Einführung in Lucene

Dr. Christian Herta

April, 2009

Outline

Lernziele - Inhalt

- Zweck von Lucene
- Wesentliche Klassen von Lucene und ihr Zusammenspiel
- Grundlegende Benutzung von Lucene zur Suche und Indizierung

Outline

Was ist Lucene?

- Information (Text) Retrieval Library für Suche und Indizierung
  - d.h. (JAVA)-API - kein Anwendungsprogramm
  - aber es gibt Anwendungen basierend auf Lucene, wie z.B. Solr, nutch
  - kein Spidering und Dokumentenverarbeitung
- Apache Jakarta Projekt
- Apache Software License

Outline

Wichtigen Klassen für die Indizierung

- `IndexWriter`
- `Directory`
- `Analyzer`
- `Document`
- `Field`

IndexWriter

- Zentrale Komponente für die Indizierung
- Schreibzugriff (write access) auf den Index
- Erzeugt neuen Index
- Fügt Dokumente zum Index hinzu

Directory

- Repräsentiert die Lokalisierung (Speicherort) des Index
- Abstrakte Klasse
- Unterklassen sind als Implementierung bei Lucene dabei, wie:
  - `FSDirectory`: Speicherung des Indexes auf Festplatte
  - `RAMDirectory`: In Memory Repräsentation

Analyzer

- Vorverarbeitung des zu indizierenden Textes
- Abstrakte Klasse
- Etliche konkrete Implementationen in Lucene enthalten
- Analyseschritte
  - Tokenisierung
  - Stemming
  - Beseitigung von Interpunktion und Sonderzeichen
  - Beseitigung von Stopwörtern
- Sprachabhängig

Initialisierung des IndexWriter

Listing 1: Initialisierung des IndexWriter

```
1 File f;
2 //... init f
3 IndexWriter writer =
4       new IndexWriter
5       (f,
6        new StandardAnalyzer(),
7        true, //create
8        IndexWriter.MaxFields.UNLIMITED);
9 // last parameter sets the maximum field lenght
```

Document

- `Document` ist die Entität, die indiziert und gefunden wird
- `Document` ist in Felder strukturiert
- Lucene indiziert Text
- aber auch numerische Werte (ab Lucene 2.9 auch native numerische Werte in `NumericField`)

Field

- Repräsentation der Felder des `Document`
- Felder können verschiedene Eigenschaften haben
  - Indiziert
  - Analysiert (wie Tokenisiert)
  - Stored

(virtuelles) Dokument erzeugen

Listing 2: Dokument erzeugen und füllen

```
1  File f;
2  ...// initialization of f
3  Document doc = new Document();
4  doc.add(new Field("content"),
5          new FileReader(f),
6          Field.Store.NO,
7          Field.Index.ANALYZED));
8  doc.add(new Field("filename"),
9          f.getCanonicalPath(),
10         Field.Store.YES,
11         Field.Index.NOT_ANALYZED);
```

Dokument zum Index hinzufügen

Listing 3: Dokumente indizieren

```
1 //for all Document doc
2 ..
3 writer.addDocument(doc);
4 ..
5
6 // writer is instance of IndexWriter
7 writer.close();
```

Outline

Wichtigen Klassen

- `IndexSearcher`
- `Term`
- `Query`
- `TermQuery`
- `TopDocs`

IndexSearcher

- Zentrale Komponente für den Zugriff auf den Index bei der Suche
- Read-only Mode

Term

- `Term`
- Paar
  - Feldname
  - Wert(Inhalt) des Feldes

Query

- `Query` ist abstrakte Klasse
- Verschiedene konkrete Implementierungen
  - `TermQuery`
  - `BooleanQuery`
  - `PhraseQuery`
  - etc.

TopDocs

- `TopDocs` ist Container für die Ergebnisse der Suche
- Für die Ergebnisse erhält man
  - `docID`
  - Such-Score (float)

Code-Schnipsel: Suche mit Lucene

Listing 4: Wichtige Klassen für die Suche

```
1 IndexSearcher searcher =
2     new IndexSearcher("/tmp/index");
3 Query query =
4     new TermQuery
5     (new Term("contents", "Lucene"));
6 TopDocs hits = searcher.search(query, 10);
7 searcher.close();
```