LuceneIndexTransformer

LuceneIndexTransformer is a component that creates or updates Lucene indexes.

This component only writes the index: to search the index, use the SearchGenerator component.

Why use it?

Instead of using LuceneIndexTransformer, you could generate an index by crawling your website. However, the LuceneIndexTransformer is *much, much* faster than crawling.

The big differences for the developer are:

- Using the LuceneIndexTransformer requires you to write a pipeline that can generate a `lucene:index` document describing your searchable URI space, so it's necessary to have a well-defined URI space. For a site with a consistent structure this should not be too hard. This pipeline can use aggregation and inclusion mechanisms to produce a full list of the pages you want to search. In this way it's also possible to generate an index for websites with forms which are not crawlable.
- On the other hand the crawler is a more generic solution, though far less efficient. It doesn't require a pipeline to "document" the entire searchable URI space. Instead, you must create a `content` view and a `links` view for each of the searchable pipelines. The URI space is then defined by crawling the `links` view.

Declaring the LuceneIndexTransformer

The transformer must be declared in the `<transformers>` section of your sitemap:

```
<map:sitemap xmlns:map="http://apache.org/cocoon/sitemap/1.0">

   <map:components>
      ...
      <map:transformers default="xslt">
         <map:transformer name="index"
            logger="sitemap.transformer.luceneindextransformer"
            src="org.apache.cocoon.transformation.LuceneIndexTransformer"/>
      </map:transformers>
      ...
   </map:components>
   ...
</map:sitemap>
```

Input document for the LuceneIndexTransformer

This is a sample of the kind of document that the transformer expects. NB In this example, I've chosen a couple of simple XHTML documents as the content to be indexed. This is only because everyone knows XHTML - in practice you should typically generate the index from an early stage in the pipeline; indexing DocBook, TEI, etc, rather than a presentation format like HTML.

```
<lucene:index xmlns:lucene="http://apache.org/cocoon/lucene/1.0"
   analyzer="org.apache.lucene.analysis.standard.StandardAnalyzer"
   directory="index"
   create="false"
   merge-factor="20">

   <lucene:document url="http://localhost/sample.html">
      <!-- here is some sample content -->
      <html>
         <head>
            <title lucene:store="true">Sample</title>
         </head>
         <body>
            <h1>Blah</h1>
            <a href="blah.jpg" title="download blah image"
               lucene:text-attr="title">
               <img src="blah-small.jpg" alt="Blah"
                  lucene:text-attr="alt"/>
            </a>
         </body>
      </html>
   </lucene:document>

   <lucene:document url="http://localhost/sample-2.html">
      <!-- Another sample doc -->
      <html>
         <head>
            <title lucene:store="true">Second Sample</title>
         </head>
         <body>
            <h1>Foo</h1>
            <p>Lorem ipsum dolor sit amet,
            consectetuer adipiscing elit. </p>
         </body>
      </html>
   </lucene:document>

</lucene:index>
```

What the lucene:index document means

The lucene:index element

The root element is `lucene:index`. The attributes of the `lucene:index` in the sample above are shown with their default values - so the effect is as if they were not specified at all.

The merge-factor and analyzer attributes

See the Lucene documentation for explanations of what they mean.

The optimize-frequency attribute (since version 2.2)

Determines how often the lucene index will be optimized. When you have 1000's of documents, optimizing the index can become quite slow (eg. 7 seconds for 9000 small docs, P4).

- 1: always optimize (default)
- 0: never optimize
- x: update every x times. You can use any number, it is a random generator which will determine to optimize or not.

You can eg. create a pipe without optimizing, which is used to index you're document everytime when it's modified. You can then create another pipe which will optimize, which is called manually. For more info see the Lucene FAQ , What is index optimization and when should I use it? :

http://wiki.apache.org/lucene-java/LuceneFAQ#head-fd848c31f4dc7b91727be6f40a7f5fbe2c66cfb8

The directory attribute

This attribute controls where the index files are stored. The path is relative to the Cocoon `work` directory.

The create attribute

This attribute controls whether the index is recreated.

- If create = "false" and the index already exists then the index will be updated. Documents which are already indexed will be removed from the index and reinserted.
- If the index does not exist then it will be created even if `create = "false"`.
- If `create = "true"` then any existing index will be destroyed and a new index created. If you are rebuilding your entire index then you should use `create = "true"` because the indexer doesn't need to remove old documents from the index, so it will be faster.

The lucene:document element

Lucene will index the content of each `lucene:document`, which may contain any xml content. The index is associated with the url specified by the `url` attribute. So this url will be returned as the results of a search.

The lucene:text-attr attribute

Normally Lucene will only index the content of these elements, not attribute values. To index the attributes of an element as well, give it an attribute called `lucene:text-attr`, containing a list of the names of the attributes you want indexed. For example, to index the value of the `alt` attribute of an `img` element, in `html`:

```
<img src="blah-small.jpg" alt="Blah" lucene:text-attr="alt"/>
```

This would index the text "Blah".

The lucene:store attribute

Normally Lucene will only index the text of an element, not store it. To store the text of an element in Lucene's index, add a `lucene:store="true"` attribute to the element. It's a good idea to store the title of a document in Lucene, so that your search results can show a document title as well as a URL.

The transformation

The transformer copies the source document to the output, except for the content of the `lucene:document` elements.

The transformer also adds an `elapsed-time` attribute to the output `lucene:document` elements, showing the time (in milliseconds) taken to index that document. You can use XSLT to transform the results into a report on the indexing operation.

Sample output

```
<?xml version="1.0" encoding="UTF-8"?>
<lucene:index xmlns:lucene="http://apache.org/cocoon/lucene/1.0"
        merge-factor="20"
        create="false"
        directory="index"
        analyzer="org.apache.lucene.analysis.standard.StandardAnalyzer">
        <lucene:document url="JCB-001/full.html" elapsed-time="3846"/>
        <lucene:document url="JCB-001/_div1-N1017B.html" elapsed-time="3735"/>
        <lucene:document url="JCB-002/full.html" elapsed-time="361"/>
        <lucene:document url="JCB-002/_div1-N10190.html" elapsed-time="1302"/>
        <lucene:document url="JCB-003/full.html" elapsed-time="300"/>
        <lucene:document url="JCB-003/_div1-N10188.html" elapsed-time="1352"/>
</lucene:index>
```

Note to users of Mac OS X

Java can not open more than 256 files at a time by default, so you may get an error like the following:

```
Description: org.apache.cocoon.ProcessingException:
Failed to execute pipeline.: java.lang.RuntimeException:
java.io.FileNotFoundException:
/usr/local/tomcat-4/work/Standalone/localhost/_/cocoon-files/index/_15.f86
(Too many open files)
```

To avoid this error, you should set your ulimit in the shell script that starts Tomcat. My line reads as follows:

```
ulimit -S -n 1000
```

Read more about this here: http://www.amug.org/~glguerin/howto/More-open-files.html

Note to users of Redhat Linux

If you get the following error: (Empty StackException) while creating the index with the LuceneIndexTransformer try to alter your merge-factor to a lower value (default should be 10). Look at the Lucene documentation for more information.