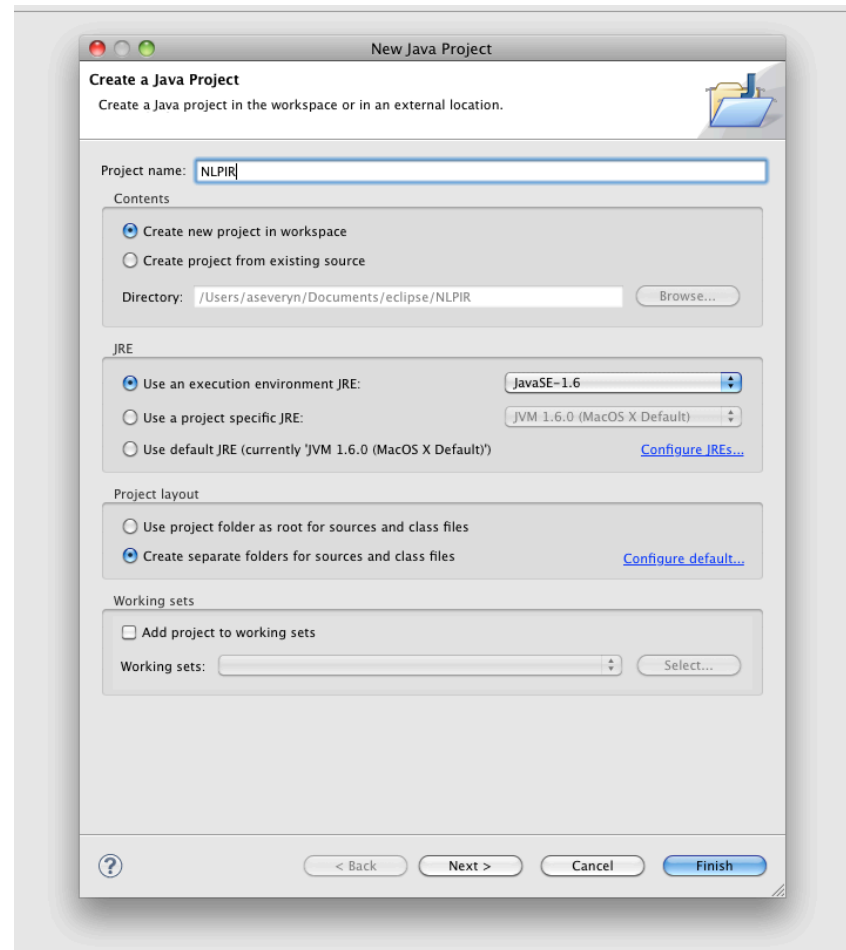
Setting up the project in Eclipse/Netbeans

- Create a new project NLPIR
- Drag **HelloLucene.java** src file to the project
- Go to project properties->Libraries
- Click on “Add External Jars...”
- Locate **lucene-core-3.5.0.jar**
- To enable documentation add path to the Javadoc

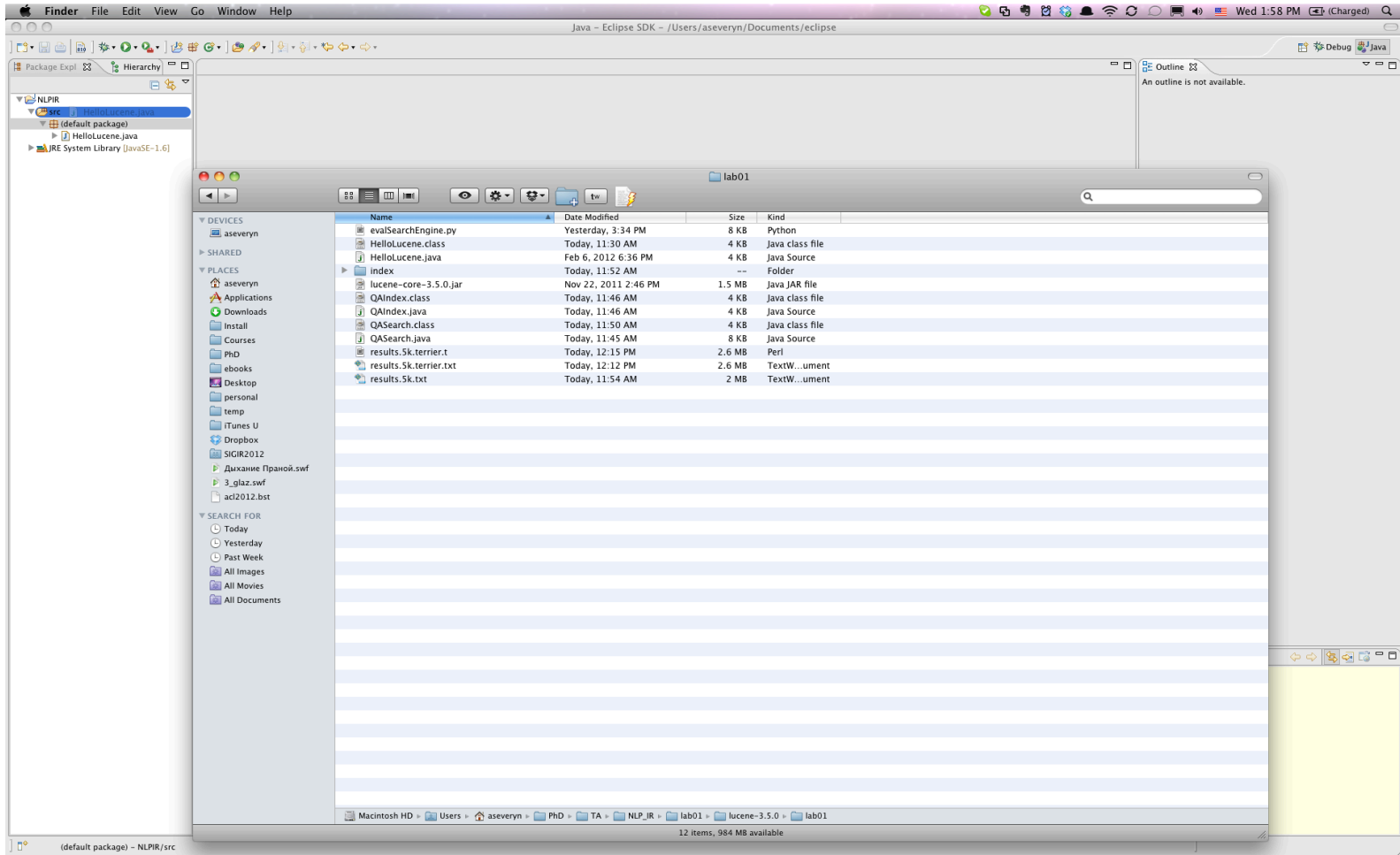
Create a new project



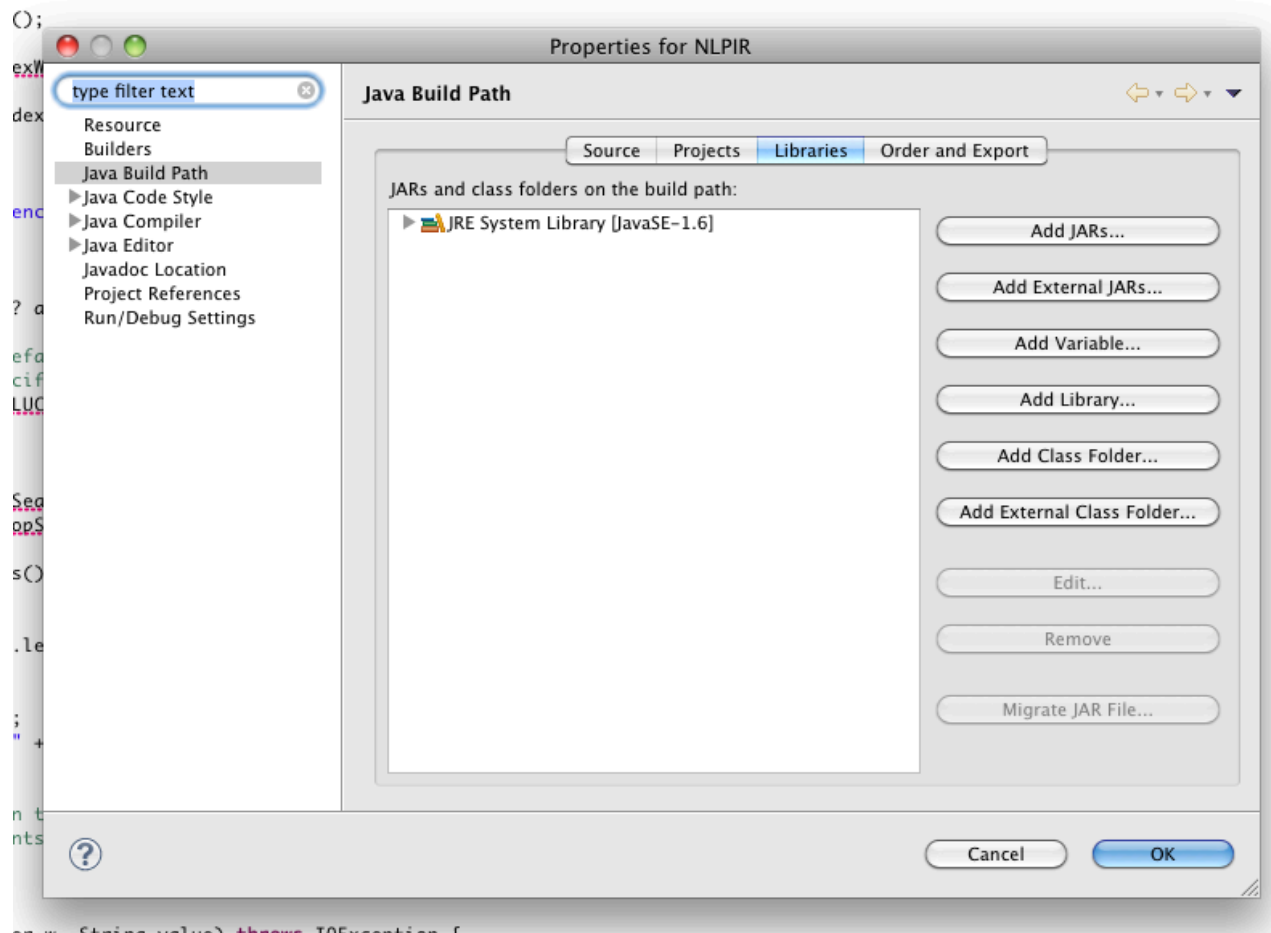
Name your project



Drag HelloLucene.jar to the src folder



Add lucene-core-3.5.0.jar



Run your first Lucene app!

```
w.addDocument(doc);
}
```

Problems Javadoc Declaration Console

<terminated> HelloLucene [Java Application] /System/Library/Frameworks/JavaVM.framework/Versions/1.4/Home/bin/java (Apr 4, 2012 2:02:07 PM)

Found 2 hits.

1. Lucene in Action
2. Lucene for Dummies

Adding Lucene documentation to the project

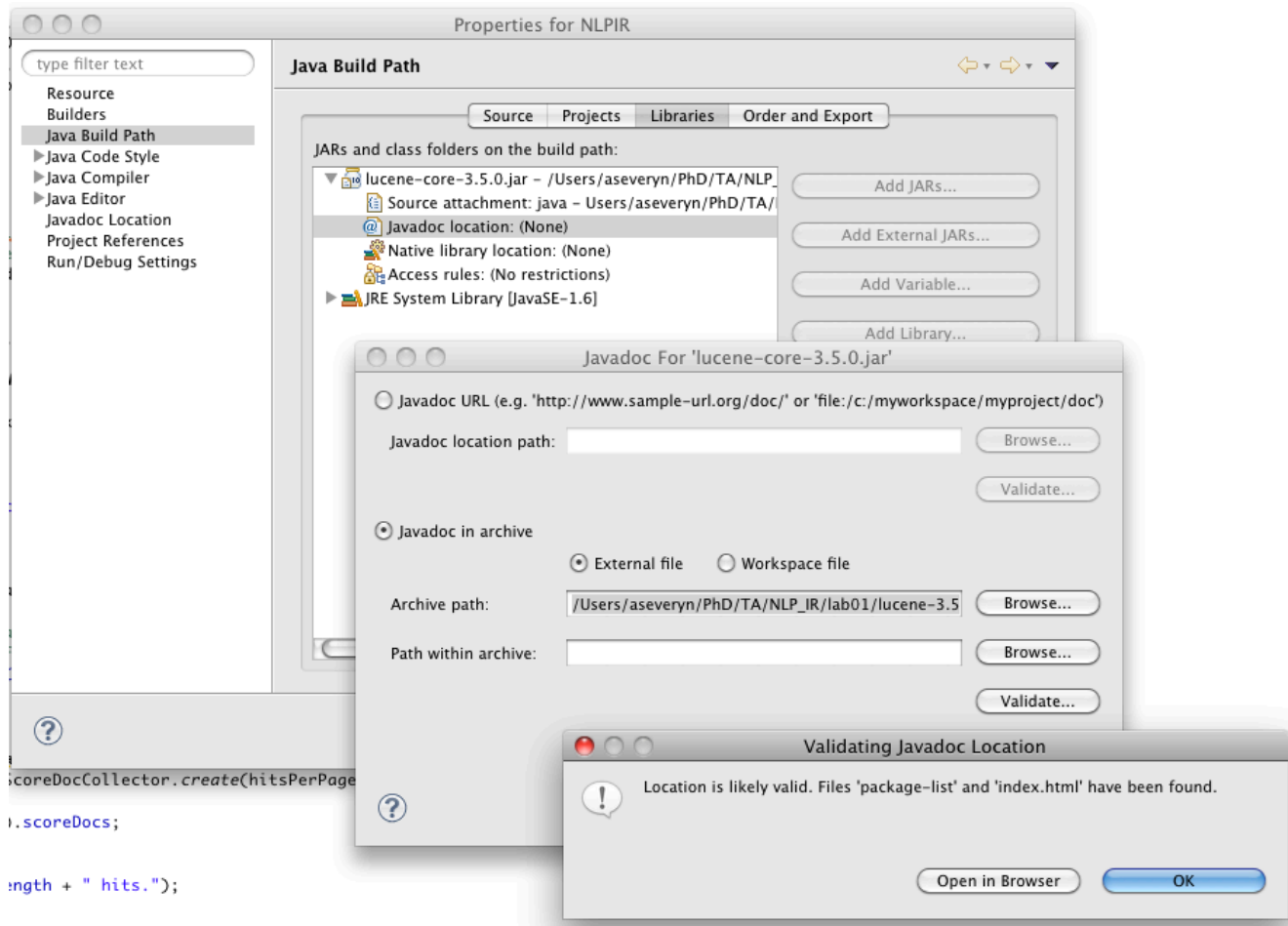
Go to Project properties->Libraries

Select lucene-core-3.5.0.jar

Select javadoc location

Locate lucene-core-3.5.0.jar

Adding javadoc



Setting the working environment

To be able to look at the Lucene internals:

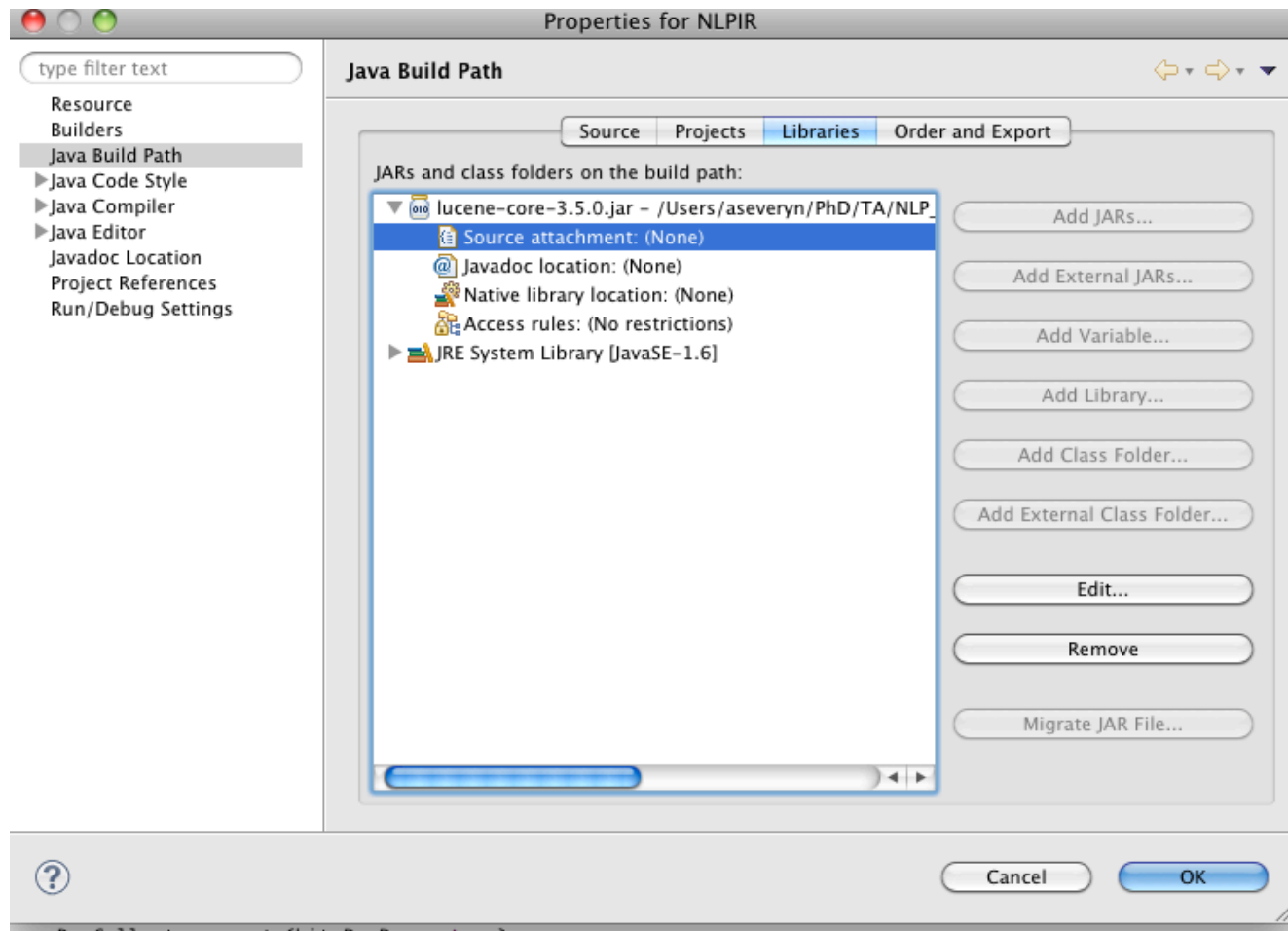
Go to Project properties->Libraries

Select lucene-core-3.5.0.jar

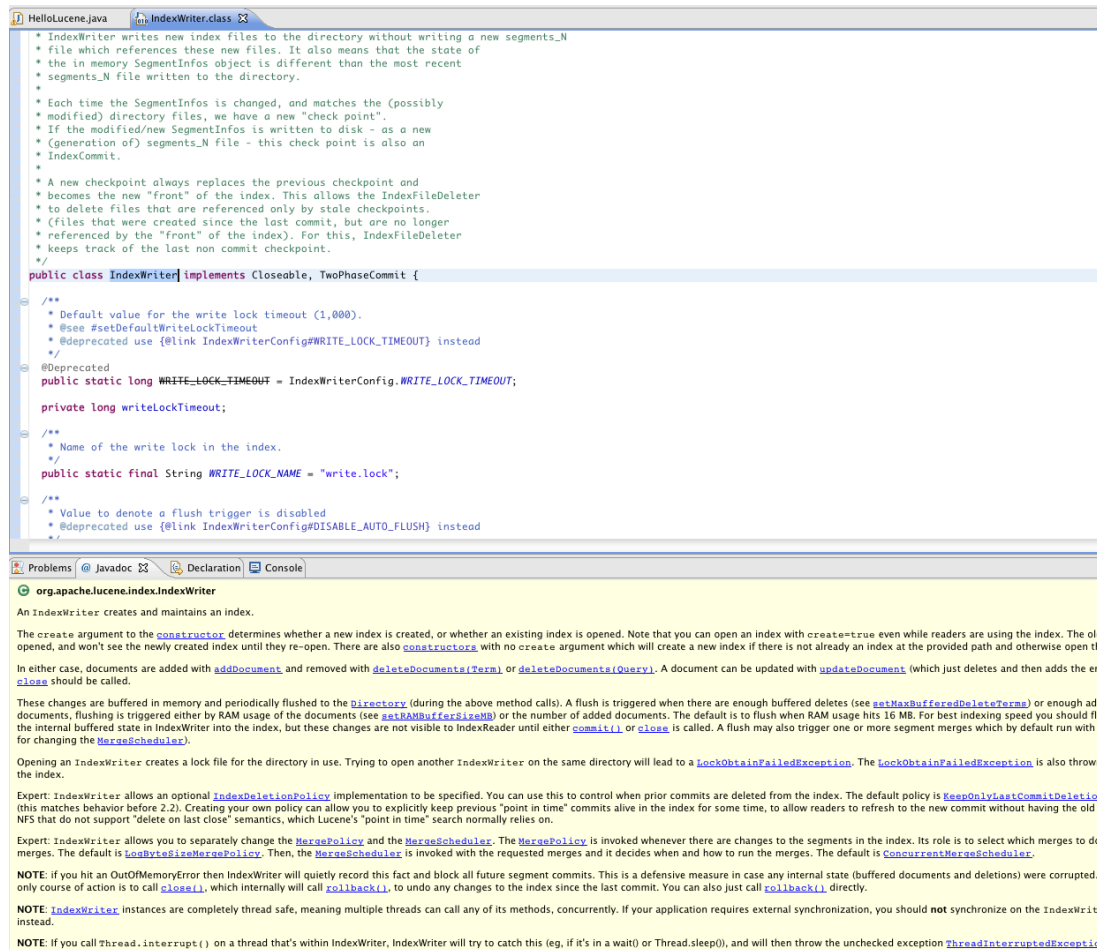
Select source attachment

Locate src folder

Adding sources to the JAR



Now we can examine the sources and documentation



The screenshot displays the source code of the `IndexWriter` class in the `org.apache.lucene.index` package. The code includes several Javadoc comments explaining the class's behavior, such as how it writes new index files, manages checkpoints, and handles flushes. Key annotations include `@Deprecated` for `WRITE_LOCK_TIMEOUT` and `DISABLE_AUTO_FLUSH`.

```

public class IndexWriter implements Closeable, TwoPhaseCommit {
    /**
     * Default value for the write lock timeout (1,000).
     * @see #setDefaultWriteLockTimeout
     * @deprecated use {@link IndexWriterConfig#WRITE_LOCK_TIMEOUT} instead
     */
    @Deprecated
    public static long WRITE_LOCK_TIMEOUT = IndexWriterConfig.WRITE_LOCK_TIMEOUT;

    private long writeLockTimeout;

    /**
     * Name of the write lock in the index.
     */
    public static final String WRITE_LOCK_NAME = "write.lock";

    /**
     * Value to denote a flush trigger is disabled
     * @deprecated use {@link IndexWriterConfig#DISABLE_AUTO_FLUSH} instead
     */
}

```

The Javadoc documentation below the code provides a detailed overview of the `IndexWriter` class:

org.apache.lucene.index.IndexWriter
 An `IndexWriter` creates and maintains an index.

The `create` argument to the `constructor` determines whether a new index is created, or whether an existing index is opened. Note that you can open an index with `create=true` even while readers are using the index. The old `n` opened, and won't see the newly created index until they re-open. There are also `constructors` with no `create` argument which will create a new index if there is not already an index at the provided path and otherwise open the `n`.

In either case, documents are added with `addDocument`, and removed with `deleteDocuments(Term)` or `deleteDocuments(Query)`. A document can be updated with `updateDocument` (which just deletes and then adds the entire `close` should be called.

These changes are buffered in memory and periodically flushed to the `Directory` (during the above method calls). A flush is triggered when there are enough buffered deletes (see `setMaxBufferedDeleteTerms`) or enough added documents, flushing is triggered either by RAM usage of the documents (see `setRAMBufferSizeMB`) or the number of added documents. The default is to flush when RAM usage hits 16 MB. For best indexing speed you should flush the internal buffered state in `IndexWriter` into the index, but these changes are not visible to `IndexReader` until either `commit()` or `close` is called. A flush may also trigger one or more segment merges which by default run with a `Thread` for changing the `MergeScheduler`.

Opening an `IndexWriter` creates a lock file for the directory in use. Trying to open another `IndexWriter` on the same directory will lead to a `LockObtainFailedException`. The `LockObtainFailedException` is also thrown if the index.

Expert: `IndexWriter` allows an optional `IndexDeletionPolicy` implementation to be specified. You can use this to control when prior commits are deleted from the index. The default policy is `KeepOnlyLastCommitDeletionPolicy` (this matches behavior before 2.2). Creating your own policy can allow you to explicitly keep previous "point in time" commits alive in the index for some time, to allow readers to refresh to the new commit without having the old `DFS` that do not support "delete on last close" semantics, which Lucene's "point in time" search normally relies on.

Expert: `IndexWriter` allows you to separately change the `MergePolicy` and the `MergeScheduler`. The `MergePolicy` is invoked whenever there are changes to the segments in the index. Its role is to select which merges to do, it merges. The default is `LogByteSizeMergePolicy`. Then, the `MergeScheduler` is invoked with the requested merges and it decides when and how to run the merges. The default is `ConcurrentMergeScheduler`.

NOTE: if you hit an `OutOfMemoryError` then `IndexWriter` will quietly record this fact and block all future segment commits. This is a defensive measure in case any internal state (buffered documents and deletions) were corrupted. An only course of action is to call `close()`, which internally will call `rollback()`, to undo any changes to the index since the last commit. You can also just call `rollback()` directly.

NOTE: `IndexWriter` instances are completely thread safe, meaning multiple threads can call any of its methods, concurrently. If your application requires external synchronization, you should **not** synchronize on the `IndexWriter` instead.

NOTE: if you call `Thread.interrupt()` on a thread that's within `IndexWriter`, `IndexWriter` will try to catch this (eg, if it's in a wait() or `Thread.sleep()`), and will then throw the unchecked exception `ThreadInterruptedException`.

Basic concepts in Lucene Search Engine

- Indexing
- Documents
- Fields
- Searching
- Queries

Indexing

- Instead of searching the text directly it searches the index
- Uses **inverted index** - inverts a document-centric data structure (document->words) to a keyword-centric data structure (word->documents)

Documents

- In Lucene, a **Document** is the unit of search and index.
- An index consists of one or more Documents.
- **Indexing** – adding Documents to an IndexWriter
- **Searching** - retrieving Documents from an index via an IndexSearcher.

Fields

- A Document consists of one or more Fields.
- A Field is simply a name-value pair.
- For example, a Field commonly found in applications is *title*.
- Indexing in Lucene thus involves creating Documents of one or more Fields, and adding these Documents to an IndexWriter.

Adding documents to the index

```
Directory index = new RAMDirectory();

IndexWriterConfig config = new IndexWriterConfig(Version.LUCENE_35, analyzer);

IndexWriter w = new IndexWriter(index, config);
addDoc(w, "Lucene in Action");
addDoc(w, "Lucene for Dummies");
addDoc(w, "Managing Gigabytes");
addDoc(w, "The Art of Computer Science");
w.close();
```

```
private static void addDoc(IndexWriter w, String value) throws IOException {
    Document doc = new Document();
    doc.add(new Field("title", value, Field.Store.YES, Field.Index.ANALYZED));
    w.addDocument(doc);
}
```

Queries

Lucene has its own mini-language for performing searches.

Allows the user to specify which field(s) to search on, which fields to give more weight to (boosting), the ability to perform boolean queries (AND, OR, NOT) and other functionality.

Query

We read the query from stdin, parse it and build a lucene Query out of it.

```
String querystr = args.length > 0 ? args[0] : "lucene";  
  
// the "title" arg specifies the default field to use  
// when no field is explicitly specified in the query.  
Query q = new QueryParser(Version.LUCENE_35, "title", analyzer).parse(querystr);
```

Searching

Searching requires an index to have already been built.

Very simple process:

- Create a **Query** (usually via a `QueryParser`)
- Handle this Query to an **IndexSearcher**
- Process a list of results

Searching

- Using the Query we create a Searcher to search the index.
- Then instantiate a TopScoreDocCollector to collect the top 10 scoring hits.

```
int hitsPerPage = 10;
IndexSearcher searcher = new IndexSearcher(index, true);
TopScoreDocCollector collector = TopScoreDocCollector.create(hitsPerPage, true);
searcher.search(q, collector);
ScoreDoc[] hits = collector.topDocs().scoreDocs;
```

Display of results

Now that we have results from our search, we display the results to the user.

```
System.out.println("Found " + hits.length + " hits.");
for(int i=0;i<hits.length;++i) {
    int docId = hits[i].doc;
    Document d = searcher.doc(docId);
    System.out.println((i + 1) + ". " + d.get("title"));
}
```

Let's get more practical

Build a Search Engine for answer passage retrieval in the Question Answering system

Use community QA site: Answerbag*

Use ~180k of automatically scraped question/answer pairs from over 20 categories

To reduce the amount of junk content focus only on professionally answered questions

QA system with AnswerBag data

Ask a question

Submit

HAPPENING NOW. The latest questions, answers and comments.



Waboo posted a new question:

What should a gentleman's reaction be if he sees camel toes?

4 SECONDS AGO



iwmit answered the question:

Richard Dawkins asks: "Is Religion Child Abuse?"

42 SECONDS AGO



carlosewers2012 answered the question:

Can you have a succesful prosperous life without being an engineer, programmer, doctor, or lawyer?

2 MINUTES AGO



dazed answered the question:

What do your parents watch that you watch as well?

3 MINUTES AGO



dazed answered the question:

When do you give random responses? Why?

4 MINUTES AGO



Drips answered the question:

What is your favourite purple fruit?

4 MINUTES AGO



dazed posted a new question:

[Ask a UK Lawyer Online](#)

A UK Lawyer Will Answer You Now!
Questions Answered Every 9 Seconds.
UK-Law.JustAnswer.com

[LL.M. in Real Estate](#)

Online or On-Campus Advanced Training
from Top Lawyers
www.nyls.edu/RealEstateLLM

[Projector Lamps on stock](#)

Lamps on stock 24 Hours Delivery Warranty
on each Lamp
www.projektoren-datenbank.com

AdChoices 

Follow us on Facebook!



Answerbag



18,162

Build high-quality QA corpus

Show:

Everything

Type ▾
1 of 2 selected

Community

Professionally Researched

Status ▾
1 of 2 selected


Unanswered

Answered

Categories ▾
All selected

[De-select All](#) | [Select All](#)

- Arts & Entertainment
- Business
- Computers
- Education
- Electronics
- Environmentalism
- Finance
- Food & Dining
- Games
- Health & Fitness
- Hobbies
- Home & Garden
- Kids
- Legal
- Life & Society



Can an antique drinking fountain bowl be refinish
asked by Answerbag Staff on May 21st, 2011 in Businesses and Co

What was the song played in the Quincy Jones vi Awards?
asked by Answerbag Staff on May 21st, 2011 in Opera and singers

Do all the flags have to be the same in high schoo
asked by Answerbag Staff on May 21st, 2011 in Modern dance | 2 a

When was Brother sewing machine's 35th annive
asked by Answerbag Staff on May 21st, 2011 in Businesses and Co







What is meaning of a multibar in a knitting machi

answers

T

Professionally researched answers

Answer Question

Share:  Email |  Facebook |  Twitter |  Other |  Like (OldCW wears The COAT of the Cosmos likes this.) |  Report

ANSWERS. 2 helpful answers below.

Sort answers by: Greatness / Likes

GREAT ANSWER

Professionally Researched. (What's this?)

by Lisa Fine on May 22nd, 2011



Antique drinking fountain bowls, if they are ceramic, can be restored to their original smoothness and luster. Almost all vintage drinking fountain bowls are ceramic. There are companies that specialize in reglazing the ceramic, bringing it back to a near-new condition.

References:

[Dea Bath: Antique Drinking Fountains](#)

Setting up QA example

Download the code from:

<http://disi.unitn.it/~severyn/NLPIR.2012/lab01/qa.tar.gz>

E.g. try the following in your terminal

```
$ wget http://disi.unitn.it/~severyn/NLPIR.2012/lab01/qa.tar.gz
$ tar xvfz qa.tar.gz
answers.txt
evalSearchEngine.py
QAIndex.java
QASearch.java
questions.5k.txt
```

Example of the QA pair: 2503031

Q: What software was used in making the special effects for "Pirates of the Caribbean?"

A: The software used in making the effects for the "Pirates of the Caribbean" films was the Electric Image Animation Software. Made by the EI Technology Group, the software runs on both Macintosh and Windows operating systems.

Let's create an index of our collection

Compile:

```
$ javac -cp .:lucene-core-3.5.0.jar QAIndex.java
```

Index QA collection:

```
$ java -cp .:lucene-core-3.5.0.jar QAIndex index  
answers.txt
```

OR:

```
$ export CLASSPATH=.:lucene-core-3.5.0.jar
```

```
$ javac QAIndex.java
```

```
$ java QAIndex index answers.txt
```

Now we can perform search

Compile:

```
$ javac -cp .:lucene-core-3.5.0.jar QASearch.java
```

Search:

```
$ java -cp .:lucene-core-3.5.0.jar QASearch index  
questions.5k.txt 15 > results.5k.txt
```

OR:

```
$ export CLASSPATH=.:lucene-core-3.5.0.jar
```

```
$ javac QASearch.java
```

```
$ java QASearch index questions.5k.txt 15 >  
results.5k.txt
```

Evaluate the results

```
$ python evalSearchEngine.py results.5k.txt
```

```
MRR^: 66.43
```

#:	REC-1	ACC
01:	57.30	57.30
02:	67.94	33.97
03:	73.12	24.37
04:	76.00	19.00
05:	78.22	15.64
06:	79.72	13.29
07:	80.70	11.53
08:	81.88	10.23
09:	82.64	9.18
10:	83.58	8.36

The baseline is very naive

You are encouraged to try at least some of this ideas to improve the SE results:

- No stemming or stop words removal
- Lucene is using a simple weighting model to score documents (Cosine similarity)
- **Next time we're going to build a re-ranker for our QA system to improve the SE results**

Homework

Play around with indexing and searching

Try out state of the art weighting models, e.g.
Okapi BM25 and BM25F.

Add your own evaluation metrics