1. **回顾索引**

定义：索引是对数据库表中一列或多列的值进行排序的一种结构

目的：加快对数据库表中记录的查询

特点：以空间换取时间，提高查询速度快

参见<<索引提高查询速度原理.JPG>>

1. **体验百度 搜索与原理图**

参见<<在baidu中搜索Lucene关健字的结果.JPG>>

参见<<百度索搜宏观原理.JPG>>

参见<<百度索搜微观原理.JPG>>

1. **什么是Lucene**

Lucene是apache软件基金会发布的一个开放源代码的全文检索引擎工具包，由资深全文检索专家Doug Cutting所撰写,它是一个全文检索引擎的架构，提供了完整的创建索引和查询索引，以及部分文本分析的引擎，Lucene的目的是为软件开发人员提供一个简单易用的工具包，以方便在目标系统中实现全文检索的功能，或者是以此为基础建立起完整的全文检索引擎，Lucene在全文检索领域是一个经典的祖先，现在很多检索引擎都是在其基础上创建的，思想是相通的。

即：Lucene是根据关健字来搜索的文本搜索工具，只能在某个网站内部搜索文本内容，不能跨网站搜索

1. **Lucene通常用在什么地方**

Lucece不能用在互联网搜索（即像百度那样），只能用在网站内部的文本搜索（即只能在CRM，RAX，ERP内部使用），但思想是相通的。

参见<<Lucene用在什么地方.JPG>>

参见<<Lucene用在服务端三层结构中的哪一层.JPG>>

1. **Lucene中存的什么内容**

Lucene中存的就是一系列的二进制压缩文件和一些控制文件，它们位于计算机的硬盘上，

这些内容统称为索引库，索引库有二部份组成：

（1）原始记录

存入到索引库中的原始文本，例如：传智是一家IT培训机构

（2）词汇表

按照一定的拆分策略（即分词器）将原始记录中的每个字符拆开后，存入一个供将来搜索的表

参见<< Lucene索引库结构与原理图.JPG>>

1. **为什么网站内部有些地方要用Lucene来索搜，而不全用SQL来搜索**

（1）SQL只能针对数据库表搜索，不能直接针对硬盘上的文本搜索

（2）SQL没有相关度排名

（3）SQL搜索结果没有关健字高亮显示

（4）SQL需要数据库的支持，数据库本身需要内存开销较大，例如：Oracle

（5）SQL搜索有时较慢，尤其是数据库不在本地时，超慢，例如：Oracle

1. **书写代码使用Lucene的流程图**

参见<<Lucene程序宏观结构.JPG>>

参见<<Lucene索引库创建的过程.JPG>>

参见<<Lucene索引库查询的过程.JPG>>

创建索引库：

1. 创建JavaBean对象
2. 创建Docment对象
3. 将JavaBean对象所有的属性值，均放到Document对象中去，属性名可以和JavaBean相同或不同
4. 创建IndexWriter对象
5. 将Document对象通过IndexWriter对象写入索引库中
6. 关闭IndexWriter对象

根据关键字查询索引库中的内容：

1. 创建IndexSearcher对象
2. 创建QueryParser对象
3. 创建Query对象来封装关键字
4. 用IndexSearcher对象去索引库中查询符合条件的前100条记录，不足100条记录的以实际为准
5. 获取符合条件的编号
6. 用indexSearcher对象去索引库中查询编号对应的Document对象
7. 将Document对象中的所有属性取出，再封装回JavaBean对象中去，并加入到集合中保存，以备将之用
8. **Lucene快速入门**

步一：创建javaweb工程，取名叫lucene-day01

步二：导入Lucene相关的jar包

lucene-core-3.0.2.jar【Lucene核心】

lucene-analyzers-3.0.2.jar【分词器】

lucene-highlighter-3.0.2.jar【Lucene会将搜索出来的字，高亮显示，提示用户】

lucene-memory-3.0.2.jar【索引库优化策略】

步三：创建包结构

cn.itcast.javaee.lucene.entity

cn.itcast.javaee.lucene.firstapp

cn.itcast.javaee.lucene.secondapp

cn.itcast.javaee.lucene.crud

cn.itcast.javaee.lucene.fy

cn.itcast.javaee.lucene.utils

。。 。。 。

步四：创建JavaBean类

|  |
| --- |
| **public** **class** Article {  **private** Integer id;//标题  **private** String title;//标题  **private** String content;//内容  **public** Article(){}  **public** Article(Integer id, String title, String content) {  **this**.id = id;  **this**.title = title;  **this**.content = content;  }  **public** Integer getId() {  **return** id;  }  **public** **void** setId(Integer id) {  **this**.id = id;  }  **public** String getTitle() {  **return** title;  }  **public** **void** setTitle(String title) {  **this**.title = title;  }  **public** String getContent() {  **return** content;  }  **public** **void** setContent(String content) {  **this**.content = content;  }  } |

步五：创建FirstLucene.java类，编写createIndexDB()和findIndexDB()二个业务方法

|  |
| --- |
| @Test  **public** **void** createIndexDB() **throws** Exception{  Article article = **new** Article(1,"培训","传智是一个Java培训机构");  Document document = **new** Document();  document.add(**new** Field("id",article.getId().toString(),Store.*YES*,Index.*ANALYZED*));  document.add(**new** Field("title",article.getTitle(),Store.*YES*,Index.*ANALYZED*));  document.add(**new** Field("content",article.getContent(),Store.*YES*,Index.*ANALYZED*));  Directory directory = FSDirectory.*open*(**new** File("E:/LuceneDBDBDBDBDBDBDBDBDB"));  Analyzer analyzer = **new** StandardAnalyzer(Version.*LUCENE\_30*);  MaxFieldLength maxFieldLength = MaxFieldLength.*LIMITED*;  IndexWriter indexWriter = **new** IndexWriter(directory,analyzer,maxFieldLength);  indexWriter.addDocument(document);  indexWriter.close();  } |

|  |
| --- |
| @Test  **public** **void** findIndexDB() **throws** Exception{  List<Article> articleList = **new** ArrayList<Article>();  String keywords = "传";  Directory directory = FSDirectory.*open*(**new** File("E:/LuceneDBDBDBDBDBDBDBDBDB"));  Version version = Version.*LUCENE\_30*;  Analyzer analyzer = **new** StandardAnalyzer(version);  QueryParser queryParser = **new** QueryParser(version,"content",analyzer);  Query query = queryParser.parse(keywords);  IndexSearcher indexSearcher = **new** IndexSearcher(directory);  TopDocs topDocs = indexSearcher.search(query,10);  **for**(**int** i=0;i<topDocs.scoreDocs.length;i++){  ScoreDoc scoreDoc = topDocs.scoreDocs[i];  **int** no = scoreDoc.doc;  Document document = indexSearcher.doc(no);  String id = document.get("id");  String title = document.get("title");  String content = document.get("content");  Article article = **new** Article(Integer.*parseInt*(id),title,content);  articleList.add(article);  }  **for**(Article article : articleList){  System.*out*.println(article.getId()+":"+article.getTitle()+":"+article.getContent());  }  } |

1. **创建LuceneUtil工具类，使用反射，封装通用的方法**

|  |
| --- |
| **public** **class** LuceneUtil {  **private** **static** Directory *directory* ;  **private** **static** Analyzer *analyzer* ;  **private** **static** Version *version*;  **private** **static** MaxFieldLength *maxFieldLength*;  **static**{  **try** {  *directory* = FSDirectory.*open*(**new** File("E:/LuceneDBDBDBDBDBDBDBDBDB"));  *version* = Version.*LUCENE\_30*;  *analyzer* = **new** StandardAnalyzer(*version*);  *maxFieldLength* = MaxFieldLength.*LIMITED*;  } **catch** (Exception e) {  **throw** **new** RuntimeException(e);  }  }  **public** **static** Directory getDirectory() {  **return** *directory*;  }  **public** **static** Analyzer getAnalyzer() {  **return** *analyzer*;  }  **public** **static** Version getVersion() {  **return** *version*;  }  **public** **static** MaxFieldLength getMaxFieldLength() {  **return** *maxFieldLength*;  }  **public** **static** Document javabean2documemt(Object obj) **throws** Exception{  Document document = **new** Document();  Class clazz = obj.getClass();  java.lang.reflect.Field[] reflectFields = clazz.getDeclaredFields();  **for**(java.lang.reflect.Field field : reflectFields){  field.setAccessible(**true**);  String fieldName = field.getName();  String init = fieldName.substring(0,1).toUpperCase();  String methodName = "get" + init + fieldName.substring(1);  Method method = clazz.getDeclaredMethod(methodName,**null**);  String returnValue = method.invoke(obj,**null**).toString();  document.add(**new** Field(fieldName,returnValue,Store.*YES*,Index.*ANALYZED*));  }  **return** document;  }  **public** **static** Object document2javabean(Document document,Class clazz) **throws** Exception{  Object obj = clazz.newInstance();  java.lang.reflect.Field[] reflectFields = clazz.getDeclaredFields();  **for**(java.lang.reflect.Field field : reflectFields){  field.setAccessible(**true**);  String fieldName = field.getName();  String fieldValue = document.get(fieldName);  BeanUtils.*setProperty*(obj,fieldName,fieldValue);  }  **return** obj;  }  } |

1. **使用LuceneUtil工具类，重构FirstLucene.java为SecondLucene.java**

|  |
| --- |
| **public** **class** SecondLucene {  @Test  **public** **void** createIndexDB() **throws** Exception{  Article article = **new** Article(1,"Java培训","传智是一个Java培训机构");  Document document = LuceneUtil.*javabean2documemt*(article);  IndexWriter indexWriter = **new** IndexWriter(LuceneUtil.*getDirectory*(),LuceneUtil.*getAnalyzer*(),LuceneUtil.*getMaxFieldLength*());  indexWriter.addDocument(document);  indexWriter.close();  }  @Test  **public** **void** findIndexDB() **throws** Exception{  List<Article> articleList = **new** ArrayList<Article>();  String keywords = "传";  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,10);  **for**(**int** i=0;i<topDocs.scoreDocs.length;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);  }  **for**(Article article : articleList){  System.*out*.println(article.getId()+":"+article.getTitle()+":"+article.getContent());  }  }  } |

1. **使用LuceneUtil工具类，完成CURD操作**

|  |
| --- |
| **public** **class** LuceneCURD {  @Test  **public** **void** addIndexDB() **throws** Exception{  Article article = **new** Article(1,"培训","传智是一个Java培训机构");  Document document = LuceneUtil.*javabean2documemt*(article);  IndexWriter indexWriter = **new** IndexWriter(LuceneUtil.*getDirectory*(),LuceneUtil.*getAnalyzer*(),LuceneUtil.*getMaxFieldLength*());  indexWriter.addDocument(document);  indexWriter.close();  }  @Test  **public** **void** updateIndexDB() **throws** Exception{  Integer id = 1;  Article article = **new** Article(1,"培训","广州传智是一个Java培训机构");  Document document = LuceneUtil.*javabean2documemt*(article);  Term term = **new** Term("id",id.toString());  IndexWriter indexWriter = **new** IndexWriter(LuceneUtil.*getDirectory*(),LuceneUtil.*getAnalyzer*(),LuceneUtil.*getMaxFieldLength*());  indexWriter.updateDocument(term,document);  indexWriter.close();  }  @Test  **public** **void** deleteIndexDB() **throws** Exception{  Integer id = 1;  Term term = **new** Term("id",id.toString());  IndexWriter indexWriter = **new** IndexWriter(LuceneUtil.*getDirectory*(),LuceneUtil.*getAnalyzer*(),LuceneUtil.*getMaxFieldLength*());  indexWriter.deleteDocuments(term);  indexWriter.close();  }  @Test  **public** **void** deleteAllIndexDB() **throws** Exception{  IndexWriter indexWriter = **new** IndexWriter(LuceneUtil.*getDirectory*(),LuceneUtil.*getAnalyzer*(),LuceneUtil.*getMaxFieldLength*());  indexWriter.deleteAll();  indexWriter.close();  }  @Test  **public** **void** searchIndexDB() **throws** Exception{  List<Article> articleList = **new** ArrayList<Article>();  String keywords = "传智";  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,10);  **for**(**int** i = 0;i<topDocs.scoreDocs.length;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);  }  **for**(Article article : articleList){  System.*out*.println(article.getId()+":"+article.getTitle()+":"+article.getContent());  }  }  } |

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** ArticleServlet **extends** HttpServlet {  **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)**throws** ServletException, IOException {  **try** {  request.setCharacterEncoding("UTF-8");  Integer currPageNum = Integer.*parseInt*(request.getParameter("currPageNum"));  String keywords = request.getParameter("keywords");  ArticleService articleService = **new** ArticleService();  PageBean pageBean = articleService.fy(keywords,currPageNum);  request.setAttribute("pageBean",pageBean);  request.getRequestDispatcher("/list.jsp").forward(request,response);  } **catch** (Exception e) {  e.printStackTrace();  }  }  } |

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>      <!-- 输入区 -->  <form action="${pageContext.request.contextPath}/ArticleServlet?currPageNum=1" method="POST">  输入关健字：<input type="text" name="keywords" value="传智" maxlength="4"/>  <input type="button" value="提交"/>  </form>      <!-- 显示区 -->  <table border="2" align="center" width="70%">  <tr>  <th>编号</th>  <th>标题</th>  <th>内容</th>  </tr>  <c:forEach var="article" items="${pageBean.articleList}">  <tr>  <td>${article.id}</td>  <td>${article.title}</td>  <td>${article.content}</td>  </tr>  </c:forEach>  </table>  <!-- 分页组件区 -->  <center>  <div id="pp" style="background:#efefef;border:1px solid #ccc;width:600px"></div>  </center>  <script type="text/javascript">  $("#pp").pagination({  total:${pageBean.allObjectNum},  pageSize:${pageBean.perPageNum},  showPageList:false,  showRefresh:false,  pageNumber:${pageBean.currPageNum}  });  $("#pp").pagination({  onSelectPage:function(pageNumber){  $("form").attr("action","${pageContext.request.contextPath}/ArticleServlet?currPageNum="+pageNumber);  $("form").submit();  }  });  </script>  <script type="text/javascript">  $(":button").click(function(){  $("form").submit();  });  </script>  </body>  </html> |

步六：在WebRoot目录下创建list2.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>  <title>根据关键字分页查询所有信息</title>  </head>  <body>    <!-- 输入区 -->  <form action="${pageContext.request.contextPath}/ArticleServlet" method="POST">  <input id="currPageNOID" type="hidden" name="currPageNO" value="1">  <table border="2" align="center">  <tr>  <th>输入关键字：</th>  <th><input type="text" name="keywords" maxlength="4" value="${requestScope.keywords}"/></th>  <th><input type="submit" value="站内搜索"/></th>  </tr>  </table>  </form>      <!-- 输出区 -->  <table border="2" align="center" width="60%">  <tr>  <th>编号</th>  <th>标题</th>  <th>内容</th>  </tr>  <c:forEach var="article" items="${requestScope.pageBean.articleList}">  <tr>  <td>${article.id}</td>  <td>${article.title}</td>  <td>${article.content}</td>  </tr>  </c:forEach>  <!-- 分页条 -->  <tr>  <td colspan="3" align="center">  <a onclick="fy(1)" style="text-decoration:none;cursor:hand">  【首页】  </a>  <c:choose>  <c:when test="${requestScope.pageBean.currPageNO+1<=requestScope.pageBean.allPageNO}">  <a onclick="fy(${requestScope.pageBean.currPageNO+1})" style="text-decoration:none;cursor:hand">  【下一页】  </a>  </c:when>  <c:otherwise>  下一页  </c:otherwise>  </c:choose>  <c:choose>  <c:when test="${requestScope.pageBean.currPageNO-1>0}">  <a onclick="fy(${requestScope.pageBean.currPageNO-1})" style="text-decoration:none;cursor:hand">  【上一页】  </a>  </c:when>  <c:otherwise>  上一页  </c:otherwise>  </c:choose>  <a onclick="fy(${requestScope.pageBean.allPageNO})" style="text-decoration:none;cursor:hand">  【未页】  </a>  </td>  </tr>  </table>  <script type="text/javascript">  function fy(currPageNO){  document.getElementById("currPageNOID").value = currPageNO;  document.forms[0].submit();  }  </script>    </body>  </html> |

|  |
| --- |
|  |

|  |
| --- |
|  |