信息检索导论--
lucene

授课人：李永可
邮箱：lyk@xjau.edu.cn

lucene检索

IndexReader

读取倒排索引，常用方法：

Static Open(Directory dir)

打开索引文件

IndexReader reader=IndexReader.open(indexpath)

IndexSearcher

检索倒排索引

IndexSearcher serarcher=new IndexSearcher(IndexReader reader)

常用方法：

1：

IndexSearcher

常用方法：

1、setDefaultFieldSortScoring(boolean doTrackScores, boolean doMaxScore)

- By default, no scores are computed when sorting by field (using search(Query,Filter,int,Sort)). You can change that, per IndexSearcher instance, by calling this method. Note that this will incur a CPU cost.
- **Parameters:** doTrackScores - If true, then scores are returned for every matching document in TopFieldDocs. doMaxScore - If true, then the max score for all matching docs is computed.

IndexSearcher

2、setSimilarity(Similarity similarity)

Expert: Set the Similarity implementation used by this Searcher.

Example:

Searcher.setSimilarity(new IKSimilarity())

IndexSearcher

3、**search**(Query query, int n, Sort sort)

Search implementation with arbitrary sorting and no filter.

4、**search**(Query query, Filter filter, int n, Sort sort)

Search implementation with arbitrary sorting.

Query

- 当用户输入一个关键字，搜索引擎接收到后，并不是立刻就将它放入后台开始进行关键字的检索，而应当首先对这个关键字进行一定的分析和处理，使之成为一种后台可以理解的形式，只有这样，才能提高检索的效率，同时检索出更加有效的结果。那么，在Lucene中，这种处理，其实就是构建一个**Query**对象。

Query

- **按词条搜索—TermQuery**

TermQuery是最简单、也是最常用的**Query**。TermQuery可以理解成为“词条搜索”，在搜索引擎中最基本的搜索就是在索引中搜索某一词条，而TermQuery就是用来完成这项工作的。

- 使用方法：

```
Term aTerm = new Term("contents", "java")
Query query = new TermQuery(aTerm)
```

Query

➢布尔搜索—BooleanQuery

BooleanQuery也是实际开发过程中经常使用的一种**Query**。它其实是一个组合的**Query**，在使用时可以把各种**Query**对象添加进去并标明它们之间的逻辑关系。

Query

- BooleanQuery 使用方法：

```
query1 = new TermQuery(new Term(“name”,word1))
query2 = new TermQuery(new Term(“name”,word2))
query = new BooleanQuery();
query.add(query1, true);
 query.add(query2, true);
```

QueryParser

- QueryParser实际上就是一个解析用户输入的工具，可以通过扫描用户输入的字符串，生成**Query**对象
- 用法：

Query query = null;

query = QueryParser.parse(keywords,fieldName,new StandardAnalyzer());

MultiFieldQueryParser

- **MultiFieldQueryParser多域解析查询，可同时对多个域进行解析查询**
- 用法：

public
MultiFieldQueryParser(Version matchVersion, String[] fields, Analyzer analyzer)