Index Drupal Content in Islandora's Solr index

NOTE: Indexing Drupal Content in Islandora Solr's index is not required, but it is a feature some users have requested

Indexing Islandora Fedora and Drupal Content in the same Solr index allows sites to simplify searching for end users by allowing them to search all content from one search interface. To search across Drupal and Fedora content you will have to modify your solr schema.xml and you will also need to update the Islandora Solr search results to make them aware of Drupal content, there are a couple of ways to modify the results but they will involve some code modifications.

You will also need to install the Drupal Apache Solr module (https://drupal.org/project/apachesolr). The Drupal Apache Solr module installation says to copy its Solr configuration files to the Solr configuration directory. **Do not do this as you will be overwriting your existing Islandora Solr configuration**. Instead update the existing schema.xml file with the required Drupal fields (it's good practice to backup any files before you edit them if they are not already under a vcs, even better try it on a dev/test install first).

This section assumes you already have the Islandora Solr module configured to search Fedora content and Gsearch configured to index Fedora content. We are also assuming you will continue to use the Islandora Solr module to view the search results.

Modify the Solr schema.xml file

Your schema.xml file will most likely be in /usr/local/fedora/solr/conf or /usr/local/fedora/solr/collection1/conf (assuming $SOLR_HOME = /usr/local/fedora/solr). We are assuming you installed Solr using the instructions provided in this guide. The collection1/conf is for more recent versions of Solr.

Add the text below to the **fields** section of the **schema.xml**

Solr Fields

```
<!-- ******** Dynamic fields required for Drupal content to be indexed ******** -->
<!-- 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="is_*" type="integer" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="im_*" type="integer" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="sis_*" type="sint" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="sim_*" type="sint" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="sm_*" type="string" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="tm_*" type="text_en" indexed="true" stored="true" multiValued="true" termVectors="true"/>
<dynamicField name="ss_*" type="string" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="ts_*" type="text_en" indexed="true" stored="true" multiValued="false" termVectors="true"/>
<dynamicField name="tsen2k_*" type="edge_n2_kw_text" indexed="true" stored="true" multiValued="false"
omitNorms="true" omitTermFreqAndPositions="true" />
<dynamicField name="ds_*" type="date" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="dm_*" type="date" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="tds_*" type="tdate" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="tdm_*" type="tdate" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="bm_*" type="boolean" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="bs_*" type="boolean" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="fs_*" type="sfloat" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="fm_*" type="sfloat" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="ps_*" type="sdouble" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="pm_*" type="sdouble" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="tis_*" type="tint" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="tim_*" type="tint" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="tls_*" type="tlong" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="tlm_*" type="tlong" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="tfs_*" type="tfloat" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="tfm_*" type="tfloat" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="tps_*" type="tdouble" indexed="true" stored="true" multiValued="false"/>
<dynamicField name="tpm_*" type="tdouble" indexed="true" stored="true" multiValued="true"/>
<!-- Sortable version of the dynamic string field -->
<dynamicField name="sort_ss_*" type="sortString" indexed="true" stored="false"/>
<copyField source="ss_" dest="sort_ss_"/>
<!-- A random sort field -->
<dynamicField name="random_*" type="rand" indexed="true" stored="true"/>
<!-- This field is used to store node access records, as opposed to CCK field data -->
<dynamicField name="nodeaccess*" type="integer" indexed="true" stored="false" multiValued="true"/>
```

You will also have to add a fieldtype for any types referenced in the list above that are not already defined in your schema. Depending on your version of Solr some of the fieldtypes below may or may not already be in your schema.xml file. Add any of the types below that do not already exist in your schema.xml.

Solr Fieldtypes

```
<!-- *******  Field types added in to allow Drupal content to be indexed here as well  ******** -->

    <!--
      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.
    -->
    <!-- ADD THESE IF THEY DON"T ALREADY EXIST -->
    <fieldType name="integer" class="solr.IntField" omitNorms="true"/>
```

```
    <fieldType name="tint" class="solr.TrieIntField" precisionStep="8" omitNorms="true" positionIncrementGap="0"
/>
    <fieldType name="tfloat" class="solr.TrieFloatField" precisionStep="8" omitNorms="true"
positionIncrementGap="0"/>
    <fieldType name="tlong" class="solr.TrieLongField" precisionStep="8" omitNorms="true" positionIncrementGap="
0"/>
    <fieldType name="tdouble" class="solr.TrieDoubleField" precisionStep="8" omitNorms="true"
positionIncrementGap="0"/>

    <!-- A Trie based date field for faster date range queries and date faceting. -->
    <fieldType name="tdate" class="solr.TrieDateField" omitNorms="true" precisionStep="6" positionIncrementGap="
0"/>

    <!-- Edge N gram type - for example for matching against queries with results
      KeywordTokenizer leaves input string intact as a single term.
      see: http://www.lucidimagination.com/blog/2009/09/08/auto-suggest-from-popular-queries-using-edgengrams/
    -->
    <fieldType name="edge_n2_kw_text" class="solr.TextField" positionIncrementGap="100">
      <analyzer type="index">
        <tokenizer class="solr.KeywordTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
        <filter class="solr.EdgeNGramFilterFactory" minGramSize="2" maxGramSize="25" />
      </analyzer>
      <analyzer type="query">
        <tokenizer class="solr.KeywordTokenizerFactory"/>
        <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
    </fieldType>
    <!--  Setup simple analysis for spell checking -->

    <fieldType name="textSpell" class="solr.TextField" positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.StandardTokenizerFactory" />
        <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>
        <filter class="solr.LengthFilterFactory" min="4" max="20" />
        <filter class="solr.LowerCaseFilterFactory" />
        <filter class="solr.RemoveDuplicatesTokenFilterFactory" />
      </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="sortString" 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 refrences to portions of the orriginal
          string matched by the pattern.

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

          http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/package-summary.html

          <filter class="solr.PatternReplaceFilterFactory"
          pattern="(^\p{Punct}+)" replacement="" replace="all"
          />
        -->
```

```
      </analyzer>
    </fieldType>

    <!-- A random sort type -->
    <fieldType name="rand" class="solr.RandomSortField" indexed="true" />
```

You will also have to copy the id field to the PID field. Add the below copyField statement in the same section as the other copyField statements.

Solr Copyfield

```
<copyField source="id" dest="PID"/>
```

In older versions of Solr you can copy the id to the PID and have the PID as the uniqueKey. In newer versions of Solr you may not be able to copy the id to PID (newer versions don't seem to let you use the uniqueKey as a destination for a copyField). In that case it may be easier to modify the GSearch xslt to create an id field instead of PID and use it as the uniqueKey in your schema.

Save the schema.xml file and restart Solr. After the restart you should be able to view solr at http://ip:port/solr.

Apache Solr Module

Once you have Solr configured properly and restarted you can install and configure the Apache Solr module in your Drupal site. See https://drupal.org/project/apachesolr for more information. Basically you just download it and enable as you would any other module. You've already modified the schema.xml so you don't need to do anything with schemal.xml.

At this point if everything is configured properly you should see some Drupal content in your solr index, if you don't see any Drupal content you may have to wait for cron to run or you can force the module to index content from the modules config interface (admin/config/search/apachesolr).

Modify Islandora search results

Islandora will not know how to properly display a result created from a Drupal node so you will either have to add a custom Islandora Solr display profile or use the theme layer to preprocess the results, **there are examples of creating a custom display profile in the islandora_solr_module.**

To modify results in your theme you could create a function called themename_preprocess_islandora_solr(&$variables). In this function you could check for a field that will only exist with drupal content, for instance bundle_name, and if it is not empty you could modify the thumbnail and the url used to link back to the content.

For example if you are using the bartik theme you could do something like:

theme_preprocess_islandora_solr

```
function bartik_preprocess_islandora_solr(&$variables) {
        $results = &$variables['results'];
                foreach ($results as &$result) {
                        if (!empty($result['solr_doc']['bundle_name'])) {
                        $path = url('http://aurltothethumbnail'); //could check bundle for type and link to
thumbnail for each type
                        $options = array('html' => TRUE);
                        $options['attributes']['title'] = $result['solr_doc']['label'];
                        $image = theme('image', array('path' => $path));
                        $key['thumbnail'] = l($image, $result['solr_doc']['url']['value'], $options);
                }
        }
}
```

Modifying the results by using a theme_preprocess hook is probably the easist way to modify your results but is probably also the most brittle. For instance if you change your theme your search results page may break.

Multi-site

If you are using a Drupal multi-site setup and you have more than one Drupal site's content in Solr you will probably want to filter on the site field or the hash field to limit results to just the drupal site you are viewing. This can be done in the Islandora Solr Settings -> Query defaults config page or you can setup a custom request handler in the solrconfig.xml.

TODO: Document how to filter drupal content by site.

Related articles

- Index Drupal Content in Islandora's Solr index