Introducing Lucene

presented by
Erik Hatcher

O’Reilly Open Source Convention
July 7 - 11, 2003
What is Lucene

- High-performance, full-featured text search engine
- 100% Java
- http://jakarta.apache.org/lucene
- It is an API only, but...
Goals

- Learn what Lucene is, see how easy it is to use
- Gain appreciation for what is hidden underneath
- Benefit from some best practices and experience
Powered by

- BobDylan.com
- jGuru
- ZOE
- Jive Forums
- Scarab
- JavaDevWithAnt
- BlogScene
- ... and many others
• An index contains a sequence of documents
• A document is a sequence of fields
• A field is a named sequence of terms
• A term is a text string, keyed by field name
Simple Examples

• Because for the developer, Lucene is, in fact, quite simple
• Keep this simplicity in mind as we delve under the hood later
Quiz: what is a document?
String[] docs = {
    "doc1 - present!",
    "doc2 is right here",
    "and do not forget lil ol doc3"
};
Directory directory = new RAMDirectory();
Analyzer analyzer = new StandardAnalyzer();

IndexWriter writer =
   new IndexWriter(directory,
                  analyzer, true);

for (int j = 0; j < docs.length; j++) {
   Document d = new Document();
   d.add(Field.Text("contents", docs[j]));
   writer.addDocument(d);
}

writer.close();
Searching

• ... is where Lucene really shines
Searching

Searcher searcher =
    new IndexSearcher(directory);

Query query =
    new TermQuery(
        new Term("contents", "doc1")
    );

Hits hits = searcher.search(query);
System.out.println("doc1 hits = " +
    hits.length());

searcher.close();

doc1 hits = 1
Inverted Index

- Stores statistics about terms
- For a term, the documents that contain it can be quickly retrieved
Index Structure

• An index is composed of segments
• Segments are created as documents are added, and can be merged
• A segment is a fully independent index
• Searching can (and often does) span multiple segments
• Each segment index contains:
  – field names, stored field values, term dictionary, frequency data, proximity data, normalization factors, deleted documents
Fields

- Stored
  - Non-inverted, content retrievable
- Indexed
  - Inverted, searchable
- Tokenized
  - Broken into tokens
# Field object

<table>
<thead>
<tr>
<th>method</th>
<th>Stored?</th>
<th>Indexed?</th>
<th>Tokenized?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyword</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text (w/Reader)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text (w/String)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UnIndexed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UnStored</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Field examples

doc.add(Field.Text("title", title));
doc.add(Field.UnIndexed("body", body));

doc.add(Field.Keyword("category", category));
doc.add(Field.Keyword("path", category));
doc.add(Field.Keyword("path", ymd));

doc.add(Field.UnStored("contents", title + " " + bodyText + " " + category + " " + docname));
Analyzers

- Tokenize text
- Can be field-specific, but not generally
- Used during both indexing and searching
- Choosing (or writing) the right Analyzer(s) is one of the most important decision to make with Lucene!
Analyzer Demo

- Tokenize a phrase
StandardAnalyzer Features

- Sequences of letters and digits
- Interior apostrophe's (e.g. o'clock)
- Acroynms (e.g. H.P.)
- Companies (e.g. AT&T)
- E-mail addresses
- Hostname (e.g. jakarta.apache.org)
- Serial numbers, IP addresses, floating point numbers (e.g. R2D2 and 192.168.1.1)
Other Analyzers

- GermanAnalyzer
- RussianAnalyzer
- SnowballAnalyzer (currently in "sandbox")
  - Danish, Dutch, English, Finnish, French, German, Italian, Lovins, Norwegian, Porter, Portuguese, Russian, Spanish, Swedish
Searching

- Instantiate IndexSearcher
- Call one of the search methods
  - search(Query)
  - search(Query, Filter)
  - search(Query, HitCollector)
- Process the hits
Query object

• Subclasses include:
  – TermQuery, RangeQuery, PrefixQuery, PhraseQuery, WildcardQuery, FuzzyQuery, and BooleanQuery

• Use these subclasses for code-generated queries

• QueryParser is useful for human-entered queries
QueryParser

- Provides powerful expression syntax
  - Boolean operators (+, AND, OR)
  - Field selection (title:foo)
  - Wildcard support (blog*, bl?g, b*g)
  - Fuzzy terms (socks~) using Levenshtein algorithm
  - Proximity ("blog lucene"~5)
  - Boosting (ant^4 blog)
  - Grouping ((lucene OR blog) AND jakarta)
  - Range (20020101 TO 20021231) - lexicographic
  - Special character escaping (+ - && || ( ) { } [ ] ^ " ~ * ? : \)
Analyzer analyzer = new StandardAnalyzer();

String q = "+blog -radio +title: lucene";

Query query =
    QueryParser.parse(q, // query
    "contents", // field
    analyzer);

Hits hits = searcher.search(query);
Deleting documents

- Documents have a document id
- id can change when index is optimized
- Do not rely on document id's between sessions (essentially per IndexReader instantiation)

```java
IndexReader reader = ....

// By document id
reader.delete(docId);

// or by Term
int numDeleted =
    reader.delete(new Term("contents", "xyz"));
```
A few FAQs

• *Is Lucene thread-safe?*  
  IndexSearcher and IndexWriter are thread-safe. QueryParser is not.

• *Do document id's change?*  
  Yes.

• *How do I update a document in an index?*  
  Delete it and re-add it.

• *Can Lucene index documents of type ...?*  
  Can you extract the text from files of that type?

• *How do I deal with dates?*  
  Lexicographically
Sandbox

- CVS: jakarta-lucene-sandbox
- LARM - Lucene Advanced Retrieval Machine
- Others
  - XML-Indexing-Demo
  - searchbean
  - parsers - only PDF currently
  - javascript - browser query creation/validation utils
  - ant - create/update index during build process
  - Etc... check repository before writing your own

- Batteries not included
Third-party

- PDFBox - http://www.pdfbox.org
- POI - http://jakarta.apache.org/poi
- TextMining - http://www.textmining.org
  - wraps PDFBox and POI
Interesting Usages

• Michaels.com - searching for colors
• JavaDevWithAnt - documentation search engine
• ZOE - "intertwangled e-mail"
• BlogScene - flattened datastore
Session Summary

- Lucene is a fast, flexible, Java-based search engine
- It is an API, with a few built-in conveniences, but the developer must do some work
- It's very simple to realize powerful capabilities
- Lots of tips, examples, and layers above the API available online
Resources

• Lucene official site
  http://jakarta.apache.org/lucene

• "Search-Enable Your Application with Lucene" by Craig Walls
  Java Developers Journal, December 2002

• JavaDevWithAnt
  http://www.ehatchersolutions.com/JavaDevWithAnt
  from Java Development with Ant (Manning)

• jGuru Lucene FAQ
  http://www.jguru.com/faq/Lucene