

赠书 | AI专栏 (AI圣经!《深度学习》中文版) 每周荐书: Kotlin、分布式、Keras (评论送书) 【获奖公布】征文 | 你会为 AI 转型么?

solr教程，值得刚接触搜索开发人员一看

2013-11-26 13:52 187

分类: Solr搜索应用服务器 (19)

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?)

[+]

Solr调研总结

开发类型	全文检索相关开发	
Solr版本	4.2	
文件内容	本文介绍solr的功能使用及相关注意事项;主要包括以下内容:环境搭建及调试;两个核心配置文件介绍;维护索引;查询索引,和在查询中可以应用的高亮显示、拼写检查、搜索建议、分组统计、拼音检索等功能的使用方法。	
版本	作者/修改人	日期
V1.0	gzk	2013-06-04

1. Solr 是什么？

Solr它是一种开放源码的、基于 **Lucene Java** 的搜索服务器，易于加入到 **Web** 应用程序中。Solr 提供了层面搜索(就是统计)、命中醒目显示并且支持多种JSON等格式）。它易于安装和配置，而且附带了一个基于HTTP 的管理界面。可以使用 **Solr** 的表现优异的基本搜索功能，也可以对它进行扩展从而满足企

- 高级的全文搜索功能
- 专为高通量的网络流量进行的优化
- 基于开放接口（XML和HTTP）的标准
- 综合的HTML管理界面
- 可伸缩性—能够有效地复制到另外一个Solr搜索服务器
- 使用XML配置达到灵活性和适配性
- 可扩展的插件体系

2. Lucene 是什么？

Lucene是一个基于Java的全文信息检索工具包，它不是一个完整的搜索应用程序，而是为你的应用程序提供索引和搜索功能。Lucene 目前是 **Apache Jak** 目。也是目前最为流行的基于Java开源全文检索工具包。目前已经有很多应用程序的搜索功能是基于 **Lucene** ，比如Eclipse 帮助系统的搜索功能。Lucene引，所以你只要把你索引的数据格式转化的文本格式，**Lucene** 就能对你的文档进行索引和搜索。

3. Solr vs Lucene

Solr与Lucene 并不是竞争对立关系，恰恰相反Solr 依存于Lucene，因为Solr底层的核心技术是使用Lucene 来实现的，Solr和Lucene的本质区别有以下三，Lucene本质上是搜索库，不是独立的应用程序，而Solr是。Lucene专注于搜索底层的建设，而Solr专注于企业应用。Lucene不负责支撑搜索服务所必须的话概括 Solr: Solr是Lucene面向企业搜索应用的扩展。

Solr与Lucene架构图:



Solr使用Lucene并且扩展了它！

- 一个真正的拥有动态字段(Dynamic Field)和唯一键(Unique Key)的数据模式(Data Schema)
- 对Lucene查询语言的强大扩展！
- 支持对结果进行动态的分组和过滤
- 高级的，可配置的文本分析
- 高度可配置和可扩展的缓存机制
- 性能优化
- 支持通过XML进行外部配置
- 拥有一个管理界面
- 可监控的日志
- 支持高速增量式更新(Fast incremental Updates)和快照发布(Snapshot Distribution)

4.搭建并调试Solr

4.1 安装虚拟机

Solr 必须运行在Java1.6 或更高版本的Java 虚拟机中，运行标准Solr 服务只需要安装JRE 即可，但如果需要扩展功能或编译源码则需要下载JDK 来完成。

或JRE：

- OpenJDK （ <http://java.sun.com/j2se/downloads.html> ）
- Sun （<http://java.sun.com/j2se/downloads.html> ）
- IBM （<http://www.ibm.com/developerworks/java/jdk/> ）
- Oracle （<http://www.oracle.com/technology/products/jrocket/index.html> ）

安装 步骤请参考相应的帮助文档。

4.2 下载Solr

本文针对Solr4.2版本进行调研的，下文介绍内容均针对Solr4.2版本， 如与Solr 最新版本有出入请以官方网站内容为准。Solr官方网站下载地址：<http://lucene.apache.org/solr/>

4.3 下载并设置Apache Ant

Solr是使用Ant进行管理的源码, Ant是一种基于Java的build工具。理论上来说，它有些类似于Maven 或者是 C中的make。下载后解压出来后，进行环境变量设置：ANT_HOME: E:\Work\apache-ant\1.9.1 (这里为你自己解压缩的目录) PATH: %ANT_HOME%\bin （这个设置是为了方便在dos环境下操作）

查看是否安装成功，在命令行窗口中输入命令ant，若出现结果：



说明ant安装成功！因为ant默认运行build.xml文件，这个文件需要我们建立。现在就可以进行build Solr源码了。在命令行窗口中进入到你的Solr源码目录，输入ant。



其它的先不用管它，我们只要针对我们使用的IDE进行build就行了，如果使用eclipse就在命令行输入：ant eclipse.如果使用IntelliJ IDEA 就在命令行输入：ant eclipse。黑窗口里提示这个。。。



失败。。。为什么呢，最后我发现是因为下载的ant中少了一个jar就是这apache-ivy（下载地址：<http://ant.apache.org/ivy/>）这东东名子真怪 ivy是ant管理ivy会自动把build中的缺少的依赖进行下载。网速慢的第一次build要好久的。。。

下载一个jar就行把jar放到ant的lib下（E:\Work\apache-ant\1.9.1\lib）这样再次运行ant 就会成功了。到现在才可以进行Solr的代码调试。

4.4 配置并运行Solr代码

不管用什么IDE首选都要设置Solr Home在IDE的JVM参数设置VM arguments写入 -Dsolr.solr.home=solr/example/solr一般就行了.不行也可以使用绝对路径。solr使用StartSolrJetty文件作为入口文件进行调试代码,在这里可以设置服务器使用的端口和solr的webapps目录.一般都不用设置,默认的就可以进行调试.Solr 4.2.0 好用。System.setProperty("solr.solr.home", "E:\\Work\\solr-4.2.0-src-idea\\solr\\example\\solr");

目前是使用自带的一个example作为solr配置的根目录，如果你有其他的solr配置目录，设置之即可。点击run即可， debug也是一样可以用了。没有别的问器使用的端口,如查提示：

FAILED SocketConnector@0.0.0.0:8983: java.NET.BindException: Address already in use: JVM_Bind 就说明当前端口占用中.改一下就可以了.如果没有输入地址: <http://localhost:8983/solr/> 就可以看到如下界面



到这里Solr就成功配置并运行了.要是想跟代码调试在启动时在这个方法里点断点就可以Initializer的initialize()方法如果想从浏览器中找断点调试就要到SolrC点了。

注：IE9在兼容模式下有bug，必须设置为非兼容模式。

5.Solr基础

因为 Solr 包装并扩展了Lucene，所以它们使用很多相同的术语。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，Solr 可以阅读和使用构建到其他 Lucene 应用程序中的索引。在 Solr 和 Lucene 中，使用一个或多个 Document 来构建索引。Document 包括一个或多个 Field，以及告诉 Solr 如何处理内容的元数据。

例如，Field 可以包含字符串、数字、布尔值或者日期，也可以包含你想添加的任何类型，只需用在solr的配置文件中进行相应的配置即可。Field 可以使用 Solr 在索引和搜索期间如何处理内容。

现在，查看一下表 1 中列出的重要属性的子集：

属性名称	描述
Indexed	Indexed Field 可以进行搜索和排序。你还可以在 indexed Field 上

	运行 Solr 分析过程，此过程可修改内容以改进或更改结果。
Stored	stored Field 内容保存在索引中。这对于检索和醒目显示内容很有用，但对于实际搜索则不是必需的。例如，很多应用程序存储指向内容位置的指针而不是存储实际的文件内容。

5.1模式配置Schema.xml

schema.xml这个配置文件可以在你下载solr包的安装解压目录的\solr\example\solr\collection1\conf中找到，它就是solr模式关联的文件。打开这个配置文件组织主要分为三个重要配置

5.1.1. types 部分

是一些常见的可重用定义，定义了 Solr（和 Lucene）如何处理 Field。也就是添加到索引中的xml文件属性中的类型，如int、text、date等。

```
<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="text_general" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" enablePositionIncrements="true" />
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" enablePositionIncrements="true" />
    <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

参数说明:

属性	描述
name	标识而已
class	和其他属性决定了这个fieldType的实际行为。
sortMissingLast	设置成true没有该field的数据排在有该field的数据之后，而不管请求时的排序规则, 默认是设置成false。
sortMissingFirst	跟上面倒过来呗。默认是设置成false
analyzer	字段类型指定的分词器
type	当前分词用于的操作.index代表生成索引时使用的分词器query代码在查询时使用的分词器
tokenizer	分词器类

filter 分词后应用的过滤器 过滤器调用顺序和配置相同.

5.1.2. fields

是你添加到索引文件中出现的属性名称，而声明类型就需要用到上面的types

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

<field name="path" type="text_smartcn" indexed="false" stored="true" multiValued="false" termVector="true" />

<field name="content" type="text_smartcn" indexed="false" stored="true" multiValued="false" termVector="true"/>

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

<field name ="pinyin" type ="text_pinyin" indexed ="true" stored ="false" multiValued ="false"/>

<field name="_version_" type="long" indexed="true" stored="true"/>

<dynamicField name="*_i" type="int" indexed="true" stored="true"/>

<dynamicField name="*_l" type="long" indexed="true" stored="true"/>

<dynamicField name="*_s" type="string" indexed="true" stored="true" />
```

- field: 固定的字段设置
- dynamicField: 动态的字段设置,用于后期自定义字段,*号通配符.例如: test_i就是int类型的动态字段.

还有一个特殊的字段copyField,一般用于检索时用的字段这样就只对这一个字段进行索引分词就行了copyField的dest字段如果有多个source一定要设置mul

```
<copyField source="content" dest="pinyin"/>

<copyField source="content" dest="text"/>

<copyField source="pinyin" dest="text"/>
```

字段属性说明:

属性	描述
name	字段类型名
class	java类名
indexed	缺省true。说明这个数据应被搜索和排序，如果数据没有indexed，则stored应是true。
stored	缺省true。说明这个字段被包含在搜索结果中是合适的。如果数据没有stored,则indexed应是true。
omitNorms	字段的长度不影响得分和在索引时不做boost时，设置它为true。 一般文本字段不设置为true。
termVectors	如果字段被用来做more like this 和highlight的特性时应设置为true。
compressed	字段是压缩的。这可能导致索引和搜索变慢，但会减少存储空间，只有StrField和TextField是可以压缩，这通常适合字段的长度超过200个字符。
multiValued	字段多于一个值的时候，可设置为true。

positionIncrementGap和multiValued一起使用，设置多个值之间的虚拟空白的数量

注意:_version_ 是一个特殊字段,不能删除,是记录当前索引版本号的.

5.1.3. 其他配置

uniqueKey: 唯一键，这里配置的是上面出现的fileds，一般是id、url等不重复的。在更新、删除的时候可以用到。

defaultSearchField:默认搜索属性，如q=solr就是默认的搜索那个字段

solrQueryParser:查询转换模式，是并且还是或者（AND/OR必须大写）

5.2. solr配置solrconfig.xml

solrconfig.xml这个配置文件可以在你下载solr包的安装解压目录的E:\Work\solr-4.2.0-src-idea\solr\example\solr\collection1\conf中找到，这个配置文件配置,包含依赖的jar和Solr的一些插件;组件信息配置;索引配置和查询配置,下面详细说一下索引配置和查询配置.

5.2.1索引indexConfig

Solr 性能因素，来了解与各种更改相关的性能权衡。表 1 概括了可控制 Solr 索引处理的各种因素：

属性	描述
useCompoundFile	通过将很多 Lucene 内部文件整合到一个文件来减少使用中的文件的数量。这可有助于减少 Solr 使用的文件句柄数目，代价是降低了性能。除非是应用程序用完了文件句柄，否则 false 的默认值应该就已经足够。
ramBufferSizeMB	在添加或删除文档时，为了减少频繁的更些索引,Solr会选缓存在内存中,当内存中的文件大于设置的值,才会更新到索引库。较大的值可使索引时间变快但会牺牲较多的内存。
maxBufferedDocs	如两个值同时设置,满足一个就会进行刷新索引。
mergeFactor	决定低水平的 Lucene 段被合并的频率。较小的值（最小为 2）使用的内存较少但导致的索引时间也更慢。较大的值可使索引时间变快但会牺牲较多的内存。
maxIndexingThreads	indexWriter生成索引时使用的最大线程数
unlockOnStartup	unlockOnStartup 告知 Solr 忽略在多线程环境中用来保护索引的锁定机制。在某些情况下，索引可能会由于不正确的关机或其他错误而一直处于锁定，这就妨碍了添加和更新。将其设置为 true 可以禁用启动锁定，进而允许进行添加和更新。
lockType	single: 在只读索引或是没有其它进程修改索引时使用。 native: 使用操作系统本地文件锁,不能使用多个Solr在同一个JVM中共享一个索引。 simple :使用一个文本文件锁定索引。

5.2.2 查询配置query

属性	描述
maxBooleanClauses	最大的BooleanQuery数量. 当值超出时，抛出 TooManyClausesException.注意这个是全局的,如果是多个SolrCore都会使用一个值,每个Core里设置不一样的化,会使用最后一个的.
filterCache	filterCache存储了无序的lucene document id集合， 1.存储了filter queries(“fq”参数)得到的 document id集合结果。2还可用于facet查询3. 3)

	如果配置了useFilterForSortedQuery，那么如果查询有filter，则使用filterCache。
queryResultCache	缓存搜索结果,一个文档ID列表
documentCache	缓存Lucene的Document对象,不会自热
fieldValueCache	字段缓存使用文档ID进行快速访问。默认情况下创建fieldValueCache即使这里没有配置。
enableLazyFieldLoading	若应用程序预期只会检索 Document 上少数几个 Field，那么可以将属性设置为 true。延迟加载的一个常见场景大都发生在应用程序返回和显示一系列搜索结果的时候，用户常常会单击其中的一个来查看存储在此索引中的原始文档。初始的显示常常只需要显示很短的一段信息。若考虑到检索大型 Document 的代价，除非必需，否则就应该避免加载整个文档。
queryResultWindowSize	一次查询中存储最多的doc的id数目。
queryResultMaxDocsCached	查询结果doc的最大缓存数量, 例如要求每页显示10条,这里设置是20条,也就是说缓存里总会给你多出10条的数据.让你点示下一页时很快拿到数据.
listener	选项定义 newSearcher 和 firstSearcher 事件，您可以使用这些事件来指定实例化新搜索程序或第一个搜索程序时应该执行哪些查询。如果应用程序期望请求某些特定的查询，那么在创建新搜索程序或第一个搜索程序时就应该反注释这些部分并执行适当的查询。
useColdSearcher	是否使用冷搜索,为false时使用自热后的searcher
maxWarmingSearchers	最大自热searcher数量

5.3Solr加入中文分词器

中文分词在solr里面是没有默认开启的，需要我们自己配置一个中文分词器。目前可用的分词器有smartcn，IK，Jeasy，庖丁。其实主要是两种，一种是科夫HMM算法的中文分词器，如smartcn，ictclas4j，优点是分词准确度高，缺点是不能使用用户自定义词库；另一种是基于最大匹配的分词器，如IK，Jieba，增加新词，缺点是分出来的垃圾词较多。各有优缺点看应用场合自己衡量选择吧。

下面给出两种分词器的安装方法，任选其一即可，推荐第一种，因为smartcn就在solr发行包的contrib/analysis-extras/lucene-libs/下，就是lucene-ana-solrconfig.xml中加一句引用analysis-extras的配置,这样我们自己加入的分词器才会引到的solr中。

```
<lib dir="../../../contrib/analysis-extras/lib" regex=".*\.jar" />
```

5.3.1. smartcn 分词器的安装

首选将发行包的contrib/analysis-extras/lucene-libs/ lucene-analyzers-smartcn-4.2.0.jar复制到\solr\contrib\analysis-extras\lib下,在solr本地应用文件夹编辑text字段类型如下，添加以下代码到scheme.xml中的相应位置，就是找到fieldType定义的那一段，在下面多添加这一段就好啦

```
<fieldType name="text_smartcn" class="solr.TextField" positionIncrementGap="0">

  <analyzer type="index">

    <tokenizer class="org.apache.lucene.analysis.cn.smart.SmartChineseSentenceTokenizerFactory"/>

    <filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

  </analyzer>

  <analyzer type="query">
```

```
<tokenizer class="org.apache.lucene.analysis.cn.smart.SmartChineseSentenceTokenizerFactory"/>

<filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

</analyzer>

</fieldType>
```

如果需要检索某个字段, 还需要在scheme.xml下面的field中, 添加指定的字段, 用text_smartcn作为type的名字, 来完成中文分词。如 text要实现中文

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

5.3.2. IK 分词器的安装

首选要去下载IKAnalyzer的发行包.下载地址: http://ik-analyzer.googlecode.com/files/IK%20Analyzer%202012FF_hf1.zip.

下载后解压出来文件中的三个复制到\solr\contrib\analysis-extras\lib目录中。

IKAnalyzer2012FF_u1.jar	分词器jar包
IKAnalyzer.cfg.xml	分词器配置文件
Stopword.dic	分词器停词字典,可自定义添加内容

复制后就可以像smartcn一样的进行配置scheme.xml了。

```
<fieldType name="text_ik" class="solr.TextField">

    <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"/>

</fieldType>

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

现在来验证下是否添加成功,首先使用StartSolrJetty来启动solr服务,启动过程中如果配置出错,一般有两个原因:一是配置的分词器jar找不到,也就是你没找到\solr\contrib\analysis-extras\lib目录下;二是分词器版本不对导致的分词器接口API不一样出的错,要是这个错的话就在检查分词器的相关文档,看一下支持的版本。

如果在启动过程中没有报错的话说明配置成功了。我们可以进入到<http://localhost:8983/solr>地址进行测试一下刚加入的中文分词器。在首页的Core Selection的Analysis,在Analyse Fieldname / FieldType里选择你刚才设置的字段名称或是分词器类型,在Field Value(index)中输入:中国人,点击右面的分词就行了。

6.Solr功能应用

我这里主要使用SolrJ进行介绍一下Solr的一些基本应用,使用SolrJ加上EmbeddedSolrServer(嵌入式服务器),方便进行代码跟踪调试。在功能上和其它服务器来提供服务API的。EmbeddedSolrServer优点是不用起http协议,直接加载SolrCore进行操作,性能上应该是最快的,方便用于把Solr单结点服务嵌入到应用的功能的应用.EmbeddedSolrServer初始化:

```
System.setProperty("solr.solr.home", "E:\\Work\\solr-4.2.0-src\\solr\\example\\solr");

CoreContainer.Initializer initializer = new CoreContainer.Initializer();

CoreContainer coreContainer = initializer.initialize();

SolrServer server = new EmbeddedSolrServer(coreContainer, "");
```

6.1维护索引

在一般系统中维护的都是增删改,在Solr中的维护功能是增删和优化功能,在Solr中的修改操作就是先删掉再添加。在做索引维护之前,首先要做的是配置schema说明设置好字段信息(名称,类型,索引,存储,分词等信息),大概就像在数据库中新建一个表一样。设置好schema.xml就可以进行索引相关操作了。

6.1.1增加索引

在增加索引之前先可构建好SolrInputDocument对象.主要操作就是给文档添加字段和值.代码如下:

```
SolrInputDocument doc = new SolrInputDocument();

doc.setField("id", "ABC");

doc.setField("content", "中华人民共和国");
```

构建好文档后添加的上面初始化好的server里就行了。


```
server.add(doc);
```

```
server.commit();//这句一般不用加因为我们可以通过在配置文件中的
```

```
//autoCommit来提高性能
```

Solr在add文档时.如果文档不存在就直接添加,如果文档存在就删除后添加,这也就是修改功能了.判断文档是否存在的依据是定义好的uniqueKey字段.

6.1.2删除索引

删除索引可以通过两种方式操作,一种是通过文档ID进行删除,别一种是通过查询到的结果进行删除.

通过ID删除方式代码:

```
server.deleteById(id);
```

```
//或是使用批量删除
```

```
server.deleteById(ids);
```

通过查询删除方式代码:

```
server.deleteByQuery("*.");//这样就删除了所有文档索引
```

```
//"*. "就查询所有内容的,介绍查询时会详细说明.
```

6.1.2优化索引

优化Lucene 的索引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。一个索引是一个耗时较多的过程。

```
server.optimize();//不要频繁的调用..尽量在无人使用时调用.
```

6.2查询索引

Solr在不修改任务配置的情况下就可以使用查询功能，在web项目中应用可以直接URL进行访问Solr服务器例如：

http://localhost:8983/solr/ collection1/select?q=%3A*&wt=xml&indent=true

上面的意思就是查询名为collection1的SolrCore的所有内容用xml格式返回并且有缩进。

返回结果如下:

```
<?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="indent">true</str>
```

```
<str name="q">*. *</str>
```

```
<str name="wt">xml</str>
```

```
</lst>
```

```
</lst>
```

```
<result name="response" numFound="17971" start="0">
```

```
<doc>
```

```
<str name="path">E:\Reduced\军事\1539.txt</str>
```

```
<str name="category_s">2</str>

<int name="browseCount_i">-1423701734</int>

<long name="modified_l">1162438568000</long>

<long name="releasedate_l">1162438568000</long>

<str name="content"> [俄罗斯lenta网站2006年2月9日报道]俄空军副总司令比热耶夫中将称，2006年春天独联体国家防空系统打击范围向西推进150千米。 2006年3月白俄罗斯4个S-300PS防空导弹营担负战斗任务，使独联体防空系统作战范围得以向西推进。比热耶夫中将还宣布，近期乌兹别克斯坦和独联体国家防空系统建于9年前，共有9个国家参加该组织。目前只有亚美尼亚、白俄罗斯、哈萨克斯坦、吉尔吉斯、俄罗斯和塔吉克斯坦支持该体系。 在双边基础上合作，格鲁吉亚和土库曼最近7年不参加独联体国家对空防御。 </str>

<str name="id">E3798D82-EAB6-2BEA-D7E2-79FBD102E845</str>

<long name="_version_">1436361868021071872</long></doc>

...

</result>

</response>
```

上面所看到的就是用xml格式返回的查询结果,其中的doc就是一个文档,在doc里面的那个就是我们开始在schema.xml中定义的字段。

如果使用SolrJ进行调用的话代码如下:

```
SolrQuery query = new SolrQuery();  
query.set("q", "*.*");  
QueryResponse rsp =server.query(query)  
  
SolrDocumentList list = rsp.getResults();
```

返回结果在SolrDocumentList中在这个对象中遍历取出值来:

```
for (int i = 0; i < list.size(); i++) {  
    SolrDocument sd = list.get(i);  
  
    String id = (String) sd.getFieldValue("id");  
  
    System.out.println(id);  
  
}
```

6.2.1 查询参数

名称	描述
q	查询字符串，必须的。
fq	filter query。使用Filter Query可以充分利用Filter Query Cache，提高检索性能。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。
fl	field list。指定返回结果字段。以空格" "或逗号","分隔。
start	用于分页定义结果起始记录数，默认为0。
rows	用于分页定义结果每页返回记录数，默认为10。
sort	排序，格式:sort=<field name>+<desc asc>[,<field name>+<desc asc>]...。示例：（inStock desc, price asc）表示先“inStock”降序,再“price”升序，默认是相关性降序。

df	默认的查询字段，一般默认指定。
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。必须大写
wt	writer type。指定查询输出结构格式，默认为“xml”。在solrconfig.xml中定义了查询输出格式：xml、json、 Python 、ruby、 PHP 、phps、custom。
qt	query type，指定查询使用的Query Handler，默认为“standard”。
explainOther	设置当debugQuery=true时，显示其他的查询说明。
defType	设置查询解析器名称。
timeAllowed	设置查询超时时间。
omitHeader	设置是否忽略查询结果返回头信息，默认为“false”。
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。
debugQuery	设置返回结果是否显示Debug信息。

6.2.2 查询语法

1. 匹配所有文档：*:*

2. 强制、阻止和可选查询：

1) Mandatory: 查询结果中必须包括的(for example, only entry name containing the word make)

Solr/Lucene Statement: +make, +make +up ,+make +up +kiss

2) prohibited: (for example, all documents except those with word believe)

Solr/Lucene Statement: +make +up -kiss

3) optional:

Solr/Lucene Statement: +make +up kiss

3. 布尔操作：AND、OR和NOT布尔操作（必须大写）与Mandatory、optional和prohibited相似。

1) make AND up = +make +up :AND左右两边的操作都是mandatory

2) make || up = make OR up=make up :OR左右两边的操作都是optional

3) +make +up NOT kiss = +make +up -kiss

4) make AND up OR french AND Kiss不可以达到期望的结果，因为AND两边的操作都是mandatory的。

4. 子表达式查询（子查询）：可以使用“()”构造子查询。

示例：(make AND up) OR (french AND Kiss)

5. 子表达式查询中阻止查询的限制：

示例：make (-up):只能取得make的查询结果；要使用make (-up *:*)查询make或者不包括up的结果。

6. 多字段fields查询：通过字段名加上分号的方式（fieldName:query）来进行查询

示例：entryNm:make AND entryId:3cdc86e8e0fb4da8ab17caed42f6760c

7. 通配符查询（wildCard Query）：

1) 通配符？和*：“*”表示匹配任意字符；“？”表示匹配出现的位置。

示例：`ma?*`（`ma`后面的一个位置匹配），`ma??*`（`ma`后面两个位置都匹配）

2) 查询字符必须要小写：`+Ma +be**`可以搜索到结果；`+Ma +Be**`没有搜索结果。

3) 查询速度较慢，尤其是通配符在首位：主要原因一是需要迭代查询字段中的每个term，判断是否匹配；二是匹配上的term被加到内部的查询，当term失败。

4) Solr中默认通配符不能出现在首位（可以修改QueryParser，设置

`setAllowLeadingWildcard`为true）

5) set `setAllowLeadingWildcard` to true.

8.模糊查询、相似查询：不是精确的查询，通过对查询的字段进行重新插入、删除和转换来取得得分较高的查询解决（由Levenstein Distance Algorithm算

1) 一般模糊查询：示例：`make-believ~`

2) 门槛模糊查询：对模糊查询可以设置查询门槛，门槛是0~1之间的数值，门槛越高表面相似度越高。示例：`make-believ~0.5`、`make-believ~0.8`、`ma`

9.范围查询（Range Query）：Lucene支持对数字、日期甚至文本的范围查询。结束的范围可以使用“*”通配符。

示例：

1) 日期范围（ISO-8601 时间GMT）：`sa_type:2 AND a_begin_date:[1990-01-01T00:00:00.000Z TO 1999-12-31T24:59:99.999Z]`

2) 数字：`salary:[2000 TO *]`

3) 文本：`entryNm:[a TO a]`

10.日期匹配：YEAR, MONTH, DAY, DATE (synonymous with DAY) HOUR, MINUTE, SECOND, MILLISECOND, and MILLI (synonymous with MILLISE

示例：

1) `r_event_date:[* TO NOW-2YEAR]`：2年前的现在这个时间

2) `r_event_date:[* TO NOW/DAY-2YEAR]`：2年前前一天的这个时间

6.2.3函数查询（Function Query）

函数查询 可以利用 numeric字段的值 或者 与字段相关的的某个特定的值的函数，来对文档进行评分。

1. 使用函数查询的方法

这里主要有三种方法可以使用函数查询，这三种方法都是通过solr http接口的。

1) 使用FunctionQParserPlugin。ie: `q={!func}log(foo)`

2) 使用“_val_”内嵌方法

内嵌在正常的solr查询表达式中。即，将函数查询写在 q这个参数中，这时候，我们使用“_val_”将函数与其他的查询加以区别。

ie: `entryNm:make && _val_:ord(entryNm)`

3) 使用dismax中的bf参数

使用明确为函数查询的参数，比如说dismax中的bf（boost function）这个参数。 注意：bf这个参数是可以接受多个函数查询的，它们之间用空格隔开，它使用bf这个参数的时候，我们必须保证单个函数中是没有空格出现的，不然程序有可能会以为是两个函数。

示例：

`q=dismax&bf="ord(popularity)^0.5 recip(rord(price),1,1000,1000)^0.3`

2. 函数的格式（Function Query Syntax）

目前，function query 并不支持 `a+b` 这样的形式，我们得把它写成一个方法形式，这就是 `sum(a,b)`。

3. 使用函数查询注意事项

1) 用于函数查询的field必须是被索引的；

2) 字段不可以是多值的（multi-value）

4. 可以利用的函数（available function）

1) constant：支持有小数点的常量； 例如：1.5； SolrQuerySyntax:_val_:1.5

2) fieldvalue：这个函数将会返回numeric field的值，这个字段必须是indexd的，非multiValued的。格式很简单，就是该字段的名字。如果这个字段中没有

3) **ord**: 对于一个字段，它所有的值都会按照字典顺序排列，这个函数返回你要查询的那个特定的值在这个顺序中的排名。这个字段，必须是非multiValued。例如：某个特定的字段只能去三个值，“apple”、“banana”、“pear”，那么ord (“apple”) =1, ord (“banana”) =2, ord (“pear”) =3.需要注意的是索引中的位置，所以当有文档被删除、或者添加的时候，ord () 的值就会发生变化。当你使用MultiSearcher的时候，这个值也就是不定的了。

4) **rord**: 这个函数将会返回与ord相对应的倒排序的排名。

格式: rord(myIndexedField)。

5) **sum**: 这个函数的意思就显而易见啦，它就是表示“和”啦。

格式: sum(x,1)、sum(x,y)、sum(sqrt(x),log(y),z,0.5)

6) **product**: product(x,y,...)将会返回多个函数的乘积。格式: product(x,2)、product(x,y)

7) **div**: div(x,y)表示x除以y的值，格式: div (1,x)、div(sum(x,100),max(y,1))

8) **pow**: pow表示幂值。pow(x,y) =x^y。例如: pow(x,0.5) 表示开方pow(x,log(y))

9) **abs**: abs(x)将返回表达式的绝对值。格式: abs(-5)、abs(x)

10) **log**: log(x)将会返回基数为10，x的对数。格式: log(x)、log(sum(x,100))

11) **Sqrt**: sqrt(x) 返回 一个数的平方根。格式: sqrt (2)、sqrt(sum(x,100))

12) **Map**: 如果 x>=min,且x<=max,那么map(x,min,max,target)=target.如果 x不在[min,max]这个区间内，那么map(x,min,max,target)=x.

格式: map(x,0,0,1)

13) **Scale**: scale(x,minTarget,maxTarget) 这个函数将会把x的值限制在[minTarget,maxTarget]范围内。

14) **query**: query(subquery,default)将会返回给定subquery的分数，如果subquery与文档不匹配，那么将会返回默认值。任何的查询类型都是受支持的。接指定查询串。

例子: q=product(popularity, query(!dismax v='solr rocks')) 将会返回popularity和通过dismax 查询得到的分数的乘积。

q=product(popularity, query(\$qq)&qq={!dismax}solr rocks 跟上一个例子的效果是一样的。不过这里使用的是引用的方式

q=product(popularity, query(\$qq,0.1)&qq={!dismax}solr rocks 在前一个例子的基础上又加了一个默认值。

15) **linear**: linear(x,m,c)表示 m*x+c,其中m和c都是常量，x是一个变量也可以是一个函数。例如: linear(x,2,4)=2*x+4.

16) **Recip**: recip(x,m,a,b)=a/(m*x+b)其中，m、a、b是常量，x是变量或者一个函数。当a=b，并且x>=0的时候，这个函数的最大值是1，值的大小随着x的增大而减小。格式: recip(rord(creationDate),1,1000,1000)

17) **Max**: max(x,c)将会返回一个函数和一个常量之间的最大值。

例如: max(myfield,0)

6.3高亮显示

我们经常使用搜索引擎，比如在baidu 搜索 java，会出现如下结果，结果中与关键字匹配的地方是红色显示与其他内容区别开来。

solr 默认已经配置了highlight 组件(详见 SOLR_HOME/conf/solrconfig.xml)。通常我只需要这样请求http://localhost:8983/solr/ collection1 /select?q=%E4%B8%AD%E5%9B%BD&start=0&rows=1&fl=content+path+&wt=xml&indent=true&hl=true&hl.fl=content

可以看到与比一般的请求多了两个参数 "hl=true" 和 "hl.fl= content "。

"hl=true" 是开启高亮，"hl.fl= content " 是告诉solr 对 name 字段进行高亮(如果你想对多个字段进行高亮，可以继续添加字段，字段间用逗号隔开，如 "hl.fl=content, path")。默认将会被 "" 和 "" 包围。还可以使用hl.simple.pre" 和 "hl.simple.post"参数设置前后标签。

查询结果如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">2</int>
    <lst name="params">
      <str name="fl">content path</str>
```

```
<str name="indent">true</str>

<str name="start">0</str>

<str name="q">中国</str>

<str name="hl.simple.pre"><em></str>

<str name="hl.simple.post"></em></str>

<str name="hl.fl">content</str>

<str name="wt">xml</str>

<str name="hl">true</str>

<str name="rows">1</str>

</lst>

</lst>

<result name="response" numFound="6799" start="0">

  <doc>

    <str name="path">E:\Reduced\IT\630.txt</str>

    <str name="content">    本报讯 中国银联股份有限公司和中国电信集团日前在北京签署全面战略合作协议。这标志着中国银联和中国电信将在通信
    品合作开发等领域建立全面合作伙伴关系。    据悉，双方签署的全面战略合作协议主要内容是：中国银联将选择中国电信作为通信信息服务的主要提供商
    信的水平和服务网络的服务水平开展全面、深入的合作；中国电信选择中国银联作为银行卡转接支付服务的主要提供商，并围绕开发、推广新型支付终端产
    （辛华） </str></doc>

  </result>

  <lst name="highlighting">

    <lst name="7D919C61-03B3-4B6F-2D10-9E3CC92D2852">

      <arr name="content">

        <str>    本报讯 <em>中国</em>银联股份有限公司和<em>中国</em>电信集团日前在北京签署全面战略合作协议。这标志着<em>中国</em>银联和<em>er
        信息增值服务、新型支付产品合作开发等领域建立全面合作伙伴关系。    据悉，双方签署</str>

      </arr>

    </lst>

  </lst>

</response>
```

使用SolrJ方法基本一样也是设置这些个参数,只不过是SolrJ封装起来了,代码如下:

```
SolrQuery query = new SolrQuery();

query.set("q","*.*");

query.setHighlight(true); // 开启高亮组件

query.addHighlightField("content");// 高亮字段

query.setHighlightSimplePre(PRE_TAG);// 标记

query.setHighlightSimplePost(POST_TAG);

QueryResponse rsp =server.query(query)

//...上面取结果的代码

//取出高亮结果

if (rsp.getHighlighting() != null) {

    if (rsp.getHighlighting().get(id) != null) {//先通过结果中的ID到高亮集合中取出文档高亮信息
```

```

Map<String, List<String>> map = rsp.getHighlighting().get(id);//取出高亮片段

if (map.get(name) != null) {

    for (String s : map.get(name)) {

        System.out.println(s);

    }

}

}

```

6.4拼写检查

首先配置 **solrconfig.xml**，文件可能已经有这两个元素(如果没有添加即可)，需要根据我们自己的系统环境做些适当的修改。

```

<searchComponent name="spellcheck" class="solr.SpellCheckComponent">

    <str name="queryAnalyzerFieldType">text_spell</str>

    <lst name="spellchecker">

        <str name="name">direct</str>

        <str name="field">spell</str>

        <str name="classname">solr.DirectSolrSpellChecker</str>

        <str name="distanceMeasure">internal</str>

        <float name="accuracy">0.5</float>

        <int name="maxEdits">2</int>

        <int name="minPrefix">1</int>

        <int name="maxInspections">5</int>

        <int name="minQueryLength">2</int>

        <float name="maxQueryFrequency">0.01</float>

    </lst>

</searchComponent>


<requestHandler name="/spell" class="solr.SearchHandler" startup="lazy">

    <lst name="defaults">

        <str name="spellcheck.dictionary">direct</str>

        <str name="spellcheck">on</str>

        <str name="spellcheck.collate">true</str>

        <str name="spellcheck.collateExtendedResults">true</str>

    </lst>

    <arr name="last-components">

        <str>spellcheck</str>

    </arr>

</requestHandler>

```

配置完成之后，我们进行一下测试,重启Solr后，访问如下链接

<http://localhost:8983/solr/collection1/spell?wt=xml&indent=true&spellcheck=true&spellcheck.q=%E4%B8%AD%E5%9B%BD>

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

<response>

  <lst name="responseHeader">

    <int name="status">0</int>

    <int name="QTime">0</int>

  </lst>

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

  <lst name="spellcheck">

    <lst name="suggestions">

      <lst name="beijink">

        <int name="numFound">1</int>

        <int name="startOffset">0</int>

        <int name="endOffset">3</int>

        <arr name="suggestion">

          <str>beijing</str>

        </arr>

      </lst>

    </lst>

  </lst>

</response>
```

使用SolrJ时也同样加入参数就可以

```
SolrQuery query = new SolrQuery();

query.set("q", ".*");

query.set("qt", "/spell");

QueryResponse rsp =server.query(query)

//...上面取结果的代码

SpellCheckResponse spellCheckResponse = rsp.getSpellCheckResponse();

if (spellCheckResponse != null) {

  String collation = spellCheckResponse.getCollatedResult();

}
```

6.5 检索建议

检索建议目前是各大搜索的标配应用，主要作用是避免用户输入错误的搜索词，同时将用户引导到相应的关键词搜索上。Solr内置了检索建议功能，它可选择基于提示词文本做检索建议，还支持通过针对索引的某个字段建立索引词库做检索建议。在诸多文档中都推荐使用基于索引来做检索建议，因此我们

现在我们来开始配置Suggest模块,首先在solrconfig.xml文件中配置Suggest依赖的SpellChecker模块，然后再配置Suggest模块,所以这两个都需要配置。

```
<searchComponent name="suggest" class="solr.SpellCheckComponent">

  <str name="queryAnalyzerFieldType">string</str>

  <lst name="spellchecker">

    <str name="name">suggest</str>
```



```

<str name="classname">org.apache.solr.spelling.suggest.Suggester</str>

<str name="lookupImpl">org.apache.solr.spelling.suggest.tst.TSTLookup</str>

<str name="field">text</str>

<float name="threshold">0.0001</float>

<str name="spellcheckIndexDir">spellchecker</str>

<str name="comparatorClass">freq</str>

<str name="buildOnOptimize">true</str>

<!--<str name="buildOnCommit">true</str-->

</lst>

</searchComponent>

<requestHandler name="/suggest" class="solr.SearchHandler" startup="lazy">

  <lst name="defaults">

    <str name="spellcheck">true</str>

    <str name="spellcheck.dictionary">suggest</str>

    <str name="spellcheck.onlyMorePopular">true</str>

    <str name="spellcheck.extendedResults">>false</str>

    <str name="spellcheck.count">10</str>

    <str name="spellcheck.collate">true</str>

  </lst>

  <arr name="components">

    <str>suggest</str>

  </arr>

</requestHandler>

```

配置完成之后, 我们进行一下测试, 重启Solr后, 访问如下链接

<http://localhost:8983/solr/ collection1/suggest?wt=xml&indent=true&spellcheck=true&spellcheck.q=%E4%B8%AD%E5%9B%BD>

```

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

<response>

<lst name="responseHeader">

  <int name="status">0</int>

  <int name="QTime">4</int>

</lst>

<lst name="spellcheck">

  <lst name="suggestions">

    <lst name="中国">

      <int name="numFound">4</int>

      <int name="startOffset">0</int>

      <int name="endOffset">2</int>

      <arr name="suggestion">

```

```
<str>中国队</str>

<str>中国证监会</str>

<str>中国足协</str>

<str>中国银行</str>

</arr>

</lst>

</lst>

</lst>

</response>
```

使用SolrJ时也同样加入参数就可以

```
SolrQuery query = new SolrQuery();

query.set("q", token);

query.set("qt", "/suggest");

query.set("spellcheck.count", "10");

QueryResponse response = server.query(query);

SpellCheckResponse spellCheckResponse = response.getSpellCheckResponse();

if (spellCheckResponse != null) {

    List<SpellCheckResponse.Suggestion> suggestionList = spellCheckResponse.getSuggestions();

    for (SpellCheckResponse.Suggestion suggestion : suggestionList) {

        List<String> suggestedWordList = suggestion.getAlternatives();

        for (int i = 0; i < suggestedWordList.size(); i++) {

            String word = suggestedWordList.get(i);

        }

    }

    return results;

}
```

通过threshold参数来限制一些不常用的词不出现在智能提示列表中，当这个值设置过大时，可能导致结果太少，需要引起注意。目前主要存在的问题是完全基于索引中字符的出现次数，没有兼顾用户搜索词语的频率，因此无法将一些热门词排在更靠前的位置。这块可定制SuggestWordScoreComparator类。

6.6 分组统计

我这里实现分组统计的方法是使用了Solr的Facet组件，Facet组件是Solr默认集成的一个组件。

6.6.1 Facet简介

Facet是solr的高级搜索功能之一，可以给用户提供更友好的搜索体验。在搜索关键字的同时，能够按照Facet的字段进行分组并统计。

6.6.2 Facet字段

1. 适宜被Facet的字段

一般代表了实体的某种公共属性，如商品的分类，商品的制造厂家，书籍的出版商等等。

2. Facet字段的要求

Facet的字段必须被索引。一般来说该字段无需分词，无需存储。

无需分词是因为该字段的值代表了一个整体概念，如电脑的品牌“联想”代表了一个整体概念，如果拆成“联”，“想”两个字都不具有实际意义。另外该字段的

理,保持其原貌即可。

无需存储是因为一般而言用户所关心的并不是该字段的具体值,而是作为对查询结果进行分组的一种手段,用户一般会沿着这个分组进一步深入搜索。

3. 特殊情况

对于一般查询而言,分词和存储都是必要的.比如CPU类型"Intel 酷睿2双核 P7570",拆分成"Intel","酷睿","P7570"这样一些关键字并分别索引,可能提供更为Facet字段,最好不进行分词.这样就造成了矛盾,解决方法为,将CPU字段设置为不分词不存储,然后建立另外一个字段为它的COPY,对这个COPY的

```
<types>

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

  <fieldType name="tokened" class="solr.TextField" >

    <analyzer>

      .....

    </analyzer>

  </fieldType>

</types>

<fields>

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

  <field name="cpuCopy" type=" tokened" indexed="true" stored="true"/>

</fields>

<copyField source="cpu" dest="cpuCopy"/>
```

6.6.2 Facet组件

Solr的默认requestHandler已经包含了Facet组件(solr.FacetComponent).如果自定义requestHandler或者对默认的requestHandler自定义组件列表,那:

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

.....

<arr name="components">

<str>自定义组件名</str>

<str>facet</str>

.....

</arr>

</requestHandler>
```

6.6.2 Facet查询

进行Facet查询需要在请求参数中加入facet=on或者facet=true只有这样Facet组件才起作用.

1. Field Facet

Facet字段通过在请求中加入facet.field参数加以声明,如果需要多个字段进行Facet查询,那么将该参数声明多次.例如:

```
http://localhost:8983/solr/ collection1/select?q=%3A*&start=0&rows=1&wt=xml&indent=true&facet=true&facet.field=category_s&facet.field=modified_l
```

返回结果:

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

<response>

<lst name="responseHeader">
```

```
<int name="status">0</int>

<int name="QTime">1</int>

<lst name="params">

  <str name="facet">true</str>

  <str name="indent">true</str>

  <str name="start">0</str>

  <str name="q">*:*</str>

<arr name="facet.field">

  <str>category_s</str>

  <str>modified_l</str>

</arr>

  <str name="wt">xml</str>

  <str name="rows">0</str>

</lst>

</lst>

<result name="response" numFound="17971" start="0">

</result>

<lst name="facet_counts">

  <lst name="facet_queries"/>

  <lst name="facet_fields">

    <lst name="category_s">

      <int name="0">5991</int>

      <int name="1">5990</int>

      <int name="2">5990</int>

    </lst>

    <lst name="modified_l">

      <int name="1162438554000">951</int>

      <int name="1162438556000">917</int>

      <int name="1162438548000">902</int>

      <int name="1162438546000">674</int>

    </lst>

  </lst>

  <lst name="facet_dates"/>

  <lst name="facet_ranges"/>

</lst>

</response>
```

各个Facet字段互不影响,且可以针对每个Facet字段设置查询参数.以下介绍的参数既可以应用于所有的Facet字段,也可以应用于每个单独的Facet字段.

f.字段名.参数名=参数值

这种方式调用.比如facet.prefix参数应用于cpu字段,可以采用如下形式

```
f.cpu.facet.prefix=Intel
```

1.1 facet.prefix

表示Facet字段值的前缀.比如facet.field=cpu&facet.prefix=Intel,那么对cpu字段进行Facet查询,返回的cpu都是以Intel开头的, AMD开头的cpu型号将不

1.2 facet.sort

表示Facet字段值以哪种顺序返回.可接受的值为true(count)|false(index,lex). true(count)表示按照count值从大到小排列. false(index,lex)表示按照字段排列.默认情况下为true(count).当facet.limit值为负数时,默认facet.sort= false(index,lex).

1.3 facet.limit

限制Facet字段返回的结果条数.默认值为100.如果此值为负数,表示不限制.

1.4 facet.offset

返回结果集的偏移量,默认为0.它与facet.limit配合使用可以达到分页的效果.

1.5 facet.mincount

限制了Facet字段值的最小count,默认为0.合理设置该参数可以将用户的关注点集中在少数比较热门的领域.

1.6 facet.missing

默认为"",如果设置为true或者on,那么将统计那些该Facet字段值为null的记录.

1.7 facet.method

取值为enum或fc,默认为fc.该字段表示了两种Facet的算法,与执行效率相关.

enum适用于字段值比较少情况,比如字段类型为布尔型,或者字段表示中国的所有省份.Solr会遍历该字段的所有取值,并从filterCache里为每个值分配一个filterCache的设置足够大).然后计算每个filter与主查询的交集.

fc(表示Field Cache)适用于字段取值比较多,但在每个文档里出现次数比较少情况.Solr会遍历所有的文档,在每个文档内搜索Cache内的值,如果找到就将C

1.8 facet.enum.cache.minDf

当facet.method=enum时,此参数其作用,minDf表示minimum document frequency.也就是文档内出现某个关键字的最少次数.该参数默认值为0.设置该参数会增加总的查询时间(计算交集的时间增加了).如果设置该值的话,官方文档建议优先尝试25-50内的值.

6.6.3 Date Facet

日期类型的字段在文档中很常见,如商品上市时间,货物出仓时间,书籍上架时间等等.某些情况下需要针对这些字段进行Facet.不过时间字段的取值有无限点而是某个时间段内的查询统计结果. Solr为日期字段提供了更为方便的查询统计方式.当然,字段的类型必须是DateField(或其子类型).

需要注意的是,使用Date Facet时,字段名,起始时间,结束时间,时间间隔这4个参数都必须提供.与Field Facet类似,Date Facet也可以对多个字段进行Facet.并数.

facet.date:该参数表示需要进行Date Facet的字段名,与facet.field一样,该参数可以被设置多次,表示对多个字段进行Date Facet.

facet.date.start:起始时间,时间的一般格式为1995-12-31T23:59:59Z,另外可以使用NOWYEAR\ MONTH等等,具体格式可以参考DateField的Java doc.

facet.date.end:结束时间.

facet.date.gap:时间间隔.如果start为2009-1-1,end为2010-1-1.gap设置为+1MONTH表示间隔1个月,那么将会把这段时间划分为12个间隔段.

注意+因为特殊字符所以应该用%2B代替.

facet.date.hardend:取值可以为true|false,默认为false.它表示gap迭代到end处采用何种处理.举例说明start为2009-1-1,end为2009-12-25,gap为+1MONTH

hardend为false的话最后一个时间段为2009-12-1至2010-1-1;

hardend为true的话最后一个时间段为2009-12-1至2009-12-25.

facet.date.other:取值范围为before|after|between|none|all,默认为none.before会对start之前的值做统计.after会对end之后的值做统计.between会对start至hardend为true的话,那么该值就是各个时间段统计值的和.none表示该项禁用.all表示before,after,all都会统计.

举例:

```
&facet=on
```

```
&facet.date=date
```

```
&facet.date.start=2009-1-1T0:0:0Z
```

```
&facet.date.end=2010-1-1T0:0:0Z
```

```
&facet.date.gap=%2B1MONTH
```

```
&facet.date.other=all
```

返回结果:

```
<lst name="facet_counts">
  <lst name="facet_queries"/>
  <lst name="facet_fields"/>
  <lst name="facet_dates">
    <int name="2009-01-01T00:00:00Z">5</int>
    <int name="2009-02-01T00:00:00Z">7</int>
    <int name="2009-03-01T00:00:00Z">4</int>
    <int name="2009-04-01T00:00:00Z">3</int>
    <int name="2009-05-01T00:00:00Z">7</int>
    <int name="2009-06-01T00:00:00Z">3</int>
    <int name="2009-07-01T00:00:00Z">6</int>
    <int name="2009-08-01T00:00:00Z">7</int>
    <int name="2009-09-01T00:00:00Z">2</int>
    <int name="2009-10-01T00:00:00Z">4</int>
    <int name="2009-11-01T00:00:00Z">1</int>
    <int name="2009-12-01T00:00:00Z">5</int>
    <str name="gap">+1MONTH</str>
    <date name="end">2010-01-01T00:00:00Z</date>
    <int name="before">180</int>
    <int name="after">5</int>
    <int name="between">54</int>
  </lst>
</lst>
```

6.6.4 Facet Query

Facet Query利用类似于filter query的语法提供了更为灵活的Facet.通过facet.query参数,可以对任意字段进行筛选.

例1:

```
&facet=on
```

```
&facet.query=date:[2009-1-1T0:0:0Z TO 2009-2-1T0:0:0Z]
```

```
&facet.query=date:[2009-4-1T0:0:0Z TO 2009-5-1T0:0:0Z]
```

返回结果:

```
<lst name="facet_counts">
```

```
<lst name="facet_queries">

  <int name="date:[2009-1-1T0:0:0Z TO 2009-2-1T0:0:0Z]">5</int>

<int name="date:[2009-4-1T0:0:0Z TO 2009-5-1T0:0:0Z]">3</int>

</lst>

<lst name="facet_fields"/>

<lst name="facet_dates"/>

</lst>
```

例2:

```
&facet=on

&facet.query=date:[2009-1-1T0:0:0Z TO 2009-2-1T0:0:0Z]

&facet.query=price:[* TO 5000]
```

返回结果:

```
<lst name="facet_counts">

  <lst name="facet_queries">

    <int name="date:[2009-1-1T0:0:0Z TO 2009-2-1T0:0:0Z]">5</int>

  <int name="price:[* TO 5000]">116</int>

</lst>

<lst name="facet_fields"/>

<lst name="facet_dates"/>

</lst>
```

例3:

```
&facet=on

&facet.query=cpu:[A TO G]
```

返回结果:

```
<lst name="facet_counts">

  <lst name="facet_queries">

    <int name="cpu:[A TO G]">11</int>

  </lst>

  <lst name="facet_fields"/>

  <lst name="facet_dates"/>

</lst>
```

6.6.5 key操作符

可以用key操作符为Facet字段取一个别名.

例:

```
&facet=on

&facet.field={!key=中央处理器}cpu
```

```
&facet.field={!key=显卡}videoCard
```

返回结果:

```
<lst name="facet_counts">

  <lst name="facet_queries"/>

  <lst name="facet_fields">

    <lst name="中央处理器">

      <int name="Intel 酷睿2双核 T6600">48</int>

      <int name="Intel 奔腾双核 T4300">28</int>

    <int name="Intel 酷睿2双核 P8700">18</int>

    <int name="Intel 酷睿2双核 T6570">11</int>

    <int name="Intel 酷睿2双核 T6670">11</int>

    <int name="Intel 奔腾双核 T4400">9</int>

    <int name="Intel 酷睿2双核 P7450">9</int>

    <int name="Intel 酷睿2双核 T5870">8</int>

    <int name="Intel 赛扬双核 T3000">7</int>

    <int name="Intel 奔腾双核 SU4100">6</int>

    <int name="Intel 酷睿2双核 P8400">6</int>

    <int name="Intel 酷睿2双核 SU7300">5</int>

    <int name="Intel 酷睿 i3 330M">4</int>

  </lst>

  <lst name="显卡">

    <int name="ATI Mobility Radeon HD 4">63</int>

    <int name="NVIDIA GeForce G 105M">24</int>

    <int name="NVIDIA GeForce GT 240M">21</int>

    <int name="NVIDIA GeForce G 103M">8</int>

    <int name="NVIDIA GeForce GT 220M">8</int>

    <int name="NVIDIA GeForce 9400M G">7</int>

    <int name="NVIDIA GeForce G 210M">6</int>

  </lst>

</lst>

  <lst name="facet_dates"/>

</lst>
```

6.6.6 tag操作符和ex操作符

当查询使用filter query的时候,如果filter query的字段正好是Facet字段,那么查询结果往往被限制在某一个值内.

例:

```
&fq=screenSize:14
```

```
&facet=on
```

```
&facet.field=screenSize
```


返回结果:

```
<lst name="facet_counts">

  <lst name="facet_queries"/>

  <lst name="facet_fields">

    <lst name=" screenSize">

      <int name="14.0">107</int>

    <int name="10.2">0</int>

    <int name="11.1">0</int>

    <int name="11.6">0</int>

    <int name="12.1">0</int>

    <int name="13.1">0</int>

    <int name="13.3">0</int>

    <int name="14.1">0</int>

    <int name="15.4">0</int>

    <int name="15.5">0</int>

    <int name="15.6">0</int>

    <int name="16.0">0</int>

    <int name="17.0">0</int>

    <int name="17.3">0</int>

  </lst>

</lst>

  <lst name="facet_dates"/>

</lst>
```

可以看到,屏幕尺寸(screenSize)为14寸的产品共有107件,其它尺寸的产品的数目都是0,这是因为在filter里已经限制了screenSize:14.这样,查询结果中,除其它项目没有实际的意义.有些时候,用户希望把结果限制在某一范围内,又希望查看该范围外的概况.比如上述情况,既要把查询结果限制在14寸屏的笔记本,又有多少产品.这个时候需要用到tag和ex操作符.tag就是把一个filter标记起来,ex(exclude)是在Facet的时候把标记过的filter排除在外.

例:

```
&fq={!tag=aa}screenSize:14

&facet=on

&facet.field={!ex=aa}screenSize
```

返回结果:

```
<lst name="facet_counts">

  <lst name="facet_queries"/>

  <lst name="facet_fields">

    <lst name=" screenSize">

      <int name="14.0">107</int>

    <int name="14.1">40</int>

    <int name="13.3">34</int>
```

```

<int name="15.6">22</int>

<int name="15.4">8</int>

<int name="11.6">6</int>

<int name="12.1">5</int>

<int name="16.0">5</int>

<int name="15.5">3</int>

<int name="17.0">3</int>

<int name="17.3">3</int>

<int name="10.2">1</int>

<int name="11.1">1</int>

<int name="13.1">1</int>

</lst>

    </lst>

    <lst name="facet_dates"/>

</lst>

```

这样其它屏幕尺寸的统计信息就有意义了。

6.6.7 SolrJ对Facet的支持

//初始化查询对象

```
String q = "**.*";
```

```
SolrQuery query = new SolrQuery(q);
```

```
query.setIncludeScore(false);//是否按每组数量高低排序
```

```
query.setFacet(true);//是否分组查询
```

```
query.setRows(0);//设置返回结果条数，如果你时分组查询，你就设置为0
```

```
query.addFacetField("modified_l");//增加分组字段 q
```

```
query.addFacetQuery ("category_s[0 TO 1]");
```

```
QueryResponse rsp = server.query(query);
```

```
...
```

//取出结果

```
List<FacetField.Count> list = rsp.getFacetField("modified_l").getValues();
```

```
Map<String, Integer> list = rsp.getFacetQuery();
```

6.7 自动聚类

Solr 使用Carrot2完成了聚类功能,能够把检索到的内容自动分类, Carrot2聚类示例:

要想Solr支持聚类功能,首选要把Solr发行包的中的dist/ solr-clustering-4.2.0.jar, 复制到solr\contrib\analysis-extras\lib下.然后打开solrconfig.xml进行配置

```

<searchComponent name="clustering"

    enable="${solr.clustering.enabled:true}"

    class="solr.clustering.ClusteringComponent" >

<lst name="engine">

<str name="name">default</str>

```

```
<str name="carrot.algorithm">org.carrot2.clustering.  
lingo.LingoClusteringAlgorithm</str>  
  
<str name="LingoClusteringAlgorithm.desiredClusterCountBase">20</str>  
  
</lst>  
  
</searchComponent>
```

配好了聚类组件后,下面配置requestHandler:

```
<requestHandler name="/clustering" startup="lazy" enable="${solr.clustering.enabled:true}" class="solr.SearchHandler">  
  
  <lst name="defaults">  
  
    <str name="echoParams">explicit</str>  
  
    <bool name="clustering">true</bool>  
  
    <str name="clustering.engine">default</str>  
  
    <bool name="clustering.results">true</bool>  
  
    <str name="carrot.title">category_s</str>  
  
    <str name="carrot.snippet">content</str>  </lst>  
  
    <arr name="last-components">  
  
      <str>clustering</str>  </arr>  
  
</requestHandler>
```

有两个参数要注意carrot.title, carrot.snippet是聚类的比较计算字段,这两个参数必须是stored="true".carrot.title的权重要高于carrot.snippet,如果只有一以去掉(是去掉不是值为空).设完了用下面的URL就可以查询了

http://localhost:8983/skyCore/clustering?q=%3A*&wt=xml&indent=true

6.8相似匹配

在我们使用网页搜索时，会注意到每一个结果都包含一个“相似页面”链接，单击该链接，就会发布另一个搜索请求，查找出与起初结果类似的文档。Solr的MoreLikeThisComponent（MLT）和 MoreLikeThisHandler 实现了一样的功能。如上所述，MLT 是与标准 SolrRequestHandler 集成在一起的；MoreLikeThisHandler 起，并添加了一些其他选项，但它要求发布一个单一的请求。我将着重讲述 MLT，因为使用它的可能性更大一些。幸运的是，不需要任何设置就可以查询相似文档。

MLT 要求字段被储存或使用检索词向量，检索词向量以一种以文档为中心的方式储存信息。MLT 通过文档的内容来计算文档中关键词语，然后使用原新的查询。提交新查询就会返回其他查询结果。所有这些都可以用检索词向量来完成：只需将 termVectors="true" 添加到 schema.xml 中的 <field> 声明。

MoreLikeThisComponent 参数：

参数	说明	值域
mlt	在查询时，打开/关闭 MoreLikeThisComponent 的布尔值。	true false
mlt.count	可选。每一个结果要检索的相似文档数。	> 0
mlt.fl	用于创建 MLT 查询的字段。	任何被储存的或含有检索词向量的字段。
mlt.maxqt	可选。查询词语的最大数量。由于长文档可能会有很多关键词语，这样 MLT 查询可能会很大，从而导致反应缓慢或可怕的 TooManyClausesException，该参数只保留关键的词语。	> 0

要想使用匹配相似首先在 solrconfig.xml 中配置 MoreLikeThisHandler

```
<requestHandler name="/mlt" class="solr.MoreLikeThisHandler">
</requestHandler>
```

然后我可以请求

<http://localhost:8983/skyCore/mlt?q=id%3A6F398CCD-2DE0-D3B1-9DD6-D4E532FFC531&mlt.true&mlt.fl=content&wt=xml&indent=true>

上面请求的意思查找 id 为 6F398CCD-2DE0-D3B1-9DD6-D4E532FFC531 的 document ,然后返回与此 document 在 name 字段上相似的其他 document
termVector=true 才有效果

```
<field name="content" type="text_smartcn" indexed="false" stored="true" multiValued="false" termVector="true"/>
```

使用SolrJ时也同样加入参数就可以

```
SolrQuery query = new SolrQuery();

query.set("qt", "/mlt");

query.set("mlt.fl","content");

query.set("fl", "id,");

query.set("q", "id: 6F398CCD-2DE0-D3B1-9DD6-D4E532FFC531");

query.setStart(0);

query.setRows(5);

QueryResponse rsp = server.query(query);

SolrDocumentList list = rsp.getResults();
```

6.9拼音检索

拼音检索中国人的专用检索,例如:中文内容为 中国 的输入zhongguo、zg、zhonggu 全拼、简拼、拼音的相邻的一部份都应该能检索出 中国 来。

想要实现拼音检索第一个就是拼音转换我这里用的是pinyin4j进行拼音转换。第二个就是N-Gram的题目，推敲到用户可能输入的既不是前缀也不是后缀巧，但不同于常用的N-Gram，我应用的从一边开端的单向的N-Gram，Solr里的实现叫EdgeNGramTokenFilter，但是分的分的太细了，不需要这么复杂Ex我们用的N-Gram不同于传统的N-Gram。

同样的例子使用EdgeNGramTokenFilter从前往后取2-Gram的结果是zh，一般是取min-max之间的所有gram，所以使用EdgeNGramTokenFilter取2-4zhong, zhongg, zhonggu, zhongguo, 从这个例子也不难理解为什么我要选择使用EdgeNGramTokenFilter而非一般意义上的N-Gram，考虑到用户可能输入照顾这些用户，我选择了从前往后和从后往前使用了两次EdgeNGramTokenFilter，这样不只是前缀、后缀，二十任意的字串都考虑进去了，所以大幅度的

现在思路明确了我们把它结合到Solr中，为了方便使用现在写了两个Filter进行处理拼音分词问题一个是拼音转换Filter（PinyinTransformTokenFilter）Filter(PinyinNGramTokenFilter),这样一来使用时就不用添加索引前做拼音的转换了。而且PinyinTransformTokenFilter还有个好处就是它只使用中文分词词都是有用的不重复的，不会对没用的停词类的做拼音转换和重复拼音转换，这样大大的提高了拼音转换速度。

想要Solr支持拼音检索就要先把拼音分词（PinyinAnalyzer）的jar复制到\solr\contrib\analysis-extras\lib下，然后在schema.xml中配置一个拼音字段类

```
<fieldType name="text_pinyin" class="solr.TextField" positionIncrementGap="0">

<analyzer type="index">

    <tokenizer class="org.apache.lucene.analysis.cn.smart.SmartChineseSentenceTokenizerFactory"/>

    <filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

    <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.cn.smart.SmartChineseSentenceTokenizerFactory"/>

    <filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

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

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

</analyzer>

</fieldType>
```

minTermLength: 最小中文词长度，意思是小于这个值的中文词不会做拼音转换。

minGram: 最小拼音切分长度。

如果想使用简拼的话在拼音转换Filter 使用这个参数**isFirstChar="true"**就可以了

在这个拼音类型中我们使用了**smartcn**的中言语分词器，如果想使用其它的自己换掉就行了。现在我们在原来索引中加入一个拼音字段，因为只做索引

```
<field name ="pinyin" type ="text_pinyin" indexed ="true" stored ="false" multiValued ="false"/>
```

加完后我们重新启动Solr测试一下看看

由于上面**minTermLength**和**minGram**设置的值，现在出现了人没有进行拼音转换并且最小拼音切分是从1个开始的。

到这里我们的配置还有没完成呢，还要加几个**copyFiled**，这样就不用单独处理我们新加的拼音字段了。方便呀~~~

```
<copyField source="content" dest="pinyin"/>
```

```
<copyField source="text" dest="spell"/>
```

到现在就可以使用拼音检索了。

拼音分词器**jar** 点击并复制就可以粘出去了。

6.10 SolrCloud

SolrCloud是基于**Solr**和**Zookeeper**的分布式搜索方案，是正在开发中的**Solr4.0**的核心组件之一，它的主要思想是使用**Zookeeper**作为集群的配置信息中心、置信息、自动容错、近实时搜索、查询时自动负载均衡。

基本可以用上面这幅图来概述，这是一个拥有**4**个**Solr**节点的集群，索引分布在两个**Shard**里面，每个**Shard**包含两个**Solr**节点，一个是**Leader**节点，一个是负责维护集群状态信息的**Overseer**节点，它是一个总控制器。集群的所有状态信息都放在**Zookeeper**集群中统一维护。从图中还可以看到，任何一个节点再将这个请求转发到文档所应该属于的那个**Shard**的**Leader**节点，**Leader**节点更新结束完成，最后将版本号 and 文档转发给同属于一个**Shard**的**replicas**节点。究明白后再单写一个文档。

附1: schema.xml

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

<schema name="example" version="1.5">

  <fields>

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

    <field name="path" type="text_ik" indexed="false" stored="true" multiValued="false" termVector="true" />

    <field name="content" type="text_ik" indexed="false" stored="true" multiValued="false" termVector="true"/>

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

    <field name ="pinyin" type ="text_pinyin" indexed ="true" stored ="false" multiValued ="false"/>

    <field name ="py" type ="text_py" indexed ="true" stored ="false" multiValued ="false"/>

    <field name="spell" type="text_spell" indexed="true" stored="false" multiValued="false" termVector="true"/>

    <field name="_version_" type="long" indexed="true" stored="true"/>

    <dynamicField name="*_i" type="int" indexed="true" stored="true"/>

    <dynamicField name="*_is" type="int" indexed="true" stored="true" multiValued="true"/>

    <dynamicField name="*_s" type="string" indexed="true" stored="true" />

    <dynamicField name="*_ss" type="string" indexed="true" stored="true" multiValued="true"/>

    <dynamicField name="*_l" type="long" indexed="true" stored="true"/>
```

```
<dynamicField name="*_ls" type="long" indexed="true" stored="true" multiValued="true"/>

<dynamicField name="*_t" type="text_general" indexed="true" stored="true"/>

<dynamicField name="*_txt" type="text_general" indexed="true" stored="true" multiValued="true"/>

<dynamicField name="*_en" type="text_en" indexed="true" stored="true" multiValued="true"/>

<dynamicField name="*_b" type="boolean" indexed="true" stored="true"/>

<dynamicField name="*_bs" type="boolean" indexed="true" stored="true" multiValued="true"/>

<dynamicField name="*_f" type="float" indexed="true" stored="true"/>

<dynamicField name="*_fs" type="float" indexed="true" stored="true" multiValued="true"/>

<dynamicField name="*_d" type="double" indexed="true" stored="true"/>

<dynamicField name="*_ds" type="double" indexed="true" stored="true" multiValued="true"/>

<!-- Type used to index the lat and lon components for the "location" FieldType -->

<dynamicField name="*_coordinate" type="tdouble" indexed="true" stored="false"/>

<dynamicField name="*_dt" type="date" indexed="true" stored="true"/>

<dynamicField name="*_dts" type="date" indexed="true" stored="true" multiValued="true"/>

<dynamicField name="*_p" type="location" indexed="true" stored="true"/>

<!-- some trie-coded dynamic fields for faster range queries -->

<dynamicField name="*_ti" type="tint" indexed="true" stored="true"/>

<dynamicField name="*_tl" type="tlong" indexed="true" stored="true"/>

<dynamicField name="*_tf" type="tfloat" indexed="true" stored="true"/>

<dynamicField name="*_td" type="tdouble" indexed="true" stored="true"/>

<dynamicField name="*_tdt" type="tdate" indexed="true" stored="true"/>

<dynamicField name="*_pi" type="pint" indexed="true" stored="true"/>

<dynamicField name="*_c" type="currency" indexed="true" stored="true"/>

<dynamicField name="ignored_*" type="ignored" multiValued="true"/>

<dynamicField name="attr_*" type="text_general" indexed="true" stored="true" multiValued="true"/>

<dynamicField name="random_*" type="random"/>

</fields>

<uniqueKey>id</uniqueKey>

<copyField source="content" dest="spell"/>

<copyField source="content" dest="pinyin"/>

<copyField source="content" dest="py"/>

<copyField source="path" dest="text"/>

<copyField source="content" dest="text"/>

<copyField source="pinyin" dest="text"/>

<copyField source="py" dest="text"/>

<defaultSearchField>text</defaultSearchField>

<types>

  <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="tint" class="solr.TrieIntField" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="tfloat" class="solr.TrieFloatField" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="tlong" class="solr.TrieLongField" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="tdouble" class="solr.TrieDoubleField" precisionStep="8" positionIncrementGap="0"/>

<fieldType name="date" class="solr.TrieDateField" precisionStep="0" positionIncrementGap="0"/>

<fieldType name="tdate" class="solr.TrieDateField" precisionStep="6" positionIncrementGap="0"/>

<fieldtype name="binary" class="solr.BinaryField"/>

<fieldType name="pint" class="solr.IntField"/>

<fieldType name="plong" class="solr.LongField"/>

<fieldType name="pfloat" class="solr.FloatField"/>

<fieldType name="pdouble" class="solr.DoubleField"/>

<fieldType name="pdate" class="solr.DateField" sortMissingLast="true"/>

<fieldType name="random" class="solr.RandomSortField" indexed="true"/>

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

  <analyzer>

    <tokenizer class="solr.WhitespaceTokenizerFactory"/>

  </analyzer>

</fieldType>

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

    <analyzer type="index">

      <tokenizer class="solr.StandardTokenizerFactory"/>

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

      <filter class="solr.LowerCaseFilterFactory"/>

    </analyzer>

    <analyzer type="query">

      <tokenizer class="solr.StandardTokenizerFactory"/>

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

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

      <filter class="solr.LowerCaseFilterFactory"/>

    </analyzer>

  </fieldType>

  <fieldType name="text_spell" class="solr.TextField" >

```

```
<analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"/>

</fieldType>

<fieldType name="text_ik" class="solr.TextField">

  <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"/>

</fieldType>

<fieldType name="text_smartcn" class="solr.TextField" positionIncrementGap="0">

  <analyzer type="index">

    <tokenizer class="org.apache.lucene.analysis.cn.smart.SmartChineseSentenceTokenizerFactory"/>

    <filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

  </analyzer>

  <analyzer type="query">

    <tokenizer class="org.apache.lucene.analysis.cn.smart.SmartChineseSentenceTokenizerFactory"/>

    <filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

  </analyzer>

</fieldType>

<fieldType name="text_pinyin" class="solr.TextField" positionIncrementGap="0">

  <analyzer type="index">

    <tokenizer class="org.apache.lucene.analysis.cn.smart.SmartChineseSentenceTokenizerFactory"/>

    <filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

    <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.cn.smart.SmartChineseSentenceTokenizerFactory"/>

    <filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

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

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

  </analyzer>

</fieldType>

<fieldType name="text_py" class="solr.TextField" positionIncrementGap="0">

  <analyzer type="index">

    <tokenizer class="org.apache.lucene.analysis.cn.smart.SmartChineseSentenceTokenizerFactory"/>

    <filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

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

  </analyzer>

  <analyzer type="query">
```



```
<tokenizer class="org.apache.lucene.analysis.cn.smart.SmartChineseSentenceTokenizerFactory"/>

    <filter class="org.apache.lucene.analysis.cn.smart.SmartChineseWordTokenFilterFactory"/>

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

</analyzer>

</fieldType>

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

  <analyzer type="index">

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <!-- in this example, we will only use synonyms at query time

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

    -->

    <!-- Case insensitive stop word removal.

    add enablePositionIncrements=true in both the index and query

    analyzers to leave a 'gap' for more accurate phrase queries.

    -->

    <filter class="solr.StopFilterFactory"

      ignoreCase="true"

      words="lang/stopwords_en.txt"

      enablePositionIncrements="true"

    />

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.EnglishPossessiveFilterFactory"/>

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

    <!-- Optionally you may want to use this less aggressive stemmer instead of PorterStemFilterFactory:

    <filter class="solr.EnglishMinimalStemFilterFactory"/>

    -->

    <filter class="solr.PorterStemFilterFactory"/>

  </analyzer>

  <analyzer type="query">

    <tokenizer class="solr.StandardTokenizerFactory"/>

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

    <filter class="solr.StopFilterFactory"

      ignoreCase="true"

      words="lang/stopwords_en.txt"

      enablePositionIncrements="true"

    />

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.EnglishPossessiveFilterFactory"/>
```

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

<!-- Optionally you may want to use this less aggressive stemmer instead of PorterStemFilterFactory:

    <filter class="solr.EnglishMinimalStemFilterFactory"/>

-->

<filter class="solr.PorterStemFilterFactory"/>

</analyzer>

</fieldType>

<fieldType name="text_en_splitting" class="solr.TextField" positionIncrementGap="100"

    autoGeneratePhraseQueries="true">

<analyzer type="index">

    <tokenizer class="solr.WhitespaceTokenizerFactory"/>

    <!-- in this example, we will only use synonyms at query time

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

    -->

    <!-- Case insensitive stop word removal.

    add enablePositionIncrements=true in both the index and query

    analyzers to leave a 'gap' for more accurate phrase queries.

    -->

    <filter class="solr.StopFilterFactory"

        ignoreCase="true"

        words="lang/stopwords_en.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.KeywordMarkerFilterFactory" protected="protwords.txt"/>

    <filter class="solr.PorterStemFilterFactory"/>

</analyzer>

<analyzer type="query">

    <tokenizer class="solr.WhitespaceTokenizerFactory"/>

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

    <filter class="solr.StopFilterFactory"

        ignoreCase="true"

        words="lang/stopwords_en.txt"

        enablePositionIncrements="true"

    />

    <filter class="solr.WordDelimiterFilterFactory" generateWordParts="1" generateNumberParts="1" catenateWords="0"

        catenateNumbers="0" catenateAll="0" splitOnCaseChange="1"/>
```

```
<filter class="solr.LowerCaseFilterFactory"/>

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

<filter class="solr.PorterStemFilterFactory"/>

</analyzer>

</fieldType>

<fieldType name="text_en_splitting_tight" class="solr.TextField" positionIncrementGap="100"

    autoGeneratePhraseQueries="true">

<analyzer>

    <tokenizer class="solr.WhitespaceTokenizerFactory"/>

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

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

    <filter class="solr.WordDelimiterFilterFactory" generateWordParts="0" generateNumberParts="0" catenateWords="1"

        catenateNumbers="1" catenateAll="0"/>

    <filter class="solr.LowerCaseFilterFactory"/>

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

    <filter class="solr.EnglishMinimalStemFilterFactory"/>

    <!-- this filter can remove any duplicate tokens that appear at the same position - sometimes

        possible with WordDelimiterFilter in conjuncton with stemming. -->

    <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>

</analyzer>

</fieldType>

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

<analyzer type="index">

    <tokenizer class="solr.StandardTokenizerFactory"/>

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

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.ReversedWildcardFilterFactory" withOriginal="true"

        maxPosAsterisk="3" maxPosQuestion="2" maxFractionAsterisk="0.33"/>

</analyzer>

<analyzer type="query">

    <tokenizer class="solr.StandardTokenizerFactory"/>

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

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

    <filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

</fieldType>

<fieldType name="alphaOnlySort" class="solr.TextField" sortMissingLast="true" omitNorms="true">

<analyzer>

    <!-- KeywordTokenizer does no actual tokenizing, so the entire
```

```
input string is preserved as a single token

-->

<tokenizer class="solr.KeywordTokenizerFactory"/>

<!-- The LowerCase TokenFilter does what you expect, which can be
      when you want your sorting to be case insensitive
-->

<filter class="solr.LowerCaseFilterFactory"/>

<!-- The TrimFilter removes any leading or trailing whitespace -->

<filter class="solr.TrimFilterFactory"/>

<!-- The PatternReplaceFilter gives you the flexibility to use
      java Regular expression to replace any sequence of characters
      matching a pattern with an arbitrary replacement string,
      which may include back references to portions of the original
      string matched by the pattern.

      See the Java Regular Expression documentation for more
      information on pattern and replacement string syntax.

      http://java.sun.com/j2se/1.6.0/docs/api/java/util/regex/package-summary.html
-->

<filter class="solr.PatternReplaceFilterFactory"
      pattern="([a-z])" replacement="" replace="all"
/>

</analyzer>

</fieldType>

<fieldtype name="phonetic" stored="false" indexed="true" class="solr.TextField">

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.DoubleMetaphoneFilterFactory" inject="false"/>

  </analyzer>

</fieldtype>

<fieldtype name="payloads" stored="false" indexed="true" class="solr.TextField">

  <analyzer>

    <tokenizer class="solr.WhitespaceTokenizerFactory"/>

    <!--

    The DelimitedPayloadTokenFilter can put payloads on tokens... for example,

    a token of "foo|1.4" would be indexed as "foo" with a payload of 1.4f

    Attributes of the DelimitedPayloadTokenFilterFactory :

    "delimiter" - a one character delimiter. Default is | (pipe)

    -->
```

```

"encoder" - how to encode the following value into a payload

float -> org.apache.lucene.analysis.payloads.FloatEncoder,

integer -> o.a.l.a.p.IntegerEncoder

identity -> o.a.l.a.p.IdentityEncoder

Fully Qualified class name implementing PayloadEncoder, Encoder must have a no arg constructor.

-->

<filter class="solr.DelimitedPayloadTokenFilterFactory" encoder="float"/>

</analyzer>

</fieldType>

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

  <analyzer>

    <tokenizer class="solr.KeywordTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

  </analyzer>

</fieldType>

<fieldType name="descendent_path" class="solr.TextField">

  <analyzer type="index">

    <tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/" />

  </analyzer>

  <analyzer type="query">

    <tokenizer class="solr.KeywordTokenizerFactory"/>

  </analyzer>

</fieldType>

<fieldType name="ancestor_path" class="solr.TextField">

  <analyzer type="index">

    <tokenizer class="solr.KeywordTokenizerFactory"/>

  </analyzer>

  <analyzer type="query">

    <tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/" />

  </analyzer>

</fieldType>

<fieldtype name="ignored" stored="false" indexed="false" multiValued="true" class="solr.StrField"/>

<fieldType name="point" class="solr.PointType" dimension="2" subFieldSuffix="_d"/>

<fieldType name="location" class="solr.LatLonType" subFieldSuffix="_coordinate"/>

<fieldType name="location_rpt" class="solr.SpatialRecursivePrefixTreeFieldType"

  geo="true" distErrPct="0.025" maxDistErr="0.000009" units="degrees"/>

<fieldType name="currency" class="solr.CurrencyField" precisionStep="8" defaultCurrency="USD"

  currencyConfig="currency.xml"/>

<!-- some examples for different languages (generally ordered by ISO code) -->

```

```
<!-- Arabic -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <!-- for any non-arabic -->

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_ar.txt"

      enablePositionIncrements="true"/>

    <!-- normalizes ى to ي, etc -->

    <filter class="solr.ArabicNormalizationFilterFactory"/>

    <filter class="solr.ArabicStemFilterFactory"/>

  </analyzer>

</fieldType>

<!-- Bulgarian -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_bg.txt"

      enablePositionIncrements="true"/>

    <filter class="solr.BulgarianStemFilterFactory"/>

  </analyzer>

</fieldType>

<!-- Catalan -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <!-- removes l', etc -->

    <filter class="solr.ElisionFilterFactory" ignoreCase="true" articles="lang/contractions_ca.txt"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_ca.txt"

      enablePositionIncrements="true"/>

    <filter class="solr.SnowballPorterFilterFactory" language="Catalan"/>

  </analyzer>

</fieldType>

<!-- CJK bigram (see text_ja for a Japanese configuration using morphological analysis) -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>
```

```
<!-- normalize width before bigram, as e.g. half-width dakuten combine -->

<filter class="solr.CJKWidthFilterFactory"/>

<!-- for any non-CJK -->

<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.CJKBigramFilterFactory"/>

</analyzer>

</fieldType>

<!-- Czech -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_cz.txt"

      enablePositionIncrements="true"/>

    <filter class="solr.CzechStemFilterFactory"/>

  </analyzer>

</fieldType>

<!-- Danish -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_da.txt" format="snowball"

      enablePositionIncrements="true"/>

    <filter class="solr.SnowballPorterFilterFactory" language="Danish"/>

  </analyzer>

</fieldType>

<!-- German -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_de.txt" format="snowball"

      enablePositionIncrements="true"/>

    <filter class="solr.GermanNormalizationFilterFactory"/>

    <filter class="solr.GermanLightStemFilterFactory"/>

    <!-- less aggressive: <filter class="solr.GermanMinimalStemFilterFactory"/> -->

    <!-- more aggressive: <filter class="solr.SnowballPorterFilterFactory" language="German2"/> -->

  </analyzer>
```

```
</fieldType>

<!-- Greek -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <!-- greek specific lowercase for sigma -->

    <filter class="solr.GreekLowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="false" words="lang/stopwords_el.txt"

      enablePositionIncrements="true"/>

    <filter class="solr.GreekStemFilterFactory"/>

  </analyzer>

</fieldType>

<!-- Spanish -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_es.txt" format="snowball"

      enablePositionIncrements="true"/>

    <filter class="solr.SpanishLightStemFilterFactory"/>

    <!-- more aggressive: <filter class="solr.SnowballPorterFilterFactory" language="Spanish"/> -->

  </analyzer>

</fieldType>

<!-- Basque -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_eu.txt"

      enablePositionIncrements="true"/>

    <filter class="solr.SnowballPorterFilterFactory" language="Basque"/>

  </analyzer>

</fieldType>

<!-- Persian -->

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

  <analyzer>

    <!-- for ZWNJ -->

    <charFilter class="solr.PersianCharFilterFactory"/>

    <tokenizer class="solr.StandardTokenizerFactory"/>
```



```
<filter class="solr.LowerCaseFilterFactory"/>

<filter class="solr.ArabicNormalizationFilterFactory"/>

<filter class="solr.PersianNormalizationFilterFactory"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_fa.txt"

    enablePositionIncrements="true"/>

</analyzer>

</fieldType>

<!-- Finnish -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_fi.txt" format="snowball"

        enablePositionIncrements="true"/>

    <filter class="solr.SnowballPorterFilterFactory" language="Finnish"/>

    <!-- less aggressive: <filter class="solr.FinnishLightStemFilterFactory"/> -->

  </analyzer>

</fieldType>

<!-- French -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <!-- removes 'l', etc -->

    <filter class="solr.ElisionFilterFactory" ignoreCase="true" articles="lang/contractions_fr.txt"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_fr.txt" format="snowball"

        enablePositionIncrements="true"/>

    <filter class="solr.FrenchLightStemFilterFactory"/>

    <!-- less aggressive: <filter class="solr.FrenchMinimalStemFilterFactory"/> -->

    <!-- more aggressive: <filter class="solr.SnowballPorterFilterFactory" language="French"/> -->

  </analyzer>

</fieldType>

<!-- Irish -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <!-- removes d', etc -->

    <filter class="solr.ElisionFilterFactory" ignoreCase="true" articles="lang/contractions_ga.txt"/>

    <!-- removes n-, etc. position increments is intentionally false! -->
```

```
<filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/hyphenations_ga.txt"

    enablePositionIncrements="false"/>

<filter class="solr.IrishLowerCaseFilterFactory"/>

<filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_ga.txt"

    enablePositionIncrements="true"/>

<filter class="solr.SnowballPorterFilterFactory" language="Irish"/>

</analyzer>

</fieldType>

<!-- Galician -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_gl.txt"

        enablePositionIncrements="true"/>

    <filter class="solr.GalicianStemFilterFactory"/>

    <!-- less aggressive: <filter class="solr.GalicianMinimalStemFilterFactory"/> -->

  </analyzer>

</fieldType>

<!-- Hindi -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <!-- normalizes unicode representation -->

    <filter class="solr.IndicNormalizationFilterFactory"/>

    <!-- normalizes variation in spelling -->

    <filter class="solr.HindiNormalizationFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_hi.txt"

        enablePositionIncrements="true"/>

    <filter class="solr.HindiStemFilterFactory"/>

  </analyzer>

</fieldType>

<!-- Hungarian -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_hu.txt" format="snowball"
```

```
        enablePositionIncrements="true"/>

        <filter class="solr.SnowballPorterFilterFactory" language="Hungarian"/>

        <!-- less aggressive: <filter class="solr.HungarianLightStemFilterFactory"/> -->

    </analyzer>

</fieldType>

<!-- Armenian -->

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

    <analyzer>

        <tokenizer class="solr.StandardTokenizerFactory"/>

        <filter class="solr.LowerCaseFilterFactory"/>

        <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_hy.txt"

            enablePositionIncrements="true"/>

        <filter class="solr.SnowballPorterFilterFactory" language="Armenian"/>

    </analyzer>

</fieldType>

<!-- Indonesian -->

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

    <analyzer>

        <tokenizer class="solr.StandardTokenizerFactory"/>

        <filter class="solr.LowerCaseFilterFactory"/>

        <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_id.txt"

            enablePositionIncrements="true"/>

        <!-- for a less aggressive approach (only inflectional suffixes), set stemDerivational to false -->

        <filter class="solr.IndonesianStemFilterFactory" stemDerivational="true"/>

    </analyzer>

</fieldType>

<!-- Italian -->

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

    <analyzer>

        <tokenizer class="solr.StandardTokenizerFactory"/>

        <!-- removes l', etc -->

        <filter class="solr.ElisionFilterFactory" ignoreCase="true" articles="lang/contractions_it.txt"/>

        <filter class="solr.LowerCaseFilterFactory"/>

        <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_it.txt" format="snowball"

            enablePositionIncrements="true"/>

        <filter class="solr.ItalianLightStemFilterFactory"/>

        <!-- more aggressive: <filter class="solr.SnowballPorterFilterFactory" language="Italian"/> -->

    </analyzer>

</fieldType>
```

```

<!-- Japanese using morphological analysis (see text_cjk for a configuration using bigramming)

NOTE: If you want to optimize search for precision, use default operator AND in your query
parser config with <solrQueryParser defaultOperator="AND"/> further down in this file. Use
OR if you would like to optimize for recall (default).

-->

<fieldType name="text_ja" class="solr.TextField" positionIncrementGap="100" autoGeneratePhraseQueries="false">

  <analyzer>

    <!-- Kuromoji Japanese morphological analyzer/tokenizer (JapaneseTokenizer)

    Kuromoji has a search mode (default) that does segmentation useful for search. A heuristic
    is used to segment compounds into its parts and the compound itself is kept as synonym.

    Valid values for attribute mode are:

        normal: regular segmentation

        search: segmentation useful for search with synonyms compounds (default)

        extended: same as search mode, but unigrams unknown words (experimental)

    For some applications it might be good to use search mode for indexing and normal mode for
    queries to reduce recall and prevent parts of compounds from being matched and highlighted.
    Use <analyzer type="index"> and <analyzer type="query"> for this and mode normal in query.

    Kuromoji also has a convenient user dictionary feature that allows overriding the statistical
    model with your own entries for segmentation, part-of-speech tags and readings without a need
    to specify weights. Notice that user dictionaries have not been subject to extensive testing.

    User dictionary attributes are:

        userDictionary: user dictionary filename

        userDictionaryEncoding: user dictionary encoding (default is UTF-8)

    See lang/userdict_ja.txt for a sample user dictionary file.

    Punctuation characters are discarded by default. Use discardPunctuation="false" to keep them.

    See http://wiki.apache.org/solr/JapaneseLanguageSupport for more on Japanese language support.

    -->

    <tokenizer class="solr.JapaneseTokenizerFactory" mode="search"/>

    <!--<tokenizer class="solr.JapaneseTokenizerFactory" mode="search" userDictionary="lang/userdict_ja.txt"/>-->

    <!-- Reduces inflected verbs and adjectives to their base/dictionary forms (辞書形) -->

    <filter class="solr.JapaneseBaseFormFilterFactory"/>

    <!-- Removes tokens with certain part-of-speech tags -->

    <filter class="solr.JapanesePartOfSpeechStopFilterFactory" tags="lang/stoptags_ja.txt"

        enablePositionIncrements="true"/>

    <!-- Normalizes full-width romaji to half-width and half-width kana to full-width (Unicode NFKC subset) -->

    <filter class="solr.CJKWidthFilterFactory"/>

    <!-- Removes common tokens typically not useful for search, but have a negative effect on ranking -->

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_ja.txt"

        enablePositionIncrements="true"/>

```

```
<!-- Normalizes common katakana spelling variations by removing any last long sound character (U+30FC) -->

<filter class="solr.JapaneseKatakanaStemFilterFactory" minimumLength="4"/>

<!-- Lower-cases romaji characters -->

<filter class="solr.LowerCaseFilterFactory"/>

</analyzer>

</fieldType>

<!-- Latvian -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_lv.txt"

      enablePositionIncrements="true"/>

    <filter class="solr.LatvianStemFilterFactory"/>

  </analyzer>

</fieldType>

<!-- Dutch -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_nl.txt" format="snowball"

      enablePositionIncrements="true"/>

    <filter class="solr.StemmerOverrideFilterFactory" dictionary="lang/stemdict_nl.txt" ignoreCase="false"/>

    <filter class="solr.SnowballPorterFilterFactory" language="Dutch"/>

  </analyzer>

</fieldType>

<!-- Norwegian -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_no.txt" format="snowball"

      enablePositionIncrements="true"/>

    <filter class="solr.SnowballPorterFilterFactory" language="Norwegian"/>

    <!-- less aggressive: <filter class="solr.NorwegianLightStemFilterFactory"/> -->

    <!-- singular/plural: <filter class="solr.NorwegianMinimalStemFilterFactory"/> -->

  </analyzer>

</fieldType>
```

```
<!-- Portuguese -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_pt.txt" format="snowball"

      enablePositionIncrements="true"/>

    <filter class="solr.PortugueseLightStemFilterFactory"/>

    <!-- less aggressive: <filter class="solr.PortugueseMinimalStemFilterFactory"/> -->

    <!-- more aggressive: <filter class="solr.SnowballPorterFilterFactory" language="Portuguese"/> -->

    <!-- most aggressive: <filter class="solr.PortugueseStemFilterFactory"/> -->

  </analyzer>

</fieldType>

<!-- Romanian -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_ro.txt"

      enablePositionIncrements="true"/>

    <filter class="solr.SnowballPorterFilterFactory" language="Romanian"/>

  </analyzer>

</fieldType>

<!-- Russian -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_ru.txt" format="snowball"

      enablePositionIncrements="true"/>

    <filter class="solr.SnowballPorterFilterFactory" language="Russian"/>

    <!-- less aggressive: <filter class="solr.RussianLightStemFilterFactory"/> -->

  </analyzer>

</fieldType>

<!-- Swedish -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>
```

```
<filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/stopwords_sv.txt" format="snowball"
    enablePositionIncrements="true"/>

<filter class="solr.SnowballPorterFilterFactory" language="Swedish"/>

<!-- less aggressive: <filter class="solr.SwedishLightStemFilterFactory"/> -->

</analyzer>

</fieldType>

<!-- Thai -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.LowerCaseFilterFactory"/>

    <filter class="solr.ThaiWordFilterFactory"/>

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

  </analyzer>

</fieldType>

<!-- Turkish -->

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

  <analyzer>

    <tokenizer class="solr.StandardTokenizerFactory"/>

    <filter class="solr.TurkishLowerCaseFilterFactory"/>

    <filter class="solr.StopFilterFactory" ignoreCase="false" words="lang/stopwords_tr.txt"
        enablePositionIncrements="true"/>

    <filter class="solr.SnowballPorterFilterFactory" language="Turkish"/>

  </analyzer>

</fieldType>

</types>

</schema>
```

附2: solrconfig.xml

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

<config>

  <.luceneMatchVersion>LUCENE_42</luceneMatchVersion>

  <lib dir="../../../lib" regex=".*\\.jar" />

  <lib dir="../../../contrib/extraction/lib" regex=".*\\.jar" />

  <lib dir="../../../dist/" regex="solr-cell-\\d.*\\.jar" />

  <lib dir="../../../contrib/clustering/lib/" regex=".*\\.jar" />

  <lib dir="../../../dist/" regex="solr-clustering-\\d.*\\.jar" />

  <lib dir="../../../contrib/langid/lib/" regex=".*\\.jar" />

  <lib dir="../../../dist/" regex="solr-langid-\\d.*\\.jar" />

  <lib dir="../../../contrib/velocity/lib" regex=".*\\.jar" />

  <lib dir="../../../dist/" regex="solr-velocity-\\d.*\\.jar" />

  <lib dir="/total/crap/dir/ignored" />

  <dataDir>${solr.data.dir}</dataDir>

  <directoryFactory name="DirectoryFactory"

    class="${solr.directoryFactory:solr.NRTCachingDirectoryFactory}"/>

  <codecFactory class="solr.SchemaCodecFactory"/>

  <indexConfig>

    <!-- maxFieldLength was removed in 4.0. To get similar behavior, include a

      LimitTokenCountFilterFactory in your fieldType definition. E.g.

      <filter class="solr.LimitTokenCountFilterFactory" maxTokenCount="10000"/>

    -->

    <!-- Maximum time to wait for a write lock (ms) for an IndexWriter. Default: 1000 -->

    <!-- <writeLockTimeout>1000</writeLockTimeout> -->

    <!-- The maximum number of simultaneous threads that may be

      indexing documents at once in IndexWriter; if more than this

      many threads arrive they will wait for others to finish.

      Default in Solr/Lucene is 8. -->

    <!-- <maxIndexingThreads>8</maxIndexingThreads> -->

    <!-- Expert: Enabling compound file will use less files for the index,

      using fewer file descriptors on the expense of performance decrease.

      Default in Lucene is "true". Default in Solr is "false" (since 3.6) -->

    <!-- <useCompoundFile>false</useCompoundFile> -->

    <!-- ramBufferSizeMB sets the amount of RAM that may be used by Lucene

      indexing for buffering added documents and deletions before they are

      flushed to the Directory.

      maxBufferedDocs sets a limit on the number of documents buffered

      before flushing.
```


If both ramBufferSizeMB and maxBufferedDocs is set, then

Lucene will flush based on whichever limit is hit first. -->

```
<ramBufferSizeMB>100</ramBufferSizeMB>
```

```
<maxBufferedDocs>1000</maxBufferedDocs>
```

<!-- Expert: Merge Policy

The Merge Policy in Lucene controls how merging of segments is done.

The default since Solr/Lucene 3.3 is TieredMergePolicy.

The default since Lucene 2.3 was the LogByteSizeMergePolicy,

Even older versions of Lucene used LogDocMergePolicy.

```
<mergePolicy class="org.apache.lucene.index.TieredMergePolicy">
```

```
<int name="maxMergeAtOnce">100</int>
```

```
<int name="segmentsPerTier">100</int>
```

```
</mergePolicy>
```

-->

<!-- Merge Factor

The merge factor controls how many segments will get merged at a time.

For TieredMergePolicy, mergeFactor is a convenience parameter which

will set both MaxMergeAtOnce and SegmentsPerTier at once.

For LogByteSizeMergePolicy, mergeFactor decides how many new segments

will be allowed before they are merged into one.

Default is 10 for both merge policies.

-->

```
<mergeFactor>50</mergeFactor>
```

<!-- Expert: Merge Scheduler

The Merge Scheduler in Lucene controls how merges are

performed. The ConcurrentMergeScheduler (Lucene 2.3 default)

can perform merges in the background using separate threads.

The SerialMergeScheduler (Lucene 2.2 default) does not.

-->

<!--

```
<mergeScheduler class="org.apache.lucene.index.ConcurrentMergeScheduler"/>
```

-->

<!-- LockFactory

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 - uses OS native file locking.

    Do not use when multiple solr webapps in the same
    JVM are attempting to share a single index.

simple = SimpleFSLockFactory - uses a plain file for locking

Defaults: 'native' is default for Solr3.6 and later, otherwise
    'simple' is the default

More details on the nuances of each LockFactory...

http://wiki.apache.org/lucene-java/AvailableLockFactories
-->

<lockType>${solr.lock.type:native}</lockType>

<!-- Unlock On Startup

    If true, unlock any held write or commit locks on startup.

    This defeats the locking mechanism that allows multiple
    processes to safely access a lucene index, and should be used
    with care. Default is "false".

    This is not needed if lock type is 'single'
-->

<!--
<unlockOnStartup>false</unlockOnStartup>

-->

<!-- Expert: Controls how often Lucene loads terms into memory

    Default is 128 and is likely good for most everyone.
-->

<!-- <termIndexInterval>128</termIndexInterval> -->

<!-- If true, IndexReaders will be reopened (often more efficient)
    instead of closed and then opened. Default: true
-->

<!--
<reopenReaders>true</reopenReaders>

-->

<!-- Commit Deletion Policy

    Custom deletion policies can be specified here. The class must
    implement org.apache.lucene.index.IndexDeletionPolicy.

    The default Solr IndexDeletionPolicy implementation supports

```

deleting index commit points on number of commits, age of
commit point and optimized status.

The latest commit point should always be preserved regardless
of the criteria.

```
-->
```

```
<!--
```

```
<deletionPolicy class="solr.SolrDeletionPolicy">
```

```
-->
```

```
<!-- The number of commit points to be kept -->
```

```
<!-- <str name="maxCommitsToKeep">1</str> -->
```

```
<!-- The number of optimized commit points to be kept -->
```

```
<!-- <str name="maxOptimizedCommitsToKeep">0</str> -->
```

```
<!--
```

Delete all commit points once they have reached the given age.

Supports DateMathParser syntax e.g.

```
-->
```

```
<!--
```

```
<str name="maxCommitAge">30MINUTES</str>
```

```
<str name="maxCommitAge">1DAY</str>
```

```
-->
```

```
<!--
```

```
</deletionPolicy>
```

```
-->
```

```
<!-- Lucene Infostream
```

To aid in advanced debugging, Lucene provides an "InfoStream"
of detailed information when indexing.

Setting The value to true will instruct the underlying Lucene

IndexWriter to write its debugging info the specified file

```
-->
```

```
<!-- <infoStream file="INFOSTREAM.txt">false</infoStream> -->
```

```
</indexConfig>
```

```
<jmx />
```

```
<updateHandler class="solr.DirectUpdateHandler2">
```

```
<updateLog>
```

```
<str name="dir">${solr.ulog.dir}</str>
```

```
</updateLog>
```

```
<autoCommit>

  <maxDocs>1000</maxDocs>

  <maxTime>15000</maxTime>

  <openSearcher>false</openSearcher>

</autoCommit>

</updateHandler>

<query>

<!-- Max Boolean Clauses

  Maximum number of clauses in each BooleanQuery, an exception
  is thrown if exceeded.

  ** WARNING **

  This option actually modifies a global Lucene property that
  will affect all SolrCores. If multiple solrconfig.xml files
  disagree on this property, the value at any given moment will
  be based on the last SolrCore to be initialized.

-->

<maxBooleanClauses>1024</maxBooleanClauses>

<!-- Solr Internal Query Caches

  There are two implementations of cache available for Solr,
  LRUCache, based on a synchronized LinkedHashMap, and
  FastLRUCache, based on a ConcurrentHashMap.

  FastLRUCache has faster gets and slower puts in single
  threaded operation and thus is generally faster than LRUCache
  when the hit ratio of the cache is high (> 75%), and may be
  faster under other scenarios on multi-cpu systems.

-->

<!-- Filter Cache

  Cache used by SolrIndexSearcher for filters (DocSets),
  unordered sets of *all* documents that match a query. When a
  new searcher is opened, its caches may be prepopulated or
  "autowarmed" using data from caches in the old searcher.

  autowarmCount is the number of items to prepopulate. For
  LRUCache, the autowarmed items will be the most recently
  accessed items.
```

Parameters:

class - the SolrCache implementation LRUCache or

(LRUCache or FastLRUCache)

size - the maximum number of entries in the cache

initialSize - the initial capacity (number of entries) of

the cache. (see java.util.HashMap)

autowarmCount - the number of entries to prepopulate from

and old cache.

-->

```
<filterCache class="solr.FastLRUCache"
```

```
  size="512"
```

```
  initialSize="512"
```

```
  autowarmCount="0"/>
```

```
<!-- Query Result Cache
```

Caches results of searches - ordered lists of document ids

(DocList) based on a query, a sort, and the range of documents requested.

-->

```
<queryResultCache class="solr.LRUCache"
```

```
  size="512"
```

```
  initialSize="512"
```

```
  autowarmCount="0"/>
```

```
<!-- Document Cache
```

Caches Lucene Document objects (the stored fields for each

document). Since Lucene internal document ids are transient,

this cache will not be autowarmed.

-->

```
<documentCache class="solr.LRUCache"
```

```
  size="512"
```

```
  initialSize="512"
```

```
  autowarmCount="0"/>
```

```
<!-- Field Value Cache
```

Cache used to hold field values that are quickly accessible

by document id. The fieldValueCache is created by default

even if not configured here.

-->

```
<!--  
  
  <fieldValueCache class="solr.FastLRUCache"  
  
    size="512"  
  
    autowarmCount="128"  
  
    showItems="32" />  
  
-->  
  
<!-- Custom Cache  
  
  Example of a generic cache. These caches may be accessed by  
  name through SolrIndexSearcher.getCache(),cacheLookup(), and  
  cacheInsert(). The purpose is to enable easy caching of  
  user/application level data. The regenerator argument should  
  be specified as an implementation of solr.CacheRegenerator  
  if autowarming is desired.  
  
-->  
  
<!--  
  
  <cache name="myUserCache"  
  
    class="solr.LRUCache"  
  
    size="4096"  
  
    initialSize="1024"  
  
    autowarmCount="1024"  
  
    regenerator="com.mycompany.MyRegenerator"  
  
    />  
  
-->  
  
<!-- Lazy Field Loading  
  
  If true, stored fields that are not requested will be loaded  
  lazily. This can result in a significant speed improvement  
  if the usual case is to not load all stored fields,  
  especially if the skipped fields are large compressed text  
  fields.  
  
-->  
  
<enableLazyFieldLoading>true</enableLazyFieldLoading>  
  
<!-- Use Filter For Sorted Query  
  
  A possible optimization that attempts to use a filter to  
  satisfy a search. If the requested sort does not include  
  score, then the filterCache will be checked for a filter  
  matching the query. If found, the filter will be used as the  
  source of document ids, and then the sort will be applied to  
  that.  
  
  For most situations, this will not be useful unless you
```

```
frequently get the same search repeatedly with different sort

options, and none of them ever use "score"

-->

<!--

<useFilterForSortedQuery>true</useFilterForSortedQuery>

-->

<!-- Result Window Size

An optimization for use with the queryResultCache. When a search

is requested, a superset of the requested number of document ids

are collected. For example, if a search for a particular query

requests matching documents 10 through 19, and queryWindowSize is 50,

then documents 0 through 49 will be collected and cached. Any further

requests in that range can be satisfied via the cache.

-->

<queryResultWindowSize>20</queryResultWindowSize>

<!-- Maximum number of documents to cache for any entry in the

queryResultCache.

-->

<queryResultMaxDocsCached>200</queryResultMaxDocsCached>

<!-- Query Related Event Listeners

Various IndexSearcher related events can trigger Listeners to

take actions.

newSearcher - fired whenever a new searcher is being prepared

and there is a current searcher handling requests (aka

registered). It can be used to prime certain caches to

prevent long request times for certain requests.

firstSearcher - fired whenever a new searcher is being

prepared but there is no current registered searcher to handle

requests or to gain autowarming data from.

-->

<!-- QuerySenderListener takes an array of NamedList and executes a

local query request for each NamedList in sequence.

-->

<listener event="newSearcher" class="solr.QuerySenderListener">

<arr name="queries">

<!--

<lst><str name="q">solr</str><str name="sort">price asc</str></lst>

<lst><str name="q">rocks</str><str name="sort">weight asc</str></lst>
```

```

-->

</arr>

</listener>

<listener event="firstSearcher" class="solr.QuerySenderListener">

  <arr name="queries">

    <lst>

      <str name="q">static firstSearcher warming in solrconfig.xml</str>

    </lst>

  </arr>

</listener>

<!-- Use Cold Searcher

  If a search request comes in and there is no current
  registered searcher, then immediately register the still
  warming searcher and use it. If "false" then all requests
  will block until the first searcher is done warming.

-->

<useColdSearcher>false</useColdSearcher>

<!-- Max Warming Searchers

  Maximum number of searchers that may be warming in the
  background concurrently. An error is returned if this limit
  is exceeded.

  Recommend values of 1-2 for read-only slaves, higher for
  masters w/o cache warming.

-->

<maxWarmingSearchers>2</maxWarmingSearchers>

</query>

<requestDispatcher handleSelect="false" >

  <requestParsers enableRemoteStreaming="true"

    multipartUploadLimitInKB="2048000"

    formdataUploadLimitInKB="2048"/>

  <httpCaching never304="true" />

</requestDispatcher>

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

  <lst name="defaults">

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

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

```



```
<str name="df">text</str>

</lst>

</requestHandler>

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

  <lst name="defaults">

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

    <str name="wt">json</str>

    <str name="indent">true</str>

    <str name="df">text</str>

  </lst>

</requestHandler>

<requestHandler name="/get" class="solr.RealTimeGetHandler">

  <lst name="defaults">

    <str name="omitHeader">true</str>

    <str name="wt">json</str>

    <str name="indent">true</str>

  </lst>

</requestHandler>

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

  <lst name="defaults">

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

    <!-- VelocityResponseWriter settings -->

    <str name="wt">velocity</str>

    <str name="v.template">browse</str>

    <str name="v.layout">layout</str>

    <str name="title">Solritas</str>

    <!-- Query settings -->

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

    <str name="qf">

      text^0.5 features^1.0 name^1.2 sku^1.5 id^10.0 manu^1.1 cat^1.4

      title^10.0 description^5.0 keywords^5.0 author^2.0 resourcename^1.0

    </str>

    <str name="df">text</str>

    <str name="mm">100%</str>

    <str name="q.alt">*:*</str>

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

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

    <str name="mlt.qf">
```

```
text^0.5 features^1.0 name^1.2 sku^1.5 id^10.0 manu^1.1 cat^1.4

title^10.0 description^5.0 keywords^5.0 author^2.0 resourcename^1.0

</str>

<str name="mlt.fl">text,features,name,sku,id,manu,cat,title,description,keywords,author,resourcename</str>

<int name="mlt.count">3</int>

<!-- Faceting defaults -->

<str name="facet">on</str>

<str name="facet.field">cat</str>

<str name="facet.field">manu_exact</str>

<str name="facet.field">content_type</str>

<str name="facet.field">author_s</str>

<str name="facet.query">ipod</str>

<str name="facet.query">GB</str>

<str name="facet.mincount">1</str>

<str name="facet.pivot">cat,inStock</str>

<str name="facet.range.other">after</str>

<str name="facet.range">price</str>

<int name="f.price.facet.range.start">0</int>

<int name="f.price.facet.range.end">600</int>

<int name="f.price.facet.range.gap">50</int>

<str name="facet.range">popularity</str>

<int name="f.popularity.facet.range.start">0</int>

<int name="f.popularity.facet.range.end">10</int>

<int name="f.popularity.facet.range.gap">3</int>

<str name="facet.range">manufacturedate_dt</str>

<str name="f.manufacturedate_dt.facet.range.start">NOW/YEAR-10YEARS</str>

<str name="f.manufacturedate_dt.facet.range.end">NOW</str>

<str name="f.manufacturedate_dt.facet.range.gap">+1YEAR</str>

<str name="f.manufacturedate_dt.facet.range.other">before</str>

<str name="f.manufacturedate_dt.facet.range.other">after</str>

<!-- Highlighting defaults -->

<str name="hl">on</str>

<str name="hl.fl">content features title name</str>

<str name="hl.encoder">html</str>

<str name="hl.simple.pre">&lt;b>&gt;</str>

<str name="hl.simple.post">&lt;/b></str>

<str name="f.title.hl.fragsize">0</str>

<str name="f.title.hl.alternateField">title</str>

<str name="f.name.hl.fragsize">0</str>
```

```

<str name="f.name.hl.alternateField">name</str>

<str name="f.content.hl.snippets">3</str>

<str name="f.content.hl.fragsize">200</str>

<str name="f.content.hl.alternateField">content</str>

<str name="f.content.hl.maxAlternateFieldLength">750</str>

<!-- Spell checking defaults -->

<str name="spellcheck">on</str>

<str name="spellcheck.extendedResults">false</str>

<str name="spellcheck.count">5</str>

<str name="spellcheck.alternativeTermCount">2</str>

<str name="spellcheck.maxResultsForSuggest">5</str>

<str name="spellcheck.collate">true</str>

<str name="spellcheck.collateExtendedResults">true</str>

<str name="spellcheck.maxCollationTries">5</str>

<str name="spellcheck.maxCollations">3</str>

</lst>

<!-- append spellchecking to our list of components -->

<arr name="last-components">

  <str>spellcheck</str>

</arr>

</requestHandler>

<requestHandler name="/update" class="solr.UpdateRequestHandler">

</requestHandler>

<requestHandler name="/update/json" class="solr.JsonUpdateRequestHandler">

  <lst name="defaults">

    <str name="stream.contentType">application/json</str>

  </lst>

</requestHandler>

<requestHandler name="/update/csv" class="solr.CSVRequestHandler">

  <lst name="defaults">

    <str name="stream.contentType">application/csv</str>

  </lst>

</requestHandler>

<requestHandler name="/update/extract"

  startup="lazy"

  class="solr.extraction.ExtractingRequestHandler" >

<lst name="defaults">

  <str name="lowernames">true</str>

  <str name="uprefix">ignored_</str>

```

```
<!-- capture link hrefs but ignore div attributes -->

<str name="captureAttr">true</str>

<str name="fmap.a">links</str>

<str name="fmap.div">ignored_</str>

</lst>

</requestHandler>

<requestHandler name="/analysis/field"

    startup="lazy"

    class="solr.FieldAnalysisRequestHandler" />

<requestHandler name="/analysis/document"

    class="solr.DocumentAnalysisRequestHandler"

    startup="lazy" />

<requestHandler name="/admin/"

    class="solr.admin.AdminHandlers" />

<requestHandler name="/admin/ping" class="solr.PingRequestHandler">

    <lst name="invariants">

        <str name="q">solrpingquery</str>

    </lst>

    <lst name="defaults">

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

    </lst>

</requestHandler>

<!-- Echo the request contents back to the client -->

<requestHandler name="/debug/dump" class="solr.DumpRequestHandler" >

    <lst name="defaults">

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

        <str name="echoHandler">true</str>

    </lst>

</requestHandler>

<requestHandler name="/replication" class="solr.ReplicationHandler" >

</requestHandler>

<!-- spell -->

<searchComponent name="spellcheck" class="solr.SpellCheckComponent">

    <lst name="spellchecker">

        <str name="name">direct</str>

        <str name="field">spell</str>

        <str name="classname">solr.DirectSolrSpellChecker</str>

        <str name="distanceMeasure">internal</str>

        <float name="accuracy">0.5</float>
```

```

<int name="maxEdits">2</int>

<int name="minPrefix">1</int>

<int name="maxInspections">5</int>

<int name="minQueryLength">2</int>

<float name="maxQueryFrequency">0.001</float>

    <str name="buildOnCommit">true</str>

</lst>

<!--

    Optional, it is required when more than one spellchecker is configured.

    Select non-default name with spellcheck.dictionary in request handler.
-->

```

name是可选的，如果只有一个spellchecker可以不写name

如果有多个spellchecker，需要在Request Handler中指定spellcheck.dictionary

```

-->

<str name="name">default</str>

<!-- The classname is optional, defaults to IndexBasedSpellChecker -->

<str name="classname">solr.IndexBasedSpellChecker</str>

<!--

    Load tokens from the following field for spell checking,

    analyzer for the field's type as defined in schema.xml are used
-->

```

下面这个field名字指的是拼写检查的依据，也就是说要根据哪个Field来检查用户输入。

```

-->

<str name="field">spell</str>

<!-- Optional, by default use in-memory index (RAMDirectory)
-->

```

SpellCheck索引文件的存放位置，是可选的，如果不写默认使用内存模式RAMDirectory。

./spellchecker1指的是：corex\data\spellchecker1

```

-->

<str name="spellcheckIndexDir">./spellchecker1</str>

<!-- Set the accuracy (float) to be used for the suggestions. Default is 0.5 -->

<str name="accuracy">0.7</str>

<!--何时创建拼写索引： buildOnCommit/buildOnOptimize -->

    <str name="buildOnCommit">true</str>

</lst>

<!-- 另一个拼写检查器，使用JaroWinklerDistance距离算法 -->

```

```

<!--

    <lst name="spellchecker">

        <str name="name">jarowinkler</str>

        <str name="classname">solr.IndexBasedSpellChecker</str>

        <str name="field">spell</str>
    </lst>
-->

```

```
<str name="distanceMeasure">org.apache.lucene.search.spell.JaroWinklerDistance</str>

<str name="spellcheckIndexDir">./spellchecker2</str>

<str name="buildOnCommit">true</str>

</lst>

<!-- 另一个拼写检查器，使用文件内容为检查依据

<lst name="spellchecker">

  <str name="classname">solr.FileBasedSpellChecker</str>

  <str name="name">file</str>

  <str name="sourceLocation">spellings.txt</str>

  <str name="characterEncoding">UTF-8</str>

  <str name="spellcheckIndexDir">./spellcheckerFile</str>

  <str name="buildOnCommit">true</str>

</lst>-->

<str name="queryAnalyzerFieldType">text_spell</str>

</searchComponent>

<queryConverter name="queryConverter" class="solr.SpellingQueryConverter"/>

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

  <lst name="defaults">

    <str name="spellcheck.dictionary">default</str>

    <str name="spellcheck.collate">true</str>

    <str name="spellcheck.onlyMorePopular">true</str>

    <str name="spellcheck.extendedResults">false</str>

    <str name="spellcheck.count">10</str>

  </lst>

  <arr name="last-components">

    <str>spellcheck</str>

  </arr>

</requestHandler>

<searchComponent name="suggest" class="solr.SpellCheckComponent">

  <str name="queryAnalyzerFieldType">string</str>

  <lst name="spellchecker">

    <str name="name">suggest</str>

    <str name="classname">org.apache.solr.spelling.suggest.Suggester</str>

    <str name="lookupImpl">org.apache.solr.spelling.suggest.tst.TSTLookup</str>

    <str name="field">text</str>

    <float name="threshold">0.0001</float>

    <str name="comparatorClass">freq</str>

    <str name="buildOnOptimize">true</str>

    <!--<str name="buildOnCommit">true</str>-->
```

```
</lst>

</searchComponent>

<requestHandler name="/suggest" class="solr.SearchHandler" startup="lazy">

  <lst name="defaults">

    <str name="spellcheck">true</str>

    <str name="spellcheck.dictionary">suggest</str>

    <str name="spellcheck.onlyMorePopular">true</str>

    <str name="spellcheck.extendedResults">false</str>

    <str name="spellcheck.count">10</str>

    <!--<str name="spellcheck.collate">true</str>-->

  </lst>

  <arr name="components">

    <str>suggest</str>

  </arr>

</requestHandler>

<requestHandler name="/mlt" class="solr.MoreLikeThisHandler">

</requestHandler>

<searchComponent name="tvComponent" class="solr.TermVectorComponent"/>

<requestHandler name="/tvrh" class="solr.SearchHandler" startup="lazy">

  <lst name="defaults">

    <str name="df">text</str>

    <bool name="tv">true</bool>

  </lst>

  <arr name="last-components">

    <str>tvComponent</str>

  </arr>

</requestHandler>

<searchComponent name="clustering"

  enable="${solr.clustering.enabled:true}"

  class="solr.clustering.ClusteringComponent" >

  <!-- Declare an engine -->

  <lst name="engine">

    <str name="name">default</str>

    <str name="carrot.algorithm">org.carrot2.clustering.lingo.LingoClusteringAlgorithm</str>

  <!-- Engine-specific parameters -->
```

```
<str name="LingoClusteringAlgorithm.desiredClusterCountBase">20</str>

</lst>

</searchComponent>

<requestHandler name="/clustering"

    startup="lazy"

    enable="{solr.clustering.enabled:true}"

    class="solr.SearchHandler">

<lst name="defaults">

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

<bool name="clustering">true</bool>

<str name="clustering.engine">default</str>

<bool name="clustering.results">true</bool>

<str name="carrot.title">category_s</str>

    <str name="carrot.snippet">content</str>

    <str name="carrot.produceSummary">true</str>

</lst>

<arr name="last-components">

    <str>clustering</str>

</arr>

</requestHandler>

<searchComponent name="terms" class="solr.TermsComponent"/>

<!-- A request handler for demonstrating the terms component -->

<requestHandler name="/terms" class="solr.SearchHandler" startup="lazy">

    <lst name="defaults">

        <bool name="terms">true</bool>

        <bool name="distrib">false</bool>

    </lst>

    <arr name="components">

        <str>terms</str>

    </arr>

</requestHandler>

<searchComponent name="elevator" class="solr.QueryElevationComponent" >

<!-- pick a fieldType to analyze queries -->

<str name="queryFieldType">string</str>

<str name="config-file">elevate.xml</str>

</searchComponent>

<!-- A request handler for demonstrating the elevator component -->

<requestHandler name="/elevate" class="solr.SearchHandler" startup="lazy">

    <lst name="defaults">
```



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

<str name="df">text</str>

</lst>

<arr name="last-components">

  <str>elevator</str>

</arr>

</requestHandler>

<searchComponent class="solr.HighlightComponent" name="highlight">

  <highlighting>

    <!-- Configure the standard fragmenter -->

    <!-- This could most likely be commented out in the "default" case -->

    <fragmenter name="gap"

      default="true"

      class="solr.highlight.GapFragmenter">

        <lst name="defaults">

          <int name="hl.fragsize">100</int>

        </lst>

      </fragmenter>

      <!-- A regular-expression-based fragmenter

        (for sentence extraction)

        -->

      <fragmenter name="regex"

        class="solr.highlight.RegexFragmenter">

          <lst name="defaults">

            <!-- slightly smaller fragsizes work better because of slop -->

            <int name="hl.fragsize">70</int>

            <!-- allow 50% slop on fragment sizes -->

            <float name="hl.regex.slop">0.5</float>

            <!-- a basic sentence pattern -->

            <str name="hl.regex.pattern">[-\w ,\n\&quot;&apos;]{20,200}</str>

          </lst>

        </fragmenter>

      <!-- Configure the standard formatter -->

      <formatter name="html"

        default="true"

        class="solr.highlight.HtmlFormatter">

          <lst name="defaults">

            <str name="hl.simple.pre"><![CDATA[<em>]]></str>
```

```
<str name="hl.simple.post"><![CDATA[</em>]]></str>

</lst>

</formatter>

<!-- Configure the standard encoder -->

<encoder name="html"

    class="solr.highlight.HtmlEncoder" />

<!-- Configure the standard fragListBuilder -->

<fragListBuilder name="simple"

    class="solr.highlight.SimpleFragListBuilder"/>

<!-- Configure the single fragListBuilder -->

<fragListBuilder name="single"

    class="solr.highlight.SingleFragListBuilder"/>

<!-- Configure the weighted fragListBuilder -->

<fragListBuilder name="weighted"

    default="true"

    class="solr.highlight.WeightedFragListBuilder"/>

<!-- default tag FragmentsBuilder -->

<fragmentsBuilder name="default"

    default="true"

    class="solr.highlight.ScoreOrderFragmentsBuilder">

</fragmentsBuilder>

<!-- multi-colored tag FragmentsBuilder -->

<fragmentsBuilder name="colored"

    class="solr.highlight.ScoreOrderFragmentsBuilder">

<lst name="defaults">

    <str name="hl.tag.pre"><![CDATA[

        <b style="background:yellow">,<b style="background:lawgreen">,

        <b style="background:aquamarine">,<b style="background:magenta">,

        <b style="background:palegreen">,<b style="background:coral">,

        <b style="background:wheat">,<b style="background:khaki">,

        <b style="background:lime">,<b style="background:deepskyblue">]]></str>

    <str name="hl.tag.post"><![CDATA[</b>]]></str>

</lst>

</fragmentsBuilder>
```

```
<boundaryScanner name="default"

    default="true"

    class="solr.highlight.SimpleBoundaryScanner">

<lst name="defaults">

    <str name="hl.bs.maxScan">10</str>

    <str name="hl.bs.chars">.,!? &#9;&#10;&#13;</str>

</lst>

</boundaryScanner>

<boundaryScanner name="breakIterator"

    class="solr.highlight.BreakIteratorBoundaryScanner">

<lst name="defaults">

    <!-- type should be one of CHARACTER, WORD(default), LINE and SENTENCE -->

    <str name="hl.bs.type">WORD</str>

    <!-- language and country are used when constructing Locale object. -->

    <!-- And the Locale object will be used when getting instance of BreakIterator -->

    <str name="hl.bs.language">en</str>

    <str name="hl.bs.country">US</str>

</lst>

</boundaryScanner>

</highlighting>

</searchComponent>

<queryResponseWriter name="json" class="solr.JSONResponseWriter">

    <str name="content-type">text/plain; charset=UTF-8</str>

</queryResponseWriter>

<!--

Custom response writers can be declared as needed...

-->

<queryResponseWriter name="velocity" class="solr.VelocityResponseWriter" startup="lazy"/>

<queryResponseWriter name="xslt" class="solr.XSLTResponseWriter">

    <int name="xsltCacheLifetimeSeconds">5</int>

</queryResponseWriter>

<admin>

<defaultQuery>*</defaultQuery>
```

```
</admin>

</config>
```

参考资料及文献

<http://wiki.apache.org/solr/> 所有的配置在这里都有说明,按需要配上就行了.

顶 33
踩 1

上一篇 超棒的环形菜单效果jQuery插件
下一篇 jsp去除空白行

相关文章推荐

- [Apache Solr入门教程\(初学者之旅\)](#)
- [Solr基础教程之环境搭建（一）](#)
- [solr 6.0 没有schema.xml未自动创建schema文件](#)
- [solr入门教程](#)
- [solr入门教程](#)

- [solr入门教程](#)
- [solr完整教程](#)
- [【solr专题之一】Solr快速入门](#)
- [JAVA的solr操作实现（基本操作）](#)
- [SolrCloud使用教程、原理介绍](#)



学生管理系统



在职研究生取消



西班牙买房移民



开发一个app多少



美国产子

猜你在找

- 【直播】机器学习&数据挖掘7周实训--韦玮
 - 【直播】3小时掌握Docker最佳实战-徐西宁
 - 【直播】计算机视觉原理及实战-屈教授
 - 【直播】机器学习之矩阵-黄博士
 - 【直播】机器学习之凸优化-马博士
- 【套餐】系统集成项目管理工程师顺利通关--徐朋
 - 【套餐】机器学习系列套餐（算法+实战）--唐宇迪
 - 【套餐】微信订阅号+服务号Java版 v2.0--翟东平
 - 【套餐】微信订阅号+服务号Java版 v2.0--翟东平
 - 【套餐】Javascript 设计模式实战--曾亮

查看评论

25楼 韦文文 2017-07-17 21:39发表

好详细，再来看



24楼 雪叮头 2017-04-22 20:13发表



非常详细，还会再来拜读。

23楼 脚踏上深圳的土地 2017-03-26 18:14发表



不错，适合初学者，只不过有些地方讲解不是很通顺。但总体上比较不错，相对比较齐全，虽然博文有点长，但是没关系，毕竟内容较多。非常感谢博主的贡献，我要收藏

22楼 wlyzt520 2016-11-22 16:02发表



Exception while processing: tbl document : SolrInputDocument(fields: [])org.apache.solr.handler.dataimport.DataImportHandlerException: java.sql.SQLException: Oper
麻烦问一下这个是什么问题？碰到过没有？在logging一直刷新显示

21楼 uchenxy 2016-10-26 16:24发表



并不适合初学者看

20楼 zcl_love_wx 2016-07-19 16:12发表



不能收藏，真恶心

19楼 yuxm909 2016-04-16 17:16发表



6.5检索建议，这部分，查询的url里没有指定要查询的字段，为什么能出结果。。。

18楼 nice_word 2016-04-11 22:15发表



[plain]

```
01. <fieldType name="string" class="solr.StrField" sortMissingLast="true"/>
02.
03. <fieldType name="boolean" class="solr.BoolField" sortMissingLast="true"/>
04.
05. <fieldType name="int" class="solr.TrieIntField" precisionStep="0" positionIncrementGap="0"/>
```

像这样就好了，这样看着有点累 orz

17楼 风尘中国 2016-03-23 14:42发表



这一篇关于solr的高亮部分的代码并不完整，无法使用，其他部分还是很赞，感谢楼主分享

16楼 jin19910624 2016-03-14 11:05发表



文章不错 对于不会英语的来说 通俗易懂 虽然 常了带你

15楼 猫神哩哩哩 2016-02-18 09:57发表



大神，写的真好

14楼 baeiou 2015-10-21 19:57发表



感谢楼主！

13楼 nedynkd 2015-08-24 12:04发表



solrcloud5.2.1+zkookeeper一部精通

讲师：夜行侠

课时数：14课时

课程介绍：<http://www.xuetuwuyou.com/course/15>

课程下载地址：链接：<http://pan.baidu.com/s/1g9FMu> 密码: h4kc

课程介绍

一、课程使用到的软件及版本： centos系统， solr5.2.1， zookeeper 3.4.6

二、课程大纲

- 1、zookeeper集群搭建，以及配置信息，zookeeper客户端命令讲解
- 2、solrcloud在生产环境下的搭建
- 3、创建collection，如何动态加载配置到zookeeper中，以及schema的详解
- 4、详细讲解文档的Fields以及solrconfig.xml的相关配置，以及solrweb管理界面描述
- 5、solr索引文件夹，uuid与uniqueKey的讲解，如何使用uuid自动生成uniqueKey
- 6、solr的dataimport讲解各种DataSource，导入mysql数据
- 7、分词器讲解，中文分词器ik。
- 8、solr定时更新mysql的新增数据
- 9、solr源码从ant如何转换为maven，以及源码的阅读，打包
- 10、修改ik分词器源码，动态从mysql中获取自定义分词，停顿词
- 11、solrj与spring整合
- 12、spring solrj的简单增删改查，权重设置
- 13、spring solrj的Faceting搜索Highlighting高亮

14、solr的近实时索引搜索以及实时索引搜索


Re: QMCoder 2017-04-25 22:34 发表

 回复nedynkd: 可以给我发一份吗？谢谢
563809169@qq.com


Re: qq_34454383 2017-02-22 14:55 发表

 回复nedynkd: 31574965@qq.com 非常感谢您也能给我发一份教程 网盘看不下了

Re: 云飞 2015-11-05 14:27 发表

 回复nedynkd: 有没有 solr与spring整合 给我私信分享下 谢谢。

12楼 我就是根号三 2015-08-10 16:25 发表

 感谢楼主的分享，受教了。

11楼 哔了狗 2015-06-05 10:39 发表

 太棒了

10楼 qq_26343409 2015-03-05 17:41 发表

 4242

9楼 qq_26343409 2015-03-05 17:41 发表

 24242

8楼 qq_26343409 2015-03-05 17:41 发表

 24242

7楼 qq_26343409 2015-03-05 17:40 发表

 24242


6楼 qq_26343409 2015-03-05 17:41 发表

 24244


5楼 qq_26343409 2015-03-05 17:41 发表

 2342424


4楼 qq_26343409 2015-03-05 17:39 发表

 *以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

3楼 liang330965445 2014-10-25 10:00 发表

 博主你好，4.7.2版本的solr如何配置synonyms 啊。我按照网上说的方法配置总是报错，新版本取消了IKAnalyzerSolrFactory类。。。

2楼 nnn9223643 2014-07-14 14:48 发表

 我觉得比较到位，但是有些小细节

1楼 super2007 2014-07-04 09:57 发表

 受教了。老师可以写本书出来。

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场