# solr 使用说明

## 安装solr

下载地址：<http://mirror.bit.edu.cn/apache/lucene/solr/> 选择.zip压缩包( windows版)

Apache Solr带有一个内置的Jetty服务器。

解压zip包，进入bin目录，cmd控制台输入“solr start”，启动默认端口8983 Solr的服务器。

浏览器访问http://localhost:8983/solr/

查看状态“solr status”;

重启服务“solr restart -p 8983”;

关闭服务“solr stop -all”;

管理页使用说明 https://m.2cto.com/kf/201608/533622.html

## 二、创建core

简单说core就是solr的一个实例，一个solr服务下可以有多个core，每个core下都有索引库和配置文件。

方法1.在bin目录下cmd控制台界面输入 “solr create -c name”。

方法2.访问http://localhost:8983/solr/，点击左侧菜单core admin，创建core。

name：自定义的名字，建议和instanceDir保持一致 ;

instanceDir： solrhome目录下的实例类目 ;

dataDir：默认填data即可 ;

config：指定配置文件，conf/solrconfig.xml ;

schema：指定schema.xml文件，conf/managed-schema;

注意，用方法二之前必须创建好name文件夹以及solrconfig.xml和managed-schema文件。

可直接复制solr-7.6.0\server\solr\configsets\\_default\conf下所有的文件即可。

## 三、导入数据dataimport

以下new\_code为例。

1、 在conf/下创建data-config.xml文件，配置数据库dataSource和数据库表对应的document。

<dataConfig>

<!--数据源 -->

<dataSource type="JdbcDataSource"

driver="oracle.jdbc.driver.OracleDriver"

url="jdbc:oracle:thin:@192.168.1.23:1521:ORCL"

user="gtxdb\_new"

password="gtxdb\_new" />

<document>

<!--entry对应数据库表 -->

<!--query 全量导入sql -->

<!--pk指定主键 -->

<!--deltaImportQuery增量导入sql -->

<!--deltaQuery 增量导入sql （只返回id主键）-->

<entity name="view\_ticket" transformer="DateFormatTransformer"

query="SELECT ticket\_id ,project\_name, cust\_name, destatus, jobs FROM view\_ticket">

pk=”ticket\_id”

deltaImportQuery="SELECT ticket\_id ,project\_name, cust\_name, destatus, jobs FROM view\_ticket"

deltaQuery=”SELECT ticket\_id FROM view\_ticket”

>

<!-- field对应于managed-schema文件中的field字段,column表示数据库字段名 -->

<field name="ticket\_id" column="ticket\_id"/>

<field name="project\_name" column="project\_name"/>

<field name="cust\_name" column="cust\_name"/>

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

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

</entity>

</document>

2、在manage-schema文件编写fields

<field name="ticket\_id" type="string" indexed="true" stored="true" multiValued="false" required="true"/>

<field name="project\_name" type="string" indexed="true" stored="true"/>

<field name="cust\_name" type="string" indexed="true" stored="true"/>

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

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

3、修改solrconfig.xml文件，使加载data-config.xml配置文件

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

<lst name="defaults">

<str name="config">data-config.xml</str>

</lst>

</requestHandler>

4、在solr\server\solr-webapp\webapp\WEB-INF\lib目录下，

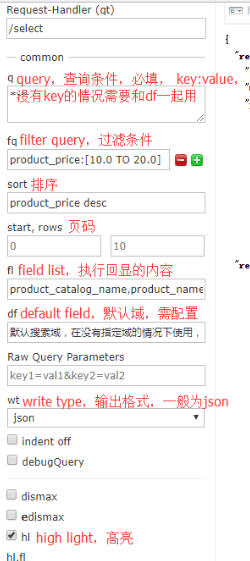
导入ojdbc6.jar（数据库驱动包）和solr-dataimporthandler-7.1.0.jar（solr）。

重启solr服务 “solr restart -p 8983”。

访问http://localhost:8983/solr/，选择core，点击Dataimport，点击Execute。

注意,相同数据不会覆盖，记得勾选Clean，清除相同数据。

6、测试查询，选择core，点击Query,输入条件，点击Execute Query查询结果



Request-Handler(qt)：select查询操作

q（query）：查询条件，key:value 形式，只能满足简单的查询

fq（filter query）：过滤条件。对q的补充，实现复杂的查询。如：product\_price:[10.0 TO 20.0] 表示价格在10~20之间。” \* ” 表示无限，[ \* TO 20.0] 表示小于20.0

sort：对查询结果排序。如：product\_price desc 表示价格降序

start，rows，开始页数，和每页多少条，简称页码

fl（field list）：指定那些字段有返回值。多个值用”,”分隔。如：product\_catalog\_name,product\_name,product\_price

df（default field）：默认域，当q查询没有key的时候，发挥作用

wt（write type）：输出格式，一般都是json

hl（high light）：高亮，搜索的结果若不高亮，那就没啥意义了。下一章会介绍

## 操作数据documents

Documents是solr用户与solr服务器进行交互的基本单元。

solr将一个Document中的数据当做一个整体进行处理，因此用户在传递信息时都是以Document位单位进行传输，一次性可以传送一个或者多个Document单元，solr服务器会解析这些Document对象，来进行相关的处理。

**新增数据：**

访问http://localhost:8983/solr/，选择core，点击Documents，

Request-Handler (qt)： /update

Documents栏里输入：

{

"jobs":"拆机穿墙机<1>,运输装卸穿墙机<1>",

"ticket\_id":"20180321002",

"destatus":"已关闭",

"project\_name":"浦发广州分行运输安装（18）",

"cust\_name":"浦发行广州分行",

"id":"1d90b4b6-880b-47fd-b719-4b66bf91f2ca",

"\_version\_":1623436806215172096

}

点击Submit Document即可

注意 ，字段已在managed-schem文件中定义存在的，id字段相同会覆盖原有数据。

**删除数据：**

访问http://localhost:8983/solr/，选择core，点击Documents，

Request-Handler (qt)： /update

Document Type: XML

Documents栏里输入：

<delete><query>\*:\*</query></delete>  
<commit/>

点击Submit Document即可

## 五、配置schema

以下new\_code为例。

managed-schema这个配置文件的根本目的是为了通过配置告诉Solr怎样建立索引。

solr的数据结构例如以下：

document：一个文档、一条记录

field：域、属性

solr通过搜索某个或某些field，返回若干个符合条件的document，或者按搜索的score排序返回。

假设跟数据库对照，document相当于数据库的表。field相当于表中的字段。而schema.xml就是为了定义一个表的结构（定义各个field的名字、类型、约束、等等）。

schema.xml的基本结构例如以下：

<schema>

<types>

<fields>

<uniqueKey>

<copyField>

</schema>

经常使用的配置说明：

field：定义一个document中的各个fields

name：必填。

该field的名字。前后都有下划线的name是系统保留的名字，比方“\_version\_”

type：必填。类型。相应于fieldType的name

default：该field的缺省值

indexed：true/false。是否为该field建立索引。以让用户能够搜索它、统计它（facet）

stored：true/false。定义这个field能否够返回给查询者

multiValued：true/false。能否够容纳多个值（比方多个copyField的dest指向它）。假设是true。则该field不能被排序、不能作为uniqueKey

required：true/false，告诉solr这个field是否接受空值。缺省为false

docValues：true/false，document-to-value索引，提高特殊搜索的效率（排序、统计、高亮）

copyField：把一个field的内容复制到另外一个field中。一般用来把几个不同的field copy到同一个field中，以方便仅仅对一个field进行搜索

source：被拷贝的field。支持用通配符指定多个field，比方：\*\_name

dest：复制到的目的field

maxChars：最大字符数

uniqueKey：指定一个field为唯一索引

fieldType：定义field的类型，包含以下一些属性

name：必填。被field配置使用

class：必填，filedType的实现类。solr.TextField是路径缩写，"org.apache.solr.schema.TextField"

multiValued：？

positionIncrementGap：指定mutiValued的距离

ananlyzer：假设class是solr.TextField。这个配置是必填的。

告诉solr怎样处理某些单词、怎样分词，比方要不要去掉“a”。要不要所有变成小写……

type：index或query

tokenizer：分词器，比方：StandardTokenizerFactory

filter：过滤器，比方：LowerCaseFilterFactory

dynamicField：用通配符定义一个field来存在没有被field定义的漏网之鱼

name：使用通配符。比方“\*\_i”，来处理类似“cost\_i”之类的field

一个简单的样例：

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

<schema name="course\_video" version="1.5">

<field name="id" type="int" indexed="true" stored="true" required="true" multiValued="false"/>

<field name="\_version\_" type="long" indexed="true" stored="true"/>

<field name="title" type="string" indexed="true" stored="true" required="true" multiValued="false" />

<field name="tags" type="string" indexed="true" stored="false" required="false" multiValued="false" />

<field name="content" type="string" indexed="true" stored="true" required="false"multiValued="false"/>

<field name="info\_text" type="text\_general" indexed="true" stored="false" multiValued="true" />

<copyField source="title" dest="info\_text" />

<copyField source="content" dest="info\_text" />

<copyField source="tags" dest="info\_text" />

<uniqueKey>id</uniqueKey>

<fieldType name="string" class="solr.StrField" sortMissingLast="true" />

<fieldType name="int" class="solr.TrieIntField" precisionStep="0" positionIncrementGap="0"/>

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

<analyzer type="index">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

<analyzer type="query">

<tokenizer class="solr.StandardTokenizerFactory"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

</fieldType>

</schema>

注：solrconfig.xml配置文件里，requestHandler "/select"的缺省“df”是“text”。

假设依照上面的配置。我们想要的缺省搜索info\_text，要在solrconfig.xml中改动：

<requestHandler name="/select" class="solr.SearchHandler">

<lst name="defaults">

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

<int name="rows">10</int>

<str name="defType">edismax</str>

<str name="df">info\_text</str>

</lst>

</requestHandler>

性能说明:可以如下来提高性能。

- 设置 stored="false" 对那些只需要搜索，无需返回的字段.

- 设置 indexed="false" 对于那些只用于返回无需进行搜索的字段.

- 删除所有不需要 copyfiled字段的声明

- 为了最好的索引大小与索引性能，设置所有一般的文本字段index=false,使用copyfile将他们copy到一个字段上，然后使用它进行搜索。

- 运行jvm服务器模式，并使用较高的日志级别，避免记录每一个请求。

## 六、配置IK中文分词器

以下new\_code为例。

先下载solr7版本的ik分词器，下载地址：[http://search.maven.org/#search%7Cga%7C1%7Ccom.github.magese](http://search.maven.org/" \l "search|ga|1|com.github.magese" \t "https://blog.csdn.net/u011052863/article/details/_blank)

把下载好的jar包放入solr-7.6.0\server\solr-webapp\webapp\WEB-INF\lib目录中

然后修改new\_code下的managed-schema文件，需改动：

<!-- ik分词器 -->

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

<analyzer type="index">

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

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

<analyzer type="query">

<tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" useSmart="true" conf="ik.conf"/>

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

</fieldType>

重启solr服务 “solr restart -p 8983”。

访问http://localhost:8983/solr/，选择core，点击Analysis，

Analyse Fieldname / FieldType:选择text\_id ,Field Value (Index):输入词组.

点击Analyse Values，分词结果。

## 七、高亮显示hl

注意高亮显示，必须要有查询条件，与高亮的字段要相同。

高亮参数详细说明：

• hl.fl: 用空格或逗号隔开的字段列表。要启用某个字段的highlight功能，就得保证该字段在schema中是stored。如果该参数未被给出，那么就会高亮默认字段 standard handler会用df参数，dismax字段用qf参数。你可以使用星号去方便的高亮所有字段。如果你使用了通配符，那么要考虑启用hl.requiredFieldMatch选项。

• hl.requireFieldMatch:

如果置为true，除非该字段的查询结果不为空才会被高亮。它的默认值是false，意味着它可能匹配某个字段却高亮一个不同的字段。如果hl.fl使用了通配符，那么就要启用该参数。尽管如此，如果你的查询是all字段（可能是使用copy-field 指令），那么还是把它设为false，这样搜索结果能表明哪个字段的查询文本未被找到

• hl.usePhraseHighlighter:

如果一个查询中含有短语（引号框起来的）那么会保证一定要完全匹配短语的才会被高亮。

• hl.highlightMultiTerm

如果使用通配符和模糊搜索，那么会确保与通配符匹配的term会高亮。默认为false，同时hl.usePhraseHighlighter要为true。

• hl.snippets：

这是highlighted片段的最大数。默认值为1，也几乎不会修改。如果某个特定的字段的该值被置为0（如f.allText.hl.snippets=0），这就表明该字段被禁用高亮了。你可能在hl.fl=\*时会这么用。

• hl.fragsize:

每个snippet返回的最大字符数。默认是100.如果为0，那么该字段不会被fragmented且整个字段的值会被返回。大字段时不会这么做。

• hl.mergeContiguous:

如果被置为true，当snippet重叠时会merge起来。

• hl.maxAnalyzedChars:

会搜索高亮的最大字符，默认值为51200，如果你想禁用，设为-1

• hl.alternateField:

如果没有生成snippet（没有terms 匹配），那么使用另一个字段值作为返回。

• hl.maxAlternateFieldLength:

如果hl.alternateField启用，则有时需要制定alternateField的最大字符长度，默认0是即没有限制。所以合理的值是应该为

• hl.snippets \* hl.fragsize这样返回结果的大小就能保持一致。

• hl.formatter:一个提供可替换的formatting算法的扩展点。默认值是simple，这是目前仅有的选项。显然这不够用，你可以看看org.apache.solr.highlight.HtmlFormatter.java 和 solrconfig.xml中highlighting元素是如何配置的。

注意在不论原文中被高亮了什么值的情况下，如预先已存在的em tags，也不会被转义，所以在有时会导致假的高亮。

• hl.fragmenter:

这个是solr制定fragment算法的扩展点。gap是默认值。regex是另一种选项，这种选项指明highlight的边界由一个正则表达式确定。这是一种非典型的高级选项。为了知道默认设置和fragmenters (and formatters)是如何配置的，可以看看solrconfig.xml中的highlight段。

regex 的fragmenter有如下选项：

• hl.regex.pattern:正则表达式的pattern

• hl.regex.slop:这是hl.fragsize能变化以适应正则表达式的因子。默认值是0.6，意思是如果hl.fragsize=100那么fragment的大小会从40-160

## 八、solrJ(java客户端)

看代码...