1. 项目源程序：
2. <div id=*"divider"*><div></div></div>

<span class=*"left"*>Version @APPVERSION@

<c:if test="${pageContext.request.remoteUser != null}">

| <fmt:message key=*"user.status"*/> <authz:authentication operation=*"fullName"*/>

</c:if>

</span>

<span class=*"right"*>

&copy; @COPYRIGHT-YEAR@ Hon Hai Precision Industry, Co., Ltd. ("Foxconn").All rights reserved.

</span>

相关技术

jstl之<fmt:requestEncoding>、<fmt:message>

2007-10-18 19:50

|  |
| --- |
| ***http://hiphotos.baidu.com/javajavajava/pic/item/31b201f3231a655b342acc3f.jpg***  ***fmt.jsp***  <%@ page language="java" contentType="text/html; charset=gb18030"%> <%@ taglib uri="<http://java.sun.com/jsp/jstl/fmt>" prefix="fmt" %> <%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c" %> <html>         <head>           <title>My JSP 'fmt.jsp' starting page</title>         </head>            <body>          <form action="**fmt.jsp**" method="post"><-- 注意,这里要用post,用get有乱码-->          <fmt:message key="label.name"/><input type="text" name="name"><br>         <fmt:message key="label.password"/><input type="password" name="password"><br>          <input type="submit" value="<fmt:message key="button.submit"/>">          <input type="reset" value="<fmt:message key="button.reset"/>">          </form>          <hr>    <fmt:requestEncoding value="gb2312"/>           <fmt:message key="label.name"/><c:out value="${param.name }"></c:out><br>         <fmt:message key="label.password"/><c:out value="${param.password }"></c:out>     </body> </html>  ***resources.properties***  label.name=姓名 label.password=密码 button.submit=提交 button.reset=重置  **web.xml**  <context-param>         <param-name>javax.servlet.jsp.jstl.fmt.localizationContext</param-name>         <param-value>resources</param-value>        </context-param> |

2．<menu:useMenuDisplayer name=*"Velocity"* config=*"WEB-INF/classes/cssHorizontalMenu.vm"* permissions=*"rolesAdapter"*>

<ul id=*"primary-nav"* class=*"menuList"*>

<menu:displayMenu name=*"SearchMenu"*/>

<c:if test="${**empty** pageContext.request.remoteUser}">

<menu:displayMenu name=*"Login"*/>

</c:if>

<c:if test="${**not empty** pageContext.request.remoteUser}">

<menu:displayMenu name=*"UserMenu"*/>

<!--Website-START-->

<menu:displayMenu name=*"WebsiteMenu"*/>

<!--Website-END-->

<!--Material-START-->

<menu:displayMenu name=*"MaterialMenu"*/>

<!--Material-END-->

<!--Quoter-START-->

<menu:displayMenu name=*"QuoterMenu"*/>

<!--Quoter-END-->

<menu:displayMenu name=*"AdminMenu"*/>

<menu:displayMenu name=*"Logout"*/>

</c:if>

</ul>

</menu:useMenuDisplayer>

3. <c:if test="${**not empty** successMessages}">

<div class=*"message"* id=*"successMessages"*>

<c:forEach var=*"msg"* items="${successMessages}">

<img src="<c:url value=*"/images/iconInformation.gif"*/>"

alt="<fmt:message key=*"icon.information"*/>" class=*"icon"* />

<c:out value="${msg}" escapeXml=*"false"*/><br />

</c:forEach>

</div>

<c:remove var=*"successMessages"* scope=*"session"*/>

</c:if>

相關信息：

**<c:remove>**

　　语法：

　　<c:remove var="varName" [scope="{ page|request|session|application }"] />

　　<c:remove>标签用于删除数据，它有如下属性：

　　名 称

　　说 明

　　EL

　　类型

　　必须

　　默认值

　　var

　　欲移除的变量名称

　　N

　　String

　　是

　　无

　　scope

　　var变量的JSP范围

　　N

　　String

　　否

　　page

　　scope范围，包括page、request、session、application等；

　　例:

　　<c:remove var="test2" scope="session"/>

　　从session中删除test2变量。

　　教材例程15-4，c\_remove.jsp,<c:remove>标签的应用。

　　<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>

　　<%@ page contentType="text/html; charset=gb2312" language="java" %>

　　<html>

　　<head>

　　<title>JSTL:c:remove的使用</title>

　　</head>

　　<body bgcolor="#FFFFFF">

　　<c:set value="10000" var="maxUser" scope="application"/>

　　<c:set value="20" var="maxIdelTime" scope="session"/>

　　<c:set value="hellking" var="accountId" scope="request"/>

　　<c:set value="next.jsp" var="nextPage" scope="page"/>

　　在没有调用c:rmeove之前，有这些参数：

　　maxUser=<c:out value="${maxUser}"/>，

　　maxIdelTime=<c:out value="${maxIdelTime}"/>，

　　accountId=<c:out value="${accountId}"/>，

　　nextPage=<c:out value="${nextPage}"/>。

　　<hr>调用c:remove...

　　<c:remove var="maxUser" scope="application"/>

　　<c:remove var="maxIdelTime" scope="session"/>

　　<c:remove var="accountId" scope="request"/>

　　<c:remove var="nextPage" scope="page"/>

　　调用了c:remove后，这些参数值为：

　　maxUser=<c:out value="${maxUser}"/>，

　　maxIdelTime=<c:out value="${maxIdelTime}"/>，

　　accountId=<c:out value="${accountId}"/>，

　　nextPage=<c:out value="${nextPage}"/>，

　　</body>

　　</html>

4. JSTL核心库常用标签举例（详解）

2009-04-09 10:06

|  |
| --- |
| 小提示：结合上一篇JSTL核心库常用标签举例（简例）更容易阅读以下内容  **struts-config.xml：**  <?xml version="1.0" encoding="ISO-8859-1" ?> <!DOCTYPE struts-config PUBLIC           "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"           "<http://struts.apache.org/dtds/struts-config_1_3.dtd>">  <struts-config>  <action-mappings>       <action path="/jstlcore"      type="com.yulin.struts.JSTLCoreAction"      >      <forward name="success" path="/jstlcore.jsp"></forward>    </action>     </action-mappings>  <message-resources parameter="MessageResources" /> </struts-config>  **演示的JSP页面(jstlcore.jsp)：**  <%@ page language="java" contentType="text/html; charset=GB18030"     pageEncoding="GB18030"%> <%@ taglib prefix="c" uri="<http://java.sun.com/jsp/jstl/core>" %> <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "<http://www.w3.org/TR/html4/loose.dtd>"> <html> <head> <meta http-equiv="Content-Type" content="text/html; charset=GB18030"> <title>Test JSTL</title> </head> <body> <h1>Test JSTL</h1> <hr> <li>TEST c:out</li><hr> core(el): ${core }<br> core(el.empty): ${empty core }<br> core(default):<c:out value="${core}"> </c:out> <br> core(default):<c:out value="${core1}" > </c:out> <br> core(default="123456"):<c:out value="${core1}" default="123456" ></c:out> <br> <hr> html(el): ${core2 }<br> html(el.requestScope): ${requestScope.core2 }<br> html(default):<c:out value="${core2}"> </c:out> <br> html(escapeXml="true"):<c:out value="${core2}" escapeXml="true" ></c:out> <br> html(escapeXml="false"):<c:out value="${core2}" escapeXml="false" ></c:out> <br> html(default="<font color='red'>输出html文件</font>"):<c:out value="${core3}" default="<font color='red'>输出html文件</font>"></c:out> <br> html(default="<font color='red'>输出html文件</font>"escapeXml="false"):<c:out value="${core3}" default="<font color='red'>输出html文件</font>" escapeXml="false"></c:out> <br> <hr> <li>TEST c:set c:remove</li><hr> <c:set var="temp" value="1000n"> </c:set> temp(c:set): ${temp }<br> <c:remove var="temp"/> temp(c:remove): ${temp }<br> <hr> <li>TEST c:if</li><hr> <c:if test="${v1 lt v2}" var="r"> v1小于v2,${r }<br> </c:if> <c:if test="${v2 gt v1}" var="r"> v2da于v1,${r }<br> </c:if> <hr> <li>TEST c:choose,c:when,c:otherwise</li><hr> <c:choose> <c:when test="${v1 lt v2}">    v1小于v2<br> </c:when> </c:choose> <c:choose> <c:when test="${v1 gt v2}">    v1da于v2<br> </c:when> <c:otherwise>    v1xiaoyuv2<br> </c:otherwise> </c:choose> <hr> <li>TEST c:forEach varStatus="vs" begin="4" end="40" step="5" </li><hr> <table border="1"> <tr bgcolor="yellow">    <td>Username</td>    <td>Age</td>    <td>Group</td> </tr> <c:choose>    <c:when test="${empty userList}">    <tr>     <td clospan="3">there is no info!</td>    </tr>    </c:when>    <c:otherwise>     <c:forEach items="${userList}" var="user" varStatus="vs" begin="4" end="40" step="5">      <c:choose>       <c:when test="${vs.count % 2 == 0}">        <c:set var="str" value="bgcolor=green" />       </c:when>       <c:otherwise>        <c:set var="str" value="bgcolor=red" />       </c:otherwise>      </c:choose>      <tr ${str }>       <td>${user.username }</td>       <td>${user.age }</td>       <td>${user.group.name }</td>      </tr>    </c:forEach>    </c:otherwise> </c:choose> </table> <hr> <li>TEST c:fortokens delims="," var="f"</li><hr> <c:forTokens items="${fortokens}" delims="," var="f"> ${f }<br> </c:forTokens> <hr> <li>TEST c:catch</li><hr> <c:catch var="e"> <% Integer.parseInt("sss"); %> </c:catch> ${e }<br> <hr> <li>TEST c:import </li><hr> <c:import url="[http://localhost:8080/struts-taglib/index.jsp"/](http://localhost:8080/struts-taglib/index.jsp%22/)> <hr> <li>TEST c:url c:param </li><hr> <c:url value="<http://localhost:8080/struts-taglib/index.jsp>" var="turl"> <c:param name="id" value="01" /> <c:param name="name" value="linzhang"/> </c:url> ${turl }<br> <hr> <li>TEST c:redirect [context]url </li><hr>  </body> </html>  **所需的JSTLCoreAction页面(JSTLCoreAction.java):**  package com.yulin.struts;  import java.util.ArrayList; import java.util.List;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.apache.struts.action.Action; import org.apache.struts.action.ActionForm; import org.apache.struts.action.ActionForward; import org.apache.struts.action.ActionMapping;  public class JSTLCoreAction extends Action {  @Override public ActionForward execute(ActionMapping mapping, ActionForm form,     HttpServletRequest request, HttpServletResponse response)     throws Exception {    request.setAttribute("core", "core");    request.setAttribute("core1", "");    //输出html文件    request.setAttribute("core2", "<font color='red'>输出html文件</font>");    request.setAttribute("core3", "");    request.setAttribute("v1", 1);    request.setAttribute("v2", 2);    request.setAttribute("v3", "");       List userList = new ArrayList();    for(int i=0;i<50;i++){     Group g = new Group();     g.setName("China");     User u = new User();     u.setUsername("LinZhang" + i);     u.setAge(23);     u.setGroup(g);     userList.add(u);    }    request.setAttribute("userList", userList);       request.setAttribute("fortokens", "a,b,c,d,e,f,g");    return mapping.findForward("success"); }  }  **涉及到的User（User.java）和Group(Group.java)页面：**  ***User.java:***  package com.yulin.struts;  public class User { private String username; private int age; private Group group; public String getUsername() {    return username; } public void setUsername(String username) {    this.username = username; } public int getAge() {    return age; } public void setAge(int age) {    this.age = age; } public Group getGroup() {    return group; } public void setGroup(Group group) {    this.group = group; } }  **Group.java:**  package com.yulin.struts;  public class Group { private String name;  public String getName() {    return name; }  public void setName(String name) {    this.name = name; } } |

## [COMPASS介绍](http://www.cnblogs.com/zhangjunpo/articles/1336841.html)

在新架构中打算选择Compass或Hibernate Search作为搜索引擎框架，比较后，感觉Hibernate Search上还是没有Compass成熟，另外考虑到后期对网页的爬取及搜索需求，决定还是基于Compass来作为架构缺省的搜索引擎。网上关于 Compass的文档很多，但说得相对完整其详细的入门文档基本上没有，Compass的官方文档倒是说得很详细，但是例子一塌糊涂，存在很大问题。记录 一下搭建的过程，作为入门的指南。   
  
    Compass 通过OSEM(Object/Search Engine Mapping)允许把应用对象的领域模型映射到搜索引擎，最终通过访问common meta data来达到访问对象的目的。   
1、几个核心概念   
  
1.1、annotation vs. xml配置文件   
  
   Compass的配置文件主要分成三类：   
  
    第一类：\*.cmd.xml文件\*   
  
      .cmd.xml文件是对common meta data进行定义，定义了最终搜索的结果中的最基本的元数据。   
  
    第二类：\*.cpm.xml文件   
  
      \*.cpm.xml是Object/Search Engine Mapping,提供了POJO到common meta data的映射。   
  
    第三类：\*.cfg.xml文件   
  
      Compass的\*.cfg.xml定义了Compass的Index存放路径、搜索引擎分词等相关信息。   
  
    与采用xml配置文件相比较，采用Annonation方式还是相对简单，尤其是采用Spring时候，不用写\*.cmd.xml文 件、\*.cpm.xml、\*.cfg.xml，相对很方便，而且不像Hibernate的Annonation很多，Compass的 Annonation的核心标注只有@Searchable、@SearchableId、@SearchableProperty、 @SearchableComponent个，很容易记忆。因此推荐使用Annonation方式   
１.2、Compass核心API   
  
Compass的核心API借鉴了Hibernate的术语，因此在操作上基本上与Hibernate类似，以下为Compass的几个核心接口：   
  
    CompassConfiguration(类似Hibernate Configuration)：用来在一些设置参数、配置文件和映射定义上配置Compass。通常用来创建Compass接口。   
    Compass(类似Hibernate SessionFactory)：为单线程使用，创建线程安全的实例来打开Compass Seesion。同样还提供了一些搜索引擎索引级别的操作。   
    CompassSesssion(类似Hibernate Session)：用来执行像保存、删除、查找、装载这样的搜索操作。很轻量但是并不是线程安全的。   
    CompassTransaction(类似Hibernate Transaction)：管理Compass事务的接口。使用它并不需要事务管理环境（像Spring、JTA）。   
  
1.3、Compass与Spring集成   
  
Compass 已经对对spring集成做了很好的封装，同时与Spring对Hibernate的支持类似，Compass也提供了CompassTemplate来 简化诸如对Session、Transaction、Exception等操作，尽量充分使用此工具，可以有效提高效率。例如：   
  
CompassTemplate ct = (CompassTemplate) context.getBean("compassTemplate");   
  
Article article = new Article();   
article.setTitle("Compass Test");   
article.setPublishDate(new Date());   
article.setAuthor(1);   
  
ct.save(article); //存储对象需要索引的数据到Compass的索引中。   
  
  
2、软件环境   
  
Spring ：2.5   
  
Compas：1.2.1   
  
Hibernate：3.2.5   
  
Mysql ：5.0.5   
3、数据库脚本   
  
CREATE TABLE `article` (   
  
`Id` int(11) NOT NULL auto\_increment,   
  
`title` varchar(40) NOT NULL default '',   
  
`author` int(11) default '0',   
  
`publish\_date` date NOT NULL default '0000-00-00',   
  
PRIMARY KEY (`Id`) ) TYPE=MyISAM;   
  
CREATE TABLE `author` (   
  
`Id` int(11) NOT NULL auto\_increment,   
  
`username` varchar(20) NOT NULL default '',   
  
`password` varchar(20) NOT NULL default '',   
  
`age` smallint(6) default '0',   
  
PRIMARY KEY (`Id`) ) TYPE=MyISAM;   
  
4、测试用例   
  
从测试用例讲起比较容易把关系理清楚，不然一堆术语和概念很让人晕乎。

Java代码

1. **import** org.apache.log4j.Logger;
2. **import** java.util.Date;
4. **import** junit.framework.TestCase;
6. **import** org.compass.core.Compass;
7. **import** org.compass.core.CompassDetachedHits;
8. **import** org.compass.core.CompassHit;
9. **import** org.compass.core.CompassHits;
10. **import** org.compass.core.CompassSession;
11. **import** org.compass.core.CompassTemplate;
12. **import** org.compass.core.CompassTransaction;
13. **import** org.compass.core.support.search.CompassSearchCommand;
14. **import** org.compass.core.support.search.CompassSearchResults;
15. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
17. **import** com.mobilesoft.esales.dao.hibernate.ArticleDAO;
18. **import** com.mobilesoft.esales.dao.hibernate.AuthorDAO;
19. **import** com.mobilesoft.esales.model.Article;
20. **import** com.mobilesoft.esales.model.Author;
21. **import** com.mobilesoft.framework.search.service.CompassSearchService;
23. /\*\*
24. \* Compass服务使用的测试用例
25. \*
26. \* @author liangchuan@mobile-soft.cn
27. \*
28. \*/
30. **public** **class** TestCompass **extends** TestCase {
32. **private** **static** **final** Logger logger = Logger.getLogger(TestCompass.**class**);
34. **private** **static** ClassPathXmlApplicationContext context = **null**;
35. **private** **static** CompassTemplate ct;
37. **static** {
38. context = **new** ClassPathXmlApplicationContext(**new** String[] {
39. "applicationContext.xml", "applicationContext-resources.xml",
40. "applicationContext-dao.xml", "applicationContext-service.xml",
41. "applicationContext-compass.xml" });
42. ct = (CompassTemplate) context.getBean("compassTemplate");
43. }
45. **protected** **void** setUp() **throws** Exception {
47. }
49. /\*\*
50. \* 插入测试数据
51. \*/
52. **public** **void** testInsert() {
54. ArticleDAO articleDao = (ArticleDAO) context.getBean("articleDAO");
55. AuthorDAO authorDao = (AuthorDAO) context.getBean("authorDAO");
56. Article article = **new** Article();
57. Author author = **new** Author();
58. author.setAge((**short**) 27);
59. author.setUsername("liangchuan");
60. author.setPassword("liangchuan");
61. article.setTitle("Compass Test");
62. article.setPublishDate(**new** Date());
63. article.setAuthor(1);
64. authorDao.save(author);
65. articleDao.save(article);
66. ct.save(article);
67. ct.save(author);
68. }
70. /\*\*
71. \* 用于测试使用CompassTransaction事务方式
72. \*/
73. **public** **void** testTransactionalFind() {
75. Compass compass = ct.getCompass();
76. CompassSession session = compass.openSession();
77. CompassTransaction tx = **null**;
78. **try** {
79. tx = session.beginTransaction();
80. CompassHits hits = session.find("Compass\*");
82. logger.error("testTransactionalFind() - CompassHits hits="
83. + hits.getLength());
84. **for** (**int** i = 0; i < hits.getLength(); i++) {
85. Object hit = hits.data(i);
86. **if** (hit **instanceof** Article) {
87. Article item = (Article) hit;
88. logger.error("testTransactionalFind() - article     hits="
89. + item.getTitle());
90. } **else** **if** (hit **instanceof** Author) {
91. Author item = (Author) hit;
92. logger.error("testTransactionalFind() - author hits="
93. + item.getUsername());
94. } **else** {
95. logger.error("testTransactionalFind() - error hits=");
96. }
97. }
98. tx.commit();
99. } **catch** (Exception e) {
100. **if** (tx != **null**) {
101. tx.rollback();
102. }
103. } **finally** {
104. session.close();
105. }
106. }
108. /\*\*
109. \* 用于演示CompassDetachedHits的使用。
110. \* 由于CompassTempalte得到的结果集必须在transactionalcontext中才能使用，
111. \* 因此必须使用CompassDetachedHits方式测试CompassDetachedHits方式
112. \*/
113. **public** **void** testDetachedFind() {
115. // 由于CompassTempalte得到的结果集必须在transactional
116. // context中才能使用，因此必须使用CompassDetachedHits方式
117. // 测试CompassDetachedHits方式
118. CompassDetachedHits hits = ct.findWithDetach("Compass\*");
120. logger.error("testDetachedFind() - CompassHits hits="
121. + hits.getLength());
122. **for** (**int** i = 0; i < hits.getLength(); i++) {
123. Object hit = hits.data(i);
124. **if** (hit **instanceof** Article) {
125. Article item = (Article) hit;
126. logger.error("testDetachedFind() - article     hits="
127. + item.getTitle());
128. } **else** **if** (hit **instanceof** Author) {
129. Author item = (Author) hit;
130. logger.error("testDetachedFind() - author hits="
131. + item.getUsername());
132. } **else** {
133. logger.error("testDetachedFind() - error hits=");
134. }
135. }
137. }
139. /\*\*
140. \* 用于演示com.mobilesoft.framework.search.service.CompassSearchService的使用
141. \*
142. \*/
143. **class** CompassSearch **extends** CompassSearchService{
144. CompassSearch(){
145. Compass compass = ct.getCompass();
146. CompassSession session = compass.openSession();
147. CompassTransaction tx = **null**;
149. **try** {
150. tx = session.beginTransaction();
151. CompassSearchCommand command = **new** CompassSearchCommand();
152. command.setQuery("Compass");
153. CompassSearchResults results= performSearch(command,session);
154. logger.error("CompassSearch() - CompassHit TotalHits value=" +results.getTotalHits());
156. **for** (**int** i = 0; i < results.getHits().length; i++) {
157. CompassHit hits=results.getHits()[i];
158. Object hit=hits.getData();
159. logger.error("CompassSearch() - CompassHit hit=" + hit); //$NON-NLS-1$
161. **if** (hit **instanceof** Article) {
162. Article item = (Article) hit;
163. logger.error("testCompassSearchService() - article     hits="
164. + item.getTitle());
165. } **else** **if** (hit **instanceof** Author) {
166. Author item = (Author) hit;
167. logger.error("testCompassSearchService() - author hits="
168. + item.getUsername());
169. } **else** {
170. logger.error("testCompassSearchService() - error hits=");
171. }
173. tx.commit();
174. }
175. } **catch** (Exception e) {
176. **if** (tx != **null**) {
177. tx.rollback();
178. }
179. } **finally** {
180. session.close();
181. }
183. }
185. }
186. **public** **void** testCompassSearchService() {
187. **new** CompassSearch();
188. }
190. **protected** **void** tearDown() **throws** Exception {
191. }
192. }

import org.apache.log4j.Logger;

import java.util.Date;

import junit.framework.TestCase;

import org.compass.core.Compass;

import org.compass.core.CompassDetachedHits;

import org.compass.core.CompassHit;

import org.compass.core.CompassHits;

import org.compass.core.CompassSession;

import org.compass.core.CompassTemplate;

import org.compass.core.CompassTransaction;

import org.compass.core.support.search.CompassSearchCommand;

import org.compass.core.support.search.CompassSearchResults;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.mobilesoft.esales.dao.hibernate.ArticleDAO;

import com.mobilesoft.esales.dao.hibernate.AuthorDAO;

import com.mobilesoft.esales.model.Article;

import com.mobilesoft.esales.model.Author;

import com.mobilesoft.framework.search.service.CompassSearchService;

/\*\*

\* Compass服务使用的测试用例

\*

\* @author liangchuan@mobile-soft.cn

\*

\*/

public class TestCompass extends TestCase {

private static final Logger logger = Logger.getLogger(TestCompass.class);

private static ClassPathXmlApplicationContext context = null;

private static CompassTemplate ct;

static {

context = new ClassPathXmlApplicationContext(new String[] {

"applicationContext.xml", "applicationContext-resources.xml",

"applicationContext-dao.xml", "applicationContext-service.xml",

"applicationContext-compass.xml" });

ct = (CompassTemplate) context.getBean("compassTemplate");

}

protected void setUp() throws Exception {

}

/\*\*

\* 插入测试数据

\*/

public void testInsert() {

ArticleDAO articleDao = (ArticleDAO) context.getBean("articleDAO");

AuthorDAO authorDao = (AuthorDAO) context.getBean("authorDAO");

Article article = new Article();

Author author = new Author();

author.setAge((short) 27);

author.setUsername("liangchuan");

author.setPassword("liangchuan");

article.setTitle("Compass Test");

article.setPublishDate(new Date());

article.setAuthor(1);

authorDao.save(author);

articleDao.save(article);

ct.save(article);

ct.save(author);

}

/\*\*

\* 用于测试使用CompassTransaction事务方式

\*/

public void testTransactionalFind() {

Compass compass = ct.getCompass();

CompassSession session = compass.openSession();

CompassTransaction tx = null;

try {

tx = session.beginTransaction();

CompassHits hits = session.find("Compass\*");

logger.error("testTransactionalFind() - CompassHits hits="

+ hits.getLength());

for (int i = 0; i < hits.getLength(); i++) {

Object hit = hits.data(i);

if (hit instanceof Article) {

Article item = (Article) hit;

logger.error("testTransactionalFind() - article hits="

+ item.getTitle());

} else if (hit instanceof Author) {

Author item = (Author) hit;

logger.error("testTransactionalFind() - author hits="

+ item.getUsername());

} else {

logger.error("testTransactionalFind() - error hits=");

}

}

tx.commit();

} catch (Exception e) {

if (tx != null) {

tx.rollback();

}

} finally {

session.close();

}

}

/\*\*

\* 用于演示CompassDetachedHits的使用。

\* 由于CompassTempalte得到的结果集必须在transactionalcontext中才能使用，

\* 因此必须使用CompassDetachedHits方式测试CompassDetachedHits方式

\*/

public void testDetachedFind() {

// 由于CompassTempalte得到的结果集必须在transactional

// context中才能使用，因此必须使用CompassDetachedHits方式

// 测试CompassDetachedHits方式

CompassDetachedHits hits = ct.findWithDetach("Compass\*");

logger.error("testDetachedFind() - CompassHits hits="

+ hits.getLength());

for (int i = 0; i < hits.getLength(); i++) {

Object hit = hits.data(i);

if (hit instanceof Article) {

Article item = (Article) hit;

logger.error("testDetachedFind() - article hits="

+ item.getTitle());

} else if (hit instanceof Author) {

Author item = (Author) hit;

logger.error("testDetachedFind() - author hits="

+ item.getUsername());

} else {

logger.error("testDetachedFind() - error hits=");

}

}

}

/\*\*

\* 用于演示com.mobilesoft.framework.search.service.CompassSearchService的使用

\*

\*/

class CompassSearch extends CompassSearchService{

CompassSearch(){

Compass compass = ct.getCompass();

CompassSession session = compass.openSession();

CompassTransaction tx = null;

try {

tx = session.beginTransaction();

CompassSearchCommand command = new CompassSearchCommand();

command.setQuery("Compass");

CompassSearchResults results= performSearch(command,session);

logger.error("CompassSearch() - CompassHit TotalHits value=" +results.getTotalHits());

for (int i = 0; i < results.getHits().length; i++) {

CompassHit hits=results.getHits()[i];

Object hit=hits.getData();

logger.error("CompassSearch() - CompassHit hit=" + hit); //$NON-NLS-1$

if (hit instanceof Article) {

Article item = (Article) hit;

logger.error("testCompassSearchService() - article hits="

+ item.getTitle());

} else if (hit instanceof Author) {

Author item = (Author) hit;

logger.error("testCompassSearchService() - author hits="

+ item.getUsername());

} else {

logger.error("testCompassSearchService() - error hits=");

}

tx.commit();

}

} catch (Exception e) {

if (tx != null) {

tx.rollback();

}

} finally {

session.close();

}

}

}

public void testCompassSearchService() {

new CompassSearch();

}

protected void tearDown() throws Exception {

}

}

5、配置文件   
applicationContext-compass.xml

Java代码

1. <?xml version="1.0"?>
2. <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"
3. "http://www.springframework.org/dtd/spring-beans-2.0.dtd">
4. <beans **default**-lazy-init="true">
5. <bean id="compassTemplate" **class**="org.compass.core.CompassTemplate">
6. <property name="compass" ref="compass"/>
7. </bean>
8. <bean id="annotationConfiguration"
9. **class**="org.compass.annotations.config.CompassAnnotationsConfiguration">
10. </bean>
11. <bean id="compass" **class**="org.compass.spring.LocalCompassBean">
12. <property name="classMappings">
13. <list>
14. <value>com.mobilesoft.esales.model.Article</value>
15. <value>com.mobilesoft.esales.model.Author</value>
16. </list>
17. </property>
18. <property name="compassConfiguration" ref="annotationConfiguration"/>
20. <property name="compassSettings">
21. <props>
22. <prop key="compass.engine.connection"> file://compass </prop>
23. <prop key="compass.transaction.factory">
24. org.compass.spring.transaction.SpringSyncTransactionFactory
25. </prop>
26. <prop
27. key="compass.engine.highlighter.default.formatter.simple.pre">
28. <![CDATA[<font color="red"><b>]]>
29. </prop>
30. <prop
31. key="compass.engine.highlighter.default.formatter.simple.post">
32. <![CDATA[</b></font>]]>
33. </prop>
34. </props>
35. </property>
37. <property name="transactionManager" ref="transactionManager"/>
38. </bean>
40. <bean id="hibernateGpsDevice"
41. **class**="org.compass.gps.device.hibernate.HibernateGpsDevice">
42. <property name="name">
43. <value>hibernateDevice</value>
44. </property>
45. <property name="sessionFactory" ref="sessionFactory"/>
46. <property name="mirrorDataChanges">
47. <value>**true**</value>
48. </property>
49. </bean>
50. <bean id="compassGps" **class**="org.compass.gps.impl.SingleCompassGps"
51. init-method="start" destroy-method="stop">
52. <property name="compass" ref="compass"/>
53. <property name="gpsDevices">
54. <list>
55. <bean
56. **class**="org.compass.spring.device.SpringSyncTransactionGpsDeviceWrapper">
57. <property name="gpsDevice" ref="hibernateGpsDevice"/>
58. </bean>
59. </list>
60. </property>
61. </bean>
62. <bean id="compassSearchService" **class**="com.mobilesoft.framework.search.service.CompassSearchService">
63. <property name="compass" ref="compass"/>
64. <property name="pageSize" value="15"/>
65. </bean>
67. <!-- 定时重建索引(利用quartz)或随Spring ApplicationContext启动而重建索引 -->
68. <bean id="compassIndexBuilder" **class**="com.mobilesoft.framework.search.service.CompassIndexBuilder" lazy-init="false">
69. <property name="compassGps" ref="compassGps"/>
70. <property name="buildIndex" value="false"/>
71. <property name="lazyTime" value="10"/>
72. </bean>

75. </beans>

<?xml version="1.0"?>

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"

"http://www.springframework.org/dtd/spring-beans-2.0.dtd">

<beans default-lazy-init="true">

<bean id="compassTemplate" class="org.compass.core.CompassTemplate">

<property name="compass" ref="compass"/>

</bean>

<bean id="annotationConfiguration"

class="org.compass.annotations.config.CompassAnnotationsConfiguration">

</bean>

<bean id="compass" class="org.compass.spring.LocalCompassBean">

<property name="classMappings">

<list>

<value>com.mobilesoft.esales.model.Article</value>

<value>com.mobilesoft.esales.model.Author</value>

</list>

</property>

<property name="compassConfiguration" ref="annotationConfiguration"/>

<property name="compassSettings">

<props>

<prop key="compass.engine.connection"> file://compass </prop>

<prop key="compass.transaction.factory">

org.compass.spring.transaction.SpringSyncTransactionFactory

</prop>

<prop

key="compass.engine.highlighter.default.formatter.simple.pre">

<![CDATA[<font color="red"><b>]]>

</prop>

<prop

key="compass.engine.highlighter.default.formatter.simple.post">

<![CDATA[</b></font>]]>

</prop>

</props>

</property>

<property name="transactionManager" ref="transactionManager"/>

</bean>

<bean id="hibernateGpsDevice"

class="org.compass.gps.device.hibernate.HibernateGpsDevice">

<property name="name">

<value>hibernateDevice</value>

</property>

<property name="sessionFactory" ref="sessionFactory"/>

<property name="mirrorDataChanges">

<value>true</value>

</property>

</bean>

<bean id="compassGps" class="org.compass.gps.impl.SingleCompassGps"

init-method="start" destroy-method="stop">

<property name="compass" ref="compass"/>

<property name="gpsDevices">

<list>

<bean

class="org.compass.spring.device.SpringSyncTransactionGpsDeviceWrapper">

<property name="gpsDevice" ref="hibernateGpsDevice"/>

</bean>

</list>

</property>

</bean>

<bean id="compassSearchService" class="com.mobilesoft.framework.search.service.CompassSearchService">

<property name="compass" ref="compass"/>

<property name="pageSize" value="15"/>

</bean>

<!-- 定时重建索引(利用quartz)或随Spring ApplicationContext启动而重建索引 -->

<bean id="compassIndexBuilder" class="com.mobilesoft.framework.search.service.CompassIndexBuilder" lazy-init="false">

<property name="compassGps" ref="compassGps"/>

<property name="buildIndex" value="false"/>

<property name="lazyTime" value="10"/>

</bean>

</beans>

applicationContext-dao.xml、applicationContext-service.xml、applicationContext-resources.xml等略去。   
  
  
6、Service层(参考了SpringSide实现)

Java代码

1. AdvancedSearchCommand.java
2. **package** com.mobilesoft.framework.search.service;
4. **import** java.util.HashSet;
5. **import** java.util.Set;
7. **import** org.apache.commons.lang.StringUtils;
8. **import** org.compass.core.CompassQuery.SortDirection;
9. **import** org.compass.core.CompassQuery.SortPropertyType;
10. **import** org.compass.core.support.search.CompassSearchCommand;
12. **import** org.springframework.util.Assert;
14. **public** **class** AdvancedSearchCommand **extends** CompassSearchCommand {
16. /\*\*
17. \* 封装基于Compass 的排序参数.
18. \*/
19. **class** CompassSort {
21. **private** String name;
23. **private** SortPropertyType type;
25. **private** SortDirection direction;
27. **public** CompassSort() {
28. }
30. **public** CompassSort(String sortParamName, String paramType,
31. **boolean** isAscend) {
32. Assert.isTrue(StringUtils.isNotBlank(sortParamName));
33. setName(sortParamName);
35. **if** ("int".equalsIgnoreCase(paramType)) {
36. setType(SortPropertyType.INT);
37. } **else** **if** ("float".equalsIgnoreCase(paramType)) {
38. setType(SortPropertyType.FLOAT);
39. } **else** **if** ("string".equalsIgnoreCase(paramType)) {
40. setType(SortPropertyType.STRING);
41. } **else** {
42. setType(SortPropertyType.AUTO);
43. }
45. **if** (isAscend) {
46. setDirection(SortDirection.AUTO);
47. } **else** {
48. setDirection(SortDirection.REVERSE);
49. }
50. }
52. **public** String getName() {
53. **return** name;
54. }
56. **public** **void** setName(String name) {
57. **this**.name = name;
58. }
60. **public** SortPropertyType getType() {
61. **return** type;
62. }
64. **public** **void** setType(SortPropertyType type) {
65. **this**.type = type;
66. }
68. **public** SortDirection getDirection() {
69. **return** direction;
70. }
72. **public** **void** setDirection(SortDirection direction) {
73. **this**.direction = direction;
74. }
75. }
77. /\*\*
78. \* 搜索结果排序表.
79. \*/
80. **private** Set<CompassSort> sortMap = **new** HashSet<CompassSort>();
82. **private** String[] highlightFields;
84. /\*\*
85. \* @param paramType 现定义了三种类型： int string 以及 float。<br>
86. \*                  除去这三种外，其他会被自动定义为SortPropertyType.AUTO 具体的可见{@link org.compass.core.CompassQuery.SortPropertyType}
87. \* @param isAscend  顺序还是倒序排序
88. \* @see org.compass.core.CompassQuery.SortPropertyType#AUTO
89. \* @see org.compass.core.CompassQuery.SortPropertyType#INT
90. \* @see org.compass.core.CompassQuery.SortPropertyType#STRING
91. \* @see org.compass.core.CompassQuery.SortPropertyType#FLOAT
92. \* @see org.compass.core.CompassQuery.SortDirection#AUTO
93. \* @see org.compass.core.CompassQuery.SortDirection#REVERSE
94. \*/
95. **public** **void** addSort(String sortParamName, String paramType, **boolean** isAscend) {
96. **this**.sortMap.add(**new** CompassSort(sortParamName, paramType, isAscend));
97. }
99. **public** Set<CompassSort> getSortMap() {
100. **return** sortMap;
101. }
103. **public** **void** setSortMap(Set<CompassSort> sortMap) {
104. **this**.sortMap = sortMap;
105. }
107. **public** String[] getHighlightFields() {
108. **return** highlightFields;
109. }
111. **public** **void** setHighlightFields(String[] highlightFields) {
112. **this**.highlightFields = highlightFields;
113. }
114. }
116. CompassIndexBuilder.java
118. **package** com.mobilesoft.framework.search.service;
120. **import** org.apache.log4j.Logger;
121. **import** org.compass.gps.CompassGps;
122. **import** org.springframework.beans.factory.InitializingBean;
123. **import** org.springframework.util.Assert;
125. /\*\*
126. \* 通过quartz定时调度定时重建索引或自动随Spring ApplicationContext启动而重建索引的Builder.
127. \* 会启动后延时数秒新开线程调用compassGps.index()函数.
128. \* 默认会在Web应用每次启动时重建索引,可以设置buildIndex属性为false来禁止此功能.
129. \* 也可以不用本Builder, 编写手动调用compassGps.index()的代码.
130. \*
131. \*/
132. **public** **class** CompassIndexBuilder **implements** InitializingBean {
134. **private** **static** **final** Logger log = Logger.getLogger(CompassIndexBuilder.**class**);
136. // 是否需要建立索引，可被设置为false使本Builder失效.
137. **private** **boolean** buildIndex = **false**;
139. // 索引操作线程延时启动的时间，单位为秒
140. **private** **int** lazyTime = 10;
142. // Compass封装
143. **private** CompassGps compassGps;
145. // 索引线程
146. **private** Thread indexThread = **new** Thread() {
148. @Override
149. **public** **void** run() {
150. **try** {
151. Thread.sleep(lazyTime \* 1000);
153. log.info("begin compass index...");
154. **long** beginTime = System.currentTimeMillis();
155. // 重建索引.
156. // 如果compass实体中定义的索引文件已存在，索引过程中会建立临时索引，
157. // 索引完成后再进行覆盖.
158. compassGps.index();
159. **long** costTime = System.currentTimeMillis() - beginTime;
160. log.info("compss index finished.");
161. log.info("costed " + costTime + " milliseconds");
162. } **catch** (InterruptedException e) {
163. // simply proceed
164. }
165. }
166. };
168. /\*\*
169. \* 实现<code>InitializingBean</code>接口，在完成注入后调用启动索引线程.
170. \*
171. \* @see org.springframework.beans.factory.InitializingBean#afterPropertiesSet()
172. \*/
173. **public** **void** afterPropertiesSet() **throws** Exception {
174. **if** (buildIndex) {
175. Assert.notNull(compassGps, "CompassIndexBuilder not set CompassGps yet.");
176. indexThread.setDaemon(**true**);
177. indexThread.setName("Compass Indexer");
178. indexThread.start();
179. }
180. }
182. **public** **void** setBuildIndex(**boolean** buildIndex) {
183. **this**.buildIndex = buildIndex;
184. }
186. **public** **void** setLazyTime(**int** lazyTime) {
187. **this**.lazyTime = lazyTime;
188. }
190. **public** **void** setCompassGps(CompassGps compassGps) {
191. **this**.compassGps = compassGps;
192. }
193. }
195. CompassSearchService.java
197. **package** com.mobilesoft.framework.search.service;
199. **import** org.compass.core.Compass;
200. **import** org.compass.core.CompassCallback;
201. **import** org.compass.core.CompassDetachedHits;
202. **import** org.compass.core.CompassHits;
203. **import** org.compass.core.CompassQuery;
204. **import** org.compass.core.CompassSession;
205. **import** org.compass.core.CompassTemplate;
206. **import** org.compass.core.CompassTransaction;
207. **import** org.compass.core.support.search.CompassSearchCommand;
208. **import** org.compass.core.support.search.CompassSearchResults;
210. **import** org.springframework.beans.factory.InitializingBean;
211. **import** org.springframework.util.Assert;
213. **import** com.mobilesoft.framework.search.service.AdvancedSearchCommand.CompassSort;
214. /\*\*
215. \* 仿照 {@link org.compass.spring.web.mvc.CompassSearchController}
216. \* 中的代码，构建了一个Service，方便不使用Spring MVC
217. \*
218. \* @see org.compass.spring.web.mvc.CompassSearchController
219. \* @see org.compass.spring.web.mvc.AbstractCompassCommandController
220. \*/
221. **public** **class** CompassSearchService **implements** InitializingBean {
223. //每页显示的条目数量
224. **private** Integer pageSize = 15;
226. **private** Compass compass;
228. **private** CompassTemplate compassTemplate;
230. /\*\*
231. \* 公开的搜索接口，返回匹配的搜索结果，与
232. \* {@link org.compass.spring.web.mvc.CompassSearchController#handle(javax.servlet.http.HttpServletRequest,
233. \*javax.servlet.http.HttpServletResponse,Object,org.springframework.validation.BindException) 处理相似
234. \*
235. \* @see org.compass.spring.web.mvc.CompassSearchController#handle(javax.servlet.http.HttpServletRequest,
236. \*javax.servlet.http.HttpServletResponse,java.lang.Object,org.springframework.validation.BindException)
237. \*/
238. **public** CompassSearchResults search(**final** CompassSearchCommand command) {
239. **return** (CompassSearchResults) getCompassTemplate().execute(
240. CompassTransaction.TransactionIsolation.READ\_ONLY\_READ\_COMMITTED, **new** CompassCallback() {
241. **public** Object doInCompass(CompassSession session) {
242. **return** performSearch(command, session);
243. }
244. });
245. }
247. /\*\*
248. \* 通过此方法调用搜索引擎,进行结果匹配搜索.
249. \*
250. \* @see org.compass.spring.web.mvc.CompassSearchController#performSearch(
251. \*org.compass.spring.web.mvc.CompassSearchCommand,org.compass.core.CompassSession)
252. \*/
253. **protected** CompassSearchResults performSearch(CompassSearchCommand searchCommand, CompassSession session) {
254. **long** time = System.currentTimeMillis();
255. CompassQuery query = buildQuery(searchCommand, session);
256. CompassHits hits = query.hits();
257. CompassDetachedHits detachedHits;
258. CompassSearchResults.Page[] pages = **null**;
259. **if** (pageSize == **null**) {
260. doProcessBeforeDetach(searchCommand, session, hits, -1, -1);
261. detachedHits = hits.detach();
262. } **else** {
263. **int** iPageSize = pageSize;
264. **int** page = 0;
265. **int** hitsLength = hits.getLength();
266. **if** (searchCommand.getPage() != **null**) {
267. page = searchCommand.getPage();
268. }
269. **int** from = page \* iPageSize;
271. **if** (from > hits.getLength()) {
273. // 如果起始的条目大于搜索到的条目
274. from = hits.getLength() - iPageSize;
275. doProcessBeforeDetach(searchCommand, session, hits, from, hitsLength);
276. detachedHits = hits.detach(from, hitsLength);
277. } **else** **if** ((from + iPageSize) > hitsLength) {
279. // 结束的条目大于搜索到的结果
280. doProcessBeforeDetach(searchCommand, session, hits, from, hitsLength);
281. detachedHits = hits.detach(from, hitsLength);
282. } **else** {
284. // 中间的页码，直接取出相应的条目
285. doProcessBeforeDetach(searchCommand, session, hits, from, iPageSize);
286. detachedHits = hits.detach(from, iPageSize);
287. }
288. doProcessAfterDetach(searchCommand, session, detachedHits);
289. **int** numberOfPages = (**int**) Math.ceil((**float**) hitsLength / iPageSize);
290. pages = **new** CompassSearchResults.Page[numberOfPages];
291. **for** (**int** i = 0; i < pages.length; i++) {
292. pages[i] = **new** CompassSearchResults.Page();
293. pages[i].setFrom(i \* iPageSize + 1);
294. pages[i].setSize(iPageSize);
295. pages[i].setTo((i + 1) \* iPageSize);
296. **if** (from >= (pages[i].getFrom() - 1) && from < pages[i].getTo()) {
297. pages[i].setSelected(**true**);
298. } **else** {
299. pages[i].setSelected(**false**);
300. }
301. }
302. **if** (numberOfPages > 0) {
303. CompassSearchResults.Page lastPage = pages[numberOfPages - 1];
304. **if** (lastPage.getTo() > hitsLength) {
305. lastPage.setSize(hitsLength - lastPage.getFrom());
306. lastPage.setTo(hitsLength);
307. }
308. }
309. }
310. time = System.currentTimeMillis() - time;
311. CompassSearchResults searchResults = **new** CompassSearchResults(detachedHits.getHits(), time, pageSize);
312. searchResults.setPages(pages);
313. **return** searchResults;
314. }
316. /\*\*
317. \* 构建Lucene搜索器.
318. \*/
319. **protected** CompassQuery buildQuery(CompassSearchCommand searchCommand, CompassSession session) {
320. CompassQuery query = session.queryBuilder().queryString(searchCommand.getQuery().trim()).toQuery();
322. **if** (AdvancedSearchCommand.**class**.isAssignableFrom(searchCommand.getClass())) {
323. AdvancedSearchCommand advancedSearchCommand = (AdvancedSearchCommand) searchCommand;
325. **for** (CompassSort sort : advancedSearchCommand.getSortMap()) {
326. query.addSort(sort.getName(), sort.getType(), sort.getDirection());
327. }
328. }
329. **return** query;
330. }
332. /\*\*
333. \* 在detach 之前，可以做一些操作。比如highlighting...
334. \*
335. \* @param from 需要注意的是，如果pageSize 没有指定，那么这里传入的参数为-1
336. \*/
337. **protected** **void** doProcessBeforeDetach(CompassSearchCommand searchCommand, CompassSession session, CompassHits hits,
338. **int** from, **int** size) {
339. **if** (AdvancedSearchCommand.**class**.isAssignableFrom(searchCommand.getClass())) {
340. **if** (from < 0) {
341. from = 0;
342. size = hits.getLength();
343. }
344. String[] highlightFields = ((AdvancedSearchCommand) searchCommand).getHighlightFields();
346. **if** (highlightFields == **null**) {
347. **return**;
348. }
350. // highlight fields
351. **for** (**int** i = from; i < size; i++) {
352. **for** (String highlightField : highlightFields) {
353. hits.highlighter(i).fragment(highlightField);
354. }
355. }
356. }
357. }
359. /\*\*
360. \* An option to perform any type of processing before the hits are detached.
361. \*/
362. **protected** **void** doProcessAfterDetach(CompassSearchCommand searchCommand, CompassSession session,
363. CompassDetachedHits hits) {
365. }
367. **public** **void** afterPropertiesSet() **throws** Exception {
368. Assert.notNull(compass, "Must set compass property");
369. **this**.compassTemplate = **new** CompassTemplate(compass);
370. }
372. **public** Integer getPageSize() {
373. **return** pageSize;
374. }
376. **public** **void** setPageSize(Integer pageSize) {
377. **this**.pageSize = pageSize;
378. }
380. **public** **void** setCompass(Compass compass) {
381. **this**.compass = compass;
382. }
384. **protected** CompassTemplate getCompassTemplate() {
385. **return** **this**.compassTemplate;
386. }

AdvancedSearchCommand.java

package com.mobilesoft.framework.search.service;

import java.util.HashSet;

import java.util.Set;

import org.apache.commons.lang.StringUtils;

import org.compass.core.CompassQuery.SortDirection;

import org.compass.core.CompassQuery.SortPropertyType;

import org.compass.core.support.search.CompassSearchCommand;

import org.springframework.util.Assert;

public class AdvancedSearchCommand extends CompassSearchCommand {

/\*\*

\* 封装基于Compass 的排序参数.

\*/

class CompassSort {

private String name;

private SortPropertyType type;

private SortDirection direction;

public CompassSort() {

}

public CompassSort(String sortParamName, String paramType,

boolean isAscend) {

Assert.isTrue(StringUtils.isNotBlank(sortParamName));

setName(sortParamName);

if ("int".equalsIgnoreCase(paramType)) {

setType(SortPropertyType.INT);

} else if ("float".equalsIgnoreCase(paramType)) {

setType(SortPropertyType.FLOAT);

} else if ("string".equalsIgnoreCase(paramType)) {

setType(SortPropertyType.STRING);

} else {

setType(SortPropertyType.AUTO);

}

if (isAscend) {

setDirection(SortDirection.AUTO);

} else {

setDirection(SortDirection.REVERSE);

}

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public SortPropertyType getType() {

return type;

}

public void setType(SortPropertyType type) {

this.type = type;

}

public SortDirection getDirection() {

return direction;

}

public void setDirection(SortDirection direction) {

this.direction = direction;

}

}

/\*\*

\* 搜索结果排序表.

\*/

private Set<CompassSort> sortMap = new HashSet<CompassSort>();

private String[] highlightFields;

/\*\*

\* @param paramType 现定义了三种类型： int string 以及 float。<br>

\* 除去这三种外，其他会被自动定义为SortPropertyType.AUTO 具体的可见{@link org.compass.core.CompassQuery.SortPropertyType}

\* @param isAscend 顺序还是倒序排序

\* @see org.compass.core.CompassQuery.SortPropertyType#AUTO

\* @see org.compass.core.CompassQuery.SortPropertyType#INT

\* @see org.compass.core.CompassQuery.SortPropertyType#STRING

\* @see org.compass.core.CompassQuery.SortPropertyType#FLOAT

\* @see org.compass.core.CompassQuery.SortDirection#AUTO

\* @see org.compass.core.CompassQuery.SortDirection#REVERSE

\*/

public void addSort(String sortParamName, String paramType, boolean isAscend) {

this.sortMap.add(new CompassSort(sortParamName, paramType, isAscend));

}

public Set<CompassSort> getSortMap() {

return sortMap;

}

public void setSortMap(Set<CompassSort> sortMap) {

this.sortMap = sortMap;

}

public String[] getHighlightFields() {

return highlightFields;

}

public void setHighlightFields(String[] highlightFields) {

this.highlightFields = highlightFields;

}

}

CompassIndexBuilder.java

package com.mobilesoft.framework.search.service;

import org.apache.log4j.Logger;

import org.compass.gps.CompassGps;

import org.springframework.beans.factory.InitializingBean;

import org.springframework.util.Assert;

/\*\*

\* 通过quartz定时调度定时重建索引或自动随Spring ApplicationContext启动而重建索引的Builder.

\* 会启动后延时数秒新开线程调用compassGps.index()函数.

\* 默认会在Web应用每次启动时重建索引,可以设置buildIndex属性为false来禁止此功能.

\* 也可以不用本Builder, 编写手动调用compassGps.index()的代码.

\*

\*/

public class CompassIndexBuilder implements InitializingBean {

private static final Logger log = Logger.getLogger(CompassIndexBuilder.class);

// 是否需要建立索引，可被设置为false使本Builder失效.

private boolean buildIndex = false;

// 索引操作线程延时启动的时间，单位为秒

private int lazyTime = 10;

// Compass封装

private CompassGps compassGps;

// 索引线程

private Thread indexThread = new Thread() {

@Override

public void run() {

try {

Thread.sleep(lazyTime \* 1000);

log.info("begin compass index...");

long beginTime = System.currentTimeMillis();

// 重建索引.

// 如果compass实体中定义的索引文件已存在，索引过程中会建立临时索引，

// 索引完成后再进行覆盖.

compassGps.index();

long costTime = System.currentTimeMillis() - beginTime;

log.info("compss index finished.");

log.info("costed " + costTime + " milliseconds");

} catch (InterruptedException e) {

// simply proceed

}

}

};

/\*\*

\* 实现<code>InitializingBean</code>接口，在完成注入后调用启动索引线程.

\*

\* @see org.springframework.beans.factory.InitializingBean#afterPropertiesSet()

\*/

public void afterPropertiesSet() throws Exception {

if (buildIndex) {

Assert.notNull(compassGps, "CompassIndexBuilder not set CompassGps yet.");

indexThread.setDaemon(true);

indexThread.setName("Compass Indexer");

indexThread.start();

}

}

public void setBuildIndex(boolean buildIndex) {

this.buildIndex = buildIndex;

}

public void setLazyTime(int lazyTime) {

this.lazyTime = lazyTime;

}

public void setCompassGps(CompassGps compassGps) {

this.compassGps = compassGps;

}

}

CompassSearchService.java

package com.mobilesoft.framework.search.service;

import org.compass.core.Compass;

import org.compass.core.CompassCallback;

import org.compass.core.CompassDetachedHits;

import org.compass.core.CompassHits;

import org.compass.core.CompassQuery;

import org.compass.core.CompassSession;

import org.compass.core.CompassTemplate;

import org.compass.core.CompassTransaction;

import org.compass.core.support.search.CompassSearchCommand;

import org.compass.core.support.search.CompassSearchResults;

import org.springframework.beans.factory.InitializingBean;

import org.springframework.util.Assert;

import com.mobilesoft.framework.search.service.AdvancedSearchCommand.CompassSort;

/\*\*

\* 仿照 {@link org.compass.spring.web.mvc.CompassSearchController}

\* 中的代码，构建了一个Service，方便不使用Spring MVC

\*

\* @see org.compass.spring.web.mvc.CompassSearchController

\* @see org.compass.spring.web.mvc.AbstractCompassCommandController

\*/

public class CompassSearchService implements InitializingBean {

//每页显示的条目数量

private Integer pageSize = 15;

private Compass compass;

private CompassTemplate compassTemplate;

/\*\*

\* 公开的搜索接口，返回匹配的搜索结果，与

\* {@link org.compass.spring.web.mvc.CompassSearchController#handle(javax.servlet.http.HttpServletRequest,

\*javax.servlet.http.HttpServletResponse,Object,org.springframework.validation.BindException) 处理相似

\*

\* @see org.compass.spring.web.mvc.CompassSearchController#handle(javax.servlet.http.HttpServletRequest,

\*javax.servlet.http.HttpServletResponse,java.lang.Object,org.springframework.validation.BindException)

\*/

public CompassSearchResults search(final CompassSearchCommand command) {

return (CompassSearchResults) getCompassTemplate().execute(

CompassTransaction.TransactionIsolation.READ\_ONLY\_READ\_COMMITTED, new CompassCallback() {

public Object doInCompass(CompassSession session) {

return performSearch(command, session);

}

});

}

/\*\*

\* 通过此方法调用搜索引擎,进行结果匹配搜索.

\*

\* @see org.compass.spring.web.mvc.CompassSearchController#performSearch(

\*org.compass.spring.web.mvc.CompassSearchCommand,org.compass.core.CompassSession)

\*/

protected CompassSearchResults performSearch(CompassSearchCommand searchCommand, CompassSession session) {

long time = System.currentTimeMillis();

CompassQuery query = buildQuery(searchCommand, session);

CompassHits hits = query.hits();

CompassDetachedHits detachedHits;

CompassSearchResults.Page[] pages = null;

if (pageSize == null) {

doProcessBeforeDetach(searchCommand, session, hits, -1, -1);

detachedHits = hits.detach();

} else {

int iPageSize = pageSize;

int page = 0;

int hitsLength = hits.getLength();

if (searchCommand.getPage() != null) {

page = searchCommand.getPage();

}

int from = page \* iPageSize;

if (from > hits.getLength()) {

// 如果起始的条目大于搜索到的条目

from = hits.getLength() - iPageSize;

doProcessBeforeDetach(searchCommand, session, hits, from, hitsLength);

detachedHits = hits.detach(from, hitsLength);

} else if ((from + iPageSize) > hitsLength) {

// 结束的条目大于搜索到的结果

doProcessBeforeDetach(searchCommand, session, hits, from, hitsLength);

detachedHits = hits.detach(from, hitsLength);

} else {

// 中间的页码，直接取出相应的条目

doProcessBeforeDetach(searchCommand, session, hits, from, iPageSize);

detachedHits = hits.detach(from, iPageSize);

}

doProcessAfterDetach(searchCommand, session, detachedHits);

int numberOfPages = (int) Math.ceil((float) hitsLength / iPageSize);

pages = new CompassSearchResults.Page[numberOfPages];

for (int i = 0; i < pages.length; i++) {

pages[i] = new CompassSearchResults.Page();

pages[i].setFrom(i \* iPageSize + 1);

pages[i].setSize(iPageSize);

pages[i].setTo((i + 1) \* iPageSize);

if (from >= (pages[i].getFrom() - 1) && from < pages[i].getTo()) {

pages[i].setSelected(true);

} else {

pages[i].setSelected(false);

}

}

if (numberOfPages > 0) {

CompassSearchResults.Page lastPage = pages[numberOfPages - 1];

if (lastPage.getTo() > hitsLength) {

lastPage.setSize(hitsLength - lastPage.getFrom());

lastPage.setTo(hitsLength);

}

}

}

time = System.currentTimeMillis() - time;

CompassSearchResults searchResults = new CompassSearchResults(detachedHits.getHits(), time, pageSize);

searchResults.setPages(pages);

return searchResults;

}

/\*\*

\* 构建Lucene搜索器.

\*/

protected CompassQuery buildQuery(CompassSearchCommand searchCommand, CompassSession session) {

CompassQuery query = session.queryBuilder().queryString(searchCommand.getQuery().trim()).toQuery();

if (AdvancedSearchCommand.class.isAssignableFrom(searchCommand.getClass())) {

AdvancedSearchCommand advancedSearchCommand = (AdvancedSearchCommand) searchCommand;

for (CompassSort sort : advancedSearchCommand.getSortMap()) {

query.addSort(sort.getName(), sort.getType(), sort.getDirection());

}

}

return query;

}

/\*\*

\* 在detach 之前，可以做一些操作。比如highlighting...

\*

\* @param from 需要注意的是，如果pageSize 没有指定，那么这里传入的参数为-1

\*/

protected void doProcessBeforeDetach(CompassSearchCommand searchCommand, CompassSession session, CompassHits hits,

int from, int size) {

if (AdvancedSearchCommand.class.isAssignableFrom(searchCommand.getClass())) {

if (from < 0) {

from = 0;

size = hits.getLength();

}

String[] highlightFields = ((AdvancedSearchCommand) searchCommand).getHighlightFields();

if (highlightFields == null) {

return;

}

// highlight fields

for (int i = from; i < size; i++) {

for (String highlightField : highlightFields) {

hits.highlighter(i).fragment(highlightField);

}

}

}

}

/\*\*

\* An option to perform any type of processing before the hits are detached.

\*/

protected void doProcessAfterDetach(CompassSearchCommand searchCommand, CompassSession session,

CompassDetachedHits hits) {

}

public void afterPropertiesSet() throws Exception {

Assert.notNull(compass, "Must set compass property");

this.compassTemplate = new CompassTemplate(compass);

}

public Integer getPageSize() {

return pageSize;

}

public void setPageSize(Integer pageSize) {

this.pageSize = pageSize;

}

public void setCompass(Compass compass) {

this.compass = compass;

}

protected CompassTemplate getCompassTemplate() {

return this.compassTemplate;

}

8、Model层   
  
@SearchableId 声明Document的id列；   
  
@SearchableProperty 声明要索引的field；   
  
@SearchableComponent 声明要索引的其他关联对象。

Java代码

1. Article.java
3. **package** com.mobilesoft.esales.model;
5. **import** java.util.Date;
7. **import** org.compass.annotations.Searchable;
8. **import** org.compass.annotations.SearchableId;
9. **import** org.compass.annotations.SearchableProperty;
10. **import** org.compass.core.CompassTemplate;

13. @Searchable
14. **public** **class** Article  **implements** java.io.Serializable {
16. @SearchableId
17. **private** Integer id;
18. @SearchableProperty(name="title")
19. **private** String title;
20. @SearchableProperty(name="author")
21. **private** Integer author;
22. @SearchableProperty(name="publishDate")
23. **private** Date publishDate;
25. /\*\* default constructor \*/
26. **public** Article() {
27. }
29. /\*\* minimal constructor \*/
30. **public** Article(String title, Date publishDate) {
31. **this**.title = title;
32. **this**.publishDate = publishDate;
33. }
35. /\*\* full constructor \*/
36. **public** Article(String title, Integer author, Date publishDate) {
37. **this**.title = title;
38. **this**.author = author;
39. **this**.publishDate = publishDate;
40. }
42. **public** Integer getId() {
43. **return** **this**.id;
44. }
46. **public** **void** setId(Integer id) {
47. **this**.id = id;
48. }
50. **public** String getTitle() {
51. **return** **this**.title;
52. }
54. **public** **void** setTitle(String title) {
55. **this**.title = title;
56. }
58. **public** Integer getAuthor() {
59. **return** **this**.author;
60. }
62. **public** **void** setAuthor(Integer author) {
63. **this**.author = author;
64. }
66. **public** Date getPublishDate() {
67. **return** **this**.publishDate;
68. }
70. **public** **void** setPublishDate(Date publishDate) {
71. **this**.publishDate = publishDate;
72. }
74. }
76. Author.java
78. **package** com.mobilesoft.esales.model;
80. **import** org.compass.annotations.Searchable;
81. **import** org.compass.annotations.SearchableId;
82. **import** org.compass.annotations.SearchableProperty;
83. **import** org.compass.core.CompassTemplate;

86. @Searchable
87. **public** **class** Author  **implements** java.io.Serializable {

90. @SearchableId
91. **private** Integer id;
92. @SearchableProperty(name="username")
93. **private** String username;
94. **private** String password;
95. @SearchableProperty(name="age")
96. **private** Short age;
98. **public** Author() {
99. }
101. /\*\* minimal constructor \*/
102. **public** Author(String username, String password) {
103. **this**.username = username;
104. **this**.password = password;
105. }
107. /\*\* full constructor \*/
108. **public** Author(String username, String password, Short age) {
109. **this**.username = username;
110. **this**.password = password;
111. **this**.age = age;
112. }
114. // Property accessors
116. **public** Integer getId() {
117. **return** **this**.id;
118. }
120. **public** **void** setId(Integer id) {
121. **this**.id = id;
122. }
124. **public** String getUsername() {
125. **return** **this**.username;
126. }
128. **public** **void** setUsername(String username) {
129. **this**.username = username;
130. }
132. **public** String getPassword() {
133. **return** **this**.password;
134. }
136. **public** **void** setPassword(String password) {
137. **this**.password = password;
138. }
140. **public** Short getAge() {
141. **return** **this**.age;
142. }
144. **public** **void** setAge(Short age) {
145. **this**.age = age;
146. }

149. }

Article.java

package com.mobilesoft.esales.model;

import java.util.Date;

import org.compass.annotations.Searchable;

import org.compass.annotations.SearchableId;

import org.compass.annotations.SearchableProperty;

import org.compass.core.CompassTemplate;

@Searchable

public class Article implements java.io.Serializable {

@SearchableId

private Integer id;

@SearchableProperty(name="title")

private String title;

@SearchableProperty(name="author")

private Integer author;

@SearchableProperty(name="publishDate")

private Date publishDate;

/\*\* default constructor \*/

public Article() {

}

/\*\* minimal constructor \*/

public Article(String title, Date publishDate) {

this.title = title;

this.publishDate = publishDate;

}

/\*\* full constructor \*/

public Article(String title, Integer author, Date publishDate) {

this.title = title;

this.author = author;

this.publishDate = publishDate;

}

public Integer getId() {

return this.id;

}

public void setId(Integer id) {

this.id = id;

}

public String getTitle() {

return this.title;

}

public void setTitle(String title) {

this.title = title;

}

public Integer getAuthor() {

return this.author;

}

public void setAuthor(Integer author) {

this.author = author;

}

public Date getPublishDate() {

return this.publishDate;

}

public void setPublishDate(Date publishDate) {

this.publishDate = publishDate;

}

}

Author.java

package com.mobilesoft.esales.model;

import org.compass.annotations.Searchable;

import org.compass.annotations.SearchableId;

import org.compass.annotations.SearchableProperty;

import org.compass.core.CompassTemplate;

@Searchable

public class Author implements java.io.Serializable {

@SearchableId

private Integer id;

@SearchableProperty(name="username")

private String username;

private String password;

@SearchableProperty(name="age")

private Short age;

public Author() {

}

/\*\* minimal constructor \*/

public Author(String username, String password) {

this.username = username;

this.password = password;

}

/\*\* full constructor \*/

public Author(String username, String password, Short age) {

this.username = username;

this.password = password;

this.age = age;

}

// Property accessors

public Integer getId() {

return this.id;

}

public void setId(Integer id) {

this.id = id;

}

public String getUsername() {

return this.username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return this.password;

}

public void setPassword(String password) {

this.password = password;

}

public Short getAge() {

return this.age;

}

public void setAge(Short age) {

this.age = age;

}

}

9、DAO层   
  
ArticleDAO.java和AuthorDAO.java省略   
  
直接用MyEclipse生成的，没有什么特别的。   
10、参考文档   
  
http://www.compass-project.org/docs/1.2.1/reference/html/   
  
The Compass Framework Search made easy.pdf   
  
Compass TSSJS Europe 06.pdf   
  
Hello World Tutorial   
  
InfoQ:  Compass: Integrate Search into your apps   
  
InfoQ: Compass: Simplifying and Extending Lucene to Provide Google-like Search   
  
InfoQ: Compass: 在你的应用中集成搜索功能   
  
Compass 指南   
  
http://www.kimchy.org/