BroadLeaf project search function improvement

Broadleaf Commerce is an open source Java e-commerce platform based on the Spring framework development that provides a reliable, scalable architecture for deep customization and rapid development.

About Solr

The search for goods in the Broadleaf project uses the embedded Solr server, which can be seen in the configuration file.

- Project homepage: http://www.broadleafcommerce.com/
- Sample website: http://demo.broadleafcommerce.org/
- Sample website source code: https://github.com/BroadleafCommerce/DemoSite

From the sample website source codeapplicationContext.xml fileYou can see the configuration about solr:

```
1        <bean id="solrEmbedded" class="java.lang.String">
```

```
        <constructor-arg value="solrhome"/>
    </bean>

    <bean id="blSearchService" class="org.broadleafcommerce.core.search.service.sol
        <constructor-arg name="solrServer" ref="${solr.source}" />
        <constructor-arg name="reindexServer" ref="${solr.source.reindex}" />
    </bean>
    <bean id="rebuildIndexJobDetail" class="org.springframework.scheduling.quartz.
        <property name="targetObject" ref="blSearchService" />
        <property name="targetMethod" value="rebuildIndex" />
    </bean>
    <bean id="rebuildIndexTrigger" class="org.springframework.scheduling.quartz.Si
        <property name="jobDetail" ref="rebuildIndexJobDetail" />
        <property name="startDelay" value="${solr.index.start.delay}" />
        <property name="repeatInterval" value="${solr.index.repeat.interval}" />
    </bean>
    <bean class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
        <property name="triggers">
            <list>
                <ref bean="rebuildIndexTrigger" />
                <!--<ref bean="purgeCartTrigger" />-->
                <!--<ref bean="purgeCustomerTrigger" />-->
            </list>
        </property>
    </bean>
```

Resource configuration file atcommon.properties:

```
web.defaultPageSize=15
web.maxPageSize=100

solr.source=solrEmbedded
solr.source.reindex=solrEmbedded
solr.index.start.delay=5000
solr.index.repeat.interval=3600000
```

It can be seen from the above that Solr is an embedded service, and the Solr configuration files (schema.xml and solrconfig.xml) arehttps://github.com/BroadleafCommerce/DemoSite/tree/master/site/src/main/resources Under contents.

As can be seen from the source code SolrSearchServiceImpl.java, a total o

services are started, corresponding to the primary and reindex two solrcore, the primary is used for query, and the reindex is used to rebuild the index.

Improved search engine

Improvement goal

This article just replaces the embedded Solr service with a standalone Solr service, and you can switch to a SolrCloud cluster.

- Separate search engine server
- Support for incremental update indexing
- Support manual rebuild index

Design ideas

- 1. Modify the embedded search engine in the original system as an independently deployed search engine. The installation method is described below.
- 2. Expand the SolrSearchServiceImpl class in the original system and add a method to increase the index.
- 3. Modify the service class of the product in the original system (I call LLCatalogServiceImpl, this class is newly added), add the method to add index to the search engine in the saveProduct method.
- 4. Modify the configuration file related to solr in the original system.
- 5. Modify the timing task of the rebuild index in the original system to support manual rebuilding of the index.

Implementation

1, build a stand-alone solr server

You can refer to:Introduction and installation of Apache Solr

The key lies in the definition of solr/home, and the creation of two directories in the directory, primary and reindex, the configuration files in the two directories are the same, the solr/home directory structure is as follows:

```
1    solrhome-solr  tree -L 3
```

```
. 3 ├── primary
4 │   └── conf
5 │       ├── schema.xml
6 │       └── solrconfig.xml
7 ├── reindex
8 │   └── conf
9 │       ├── schema.xml
10 │       └── solrconfig.xml
11 └── solr.xml
12
13 4 directories, 5 files
```

The contents of the solr.xml file are as follows:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <solr persistent="true">
3   <cores defaultCoreName="primary" adminPath="/admin/cores">
4     <core instanceDir="reindex" name="reindex"/>
5     <core instanceDir="primary" name="primary"/>
6   </cores>
7 </solr>
```

Schema.xml content andThe originalBasically the same, just added a line:

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

The contents of solrconfig.xml are as follows:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <config>
3   <luceneMatchVersion>4.4</luceneMatchVersion>
4   <directoryFactory name="DirectoryFactory" class="${solr.directoryFactory:solr.St
5
6   <schemaFactory class="ClassicIndexSchemaFactory"/>
7
8   <updateHandler class="solr.DirectUpdateHandler2">
9         <autoCommit>
10                <maxDocs>2</maxDocs>
11                <maxTime>3000</maxTime>
```

```
        </autoCommit>
    </updateHandler>

  <requestHandler name="/get" class="solr.RealTimeGetHandler">
    <lst name="defaults">
      <str name="omitHeader">true</str>
    </lst>
  </requestHandler>

  <requestHandler name="/replication" class="solr.ReplicationHandler" startup="laz

  <requestDispatcher handleSelect="true" >
    <requestParsers enableRemoteStreaming="false" multipartUploadLimitInKB="2048"
    <httpCaching never304="true" />
  </requestDispatcher>

  <requestHandler name="standard" class="solr.StandardRequestHandler" default="tru
  <requestHandler name="/analysis/field" startup="lazy" class="solr.FieldAnalysisR
  <requestHandler name="/update" class="solr.UpdateRequestHandler"  />
  <requestHandler name="/update/csv" class="solr.CSVRequestHandler" startup="lazy"
  <requestHandler name="/update/json" class="solr.JsonUpdateRequestHandler" startu
  <requestHandler name="/admin/" class="org.apache.solr.handler.admin.AdminHandler

  <requestHandler name="/admin/ping" class="solr.PingRequestHandler">
    <lst name="invariants">
      <str name="q">solrpingquery</str>
    </lst>
    <lst name="defaults">
      <str name="echoParams">all</str>
    </lst>
  </requestHandler>

  <queryResponseWriter name="json" class="solr.JSONResponseWriter">
        <str name="content-type">text/plain; charset=UTF-8</str>
  </queryResponseWriter>

  <!-- config for the admin interface -->
  <admin>
    <defaultQuery>solr</defaultQuery>
  </admin>
</config>
```

2, expand the SolrSearchServiceImpl class

Create the org.broadleafcommerce.core.search.service.solr.ExtSolrSearchService interface in the core module, which is defined as follows:

```
1  package org.broadleafcommerce.core.search.service.solr;
2  import java.io.IOException;
3  import org.broadleafcommerce.common.exception.ServiceException;
4  import org.broadleafcommerce.core.catalog.domain.Product;
5  import org.broadleafcommerce.core.search.service.SearchService;
6
7  public interface ExtSolrSearchService extends SearchService {
8      public void addProductIndex(Product product) throws ServiceException,
9              IOException;
10 }
```

Then, create its implementation class org.broadleafcommerce.core.search.service.solr.ExtSolrSearchServiceImpl, which is defined as follows:

```
1  package org.broadleafcommerce.core.search.service.solr;
2  import java.io.IOException;
3  import java.util.ArrayList;
4  import java.util.Collections;
5  import java.util.Comparator;
6  import java.util.List;
7  import javax.annotation.Resource;
8  import javax.xml.parsers.ParserConfigurationException;
9  import org.apache.commons.logging.Log;
10 import org.apache.commons.logging.LogFactory;
11 import org.apache.solr.client.solrj.SolrServer;
12 import org.apache.solr.client.solrj.SolrServerException;
13 import org.apache.solr.client.solrj.response.QueryResponse;
14 import org.apache.solr.common.SolrDocument;
15 import org.apache.solr.common.SolrDocumentList;
16 import org.apache.solr.common.SolrInputDocument;
17 import org.broadleafcommerce.common.exception.ServiceException;
18 import org.broadleafcommerce.common.locale.domain.Locale;
19 import org.broadleafcommerce.common.util.StopWatch;
20 import org.broadleafcommerce.common.util.TransactionUtils;
21 import org.broadleafcommerce.core.catalog.domain.Product;
22 import org.broadleafcommerce.core.search.domain.Field;
```

```
23 import org.springframework.transaction.TransactionDefinition; 24
import org.springframework.transaction.TransactionStatus; 25
import org.xml.sax.SAXException; 26
27 public class ExtSolrSearchServiceImpl extends SolrSearchServiceImpl implements
28         ExtSolrSearchService {
29     private static final Log LOG = LogFactory
30             .getLog(ExtSolrSearchServiceImpl.class);
31     @Resource(name = "blSolrIndexService")
32     protected SolrIndexServiceImpl solrIndexServiceImpl;
33     public ExtSolrSearchServiceImpl(SolrServer solrServer,
34             SolrServer reindexServer) {
35         super(solrServer, reindexServer);
36     }
37     public ExtSolrSearchServiceImpl(SolrServer solrServer) {
38         super(solrServer);
39     }
40     public ExtSolrSearchServiceImpl(String solrServer, String reindexServer)
41             throws IOException, ParserConfigurationException, SAXException {
42         super(solrServer, reindexServer);
43     }
44     public ExtSolrSearchServiceImpl(String solrServer) throws IOException,
45             ParserConfigurationException, SAXException {
46         super(solrServer);
47     }
48     public void addProductIndex(Product product) throws ServiceException,
49             IOException {
50         TransactionStatus status = TransactionUtils.createTransaction(
51                 "saveProduct", TransactionDefinition.PROPAGATION_REQUIRED,
52                 solrIndexServiceImpl.transactionManager, true);
53         StopWatch s = new StopWatch();
54         try {
55             List<Field> fields = fieldDao.readAllProductFields();
56             List<Locale> locales = solrIndexServiceImpl.getAllLocales();
57             SolrInputDocument document = solrIndexServiceImpl.buildDocument(
58                     product, fields, locales);
59             if (LOG.isTraceEnabled()) {
60                 LOG.trace(document);
61             }
62             SolrContext.getServer().add(document);
63             SolrContext.getServer().commit();
64             TransactionUtils.finalizeTransaction(status,
65                     solrIndexServiceImpl.transactionManager, false);
66         } catch (SolrServerException e) {
```

```
            TransactionUtils.finalizeTransaction(status,
                    solrIndexServiceImpl.transactionManager, true);
            throw new ServiceException("Could not rebuild index", e);
        } catch (IOException e) {
            TransactionUtils.finalizeTransaction(status,
                    solrIndexServiceImpl.transactionManager, true);
            throw new ServiceException("Could not rebuild index", e);
        } catch (RuntimeException e) {
            TransactionUtils.finalizeTransaction(status,
                    solrIndexServiceImpl.transactionManager, true);
            throw e;
        }
        LOG.info(String.format("Finished adding index in %s", s.toLapString()));
    }
    protected List<Product> getProducts(QueryResponse response) {
        final List<Long> productIds = new ArrayList<Long>();
        SolrDocumentList docs = response.getResults();
        for (SolrDocument doc : docs) {
            productIds
                    .add((Long) doc.getFieldValue(shs.getProductIdFieldName()));
        }
        /**
         * TODO Please add a cache related code
         */
        List<Product> products = productDao.readProductsByIds(productIds);
        // We have to sort the products list by the order of the productIds list
        // to maintain sortability in the UI
        if (products != null) {
            Collections.sort(products, new Comparator<Product>() {
                public int compare(Product o1, Product o2) {
                    return new Integer(productIds.indexOf(o1.getId()))
                            .compareTo(productIds.indexOf(o2.getId()));
                }
            });
        }
        return products;
    }
}
```

3, modify the solr related configuration file

a. Delete the following code in /web/src/main/webapp/WEB-INF/applicationContext.xml in the web module:

```
<bean id="blSearchService" class="org.broadleafcommerce.core.search.service.solr.S
    <constructor-arg name="solrServer" ref="${solr.source}" />
    <constructor-arg name="reindexServer" ref="${solr.source.reindex}" />
</bean>

```

b. Delete the following code for web/src/main/resources/runtime-properties /common.properties in the web module:

```
solr.source=solrEmbedded
solr.source.reindex=solrEmbedded
```

c. Add the following code in core/src/main/resources/applicationContext.xml in the core module:

```
<bean id="solrServer" class="org.apache.solr.client.solrj.impl.HttpSolrServer">
    <constructor-arg value="${solr.url}" />
</bean>

<bean id="solrReindexServer" class="org.apache.solr.client.solrj.impl.HttpSolrServer
    <constructor-arg value="${solr.url.reindex}" /> 6 </bean>
<bean id="blSearchService"

 class="org.broadleafcommerce.core.search.service.solr.ExtSolrSearchServiceImpl">
    <constructor-arg name="solrServer" ref="${solr.source}" />10
 <constructor-arg name="reindexServer" ref="${solr.source.reindex}" />11 </bean>

```

d. Add the following code in core/src/main/resources/runtime-properties/common-shared.properties in the core module:

```
solr.url=http://localhost:8080/solr
solr.url.reindex=http://localhost:8080/solr/reindex
solr.source=solrServer
solr.source.reindex=solrReindexServer
```

4, modify the LLCatalogServiceImpl class

Add the following code:

```
@Resource(name = "blSearchService")
private ExtSolrSearchService extSolrSearchService;
```

Modify the saveProduct method of this class as follows:

```
@Override
@Transactional("blTransactionManager")
public Product saveProduct(Product product) {
    Product dbProduct = catalogService.saveProduct(product);
    try {
        extSolrSearchService.addProductIndex(dbProduct);
    } catch (ServiceException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    } catch (IOException e) {
        e.printStackTrace();
        throw new RuntimeException(e);
    }
    return dbProduct;
}
```

5, modify the timing task

a. When the web system starts, it will query the products in the database and then rebuild the index. This function has defined a scheduled task in applicationContext.xml. It is recommended to cancel the scheduled task. Remove the following code:

```
<bean id="rebuildIndexJobDetail"

 class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean"
    <property name="targetObject" ref="blSearchService" /> 4
 <property name="targetMethod" value="rebuildIndex" /> 5 </bean>

<bean id="rebuildIndexTrigger" class="org.springframework.scheduling.quartz.SimpleTr
    <property name="jobDetail" ref="rebuildIndexJobDetail" /> 8
 <property name="startDelay" value="${solr.index.start.delay}" /> 9
```

```
    <property name="repeatInterval" value="${solr.index.repeat.interval}" />10
</bean>11 <bean class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
12     <property name="triggers">13         <list>
14             <ref bean="rebuildIndexTrigger" />
15         </list>
16     </property>
17 </bean>
```

b. Write the main method, type it into a jar, and then write a shell script to manually rebuild the index or set up a scheduled task. This class needs to get a bean named blSearchService, and then call the bean's rebuildIndex method, the main code is as follows.

```
1 @Resource(name = "blSearchService")
2 private SearchService extSolrSearchService;
3
4 public void doRebuild(){
5     extSolrSearchService.rebuildIndex();
6 }
```

6, single node integration to create an index problem

a, create an index exception If there are multiple cores in the single-node Solr installation process, the process of creating an index uses the reindex core, if there is no reindex this core may throw the following exception when starting the project:

```
Caused by: org.apache.solr.client.solrj.impl.HttpSolrServer$RemoteSolrException: Server
```

Solution:

Modify the configuration file solr.xml in Solrhome, make sure that the core in the configuration solr contains reindex, and the reindex directory and configuration file in the solrhome directory, as shown below:

```
<core name="reindex" instanceDir="reindex" />
```

b, solr query exception

If there are multiple cores in a single node solr, the default core is primary, the query uses primary, and the index used is reindex. At this time, the web query is still the original primary data, not the index data created.

Solution: Use the core that created the index as the default. Modify solrhome/solr.xml as follows:

```
<cores defaultCoreName="reindex" adminPath="/admin/cores">
```

c, start the project to create an index, solr can not find

Complete single-node integration, start the project for testing, start the completion, after the index is created, query in the single node of solr, and find that no index is added in docs.

Solution:

After debugging the process of adding an index, the following code was found to cause the docs to be added to the index to be deleted. The reason is that the core configured in solr is multiple, the deleteAllDocuments method is executed, so the added indexes are deleted. In the case of multiple cores in solr, the following code should be blocked.

```
//      // Swap the active and the reindex cores
//      shs.swapActiveCores();
//      // If we are not in single core mode, we delete the documents for the
//      // unused core after swapping
//      if (!SolrContext.isSingleCoreMode()) {
//        deleteAllDocuments();
//      }
```

- **Related Posts**
  - BroadLeaf project integrates with SolrCloud
  - Function improvement
  - CRM project dynamic search function
  - Search experience improvement attempt

- Django (2) project improvement
- Project Requirements Discussion - Search function on the title bar
- Sigsuspend function (improvement of mysleep function)
- Project2-3: Lucene search improvement
- Hibernate Search will be integrated into the existing project, the full-text search function
- Android code optimization and project improvement