下载解压solr

进入solr/bin 目录：

启动solr：./solr start -force

检查solr：./solr status

重启：bin/solr restart -p 8983

停止应用：./bin/solr stop -all

## 创建配置core：

### 1用命令创建

只限于用solr自带的web服务器部署的项目

bin/solr create -c my\_core

### 2 tomcat部署时创建core

进入到solrhome文件夹，建立一个文件夹，用于存放你的core，例如我的cores

1 进入到server\solr\configsets\sample\_techproducts\_configs 文件夹，将此文件夹下的conf 目录拷贝到刚刚建立的core的文件夹下，即我的cores文件夹下

2 在浏览器中输入[http://localhost:8080/solr/index.html](http://localhost:8080/solr/index.html" \t "http://blog.csdn.net/crazylzxlzx/article/details/_blank)，进入solr管理页面，点击左侧的add core，在name 和instanceDir下都输入cores(在步骤1中取得名称) ，点击add core即可   
-此时在输入[http://localhost:8080/solr/index.html](http://localhost:8080/solr/index.html" \t "http://blog.csdn.net/crazylzxlzx/article/details/_blank)， 在左侧即可看见刚刚建立的core

然后在进入到\cores目录，此时会发现多了一个data文件夹和一个properties文件，如图：

## **配置conf**

1. 然后把官网下下来的solr项目中solr-6.5.0\server\solr\configsets\  
   data\_driven\_schema\_configs\conf下的所有东西复制到conf中去。（注意不要复制错！）
2. 最后把solr-6.5\solr-6.5.0\example\example-DIH\solr\db\conf下的admin-extra.html， admin-extra.menu-bottom.html ，admin-extra.menu-top.html三个文件也复制到conf中去。

## **配置文件配置数据库信息**

### 导入所需jar包

把mysql所需的jar包和solr-6.0\solr-6.0.0\dist下的solr-dataimporthandler-6.0.0.jar和solr-dataimporthandler-extras-6.0.0.jar都复制到项目WEB-INF\lib下。

### solrconfig.xml

然后在solrconfig.xml文件中加入

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-dataimporthandler-.\*\.jar" />    <lib dir="${solr.install.dir:../../../..}/dist/" regex="mysql-connector-java-5.1.41-bin.jar" />     <lib dir="${solr.install.dir:../../../..}/contrib/analysis-extras/lucene-libs/" regex="lucene-analyzers-smartcn-\d.\*\.jar" />    <lib dir="${solr.install.dir:../../../..}/contrib/extraction/lib" regex=".\*\.jar" />    <lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-cell-\d.\*\.jar" />      <lib dir="${solr.install.dir:../../../..}/contrib/clustering/lib/" regex=".\*\.jar" />    <lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-clustering-\d.\*\.jar" />      <lib dir="${solr.install.dir:../../../..}/contrib/langid/lib/" regex=".\*\.jar" />    <lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-langid-\d.\*\.jar" />      <lib dir="${solr.install.dir:../../../..}/contrib/velocity/lib" regex=".\*\.jar" />    <lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-velocity-\d.\*\.jar" /> |

就是把WEB-INF\lib里面的jar包配置到项目中，我这里用的是相对地址。

在solrconfig.xml的 <requestHandler name="/select" class="solr.SearchHandler">之上添加

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <requestHandler name="/dataimport" **class**="org.apache.solr.handler.dataimport.DataImportHandler  ">      <lst name="defaults">        <str name="config">data-config.xml</str>      </lst>    </requestHandler> |

### data-config.xml

然后在conf下新建solr-data-config.xml文件。里面内容如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | <dataConfig>      <dataSource type="JdbcDataSource" driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost/product" user="xxx" password="xxx"/>      <document>          <entity name="jihes\_goods" query="SELECT                  g.id,                  g.name,                  g.tag,                                  g.sales\_amount,                                  g.view,                                  g.refresh\_time,                  cg.name category\_name                  FROM jihes\_goods AS g                  LEFT JOIN jihes\_goods\_category AS cg ON g.category\_id=cg.id                  WHERE g.status=1"              deltaImportQuery="SELECT                  g.id,                  g.name,                  g.tag,                                  g.sales\_amount,                                  g.view,                                  g.refresh\_time,                  cg.name category\_name                  FROM jihes\_goods AS g                  LEFT JOIN jihes\_goods\_category AS cg ON g.category\_id=cg.id                  WHERE g.status=1 AND g.id='${dih.delta.id}'"              deltaQuery="SELECT                  g.id,                  g.name,                  g.tag,                                  g.sales\_amount,                                  g.view,                                  g.refresh\_time,                  cg.name category\_name                  FROM jihes\_goods AS g                  LEFT JOIN jihes\_goods\_category AS cg ON g.category\_id=cg.id                  WHERE g.status=1 AND FROM\_UNIXTIME(g.update\_time) > DATE\_ADD('${dih.last\_index\_time}', INTERVAL +8 HOUR)">          </entity>      </document>  </dataConfig> |

网上的例子：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?xml version="1.0" encoding="UTF-8"?>      <dataConfig>          <dataSource name="source1" type="JdbcDataSource" driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/solrdata" user="root" password="220316" batchSize="-1" />      　　<document>              <entity name="goods" pk="id"  dataSource="source1"                      query="select \* from  goods"                       deltaImportQuery="select \* from goods where id='${dih.delta.id}'"                      deltaQuery="select id from goods where updateTime> '${dataimporter.last\_index\_time}'">        　　　      <field column="id" name="id"/>      　　　     <field column="name" name="name"/>                  <field column="number" name="number"/>                  <field column="updateTime" name="updateTime"/>      　　　  </entity>      　　</document>      </dataConfig> |

说明：  
dataSource是数据库数据源。Entity就是一张表对应的实体，pk是主键，query是查询语句。Field对应一个字段，column是数据库里的column名，后面的name属性对应着Solr的Filed的名字。其中solrdata是数据库名，goods是表名。

其中deltaQuery是增量索引，原理是从数据库中根据deltaQuery指定的SQL语句查询出所有需要增量导入的数据的ID号。然后根据deltaImportQuery指定的SQL语句返回所有这些ID的数据，即为这次增量导入所要处理的数据。核心思想是：通过内置变量“${dih.delta.id}”和 “${dataimporter.last\_index\_time}”来记录本次要索引的id和最近一次索引的时间。

另外：

<!--  transformer 格式转化：HTMLStripTransformer 索引中忽略HTML标签   --->     
<!--  query:查询数据库表符合记录数据   --->     
<!--  deltaQuery:增量索引查询主键ID    --->    注意这个只能返回ID字段     
<!--  deltaImportQuery:增量索引查询导入的数据  --->     
<!--  deletedPkQuery:增量索引删除主键ID查询  ---> 注意这个只能返回ID字段

添加 transformer="HTMLStripTransformer"   stripHTML="true"是支持过滤疑似html标签的内容

convertType 是支持clob大字段

<?xml version="1.0" encoding="UTF-8"?>    
<dataConfig>     
    <dataSource name="source1" type="JdbcDataSource" driver="com.**[MySQL](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/liuzhen917/article/details/_blank)**.jdbc.Driver"   
url="jdbc:**[mysql](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/liuzhen917/article/details/_blank)**://localhost:3306/baike?failOverReadOnly=false&amp;autoReconnect=true&amp;characterEncoding=utf8&amp;characterSetResults=utf8" user="root1" password="root" batchSize="-1"convertType="true" />    
<document>    
        <entity name="baike" pk="id"  dataSource="source1" query="select t.ID, ty.`NAME`, t.TITLE, t.CONTENT, t.KEYWORD from VOCABULARY\_ENTRY t,VOCABULARY\_TYPE ty where t.TYPE\_ID=ty.ID"  transformer="HTMLStripTransformer">  
<field column="id" name="id"/>    
<field column="NAME" name="vocabulary\_type\_s"/>  
            <field column="CONTENT" name="vocabulary\_ch\_text" stripHTML="true"/>  
<field column="KEYWORD" name="vocabulary\_ch\_keyword"/>              
<field column="TITLE" name="vocabulary\_ch\_title"/>   
</entity>      
</document>    
</dataConfig>

### managed-schema

最后在conf文件下的managed-schema配置field信息：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" />      <!-- doc values are enabled by **default** **for** primitive types such **as** **long** so we don't index the version field  -->      <field name="\_version\_" type="long" indexed="false" stored="false"/>      <field name="category\_name" type="string" indexed="false" stored="false"/>      <field name="name" type="string" indexed="false" stored="false"/>      <field name="tag" type="string" indexed="false" stored="false"/>        <field name="sales\_amount" type="int" indexed="false" stored="false"/>      <field name="view" type="int" indexed="false" stored="false"/>      <field name="refresh\_time" type="int" indexed="false" stored="false"/>        <field name="keyword" type="text\_cn\_ik" indexed="true" stored="true" multiValued="true" /> |
|  |  |

需要注意的是，数据导入到solr后，solr的检索，其实不是逐个字段检索的，而是默认把需要检索的字段合并到一个叫做text的内置字段里，再进行搜索的。这里就需要用到schema.xml中的copyField配置。

<copyField source="title" dest="text"/>  
<copyField source="detail" dest="text"/>

如果加入了中文分词器，则还需要把text字段的type属性设置成你定义的分词器类型

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

在倒数第二行添加对中文分词的支持和过滤html标签

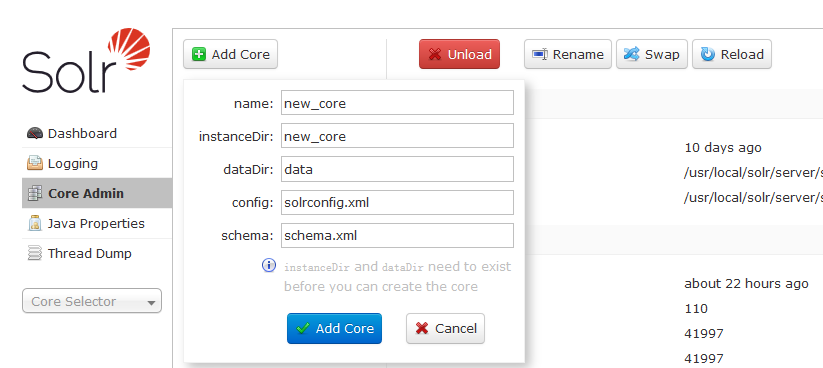
<fieldType name="text\_smartcn" class="solr.TextField" positionIncrementGap="0">  
    <analyzer type="index">  
      <tokenizer class="solr.HMMChineseTokenizerFactory"/>  
    </analyzer>  
    <analyzer type="query">

<charFilter class="solr.HTMLStripCharFilterFactory"/>  
      <tokenizer class="solr.HMMChineseTokenizerFactory"/>  
    </analyzer>  
  </fieldType>

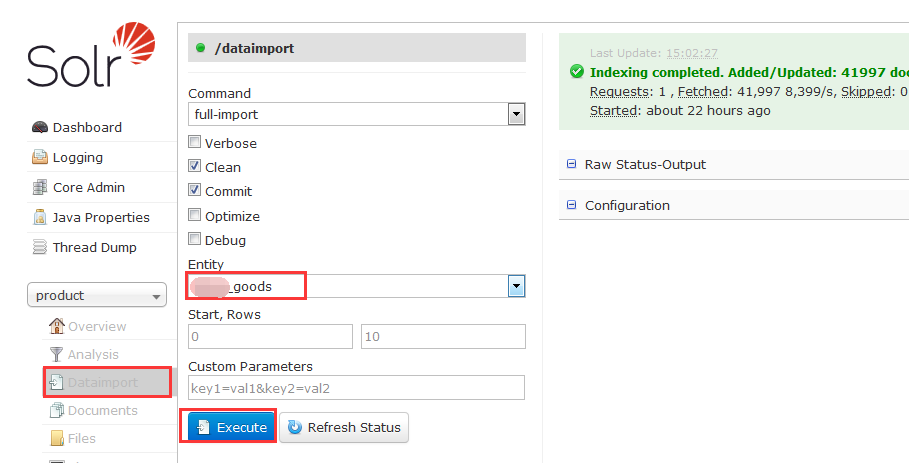
万能的schema配置：

|  |
| --- |
| <schema name="lawinfo" version="1.5">  <dynamicField name="\*" type="text\_ik" indexed="true" stored="true" />  <fieldType name="string" class="solr.StrField" sortMissingLast="true" />  <fieldType name="boolean" class="solr.BoolField" sortMissingLast="true"/>  <fieldType name="int" class="solr.TrieIntField" precisionStep="0" positionIncrementGap="0"/>  <fieldType name="float" class="solr.TrieFloatField" precisionStep="0" positionIncrementGap="0"/>  <fieldType name="long" class="solr.TrieLongField" precisionStep="0" positionIncrementGap="0"/>  <fieldType name="double" class="solr.TrieDoubleField" precisionStep="0" positionIncrementGap="0"/>  <fieldType name="date" class="solr.TrieDateField" precisionStep="0" positionIncrementGap="0"/>  <fieldType name="text\_ik" class="solr.TextField">  <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"/>  </fieldType>  </schema> |

## **创建 core**



## **导入数据库**



## 配置定时增量：

### 1 下载所需的jar包：

<http://download.csdn.net/detail/a491857321/9625835>

将下载的apache-solr-dataimportscheduler-1.0.jar 和solr自带的 apache-solr-dataimporthandler-.jar, apache-solr-dataimporthandler-extras-.jar 放到solr的lib目录下面

### 2、修改solr中WEB-INF/web.xml, 在servlet节点前面增加:

**<listener>**

**<listener-class>**

               org.apache.solr.handler.dataimport.scheduler.ApplicationListener

**</listener-class>**

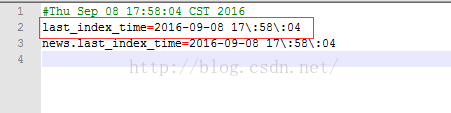
**</listener>**

### 3 配置定时[dataimport.properties](http://download.csdn.net/detail/a491857321/9632472" \t "http://blog.csdn.net/a491857321/article/details/_blank)

将http://download.csdn.net/detail/a491857321/9632472下载并该文件并根据实际情况修改,然后放到 solr.home/conf (不是solr.home/core/conf) 目录下面,该目录是没有的，需要新建（[dataimport.properties](http://download.csdn.net/detail/a491857321/9632472" \t "http://blog.csdn.net/a491857321/article/details/_blank)）

|  |
| --- |
| #################################################  # #  # dataimport scheduler properties #  # #  #################################################  # to sync or not to sync  # 1 - active; anything else - inactive  syncEnabled=1  # which cores to schedule  # in a multi-core environment you can decide which cores you want syncronized  # leave empty or comment it out if using single-core deployment  syncCores=collection1,hangyan  # solr server name or IP address  # [defaults to localhost if empty]  server=localhost  port=8983  # application name/context  # [defaults to current ServletContextListener's context (app) name]  webapp=solr  # URL params [mandatory]  # remainder of URL  #增量  params=/dataimport?command=delta-import&clean=false&commit=true  # schedule interval  # number of minutes between two runs  # [defaults to 30 if empty]  interval=360  # 重做索引的时间间隔，单位分钟，默认7200，即1天;  # 为空,为0,或者注释掉:表示永不重做索引  reBuildIndexInterval=10080  # 重做索引的参数  reBuildIndexParams=/dataimport?command=full-import&clean=true&commit=true  # 重做索引时间间隔的计时开始时间，第一次真正执行的时间=reBuildIndexBeginTime+reBuildIndexInterval\*60\*1000；  # 两种格式：2012-04-11 03:10:00 或者 03:10:00，后一种会自动补全日期部分为服务启动时的日期  reBuildIndexBeginTime=00:00:00 |

可以看出是根据updateTime时间来增量更新索引的，所以新数据的updateTime要大于apache-tomcat-8.0.9\webapps\solr\solrhome\myindex\conf目录下dataimport.properties文件中的last\_index\_time的时间值。



## 配置Ik分词器：

### 首先下载[IKAnalyzer](http://files.cnblogs.com/files/wander1129/ikanalyzer-solr6.5.zip" \t "http://www.cnblogs.com/wander1129/archive/2017/04/05/_blank)这是最新的支持solr6.5.

<http://files.cnblogs.com/files/immortal-ghost/IKAnalyzer.zip>

解压后会有四个文件。

[root@localhost ikanalyzer-solr5]# ls

ext.dic IKAnalyzer.cfg.xml solr-analyzer-ik-5.1.0.jar ik-analyzer-solr5-5.x.jar stopword.dic

ext.dic为扩展字典,stopword.dic为停止词字典，IKAnalyzer.cfg.xml为配置文件，solr-analyzer-ik-5.1.0.jar ik-analyzer-solr5-5.x.jar为分词jar包。

### 2、将文件夹下的IKAnalyzer.cfg.xml , ext.dic和stopword.dic 三个文件 复制到/webapps/solr/WEB-INF/classes 目录下，并修改IKAnalyzer.cfg.xml

[root@localhost ikanalyzer-solr5]# cp ext.dic IKAnalyzer.cfg.xml stopword.dic /down/apache-tomcat-8.5.12/webapps/solr/WEB-INF/classes/

<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd"><properties>

<comment>IK Analyzer 扩展配置</comment>

<!--用户可以在这里配置自己的扩展字典 -->

<entry key="ext\_dict">ext.dic;</entry>

<!--用户可以在这里配置自己的扩展停止词字典-->

<entry key="ext\_stopwords">stopword.dic;</entry>

</properties>

### 3、在ext.dic 里增加自己的扩展词典，例如，唯品会 聚美优品

### 4、复制solr-analyzer-ik-5.1.0.jar ik-analyzer-solr5-5.x.jar到/down/apache-tomcat-8.5.12/webapps/solr/WEB-INF/lib/目录下。

[root@localhost down]# cp /down/ikanalyzer-solr5/solr-analyzer-ik-5.1.0.jar ik-analyzer-solr5-5.x.jar /down/apache-tomcat-8.5.12/webapps/solr/WEB-INF/lib/

### 5、在 solrhome\mycore\conf\managed-schema 文件</schema>前增加如下配置

<!-- IKAnalyzer -->

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

<analyzer type="index">

<tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="false"/>

</analyzer>

<analyzer type="query">

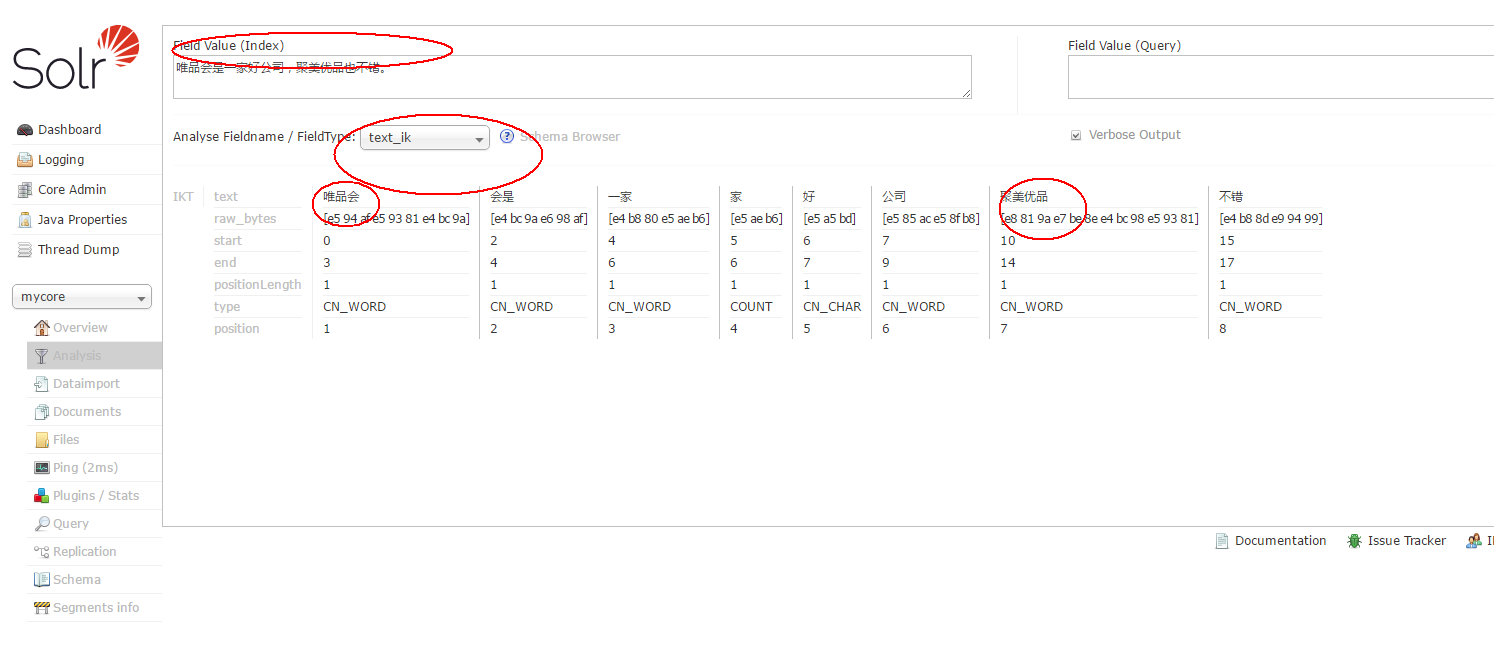
<tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="false"/>

</analyzer>

</fieldType>

注意:  记得将stopword.dic，ext.dic的编码方式为UTF-8 无BOM的编码方式。

重启tomcat查看分词效果。



## **配置拼音检索：**

1. 前期准备，需要用到pinyin4j-2.5.0.jar、pinyinAnalyzer.jar这两个jar包,

[下载地址](http://files.cnblogs.com/files/wander1129/pinyin.zip" \t "http://www.cnblogs.com/wander1129/archive/2017/04/05/_blank)http://files.cnblogs.com/files/wander1129/pinyin.zip

2、将pinyin4j-2.5.0.jar、pinyinAnalyzer.jar这两个jar包复制到/down/apache-tomcat-8.5.12/webapps/solr/WEB-INF/lib/目录下。

[root@localhost down]# cp pinyin4j-2.5.0.jar pinyinAnalyzer4.3.1.jar /down/apache-tomcat-8.5.12/webapps/solr/WEB-INF/lib/

3、在 solrhome\mycore\conf\managed-schema 文件</schema>前增加如下配置：

<fieldType name="text\_pinyin" class="solr.TextField" positionIncrementGap="0">

<analyzer type="index">

<tokenizer class="org.apache.lucene.analysis.ik.IKTokenizerFactory"/>

<filter class="com.shentong.search.analyzers.PinyinTransformTokenFilterFactory" minTermLenght="2" />

<filter class="com.shentong.search.analyzers.PinyinNGramTokenFilterFactory" minGram="1" maxGram="20" />

</analyzer>

<analyzer type="query">

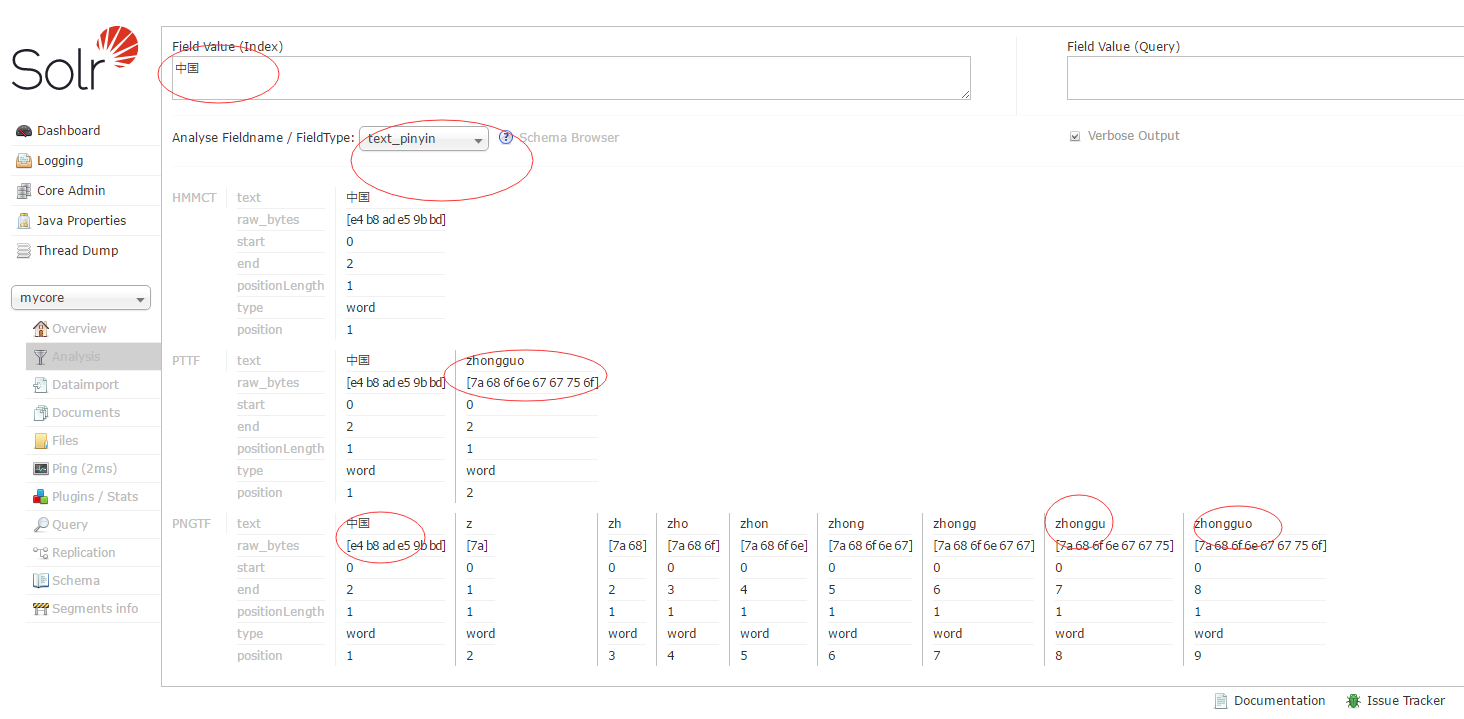
<tokenizer class="org.apache.lucene.analysis.ik.IKTokenizerFactory"/>

<filter class="com.shentong.search.analyzers.PinyinTransformTokenFilterFactory" minTermLenght="2" />

<filter class="com.shentong.search.analyzers.PinyinNGramTokenFilterFactory" minGram="1" maxGram="20" />

</analyzer></fieldType>

重启tomcat查看拼音检索效果。



这里用的是solr自带的中文分词加上pinyin4j来实现的。

## Solr操作：

删除数据：

1、在solr客户端，访问你的索引库**（我认为最方便的方法）**

1)documents type 选择 XML   
2)documents 输入下面语句  
<delete><query>\*:\*</query></delete>  
<commit/>

4、

