hybris 5.6 with SolrCloud 5.2.1

Run hybris 5.6 on the latest SolrCloud 5.2.1.

About this document

Audience: Technical consultants

Based on: hybris 5.6 / SolrCloud 5.2.1

Related to:

Page Views: 43

See also

1. Major+Changes+from+Solr+4+to+Solr+5
2. https://cwiki.apache.org/confluence/display/solr/SolrCloud

Purpose:

1) leverage SolrCloud for scalability and high availability.

2) migrate solr from 4.6.1 to 5.2.1

> **Warning**
> Don't try this on production server if you are not willing to try cutting edge technology :)

Approach:

1) hybris 5.6 with solr client 4.6.1 --> SolrCloud 5.2.1 (4.6.1 solr client connects to solr server 5.2.1)

2) hybris 5.6 solrfacetearch upgrade to 5.2.1 as embeded or standalone (both client and server are upgraded to 5.2.1)

Implementation:

1) solrfacetsearch customization

- code changes
- solr.xml

solr.xml Expand source

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
 Licensed to the Apache Software Foundation (ASF) under one or more
 contributor license agreements.  See the NOTICE file distributed with
 this work for additional information regarding copyright ownership.
 The ASF licenses this file to You under the Apache License, Version 2.0
 (the "License"); you may not use this file except in compliance with
 the License.  You may obtain a copy of the License at
     http://www.apache.org/licenses/LICENSE-2.0
 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.
-->
<!--
   This is an example of a simple "solr.xml" file for configuring one or
   more Solr Cores, as well as allowing Cores to be added, removed, and
   reloaded via HTTP requests.
   More information about options available in this configuration file,
   and Solr Core administration can be found online:
   http://wiki.apache.org/solr/CoreAdmin
-->
<solr>
  <str
name="adminHandler">${adminHandler:org.apache.solr.handler.admin.CoreAdminHandler
}</str>
  <str name="sharedLib">${sharedLib:hybris}</str>
  <solrcloud>
    <str name="host">${host:}</str>
    <int name="hostPort">${jetty.port:8983}</int>
    <str name="hostContext">${hostContext:solr}</str>
    <bool name="genericCoreNodeNames">${genericCoreNodeNames:true}</bool>
    <int name="zkClientTimeout">${zkClientTimeout:30000}</int>
    <int name="distribUpdateSoTimeout">${distribUpdateSoTimeout:600000}</int>
    <int name="distribUpdateConnTimeout">${distribUpdateConnTimeout:60000}</int>
  </solrcloud>
  <shardHandlerFactory name="shardHandlerFactory"
    class="HttpShardHandlerFactory">
    <int name="socketTimeout">${socketTimeout:600000}</int>
    <int name="connTimeout">${connTimeout:60000}</int>
  </shardHandlerFactory>
</solr>
```

- solrconfig.xml

solrconfig.xml Expand source

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
 Licensed to the Apache Software Foundation (ASF) under one or more
 contributor license agreements.  See the NOTICE file distributed with
```

```
 this work for additional information regarding copyright ownership.
 The ASF licenses this file to You under the Apache License, Version 2.0
 (the "License"); you may not use this file except in compliance with
 the License.  You may obtain a copy of the License at
     http://www.apache.org/licenses/LICENSE-2.0
 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.
-->
<!--
     For more details about configurations options that may appear in
     this file, see http://wiki.apache.org/solr/SolrConfigXml.
-->
<config>
  <!-- In all configuration below, a prefix of "solr." for class names
       is an alias that causes solr to search appropriate packages,
       including org.apache.solr.(search|update|request|core|analysis)
       You may also specify a fully qualified Java classname if you
       have your own custom plugins.
    -->
  <!-- Controls what version of Lucene various components of Solr
       adhere to.  Generally, you want to use the latest version to
       get all bug fixes and improvements. It is highly recommended
       that you fully re-index after changing this setting as it can
       affect both how text is indexed and queried.
  -->
  <luceneMatchVersion>5.2.1</luceneMatchVersion>
  <!-- <lib/> directives can be used to instruct Solr to load any Jars
       identified and use them to resolve any "plugins" specified in
       your solrconfig.xml or schema.xml (ie: Analyzers, Request
       Handlers, etc...).
       All directories and paths are resolved relative to the
       instanceDir.
       Please note that <lib/> directives are processed in the order
       that they appear in your solrconfig.xml file, and are "stacked"
       on top of each other when building a ClassLoader - so if you have
       plugin jars with dependencies on other jars, the "lower level"
       dependency jars should be loaded first.
       If a "./lib" directory exists in your instanceDir, all files
       found in it are included as if you had used the following
       syntax...

              <lib dir="./lib" />
    -->
  <!-- A 'dir' option by itself adds any files found in the directory
       to the classpath, this is useful for including all jars in a
       directory.
       When a 'regex' is specified in addition to a 'dir', only the
       files in that directory which completely match the regex
       (anchored on both ends) will be included.
       If a 'dir' option (with or without a regex) is used and nothing
       is found that matches, a warning will be logged.
       The examples below can be used to load some solr-contribs along
       with their external dependencies.
    -->
  <lib dir="${solr.install.dir:../../../..}/contrib/extraction/lib"
regex=".*\.jar" />
```

```
  <lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-cell-\d.*\.jar" />
  <lib dir="${solr.install.dir:../../../..}/contrib/clustering/lib/"
regex=".*\.jar" />
  <lib dir="${solr.install.dir:../../../..}/dist/"
regex="solr-clustering-\d.*\.jar" />
  <lib dir="${solr.install.dir:../../../..}/contrib/langid/lib/" regex=".*\.jar"
/>
  <lib dir="${solr.install.dir:../../../..}/dist/" regex="solr-langid-\d.*\.jar"
/>
  <lib dir="${solr.install.dir:../../../..}/contrib/velocity/lib" regex=".*\.jar"
/>
  <lib dir="${solr.install.dir:../../../..}/dist/"
regex="solr-velocity-\d.*\.jar" />


  <!-- an exact 'path' can be used instead of a 'dir' to specify a
       specific jar file.  This will cause a serious error to be logged
       if it can't be loaded.
    -->
  <!--
     <lib path="../a-jar-that-does-not-exist.jar" />
  -->

  <!-- Data Directory
       Used to specify an alternate directory to hold all index data
       other than the default ./data under the Solr home.  If
       replication is in use, this should match the replication
       configuration.
    -->
  <dataDir>${solr.data.dir:./data}/${solr.core.name}</dataDir>

  <!-- 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, not
       persistent, and doesn't work with replication.
    -->
  <directoryFactory name="DirectoryFactory"

class="${solr.directoryFactory:solr.NRTCachingDirectoryFactory}"/>
  <!-- The CodecFactory for defining the format of the inverted index.
       The default implementation is SchemaCodecFactory, which is the official
Lucene
       index format, but hooks into the schema to provide per-field customization
of
       the postings lists and per-document values in the fieldType element
       (postingsFormat/docValuesFormat). Note that most of the alternative
implementations
       are experimental, so if you choose to customize the index format, it's a
good
       idea to convert back to the official format e.g. via
IndexWriter.addIndexes(IndexReader)
       before upgrading to a newer version to avoid unnecessary reindexing.
```

```
  -->
  <codecFactory class="solr.SchemaCodecFactory"/>
  <!-- To enable dynamic schema REST APIs, use the following for <schemaFactory>:

       <schemaFactory class="ManagedIndexSchemaFactory">
         <bool name="mutable">true</bool>
         <str name="managedSchemaResourceName">managed-schema</str>
       </schemaFactory>

       When ManagedIndexSchemaFactory is specified, Solr will load the schema
from
       the resource named in 'managedSchemaResourceName', rather than from
schema.xml.
       Note that the managed schema resource CANNOT be named schema.xml.  If the
managed
       schema does not exist, Solr will create it after reading schema.xml, then
rename
       'schema.xml' to 'schema.xml.bak'.

       Do NOT hand edit the managed schema - external modifications will be
ignored and
       overwritten as a result of schema modification REST API calls.
       When ManagedIndexSchemaFactory is specified with mutable = true, schema
       modification REST API calls will be allowed; otherwise, error responses
will be
       sent back for these requests.
  -->
  <schemaFactory class="ClassicIndexSchemaFactory"/>
  <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       Index Config - These settings control low-level behavior of indexing
       Most example settings here show the default value, but are commented
       out, to more easily see where customizations have been made.

       Note: This replaces <indexDefaults> and <mainIndex> from older versions
       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
  <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.
```

```
     The default is 100 MB.  -->
<!-- <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">10</int>
      <int name="segmentsPerTier">10</int>
      <double name="noCFSRatio">0.1</double>
    </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>10</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>
    -->
  <!-- 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 info stream to solr's log. By default,
       this is enabled here, and controlled through log4j.properties.
    -->
   <infoStream>true</infoStream>
</indexConfig>

<!-- JMX

     This example enables JMX if and only if an existing MBeanServer
     is found, use this if you want to configure JMX through JVM
     parameters. Remove this to disable exposing Solr configuration
     and statistics to JMX.
     For more details see http://wiki.apache.org/solr/SolrJmx
  -->
<jmx />
<!-- If you want to connect to a particular server, specify the
     agentId
  -->
<!-- <jmx agentId="myAgent" /> -->
<!-- If you want to start a new MBeanServer, specify the serviceUrl -->
<!-- <jmx serviceUrl="service:jmx:rmi:///jndi/rmi://localhost:9999/solr"/>
```

```
  -->
<!-- The default high-performance update handler -->
<updateHandler class="solr.DirectUpdateHandler2">
  <!-- Enables a transaction log, used for real-time get, durability, and
       and solr cloud replica recovery.  The log can grow as big as
       uncommitted changes to the index, so use of a hard autoCommit
       is recommended (see below).
       "dir" - the target directory for transaction logs, defaults to the
              solr data directory.
       "numVersionBuckets" - sets the number of buckets used to keep
              track of max version values when checking for re-ordered
              updates; increase this value to reduce the cost of
              synchronizing access to version buckets during high-volume
              indexing, this requires 8 bytes (long) * numVersionBuckets
              of heap space per Solr core.
  -->
  <updateLog>
    <str name="dir">${solr.ulog.dir:}</str>
    <int name="numVersionBuckets">${solr.ulog.numVersionBuckets:65536}</int>
  </updateLog>

  <!-- AutoCommit
       Perform a hard commit automatically under certain conditions.
       Instead of enabling autoCommit, consider using "commitWithin"
       when adding documents.
       http://wiki.apache.org/solr/UpdateXmlMessages
       maxDocs - Maximum number of documents to add since the last
                 commit before automatically triggering a new commit.
       maxTime - Maximum amount of time in ms that is allowed to pass
                 since a document was added before automatically
                 triggering a new commit.
       openSearcher - if false, the commit causes recent index changes
         to be flushed to stable storage, but does not cause a new
         searcher to be opened to make those changes visible.
       If the updateLog is enabled, then it's highly recommended to
       have some sort of hard autoCommit to limit the log size.
    -->
   <autoCommit>
     <maxTime>${solr.autoCommit.maxTime:15000}</maxTime>
     <openSearcher>false</openSearcher>
   </autoCommit>
  <!-- softAutoCommit is like autoCommit except it causes a
       'soft' commit which only ensures that changes are visible
       but does not ensure that data is synced to disk.  This is
       faster and more near-realtime friendly than a hard commit.
    -->
   <autoSoftCommit>
     <maxTime>${solr.autoSoftCommit.maxTime:-1}</maxTime>
   </autoSoftCommit>
  <!-- Update Related Event Listeners

       Various IndexWriter related events can trigger Listeners to
       take actions.
       postCommit - fired after every commit or optimize command
       postOptimize - fired after every optimize command
    -->
  <!-- The RunExecutableListener executes an external command from a
       hook such as postCommit or postOptimize.
```

```
       exe - the name of the executable to run
       dir - dir to use as the current working directory. (default=".")
       wait - the calling thread waits until the executable returns.
              (default="true")
       args - the arguments to pass to the program.  (default is none)
       env - environment variables to set.  (default is none)
     -->
  <!-- This example shows how RunExecutableListener could be used
       with the script based replication...
       http://wiki.apache.org/solr/CollectionDistribution
     -->
  <!--
     <listener event="postCommit" class="solr.RunExecutableListener">
       <str name="exe">solr/bin/snapshooter</str>
       <str name="dir">.</str>
       <bool name="wait">true</bool>
       <arr name="args"> <str>arg1</str> <str>arg2</str> </arr>
       <arr name="env"> <str>MYVAR=val1</str> </arr>
     </listener>
    -->
</updateHandler>

<!-- IndexReaderFactory
     Use the following format to specify a custom IndexReaderFactory,
     which allows for alternate IndexReader implementations.
     ** Experimental Feature **
     Please note - Using a custom IndexReaderFactory may prevent
     certain other features from working. The API to
     IndexReaderFactory may change without warning or may even be
     removed from future releases if the problems cannot be
     resolved.

     ** Features that may not work with custom IndexReaderFactory **
     The ReplicationHandler assumes a disk-resident index. Using a
     custom IndexReader implementation may cause incompatibility
     with ReplicationHandler and may cause replication to not work
     correctly. See SOLR-1366 for details.
  -->
<!--
<indexReaderFactory name="IndexReaderFactory" class="package.class">
  <str name="someArg">Some Value</str>
</indexReaderFactory >
-->
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     Query section - these settings control query time things like caches
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<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>
```

```
    <!-- Slow Query Threshold (in millis)

         At high request rates, logging all requests can become a bottleneck
         and therefore INFO logging is often turned off. However, it is still
         useful to be able to set a latency threshold above which a request
         is considered "slow" and log that request at WARN level so we can
         easily identify slow queries.
    -->
    <slowQueryThresholdMillis>-1</slowQueryThresholdMillis>

    <!-- 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.
        Additional supported parameter by LRUCache:
           maxRamMB - the maximum amount of RAM (in MB) that this cache is
allowed
                       to occupy
     -->
    <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"/>

 <!-- custom cache currently used by block join -->
 <cache name="perSegFilter"
   class="solr.search.LRUCache"
   size="10"
   initialSize="0"
   autowarmCount="10"
   regenerator="solr.NoOpRegenerator" />
 <!-- 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>

<!-- Request Dispatcher
     This section contains instructions for how the SolrDispatchFilter
     should behave when processing requests for this SolrCore.
     handleSelect is a legacy option that affects the behavior of requests
     such as /select?qt=XXX
     handleSelect="true" will cause the SolrDispatchFilter to process
     the request and dispatch the query to a handler specified by the
     "qt" param, assuming "/select" isn't already registered.
     handleSelect="false" will cause the SolrDispatchFilter to
     ignore "/select" requests, resulting in a 404 unless a handler
     is explicitly registered with the name "/select"
     handleSelect="true" is not recommended for new users, but is the default
     for backwards compatibility
  -->
<requestDispatcher handleSelect="false" >
  <!-- Request Parsing
       These settings indicate how Solr Requests may be parsed, and
       what restrictions may be placed on the ContentStreams from
       those requests
       enableRemoteStreaming - enables use of the stream.file
       and stream.url parameters for specifying remote streams.
       multipartUploadLimitInKB - specifies the max size (in KiB) of
       Multipart File Uploads that Solr will allow in a Request.

       formdataUploadLimitInKB - specifies the max size (in KiB) of
       form data (application/x-www-form-urlencoded) sent via
       POST. You can use POST to pass request parameters not
       fitting into the URL.

       addHttpRequestToContext - if set to true, it will instruct
       the requestParsers to include the original HttpServletRequest
       object in the context map of the SolrQueryRequest under the
       key "httpRequest". It will not be used by any of the existing
       Solr components, but may be useful when developing custom
       plugins.

       *** WARNING ***
       The settings below authorize Solr to fetch remote files, You
       should make sure your system has some authentication before
       using enableRemoteStreaming="true"
    -->
  <requestParsers enableRemoteStreaming="true"
                  multipartUploadLimitInKB="2048000"
                  formdataUploadLimitInKB="2048"
                  addHttpRequestToContext="false"/>
  <!-- HTTP Caching
       Set HTTP caching related parameters (for proxy caches and clients).
       The options below instruct Solr not to output any HTTP Caching
       related headers
    -->
  <httpCaching never304="true" />
  <!-- If you include a <cacheControl> directive, it will be used to
```

```
       generate a Cache-Control header (as well as an Expires header
       if the value contains "max-age=")

       By default, no Cache-Control header is generated.

       You can use the <cacheControl> option even if you have set
       never304="true"
    -->
  <!--
     <httpCaching never304="true" >
       <cacheControl>max-age=30, public</cacheControl>
     </httpCaching>
    -->
  <!-- To enable Solr to respond with automatically generated HTTP
       Caching headers, and to response to Cache Validation requests
       correctly, set the value of never304="false"

       This will cause Solr to generate Last-Modified and ETag
       headers based on the properties of the Index.
       The following options can also be specified to affect the
       values of these headers...
       lastModFrom - the default value is "openTime" which means the
       Last-Modified value (and validation against If-Modified-Since
       requests) will all be relative to when the current Searcher
       was opened.  You can change it to lastModFrom="dirLastMod" if
       you want the value to exactly correspond to when the physical
       index was last modified.
       etagSeed="..." is an option you can change to force the ETag
       header (and validation against If-None-Match requests) to be
       different even if the index has not changed (ie: when making
       significant changes to your config file)
       (lastModifiedFrom and etagSeed are both ignored if you use
       the never304="true" option)
    -->
  <!--
     <httpCaching lastModifiedFrom="openTime"
                  etagSeed="Solr">
       <cacheControl>max-age=30, public</cacheControl>
     </httpCaching>
    -->
</requestDispatcher>
<!-- Request Handlers
     http://wiki.apache.org/solr/SolrRequestHandler
     Incoming queries will be dispatched to a specific handler by name
     based on the path specified in the request.
     Legacy behavior: If the request path uses "/select" but no Request
     Handler has that name, and if handleSelect="true" has been specified in
     the requestDispatcher, then the Request Handler is dispatched based on
     the qt parameter.  Handlers without a leading '/' are accessed this way
     like so: http://host/app/[core/]select?qt=name  If no qt is
     given, then the requestHandler that declares default="true" will be
     used or the one named "standard".
     If a Request Handler is declared with startup="lazy", then it will
     not be initialized until the first request that uses it.
  -->
<!-- SearchHandler
     http://wiki.apache.org/solr/SearchHandler
     For processing Search Queries, the primary Request Handler
     provided with Solr is "SearchHandler" It delegates to a sequent
```

```
       of SearchComponents (see below) and supports distributed
       queries across multiple shards
    -->
  <requestHandler name="/select" class="solr.SearchHandler">
    <!-- default values for query parameters can be specified, these
         will be overridden by parameters in the request
      -->
     <lst name="defaults">
       <str name="echoParams">explicit</str>
       <int name="rows">10</int>
     <str name="spellcheck.onlyMorePopular">true</str>
    <bool name="preferLocalShards">false</bool>
     </lst>
     <arr name="last-components">
    <str>spellcheck</str>
  </arr>
       <!-- Controls the distribution of a query to shards other than itself.
            Consider making 'preferLocalShards' true when:
              1) maxShardsPerNode > 1
              2) Number of shards > 1
              3) CloudSolrClient or LbHttpSolrServer is used by clients.
           Without this option, every core broadcasts the distributed query to
           a replica of each shard where the replicas are chosen randomly.
           This option directs the cores to prefer cores hosted locally, thus
           preventing network delays between machines.
           This behavior also immunizes a bad/slow machine from slowing down all
           the good machines (if those good machines were querying this bad
one).
           Specify this option=false for clients connecting through
HttpSolrServer
       -->
    <!-- In addition to defaults, "appends" params can be specified
         to identify values which should be appended to the list of
         multi-val params from the query (or the existing "defaults").
      -->
    <!-- In this example, the param "fq=instock:true" would be appended to
         any query time fq params the user may specify, as a mechanism for
         partitioning the index, independent of any user selected filtering
         that may also be desired (perhaps as a result of faceted searching).
         NOTE: there is *absolutely* nothing a client can do to prevent these
         "appends" values from being used, so don't use this mechanism
         unless you are sure you always want it.
      -->
    <!--
       <lst name="appends">
         <str name="fq">inStock:true</str>
       </lst>
      -->
    <!-- "invariants" are a way of letting the Solr maintainer lock down
         the options available to Solr clients.  Any params values
         specified here are used regardless of what values may be specified
         in either the query, the "defaults", or the "appends" params.
         In this example, the facet.field and facet.query params would
         be fixed, limiting the facets clients can use.  Faceting is
         not turned on by default - but if the client does specify
         facet=true in the request, these are the only facets they
         will be able to see counts for; regardless of what other
         facet.field or facet.query params they may specify.
         NOTE: there is *absolutely* nothing a client can do to prevent these
```

```
       "invariants" values from being used, so don't use this mechanism
       unless you are sure you always want it.
    -->
  <!--
     <lst name="invariants">
       <str name="facet.field">cat</str>
       <str name="facet.field">manu_exact</str>
       <str name="facet.query">price:[* TO 500]</str>
       <str name="facet.query">price:[500 TO *]</str>
     </lst>
    -->
  <!-- If the default list of SearchComponents is not desired, that
       list can either be overridden completely, or components can be
       prepended or appended to the default list.  (see below)
    -->
  <!--
     <arr name="components">
       <str>nameOfCustomComponent1</str>
       <str>nameOfCustomComponent2</str>
     </arr>
    -->
  </requestHandler>
<!-- A request handler that returns indented JSON by default -->
<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>
<!--
  The export request handler is used to export full sorted result sets.
  Do not change these defaults.
-->
<requestHandler name="/export" class="solr.SearchHandler">
  <lst name="invariants">
    <str name="rq">{!xport}</str>
    <str name="wt">xsort</str>
    <str name="distrib">false</str>
  </lst>
  <arr name="components">
    <str>query</str>
  </arr>
</requestHandler>

<!--
Distributed Stream processing.
-->
<requestHandler name="/stream" class="solr.StreamHandler">
  <lst name="invariants">
    <str name="wt">json</str>
    <str name="distrib">false</str>
  </lst>
</requestHandler>

<!-- A Robust Example

     This example SearchHandler declaration shows off usage of the
```

```
       SearchHandler with many defaults declared
       Note that multiple instances of the same Request Handler
       (SearchHandler) can be registered multiple times with different
       names (and different init parameters)
    -->
  <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="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,autho
r,resourcename</str>
       <int name="mlt.count">3</int>
       <!-- Faceting defaults -->
       <str name="facet">on</str>
       <str name="facet.missing">true</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.preserveMulti">true</str>
       <str name="hl.encoder">html</str>
       <str name="hl.simple.pre">&lt;b&gt;</str>
       <str name="hl.simple.post">&lt;/b&gt;</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>

  <!-- Update Request Handler.
       http://wiki.apache.org/solr/UpdateXmlMessages
       The canonical Request Handler for Modifying the Index through
       commands specified using XML, JSON, CSV, or JAVABIN
       Note: Since solr1.1 requestHandlers requires a valid content
       type header if posted in the body. For example, curl now
       requires: -H 'Content-type:text/xml; charset=utf-8'
       To override the request content type and force a specific
       Content-type, use the request parameter:
         ?update.contentType=text/csv
       This handler will pick a response format to match the input
       if the 'wt' parameter is not explicit
    -->
  <!--<requestHandler name="/update" class="solr.UpdateRequestHandler">
  </requestHandler>-->
  <initParams path="/update/**,/query,/select,/tvrh,/elevate,/spell,/browse">
    <lst name="defaults">
      <str name="df">text</str>
    </lst>
  </initParams>
  <initParams path="/update/json/docs">
    <lst name="defaults">
      <!--this ensures that the entire json doc will be stored verbatim into one
field-->
      <str name="srcField">_src_</str>
      <!--This means a the uniqueKeyField will be extracted from the fields and
       all fields go into the 'df' field. In this config df is already configured
to be 'text'
        -->
```

```
    <str name="mapUniqueKeyOnly">true</str>
  </lst>
</initParams>
<!-- The following are implicitly added
<requestHandler name="/update/json" class="solr.UpdateRequestHandler">
      <lst name="defaults">
       <str name="stream.contentType">application/json</str>
     </lst>
</requestHandler>
<requestHandler name="/update/csv" class="solr.UpdateRequestHandler">
      <lst name="defaults">
       <str name="stream.contentType">application/csv</str>
     </lst>
</requestHandler>
-->
<!-- Solr Cell Update Request Handler
     http://wiki.apache.org/solr/ExtractingRequestHandler
  -->
<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>

<!-- Field Analysis Request Handler
     RequestHandler that provides much the same functionality as
     analysis.jsp. Provides the ability to specify multiple field
     types and field names in the same request and outputs
     index-time and query-time analysis for each of them.
     Request parameters are:
     analysis.fieldname - field name whose analyzers are to be used
     analysis.fieldtype - field type whose analyzers are to be used
     analysis.fieldvalue - text for index-time analysis
     q (or analysis.q) - text for query time analysis
     analysis.showmatch (true|false) - When set to true and when
         query analysis is performed, the produced tokens of the
         field value analysis will be marked as "matched" for every
         token that is produces by the query analysis
 -->
<requestHandler name="/analysis/field"
                startup="lazy"
                class="solr.FieldAnalysisRequestHandler" />

<!-- Document Analysis Handler
     http://wiki.apache.org/solr/AnalysisRequestHandler
     An analysis handler that provides a breakdown of the analysis
     process of provided documents. This handler expects a (single)
     content stream with the following format:
     <docs>
       <doc>
         <field name="id">1</field>
         <field name="name">The Name</field>
```

```
          <field name="text">The Text Value</field>
        </doc>
        <doc>...</doc>
        <doc>...</doc>
        ...
      </docs>
    Note: Each document must contain a field which serves as the
    unique key. This key is used in the returned response to associate
    an analysis breakdown to the analyzed document.
    Like the FieldAnalysisRequestHandler, this handler also supports
    query analysis by sending either an "analysis.query" or "q"
    request parameter that holds the query text to be analyzed. It
    also supports the "analysis.showmatch" parameter which when set to
    true, all field tokens that match the query tokens will be marked
    as a "match".
  -->
  <requestHandler name="/analysis/document"
                  class="solr.DocumentAnalysisRequestHandler"
                  startup="lazy" />
  <!-- This single handler is equivalent to the following... -->
  <!--
     <requestHandler name="/admin/luke"
class="solr.admin.LukeRequestHandler" />
     <requestHandler name="/admin/system"
class="solr.admin.SystemInfoHandler" />
     <requestHandler name="/admin/plugins"
class="solr.admin.PluginInfoHandler" />
     <requestHandler name="/admin/threads"
class="solr.admin.ThreadDumpHandler" />
     <requestHandler name="/admin/properties"
class="solr.admin.PropertiesRequestHandler" />
     <requestHandler name="/admin/file"
class="solr.admin.ShowFileRequestHandler" >
    -->
  <!-- If you wish to hide files under ${solr.home}/conf, explicitly
       register the ShowFileRequestHandler using the definition below.
       NOTE: The glob pattern ('*') is the only pattern supported at present,
*.xml will
             not exclude all files ending in '.xml'. Use it to exclude _all_
updates
    -->
  <!--
     <requestHandler name="/admin/file"
                     class="solr.admin.ShowFileRequestHandler" >
       <lst name="invariants">
         <str name="hidden">synonyms.txt</str>
         <str name="hidden">anotherfile.txt</str>
         <str name="hidden">*</str>
       </lst>
     </requestHandler>
    -->
  <!--
    Enabling this request handler (which is NOT a default part of the admin
handler) will allow the Solr UI to edit
    all the config files. This is intended for secure/development use ONLY!
Leaving available and publically
    accessible is a security vulnerability and should be done with extreme
caution!
  -->
```

```
  <!--
  <requestHandler name="/admin/fileedit"
class="solr.admin.EditFileRequestHandler" >
    <lst name="invariants">
         <str name="hidden">synonyms.txt</str>
         <str name="hidden">anotherfile.txt</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>

  <!-- Search Components
       Search components are registered to SolrCore and used by
       instances of SearchHandler (which can access them by name)

       By default, the following components are available:

       <searchComponent name="query"     class="solr.QueryComponent" />
       <searchComponent name="facet"     class="solr.FacetComponent" />
       <searchComponent name="mlt"       class="solr.MoreLikeThisComponent" />
       <searchComponent name="highlight" class="solr.HighlightComponent" />
       <searchComponent name="stats"     class="solr.StatsComponent" />
       <searchComponent name="debug"     class="solr.DebugComponent" />

       Default configuration in a requestHandler would look like:
       <arr name="components">
         <str>query</str>
         <str>facet</str>
         <str>mlt</str>
         <str>highlight</str>
         <str>stats</str>
         <str>debug</str>
       </arr>
       If you register a searchComponent to one of the standard names,
       that will be used instead of the default.
       To insert components before or after the 'standard' components, use:

       <arr name="first-components">
         <str>myFirstComponentName</str>
       </arr>

       <arr name="last-components">
         <str>myLastComponentName</str>
       </arr>
       NOTE: The component registered with the name "debug" will
       always be executed after the "last-components"

     -->

   <!-- Spell Check
        The spell check component can return a list of alternative spelling
        suggestions.
        http://wiki.apache.org/solr/SpellCheckComponent
```

```
     -->
  <searchComponent name="spellcheck" class="solr.SpellCheckComponent">
    <str name="queryAnalyzerFieldType">text_spell</str>
    <!-- Multiple "Spell Checkers" can be declared and used by this
         component
      -->
    <!-- a spellchecker built from a field of the main index -->
    <lst name="spellchecker">
      <str name="name">default</str>
      <str name="field">spellcheck</str>
      <str name="classname">solr.DirectSolrSpellChecker</str>
      <!-- the spellcheck distance measure used, the default is the internal
levenshtein -->
      <str name="distanceMeasure">internal</str>
      <!-- minimum accuracy needed to be considered a valid spellcheck suggestion
-->
      <float name="accuracy">0.5</float>
      <!-- the maximum #edits we consider when enumerating terms: can be 1 or 2
-->
      <int name="maxEdits">2</int>
      <!-- the minimum shared prefix when enumerating terms -->
      <int name="minPrefix">1</int>
      <!-- maximum number of inspections per result. -->
      <int name="maxInspections">5</int>
      <!-- minimum length of a query term to be considered for correction -->
      <int name="minQueryLength">4</int>
      <!-- maximum threshold of documents a query term can appear to be
considered for correction -->
      <float name="maxQueryFrequency">0.01</float>
      <!-- uncomment this to require suggestions to occur in 1% of the documents
       <float name="thresholdTokenFrequency">.01</float>
      -->
    </lst>

    <!-- a spellchecker that can break or combine words.  See "/spell" handler
below for usage -->
    <lst name="spellchecker">
      <str name="name">wordbreak</str>
      <str name="classname">solr.WordBreakSolrSpellChecker</str>
      <str name="field">spellcheck</str>
      <str name="combineWords">true</str>
      <str name="breakWords">true</str>
      <int name="maxChanges">10</int>
    </lst>

 <lst name="spellchecker">
  <str name="name">en</str>
  <str name="field">spellcheck_en</str>
  <str name="classname">solr.DirectSolrSpellChecker</str>
  <str name="buildOnCommit">true</str>
  <str name="buildOnOptimize">true</str>
 </lst>
 <lst name="spellchecker">
  <str name="name">de</str>
  <str name="field">spellcheck_de</str>
  <str name="classname">solr.DirectSolrSpellChecker</str>
  <str name="buildOnCommit">true</str>
  <str name="buildOnOptimize">true</str>
 </lst>
```

```
<lst name="spellchecker">
 <str name="name">fr</str>
 <str name="field">spellcheck_fr</str>
 <str name="classname">solr.DirectSolrSpellChecker</str>
 <str name="buildOnCommit">true</str>
 <str name="buildOnOptimize">true</str>
</lst>
<lst name="spellchecker">
 <str name="name">ja</str>
 <str name="field">spellcheck_ja</str>
 <str name="classname">solr.DirectSolrSpellChecker</str>
 <str name="buildOnCommit">true</str>
 <str name="buildOnOptimize">true</str>
</lst>
<lst name="spellchecker">
 <str name="name">zh</str>
 <str name="field">spellcheck_zh</str>
 <str name="classname">solr.DirectSolrSpellChecker</str>
 <str name="buildOnCommit">true</str>
 <str name="buildOnOptimize">true</str>
</lst>
<lst name="spellchecker">
 <str name="name">pt</str>
 <str name="field">spellcheck_pt</str>
 <str name="classname">solr.DirectSolrSpellChecker</str>
 <str name="buildOnCommit">true</str>
 <str name="buildOnOptimize">true</str>
</lst>
   <!-- a spellchecker that uses a different distance measure -->
   <!--
      <lst name="spellchecker">
        <str name="name">jarowinkler</str>
        <str name="field">spell</str>
        <str name="classname">solr.DirectSolrSpellChecker</str>
        <str name="distanceMeasure">
          org.apache.lucene.search.spell.JaroWinklerDistance
        </str>
      </lst>
    -->
   <!-- a spellchecker that use an alternate comparator
        comparatorClass be one of:
         1. score (default)
         2. freq (Frequency first, then score)
         3. A fully qualified class name
     -->
   <!--
      <lst name="spellchecker">
        <str name="name">freq</str>
        <str name="field">lowerfilt</str>
        <str name="classname">solr.DirectSolrSpellChecker</str>
        <str name="comparatorClass">freq</str>
     -->
   <!-- A spellchecker that reads the list of words from a file -->
   <!--
      <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>
      </lst>
      -->
  </searchComponent>

  <!-- A request handler for demonstrating the spellcheck component.
       NOTE: This is purely as an example.  The whole purpose of the
       SpellCheckComponent is to hook it into the request handler that
       handles your normal user queries so that a separate request is
       not needed to get suggestions.
       IN OTHER WORDS, THERE IS REALLY GOOD CHANCE THE SETUP BELOW IS
       NOT WHAT YOU WANT FOR YOUR PRODUCTION SYSTEM!

       See http://wiki.apache.org/solr/SpellCheckComponent for details
       on the request parameters.
    -->
  <requestHandler name="/spell" class="solr.SearchHandler" startup="lazy">
    <lst name="defaults">
      <!-- Solr will use suggestions from both the 'default' spellchecker
           and from the 'wordbreak' spellchecker and combine them.
           collations (re-written queries) can include a combination of
           corrections from both spellcheckers -->
      <str name="spellcheck.dictionary">default</str>
      <str name="spellcheck.dictionary">wordbreak</str>
      <str name="spellcheck">on</str>
      <str name="spellcheck.extendedResults">true</str>
      <str name="spellcheck.count">10</str>
      <str name="spellcheck.alternativeTermCount">5</str>
      <str name="spellcheck.maxResultsForSuggest">5</str>
      <str name="spellcheck.collate">true</str>
      <str name="spellcheck.collateExtendedResults">true</str>
      <str name="spellcheck.maxCollationTries">10</str>
      <str name="spellcheck.maxCollations">5</str>
    </lst>
    <arr name="last-components">
      <str>spellcheck</str>
    </arr>
  </requestHandler>
  <!-- The SuggestComponent in Solr provides users with automatic suggestions for
query terms.
       You can use this to implement a powerful auto-suggest feature in your
search application.
       As with the rest of this solrconfig.xml file, the configuration of this
component is purely
       an example that applies specifically to this configset and example
documents.

       More information about this component and other configuration options are
described in the
       "Suggester" section of the reference guide available at
       http://archive.apache.org/dist/lucene/solr/ref-guide
    -->
  <searchComponent name="suggest" class="solr.SuggestComponent">
    <lst name="suggester">
      <str name="name">default</str>
      <str name="lookupImpl">FuzzyLookupFactory</str>
      <str name="dictionaryImpl">DocumentDictionaryFactory</str>
      <str name="field">autosuggest</str>
      <str name="weightField">price</str>
```

```
      <str name="suggestAnalyzerFieldType">text_spell</str>
      <str name="buildOnStartup">false</str>
    </lst>
    <lst name="suggester">
      <str name="name">en</str>
      <str name="lookupImpl">FuzzyLookupFactory</str>
      <str name="dictionaryImpl">DocumentDictionaryFactory</str>
      <str name="field">autosuggest_en</str>
   <str name="threshold">0.2</str>
      <str name="weightField">price</str>
      <str name="suggestAnalyzerFieldType">text_spell_en</str>
      <str name="buildOnStartup">false</str>
    </lst>
    <lst name="suggester">
      <str name="name">de</str>
      <str name="lookupImpl">FuzzyLookupFactory</str>
      <str name="dictionaryImpl">DocumentDictionaryFactory</str>
      <str name="field">autosuggest_de</str>
   <str name="threshold">0.2</str>
      <str name="weightField">price</str>
      <str name="suggestAnalyzerFieldType">text_spell_de</str>
      <str name="buildOnStartup">false</str>
    </lst>
    <lst name="suggester">
      <str name="name">zh</str>
      <str name="lookupImpl">FuzzyLookupFactory</str>
      <str name="dictionaryImpl">DocumentDictionaryFactory</str>
      <str name="field">autosuggest_zh</str>
   <str name="threshold">0.2</str>
      <str name="weightField">price</str>
      <str name="suggestAnalyzerFieldType">text_spell_zh</str>
      <str name="buildOnStartup">false</str>
    </lst>
  </searchComponent>
  <requestHandler name="/suggest" class="solr.SearchHandler" startup="lazy">
    <lst name="defaults">
      <str name="suggest">true</str>
      <str name="suggest.count">10</str>
    </lst>
    <arr name="components">
      <str>suggest</str>
    </arr>
  </requestHandler>
  <!-- Hybris update synonyms request handler -->
  <requestHandler name="/updateSynonyms"

class="de.hybris.platform.solrfacetsearch.ysolr.handlers.UpdateSynonymsRequestHan
dler" />
  <requestHandler name="/updateStopwords"

class="de.hybris.platform.solrfacetsearch.ysolr.handlers.UpdateStopWordsRequestHa
ndler" />
  <!-- Term Vector Component
       http://wiki.apache.org/solr/TermVectorComponent
    -->
  <searchComponent name="tvComponent" class="solr.TermVectorComponent"/>
  <!-- A request handler for demonstrating the term vector component
       This is purely as an example.
       In reality you will likely want to add the component to your
```

```
       already specified request handlers.
    -->
  <requestHandler name="/tvrh" class="solr.SearchHandler" startup="lazy">
    <lst name="defaults">
      <bool name="tv">true</bool>
    </lst>
    <arr name="last-components">
      <str>tvComponent</str>
    </arr>
  </requestHandler>
  <!-- Clustering Component
       You'll need to set the solr.clustering.enabled system property
       when running solr to run with clustering enabled:
            java -Dsolr.clustering.enabled=true -jar start.jar
       http://wiki.apache.org/solr/ClusteringComponent
       http://carrot2.github.io/solr-integration-strategies/
    -->
  <searchComponent name="clustering"
                   enable="${solr.clustering.enabled:false}"
                   class="solr.clustering.ClusteringComponent" >
    <lst name="engine">
      <str name="name">lingo</str>
      <!-- Class name of a clustering algorithm compatible with the Carrot2
framework.
           Currently available open source algorithms are:
           * org.carrot2.clustering.lingo.LingoClusteringAlgorithm
           * org.carrot2.clustering.stc.STCClusteringAlgorithm
           * org.carrot2.clustering.kmeans.BisectingKMeansClusteringAlgorithm
           See http://project.carrot2.org/algorithms.html for more information.
           A commercial algorithm Lingo3G (needs to be installed separately) is
defined as:
           * com.carrotsearch.lingo3g.Lingo3GClusteringAlgorithm
        -->
      <str
name="carrot.algorithm">org.carrot2.clustering.lingo.LingoClusteringAlgorithm</st
r>
      <!-- Override location of the clustering algorithm's resources
           (attribute definitions and lexical resources).
           A directory from which to load algorithm-specific stop words,
           stop labels and attribute definition XMLs.
           For an overview of Carrot2 lexical resources, see:
           http://download.carrot2.org/head/manual/#chapter.lexical-resources
           For an overview of Lingo3G lexical resources, see:

http://download.carrotsearch.com/lingo3g/manual/#chapter.lexical-resources
       -->
      <str name="carrot.resourcesDir">clustering/carrot2</str>
    </lst>
    <!-- An example definition for the STC clustering algorithm. -->
    <lst name="engine">
      <str name="name">stc</str>
      <str
name="carrot.algorithm">org.carrot2.clustering.stc.STCClusteringAlgorithm</str>
    </lst>
    <!-- An example definition for the bisecting kmeans clustering algorithm. -->
    <lst name="engine">
      <str name="name">kmeans</str>
      <str
name="carrot.algorithm">org.carrot2.clustering.kmeans.BisectingKMeansClusteringAl
```

```
gorithm</str>
    </lst>
  </searchComponent>
  <!-- A request handler for demonstrating the clustering component
       This is purely as an example.
       In reality you will likely want to add the component to your
       already specified request handlers.
    -->
  <requestHandler name="/clustering"
                  startup="lazy"
                  enable="${solr.clustering.enabled:false}"
                  class="solr.SearchHandler">
    <lst name="defaults">
      <bool name="clustering">true</bool>
      <bool name="clustering.results">true</bool>
      <!-- Field name with the logical "title" of a each document (optional) -->
      <str name="carrot.title">name</str>
      <!-- Field name with the logical "URL" of a each document (optional) -->
      <str name="carrot.url">id</str>
      <!-- Field name with the logical "content" of a each document (optional)
-->
      <str name="carrot.snippet">features</str>
      <!-- Apply highlighter to the title/ content and use this for clustering.
-->
      <bool name="carrot.produceSummary">true</bool>
      <!-- the maximum number of labels per cluster -->
      <!--<int name="carrot.numDescriptions">5</int>-->
      <!-- produce sub clusters -->
      <bool name="carrot.outputSubClusters">false</bool>
      <!-- Configure the remaining request handler parameters. -->
      <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
      </str>
      <str name="q.alt">*:*</str>
      <str name="rows">10</str>
      <str name="fl">*,score</str>
    </lst>
    <arr name="last-components">
      <str>clustering</str>
    </arr>
  </requestHandler>

  <!-- Terms Component
       http://wiki.apache.org/solr/TermsComponent
       A component to return terms and document frequency of those
       terms
    -->
  <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>
```

```
<!-- Query Elevation Component
     http://wiki.apache.org/solr/QueryElevationComponent
     a search component that enables you to configure the top
     results for a given query regardless of the normal lucene
     scoring.
  -->
<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>
  </lst>
  <arr name="last-components">
    <str>elevator</str>
  </arr>
</requestHandler>
<!-- Highlighting Component
     http://wiki.apache.org/solr/HighlightingParameters
  -->
<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">
        <!--
        <lst name="defaults">
          <str name="hl.multiValuedSeparatorChar">/</str>
        </lst>
        -->
      </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>
  <!-- Update Processors
       Chains of Update Processor Factories for dealing with Update
       Requests can be declared, and then used by name in Update
       Request Processors
       http://wiki.apache.org/solr/UpdateRequestProcessor
    -->
  <!-- Deduplication
       An example dedup update processor that creates the "id" field
       on the fly based on the hash code of some other fields.  This
       example has overwriteDupes set to false since we are using the
       id field as the signatureField and Solr will maintain
       uniqueness based on that anyway.

    -->
  <!--
     <updateRequestProcessorChain name="dedupe">
       <processor class="solr.processor.SignatureUpdateProcessorFactory">
         <bool name="enabled">true</bool>
         <str name="signatureField">id</str>
         <bool name="overwriteDupes">false</bool>
         <str name="fields">name,features,cat</str>
         <str name="signatureClass">solr.processor.Lookup3Signature</str>
       </processor>
       <processor class="solr.LogUpdateProcessorFactory" />
       <processor class="solr.RunUpdateProcessorFactory" />
     </updateRequestProcessorChain>
    -->

  <!-- Language identification
       This example update chain identifies the language of the incoming
       documents using the langid contrib. The detected language is
       written to field language_s. No field name mapping is done.
       The fields used for detection are text, title, subject and description,
       making this example suitable for detecting languages form full-text
       rich documents injected via ExtractingRequestHandler.
       See more about langId at http://wiki.apache.org/solr/LanguageDetection
    -->
    <!--
     <updateRequestProcessorChain name="langid">
       <processor
class="org.apache.solr.update.processor.TikaLanguageIdentifierUpdateProcessorFact
ory">
         <str name="langid.fl">text,title,subject,description</str>
         <str name="langid.langField">language_s</str>
         <str name="langid.fallback">en</str>
       </processor>
       <processor class="solr.LogUpdateProcessorFactory" />
       <processor class="solr.RunUpdateProcessorFactory" />
     </updateRequestProcessorChain>
    -->
  <!-- Script update processor
    This example hooks in an update processor implemented using JavaScript.
    See more about the script update processor at
```

```
http://wiki.apache.org/solr/ScriptUpdateProcessor
  -->
  <!--
    <updateRequestProcessorChain name="script">
      <processor class="solr.StatelessScriptUpdateProcessorFactory">
        <str name="script">update-script.js</str>
        <lst name="params">
          <str name="config_param">example config parameter</str>
        </lst>
      </processor>
      <processor class="solr.RunUpdateProcessorFactory" />
    </updateRequestProcessorChain>
  -->

  <!-- Response Writers
       http://wiki.apache.org/solr/QueryResponseWriter
       Request responses will be written using the writer specified by
       the 'wt' request parameter matching the name of a registered
       writer.
       The "default" writer is the default and will be used if 'wt' is
       not specified in the request.
    -->
  <!-- The following response writers are implicitly configured unless
       overridden...
    -->
  <!--
     <queryResponseWriter name="xml"
                          default="true"
                          class="solr.XMLResponseWriter" />
     <queryResponseWriter name="json" class="solr.JSONResponseWriter"/>
     <queryResponseWriter name="python" class="solr.PythonResponseWriter"/>
     <queryResponseWriter name="ruby" class="solr.RubyResponseWriter"/>
     <queryResponseWriter name="php" class="solr.PHPResponseWriter"/>
     <queryResponseWriter name="phps" class="solr.PHPSerializedResponseWriter"/>
     <queryResponseWriter name="csv" class="solr.CSVResponseWriter"/>
     <queryResponseWriter name="schema.xml"
class="solr.SchemaXmlResponseWriter"/>
    -->
  <queryResponseWriter name="json" class="solr.JSONResponseWriter">
     <!-- For the purposes of the tutorial, JSON responses are written as
      plain text so that they are easy to read in *any* browser.
      If you expect a MIME type of "application/json" just remove this override.

    <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">
      <str name="template.base.dir">${velocity.template.base.dir:}</str>
    </queryResponseWriter>

  <!-- XSLT response writer transforms the XML output by any xslt file found
       in Solr's conf/xslt directory.  Changes to xslt files are checked for
       every xsltCacheLifetimeSeconds.
    -->
  <queryResponseWriter name="xslt" class="solr.XSLTResponseWriter">
```

```
    <int name="xsltCacheLifetimeSeconds">5</int>
  </queryResponseWriter>
  <!-- Query Parsers
       http://wiki.apache.org/solr/SolrQuerySyntax
       Multiple QParserPlugins can be registered by name, and then
       used in either the "defType" param for the QueryComponent (used
       by SearchHandler) or in LocalParams
    -->
  <!-- example of registering a query parser -->
  <!--
     <queryParser name="myparser" class="com.mycompany.MyQParserPlugin"/>
    -->
  <!-- Function Parsers
       http://wiki.apache.org/solr/FunctionQuery
       Multiple ValueSourceParsers can be registered by name, and then
       used as function names when using the "func" QParser.
    -->
  <!-- example of registering a custom function parser  -->
  <!--
     <valueSourceParser name="myfunc"
                        class="com.mycompany.MyValueSourceParser" />
    -->


  <!-- Document Transformers
       http://wiki.apache.org/solr/DocTransformers
    -->
  <!--
     Could be something like:
     <transformer name="db" class="com.mycompany.LoadFromDatabaseTransformer" >
       <int name="connection">jdbc://....</int>
     </transformer>

     To add a constant value to all docs, use:
     <transformer name="mytrans2"
class="org.apache.solr.response.transform.ValueAugmenterFactory" >
       <int name="value">5</int>
     </transformer>

     If you want the user to still be able to change it with _value:something_
use this:
     <transformer name="mytrans3"
class="org.apache.solr.response.transform.ValueAugmenterFactory" >
       <double name="defaultValue">5</double>
     </transformer>
      If you are using the QueryElevationComponent, you may wish to mark
documents that get boosted.  The
      EditorialMarkerFactory will do exactly that:
     <transformer name="qecBooster"
class="org.apache.solr.response.transform.EditorialMarkerFactory" />
    -->

  <!-- Legacy config for the admin interface -->
  <admin>
    <defaultQuery>*:*</defaultQuery>
  </admin>
```

```
</config>
```

- schema.xml

schema.xml Expand source

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
 Licensed to the Apache Software Foundation (ASF) under one or more
 contributor license agreements.  See the NOTICE file distributed with
 this work for additional information regarding copyright ownership.
 The ASF licenses this file to You under the Apache License, Version 2.0
 (the "License"); you may not use this file except in compliance with
 the License.  You may obtain a copy of the License at
     http://www.apache.org/licenses/LICENSE-2.0
 Unless required by applicable law or agreed to in writing, software
 distributed under the License is distributed on an "AS IS" BASIS,
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 See the License for the specific language governing permissions and
 limitations under the License.
-->
<!--
 This is the Solr schema file. This file should be named "schema.xml" and
 should be in the conf directory under the solr home
 (i.e. ./solr/conf/schema.xml by default)
 or located where the classloader for the Solr webapp can find it.
 This example schema is the recommended starting point for users.
 It should be kept correct and concise, usable out-of-the-box.
 For more information, on how to customize this file, please see
 http://wiki.apache.org/solr/SchemaXml
 PERFORMANCE NOTE: this schema includes many optional features and should not
 be used for benchmarking.  To improve performance one could
  - set stored="false" for all fields possible (esp large fields) when you
    only need to search on the field but don't need to return the original
    value.
  - set indexed="false" if you don't need to search on the field, but only
    return the field as a result of searching on other indexed fields.
  - remove all unneeded copyField statements
  - for best index size and searching performance, set "index" to false
    for all general text fields, use copyField to copy them to the
    catchall "text" field, and use that for searching.
  - For maximum indexing performance, use the ConcurrentUpdateSolrServer
    java client.
  - Remember to run the JVM in server mode, and use a higher logging level
    that avoids logging every request
-->
<schema name="hybris" version="1.5">
  <!-- attribute "name" is the name of this schema and is only used for display
purposes.
       version="x.y" is Solr's version number for the schema syntax and
       semantics.  It should not normally be changed by applications.
       1.0: multiValued attribute did not exist, all fields are multiValued
            by nature
       1.1: multiValued attribute introduced, false by default
       1.2: omitTermFreqAndPositions attribute introduced, true by default
```

```
            except for text fields.
       1.3: removed optional field compress feature
       1.4: autoGeneratePhraseQueries attribute introduced to drive QueryParser
            behavior when a single string produces multiple tokens.  Defaults
            to off for version >= 1.4
       1.5: omitNorms defaults to true for primitive field types
            (int, float, boolean, string...)
     -->

   <!-- Valid attributes for fields:
     name: mandatory - the name for the field
     type: mandatory - the name of a field type from the
       <types> fieldType section
     indexed: true if this field should be indexed (searchable or sortable)
     stored: true if this field should be retrievable
     docValues: true if this field should have doc values. Doc values are
       useful for faceting, grouping, sorting and function queries. Although not
       required, doc values will make the index faster to load, more
       NRT-friendly and more memory-efficient. They however come with some
       limitations: they are currently only supported by StrField, UUIDField
       and all Trie*Fields, and depending on the field type, they might
       require the field to be single-valued, be required or have a default
       value (check the documentation of the field type you're interested in
       for more information)
     multiValued: true if this field may contain multiple values per document
     omitNorms: (expert) set to true to omit the norms associated with
       this field (this disables length normalization and index-time
       boosting for the field, and saves some memory).  Only full-text
       fields or fields that need an index-time boost need norms.
       Norms are omitted for primitive (non-analyzed) types by default.
     termVectors: [false] set to true to store the term vector for a
       given field.
       When using MoreLikeThis, fields used for similarity should be
       stored for best performance.
     termPositions: Store position information with the term vector.
       This will increase storage costs.
     termOffsets: Store offset information with the term vector. This
       will increase storage costs.
     termPayloads: Store payload information with the term vector. This
       will increase storage costs.
     required: The field is required.  It will throw an error if the
       value does not exist
     default: a value that should be used if no value is specified
       when adding a document.
   -->
   <!-- field names should consist of alphanumeric or underscore characters only
and
      not start with a digit.  This is not currently strictly enforced,
      but other field names will not have first class support from all components
      and back compatibility is not guaranteed.  Names with both leading and
      trailing underscores (e.g. _version_) are reserved.
   -->
   <!-- If you remove this field, you must _also_ disable the update log in
solrconfig.xml
      or Solr won't start. _version_ and update log are required for SolrCloud
   -->
   <field name="_version_" type="long" indexed="true" stored="true"/>

   <!-- points to the root document of a block of nested documents. Required for
```

```
nested
      document support, may be removed otherwise
   -->
   <field name="_root_" type="string" indexed="true" stored="false"/>
   <!-- Only remove the "id" field if you have a very good reason to. While not
strictly
     required, it is highly recommended. A <uniqueKey> is present in almost all
Solr
     installations. See the <uniqueKey> declaration below where <uniqueKey> is
set to "id".
     Do NOT change the type and apply index-time analysis to the <uniqueKey> as
it will likely
     make routing in SolrCloud and document replacement in general fail. Limited
_query_ time
     analysis is possible as long as the indexing process is guaranteed to index
the term
     in a compatible way. Any analysis applied to the <uniqueKey> should _not_
produce multiple
     tokens
   -->
    <field name="id" type="string" indexed="true" stored="true" required="true"
multiValued="false" />
 <field name="pk" type="long" indexed="true" stored="true" multiValued="false" />

 <field name="catalogId" type="string" indexed="true" stored="true"
multiValued="false" />
 <field name="catalogVersion" type="string" indexed="true" stored="true"
multiValued="false" />
 <field name="text" type="text_general" indexed="true" stored="true" />
 <field name="code_string" type="string" indexed="true" stored="true"
multiValued="false" />
 <field name="autosuggest" type="text_spell" indexed="true" stored="true"
multiValued="true" />
 <field name="autosuggest_de" type="text_spell_de" indexed="true" stored="true"
multiValued="true" />
 <field name="autosuggest_en" type="text_spell_en" indexed="true" stored="true"
multiValued="true" />
 <field name="autosuggest_fr" type="text_spell_fr" indexed="true" stored="true"
multiValued="true" />
 <field name="autosuggest_ja" type="text_spell" indexed="true" stored="true"
multiValued="true" />
 <field name="autosuggest_zh" type="text_spell_zh" indexed="true" stored="true"
multiValued="true" />
 <field name="autosuggest_pt" type="text_spell_pt" indexed="true" stored="true"
multiValued="true" />
 <field name="autosuggest_it" type="text_spell_it" indexed="true" stored="true"
multiValued="true" />
 <copyField source="autosuggest" dest="autosuggest_de" />
 <copyField source="autosuggest" dest="autosuggest_en" />
 <copyField source="autosuggest" dest="autosuggest_fr" />
 <copyField source="autosuggest" dest="autosuggest_ja" />
 <copyField source="autosuggest" dest="autosuggest_zh" />
 <copyField source="autosuggest" dest="autosuggest_pt" />
 <copyField source="autosuggest" dest="autosuggest_it" />
 <field name="spellcheck" type="text_spell" indexed="true" stored="true"
multiValued="true" />
 <field name="spellcheck_de" type="text_spell_de" indexed="true" stored="true"
multiValued="true" />
 <field name="spellcheck_en" type="text_spell_en" indexed="true" stored="true"
```

```
multiValued="true" />
 <field name="spellcheck_fr" type="text_spell_fr" indexed="true" stored="true"
multiValued="true" />
 <field name="spellcheck_ja" type="text_spell" indexed="true" stored="true"
multiValued="true" />
 <field name="spellcheck_zh" type="text_spell_zh" indexed="true" stored="true"
multiValued="true" />
 <field name="spellcheck_pt" type="text_spell_pt" indexed="true" stored="true"
multiValued="true" />
 <field name="spellcheck_it" type="text_spell_it" indexed="true" stored="true"
multiValued="true" />
 <copyField source="spellcheck" dest="spellcheck_de" />
 <copyField source="spellcheck" dest="spellcheck_en" />
 <copyField source="spellcheck" dest="spellcheck_fr" />
 <copyField source="spellcheck" dest="spellcheck_ja" />
 <copyField source="spellcheck" dest="spellcheck_zh" />
 <copyField source="spellcheck" dest="spellcheck_pt" />
 <copyField source="spellcheck" dest="spellcheck_it" />
 <dynamicField name="*_path" type="path" indexed="true" stored="true" />
 <dynamicField name="*_path_mv" type="path" indexed="true" stored="true"
multiValued="true"/>
 <dynamicField name="*_text_de" type="text_de" indexed="true" stored="true" />
 <dynamicField name="*_text_de_mv" type="text_de" indexed="true" stored="true"
multiValued="true"/>

 <dynamicField name="*_text_en" type="text_en" indexed="true" stored="true" />
 <dynamicField name="*_text_en_mv" type="text_en" indexed="true" stored="true"
multiValued="true"/>
 <dynamicField name="*_text_fr" type="text_fr" indexed="true" stored="true" />
 <dynamicField name="*_text_fr_mv" type="text_fr" indexed="true" stored="true"
multiValued="true"/>
 <dynamicField name="*_text_ja" type="text_ja" indexed="true" stored="true" />
 <dynamicField name="*_text_ja_mv" type="text_ja" indexed="true" stored="true"
multiValued="true"/>
 <dynamicField name="*_text_zh" type="text_zh" indexed="true" stored="true" />
 <dynamicField name="*_text_zh_mv" type="text_zh" indexed="true" stored="true"
multiValued="true"/>
 <dynamicField name="*_text_pt" type="text_pt" indexed="true" stored="true" />
 <dynamicField name="*_text_pt_mv" type="text_pt" indexed="true" stored="true"
multiValued="true"/>
 <dynamicField name="*_text_it" type="text_it" indexed="true" stored="true" />
 <dynamicField name="*_text_it_mv" type="text_it" indexed="true" stored="true"
multiValued="true"/>
 <copyField source="*_text_de" dest="fulltext_de" />
 <copyField source="*_text_en" dest="fulltext_en" />
 <copyField source="*_text_fr" dest="fulltext_fr" />
 <copyField source="*_text_ja" dest="fulltext_ja" />
 <copyField source="*_text_zh" dest="fulltext_zh" />
 <copyField source="*_text_pt" dest="fulltext_pt" />
 <copyField source="*_text_it" dest="fulltext_it" />
 <field name="fulltext_de" type="text_de" indexed="true" stored="false"
multiValued="true" />
 <field name="fulltext_en" type="text_en" indexed="true" stored="false"
multiValued="true" />
 <field name="fulltext_fr" type="text_fr" indexed="true" stored="false"
multiValued="true" />
 <field name="fulltext_ja" type="text_ja" indexed="true" stored="false"
multiValued="true" />
 <field name="fulltext_zh" type="text_zh" indexed="true" stored="false"
```

```
multiValued="true" />
 <field name="fulltext_pt" type="text_pt" indexed="true" stored="false"
multiValued="true" />
 <field name="fulltext_it" type="text_it" indexed="true" stored="false"
multiValued="true" />
 <!--
  Dynamic field definitions. If a field name is not found,
  dynamicFields will be used if the name matches any of the patterns.
  RESTRICTION: the glob-like pattern in the name attribute must have a
  "*" only at the start or the end. EXAMPLE: name="*_i" will match any
  field ending in _i (like myid_i, z_i) Longer patterns will be matched
  first. if equal size patterns both match, the first appearing in the
  schema will be used.
 -->
 <dynamicField name="*_int" type="int" indexed="true" stored="true" />
 <dynamicField name="*_int_mv" type="int" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_string" type="string" indexed="true" stored="true" />
 <dynamicField name="*_string_mv" type="string" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_long" type="long" indexed="true" stored="true" />
 <dynamicField name="*_long_mv" type="long" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_text" type="text" indexed="true" stored="true" />
 <dynamicField name="*_text_mv" type="text" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_boolean" type="boolean" indexed="true" stored="true" />
 <dynamicField name="*_boolean_mv" type="boolean" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_float" type="float" indexed="true" stored="true" />
 <dynamicField name="*_float_mv" type="float" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_double" type="double" indexed="true" stored="true" />
 <dynamicField name="*_double_mv" type="double" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_date" type="date" indexed="true" stored="true" />
 <dynamicField name="*_date_mv" type="date" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_sortabletext" type="sortabletext" indexed="true"
stored="true" />
 <dynamicField name="*_sortabletext_mv" type="sortabletext" indexed="true"
stored="true" multiValued="true" />
 <!-- some trie-coded dynamic fields for faster range queries -->
 <dynamicField name="*_tint" type="tint" indexed="true" stored="true" />
 <dynamicField name="*_tint_mv" type="tint" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_tlong" type="tlong" indexed="true" stored="true" />
 <dynamicField name="*_tlong_mv" type="tlong" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_tfloat" type="tfloat" indexed="true" stored="true" />
 <dynamicField name="*_tfloat_mv" type="tfloat" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_tdouble" type="tdouble" indexed="true" stored="true" />
 <dynamicField name="*_tdouble_mv" type="tdouble" indexed="true" stored="true"
multiValued="true" />
 <dynamicField name="*_tdate" type="tdate" indexed="true" stored="true" />
 <dynamicField name="*_tdate_mv" type="tdate" indexed="true" stored="true"
multiValued="true" />
```

```
 <!-- Field to use to determine and enforce document uniqueness.
      Unless this field is marked with required="false", it will be a required
field
   -->
 <uniqueKey>id</uniqueKey>
 <!-- DEPRECATED: The defaultSearchField is consulted by various query parsers
when
  parsing a query string that isn't explicit about the field.  Machine (non-user)
  generated queries are best made explicit, or they can use the "df" request
parameter
  which takes precedence over this.
  Note: Un-commenting defaultSearchField will be insufficient if your request
handler
  in solrconfig.xml defines "df", which takes precedence. That would need to be
removed.
 <defaultSearchField>text</defaultSearchField> -->
 <!-- DEPRECATED: The defaultOperator (AND|OR) is consulted by various query
parsers
  when parsing a query string to determine if a clause of the query should be
marked as
  required or optional, assuming the clause isn't already marked by some
operator.
  The default is OR, which is generally assumed so it is not a good idea to
change it
  globally here.  The "q.op" request parameter takes precedence over this.
 <solrQueryParser defaultOperator="OR"/> -->

    <!-- field type definitions. The "name" attribute is
       just a label to be used by field definitions.  The "class"
       attribute and any other attributes determine the real
       behavior of the fieldType.
         Class names starting with "solr" refer to java classes in a
       standard package such as org.apache.solr.analysis
    -->
    <!-- The StrField type is not analyzed, but indexed/stored verbatim.
       It supports doc values but in that case the field needs to be
       single-valued and either required or have a default value.
      -->
    <fieldType name="string" class="solr.StrField" sortMissingLast="true" />
    <!-- boolean type: "true" or "false" -->
    <fieldType name="boolean" class="solr.BoolField" sortMissingLast="true"/>
    <!-- sortMissingLast and sortMissingFirst attributes are optional attributes
are
         currently supported on types that are sorted internally as strings
         and on numeric types.
      This includes "string","boolean", and, as of 3.5 (and 4.x),
      int, float, long, date, double, including the "Trie" variants.
       - If sortMissingLast="true", then a sort on this field will cause
documents
         without the field to come after documents with the field,
         regardless of the requested sort order (asc or desc).
       - If sortMissingFirst="true", then a sort on this field will cause
documents
         without the field to come before documents with the field,
         regardless of the requested sort order.
       - If sortMissingLast="false" and sortMissingFirst="false" (the default),
         then default lucene sorting will be used which places docs without the
         field first in an ascending sort and last in a descending sort.
    -->
```

```
    <!--
      Default numeric field types. For faster range queries, consider the
tint/tfloat/tlong/tdouble types.
      These fields support doc values, but they require the field to be
      single-valued and either be required or have a default value.
    -->
    <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"/>
    <!--
     Numeric field types that index each value at various levels of precision
     to accelerate range queries when the number of values between the range
     endpoints is large. See the javadoc for NumericRangeQuery for internal
     implementation details.
     Smaller precisionStep values (specified in bits) will lead to more tokens
     indexed per value, slightly larger index size, and faster range queries.
     A precisionStep of 0 disables indexing at different precision levels.
    -->
    <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"/>
    <!-- The format for this date field is of the form 1995-12-31T23:59:59Z, and
         is a more restricted form of the canonical representation of dateTime
         http://www.w3.org/TR/xmlschema-2/#dateTime
         The trailing "Z" designates UTC time and is mandatory.
         Optional fractional seconds are allowed: 1995-12-31T23:59:59.999Z
         All other components are mandatory.
         Expressions can also be used to denote calculations that should be
         performed relative to "NOW" to determine the value, ie...
               NOW/HOUR
                  ... Round to the start of the current hour
               NOW-1DAY
                  ... Exactly 1 day prior to now
               NOW/DAY+6MONTHS+3DAYS
                  ... 6 months and 3 days in the future from the start of
                      the current day

         Consult the TrieDateField javadocs for more information.
         Note: For faster range queries, consider the tdate type
      -->
    <fieldType name="date" class="solr.TrieDateField" precisionStep="0"
positionIncrementGap="0"/>
    <!-- A Trie based date field for faster date range queries and date faceting.
-->
    <fieldType name="tdate" class="solr.TrieDateField" precisionStep="6"
positionIncrementGap="0"/>

    <!--Binary data type. The data should be sent/retrieved in as Base64 encoded
Strings -->
```

```
    <fieldType name="binary" class="solr.BinaryField"/>
    <!-- The "RandomSortField" is not used to store or search any
         data.  You can declare fields of this type it in your schema
         to generate pseudo-random orderings of your docs for sorting
         or function purposes.  The ordering is generated based on the field
         name and the version of the index. As long as the index version
         remains unchanged, and the same field name is reused,
         the ordering of the docs will be consistent.
         If you want different psuedo-random orderings of documents,
         for the same version of the index, use a dynamicField and
         change the field name in the request.
     -->
    <fieldType name="random" class="solr.RandomSortField" indexed="true" />
    <!-- solr.TextField allows the specification of custom text analyzers
         specified as a tokenizer and a list of token filters. Different
         analyzers may be specified for indexing and querying.
         The optional positionIncrementGap puts space between multiple fields of
         this type on the same document, with the purpose of preventing false
phrase
         matching across fields.
         For more info on customizing your analyzer chain, please see
         http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters
     -->
    <!-- One can also specify an existing Analyzer class that has a
         default constructor via the class attribute on the analyzer element.
         Example:
    <fieldType name="text_greek" class="solr.TextField">
      <analyzer class="org.apache.lucene.analysis.el.GreekAnalyzer"/>
    </fieldType>
    -->
    <!-- A text field that only splits on whitespace for exact matching of words
-->
    <fieldType name="text_ws" class="solr.TextField" positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      </analyzer>
    </fieldType>
    <!-- A text type for English text where stopwords and synonyms are managed
using the REST API -->
    <fieldType name="managed_en" class="solr.TextField"
positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.StandardTokenizerFactory"/>
        <filter class="solr.ManagedStopFilterFactory" managed="english" />
        <filter class="solr.ManagedSynonymFilterFactory" managed="english" />
      </analyzer>
    </fieldType>

 <fieldType name="text" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <filter class="solr.WordDelimiterFilterFactory"
    generateWordParts="1" generateNumberParts="1" catenateWords="1"
    catenateNumbers="1" catenateAll="0" splitOnCaseChange="0" />
   <filter class="solr.StopFilterFactory" ignoreCase="true"
words="stopwords.txt"/>
  </analyzer>
 </fieldType>
```

```
    <!-- A general text field that has reasonable, generic
         cross-language defaults: it tokenizes with StandardTokenizer,
  removes stop words from case-insensitive "stopwords.txt"
  (empty by default), and down cases.  At query time only, it
  also applies synonyms. -->
    <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" />
        <!-- in this example, we will only use synonyms at query time
        <filter class="solr.SynonymFilterFactory" synonyms="index_synonyms.txt"
ignoreCase="true" expand="false"/>
        -->
        <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
      <analyzer type="query">
        <tokenizer class="solr.StandardTokenizerFactory"/>
        <filter class="solr.StopFilterFactory" ignoreCase="true"
words="stopwords.txt" />
        <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt"
ignoreCase="true" expand="true"/>
        <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
    </fieldType>
    <!-- A text field with defaults appropriate for English: it
         tokenizes with StandardTokenizer, removes English stop words
         (lang/stopwords_en.txt), down cases, protects words from protwords.txt,
and
         finally applies Porter's stemming.  The query time analyzer
         also applies synonyms from synonyms.txt. -->
    <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.
        -->
        <filter class="solr.StopFilterFactory"
                ignoreCase="true"
                words="lang/stopwords_en.txt"
                />
        <filter class="solr.LowerCaseFilterFactory"/>
  <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="en" coreName="${solr.core.name}"/>
     <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"
                />
        <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>
    <!-- A text field with defaults appropriate for English, plus
  aggressive word-splitting and autophrase features enabled.
  This field is just like text_en, except it adds
  WordDelimiterFilter to enable splitting and matching of
  words on case-change, alpha numeric boundaries, and
  non-alphanumeric chars.  This means certain compound word
  cases will work, for example query "wi fi" will match
  document "WiFi" or "wi-fi".
        -->
    <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.
        -->
        <filter class="solr.StopFilterFactory"
                ignoreCase="true"
                words="lang/stopwords_en.txt"
                />
        <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"
                />
        <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>
    <!-- Less flexible matching, but less false matches.  Probably not ideal for
product names,
         but may be good for SKUs.  Can insert dashes in the wrong place and
still match. -->
    <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>
    <!-- Just like text_general except it reverses the characters of
  each token, to enable more efficient leading wildcard queries. -->
    <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" />
        <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" />
        <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
    </fieldType>
    <!-- charFilter + WhitespaceTokenizer  -->
    <!--
    <fieldType name="text_char_norm" class="solr.TextField"
positionIncrementGap="100" >
      <analyzer>
        <charFilter class="solr.MappingCharFilterFactory"
```

```
mapping="mapping-ISOLatin1Accent.txt"/>
        <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      </analyzer>
    </fieldType>
    -->
    <!-- This is an example of using the KeywordTokenizer along
         With various TokenFilterFactories to produce a sortable field
         that does not include some properties of the source text
      -->
    <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://docs.oracle.com/javase/7/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 playload
     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>
    <!-- lowercases the entire field value, keeping it as a single token.  -->
    <fieldType name="lowercase" class="solr.TextField"
positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.KeywordTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory" />
      </analyzer>
    </fieldType>
    <!--
      Example of using PathHierarchyTokenizerFactory at index time, so
      queries for paths match documents at that path, or in descendent paths
    -->
    <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>
    <!--
      Example of using PathHierarchyTokenizerFactory at query time, so
      queries for paths match documents at that path, or in ancestor paths
    -->
    <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="path" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
   <filter class="solr.TrimFilterFactory" />
   <tokenizer class="solr.PathHierarchyTokenizerFactory" delimiter="/" />
  </analyzer>
  <analyzer type="query">
   <tokenizer class="solr.KeywordTokenizerFactory" />
   <filter class="solr.TrimFilterFactory" />
  </analyzer>
  <!-- Disable TF/IDF scoring for string fields so we can use them for boosting
  <similarity
class="de.hybris.platform.solrfacetsearch.ysolr.similarity.FixedTFIDFSimilarityFa
ctory" />-->
 </fieldType>
    <!-- since fields of this type are by default not stored or indexed,
         any data added to them will be ignored outright.  -->
    <fieldType name="ignored" stored="false" indexed="false" multiValued="true"
class="solr.StrField" />
```

```
    <!-- This point type indexes the coordinates as separate fields (subFields)
      If subFieldType is defined, it references a type, and a dynamic field
      definition is created matching *___<typename>.  Alternately, if
      subFieldSuffix is defined, that is used to create the subFields.
      Example: if subFieldType="double", then the coordinates would be
        indexed in fields myloc_0___double,myloc_1___double.
      Example: if subFieldSuffix="_d" then the coordinates would be indexed
        in fields myloc_0_d,myloc_1_d
      The subFields are an implementation detail of the fieldType, and end
      users normally should not need to know about them.
     -->
    <fieldType name="point" class="solr.PointType" dimension="2"
subFieldSuffix="_d"/>
    <!-- A specialized field for geospatial search. If indexed, this fieldType
must not be multivalued. -->
    <fieldType name="location" class="solr.LatLonType"
subFieldSuffix="_coordinate"/>
    <!-- An alternative geospatial field type new to Solr 4.  It supports
multiValued and polygon shapes.
      For more information about this and other Spatial fields new to Solr 4,
see:
      http://wiki.apache.org/solr/SolrAdaptersForLuceneSpatial4
    -->
    <fieldType name="location_rpt"
class="solr.SpatialRecursivePrefixTreeFieldType"
        geo="true" distErrPct="0.025" maxDistErr="0.001"
distanceUnits="kilometers" />
    <!-- Spatial rectangle (bounding box) field. It supports most spatial
predicates, and has
     special relevancy modes: score=overlapRatio|area|area2D (local-param to the
query).  DocValues is recommended for
     relevancy. -->
    <fieldType name="bbox" class="solr.BBoxField"
               geo="true" distanceUnits="kilometers" numberType="_bbox_coord" />
    <fieldType name="_bbox_coord" class="solr.TrieDoubleField" precisionStep="8"
docValues="true" stored="false"/>
   <!-- Money/currency field type. See http://wiki.apache.org/solr/MoneyFieldType
        Parameters:
          defaultCurrency: Specifies the default currency if none specified.
Defaults to "USD"
          precisionStep:   Specifies the precisionStep for the TrieLong field
used for the amount
          providerClass:   Lets you plug in other exchange provider backend:
                           solr.FileExchangeRateProvider is the default and takes
one parameter:
                             currencyConfig: name of an xml file holding exchange
rates
                           solr.OpenExchangeRatesOrgProvider uses rates from
openexchangerates.org:
                             ratesFileLocation: URL or path to rates JSON file
(default latest.json on the web)
                             refreshInterval: Number of minutes between each
rates fetch (default: 1440, min: 60)
   -->
    <fieldType name="currency" class="solr.CurrencyField" precisionStep="8"
defaultCurrency="USD" currencyConfig="currency.xml" />


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

```
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
  </analyzer>
 </fieldType>
 <fieldType name="text_spell_en" class="solr.TextField"
positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <filter class="solr.PatternReplaceFilterFactory" pattern="(['’])"
replacement=" " />
   <filter class="solr.EnglishMinimalStemFilterFactory" />
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="en" coreName="${solr.core.name}"/>
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="en" coreName="${solr.core.name}"/>
   <filter class="solr.StopFilterFactory" words="lang/stopwords_en.txt"
ignoreCase="true" />
   <filter class="solr.TrimFilterFactory" />
   <filter class="solr.RemoveDuplicatesTokenFilterFactory" />
  </analyzer>
 </fieldType>
 <fieldType name="text_spell_de" class="solr.TextField"
positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <filter class="solr.GermanLightStemFilterFactory" />
   <!--
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="de" coreName="${solr.core.name}"/>
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="de" coreName="${solr.core.name}"/>-->
   <filter class="solr.StopFilterFactory" words="lang/stopwords_de.txt"
ignoreCase="true" />
   <filter class="solr.TrimFilterFactory" />
   <filter class="solr.RemoveDuplicatesTokenFilterFactory" />
  </analyzer>
 </fieldType>

 <fieldType name="text_spell_zh" class="solr.TextField"
positionIncrementGap="100">
  <analyzer>
   <tokenizer class="com.chenlb.mmseg4j.solr.MMSegTokenizerFactory"
mode="complex" dicPath="dic"/>
  </analyzer>
 </fieldType>
 <fieldType name="text_spell_fr" class="solr.TextField"
positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <filter class="solr.FrenchMinimalStemFilterFactory" />
   <!--
```

```
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="fr" coreName="${solr.core.name}"/>
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="fr" coreName="${solr.core.name}"/>-->
   <filter class="solr.StopFilterFactory" words="lang/stopwords_fr.txt"
ignoreCase="true" />
   <filter class="solr.TrimFilterFactory" />
   <filter class="solr.RemoveDuplicatesTokenFilterFactory" />
  </analyzer>
 </fieldType>
 <fieldType name="text_spell_pt" class="solr.TextField"
positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <filter class="solr.PortugueseMinimalStemFilterFactory" />
   <!--
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="pt" coreName="${solr.core.name}"/>
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="pt" coreName="${solr.core.name}"/>-->
   <filter class="solr.StopFilterFactory" words="lang/stopwords_pt.txt"
ignoreCase="true" />
   <filter class="solr.TrimFilterFactory" />
   <filter class="solr.RemoveDuplicatesTokenFilterFactory" />
  </analyzer>
 </fieldType>


 <fieldType name="text_spell_it" class="solr.TextField"
positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <filter class="solr.ItalianLightStemFilterFactory" />
   <!--
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="it" coreName="${solr.core.name}"/>
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="it" coreName="${solr.core.name}"/>-->
   <filter class="solr.StopFilterFactory" words="lang/stopwords_it.txt"
ignoreCase="true" />
   <filter class="solr.TrimFilterFactory" />
   <filter class="solr.RemoveDuplicatesTokenFilterFactory" />
  </analyzer>
 </fieldType>

 <fieldType name="text_de" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.StandardFilterFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <!--
```

```
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="de" coreName="${solr.core.name}"/>
   <filter class="solr.WordDelimiterFilterFactory"
    generateWordParts="1" generateNumberParts="1" catenateWords="1"
    catenateNumbers="1" catenateAll="0" splitOnCaseChange="0" />
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="de" coreName="${solr.core.name}"/>-->
   <filter class="solr.StopFilterFactory" words="lang/stopwords_de.txt"
ignoreCase="true" />
   <filter class="solr.ASCIIFoldingFilterFactory" />
   <filter class="solr.SnowballPorterFilterFactory" language="German2" />
  </analyzer>
 </fieldType>
 <fieldType name="text_fr" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.StandardFilterFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <!--
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="fr" coreName="${solr.core.name}"/>
   <filter class="solr.WordDelimiterFilterFactory"
    generateWordParts="1" generateNumberParts="1" catenateWords="1"
    catenateNumbers="1" catenateAll="0" splitOnCaseChange="0" />
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="fr" coreName="${solr.core.name}"/>-->
   <filter class="solr.StopFilterFactory" words="lang/stopwords_fr.txt"
ignoreCase="true" />
   <filter class="solr.ASCIIFoldingFilterFactory" />
   <filter class="solr.SnowballPorterFilterFactory" language="French" />
  </analyzer>
 </fieldType>

 <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.

           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="userdict_ja.txt"/>-->
        <!-- Reduces inflected verbs and adjectives to their base/dictionary
forms () -->
        <filter class="solr.JapaneseBaseFormFilterFactory"/>
        <!-- Removes tokens with certain part-of-speech tags -->
        <filter class="solr.JapanesePartOfSpeechStopFilterFactory"
tags="lang/stoptags_ja.txt"/>
        <!-- Normalizes full-width romaji to half-width and half-width kana to
full-width (Unicode NFKC subset) -->
        <filter class="solr.CJKWidthFilterFactory"/>
        <!-- Hybris Filters
        <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="ja" coreName="${solr.core.name}" mode="search"
tokenizerFactory ="solr.JapaneseTokenizerFactory"/>
        <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="ja" coreName="${solr.core.name}" mode="search"
tokenizerFactory ="solr.JapaneseTokenizerFactory"/>-->
        <!-- 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"/>
        <!-- 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>
 <fieldType name="text_zh" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
   <tokenizer class="com.chenlb.mmseg4j.solr.MMSegTokenizerFactory"
mode="complex" dicPath="dic"/>
   <!--
   <tokenizer class="solr.StandardTokenizerFactory"/>
       <filter class="solr.CJKWidthFilterFactory"/>
```

```
       <filter class="solr.LowerCaseFilterFactory"/>
       <filter class="solr.CJKBigramFilterFactory"
         han="true" hiragana="true"
         katakana="true" hangul="true" outputUnigrams="false" />

   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="zh" coreName="${solr.core.name}"/>-->
  </analyzer>
 </fieldType>
 <fieldType name="text_pt" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.StandardFilterFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <!--
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="pt" coreName="${solr.core.name}"/>
   <filter class="solr.WordDelimiterFilterFactory"
    generateWordParts="1" generateNumberParts="1" catenateWords="1"
    catenateNumbers="1" catenateAll="0" splitOnCaseChange="0" />
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="pt" coreName="${solr.core.name}"/>-->
   <filter class="solr.StopFilterFactory" words="lang/stopwords_pt.txt"
ignoreCase="true" />
   <filter class="solr.ASCIIFoldingFilterFactory" />
   <filter class="solr.BrazilianStemFilterFactory"/>
  </analyzer>
 </fieldType>
 <fieldType name="text_it" class="solr.TextField" positionIncrementGap="100">
  <analyzer>
   <tokenizer class="solr.StandardTokenizerFactory" />
   <filter class="solr.StandardFilterFactory" />
   <filter class="solr.LowerCaseFilterFactory" />
   <!--
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.synonyms.HybrisSynonymFilterFacto
ry" ignoreCase="true" synonyms="it" coreName="${solr.core.name}"/>
   <filter class="solr.WordDelimiterFilterFactory"
    generateWordParts="1" generateNumberParts="1" catenateWords="1"
    catenateNumbers="1" catenateAll="0" splitOnCaseChange="0" />
   <filter
class="de.hybris.platform.solrfacetsearch.ysolr.stopwords.HybrisStopWordsFilterFa
ctory" ignoreCase="true" lang="it" coreName="${solr.core.name}"/>-->
   <filter class="solr.StopFilterFactory" words="lang/stopwords_it.txt"
ignoreCase="true" />
   <filter class="solr.ASCIIFoldingFilterFactory" />
   <filter class="solr.SnowballPorterFilterFactory" language="Italian"/>
  </analyzer>
 </fieldType>
 <fieldType name="sortabletext" 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" />
     </analyzer>
 </fieldType>


<!-- Similarity is the scoring routine for each document vs. a query.
     A custom Similarity or SimilarityFactory may be specified here, but
     the default is fine for most applications.
     For more info: http://wiki.apache.org/solr/SchemaXml#Similarity
  -->
<!--
   <similarity class="com.example.solr.CustomSimilarityFactory">
     <str name="paramkey">param value</str>
   </similarity>
  -->
```

```
</schema>
```

- unit test

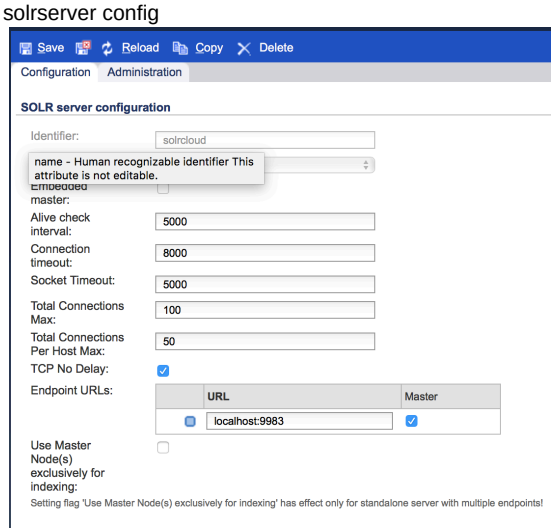
2) solrcloud configuration

SolrCloud command

```
# start solrcloud node1 with embedded zk
bin/solr start -cloud -s server/solr -p 8983

# start solrcloud node2
bin/solr start -cloud -s server/solr2 -p 8984 -z localhost:9983

# stop all nodes
bin/solr stop -all

# create a collection
bin/solr create -c electronics_Product -d
/Users/i306724/Downloads/solr-5.2.1/server/solr/configsets -n hybris -p 8983 -s 1 -rf
2

# delete a collection
bin/solr delete -c electronics_Product

# upload config to zk
sh zkcli.sh -cmd upconfig -zkhost localhost:9983 -confname hybris -confdir
/Users/i306724/Downloads/solr-5.2.1/server/solr/configsets/hybris/conf
```

3) solrserver config

1. index mode = DIRECT
2. solrserver config



Verification

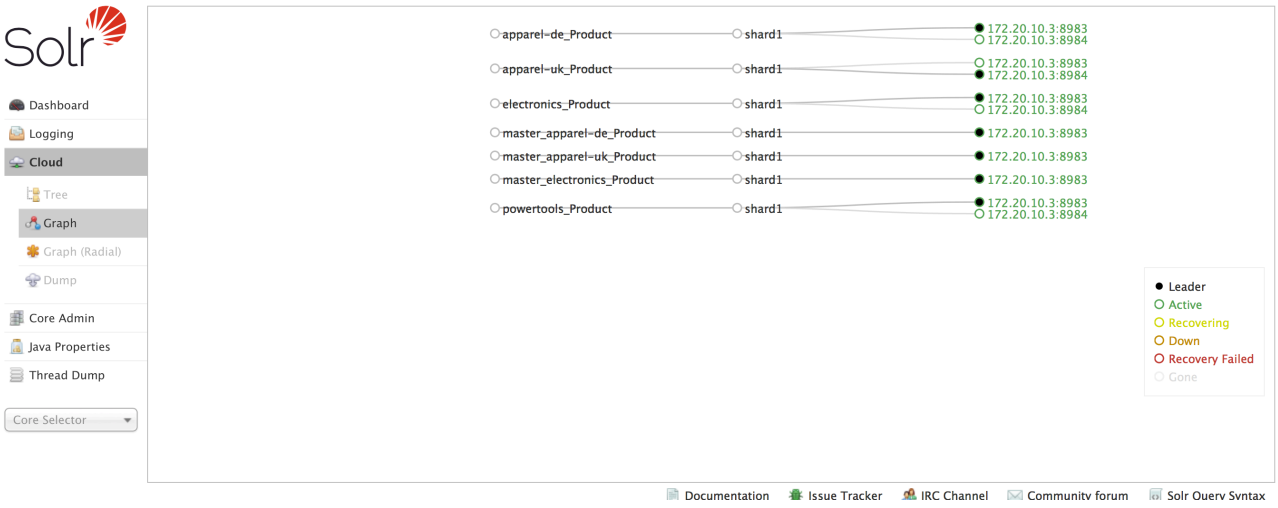
1) Funcations

No.	Item	Verify	Memo
1	auto suggestion	✅	no builtin, need to customise DefaultSolrAutoSuggestService, `DefaultSolrProductSearchAutocompleteService.findBestSuggestions`
2	spell check	✅	
3	free text search	✅	
4	category search	✅	
5	commerce search	✅	
6	boost	✅	
7	hero product	✅	
8	searchandizing	✅	
9	Chinese segmentation (mmseg/jcseg)	✅	
10	synonyms	TBC	
11	stopwords	TBC	
12			
13	index (full, update, delete)	✅	
14	two phase index	❓	embedded not ok, standalone ok, cloud no? can be customised with **collection alias** in SolrCloud
15	lucenesearch	✅	➖ need to update same version, updated lucene to 5.2.1

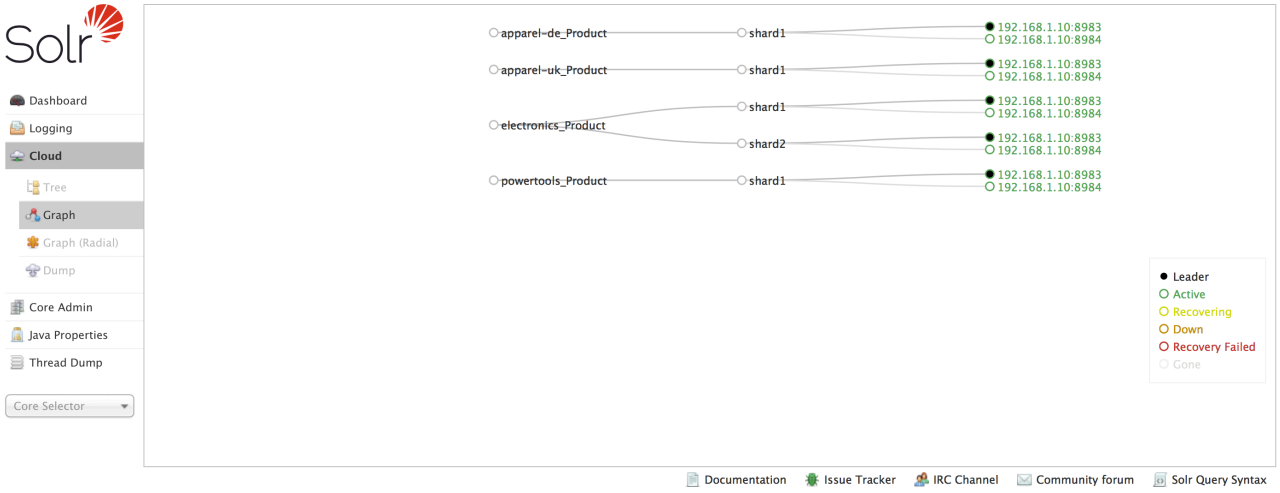
2) NFR – scalability & high availability

No.	NFR	Scalability	High Availability	Memo
1	Stop non-zk nodes		✅	reconnect when start again, highly recommended to install standalone ZK ensemble
2	Stop zk nodes		➖	
3	scale nodes			shards re-balancing, numShards
4	add replicas			replicas, replicationFactor
5	partial availability			
6	resizing			
7	Near Realtime Search			
8				

3) Index in solrcloud

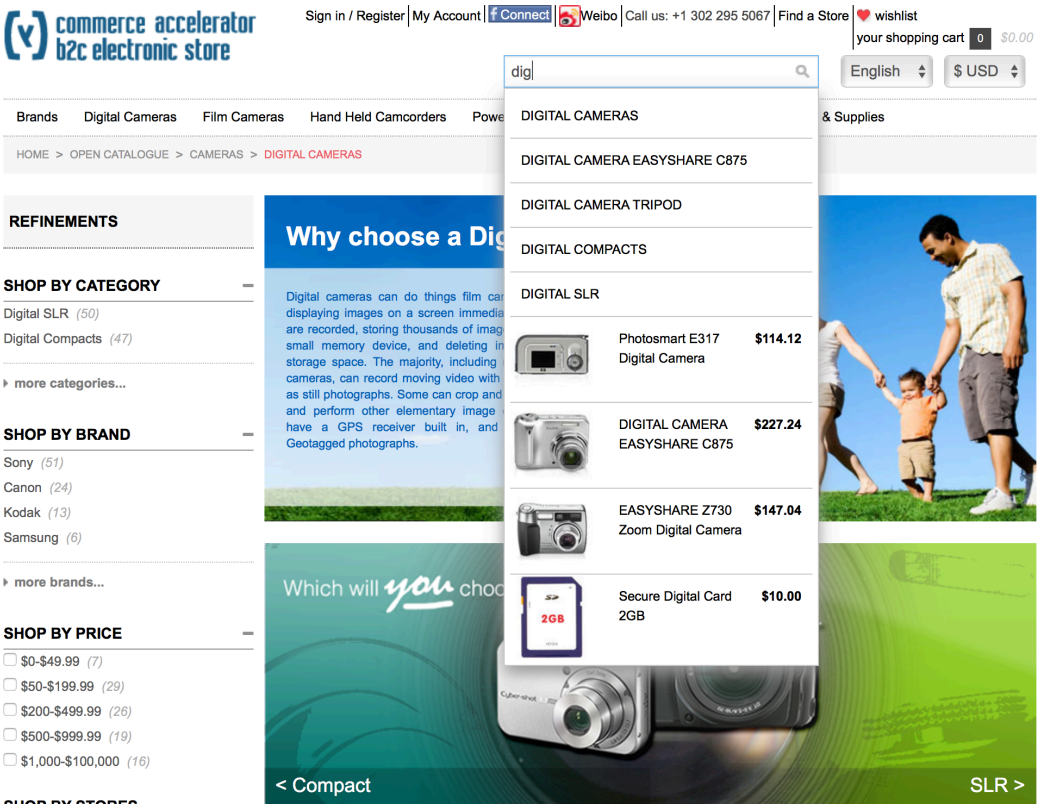


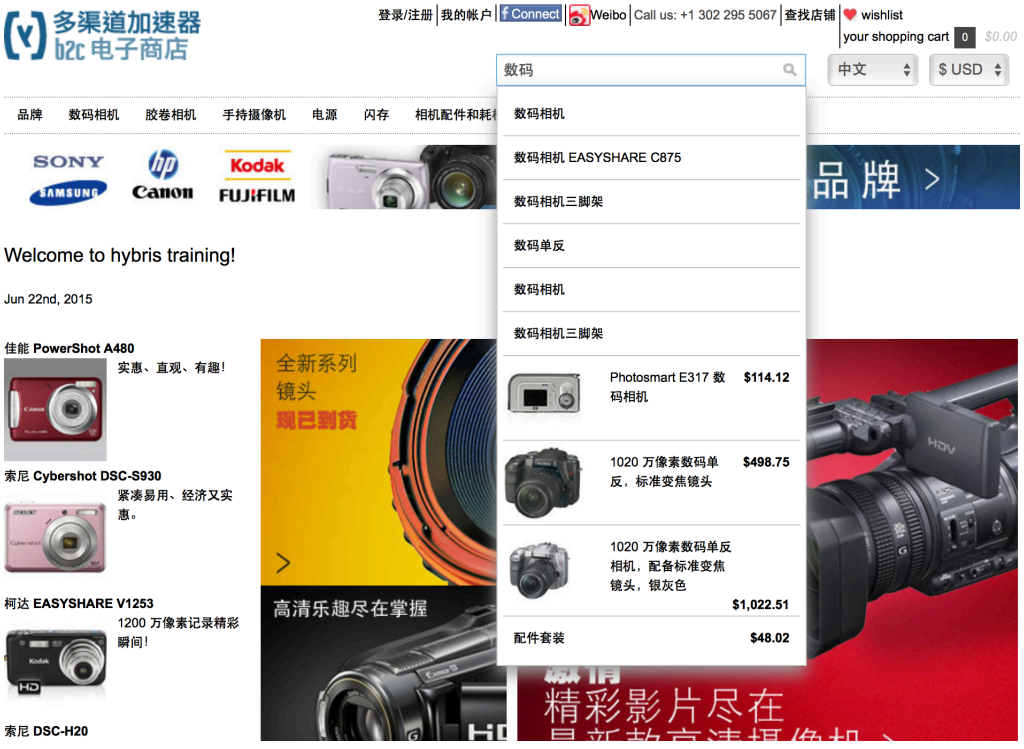
all collections with 1 shard



electronics_product with 2 shards.

4) Auto suggestion





Open Issues

Item	Status	Memo
Embedded 5.2.1	?	indexing seems ok, but no core created

Standalone 5.2.1	✅	
Cloud 5.2.1	✅	

References:

1. https://cwiki.apache.org/confluence/display/solr/Major+Changes+from+Solr+4+to+Solr+5
2. https://cwiki.apache.org/confluence/display/solr/SolrCloud