



□ □ □

□

!

!

并不是不分享代码 <----- > 而是直接上代码 展示效果非常差(所以直接看图吧)!

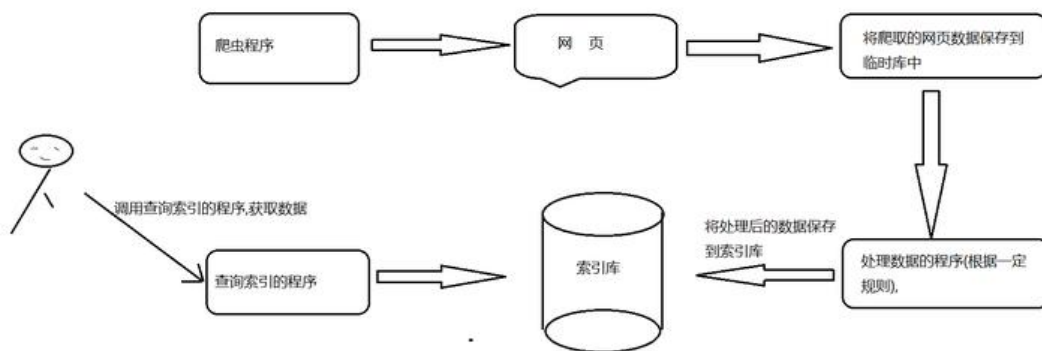
谢谢~~~

1. 搜索引擎

1.1 什么是搜索引擎

!

1.2 搜索引擎基本的运行原理



1.3 原始数据库查询的缺陷

1.4 倒排索引技术



□

2. Lucene

2.2 使用Lucene如何构建索引

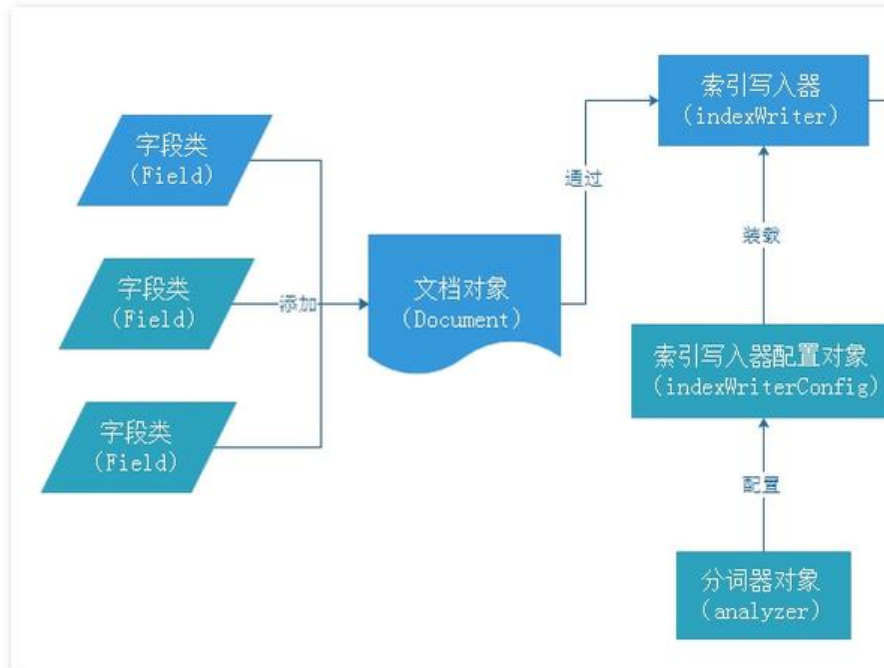
2.2.1 第一步: 导入相关的jar包(pom依赖)

```
<artifactId> lucene-core</artifactId>
<version> 4.10.2</version>
</dependency>
<!--query 查询相关的包-->
<dependency>
  <groupId> org.apache.lucene</groupId>
  <artifactId> lucene-queries</artifactId>
  <version> 4.10.2</version>
</dependency>
<!--用于测试-->
<dependency>
  <groupId> org.apache.lucene</groupId>
  <artifactId> lucene-test-framework</artifactId>
  <version> 4.10.2</version>
</dependency>
```

Dependencies:

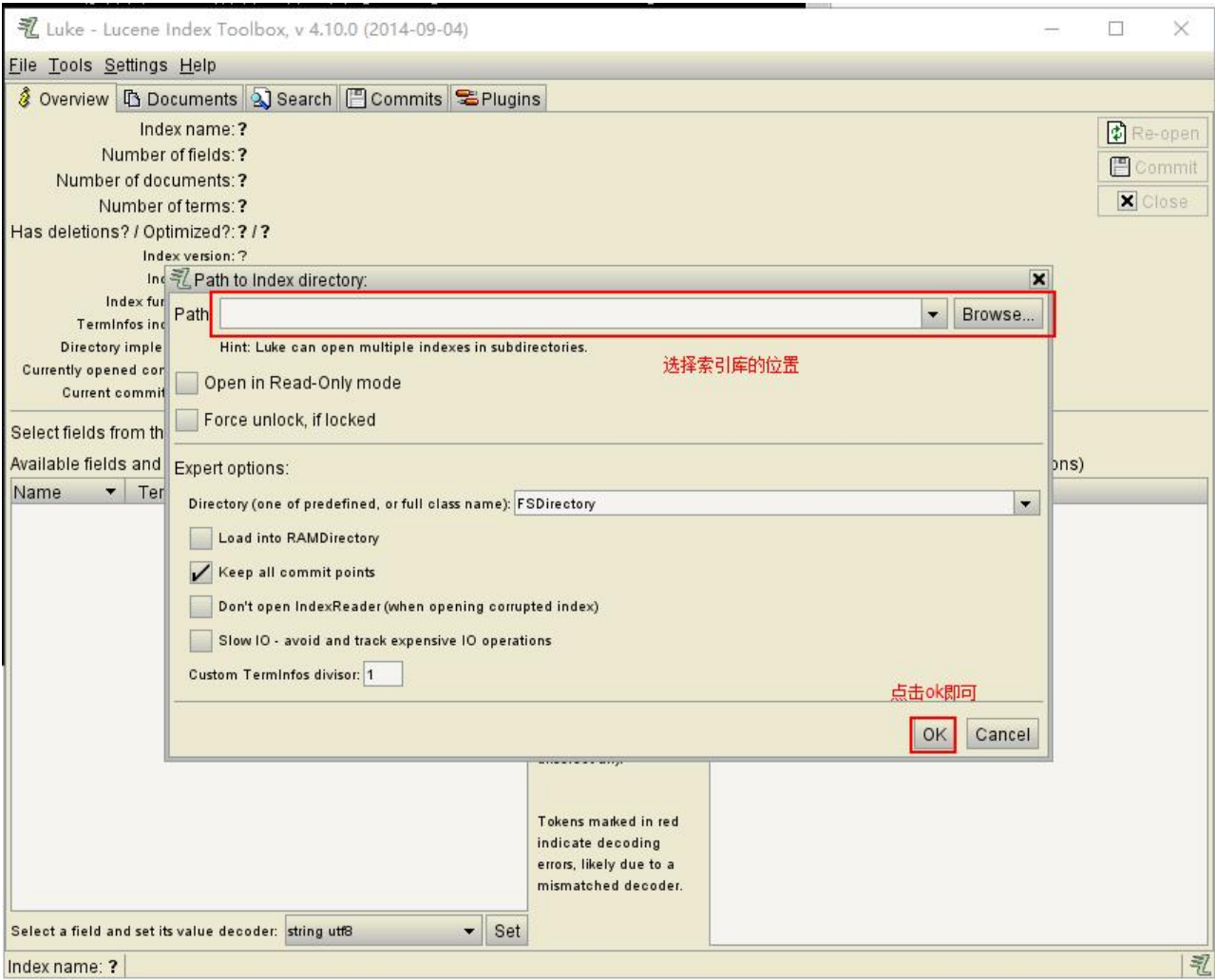
- org.apache.lucene:lucene-core:4.10.2
- org.apache.lucene:lucene-queries:4.10.2
 - org.apache.lucene:lucene-core:4.10.2 (omitted for duplicate)
- org.apache.lucene:lucene-test-framework:4.10.2
 - org.apache.lucene:lucene-codecs:4.10.2
 - org.apache.lucene:lucene-core:4.10.2 (omitted for duplicate)
 - com.carrotsearch.randomizedtesting:junit4-ant:2.1.6
 - com.carrotsearch.randomizedtesting:randomizedtesting-runner:2.1.6
 - junit:junit:4.10
 - org.apache.ant:ant:1.8.2
- org.apache.lucene:lucene-analyzers-common:4.10.2
 - org.apache.lucene:lucene-core:4.10.2 (omitted for duplicate)
- org.apache.lucene:lucene-queryparser:4.10.2
 - org.apache.lucene:lucene-core:4.10.2 (omitted for duplicate)
 - org.apache.lucene:lucene-queries:4.10.2 (omitted for duplicate)
 - org.apache.lucene:lucene-sandbox:4.10.2
- org.apache.lucene:lucene-highlighter:4.10.2
 - org.apache.lucene:lucene-core:4.10.2 (omitted for duplicate)
 - org.apache.lucene:lucene-memory:4.10.2
- org.apache.lucene:lucene-queries:4.10.2 (omitted for duplicate)

2.2.2 第二步: 书写写入索引的代码



```
//1. 需要创建 indexwriter对象
//1.1 创建 索引库
FSDirectory directory = FSDirectory.open(new File( pathname: "H:\\test"));
//1.2 创建 写入器配置对象: 参数1 版本号, 参数2 分词器
IndexWriterConfig config = new IndexWriterConfig(Version.LATEST,new StandardAnalyzer());
IndexWriter indexWriter = new IndexWriter(directory,config);
//2. 写入文档
//2.1 创建文档对象
Document doc = new Document();
//2.2 添加文档的属性
doc.add(new StringField( name: "title", value: "Lucene介绍", Field.Store.YES));
doc.add(new IntField( name: "id", value: 1, Field.Store.YES));
doc.add(new TextField( name: "content", value: "Lucene是一个全文检索的工具包", Field.Store.YES));
indexWriter.addDocument(doc);
//3. 提交数据
indexWriter.commit();
//4. 释放资源
indexWriter.close();
directory.close();
```

2.3 索引查看工具



Index name: **H:test** 索引库的位置

Number of fields: **3** 字段的个数

Number of documents: **5** 文档的个数

Number of terms: **40** 词条的个数

Has deletions? / Optimized?: **No / No**

Index version: 7

Index format: Lucene 4.x, segment ver.:3

Index functionality: flexible, codec-specific

TermInfos index divisor: 1

Directory implementation: org.apache.lucene.store.MMapDirectory

Currently opened commit point: segments_3 (generation=3, segs=3)

Current commit user data: --

- Re-open
- Commit
- Close

Select fields from the list below, and press button to view top terms in these fields. 显示具体的分词效果

Available fields and term counts per field:

Name	Term count	%	Decoder
content	31	77.5 %	string utf8
id	6	15 %	string utf8
title	3	7.5 %	string utf8

显示索引库中的字段列表, 以及每个字段一共有多少词条

Show top terms >>

Number of top terms:

50

Hint: use Shift-Click to select ranges, or Ctrl-Click to select multiple fields (or unselect all).

Tokens marked in red indicate decoding errors, likely due to a mismatched decoder.

Top ranking terms. (Right-click for more options)

Rank	Freq	Field	Text
1	5	id	x□
2	5	id	p□
3	5	id	h□
4	4	content	—
5	4	content	是
6	3	title	Lucene介绍
7	3	content	个
8	3	content	检索
9	3	content	工具
10	3	content	lucene
11	3	content	包
12	3	content	一个
13	3	id	'□□
14	3	content	工具包
15	3	content	全文
16	2	content	碉堡了

Select a field and set its value decoder: string utf8 Set

Index name: **H:test** Index successfully open.

Luke - Lucene Index Toolbox, v 4.10.0 (2014-09-04)

File Tools Settings Help 文档查看窗口

Overview Documents Search Commits Plugins

Browse by document number:
 Doc. #: 0 ← 1 → 4 文档编号
 Add Reconstruct & Edit
 More like this... 默认从0开始

Browse by term:
 (Hint: enter a substring and press Next to start at the nearest term).
 First Term Term: content → Next Term
 Decoded value:

Browse documents with this term (0 documents)
 Document: ? of ? First Doc → Next Doc Show All Docs Delete All Docs
 Term freq in this doc: ? Show Positions

Doc #: 1 **Flags:** I - Indexed (docs,freqs,pos,offsets) P - Payloads S - Stored; V - Term Vector
 B - Binary; Nbx - Norms (type/precision); #bx - Numeric (type/precision); Dbx - DocValues (type/precision)

Field	IdfpoPSVBNbx#bxDbx	Norm	Value
content	ldfp--S--Nnum-----	0.25	Lucene是一个全文检索的工具包,碉堡了
id	Id---S-----#i32----	---	1
title	Id---S-----	---	Lucene介绍

显示文档各个字段的原始的内容

Selected field: TV Show Examine norm Save Copy text to Clipboard: Selected fields Complete document
 Index name: H:test

2.4 API详解

Directory indexWriterConfig

Directory

IndexWriterConfig

```

// 参数值: APPEND CREATE CREATE_OR_APPEND
/**
 * APPEND: 表示追加, 如果索引库存在, 就会向索引库中追加数据, 如果索引库不存在, 直接报错
 *
 * CREATE: 表示创建, 不管索引库有没有, 每一次都是重新创建一个新的索引库
 *
 * CREATE_OR_APPEND: 如果索引库有, 就会追加, 如果没有 就会创建索引库
 * 默认值也是 CREATE_OR_APPEND
 */
config.setOpenMode(IndexWriterConfig.OpenMode.CREATE_OR_APPEND);

```

□

□

Field类	数据类型	Analyzed 是否分析	Indexed 是否索引	Stored 是否存储	说明
StringField(FieldName, FieldValue, Store.YES))	字符串	N	Y	Y或N	这个Field用来构建一个字符串Field, 但是不会进行分析, 会将整个串存储在索引中, 比如(订单号,姓名等)是否存储在文档中用Store.YES或Store.NO决定
LongField(FieldName, FieldValue, Store.YES)	Long型	Y	Y	Y或N	这个Field用来构建一个Long数字型Field, 进行分析和索引, 比如(价格)是否存储在文档中用Store.YES或Store.NO决定
StoredField(FieldName, FieldValue)	重载方法, 支持多种类型	N	N	Y	这个Field用来构建不同类型Field不分析, 不索引, 但要Field存储在文档中
TextField(FieldName, FieldValue, Store.NO)或 TextField(FieldName, reader)	字符串 或流	Y	Y	Y或N	如果是一个Reader, lucene猜测内容比较多, 会采用Unstored的策略.

paoding : Lucene中文分词“庖丁解牛” Paoding Analysis
在PIII 1G内存个人机器上, 1秒可准确分词 100万 汉字
<http://code.google.com/p/paoding/>
imdict : imdict智能词典所采用的智能中文分词程序
483.64 (字节/秒), 259517(汉字/秒)
<http://code.google.com/p/imdict-chinese-analyzer/>
mmseg4j : 用 Chih-Hao Tsai 的 MMSeg 算法实现的中文分词器
complex 1200kb/s左右, simple 1900kb/s左右
<http://code.google.com/p/mmseg4j/>
ik : 采用了特有的“正向迭代最细粒度切分算法”, 多子处理器分析模式
具有50万字/秒的高速处理能力
<http://code.google.com/p/ik-analyzer/>

2.5 集成IK分词器



The screenshot shows the Google Code project page for 'ik-analyzer'. The page title is 'ik-analyzer' with the subtitle 'java开源中文分词器'. The main content area contains a description of the project and a list of features for the 2012 version. The features are:

1. 采用了特有的“正向迭代最细粒度切分算法”, 支持细粒度和智能分词两种切分模式;
2. 在系统环境: Core2 i7 3.4G双核, 4G内存, window 7 64位, Sun JDK 1.6_29 64位 普通pc环境测试, IK2012具有160万字/秒 (3000KB/S) 的高速处理能力。
3. 2012版本的智能分词模式支持简单的分词排歧义处理和数量词合并输出。
4. 采用了多子处理器分析模式, 支持: 英文字母、数字、中文词汇等分词处理, 兼容韩文、日文字符
5. 优化的词典存储, 更小的内存占用。支持用户词典扩展定义。特别的, 在2012版本, 词典支持中文, 英文, 数字混合词语。

ikanalyzer-4-support

ikanalyzer-4-support

Project Home Downloads Wiki Issues Source Export to GitHub

Summary People

Project Information

★ Starred by 0 users
[Project feeds](#)

Code license
[MIT License](#)

Labels
ikanalyzer, lucene, solr,
ikanalyzer4, ikanalyzer4.0,
ikanalyzer5.0

Members
[qibaoyuan@gmail.com](#)

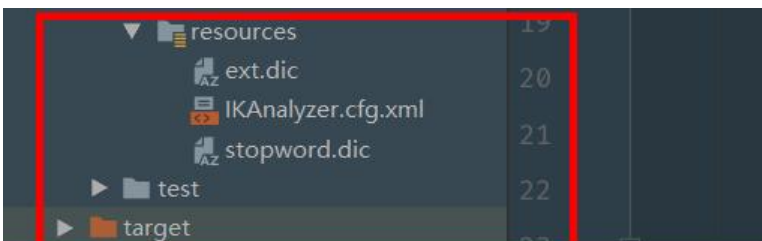
这个工程是为了支持在lucene4上进行分词调用，代码全部基于IKAnalyzer。 This Project add support for Lucene 4.0 beta and Lucene 5.0 trunk. The project is based on <http://code.google.com/p/ik-analyzer/>

Donate for the modifier 【支付宝 [qibaoyuan@126.com](#)(齐保元) PayPal: qibaoyuan@126.com】

IK Analyzer是一个开源的，基于java语言开发的轻量级的中文分词工具包。从2006年12月推出1.0版开始，IKAnalyzer已经推出了4个大版本。最初，它是以开源项目Luence为应用主体的，结合词典分词和文法分析算法的中文分词组件。从3.0版本开始，IK发展为面向Java的公用分词组件，独立于Lucene项目，同时提供了对Lucene的默认优化实现。在2012版本中，IK实现了简单的分词歧义排除算法，标志着IK分词器从单纯的词典分词向模拟语义分词衍化。IK Analyzer 2012特性：

- 1.采用了特有的“正向迭代最细粒度切分算法”，支持细粒度和智能分词两种切分模式；
- 2.在系统环境：Core2 i7 3.4G双核，4G内存，window 7 64位， Sun JDK 1.6_29 64位 普通pc环境测试，IK2012具有160万字/秒（3000KB/S）的高速处理能力。
- 3.2012版本的智能分词模式支持简单的分词排歧义处理和数量词合并输出。
- 4.采用了多子处理器分析模式，支持：英文字母、数字、中文词汇等分词处理，兼容韩文、日文字符
- 5.优化的词典存储，更小的内存占用。支持用户词典扩展定义。特别的，在2012版本，词典支持中文，英文，数字混合词语。

□



2.6 查询索引

2.6.1 查询入门:

2.7 查询相关API详解

2.8 多样化查询(特殊查询)

2.8.1 词条查询: TermQuery

```

// 词条查询
public void termQuery() throws Exception {
    //创建词条对象
    //注意：词条是不可分割的，词条可以是一个字，也可以是一句话
    //使用场景：主要是针对的是不可分割的字段，例如id
    //由于其不可再分，可以搜索 全文，但是不能搜索 全文检索
    TermQuery termQuery = new TermQuery(new Term( fld: "content", text: "全文"));
    query(termQuery);
}

```

2.8.2通配符查询: WildcardQuery

```

@Test
// 通配符查询
public void wildcardQuery()throws Exception{
    //通配符：
    //*: 代表多个字符
    //?: 代表一个占位符
    WildcardQuery wildcardQuery = new WildcardQuery(new Term( fld: "content", text: "?uce*"));
    query(wildcardQuery);
}

```

2.8.3模糊查询: FuzzyQuery

```

@Test
//模糊查询...fuzzQuery
public void fuzzQuery ()throws Exception{
    /**
     * 模糊查询:
     * 指的是通过替换，补位，移动 能够在二次切换内查询数据即可返回
     * 参数1: term 指定查询的字段和内容
     * 参数2: int n 表示最大编辑的次数 最大2
     */
    FuzzyQuery fuzzyQuery = new FuzzyQuery(new Term( fld: "content", text: "lucene"), maxEdits: 1);
    query(fuzzyQuery);
}

```

2.8.4数值范围查询: NumericRangeQuery

```

@Test
public void numericRangeQuery() throws Exception {
    /**
     * 获取NumericRangeQuery的方式:
     * 通过提供的静态方法获取:
     *     NumericRangeQuery.newIntRange()
     *     NumericRangeQuery.newFloatRange()
     *     NumericRangeQuery.newDoubleRange()
     *     NumericRangeQuery.newLongRange()
     * 数值范围查询:
     * 参数1: 指定要查询的字段
     * 参数2: 指定要查询的开始值
     * 参数3: 指定要查询的结束值
     * 参数4: 是否包含开始
     * 参数5: 是否包含结束
     */

    NumericRangeQuery numericRangeQuery = NumericRangeQuery.newIntRange( field: "id", min: 2, max: 4, minInclusive: false,
        maxInclusive: false);
    query(numericRangeQuery);
}

```

2.8.5 组合查询: BooleanQuery

```

@Test
public void testBooleanQuery() throws Exception {

    Query query1 = NumericRangeQuery.newLongRange( field: "id", min: 8L, max: 4L, minInclusive: true, maxInclusive: true);
    Query query2 = NumericRangeQuery.newLongRange( field: "id", min: 0L, max: 3L, minInclusive: true, maxInclusive: true);
    // boolean查询本身没有查询条件, 它可以组合其他查询

    BooleanQuery query = new BooleanQuery();
    // 交集: Occur.MUST + Occur.MUST
    // 并集: Occur.SHOULD + Occur.SHOULD
    // 非: Occur.MUST_NOT

    query.add(query1, BooleanClause.Occur.SHOULD);
    query.add(query2, BooleanClause.Occur.SHOULD);
    search(query);
}

```

2.8 Lucene的索引修改

```

public void testUpdate() throws IOException{

    // 创建文档对象
    Document document = new Document();
    document.add(new StringField( name: "id", value: "9", Field.Store.YES));
    document.add(new TextField( name: "title", value: "谷歌地图之父跳槽FaceBook", Field.Store.YES));

    // 索引库对象
    Directory directory = FSDirectory.open(new File( pathname: "C:\\tmp\\index"));
    // 索引写入器配置对象
    IndexWriterConfig conf = new IndexWriterConfig(Version.LATEST, new IKAnalyzer());
    // 索引写入器对象
    IndexWriter indexWriter = new IndexWriter(directory, conf);

    // 执行更新操作
    indexWriter.updateDocument(new Term( fld: "id", text: "1"), document);
    // 提交
    indexWriter.commit();
    // 关闭
    indexWriter.close();

}

```

2.9 Lucene 的索引删除

```

/**
 * 更新索引 本质先删除再添加
 * 先删除所有满足条件的文档，再创建文档
 * 因此，更新索引通常要根据唯一字段
 * @throws IOException
 */
@Test
public void testDelete() throws IOException {
    // 创建目录对象
    Directory directory = FSDirectory.open(new File( pathname: "C:\\tmp\\indexDir"));
    // 创建索引写入器配置对象
    IndexWriterConfig conf = new IndexWriterConfig(Version.LATEST, new IKAnalyzer());
    // 创建索引写入器对象
    IndexWriter indexWriter = new IndexWriter(directory, conf);
    // 执行删除操作(根据词条)，要求id字段必须是字符串类型
    // indexWriter.deleteDocuments(new Term("id", "5"));
    // 根据查询条件删除
    // indexWriter.deleteDocuments(NumericRangeQuery.newLongRange("id", 8L, 4L, true, false));
    indexWriter.deleteAll(); // 删除所有
    indexWriter.commit(); // 提交
    indexWriter.close(); // 关闭
}

```

3. Lucene的有趣部分()

3.1 Lucene的高亮显示

3.1.1 Lucene的高亮

3.2 Lucene的排序

3.3 Lucene的分页

3.4 Lucene的加权因子 (激励因子)

/大白话就不说了 大家应该都知道这个激励因子咋回事/