企业级搜索引擎Solr使用入门指南

    由于搜索引擎功能在门户社区中对提高用户体验有着重在门户社区中涉及大量需要搜索引擎的功能需求，目前在实现搜索引擎的方案上有集中方案可供选择：

* 基于Lucene自己进行封装实现站内搜索。

工作量及扩展性都较大，不采用。

* 调用Google、Baidu的API实现站内搜索

同第三方搜索引擎绑定太死，无法满足后期业务扩展需要，暂时不采用。

* 基于Compass+Lucene实现站内搜索

    适合于对数据库驱动的应用数据进行索引，尤其是替代传统的like ‘%expression%’来实现对varchar或clob等字段的索引，对于实现站内搜索是一种值得采纳的方案。但在分布式处理、接口封装上尚需要自己进行一定程度的封装，暂时不采用。

* 基于Solr实现站内搜索

封装及扩展性较好，提供了较为完备的解决方案，因此在门户社区中采用此方案，后期加入Compass方案。

**1、 Solr简介**

    Solr是一个基于Lucene的Java搜索引擎服务器。Solr 提供了层面搜索、命中醒目显示并且支持多种输出格式（包括 XML/XSLT 和 JSON 格式）。它易于安装和配置，而且附带了一个基于 HTTP 的管理界面。Solr已经在众多大型的网站中使用，较为成熟和稳定。Solr 包装并扩展了 Lucene，所以Solr的基本上沿用了Lucene的相关术语。更重要的是，Solr 创建的索引与 Lucene 搜索引擎库完全兼容。通过对 Solr 进行适当的配置，某些情况下可能需要进行编码，Solr 可以阅读和使用构建到其他 Lucene 应用程序中的索引。此外，很多 Lucene 工具（如Nutch、 Luke）也可以使用 Solr 创建的索引。

**2、 Tomcat下Solr安装配置**

    由于Solr基于java开发，因此Solr在windows及Linux都能较好部署使用，但由于Solr提供了一些用于测试及管理、维护较为方便的shell脚本，因此在生产部署时候建议安装在Linux上，测试时候可以在windows使用。

下面以Linux下安装配置Solr进行说明，windows与此类似。

wget <http://apache.mirror.phpchina.com/tomcat/tomcat-6/v6.0.16/bin/apache-tomcat-6.0.16.zip>

unzip apache-tomcat-6.0.16.zip

mv apache-tomcat-6.0.16 /opt/tomcat

chmod 755 /opt/tomcat/bin/\*

wget <http://apache.mirror.phpchina.com/lucene/solr/1.2/apache-solr-1.2.0.tgz>

tar zxvf apache-solr-1.2.0.tgz

Solr的安装配置最为麻烦的是对solr.solr.home的理解和配置，主要有三种

* 基于当前路径的方式

cp apache-solr-1.2.0/dist/apache-solr-1.2.0.war /opt/tomcat/webapps/solr.war

mkdir /opt/solr-tomcat

cp -r apache-solr-1.2.0/example/solr/ /opt/solr-tomcat/

cd /opt/solr-tomcat

/opt/tomcat/bin/startup.sh

由于在此种情况下（没有设定solr.solr.home环境变量或JNDI的情况下），Solr查找./solr，因此在启动时候需要切换到/opt/solr-tomcat

* 基于环境变量solr.solr.home

在当前用户的环境变量中（.bash\_profile）或在/opt/tomcat/catalina.sh中添加如下环境变量

export JAVA\_OPTS="$JAVA\_OPTS -Dsolr.solr.home=/opt/solr-tomcat/solr"

* 基于JNDI配置

mkdir –p /opt/tomcat/conf/Catalina/localhost

touch /opt/tomcat/conf/Catalina/localhost/solr.xml ，内容如下:

*<Context docBase="/opt/tomcat/webapps/solr.war" debug="0" crossContext="true" >*

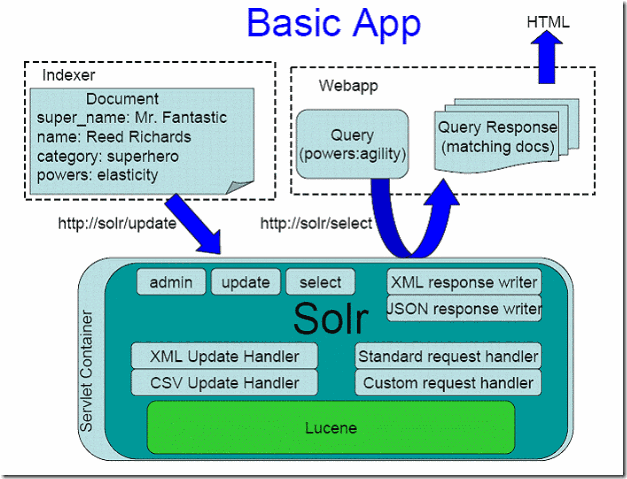
*<Environment name="solr/home" type="java.lang.String" value="/opt/solr-tomcat/solr" override="true" />*

*</Context>*

访问solr管理界面

<http://ip:port/solr>

**3、 Solr原理**

[](http://e3gtca.bay.livefilestore.com/y1pEb4KCtdIpP898kNj3YMuu3quK1GCFbWMJig0sjgeuCTnPUM2fSawH7oFKlVZyx4AyQf827TFyAE?PARTNER=WRITER)

    Solr对外提供标准的http接口来实现对数据的索引的增加、删除、修改、查询。在 Solr 中，用户通过向部署在servlet 容器中的 Solr Web 应用程序发送 HTTP 请求来启动索引和搜索。Solr 接受请求，确定要使用的适当SolrRequestHandler，然后处理请求。通过 HTTP 以同样的方式返回响应。默认配置返回 Solr 的标准 XML 响应，也可以配置 Solr 的备用响应格式。

可以向 Solr 索引 servlet 传递四个不同的索引请求：

* add/update 允许向 Solr 添加文档或更新文档。直到提交后才能搜索到这些添加和更新。
* commit 告诉 Solr，应该使上次提交以来所做的所有更改都可以搜索到。
* optimize 重构 Lucene 的文件以改进搜索性能。索引完成后执行一下优化通常比较好。如果更新比较频繁，则应该在使用率较低的时候安排优化。一个索引无需优化也可以正常地运行。优化是一个耗时较多的过程。
* delete 可以通过 id 或查询来指定。按 id 删除将删除具有指定 id 的文档；按查询删除将删除查询返回的所有文档。

一个典型的Add请求报文

<add>

<doc>

  <field name="id">TWINX2048-3200PRO</field>

  <field name="name">CORSAIR  XMS 2GB (2 x 1GB) 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) Dual Channel Kit System Memory - Retail</field>

  <field name="manu">Corsair Microsystems Inc.</field>

  <field name="cat">electronics</field>

  <field name="cat">memory</field>

  <field name="features">CAS latency 2, 2-3-3-6 timing, 2.75v, unbuffered, heat-spreader</field>

  <field name="price">185</field>

  <field name="popularity">5</field>

  <field name="inStock">true</field>

</doc>

<doc>

  <field name="id">VS1GB400C3</field>

  <field name="name">CORSAIR ValueSelect 1GB 184-Pin DDR SDRAM Unbuffered DDR 400 (PC 3200) System Memory - Retail</field>

  <field name="manu">Corsair Microsystems Inc.</field>

  <field name="cat">electronics</field>

  <field name="cat">memory</field>

  <field name="price">74.99</field>

  <field name="popularity">7</field>

  <field name="inStock">true</field>

</doc>

</add>

一个典型的搜索结果报文：

<response>

    <lst name="responseHeader">

        <int name="status">0</int>

        <int name="QTime">6</int>

        <lst name="params">

            <str name="rows">10</str>

            <str name="start">0</str>

            <str name="fl">\*,score</str>

            <str name="hl">true</str>

            <str name="q">content:"faceted browsing"</str>

        </lst>

    </lst>

    <result name="response" numFound="1" start="0" maxScore="1.058217">

        <doc>

            <float name="score">1.058217</float>

            <arr name="all">

                <str>[http://localhost/myBlog/solr-rocks-again.html</str](http://localhost/myBlog/solr-rocks-again.html%3c/str)>

                <str>Solr is Great</str>

                <str>solr,lucene,enterprise,search,greatness</str>

                <str>Solr has some really great features, like faceted browsing

                and replication</str>

            </arr>

            <arr name="content">

                <str>Solr has some really great features, like faceted browsing

                and replication</str>

            </arr>

            <date name="creationDate">2007-01-07T05:04:00.000Z</date>

            <arr name="keywords">

                <str>solr,lucene,enterprise,search,greatness</str>

            </arr>

            <int name="rating">8</int>

            <str name="title">Solr is Great</str>

            <str name="url">[http://localhost/myBlog/solr-rocks-again.html</str](http://localhost/myBlog/solr-rocks-again.html%3c/str)>

        </doc>

    </result>

    <lst name="highlighting">

        <lst name="<http://localhost/myBlog/solr-rocks-again.html>">

            <arr name="content">

                <str>Solr has some really great features, like <em>faceted</em>

                <em>browsing</em> and replication</str>

            </arr>

        </lst>

    </lst>

</response>

关于solr的详细使用说明，请参考

<http://wiki.apache.org/solr/FrontPage>

**4、 Solr测试使用**

Solr的安装包包含了相关的测试样例，路径在apache-solr-1.2.0/example/exampledocs

* 使用shell脚本（curl）测试Solr的操作：

cd apache-solr-1.2.0/example/exampledocs

vi post.sh，根据tomcat的ip、port修改URL变量的值URL=http://localhost:8080/solr/update

./post.sh \*.xml                 #

* 使用Solr的java 包测试Solr的操作：

查看帮助：java -jar post.jar –help

提交测试数据：

java -Durl=http://localhost:8080/solr/update -Ddata=files -jar post.jar  \*.xml

下面以增加索引字段liangchuan、url为例，说明一下Solr中索引命令的使用

1) 修改solr的schema，配置需要索引字段的说明：

vi /opt/solr-tomcat/solr/conf/schema.xml ,在<fields>中增加如下内容

   <field name="liangchuan"  type="string" indexed="true" stored="true"/>

   <field name="url"  type="string" indexed="true" stored="true"/>

2) 创建增加索引请求的xml测试文件

touch /root/apache-solr-1.2.0/example/exampledocs/liangchuan.xml,内容如下：

<add>

<doc>

  <field name="id">liangchuan000</field>

  <field name="name">Solr, the Enterprise Search Server</field>

  <field name="manu">Apache Software Foundation</field>

  <field name="liangchuan">liangchuan's solr "hello,world" test</field>

  <field name="url">http://www.google.com</field>

</doc>

</add>

3) 提交索引请求

cd apache-solr-1.2.0/example/exampledocs

    ./post.sh liangchuan.xml

4) 查询

通过solr的管理员界面<http://localhost:8080/solr/admin>查询

或通过curl 测试：

       export URL="<http://localhost:8080/solr/select/>"

       curl "$URL?indent=on&q=liangchuan&fl=\*,score"

5、Solr查询条件参数说明

|  |  |  |
| --- | --- | --- |
| **参数** | **描述** | **示例** |
| q | Solr 中用来搜索的查询。可以通过追加一个分号和已索引且未进行断词的字段的名称来包含排序信息。默认的排序是 score desc，指按记分降序排序。 | q=myField:Java AND otherField:developerWorks; date asc  此查询搜索指定的两个字段并根据一个日期字段对结果进行排序。 |
| start | 将初始偏移量指定到结果集中。可用于对结果进行分页。默认值为 0。 | start=15  返回从第 15 个结果开始的结果。 |
| rows | 返回文档的最大数目。默认值为 10。 | rows=25 |
| fq | 提供一个可选的筛选器查询。查询结果被限制为仅搜索筛选器查询返回的结果。筛选过的查询由 Solr 进行缓存。它们对提高复杂查询的速度非常有用。 | 任何可以用 q 参数传递的有效查询，排序信息除外。 |
| hl | 当 hl=true 时，在查询响应中醒目显示片段。默认为 false。参看醒目显示参数上的 Solr Wiki 部分可以查看更多选项 | hl=true |
| fl | 作为逗号分隔的列表指定文档结果中应返回的 Field 集。默认为 “\*”，指所有的字段。“score” 指还应返回记分。 | \*,score |

其中关于Solr查询相关的参数详细的信息请参看：

<http://wiki.apache.org/solr/CommonQueryParameters>

    Solr的查询条件参数q的格式与Lucene相同，具体参看：

<http://lucene.apache.org/java/docs/queryparsersyntax.html>

**6、 在门户社区中solr使用模式**

在门户社区中需要使用solr，可采用如下模式：

* 对原有系统已有的数据或需要索引的数据量较大的情况

    直接采用通过http方式调用solr的接口方式，效率较差，采用solr本身对csv 的支持（<http://wiki.apache.org/solr/UpdateCSV>

），将数据导出为csv格式，然后调用solr的csv接口<http://localhost:8080/solr/update/csv>

* 对系统新增的数据

    先将需要索引查询的数据组装成xml格式，然后使用httpclient 将数据提交到solr 的http接口，例如

<http://localhost:8080/solr/update>

   也可以参考post.jar中的SimplePostTool的实现。

<http://svn.apache.org/viewvc/lucene/solr/trunk/src/java/org/apache/solr/util/SimplePostTool.java?view=co>

* 中文分词

    采用庖丁解牛作为solr（Lucene）缺省的中文分词方案

    项目库：<http://code.google.com/p/paoding/>

    Google groups：<http://groups.google.com/group/paoding>

    Javaeye的groups：<http://analysis.group.javaeye.com/>

* 与nutch的集成使用

<http://blog.foofactory.fi/2007/02/online-indexing-integrating-nutch-with.html>

* 嵌入式Solr

<http://wiki.apache.org/solr/Solrj#EmbeddedSolrServer>

* 分布式索引

<http://wiki.apache.org/solr/CollectionDistribution>

7、参考资料

[http://wiki.apache.org/solr/](http://wiki.apache.org/solr/CollectionDistribution)

<http://www.ibm.com/developerworks/cn/java/j-solr1/>

<http://www.ibm.com/developerworks/cn/java/j-solr2/>

<http://www.xml.com/pub/a/2006/08/09/solr-indexing-xml-with-lucene-andrest.html?page=1>

<http://lucene.apache.org/java/docs/queryparsersyntax.html>

<http://www.blogjava.net/RongHao/archive/2007/11/06/158621.html>

在网络上找了很多的关于solr的资料，发现非常的不全面，即使就是官方的wiki，也是如此！

　　基于现阶段正在研究的solr应用，陆续的会分享我的一些经验！

　　今天要说的是： 怎么跑起来！

　　1》 首先下载好solr，我用的是 solr1.3，下载地址：

　　windows版本

　　http://labs.xiaonei.com/apache-mirror/lucene/solr/1.3.0/apache-solr-1.3.0.zip

　　linux版本

　　http://labs.xiaonei.com/apache-mirror/lucene/solr/1.3.0/apache-solr-1.3.0.tgz

　　2》准备运行容器，我用的是tomcat6.0.20.如果是玩的话，也可以不用准备专门的容易，你只需解压好solr的下载包，找到 example文件夹，然后运行 start.jar。具体指令：java -jar start.jar做应用的时候，不建议采用该方式。该方式内部包含了jetty！

　　3》有关tomcat的使用，不是本文的重点，如果你有问题，请参见tomcat的使用。解压好tomcat，将solr包下面的dist文件夹中的apache-solr-1.3.0.war 拷贝到 tomcat的webapps，并且改名为 solr.war。

　　4》新建一个 solr-tomcat文件夹，我是把这个文件夹建立在C盘根目录，你可以采用同样方式，也可以不这样，放哪由你决定。建立好该文件夹以后,在把它下面在建立一个solr文件夹，把solr包里面的example\solr文件夹下面的所有文件放入到 solr里面。

　　5》最后一步 配置 solr.home，可以有三种方式。

　　1）基于当前路径的方式

　　这种情况需要在c:\solr-tomcat\目录下去启动tomcat，Solr查找./solr，因此在启动时候需要切换到c:\solr-tomcat\

　　2）基于环境变量

　　windows在环境变量中建立solr.home,值为c:\solr-tomcat

　　linux在当前用户的环境变量中（.bash\_profile）或在catalina.sh中添加如下环境变量

　　export JAVA\_OPTS="$JAVA\_OPTS -Dsolr.solr.home=/opt/solr-tomcat/solr"

　　3）基于JNDI

　　在tomcat的conf文件夹建立Catalina文件夹，然后在Catalina文件夹中在建立localhost文件夹，在该文件夹下面建立solr.xml，其中内容：

　　Xml代码

<Context　docBase="c:\tomcat\webapps\solr.war"　debug="0"　crossContext="true"　>

　　　<Environment　name="solr/home"　type="java.lang.String"　value="c:/solr-tomcat/solr"　override="true"　/>

</Context>

　　问题描述：

　　个人发现的一个问题，就是如果配置好JNDI的话，然后在tomcat的bin文件夹下面启动 tomcat的话，会在tomcat的bin下面建立solr文件夹，这个文件夹中主要存放的索引文件。本来这些东西应该放入c:\solr- tomcat\solr。如果你不想出现这种情况的话，请使用基于当前路径的方式。

　　6》打开浏览器，请看看能否访问该服务。如果能够访问，恭喜您跑起来了。

怎么在solr中加入中文分词，参考了一些文章，但是还是花了不少时间才搞出的。可能是大侠们太牛了，所以很多细节东西都没有写出来！但是要指出的是很多文章都是抄来抄去的！

　　入正题：

　　在上面的一个文章中，已经使solr跑起来了，在上面的基础上，加入中文分词。我用的是paoding分词器！

　　1》请下好paoding分词器，下载地址：

　　http://code.google.com/p/paoding/downloads/list ，在这里要非常感谢paoding作者：qieqie

　　在使用paoding的时候要注意：paoding的dic位置，也就是字典的位置，有两种办法解决：

　　1） 在系统环境变量中加入PAODING\_DIC\_HOME这个变量，值为paoding压缩包下面的dic的解压位置。

　　2）paoding-analysis.jar里面有个paoding-dic-home.properties文件，指定dic也可，但是要重新编译这个jar包，我采用的后面一种办法，只要每次固定好dic位置，部署还不是很不方便，设置环境变量我比较反感

　　2》建立文件

　　Java代码

package　com.yeedoo.slor.tokenizer;

import　java.io.Reader;

import　java.util.Map;

import　net.paoding.analysis.analyzer.PaodingTokenizer;

import　net.paoding.analysis.analyzer.TokenCollector;

import　net.paoding.analysis.analyzer.impl.MaxWordLengthTokenCollector;

import　net.paoding.analysis.analyzer.impl.MostWordsTokenCollector;

import　net.paoding.analysis.knife.PaodingMaker;

import　org.apache.lucene.analysis.TokenStream;

import　org.apache.solr.analysis.BaseTokenizerFactory;

public　class　ChineseTokenizerFactory　extends　BaseTokenizerFactory　{

　/\*\*

　\*　最多切分　默认模式

　\*/

　public　static　final　String　MOST\_WORDS\_MODE　=　"most-words";

　/\*\*

　\*　按最大切分

　\*/

　public　static　final　String　MAX\_WORD\_LENGTH\_MODE　=　"max-word-length";

　private　String　mode　=　null;

　public　void　setMode(String　mode)　{

　if　(mode　==　null　||　MOST\_WORDS\_MODE.equalsIgnoreCase(mode)　||　"default".equalsIgnoreCase(mode))　{

　　this.mode　=　MOST\_WORDS\_MODE;

　}　else　if　(MAX\_WORD\_LENGTH\_MODE.equalsIgnoreCase(mode))　{

　　this.mode　=　MAX\_WORD\_LENGTH\_MODE;

　}　else　{

　　throw　new　IllegalArgumentException("不合法的分析器Mode参数设置:"　+　mode);

　}

　}

　@Override

　public　void　init(Map<String,String>　args)　{

　super.init(args);

　setMode(args.get("mode").toString());

　}

　public　TokenStream　create(Reader　input)　{

　return　new　PaodingTokenizer(input,　PaodingMaker.make(),　createTokenCollector());

　}

　private　TokenCollector　createTokenCollector()　{

　if　(MOST\_WORDS\_MODE.equals(mode))

　　return　new　MostWordsTokenCollector();

　if　(MAX\_WORD\_LENGTH\_MODE.equals(mode))

　　return　new　MaxWordLengthTokenCollector();

　throw　new　Error("never　happened");

　}

}

　　将该文件打包，如果你不愿打包，请到附件里面下吧。

　　请将两个jar，一个是这个，还有一个是 paoding-analysis.jar 放到tomcat的webapps\solr\WEB-INF\lib\下

　　3》更改schema.xml文件，使分词器起到作用。如果你看过前面一个文章，schema.xml的具体位置在c:\solr-tomcat\solr\conf\下面。

　　更改内容为：

　　Xml代码

<fieldType　name="text"　class="solr.TextField"　positionIncrementGap="100">

　　　<analyzer　type="index">

　　　　<!--<tokenizer　class="solr.WhitespaceTokenizerFactory"/>-->

　　　　<tokenizer　class="com.yeedoo.slor.tokenizer.ChineseTokenizerFactory"　mode="most-words"/>

　　　···　···

　　　</analyzer>

　　　<analyzer　type="query">

　　　　　<!--<tokenizer　class="solr.WhitespaceTokenizerFactory"/>-->

　　　　　<tokenizer　　　class="com.yeedoo.slor.tokenizer.ChineseTokenizerFactory"　mode="most-words"/>

　　　　···　···

　　　</analyzer>

</fieldType>

　　<!-- --> 里面的为需要更改的内容

　　4》重启你的tomcat ，ok！

　　附：

　　[Solr分词顺序]

　　Solr建立索引和对关键词进行查询都得对字串进行分词，在向索引库中添加全文检索类型的索引的时候，Solr会首先用空格进行分词，然后把分词结果依次使用指定的过滤器进行过滤，最后剩下的结果才会加入到索引库中以备查询。分词的顺序如下：

　　索引

　　1：空格whitespaceTokenize

　　2：过滤词StopFilter

　　3：拆字WordDelimiterFilter

　　4：小写过滤LowerCaseFilter

　　5：英文相近词EnglishPorterFilter

　　6：去除重复词RemoveDuplicatesTokenFilter

　　查询

　　1：查询相近词

　　2：过滤词

　　3：拆字

　　4：小写过滤

　　5：英文相近词

　　6：去除重复词

　　以上是针对英文，中文的除了空格，其他都类似

　　运行solr是个很简单的事，如何让solr高效运行你的项目，这个就不容易了。要考虑的因素太多。这里很重要一个就是对solr的配置要了解。懂得配置文件每个配置项的含义，这样操作起来就会如鱼得水！

　　在solr里面主要的就是solr的主目录下面的schema.xml,solrConfig.xml，如果你看过前两篇文章的话，你应该知道solr的主目录处于什么位置（c:\solr-tomcat\solr\conf\）。

　　在这个文章中，我们首先来说说这个schema.xml。

　　schema.xml，这个相当于数据表配置文件，它定义了加入索引的数据的数据类型的。主要包括types、fields和其他的一些缺省设置。

　　1》首先需要在types结点内定义一个FieldType子结点，包括name,class,positionIncrementGap等等一些参数，name就是这个FieldType的名称，class指向org.apache.solr.analysis包里面对应的class名称，用来定义这个类型的行为。在FieldType定义的时候最重要的就是定义这个类型的数据在建立索引和进行查询的时候要使用的分析器analyzer,包括分词和过滤。在例子中text这个FieldType在定义的时候，在index的analyzer中使用 solr.WhitespaceTokenizerFactory这个分词包，就是空格分词，然后使用 solr.StopFilterFactory，solr.WordDelimiterFilterFactory，solr.LowerCaseFilterFactory，solr.EnglishPorterFilterFactory，solr.RemoveDuplicatesTokenFilterFactory 这几个过滤器。在向索引库中添加text类型的索引的时候，Solr会首先用空格进行分词，然后把分词结果依次使用指定的过滤器进行过滤，最后剩下的结果才会加入到索引库中以备查询。Solr的analysis包并没有带支持中文的包，在第二篇文章中详细讲了怎样添加paoding中文分词器，详情请参见http://lianj-lee.javaeye.com/blog/424474

　　2》接下来的工作就是在fields结点内定义具体的字段（类似数据库中的字段），就是filed，filed定义包括name,type（为之前定义过的各种FieldType）,indexed（是否被索引）,stored（是否被储存），multiValued（是否有多个值）等等。

　　例：

　　Xml代码

<fields>

　<field　name="id"　type="integer"　indexed="true"　stored="true"　required="true"　/>

　<field　name="name"　type="text"　indexed="true"　stored="true"　/>

　<field　name="summary"　type="text"　indexed="true"　stored="true"　/>

　<field　name="author"　type="string"　indexed="true"　stored="true"　/>

　<field　name="date"　type="date"　indexed="false"　stored="true"　/>

　<field　name="content"　type="text"　indexed="true"　stored="false"　/>

　<field　name="keywords"　type="keyword\_text"　indexed="true"　stored="false"　multiValued="true"　/>

　<field　name="all"　type="text"　indexed="true"　stored="false"　multiValued="true"/>

</fields>

　　field的定义相当重要，有几个技巧需注意一下，对可能存在多值得字段尽量设置multiValued属性为true，避免建索引是抛出错误；如果不需要存储相应字段值，尽量将stored属性设为false。

　　3》建议建立了一个拷贝字段，将所有的全文字段复制到一个字段中，以便进行统一的检索：

　　Xml代码

<field　name="all"　type="text"　indexed="true"　stored="false"　multiValued="true"/>

　　并在拷贝字段结点处完成拷贝设置：

　　Xml代码 <copyField　source="name"　dest="all"/>

<copyField　source="summary"　dest="all"/>

　　4》除此之外，还可以定义动态字段，所谓动态字段就是不用指定具体的名称，只要定义字段名称的规则，例如定义一个 dynamicField，name 为\*\_i，定义它的type为text，那么在使用这个字段的时候，任何以\_i结尾的字段都被认为是符合这个定义的，例如：name\_i，gender\_i，school\_i等。

　　Solr Multicore 是 solr 1.3 的新特性。其目的一个solr实例，可以有多个搜索应用。

　　下面着手来将solr给出的一个example跑出来，在《利用SOLR搭建企业搜索平台 之一（运行solr）》这篇文章里面已经讲了怎样来运行solr，这篇文章是基于《利用SOLR搭建企业搜索平台 之一（运行solr）》，有不明白的请参见http://lianj-lee.javaeye.com/blog/424383

　　1》找到solr下载包中的example文件夹，在它的下面有个multicore文件夹，将这个文件夹下面的所有东西copy到 c:\solr-tomcat\solr下面。

　　注意：有一个 solr.xml（这只是默认文件，当然也可以指定别的文件），如：

　　Xml代码

<?xml　version="1.0"　encoding="UTF-8"　?>

　<solr　persistent="false">

　　<cores　adminPath="/admin/cores">

　　　<core　name="core0"　instanceDir="core0"　/>

　　　<core　name="core1"　instanceDir="core1"　/>

　　</cores>

　</solr>

　　这个文件是告诉solr应该加载哪些core，cores里有 core0,core1。core0（可以类比以前的solr.home）/conf目录下有schema.xml与solrconfig.xml，可以把实际应用的复制过来。现示例就用官方的了。

　　2》启动tomcat，访问应用，就可以看到有 Admin core0 和 Admin core1

　　3》采用上面的默认solr.xml，索引文件将存放在同一个目录下面，在这里将存放在C:\solr-tomcat\solr\data，如果你想更改目录，或者两个应用存放在不同的目录，请参见下面的xml。

　　Xml代码

<core　name="core0"　instanceDir="core0">

　　<property　name="dataDir"　value="/data/core0"　/>

　</core>

　　给core添加子元素 property，property的两个属性就不说了，一看就能明白！

　　solr.core.name -- The core's name as defined in solr.xml

　　solr.core.instanceDir -- The core's instance directory (i.e. the directory under which that core's conf/ and data/ directory are located)

　　solr.core.dataDir -- The core's data directory (i.e. the directory under which that core's index directory are located)

　　solr.core.configName -- The name of the core's config file (solrconfig.xml by default)

　　solr.core.schemaName -- The name of the core's schema file (schema.xml by default)

　　4》solr.xml具体含义：

　　1）solr

　　The <solr> tag accepts two attributes:

　　persistent - By default, should runtime core manipulation be saved in solr.xml so that it is available after a restart.

　　sharedLib - Path to a directory containing .jar files that are added to the classpath of every core. The path is relative to solr.home (where solr.xml sits)

　　2）cores

　　The <cores> tag accepts two attribute:

　　adminPath - Relative path to access the CoreAdminHandler for dynamic core manipulation. For example, adminPath="/admin/cores" configures access via　 http://localhost:8983/solr/admin/cores. If this attribute is not specified, dynamic manipulation is unavailable.

　　3）core

　　The <core> tag accepts two attributes:

　　name - The registered core name. This will be how the core is accessed.

　　instanceDir - The solr.home directory for a given core.

　　dataDir - The data directory for a given core. The default is <instanceDir>/data . It can take an absolute path or a relative path w.r.t instanceDir .　 Solr1.4

　　4）property

　　The <property> tag accepts two attributes:

　　name - The name of the property

　　value - The value of the property

　　由于E文过于简单，所以就不翻译了！

　　相信很多人，在准备提交数据让solr建立索引的那刻，很纳闷，尽管看了不少网上的一些文章，但是我想依然还是有不少不理解的地方。

　　比如提交一个xml，采用post方式，尽管有些文章说了可以采用httpclient。但是我那个时候，还不是很理解，当然现在看来其实也没有什么了。但是对于一个刚入门solr的初学者，我想讲讲关于solr1.3的 solrj（ sorlr J 目前使用二进制的格式作为默认的格式。对于solr1.2的用户通过显示的设置才能使用XML格式。）！

　　先上一个例子：

　　Java代码

public　static　final　String　SOLR\_URL　=　"http://localhost/solr/core0";

public　static　void　commit()　{

　Date　date　=　new　Date();

　SolrServer　solr　=　null;

　try　{

　solr　=　new　CommonsHttpSolrServer(SOLR\_URL);

　}　catch　(MalformedURLException　e1)　{

　e1.printStackTrace();

　}

for　(int　i　=　0;　i　<　10000;　i++)　{

　　SolrInputDocument　sid　=　new　SolrInputDocument();

　　sid.addField("id",　i);

　　sid.addField("name",　"struts+hibernate+spring　开发大全"　+　i);

　　sid.addField("summary",　"三种框架的综合应用"　+　i);

　　sid.addField("author",　"李良杰"　+　i);

　　sid.addField("date",　new　Date());

　　sid.addField("content",　"高级应用类书籍"　+　i);

　　sid.addField("keywords",　"SSH"　+　i);

　　try　{

　　solr.add(sid);

　　}　catch　(MalformedURLException　e)　{

　　e.printStackTrace();

　　}　catch　(SolrServerException　e)　{

　　e.printStackTrace();

　　}　catch　(IOException　e)　{

　　e.printStackTrace();

　　}

　　System.out.println(i);

　　if　(i　==　999)

　　System.out.println((new　Date().getTime()　-　date.getTime())　/　60000　+　"分钟");

　}

　try　{

　　solr.commit();

　}　catch　(SolrServerException　e)　{

　　e.printStackTrace();

　}　catch　(IOException　e)　{

　　e.printStackTrace();

　}

}

　　上面这段代码的意思是：利用for提交10000个document，并打印提交10000所需的时间。

　　1》CommonsHttpSolrServer 使用HTTPClient 和solr服务器进行通信。

　　2》CommonsHttpSorlrServer 允许设置链接属性。

　　Java代码

server.setSoTimeout(1000);　//　socket　read　timeout

　server.setConnectionTimeout(100);

　server.setDefaultMaxConnectionsPerHost(100);

　server.setMaxTotalConnections(100);

　server.setFollowRedirects(false);　//　defaults　to　false

　//　allowCompression　defaults　to　false.

　//　Server　side　must　support　gzip　or　deflate　for　this　to　have　any　effect.

　server.setAllowCompression(true);

　server.setMaxRetries(1);　//　defaults　to　0.　>　1　not　recommended.

　　3》实现SolrServer接口的另一个类：EmbeddedSorrServer，它不需要http连接。

　　4》在构造document的时候，可以一个一个添加到solrServer，也可以构建一个包含document的Collection，将Collection添加到solrServer，然后commit。

　　5》也可以构造一个跟document匹配的JavaBean进行提交

　　使用 java 注释创建java bean。@Field ，可以被用在域上，或者是setter方法上。如果一个域的名称跟bean的名称是不一样的，那么在java注释中填写别名，具体的，可以参照下面的域categories

　　Java代码

import　org.apache.solr.client.solrj.beans.Field;

　public　class　Item　{

　　@Field

　　String　id;

　　@Field("cat")

　　String[]　categories;

　　@Field

　　List<String>　features;

　}

　　java注释也可以使用在setter方法上，如下面的例子：

　　Java代码

@Field("cat")

　public　void　setCategory(String[]　c){

　　　this.categories　=　c;

　}

　　这里应该要有一个相对的，get方法（没有加java注释的）来读取属性

　　Java代码

Item　item　=　new　Item();

item.id　=　"one";

item.categories　=　new　String[]　{　"aaa",　"bbb",　"ccc"　};

　　添加给solr

　　Java代码

server.addBean(item);

　　将多个bean提交给solr

　　Java代码

List<Item>　beans　;

//add　Item　objects　to　the　list

server.addBeans(beans);

　　注意： 你可以重复使用SolrServer，这样可以提高性能。

　　6》

　　Java代码

public　static　void　update()　{

　SolrServer　solrServer　=　null;

　try　{

　solrServer　=　new　CommonsHttpSolrServer(SOLR\_URL);

　}　catch　(MalformedURLException　e)　{

　e.printStackTrace();

　}

　UpdateRequest　updateRequest　=　new　UpdateRequest();

　SolrInputDocument　sid　=　new　SolrInputDocument();

　sid.addField("id",　100000);

　sid.addField("name",　"struts+hibernate+spring　开发大全");

　sid.addField("summary",　"三种框架的综合应用");

　sid.addField("author",　"李良杰");

　sid.addField("date",　new　Date());

　sid.addField("content",　"高级应用类书籍");

　sid.addField("keywords",　"SSH");

　updateRequest.setAction(UpdateRequest.ACTION.COMMIT,　false,　false);

　updateRequest.add(sid);

　try　{

　UpdateResponse　updateResponse　=　updateRequest.process(solrServer);

　System.out.println(updateResponse.getStatus());

　}　catch　(SolrServerException　e)　{

　e.printStackTrace();

　}　catch　(IOException　e)　{

　e.printStackTrace();

　}

}

　　提交一个document，采用更新方式，注意：

　　Java代码

updateRequest.setAction(UpdateRequest.ACTION.COMMIT,　false,　false);

　　7》

　　Java代码

public　static　void　query()　{

　SolrServer　solr　=　null;

　try　{

　solr　=　new　CommonsHttpSolrServer(SOLR\_URL);

　}　catch　(MalformedURLException　e)　{

　e.printStackTrace();

　return;

　}

　//　http://localhost:8983/solr/spellCheckCompRH?q=epod&spellcheck=on&spellcheck.build=true

　ModifiableSolrParams　params　=　new　ModifiableSolrParams();

　params.set("qt",　"/spellCheckCompRH");

　params.set("q",　"编程");

　params.set("spellcheck",　"on");

　params.set("spellcheck.build",　"true");

　QueryResponse　response　=　null;

　try　{

　response　=　solr.query(params);

　}　catch　(SolrServerException　e)　{

　e.printStackTrace();

　return;

　}

　System.out.println("response　=　"　+　response);

}

　　这是一个查询方法。关键字：“编程”。关于查询的关键字，请参见slor wiki http://wiki.apache.org/solr/QueryParametersIndex ，或等待我的博客更新，在后面会有篇文章详细讲这个问题！

　　8》给solr的索引文件手动进行优化，

solr.optimize();

　　9》solrJ 提供了一组API，来帮助我们创建查询，下面是一个faceted query的例子。

SolrServer　server　=　getSolrServer();

SolrQuery 　solrQuery　=　new　 SolrQuery().setQuery("ipod").setFacet(true).setFacetMinCount(1).setFacetLimit(8). 　　　addFacetField("category").addFacetField("inStock");

QueryResponse　rsp　=　server.query(solrQuery);

　　所有的 setter/add 方法都是返回它自己本身的实例，所以就像你所看到的一样，上面的用法是链式的。

　　在做solr查询的时候，solr提供了很多参数来扩展它自身的强大功能！以下是使用频率最高的一些参数！

　　具体请看：

　　1.常用

　　q - 查询字符串，必须的。查询语句(类似SQL) 相关详细的操作还需lucene 的query 语法

　　fl - 指定返回那些字段内容，用逗号或空格分隔多个。

　　start - 返回第一条记录在完整找到结果中的偏移位置，0开始，一般分页用。

　　rows - 指定返回结果最多有多少条记录，配合start来实现分页。

　　sort - 排序，格式：sort=<field name>+<desc|asc>[,<field name>+<desc|asc>]… 。示例：（score desc, price asc）表示先 “score” 降序, 再 “price” 升序，默认是相关性降序。

　　wt - (writer type)指定输出格式，可以有 xml, json, php, phps, 后面 solr 1.3增加的，要用通知我们，因为默认没有打开。

　　fl表示索引显示那些field(\*表示所有field, score 是solr 的一个匹配热度)

　　q.op 表示q 中 查询语句的 各条件的逻辑操作 AND(与) OR(或)

　　hl 是否高亮

　　hl.fl 高亮field

　　hl.snippets 不太清楚(反正是设置高亮3就可以了)

　　hl.simple.pre 高亮前面的格式

　　hl.simple.post 高亮后面的格式

　　facet 是否启动统计

　　facet.field　 统计field

　　fq - （filter query）过虑查询，作用：在q查询符合结果中同时是fq查询符合的，例如：q=mm&fq=date\_time:[20081001 TO 20091031]，找关键字mm，并且date\_time是20081001到20091031之间的。官方文档：http://wiki.apache.org/solr/CommonQueryParameters#head-6522ef80f22d0e50d2f12ec487758577506d6002

　　2.不常用

　　q.op - 覆盖schema.xml的defaultOperator（有空格时用"AND"还是用"OR"操作逻辑），一般默认指定

　　df - 默认的查询字段，一般默认指定

　　qt - （query type）指定那个类型来处理查询请求，一般不用指定，默认是standard。

　　3.其它

　　indent - 返回的结果是否缩进，默认关闭，用 indent=true|on 开启，一般调试json,php,phps,ruby输出才有必要用这个参数。

　　version - 查询语法的版本，建议不使用它，由服务器指定默认值。

记录下所有在solr中使用的毛病。而且我希望广大的看友们也能一起来说说你们平时遇到的各种错误，这样大家才能一起更好的进步！

　　话不多说，进入正题

　　1》solr 做索引时报 Lock obtain timed out: SingleInstanceLock: write.lock

　　有个频繁做索引的应用，它同时也对外提供搜索服务。大部分是 solr 1.3 的默认配置。solr 做索引，有时候报：

　　Xml代码

2009-7-13　9:48:06　org.apache.solr.common.SolrException　log

严重:　org.apache.lucene.store.LockObtainFailedException:　Lock　obtain　timed　out:　SingleInstanceLock:　write.lock

　　　　at　org.apache.lucene.store.Lock.obtain(Lock.java:85)

　　　　at　org.apache.lucene.index.IndexWriter.init(IndexWriter.java:1140)

　　　　at　org.apache.lucene.index.IndexWriter.<init>(IndexWriter.java:938)

　　　　at　org.apache.solr.update.SolrIndexWriter.<init>(SolrIndexWriter.java:116)

...

　　是写锁取不到。但重启 solr 又可以正常做，主要是运行时间长了就报这个错了。还是看下配置吧。

　　看到 solr 1.3 默认的配置是：

　　Xml代码

<indexDefaults>

　　<!--　...　-->

　　<!--

　　　This　option　specifies　which　Lucene　LockFactory　implementation　to　use.

　　　single　=　SingleInstanceLockFactory　-　suggested　for　a　read-only　index

　　　　　　　　or　when　there　is　no　possibility　of　another　process　trying

　　　　　　　　to　modify　the　index.

　　　native　=　NativeFSLockFactory

　　　simple　=　SimpleFSLockFactory

　　　(For　backwards　compatibility　with　Solr　1.2,　'simple'　is　the　default

　　　　if　not　specified.)

　　-->

　　<lockType>single</lockType>

</indexDefaults>

　　默认锁是 single ，只读的。solr 1.2 是 simple，把它改回去了，运行了几天，没事。

　　2》lucene & solr optimize 索引后结果与平台有点关系

　　昨日做索引的程序重构下，测试 optimize 索引（在原有数据基础上提交索引）时，在开发的机器（windows）里总是会有两段索引，要再 optimize 才只是一个索引段，当然不是设置 maxSegments=2。反复运行还是如此，为了说明是否写的程序有问题，就用 solr 自带的 post.sh （或 post.jar）提交 optimize。结果还是有两段，再提交一次optimize 才是一个段。这问题……

　　旧的程序运行得很正常，看了下它也没有提交两次优化。然后把新的程序也放到服务器（linux）上运行，结果是只有一个段。

　　恩，可以认为是与文件系统有关，optimize 的时候是先新生成一段，然后再删除旧的索引，windows 可能是这样在运行期间与文件关联着删除不了旧的。linux 可能是不用（不打开）文件就可以删除。现只能这样简单解释。

　　3》换 solr 里的 lucene 包

　　solr 1.3 发布的时候，lucene 2.4还没有正式发布，其的 lucene 是开发版，现在lucene 2.4早已发布，那就换上新的 lucene 吧。

　　下载 solr 1.3 http://labs.xiaonei.com/apache-mirror/lucene/solr/1.3.0/apache-solr-1.3.0.zip 和 lucene 2.4 http://labs.xiaonei.com/apache-mirror/lucene/java/lucene-2.4.0.zip 到目录如e:/search/，

　　把 e:/search/apache-solr-1.3/lib 目录下的 lucene 相关的\*.jar删除：

　　lucene-analyzers-2.4-dev.jar

　　lucene-core-2.4-dev.jar

　　lucene-highlighter-2.4-dev.jar

　　lucene-memory-2.4-dev.jar

　　lucene-queries-2.4-dev.jar

　　lucene-snowball-2.4-dev.jar

　　lucene-spellchecker-2.4-dev.jar

　　从 e:/search/lucene-2.4/（或contrib/）目录下找到对应的放到solr-1.3/lib下

　　然后构建 solr，到e:/search/apache-solr-1.3目录，ant dist-war

　　4》solr q查询容错性

　　当solr接收没q参数（或q参数值为空）请求时，会报错。报错十分讨厌，对开发调试时才比较有用，但实际运行环境报错就不太好了，java异常可能有点性能消耗，那干脆就返回正常的结果好了（只是结果里没有找到的数据）。

　　solr 1.3 可以写个组件去做。判断到空的时候，加一个q参数，其值为在索引里没有的数据。这样就可以返回没有数据的结果。

　　其实这样实现还是比较麻烦。可以在 solrconfig.xml的requestHandler里加一个默认参数。如q=abcdefghijk。配置如下：

　　Xml代码

<requestHandler　name="standard"　class="solr.SearchHandler"　default="true">

　　<lst　name="defaults">

　　　<str　name="q">abcdefghijk</str>

　　</lst>

</requestHandler>

　　这样的话查询容错性比较好，查询请求没有带q参数也可，q参数值为空也可。tomcat不会报错，还可以返回结果。

　　引用Bory.Chanhttp://blog.chenlb.com/2009/04/apply-solr-collapsing-patch-remove-duplicate-result.html

　　打上SOLR-236\_collapsing.patch补丁，实现 solr 搜索结果折叠、除去重复的搜索结果，可以实现类似google搜索结果的“站内的其它相关信息 ”。solr collapsing patch 是用 hash 某个字段来实现折叠重复结果的。下面我演示下应用这个补丁并搜索试用下。

　　其实 solr 上已经有了这功能的实现：solr 1.3 collapse patch， 请看：https://issues.apache.org/jira/browse/SOLR-236 ，我这里下载是了新的：https://issues.apache.org/jira/secure/attachment/12403590/SOLR-236\_collapsing.patch 。

　　下载好后就需要打上补丁了，先准备一份源码在D:/apache-solr-1.3.0目录下。没有可以去下载：http: //archive.apache.org/dist/lucene/solr/1.3.0/apache-solr-1.3.0.zip。把SOLR- 236\_collapsing.patch文件放在D:/apache-solr-1.3.0目录下， 打补丁有我知道的有两种：用linux工具 patch（windows 下有 cygwin）；用 ant 的 patch。

　　windows cygwin 的 patch：

　　Html代码

D:\apache-solr-1.3.0>patch　-p0　<　SOLR-236\_collapsing.patch

patching　file　src/test/org/apache/solr/search/TestDocSet.java

patching　file　src/java/org/apache/solr/search/CollapseFilter.java

patching　file　src/java/org/apache/solr/search/DocSet.java

patching　file　src/java/org/apache/solr/search/NegatedDocSet.java

patching　file　src/java/org/apache/solr/search/SolrIndexSearcher.java

patching　file　src/java/org/apache/solr/common/params/CollapseParams.java

patching　file　src/java/org/apache/solr/handler/component/CollapseComponent.java

　　ant patch，把下面的内容保存为 patch-build.xml 放到 D:\apache-solr-1.3.0 目录下：

　　Xml代码

<?xml　version="1.0"　encoding="UTF-8"?>

<project　name="solr-patch"　default="apply-patch"　basedir=".">

　　<target　name="apply-patch"　description="Apply　a　patch　file.　Set　-Dpatch.file">

　　　　<patch　patchfile="${patch.file}"　strip="0"/>

　　</target>

</project>

　　ant 打补丁：

　　Html代码 D:\apache-solr-1.3.0>ant　-Dpatch.file=SOLR-236\_collapsing.patch　-f　patch-build.xml

Buildfile:　patch-build.xml

apply-patch:

[patch]　patching　file　src/test/org/apache/solr/search/TestDocSet.java

[patch]　patching　file　src/java/org/apache/solr/search/CollapseFilter.java

[patch]　patching　file　src/java/org/apache/solr/search/DocSet.java

[patch]　patching　file　src/java/org/apache/solr/search/NegatedDocSet.java

[patch]　patching　file　src/java/org/apache/solr/search/SolrIndexSearcher.java

[patch]　patching　file　src/java/org/apache/solr/common/params/CollapseParams.java

[patch]　patching　file　src/java/org/apache/solr/handler/component/CollapseComponent.java

BUILD　SUCCESSFUL

Total　time:　0　seconds

　　源码打上了补丁，然后用 ant 构建源码：

　　D:\apache-solr-1.3.0>ant dist

　　在 D:/apache-solr-1.3.0/dist 目录下可以找到编译好的 solr 了。然后把 solr 放到 tomcat 中去运行它，把下面的内容保存在 TOMCAT\_HOME/conf/Catalina/localhost/solr.xml 文件中：

　　Xml代码

<Context　docBase="D:\apache-solr-1.3.0\dist\apache-solr-1.3.0.war"　reloadable="true"　>

　　<Environment　name="solr/home"　type="java.lang.String"　value="D:\apache-solr-1.3.0\example\solr"　override="true"　/>

</Context>

　　修改 D:\apache-solr-1.3.0\example\solr\conf\solrconfig.xml 使 solr 可以支持 collapse。

　　定义搜索组件，在 QueryComponent 附近：

　　<searchComponent name="collapse" class="org.apache.solr.handler.component.CollapseComponent" />

　　定义一个 handler 使用上面的搜索组件：

　　Xml代码

<requestHandler　name="collapse"　class="solr.SearchHandler">

　<!--　default　values　for　query　parameters　-->

　　<lst　name="defaults">

　　　<str　name="echoParams">explicit</str>

　　</lst>

　<arr　name="components">

　　<str>collapse</str>

　　<str>debug</str>

　</arr>

</requestHandler>

　　安装启动 tomcat，现在提交一些数据给它，用官方的示例数据就可以了。运行：

　　Html代码 D:\apache-solr-1.3.0\example\exampledocs>java　-Durl=http://localhost:8080/solr/update　-Dcommit=yes　-jar　post.jar　\*.xml

SimplePostTool:　version　1.2

SimplePostTool:　WARNING:　Make　sure　your　XML　documents　are　encoded　in　UTF-8,　other　encodings　are　not　currently　supported

SimplePostTool:　POSTing　files　to　http://localhost:8080/solr/update..

SimplePostTool:　POSTing　file　hd.xml

SimplePostTool:　POSTing　file　ipod\_other.xml

SimplePostTool:　POSTing　file　ipod\_video.xml

SimplePostTool:　POSTing　file　mem.xml

SimplePostTool:　POSTing　file　monitor.xml

SimplePostTool:　POSTing　file　monitor2.xml

SimplePostTool:　POSTing　file　mp500.xml

SimplePostTool:　POSTing　file　sd500.xml

SimplePostTool:　POSTing　file　solr.xml

SimplePostTool:　POSTing　file　spellchecker.xml

SimplePostTool:　POSTing　file　utf8-example.xml

SimplePostTool:　POSTing　file　vidcard.xml

SimplePostTool:　COMMITting　Solr　index　changes..

　　http://localhost:8080/solr/admin/stats.jsp 有结果了？ 有了。然后开始查询试试看。

　　查询：http://localhost:8080/solr/select/?q=\*%3A\*&indent=on& qt=collapse&collapse=true&collapse.field=popularity&collapse.threshold=1

　　结果：

　　Xml代码

<?xml　version="1.0"　encoding="UTF-8"?>

<response>

<lst　name="responseHeader">

　<int　name="status">0</int>

　<int　name="QTime">0</int>

　<lst　name="params">

　<str　name="collapse.field">popularity</str>

　<str　name="fl">id</str>

　<str　name="collapse.threshold">1</str>

　<str　name="indent">on</str>

　<str　name="q">\*:\*</str>

　<str　name="qt">collapse</str>

　<str　name="collapse">true</str>

　</lst>

</lst>

<lst　name="collapse\_counts">

　<str　name="field">popularity</str>

　<lst　name="doc">

　<int　name="SP2514N">4</int>

　<int　name="F8V7067-APL-KIT">1</int>

　<int　name="MA147LL/A">1</int>

　<int　name="TWINX2048-3200PRO">1</int>

　<int　name="VS1GB400C3">3</int>

　<int　name="1">10</int>

　</lst>

　<lst　name="count">

　<int　name="6">4</int>

　<int　name="1">1</int>

　<int　name="10">1</int>

　<int　name="5">1</int>

　<int　name="7">3</int>

　<int　name="0">10</int>

　</lst>

　<str　name="debug">HashDocSet(6)　Time(ms):　0/0/0/0</str>

</lst>

<result　name="response"　numFound="6"　start="0">

　<doc>

　<str　name="id">SP2514N</str>

　</doc>

　<doc>

　<str　name="id">F8V7067-APL-KIT</str>

　</doc>

　<doc>

　<str　name="id">MA147LL/A</str>

　</doc>

　<doc>

　<str　name="id">TWINX2048-3200PRO</str>

　</doc>

　<doc>

　<str　name="id">VS1GB400C3</str>

　</doc>

　<doc>

　<str　name="id">1</str>

　</doc>

</result>

</response>

可以看到　collapse\_counts　相关的输出：

<lst　name="collapse\_counts">

　<str　name="field">popularity</str>

　<lst　name="doc">

　<int　name="SP2514N">4</int>

...

　</lst>

　<lst　name="count">

　<int　name="6">4</int>

　<int　name="1">1</int>

　<int　name="10">1</int>

　<int　name="5">1</int>

　<int　name="7">3</int>

　<int　name="0">10</int>

　</lst>

　<str　name="debug">HashDocSet(6)　Time(ms):　0/0/0/0</str>

</lst>

　　上面的 count 下的内容（它的顺序是result/doc的顺序），表示 popularity=6 相同的结果还有 4 个，与 popularity=1 相同的结果还有 1 个，依此类推。这样就可以显示给用户的界面里提示“相同的其它内容不有N个”。

　　使用的参数有：

　　#启用 collapse 组件

　　collapse=true

　　#用那个字段来 hash 去除重量内容

　　collapse.field=popularity

　　#要结果中可以最多出现几个相同的文档

　　collapse.threshold=1

　　当然还有其它参数，请看：org.apache.solr.common.params.CollapseParams 类。

　　solr的一些查询语法

　　1. 首先假设我的数据里fields有:name, tel, address 预设的搜寻是name这个字段, 如果要搜寻的数据刚好就是 name 这个字段,就不需要指定搜寻字段名称.

　　2. 查询规则:

　　如欲查询特定字段(非预设字段),请在查询词前加上该字段名称加 “:” (不包含”号) 符号,

　　例如: address:北京市海淀区上地软件园 tel:88xxxxx1

　　1>. q代表query input

　　2>. version代表solr版本(建议不要变动此变量)

　　3>. start代表显示结果从哪一笔结果资料开始,预设为0代表第一笔, rows是说要显示几笔数据,预设为10笔

　　(因为有时查询结果可能有几百笔,但不需要显示所有结果,所以预设是从第一笔开始到第十笔)

　　所以若要显示第10到30笔就改为:

　　http: //localhost:8080/solr/select/?indent=on&version=2.2&q=address:北京市海淀区上地软件园+tel:88xxxxx1&version=2.2&start=10&rows= 20&indent=on

　　(indent代表输出的xml要不要缩行.预设为开启 on)

　　3. 另外,要限定输出结果的内容可用 “fl=” 加上你要的字段名称,如以下这个范例:

　　http: //localhost:8080/solr/select/?indent=on&version=2.2&q=text:北京+ OR+text:亿度&start=0&rows=10&fl=name,address,tel

　　在fl=之后加上了name,adress,tel

　　所以结果会如下:

<result name=”response” numFound=”1340″ start=”0″>

<doc>

<str name=”name”>北京亿度</str>

<str name=”address”>北京市海淀区上地软件园</str>

<str name=”tel”>88xxxxxx1</str>

</doc>

<doc>

<str name=”name”>北京亿度</str>

<str name=”address”/>

<str name=”tel”>88xxxxxx1</str>

</doc>

</result>

　　5. 查询 name 或 address:直接输入查询词, 如: 亿度

　　送出的内容即为:

　　name:亿度 AND address:海淀

　　6. 若要搜寻联集结果,请在词与词间空格或加上大写 “OR” (不包含”号).

　　例如: text:海淀 OR text:亿度

　　text:海淀 OR 亿度

　　或

　　海淀 亿度

　　或

　　name:亿度 OR tel:88xxxxxx1

　　或

　　name:亿度 tel:88xxxxxx1

　　5. 若要搜寻交集结果,请在词与词间加上大写 “AND” 或 “+” (不包含”号).

　　例如: text:海淀 AND 亿度

　　或

　　+text:海淀 +text:亿度

　　或

　　name:亿度 AND tel:88xxxxxx1

　　或

　　name: ( +亿度 +海淀)

　　6. 排除查询

　　在要排除的词前加上 “-” (不包含”号) 号

　　例如: 海淀 -亿度

　　搜寻结果不会有包含亿度的词的结果在内

　　7. Group 搜寻

　　使用 “()” 来包含一个group

　　如希望搜寻在店名字段内同时有 “台北”(不包含”号) 及 “火车站”(不包含”号)

　　8. 增加权重: 如要搜寻 “北京 加油站”(不包含”号) 但因为回传太多笔资料内有 “中华”(不包含”号) 或 “加油站”(不包含”号) 的结果,

　　所以想要把有包含 “加油站”(不包含”号)的数据往前排,可使用 “^”(不包含”号)符号在后面加上愈增加的权重数,

　　像是 “2″,则可以这样做:

　　北京 加油站^2

　　会同时搜寻含有北京或加油站的结果,并把加油站这个词加权所以搜寻时会先判断加油站这一个词在

　　搜寻结果中的比重,甚至假设一笔数据内加油站出现过两次以上的就更加会有优先权.

　　查询时在查询词后加上 “^” (不包含”号) 再加上权重分数

　　例如: 亿度 AND “北京”^2

　　或

　　亿度^2 OR 北京

　　9. Wildcard 搜寻使用 “\*” 符号; 如果输入 “中国\*银” (不包含”号), 结果会有中国信托商业银行, 中国输出入银行图书阅览室, 中国商银证券

　　中国及银之间可夹任何长短字词.

　　solr不可谓是个好东西啊，越往下挖掘，他的各种功能逐渐的展现在我的面前，对于solr的架构人员，不得不令人佩服啊。

　　几天前偶尔看到IBM developmentWorks上面的一片文章，看到了数据库数据的导入，以前我一直是这么认为的，像这种导入可以自己去写程序去导入。

　　写程序 可以将数据读出100条，如果你的内存够大，可以是1000条甚至更多，然后放入Collection中，批量提交至solr。或者读取数据写入xml文件中，再将该文件提交到solr等等。但是，在我看到那一篇文章的时候，原来还有这么巧妙的招。

　　废话不多说，入正题。

　　一.首先准备好solr的dataimport功能需要的东西，在solr的下载包中。分别在：

　　1》Solr-1.3.0\dist\apache-solr-dataimporthandler-1.3.0.jar

　　2》E:\education\search\Solr-1.3.0\example\example-DIH\solr\

　　3》你是哪种数据库，提供该数据库的jdbc驱动。

　　二.如果你还不会运行solr，请参考本人的前几篇博客。这里要做的是，先把E:\education\search\Solr-1.3.0 \example\example-DIH\solr\下面的东西拷贝到solr的HOME目录，然后删除rss，这个是另外一个功能是导入rss订阅信息到solr中，确实很强，这都想到了。将jar文件，实际就两个拷贝到tomcat的webapps下面的solr的WEB-INF的lib文件夹下面。

　　三.更改solr Home目录下的conf/solrconfig.xml，其实就是提交一个solrRequestHandler,代码如下：

　　Xml代码

<requestHandler　name="/dataimport"　class="org.apache.solr.handler.dataimport.DataImportHandler">

　　<lst　name="defaults">

　　　<str　name="config">C:\solr-tomcat\solr\db\conf\db-data-config.xml</str>

　　</lst>

　</requestHandler>

　　四.将solr Home目录下面的solrconfig.xml和schema.xml拷贝到db文件夹下面的conf中。

　　五.修改db\conf\db-data-config.xml

　　Xml代码

<dataConfig>

　 <dataSource　type="JdbcDataSource"　driver="com.mysql.jdbc.Driver"　 url="jdbc:mysql://localhost:3306/tuitui"　user="root"　 password="mysql"/>

　　<document　name="shop">

　　　　　　<entity　name="tuitui\_shop"　pk="shopId"　query="select　\*　from　tuitui\_shop">

　　　　　　<field　column="shopid"　name="shopId"　/>

　　　　　　<field　column="shopName"　name="shopName"　/>

　　　　　　<field　column="shopUrl"　name="shopUrl"　/>

　　　　　　<field　column="keyword"　name="keyword"　/>

　　　　　　<field　column="synopsis"　name="synopsis"　/>

　　　　　　<field　column="province"　name="province"　/>

　　　　　　<field　column="city"　name="city"　/>

　　　　　　<field　column="domain"　name="domain"　/>

　　　　　　<field　column="address"　name="address"　/>

　　　　　　<field　column="coordinate"　name="coordinate"　/>

　　　　　　<field　column="shopSspn"　name="shopSspn"　/>

　　　　　　<field　column="phone"　name="phone"　/>

　　　　　　<field　column="createTime"　name="createTime"　/>

　　　　</entity>

　　</document>

</dataConfig>

　　其中的意思我做简单解释，具体大家可以去看看官方wiki。

　　document：一个文档也就是lucene的document这个没什么解释的；

　　entity：主要针对的是一个数据库表；

　　filed：属性column是数据库的字段，name是filed的名字，即schema中的field name

　　http://wiki.apache.org/solr/DataImportHandler

　　我的数据库表结构发出来：

　　图片看不清楚？请点击这里查看原图（大图）。

　　六.启动TOMCAT，输入地址进行导入，导入分为很多模式：我选用的全部倒入模式。

　　http://localhost/solr/dataimport?command=full-import

　　结果：

　　 00C:\solr-tomcat\solr\db\conf\db-data-config.xmlfull-importidle1202009-09-05 21:28:08Indexing completed. Added/Updated: 2 documents. Deleted 0 documents.2009-09-05 21:28:092009-09-05 21:28:090:0:0.579This response format is experimental. It is likely to change in the future.

　　七.在去查询你刚才提交的数据，搞定。

　　最后在说说这个功能。上面的例子只不过是很简单的一个部分。针对solr的MultiCore，通过配置db-data-config.xml也可以实现，还有多表，或者多表关联等等操作只要在db-data-config.xml配置清楚都可以进行数据的导入。

　　在solr1.4中还有更多的扩展功能，这些功能为重建索引提供能很方便的操作。而且，datasource不单单指的是database，可以是xml文件，还可以是来自网络上的等等。

　　在经过使用了庖丁以后，这里说说怎么将目前很火很流行的IK集成进SOLR，其实方法真的很简单，比paoding方便不少。这里很感谢IK的作者，蓝山咖啡，很感谢你为中文分词做出的贡献。 作者博客：http://linliangyi2007.javaeye.com

　　入正题：

　　1》请先去作者博客参看IK下载地址，主要就是一个IKAnalyzer3.1.1Stable.jar。我这里用的是最新版！

　　Java代码 package　com.yeedoo.slor.tokenizer;

import　java.io.Reader;

import　org.apache.lucene.analysis.TokenStream;

import　org.apache.solr.analysis.BaseTokenizerFactory;

import　org.wltea.analyzer.lucene.IKAnalyzer;

public　class　ChineseTokenizerFactory　extends　BaseTokenizerFactory　{

　@Override

　public　TokenStream　create(Reader　reader)　{

　return　new　IKAnalyzer().tokenStream("text",　reader);

　}

}

　　从代码就可以看得出来真的很方便！将它打包放入solr.war中同时还有IK的jar包。如果你不想打包，请去附件下载已经打好的包。

　　2》配置文件

　　Xml代码 <fieldType　name="text"　class="solr.TextField"　positionIncrementGap="100">

　　<analyzer　type="index">

　　<tokenizer　class="com.yeedoo.slor.tokenizer.ChineseTokenizerFactory"　/>

　　<filter　class="solr.StopFilterFactory"　ignoreCase="true"　words="stopwords.txt"　enablePositionIncrements="true"　/>

　　<filter　class="solr.WordDelimiterFilterFactory"　 generateWordParts="1"　generateNumberParts="1"　catenateWords="1"　 catenateNumbers="1"　catenateAll="0"　splitOnCaseChange="1"　/>

　　<filter　class="solr.LowerCaseFilterFactory"　/>

　　<filter　class="solr.EnglishPorterFilterFactory"　protected="protwords.txt"　/>

　　<filter　class="solr.RemoveDuplicatesTokenFilterFactory"　/>

　　</analyzer>

　　<analyzer　type="query">

　　<tokenizer　class="com.yeedoo.slor.tokenizer.ChineseTokenizerFactory"　/>

　　<filter　class="solr.SynonymFilterFactory"　synonyms="synonyms.txt"　ignoreCase="true"　expand="true"　/>

　　<filter　class="solr.StopFilterFactory"　ignoreCase="true"　words="stopwords.txt"　/>

　　<filter　class="solr.WordDelimiterFilterFactory"　 generateWordParts="1"　generateNumberParts="1"　catenateWords="0"　 catenateNumbers="0"　catenateAll="0"　splitOnCaseChange="1"　/>

　　<filter　class="solr.LowerCaseFilterFactory"　/>

　　<filter　class="solr.EnglishPorterFilterFactory"　protected="protwords.txt"　/>

　　<filter　class="solr.RemoveDuplicatesTokenFilterFactory"　/>

　　</analyzer>

　</fieldType>

　　关于这里的配置文件，如果你更详细的了解意思，请参见我的另一篇博客，博客地址：http://lianj-lee.javaeye.com/blog/424474

　　3》如果在启动tomcat没有报错的情况下，可以去solr的admin那里尝试看看该分词器的效果，效果如下图：

　　图片看不清楚？请点击这里查看原图（大图）。

本文来自CSDN博客，转载请标明出处：http://blog.csdn.net/lin\_zyang/archive/2010/01/04/5127498.aspx

本文来自CSDN博客，转载请标明出处：http://blog.csdn.net/lin\_zyang/archive/2010/01/04/5127498.aspx