1. **索引库优化**

1．1什么是索引库

索引库是Lucene的重要的存储结构，它包括二部份：原始记录表，词汇表

原始记录表：存放的是原始记录信息，Lucene为存入的内容分配一个唯一的编号

词汇表：存放的是经过分词器拆分出来的词汇和该词汇在原始记录表中的编号

1．2为什么要将索引库进行优化

在默认情况下，向索引库中增加一个Document对象时，索引库自动会添加一个扩展名叫\*.cfs的二进制压缩文件，如果向索引库中存Document对象过多，那么\*.cfs也会不断增加，同时索引库的容量也会不断增加，影响索引库的大小。

1．3索引库优化方案

1．3．1合并cfs文件，合并后的cfs文件是二进制压缩字符，能解决是的文件大小和数量的问题

|  |
| --- |
| indexWriter.addDocument(document);  indexWriter.optimize();  indexWriter.close(); |

1．3．2设定合并因子，自动合并cfs文件，默认10个cfs文件合并成一个cfs文件

|  |
| --- |
| indexWriter.addDocument(document);  indexWriter.setMergeFactor(3);  indexWriter.close(); |

1．3．3使用RAMDirectory，类似于内存索引库，能解决是的读取索引库文件的速度问题，

它能以空换时，提高速度快，但不能持久保存，因此启动时加载硬盘中的索引库到内存中的索引库，退出时将内存中的索引库保存到硬盘中的索引库，且内容不能重复。

|  |
| --- |
| Article article = **new** Article(1,"培训","传智是一家Java培训机构");  Document document = LuceneUtil.*javabean2document*(article);    Directory fsDirectory = FSDirectory.*open*(**new** File("E:/indexDBDBDBDBDBDBDBDB"));  Directory ramDirectory = **new** RAMDirectory(fsDirectory);    IndexWriter fsIndexWriter = **new** IndexWriter(fsDirectory,LuceneUtil.*getAnalyzer*(),**true**,LuceneUtil.*getMaxFieldLength*());  IndexWriter ramIndexWriter = **new** IndexWriter(ramDirectory,LuceneUtil.*getAnalyzer*(),LuceneUtil.*getMaxFieldLength*());    ramIndexWriter.addDocument(document);  ramIndexWriter.close();    fsIndexWriter.addIndexesNoOptimize(ramDirectory);  fsIndexWriter.close(); |

1. **分词器**

2．1什么是分词器

采用一种算法，将中英文本中的字符拆分开来，形成词汇，以待用户输入关健字后搜索

2．2为什么要分词器

因为用户输入的搜索的内容是一段文本中的一个关健字，和原始表中的内容有差别，

但作为搜索引擎来讲，又得将相关的内容搜索出来，此时就得采用分词器来最大限度

匹配原始表中的内容

2．3分词器工作流程

步一：按分词器拆分出词汇

步二：去除停用词和禁用词

步三：如果有英文，把英文字母转为小写，即搜索不分大小写

2．4分词器例子图解：“传智播客说我们的首都是北京呀I AM zhaojun”

2．5演示常用分词器测试，只观查结果

|  |
| --- |
| **private** **static** **void** testAnalyzer(Analyzer analyzer, String text) **throws** Exception {  System.*out*.println("当前使用的分词器：" + analyzer.getClass());  TokenStream tokenStream = analyzer.tokenStream("content",**new** StringReader(text));  tokenStream.addAttribute(TermAttribute.**class**);  **while** (tokenStream.incrementToken()) {  TermAttribute termAttribute = tokenStream.getAttribute(TermAttribute.**class**);  System.*out*.println(termAttribute.term());  }  } |

2．6使用第三方IKAnalyzer分词器--------中文首选

需求：过滤掉上面例子中的“说”，“的”，“呀”，且将“传智播客”看成一个整体 关健字

步一：导入IKAnalyzer分词器核心jar包，IKAnalyzer3.2.0Stable.jar

步二：将IKAnalyzer.cfg.xml和stopword.dic和xxx.dic文件复制到MyEclipse的src目录下，

再进行配置，在配置时，首行需要一个空行

1. **搜索结果高亮**

3．1什么是搜索结果高亮

在搜索结果中，将与关健字相同的字符用红色显示

|  |
| --- |
| String keywords = "培训";  List<Article> articleList = **new** ArrayList<Article>();  QueryParser queryParser = **new** QueryParser(LuceneUtil.*getVersion*(),"content",LuceneUtil.*getAnalyzer*());  Query query = queryParser.parse(keywords);  IndexSearcher indexSearcher = **new** IndexSearcher(LuceneUtil.*getDirectory*());  TopDocs topDocs = indexSearcher.search(query,1000000);    Formatter formatter = **new** SimpleHTMLFormatter("<font color='red'>","</font>");  Scorer scorer = **new** QueryScorer(query);  Highlighter highlighter = **new** Highlighter(formatter,scorer);    **for**(**int** i=0;i<topDocs.scoreDocs.length;i++){  ScoreDoc scoreDoc = topDocs.scoreDocs[i];  **int** no = scoreDoc.doc;  Document document = indexSearcher.doc(no);    String highlighterContent = highlighter.getBestFragment(LuceneUtil.*getAnalyzer*(),"content",document.get("content"));  document.getField("content").setValue(highlighterContent);    Article article = (Article) LuceneUtil.*document2javabean*(document,Article.**class**);  articleList.add(article);  }  **for**(Article article : articleList){  System.*out*.println(article);  }  } |

1. **搜索结果摘要**

4．1什么是搜索结果搞要

如果搜索结果内容太多，我们只想显示前几个字符， 必须与高亮一起使用

|  |
| --- |
| String keywords = "培训";  List<Article> articleList = **new** ArrayList<Article>();  QueryParser queryParser = **new** QueryParser(LuceneUtil.*getVersion*(),"content",LuceneUtil.*getAnalyzer*());  Query query = queryParser.parse(keywords);  IndexSearcher indexSearcher = **new** IndexSearcher(LuceneUtil.*getDirectory*());  TopDocs topDocs = indexSearcher.search(query,1000000);    Formatter formatter = **new** SimpleHTMLFormatter("<font color='red'>","</font>");  Scorer scorer = **new** QueryScorer(query);  Highlighter highlighter = **new** Highlighter(formatter,scorer);    Fragmenter fragmenter = **new** SimpleFragmenter(4);  highlighter.setTextFragmenter(fragmenter);    **for**(**int** i=0;i<topDocs.scoreDocs.length;i++){  ScoreDoc scoreDoc = topDocs.scoreDocs[i];  **int** no = scoreDoc.doc;  Document document = indexSearcher.doc(no);    String highlighterContent = highlighter.getBestFragment(LuceneUtil.*getAnalyzer*(),"content",document.get("content"));  document.getField("content").setValue(highlighterContent);    Article article = (Article) LuceneUtil.*document2javabean*(document,Article.**class**);  articleList.add(article);  }  **for**(Article article : articleList){  System.*out*.println(article);  }  } |

1. **搜索结果排序**

5．1什么是搜索结果排序

搜索结果是按某个或某些字段高低排序来显示的结果

5．2影响网站排名的先后的有多种

head/meta/

网页的标签整洁

网页执行速度

采用div+css

。。。。。。

5．3Lucene中的显示结果次序与相关度得分有关

ScoreDoc.score;

默认情况下，Lucene是按相关度得分排序的，得分高排在前，得分低排在后

如果相关度得分相同，按插入索引库的先后次序排序

5．4Lucene中的设置相关度得分

|  |
| --- |
| IndexWriter indexWriter = **new** IndexWriter(LuceneUtil.*getDirectory*(),LuceneUtil.*getAnalyzer*(),LuceneUtil.*getMaxFieldLength*());  document.setBoost(20F);  indexWriter.addDocument(document);  indexWriter.close(); |

5．5Lucene中按单个字段排序

|  |
| --- |
| Sort sort = new Sort(new SortField("id",SortField.INT,true));  TopDocs topDocs = indexSearcher.search(query,null,1000000,sort); |

5．6Lucene中按多个字段排序

|  |
| --- |
| Sort sort = **new** Sort(**new** SortField("count",SortField.*INT*,**true**),**new** SortField("id",SortField.*INT*,**true**));  TopDocs topDocs = indexSearcher.search(query,**null**,1000000,sort); |

在多字段排序中，只有第一个字段排序结果相同时，第二个字段排序才有作用

提倡用数值型排序

1. **条件搜索**

6．1什么是条件搜索

用关健字与指定的单列或多例进行匹配的搜索

6．2单字段条件搜索

|  |
| --- |
| QueryParser queryParser = new QueryParser(LuceneUtil.getVersion(),"content",LuceneUtil.getAnalyzer()); |

6．3多字段条件搜索，项目中提倡多字段搜索

|  |
| --- |
| QueryParser queryParser = new MultiFieldQueryParser(LuceneUtil.getVersion(),new String[]{"content","title"},LuceneUtil.getAnalyzer()); |

1. **用第三方工具类，将JavaBean、List、Map<String,Object>转成JSON文本**

导入第三方jar包：

》commons-beanutils-1.7.0.jar

》commons-collections-3.1.jar

》commons-lang-2.5.jar

》commons-logging-1.1.1.jar

》ezmorph-1.0.3.jar

》json-lib-2.1-jdk15.jar

（1）JavaBean->JSON

》JSONArray jsonArray = JSONArray.fromObject(city);

》String jsonJAVA = jsonArray.toString();

（2）List<JavaBean>->JSON

》JSONArray jsonArray = JSONArray.fromObject(cityList);

》String jsonJAVA = jsonArray.toString();

（3）List<String>->JSON

》JSONArray jsonArray = JSONArray.fromObject(stringList);

》String jsonJAVA = jsonArray.toString();

（4）Map<String,Object>->JSON【重点】

|  |
| --- |
| List<User> userList = **new** ArrayList<User>();  userList.add(**new** User(100,"哈哈",1000));  userList.add(**new** User(200,"呵呵",2000));  userList.add(**new** User(300,"嘻嘻",3000));    Map<String,Object> map = **new** LinkedHashMap<String,Object>();  map.put("total",userList.size());  map.put("rows",userList);    JSONArray jsonArray = JSONArray.*fromObject*(map);  String jsonJAVA = jsonArray.toString();  System.*out*.println(jsonJAVA);    jsonJAVA = jsonJAVA.substring(1,jsonJAVA.length()-1);  System.*out*.println(jsonJAVA); |

1. **用JSON文本动态创建DataGrid**

|  |
| --- |
| <table id="dg"></table>  $('#dg').datagrid({  url : 'data/datagrid\_data.json',  columns:[[  {field:'code',title:'编号',width:100},  {field:'name',title:'姓名',width:100},  {field:'price',title:'薪水',width:100}  ]]  }); |

1. **用Servlet返回JSON文本动态创建DataGrid**

|  |
| --- |
| <table id="dg"></table>  $('#dg').datagrid({  url : '/lucene-day02/JsonServlet',  columns:[[  {field:'code',title:'编号',width:100},  {field:'name',title:'姓名',width:100},  {field:'price',title:'薪水',width:100}  ]]  });  Servlet：  public void doPost(HttpServletRequest request, HttpServletResponse response)  request.setCharacterEncoding("UTF-8");  Integer currPageNO = null;  try {  //DateGrid会向服务端传入page参数，表示第几页  currPageNO = Integer.parseInt(request.getParameter("page"));  } catch (Exception e) {  currPageNO = 1;  }  //DateGrid会向服务端传入rows参数，表示几条记录  //Integer rows = Integer.parseInt(request.getParameter("rows"));  //System.out.println(currPageNO+":"+rows);    UserService userService = new UserService();  PageBean pageBean = userService.fy(currPageNO);    Map<String,Object> map = new LinkedHashMap<String,Object>();  map.put("total",pageBean.getAllRecordNO());  map.put("rows",pageBean.getUserList());    JSONArray jsonArray = JSONArray.fromObject(map);  String jsonJAVA = jsonArray.toString();  jsonJAVA = jsonJAVA.substring(1,jsonJAVA.length()-1);    System.out.println(jsonJAVA);  response.setContentType("text/html;charset=UTF-8");  response.getWriter().write(jsonJAVA);  response.getWriter().flush();  response.getWriter().close();    } |

1. **使用Jsp +Js + Jquery + EasyUI + Servlet + Lucene，完成分页**

步一：创建ArticleDao.java类

|  |
| --- |
| **public** **class** ArticleDao {  **public** Integer getAllObjectNum(String keywords) **throws** Exception{  QueryParser queryParser = **new** QueryParser(LuceneUtil.*getVersion*(),"content",LuceneUtil.*getAnalyzer*());  Query query = queryParser.parse(keywords);  IndexSearcher indexSearcher = **new** IndexSearcher(LuceneUtil.*getDirectory*());  TopDocs topDocs = indexSearcher.search(query,3);  **return** topDocs.totalHits;  }  **public** List<Article> findAllObjectWithFY(String keywords,Integer start,Integer size) **throws** Exception{  List<Article> articleList = **new** ArrayList<Article>();  QueryParser queryParser = **new** QueryParser(LuceneUtil.*getVersion*(),"content",LuceneUtil.*getAnalyzer*());  Query query = queryParser.parse(keywords);  IndexSearcher indexSearcher = **new** IndexSearcher(LuceneUtil.*getDirectory*());  TopDocs topDocs = indexSearcher.search(query,100000000);  **int** middle = Math.*min*(start+size,topDocs.totalHits);  **for**(**int** i=start;i<middle;i++){  ScoreDoc scoreDoc = topDocs.scoreDocs[i];  **int** no = scoreDoc.doc;  Document document = indexSearcher.doc(no);  Article article = (Article) LuceneUtil.*document2javabean*(document,Article.**class**);  articleList.add(article);  }  **return** articleList;  }  } |

步二：创建PageBean.java类

|  |
| --- |
| **public** **class** PageBean {  **private** Integer allObjectNum;  **private** Integer allPageNum;  **private** Integer currPageNum;  **private** Integer perPageNum = 2;  **private** List<Article> articleList = **new** ArrayList<Article>();  **public** PageBean(){}  **public** Integer getAllObjectNum() {  **return** allObjectNum;  }  **public** **void** setAllObjectNum(Integer allObjectNum) {  **this**.allObjectNum = allObjectNum;  **if**(**this**.allObjectNum % **this**.perPageNum == 0){  **this**.allPageNum = **this**.allObjectNum / **this**.perPageNum;  }**else**{  **this**.allPageNum = **this**.allObjectNum / **this**.perPageNum + 1;  }  }  **public** Integer getAllPageNum() {  **return** allPageNum;  }  **public** **void** setAllPageNum(Integer allPageNum) {  **this**.allPageNum = allPageNum;  }  **public** Integer getCurrPageNum() {  **return** currPageNum;  }  **public** **void** setCurrPageNum(Integer currPageNum) {  **this**.currPageNum = currPageNum;  }  **public** Integer getPerPageNum() {  **return** perPageNum;  }  **public** **void** setPerPageNum(Integer perPageNum) {  **this**.perPageNum = perPageNum;  }  **public** List<Article> getArticleList() {  **return** articleList;  }  **public** **void** setArticleList(List<Article> articleList) {  **this**.articleList = articleList;  }  } |

步三：创建ArticleService.java类

|  |
| --- |
| **public** **class** ArticleService {  **private** ArticleDao articleDao = **new** ArticleDao();  **public** PageBean fy(String keywords,Integer currPageNum) **throws** Exception{  PageBean pageBean = **new** PageBean();  pageBean.setCurrPageNum(currPageNum);  Integer allObjectNum = articleDao.getAllObjectNum(keywords);  pageBean.setAllObjectNum(allObjectNum);  Integer size = pageBean.getPerPageNum();  Integer start = (pageBean.getCurrPageNum()-1) \* size;  List<Article> articleList = articleDao.findAllObjectWithFY(keywords,start,size);  pageBean.setArticleList(articleList);  **return** pageBean;  }  } |

步四：创建ArticleServlet.java类

|  |
| --- |
| **public** **class** UserServlet **extends** HttpServlet {  **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)**throws** ServletException, IOException {  **try** {  //获取当前页号，默认1  String strCurrPageNO = request.getParameter("page");  **if**(strCurrPageNO == **null**){  strCurrPageNO = "1";  }  Integer currPageNO = Integer.*parseInt*(strCurrPageNO);  //获取关健字  String keywords = request.getParameter("keywords");  //创建业务对象  UserService userService = **new** UserService();  //调用业务层  PageBean pageBean = userService.fy(keywords,currPageNO);  //以下代码生成DateGrid需要的JSON文本  Map<String,Object> map = **new** LinkedHashMap<String,Object>();  //总记录数  map.put("total",pageBean.getAllRecordNO());  //该页显示的内容  map.put("rows",pageBean.getUserList());  JSONArray jsonArray = JSONArray.*fromObject*(map);  String jsonJAVA = jsonArray.toString();  jsonJAVA = jsonJAVA.substring(1,jsonJAVA.length()-1);  //以下代码是将json文本输出到浏览器给DateGrid组件  response.setContentType("text/html;charset=UTF-8");  response.getWriter().write(jsonJAVA);  response.getWriter().flush();  response.getWriter().close();  } **catch** (Exception e) {  }  }  } |

步五：导入EasyUI相关的js包的目录

|  |
| --- |
|  |

步六：在WebRoot目录下创建list.jsp

|  |
| --- |
| <%@ page language="java" pageEncoding="UTF-8"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  <html>  <head>  <link rel="stylesheet" href="themes/default/easyui.css" type="text/css"></link>  <link rel="stylesheet" href="themes/icon.css" type="text/css"></link>  <script type="text/javascript" src="js/jquery.min.js"></script>  <script type="text/javascript" src="js/jquery.easyui.min.js"></script>  <script type="text/javascript" src="locale/easyui-lang-zh\_CN.js"></script>  </head>    <body>    输入姓名关健字：  <input type="text" size="4px" id="name"/>  <input type="button" value="搜索" id="find"/>      <table id="dg" style="width:500px"></table>    <script type="text/javascript">  //定位"搜索"按钮,同时添加单击事件  $("#find").click(function(){  //获取用户名  var name = $("#name").val();  //去二边的空格  name = $.trim(name);  //加载最新数据  $("#dg").datagrid("load",{  "keywords" : name  });  });  </script>    <script type="text/javascript">  //动态创建表格  $("#dg").datagrid({  url:'${pageContext.request.contextPath}/UserServlet?id=' + new Date().getTime(),  fitColumns : true,  singleSelect : true,  columns:[[  {field:'id',title:'编号',width:100,align:'center'},  {field:'name',title:'姓名',width:100,align:'center'},  {field:'sal',title:'薪水',width:100,align:'center'}  ]],  pagination : true,  pageNumber : 1,  pageSize : 2,  pageList:[2]  });  </script>    </body>    </html> |