HowToMakeCustomSearch

[This is for Nutch 1.0]

How do you index your custom data and then search for the same using the Nutch web interface?

Use Case

Suppose we want to search for the author of the website by his email id.

Indexing the email id

Before we can search for our custom data, we need to index it. Nutch has a plugin architecture very similar to that of Eclipse. We can write our own plugin for indexing. Here is the source code:

```
package com.swayam.nutch.plugins.indexfilter;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.Text;
import org.apache.nutch.crawl.CrawlDatum;
import org.apache.nutch.crawl.Inlinks;
import org.apache.nutch.indexer.IndexingException;
import org.apache.nutch.indexer.IndexingFilter;
import org.apache.nutch.indexer.NutchDocument;
import org.apache.nutch.indexer.lucene.LuceneWriter;
import org.apache.nutch.parse.Parse;

/**
 *@author paawak
 */
public class EmailIndexingFilter implements IndexingFilter {

    private static final Log LOG = LogFactory.getLog(EmailIndexingFilter.class);

    private static final String KEY_CREATOR_EMAIL = "email";

    private Configuration conf;


    public NutchDocument filter(NutchDocument doc, Parse parse, Text url,
            CrawlDatum datum, Inlinks inlinks) throws IndexingException {

        // look up email of the author based on the url of the site
        String creatorEmail = EmailLookup.getCreatorEmail(url.toString());

        LOG.info("######## creatorEmail = " + creatorEmail);

        if (creatorEmail != null) {
            doc.add(KEY_CREATOR_EMAIL, creatorEmail);
        }

        return doc;
    }

    public void addIndexBackendOptions(Configuration conf) {

        LuceneWriter.addFieldOptions(KEY_CREATOR_EMAIL, LuceneWriter.STORE.YES,
                LuceneWriter.INDEX.TOKENIZED, conf);

    }

    public Configuration getConf() {
        return conf;
    }

    public void setConf(Configuration conf) {
        this.conf = conf;
    }

}
```

Also, you need to create a *plugin.xml*:

```
<plugin id="index-email" name="Email Indexing Filter" version="1.0.0"
  provider-name="swayam">

  <runtime>
    <library name="EmailIndexingFilterPlugin.jar">
      <export name="*" />
    </library>
  </runtime>

  <requires>
    <import plugin="nutch-extensionpoints" />
  </requires>

  <extension id="com.swayam.nutch.plugins.indexfilter.EmailIndexingFilter"
    name="Email Indexing Filter"
    point="org.apache.nutch.indexer.IndexingFilter">
    <implementation id="index-email"
      class="com.swayam.nutch.plugins.indexfilter.EmailIndexingFilter" />
  </extension>

</plugin>
```

This done, create a new folder in the *$NUTCH_HOME/plugins* and put the jar and the plugin.xml there.

Now we have to activate this plugin. To do this, we have to edit the *conf/nutch-site.xml*.

```
<property>
  <name>plugin.includes</name>
  <value>nutch-extensionpoints|protocol-http|parse-(text|html)|index-(basic|email)|query-(basic|site|url)<
/value>
  <description>Regular expression naming plugin id names to
  include.  Any plugin not matching this expression is excluded.
  In any case you need at least include the nutch-extensionpoints plugin. By
  default Nutch includes crawling just HTML and plain text via HTTP,
  and basic indexing and search plugins.
  </description>
</property>
```

Now, how do I search my indexed data?

Option 1 [cumbersome]:

Add my own query plugin:

```
package com.swayam.nutch.plugins.queryfilter;

import org.apache.nutch.searcher.FieldQueryFilter;

/**
 *@author paawak
 */
public class MyEmailQueryFilter extends FieldQueryFilter {

    public MyEmailQueryFilter() {
        super("email");
    }

}
```

Do not forget to edit the plugin.xml.

```
<plugin
   id="query-email"
   name="Email Query Filter"
   version="1.0.0"
   provider-name="swayam">

   <runtime>
      <library name="EmailQueryFilterPlugin.jar">
         <export name="*"/>
      </library>
   </runtime>

   <requires>
      <import plugin="nutch-extensionpoints"/>
   </requires>

   <extension id="com.swayam.nutch.plugins.queryfilter.MyEmailQueryFilter"
              name="Email Query Filter"
              point="org.apache.nutch.searcher.QueryFilter">
      <implementation id="query-email"
                      class="com.swayam.nutch.plugins.queryfilter.MyEmailQueryFilter">
        <parameter name="fields" value="email"/>
      </implementation>

   </extension>

</plugin>
```

This line is particularly important:

<parameter name="fields" value="email"/>

If you skip this line, you will never be able to see this in search results.

The only catch here is you have to prepend the keyword ***email:*** to the search key. For example, if you want to search for ***jsmith@mydomain.com***, you have to search for ***email:jsmith@mydomain.com*** or ***email:jsmith***.

There is an easier and more elegant way.

Option 2 [smart]

Use the existing query-basic plugin.

This involves editing just one file: conf/nutch-default.xml.

In the default distribution, you can see some commented lines like this:

```
<!--
<property>
  <name>query.basic.description.boost</name>
  <value>1.0</value>
  <description> Declares a custom field and its boost to be added to the default fields of the Lucene query.
  </description>
</property>
-->
```

All you have to do is un-comment them and put your custom field, email, in our case in place of description. The resulting fragment will look like:

```
<property>
  <name>query.basic.email.boost</name>
  <value>1.0</value>
  <description> Queries the author of the site by his email-id
  </description>
</property>
```

With this while looking for ***jsmith@mydomain.com***, you can simply enter ***jsmith@mydomain.com*** or a part the name like ***jsmit***.

Building a Nutch plugin

The preferred/official way is by ant, but I have used maven with the following dependencies:

```
<project>
...
        <dependencies>
                ...
                <!-- nutch -->
                <dependency>
                        <groupId>org.apache.lucene</groupId>
                      <artifactId>lucene-core</artifactId>
                      <version>2.4.0</version>
                      <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.apache.lucene</groupId>
                      <artifactId>lucene-misc</artifactId>
                      <version>2.4.0</version>
                      <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.apache.nutch</groupId>
                      <artifactId>nutch</artifactId>
                      <version>1.0</version>
                      <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.apache.taglibs</groupId>
                      <artifactId>taglibs-i18n</artifactId>
                      <version>1.0.N20030822</version>
                      <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.apache.tika</groupId>
                        <artifactId>tika</artifactId>
                        <version>0.1-incubating</version>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>xerces</groupId>
                        <artifactId>xerces</artifactId>
                        <version>2.6.2</version>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>xerces</groupId>
                        <artifactId>xerces-apis</artifactId>
                        <version>2.6.2</version>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.jets3t.service</groupId>
                        <artifactId>jets3t</artifactId>
                        <version>0.6.1</version>
```

```
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>oro</groupId>
                          <artifactId>oro</artifactId>
                          <version>2.0.8</version>
                          <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>com.ibm.icu</groupId>
                          <artifactId>icu4j</artifactId>
                          <version>4.0.1</version>
                          <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.apache.hadoop</groupId>
                          <artifactId>hadoop-core</artifactId>
                          <version>0.19.1</version>
                          <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.apache.solr</groupId>
                          <artifactId>solr-common</artifactId>
                          <version>1.3.0</version>
                          <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.apache.solr</groupId>
                          <artifactId>solrj</artifactId>
                          <version>1.3.0</version>
                          <scope>provided</scope>
                </dependency>
                <!-- end nutch -->
            ...
        </dependencies>
...
</project>
```