```
         of documents requested.  -->
    <queryResultCache class="solr.LRUCache" size="512" initialSize="512" autowarmCount="32"/>
    <!-- documentCache 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"/>
    <!-- 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>
    <!-- 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.search.CacheRegenerator if autowarming is desired.  -->
    <!--
    <cache name="myUserCache"
      class="solr.LRUCache"
      size="4096"
      initialSize="1024"
      autowarmCount="1024"
      regenerator="org.mycompany.mypackage.MyRegenerator"
      />
    -->
    <!-- An 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.
    <useFilterForSortedQuery>true</useFilterForSortedQuery>
   -->
    <!-- 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>50</queryResultWindowSize>
    <!-- Maximum number of documents to cache for any entry in the
         queryResultCache. -->
    <queryResultMaxDocsCached>200</queryResultMaxDocsCached>
    <!-- This entry enables an int hash representation for filters (DocSets)
         when the number of items in the set is less than maxSize.  For smaller
         sets, this representation is more memory efficient, more efficient to
         iterate over, and faster to take intersections.  -->
    <HashDocSet maxSize="3000" loadFactor="0.75"/>
    <!-- a newSearcher event is fired whenever a new searcher is being prepared
         and there is a current searcher handling requests (aka registered). -->
    <!-- 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="start">0</str>
          <str name="rows">10</str>
        </lst>
        <lst>
          <str name="q">rocks</str>
          <str name="start">0</str>
          <str name="rows">10</str>
        </lst>
        <lst>
          <str name="q">static newSearcher warming query from solrconfig.xml</str>
        </lst>
      </arr>
    </listener>
    <!-- a firstSearcher event is fired whenever a new searcher is being
         prepared but there is no current registered searcher to handle
         requests or to gain autowarming data from. -->
    <listener event="firstSearcher" class="solr.QuerySenderListener">
      <arr name="queries">
        <lst>
          <str name="q">fast_warm</str>
          <str name="start">0</str>
          <str name="rows">10</str>
        </lst>
        <lst>
          <str name="q">static firstSearcher warming query from solrconfig.xml</str>
        </lst>
      </arr>
    </listener>
    <!-- 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>
    <!-- Maximum number of searchers that may be warming in the background
      concurrently.  An error is returned if this limit is exceeded. Recommend
      1-2 for read-only slaves, higher for masters w/o cache warming. -->
    <maxWarmingSearchers>2</maxWarmingSearchers>
  </query>
  <!--
    Let the dispatch filter handler /select?qt=XXX
```