利用Compass实现一个简单的搜索引擎[转贴]

2006年08月29日 星期二 10:40 A.M.

|  |
| --- |
| 这是我朋友darkhe所写的一篇文章，将其转贴在此。        Compass是一流的开放源码JAVA搜索引擎框架，对于你的应用修饰，搜索引擎语义更具有能力。依靠顶级的Lucene搜索引擎，Compass 结合了，像 Hibernate和 Sprin的流行的框架，为你的应用提供了从数据模型和数据源同步改变的搜索力.并且添加了2方面的特征,事物管理和快速更新优化. Compass的目标是：把java应用简单集成到搜索引擎中.编码更少，查找数据更便捷。     下面以一个应用场景分步骤讲解如何利用compass实现搜索引擎： 1. 这里我们有一个Article表，希望利用compass实现对它的搜索。   Article的结构如下：   CREATE TABLE `article` (     `ArticleID` bigint(20) NOT NULL,     `PersonInfoID` bigint(20) default NULL,     `ArticleTitle` varchar(200) default NULL,     `PublishDate` datetime default NULL,     `Summary` text,     `Content` longtext,     `KeyList` text,     PRIMARY KEY  (`ArticleID`),     KEY `PersonInfoArticle\_FK` (`PersonInfoID`)   ) ENGINE=InnoDB DEFAULT CHARSET=utf8;   我们希望利用compass对它的ArticleTitle、Summary、Content和KeyList进行全文检索。下面开始行动吧。    2. 首先到<http://www.opensymphony.com/compass/download.action> 上下载一个compass的发布版，我们下载的是Version 1.0.0的With Dependencies 。这样就可能省去寻找相关信赖库的麻烦了。  3. 将compass1.0解压到一个合适的目录，我们的工作目录是d:\develop\compass1.0  4. 我们是在eclipse环境下实现当前要求的，所以建议你也安装一个eclipse 3.2。  5. 首先我们在eclipse中建立了一个java工程，名为mycompass。  6. 然后我们在工程目录中建立了一个lib目录，用来存放本次工程所需要的所有compass和其它相关的库文件，并将他们设置为当前工程构建路径中需要的库文件。所有这些文件可以在compass的安装目录的lib目录找到。    下面是我们的库文件列表：   7. 建立了Article表的pojo类。   package com.darkhe.sample.mycompass;      // Generated 2006-8-2 10:57:06 by Hibernate Tools 3.2.0.beta6a      import java.util.Date;      /\*\*    \* Article generated by hbm2java    \*/   public class Article implements java.io.Serializable {       // Fields           private long articleId;       private Long personInfoId;       private String articleTitle;       private Date publishDate;       private String summary;       private String content;       private String keyList;       // Constructors       /\*\* default constructor \*/    public Article() {    }       /\*\* minimal constructor \*/    public Article(long articleId) {     this.articleId = articleId;    }       /\*\* full constructor \*/    public Article(long articleId, Long personInfoId, String articleTitle,      Date publishDate, String summary, String content, String keyList) {     this.articleId = articleId;     this.personInfoId = personInfoId;     this.articleTitle = articleTitle;     this.publishDate = publishDate;     this.summary = summary;     this.content = content;     this.keyList = keyList;    }       // Property accessors    public long getArticleId() {     return this.articleId;    }       public void setArticleId(long articleId) {     this.articleId = articleId;    }       public Long getPersonInfoId() {     return this.personInfoId;    }       public void setPersonInfoId(Long personInfoId) {     this.personInfoId = personInfoId;    }       public String getArticleTitle() {     return this.articleTitle;    }       public void setArticleTitle(String articleTitle) {     this.articleTitle = articleTitle;    }       public Date getPublishDate() {     return this.publishDate;    }       public void setPublishDate(Date publishDate) {     this.publishDate = publishDate;    }       public String getSummary() {     return this.summary;    }       public void setSummary(String summary) {     this.summary = summary;    }       public String getContent() {     return this.content;    }       public void setContent(String content) {     this.content = content;    }       public String getKeyList() {     return this.keyList;    }       public void setKeyList(String keyList) {     this.keyList = keyList;    }      }  8. 建立hibernate的pojo到数据表映射文件  <?xml version="1.0"?>  <!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"  "<http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd>">  <!-- Generated 2006-8-2 10:57:07 by Hibernate Tools 3.2.0.beta6a -->  <hibernate-mapping>     <class name="com.darkhe.sample.mycompass.Article" table="article" catalog="freedom">         <comment></comment>         <id name="articleId" type="long">             <column name="ArticleID" />             <generator class="assigned" />         </id>         <property name="personInfoId" type="java.lang.Long">             <column name="PersonInfoID">                 <comment></comment>             </column>         </property>         <property name="articleTitle" type="string">             <column name="ArticleTitle" length="200">                 <comment></comment>             </column>         </property>         <property name="publishDate" type="timestamp">             <column name="PublishDate" length="19">                 <comment></comment>             </column>         </property>         <property name="summary" type="string">             <column name="Summary" length="65535">                 <comment></comment>             </column>         </property>         <property name="content" type="string">             <column name="Content">                 <comment></comment>             </column>         </property>         <property name="keyList" type="string">             <column name="KeyList" length="65535">                 <comment></comment>             </column>         </property>     </class>  </hibernate-mapping>  9. 开始配置compass，首先是compass的系统配置文件 mycompass.cfg.xml <?xml version="1.0" encoding="UTF-8"?> <compass-core-config  xmlns="<http://www.opensymphony.com/compass/schema/core-config>"  xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"  xsi:schemaLocation="<http://www.opensymphony.com/compass/schema/core-config>            <http://www.opensymphony.com/compass/schema/compass-core-config.xsd>">   <compass name="default"> <!—这个名字随你取了，但它是必须的-->    <connection>    <file path="target" /> <!—这里是索引文件的存放路径，我们设置的是当前工程的相对路径target-?   </connection>    <searchEngine>     <!-- 因是使用自己的分词算法，所以这里的类型必须是CustomAnalyzer -->           <analyzer name="MMAnalyer" type="CustomAnalyzer" analyzerClass="jeasy.analysis.MMAnalyzer">               <stopWords>                   <stopWord value="test" />               </stopWords>           </analyzer>       </searchEngine>bushi   </compass> </compass-core-config>  在上面的配置中，我们使用的我们选用的一个中文分词算法库，你可以用compass自带的。具体compass提供了哪些分词算法，请查阅compass的手册。  10. 然后是mycompass.cmd.xml  <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE compass-core-meta-data PUBLIC      "-//Compass/Compass Core Meta Data DTD 1.0//EN"     "<http://www.opensymphony.com/compass/dtd/compass-core-meta-data.dtd>">  <compass-core-meta-data>  <!-- 定义一个实体和字段组-->     <meta-data-group id="mycompass" displayName="My Compass">          <description>Mycompass Meta Data</description>                <uri>http://com/darkhe/sample/mycompass</uri>    <!-- 申明所有需要检索的实体-->                 <alias id="Article" displayName="Article">             <description>Article alias</description>             <uri>http://com/darkhe/sample/mycompass/alias/Article</uri>             <name>Article</name>         </alias>           <!-- 申明所有需要检索的属性或者字段，而不管这些属性或者字段是哪个实体的 -->                 <meta-data id="ArticleTitle" displayName="ArticleTitle">             <description>ArticleTitle</description>             <uri>http://com/darkhe/sample/mycompass/alias/ArticleTitle</uri>             <name>ArticleTitle</name>         </meta-data>                  <meta-data id="PublishDate" displayName="PublishDate">             <description>PublishDate</description>             <uri>http://com/darkhe/sample/mycompass/alias/PublishDate</uri>             <name format="yyyy-MM-dd hh:mm:ss">date</name>         </meta-data>                  <meta-data id="Summary" displayName="Summary">             <description>Summary</description>             <uri>http://com/darkhe/sample/mycompass/alias/Summary</uri>             <name>Summary</name>         </meta-data>                  <meta-data id="Content" displayName="Content">             <description>Content</description>             <uri>http://com/darkhe/sample/mycompass/alias/Content</uri>             <name>Content</name>         </meta-data>          <meta-data id="KeyList" displayName="KeyList">             <description>KeyList</description>             <uri>http://com/darkhe/sample/mycompass/alias/KeyList</uri>             <name>KeyList</name>         </meta-data>                                      </meta-data-group>      </compass-core-meta-data>  11. 再是mycompass.cpm.xml <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE compass-core-mapping PUBLIC     "-//Compass/Compass Core Mapping DTD 1.0//EN"     "<http://www.opensymphony.com/compass/dtd/compass-core-mapping.dtd>">  <!-- 这里的包名必须和pojo的包名一致 --> <compass-core-mapping package="com.darkhe.sample.mycompass">  <!-- 定义实体及其字段的对应关系 -->  <!-- 注意实体及其字段的名称的大小写应当与pojo对象一致，而不是与数据库一致   关于pojo与数据库的对应表的一致性关系由hibernate的映谢文件定义，而不是这个文件   当前映射文件只定义compass与hibernate的关系 -->   <class name="Article" alias="${mycompass.Article}">   <id name="ArticleId" />      <property name="ArticleTitle">    <meta-data>${mycompass.ArticleTitle}</meta-data>   </property>    <property name="PublishDate">    <meta-data>${mycompass.PublishDate}</meta-data>   </property>    <property name="Summary">    <meta-data>${mycompass.Summary}</meta-data>   </property>    <property name="Content">    <meta-data>${mycompass.Content}</meta-data>   </property>    <property name="KeyList">    <meta-data>${mycompass.KeyList}</meta-data>   </property>  </class>  </compass-core-mapping>  12. log4j.properties log4j.rootLogger=WARN, stdout log4j.appender.stdout=org.apache.log4j.ConsoleAppender log4j.appender.stdout.layout=org.apache.log4j.PatternLayout log4j.appender.stdout.layout.ConversionPattern=%d %p %c - %m%n log4j.logger.org.compass=INFO  13. jdbc.properties # Properties file with JDBC-related settings. # Applied by PropertyPlaceholderConfigurer from "applicationContext-\*.xml". # Targeted at system administrators, to avoid touching the context XML files. jdbc.driverClassName=com.mysql.jdbc.Driver #jdbc.driverClassName=org.hsqldb.jdbcDriver #jdbc.url=jdbc:hsqldb:hsql://localhost:9001 jdbc.url=jdbc:mysql://localhost:3306/testdb jdbc.username=test jdbc.password=test # Property that determines the Hibernate dialect # (only applied with "applicationContext-hibernate.xml") #hibernate.dialect=org.hibernate.dialect.HSQLDialect hibernate.dialect=org.hibernate.dialect.MySQLDialect  14. 最后是applicationContext-hibernate.xml，这里集中配置了compass如何与spring与hibernate结合的。 <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "<http://www.springframework.org/dtd/spring-beans.dtd>">  <!--  - Application context definition for Petclinic on Hibernate. --> <beans>  <!-- ========================= RESOURCE DEFINITIONS ========================= -->  <!-- Configurer that replaces ${...} placeholders with values from a properties file -->  <!-- (in this case, JDBC-related settings for the dataSource definition below) -->  <bean id="propertyConfigurer"   class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">   <property name="location">    <value>classpath:jdbc.properties</value>   </property>  </bean>  <!-- Local DataSource that works in any environment -->  <!-- Note that DriverManagerDataSource does not pool; it is not intended for production -->  <!-- See JPetStore for an example of using Commons DBCP BasicDataSource as alternative -->  <!-- See Image Database for an example of using C3P0 ComboPooledDataSource as alternative -->  <bean id="dataSource"   class="org.springframework.jdbc.datasource.DriverManagerDataSource">   <property name="driverClassName">    <value>${jdbc.driverClassName}</value>   </property>   <property name="url">    <value>${jdbc.url}</value>   </property>   <property name="username">    <value>${jdbc.username}</value>   </property>   <property name="password">    <value>${jdbc.password}</value>   </property>  </bean>  <!-- JNDI DataSource for J2EE environments -->  <!--   <bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">   <property name="jndiName"><value>java:comp/env/jdbc/petclinic</value></property>   </bean>  -->  <!-- Hibernate SessionFactory -->  <bean id="sessionFactory"   class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">   <property name="dataSource">    <ref local="dataSource" />   </property>   <property name="mappingResources">    <list>     <value>      com/darkhe/sample/mycompass/Article.hbm.xml <!-- 这里是hibernate里需要的数据映射文件 -->     </value>    </list>   </property>   <property name="hibernateProperties">    <props>     <prop key="hibernate.dialect">      ${hibernate.dialect}     </prop>     <prop key="hibernate.show\_sql">false</prop>     <prop key="hibernate.generate\_statistics">true</prop>    </props>   </property>   <property name="eventListeners">    <map>     <entry key="merge">      <bean       class="org.springframework.orm.hibernate3.support.IdTransferringMergeEventListener" />     </entry>    </map>   </property>  </bean>   <!-- COMPASS START -->  <bean id="compass" class="org.compass.spring.LocalCompassBean">   <property name="resourceLocations">    <list>     <value>classpath:mycompass.cmd.xml</value> <!-- 这里是compass所需要的两个关于数据项的配置文件 -->     <value>classpath:mycompass.cpm.xml</value>    </list>   </property>   <property name="configLocation">    <value>classpath:mycompass.cfg.xml</value> <!-- 这里是compass的系统配置文件的路径 -->   </property>   <!--         <property name="compassSettings">    <props>    <prop key="compass.engine.connection">file://d:/target</prop>    <prop key="compass.transaction.factory">org.compass.spring.transaction.SpringSyncTransactionFactory</prop>    </props>    </property>-->    <property name="transactionManager">    <ref local="transactionManager" />   </property>  </bean>   <bean id="hibernateGpsDevice"   class="org.compass.spring.device.hibernate.SpringHibernate3GpsDevice">   <property name="name">    <value>hibernateDevice</value>   </property>   <property name="sessionFactory">    <ref local="sessionFactory" />   </property>  </bean>  <bean id="compassGps" class="org.compass.gps.impl.SingleCompassGps"   init-method="start" destroy-method="stop">   <property name="compass">    <ref bean="compass" />   </property>   <property name="gpsDevices">    <list>     <bean      class="org.compass.spring.device.SpringSyncTransactionGpsDeviceWrapper">      <property name="gpsDevice" ref="hibernateGpsDevice" />     </bean>    </list>   </property>  </bean>  <!-- COMPASS END --> </beans>  15. 注意上面的所以配置文件，根据我们上面的配置，都应当放到classpath的根路径。 16. 建立工具类，用来进行spring引擎的初始化工作。 /\*\*  \* <p>@(#) IOC.java 2006-2-1 0:08:23</p>  \* <p>Copyright (c) 2005-2006 ???????????????????</p>  \*/ package com.darkhe.sample.mycompass;  import org.springframework.context.ApplicationContext; import org.springframework.context.support.ClassPathXmlApplicationContext;  /\*\*  \*   \*   \* @version 1.0 2006-2-1  \* @author darkhe  \*/ public class IOC {  private static ApplicationContext context = null;   private static boolean isInit = false;   private IOC() {   super();  }   private static void init() {     if (isInit == false) {    String[] xmlfilenames = { "applicationContext-hibernate.xml" };     context = new ClassPathXmlApplicationContext(xmlfilenames);     isInit = true;   }  }   /\*\*   \*     \* @return   \*/  public static ApplicationContext getContext() {   if (context == null || isInit == false) {    init();   }   return context;  }   /\*\*   \*    \* @param name   \* @return   \*/  public static Object getBean(String name) {   return getContext().getBean(name);  }  }  17. 建立索引程序，用来数据库中的建立索引 /\*  \* Copyright (c) 2005-2006   \* ChongQing Man-Month Technology Development Co. ,Ltd  \*   \* ---------------------------------------------------------------------------------  \* @(#) Inder.java, 2006-8-1 下午09:01:14  \* ---------------------------------------------------------------------------------  \*/ package com.darkhe.sample.mycompass;  import java.io.FileNotFoundException;  import org.compass.gps.CompassGps; import org.springframework.context.ApplicationContext;  /\*\*  \* @author darkhe  \*   \*/ public class Indexer {   /\*\*   \* @param args   \* @throws FileNotFoundException    \*/  public static void main(String[] args) throws FileNotFoundException {     // 加裁自定义词典   DictionaryUtils.loadCustomDictionary();    ApplicationContext context = IOC.getContext();    // 得到spring环境中已经配置和初始化好的compassGps对象   CompassGps compassGps = (CompassGps) context.getBean("compassGps");   // 调用index方法建立索引   compassGps.index();   }  }  18. 建立搜索程序，检证compass的应用。 /\*  \* Copyright (c) 2005-2006   \* ChongQing Man-Month Technology Development Co. ,Ltd  \*   \* ---------------------------------------------------------------------------------  \* @(#) Searcher.java, 2006-8-1 下午09:36:29  \* ---------------------------------------------------------------------------------  \*/  package com.darkhe.sample.mycompass;  import java.io.FileNotFoundException;  import org.compass.core.Compass; import org.compass.core.CompassCallbackWithoutResult; import org.compass.core.CompassException; import org.compass.core.CompassHits; import org.compass.core.CompassSession; import org.compass.core.CompassTemplate; import org.compass.core.Resource; import org.springframework.context.ApplicationContext;  /\*\*  \* @author darkhe  \*   \*/ public class Searcher {   /\*\*   \* @param args   \* @throws FileNotFoundException   \*/  public static void main(String[] args) throws FileNotFoundException {    // 加裁自定义词典   DictionaryUtils.loadCustomDictionary();    ApplicationContext context = IOC.getContext();    Compass compass = (Compass) context.getBean("compass");    CompassTemplate template = new CompassTemplate(compass);    template.execute(new CompassCallbackWithoutResult() {    protected void doInCompassWithoutResult(CompassSession session)      throws CompassException {     CompassHits hits = session.find("大头人");      System.out.println("Found [" + hits.getLength()       + "] hits for [大头人] query");     System.out       .println("======================================================");     for (int i = 0; i < hits.getLength(); i++) {      print(hits, i);     }      hits.close();    }   });   }   public static void print(CompassHits hits, int hitNumber) {   Object value = hits.data(hitNumber);   Resource resource = hits.resource(hitNumber);   System.out.println("ALIAS [" + resource.getAlias() + "]  SCORE ["     + hits.score(hitNumber) + "]");   System.out.println(":::: " + value);   System.out.println("");  } }  19. 工具类DictionaryUtils是用来管理我们自己采用的中文分词算法的加载自定义词典的。 /\*\*  \* Copyright (c) 2005-2006 重庆人月科技发展有限公司  \*   \* ------------------------------------------------------------------------------  \* @(#) DictionaryUtils.java, 2006-8-2 下午04:55:22  \* ------------------------------------------------------------------------------  \*/ package com.darkhe.sample.mycompass;  import java.io.File; import java.io.FileNotFoundException; import java.io.FileReader;  import jeasy.analysis.MMAnalyzer;  /\*\*  \*   \* @author darkhe  \* @version 1.0.0  \*/ public class DictionaryUtils {  // 静态变量  private static boolean isInit = false;   // 静态初始化   // 静态方法  public static void loadCustomDictionary() throws FileNotFoundException {    if (isInit == false) {     // 添加我们自己的词典    FileReader fr = new FileReader(new File("dict.txt"));    MMAnalyzer.addDictionary(fr);        //System.out.println("添加我们自己的词典");     isInit = true;   }  } } 20. 执行Indexer，再执行Seracher后控制台信息如下：  Found [1] hits for [大头人] query ================================================ ALIAS [Article] SCORE [0.3988277] :::: [com.darkhe.sample.mycompass.Article@bla4e2](mailto:com.darkhe.sample.mycompass.Article@bla4e2)  具体结果和你的数据表中的内容有别。  21. 这样，我们便实现了如何利用compass构建我们自己的搜索引擎的一个简单实现。 22. 大家去试试吧，呵呵，有问题欢迎交流 [dark\_he@hotmail.com](mailto:dark_he@hotmail.com) |