链滴

Solr 8.1.1 Authorization、Core、HanLP、Searching、Commits、NRT ...

作者：14032

原文链接：https://ld246.com/article/1561546264848

来源网站：链滴

本文 using Solr in standalone mode

intro

- 基本搭建部署
- 认证、授权
- Core 创建配置
- Data Import Handler
- managed-schema 配置
- HanLP 自然语言分词处理
- Standard Query Parse 语法
- Suggester 智能提醒组件
- NRT 近实时搜索概念
- Soft 提交、Hard 提交
- DirectoryFactor
- indexConfig
- Segments
- Cache
- Java client SolrJ

start、stop、restart

```
# using Solr in standalone mode
# help
```

原文链接：Solr 8.1.1 Authorization、Core、HanLP、Searching、Commits、NRT ...

```
./solr -help
./solr start -help
./solr stop -help

# start、stop、restart（root用户启动Solr需加 -force）
./solr start -p 8983 -force
./solr restart -p 8983 -force
./solr stop -p 8983
./solr stop -all
```

Authorization Plugins

Solr支持Basic Authentication、JWT Authentication等多种认证， Solr Ref Guide 8.1-securing-solr.html

- Enable Basic Authentication

在/usr/solr-8.1.1/server/solr文件夹下创建文件security.json，内容如下

```
{
  "authentication":{
    "blockUnknown":  true,
    "class": "solr.BasicAuthPlugin",
    "credentials": {"solr":"IV0EHq1OnNrj6gvR....."},
    "realm": "My Solr users",
    "forwardCredentials": false
  },
  "authorization":{
    "class": "solr.RuleBasedAuthorizationPlugin",
    "permissions": [{"name":"security-edit", "role":"admin"}],
    "user-role": {"solr":"admin"}
  }
}
```

1. 启用基本身份验证和基于规则的授权插件。
2. 参数 blockUnknown: true 表示不允许未经身份验证的请求通过。
3. 已定义了一个名为 solr 的用户，默认密码为 SolrRocks。
4. admin 角色已定义，并且具有编辑安全设置的权限。
5. solr 用户已被定义为 admin 角色。

Create Core

1. 通过 Admin UI 界面创建。创建 Core时必须能够找到以下配置，否则将创建失败；

- instanceDir 必须已经存在
- instanceDir 必须包含一个 conf 文件夹
- conf 文件夹下必须包含 solrconfig.xml 和 managed-schema
- 创建成功后，在 /instanceDir 下将会创建 core.properties

```
# core.properties
```

```
name=blog
config=./conf/solrconfig.xml
schema=./conf/managed-schema
dataDir=./data
```

2. 调用 CoreAdmin API 创建。

```
GET /solr/admin/cores?action=CREATE&name=blog&instanceDir=/usr/solr-8.1.1/server/blo
&config=solrconfig.xml&dataDir=data HTTP/1.1
Host: 127.0.0.1:8983
Authorization: Basic c29scjpTb2xyUm9ja3M=
Postman-Token: 66fcf8ec-aa89-4ba6-98f3-c36339d71a84,dbbf0e06-62b5-4994-a160-0feb4fc
fcc4
```

- link coreadmin-api
- link Configsets in Standalone Mode

Managed-schema

1. 修改 uniqueKey

```
<uniqueKey>id</uniqueKey>
```

2. 加入索引字段

```
<uniqueKey>id</uniqueKey>
#field
<field name="id" type="string" multiValued="false" indexed="true" required="true" stored=
true"/>
<field name="article" type="text_cn" multiValued="true" indexed="true" stored="true"/>
<field name="authorName" type="string" indexed="true" stored="true"/>
<field name="language" type="string" indexed="true" stored="false"/>
<field name="mobileUrl" type="string" indexed="true" stored="true"/>
<field name="publishDate" type="tdate" indexed="true" stored="true" default="NOW+8HO
R"/>
<field name="siteName" type="string" indexed="true" stored="true"/>
<field name="title" type="text_cn" indexed="true" stored="true"/>
<field name="summary" type="text_cn" indexed="true" stored="true"/>
<field name="summaryAuto" type="text_cn" indexed="true" stored="true"/>
<field name="url" type="string" indexed="true" stored="true"/>
#copyField
<copyField source="authorName" dest="article"/>
<copyField source="siteName" dest="article"/>
<copyField source="title" dest="article"/>
<copyField source="summary" dest="article"/>
```

HanLP or IKAnalyzer

1. **HanLP 中文分词（推荐）**

- 集成 HanLP 分词 full-text-retrieval-solr-integrated-hanlp-chinese-word-segmentation.h
ml

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

```
  <analyzer type="index">
    <tokenizer class="com.hankcs.lucene.HanLPTokenizerFactory" enableIndexMode="true"/>
    <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
   <!-- 切记不要在query中开启index模式 -->
   <tokenizer class="com.hankcs.lucene.HanLPTokenizerFactory" enableIndexMode="false"/>
     <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
     <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" ex
and="true"/>
     <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

2. IKAnalyzer

- IKAnalyzer （IKAnalyzer2012_u6.jar很古老了cold_sweat），已经不兼
后续新版本
- GitHub https://github.com/magese/ik-analyzer-solr 支持solr 7&8；
- 将 ik-analyzer-8.1.0.jar 放入/usr/solr-8.1.1/server/solr-webapp/webapp/WEB-INF/lib
- 配置 managed-schema，添加ik分词器，还有同义词、停止词等配置

```
<!-- ik分词器 -->
<fieldType name="text_ik" class="solr.TextField">
  <analyzer type="index">
    <tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" conf="ik.conf" useSmart="
alse"/>
     <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
     <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="org.wltea.analyzer.lucene.IKTokenizerFactory" conf="ik.conf" useSmart="
rue"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
    <filter class="solr.SynonymFilterFactory" expand="true" ignoreCase="true" synonyms="sy
onyms.txt"/>
    <filter class="solr.LowerCaseFilterFactory"/>
  </analyzer>
</fieldType>
```

Data Import Handler

1. 在 /usr/solr-8.1.1/server/solr/blog/conf下创建data-config.xml文件；内容如下：

```
<dataConfig>
  <dataSource type="JdbcDataSource" driver="com.mysql.jdbc.Driver"
          url="jdbc:mysql://127.0.0.1:3306/solr"
          user="solr" password="xxoo" batchSize="-1" />
  <document>
    <entity name="article" pk="id" transformer="DateFormatTransformer"
      query="SELECT * from readhub"
```

```
      deltaImportQuery="SELECT * from readhub where oId='${dataimporter.delta.id}'"
      deltaQuery="SELECT id FROM readhub where date_format(publishDate,'yyyy-MM-dd HH
mi:ss') > '${dataimporter.last_index_time}'"  >
      <field column="id" name="id" />
      <field column="authorName" name="authorName" />
      <field column="language" name="language" />
      <field column="mobileUrl" name="mobileUrl" />
      <field column="publishDate" name="publishDate" />
      <field column="siteName" name="siteName" />
      <field column="title" name="title" />
      <field column="summary" name="summary" />
      <field column="summaryAuto" name="summaryAuto" />
      <field column="url" name="url" />
    </entity>
 </document>
 <propertyWriter dateFormat="yyyy-MM-dd HH:mm:ss" type="SimplePropertiesWriter" />
</dataConfig>
```

2. 数据库密码加密 link encrypting a-database-password

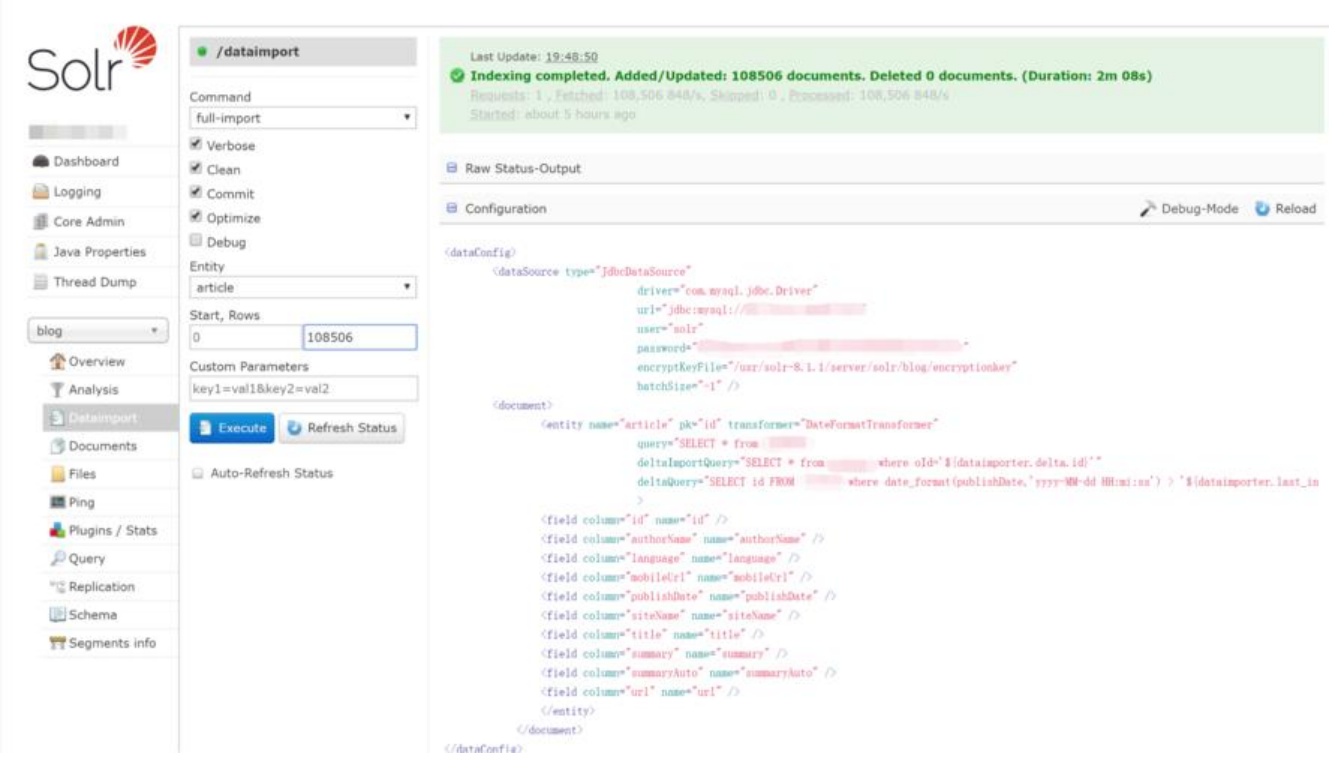
在/usr/solr-8.1.1/server/solr/blog/下创建encryptionkey文件，并写入明文密码，通过命令获取加密码：

```
echo -n "my-jdbc-password" | openssl enc -aes-128-cbc -a -salt -md md5 -pass  file:/usr/solr
8.1.1/server/solr/blog/encryptionkey

# 将上面输出的字符写入`data-config.xml password属性`配置，修改如下
<dataSource type="JdbcDataSource"
  driver="com.mysql.jdbc.Driver"
  url="jdbc:mysql://127.0.0.1:3306/solr"
  user="solr"
  password="U2FsdGVkXXXXXXXXXXXXXXXXXXX="
  encryptKeyFile="/usr/solr-8.1.1/server/solr/blog/encryptionkey"
  batchSize="-1" />
```

3. 配置 DataImportHandler 在/usr/solr-8.1.1/server/solr/blog/conf/solrconfig.xml，加入如配置

```
<lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-dataimporthandler-.*\.jar" />
<requestHandler name="/dataimport" class="org.apache.solr.handler.dataimport.DataImpor
Handler">
  <lst name="defaults">
    <str name="config">data-config.xml</str>
  </lst>
</requestHandler>
```

Uploading Structured Data



CoreAdmin API

- /solr/admin/cores?action=STATUS&core=blog
- /solr/admin/cores?action=CREATE&name=blog&instanceDir=/usr/solr-8.1.1/server/blog&onfig=solrconfig.xml&dataDir=data
- /solr/admin/cores?action=RELOAD&core=blog
- /solr/admin/cores?action=SWAP&core=core-name&other=other-core-name

Searching

默认会使用Standard Query Parser即defType=lucene

模糊查询 sol~0.5 (相似度数值)
邻近查询 "solr apache" ~ 5
范围查询 date：[20190810 TO 20190910] 中括号[]表示包含边界，花括号{}表示排除边界
权重值查询 Solr^4 apache
Boolean操作符 AND、+、OR、NOT、-
子查询表达式（Solr OR Apache) AND Jetty

1. Standard Query Parser 8_1/the-standard-query-parser.html

   - q
   - fq
   - sort
   - fl
   - df
   - start, rows

- q.op

2. Highlighting 8_1/highlighting.html
    - hl
    - hl.fl
    - hl.fragsize
    - hl.snippets
    - hl.tag.pre
    - hl.tag.post
3. Faceting 8_1/faceting.html
    - facet
    - facet.query
    - facet.field
    - facet.prefix
    - facet.sort
    - facet.limit
    - facet.mincount
    - facet.threads
4. Grouping 8_1/result-grouping.html
    - group
    - group.field
    - group.query
    - group.sort

更多请查看官方文档 /solr/guide/8_1/searching.html

Suggester

```
<requestHandler name="/suggest" class="solr.SearchHandler" startup="lazy">
  <lst name="defaults">
    <str name="suggest">true</str>
    <str name="suggest.count">10</str>
    <str name="suggest.dictionary">titleSuggester</str>
  </lst>
  <arr name="components">
    <str>suggest</str>
  </arr>
</requestHandler>

<searchComponent name="suggest" class="solr.SuggestComponent">
  <lst name="suggester">
    <str name="name">titleSuggester</str>
```

```
    <str name="lookupImpl">FuzzyLookupFactory</str>
    <str name="dictionaryImpl">DocumentDictionaryFactory</str>
    <str name="field">title</str>
    <str name="weightField">title</str>
    <str name="suggestAnalyzerFieldType">text_ik</str>
    <str name="buildOnStartup">true</str>
  </lst>
</searchComponent>

localhost:8983/solr/blog/suggest?suggest=true&suggest.build=false&suggest.dictionary=sit
NameSuggester&suggest.q=新浪

# Response
{
  "responseHeader":{
    "status":0,
    "QTime":93},
  "suggest":{"siteNameSuggester":{
      "新浪":{
        "numFound":2,
        "suggestions":[{
            "term":"新浪",
            "weight":0,
            "payload":""},
          {
            "term":"新浪科技",
            "weight":0,
            "payload":""}]}}}}
```

Near Real Time

link near-real-time-searching

Document durability and searchability are controlled by commits. The "Near" in "Near Real T me" is configurable to meet the needs of your application. Commits are either "hard" or "soft" and can be issued by a client (say SolrJ), via a REST call or configured to occur automatically in olrconfig.xml. The recommendation usually gives is to configure your commit strategy in solr onfig.xml (see below) and avoid issuing commits externally.

Typically in NRT applications, hard commits are configured with openSearcher=false, and soft commits are configured to make documents visible for search.

- hard commit

A hard commitcallsfsync on the index files to ensure they have been flushed to stable storage. The current transaction log is closed and a new one is opened. See the "transaction log" discu sion below for how data is recovered in the absence of a hard commit. Optionally a hard co mit can also make documents visible for search, but this is not recommended for NRT searchi g as it is more expensive than a soft commit.

- soft commit

A soft commitis faster since it only makes index changes visible and does notfsync index files, start a new segment or start a new transaction log. Search collections that have NRT require ents will want to soft commit often enough to satisfy the visibility requirements of the applica ion. A softCommit may be "less expensive" than a hard commit (openSearcher=true), but it is

not free. It is recommended that this be set for as long as is reasonable given the application equirements.

Commits

- commit and softCommit
- autoCommit
- commitWithin

在autoSoftCommit 不启用（maxTime = -1）情况下，autoCommit 的 openSearcher 若为 false 则需要 reload core api 之后才能看到 update/add index，可见 autoSoftCommit 带有 openSearcher 功能。

openSearcher如果设置为false，提交会使最近的索引更新写入到索引目录，但不会创建一个新的IndexSearcher 实例。创建一个新的IndexSearcher实例使那些最近更新的索引数据立即

可见。

```
<updateHandler class="solr.DirectUpdateHandler2">
  <updateLog>
     <str name="dir">${solr.ulog.dir:}</str>
     <int name="numVersionBuckets">${solr.ulog.numVersionBuckets:65536}</int>
  </updateLog>
  # hard commit
  <autoCommit>
    <maxDocs>10000</maxDocs>
    <maxTime>30000</maxTime>
    <maxSize>512m</maxSize>
    # 自动硬提交完成后是否开启一个新的 `IndexSearcher` 实例
    <openSearcher>false</openSearcher>
  </autoCommit>

   <!-- softAutoCommit is like autoCommit except it causes a
  'soft' commit which only ensures that changes are visible
  but does not ensure that data is synced to disk.  This is
  faster and more near-realtime friendly than a hard commit.
  maxTime = -1 不执行 softCommit
  -->
  # soft commit
  <autoSoftCommit>
     <maxTime>3000</maxTime>
  </autoSoftCommit>

  <!-- With this configuration, when you call `commitWithin` as part of your update message,
  it will automatically perform a hard commit every time. -->
  # commitWithin
  <commitWithin>
    <softCommit>false</softCommit>
  </commitWithin>
</updateHandler>
```

DirectoryFactor

- solr.NRTCachingDirectoryFactory（default）

它包装了StandardDirectoryFactory并在内存中缓存了一些索引文件为了提升近实时搜索性能。

- solr.MMapDirectoryFactory
- solr.NIOFSDirectoryFactory
- solr.SimpleFSDirectoryFactory
- solr.RAMDirectoryFactory
- solr.HdfsDirectoryFactory - Running Solr on HDFS

```
<!-- The DirectoryFactory to use for indexes.

     solr.StandardDirectoryFactory is filesystem
     based and tries to pick the best implementation for the current
     JVM and platform.  solr.NRTCachingDirectoryFactory, the default,
     wraps solr.StandardDirectoryFactory and caches small files in memory
     for better NRT performance.

     One can force a particular implementation via solr.MMapDirectoryFactory,
     solr.NIOFSDirectoryFactory, or solr.SimpleFSDirectoryFactory.

     solr.RAMDirectoryFactory is memory based and not persistent.
-->
<directoryFactory  name="DirectoryFactory"  class="${solr.directoryFactory:solr.NRTCachingD
rectoryFactory}"/>
```

indexConfig

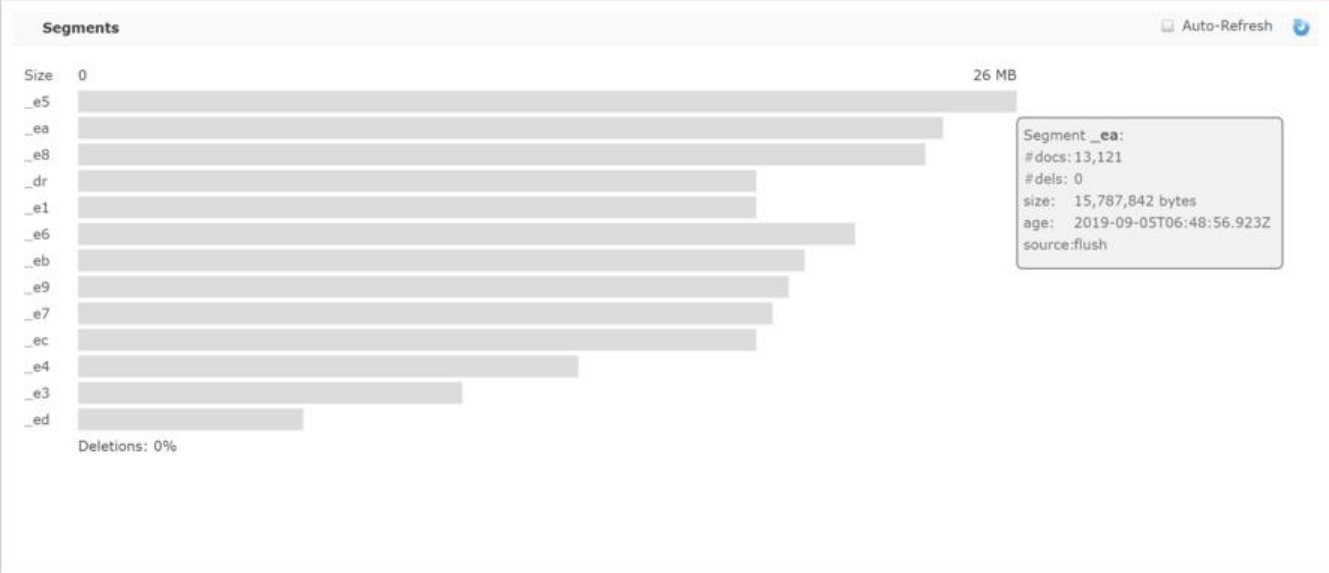
```
<!-- ramBufferSizeMB sets the amount of RAM that may be used by Lucene
     indexing for buffering added documents and deletions before they are
     flushed to the Directory.
     maxBufferedDocs sets a limit on the number of documents buffered
     before flushing.
     If both ramBufferSizeMB and maxBufferedDocs is set, then
     Lucene will flush based on whichever limit is hit first.  -->

# Writing New Segments
<ramBufferSizeMB>100</ramBufferSizeMB>
<maxBufferedDocs>1000</maxBufferedDocs>
<useCompoundFile>false</useCompoundFile>

<!-- 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.
-->

# Merging Index Segments 索引的段合并策略
<mergePolicyFactory class="org.apache.solr.index.TieredMergePolicyFactory">
  # 一次最多合并多少个段文件
  <int name="maxMergeAtOnce">10</int>
```

```
  <int name="segmentsPerTier">10</int>
</mergePolicyFactory>
```

Segments

Lucene 内部数据写入会产生很多 Segment，查询时会对多个 Segment 查询并合并结果。所以 Segment 的数量一定程度上会影响查询的效率，所以需要对 Segment 进行合并，合并的过程就称为 Merge 而何时触发 Merge 由 MergePolicy 决定。

Merge 是对 Segment文件合并的动作，合并的好处是能够提高查询的效率以及回收一些被删除的文档



Cache

```
<!-- 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.
        maxRamMB - the maximum amount of RAM (in MB) that this cache is allowed
                to occupy. Note that when this option is specified, the size
                and initialSize parameters are ignored.
    -->
<filterCache  class="solr.FastLRUCache"  size="512"  initialSize="512"  autowarmCount="0"/>
```

```
# Query Result Cache
<queryResultCache class="solr.LRUCache" size="512" initialSize="512" autowarmCount="0
/>

# Document Cache
<documentCache class="solr.LRUCache" size="512" initialSize="512" autowarmCount="0"
>

# 在一次查询中缓存的 Documentid 最大个数
<queryResultWindowSize>20</queryResultWindowSize>

# 查询结果集缓存中能够缓存的 Document 最大个数
<queryResultMaxDocsCached>200</queryResultMaxDocsCached>
```

Java Using SolrJ

```
# Per-Request Basic Auth Credentials
# Use SolrJ
QueryRequest req = new QueryRequest(new SolrQuery("*:*"));
req.setBasicAuthCredentials(userName, password);
QueryResponse rsp = req.process(solrClient);
```

- https://lucene.apache.org/solr/guide/8_1/using-solrj.html

link

- Lucene 查询原理及解析
- Elasticsearch内核剖析
- Elasticsearch技术研讨
- 苏宁 11.11：搜索引擎 Solr 在苏宁易购商品评价系统中的应用
- 每秒20W次并发分词检索，架构如何设计?