

Lius 1.0 Tutorial by examples

Par : Rida Benjelloun

Rida.benjelloun@doculibre.com

Table des matières

1- Factory indexing.....	2
2- Mixed indexing.....	5
3- XML Node Indexing.....	7
4- JavaBeans indexing.....	8
5- Recherche.....	9

1- Factory indexing

Le factory indexing permet d'indexer des fichiers XML, HTML, Word, Excel, PowerPoint, RTF, PDF, ZIP, Txt, OpenOffice et MP3. Il se base sur le fichier de configuration de Lius et utilise le Mime type du fichier pour appliquer le bon indexeur sur le document.

Exemple 1 : Indexation avec le factory:

```
String indexDir = "c:\\index" ;
String log4j = "c:\\lius\\config\\log4j.properties";
String toIndex = "c:\\files\\test.pdf";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

Indexer indexer = IndexerFactory.getIndexer(toIndex, lc);

indexer.index(indexDir);
```

Note : la version 1.0-RC2 ajoute automatiquement le chemin du fichier dans le champ « filePath »

Exemple 2 : Ajout de champs personnalisés

```
String indexDir = "c:\\index" ;
String log4j = "c:\\lius\\config\\log4j.properties";
String toIndex = "c:\\files\\test.pdf";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

Indexer indexer = IndexerFactory.getIndexer(toIndex, lc);

//Add custom fields
List fields = new ArrayList();
fields.add(new Field( "fieldTest", "Test new field", Field.Store.YES,
Field.Index.TOKENIZED));

indexer.index(indexDir, fields);
```

Exemple 3 :

Fichier de configuration de lius permettant d'indexer un fichier PDF. Pour plus d'exemples, il est recommandé de voir le fichier liusconfig.xml qui se trouve dans dossier « config » de la distribution Lius .

```
<?xml version="1.0" encoding="UTF-8" ?>
<.luceneIndex>
<properties>
  <analyzer class="org.apache.lucene.analysis.lius.unicode.UTF8AccentRemoverAnalyzer" />
  <createIndex value="auto" />
  <indexWriterProperty mergeFactor="10" maxMergeDocs="100" optimize="true" />
</properties>
<index>
  <pdf setBoost="1.6">
    <indexer class="lius.index.PDF.PdfIndexer">
      <mime>application/pdf</mime>
    </indexer>
    <fields>
      <.luceneField name="fullText" get="content" type="Text" />
      <.luceneField name="title" get="title" type="Text" />
      <.luceneField name="author" get="author" type="Text" />
      <.luceneField name="creator" get="creator" type="Text" />
      <.luceneField name="summary" get="summary" type="Text" />
      <.luceneField name="keywords" get="keywords" type="Text" />
      <.luceneField name="subject" get="subject" type="Text" />
    </fields>
  </pdf>
</index>
<search>
  <multiFieldQueryParser>
    <searchFields sep=",">title,author,creator,summary,fullText</searchFields>
  </multiFieldQueryParser>
</search>
<searchResult>
  <fieldsToDisplay setHighlighter="true">
    <.luceneField name="title" label="title" />
    <.luceneField name="subject" label="subject" />
    <.luceneField name="creator" label="creator" />
    <.luceneField name="filePath" label="File path" />
    <.luceneField name="fullText" label="full text" setFragmenter="50" />
  </fieldsToDisplay>
</searchResult>
</.luceneIndex>
```

Exemple 4: Indexation de sous dossiers

```
String indexDir = "c:\\index" ;
```

```
String log4j = "c:\\lius\\config\\log4j.properties";
String toIndex = "c:\\files\\directory";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

LuceneActions.getSingletonInstance().indexSubDirectories(toIndex, indexDir, lc);
```

Exemple 5 : Extraction de texte à partir d'un document

```
String indexDir = "c:\\index" ;
String log4j = "c:\\lius\\config\\log4j.properties";
String toIndex = "c:\\files\\test.pdf";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

Indexer indexer = IndexerFactory.getIndexer(toIndex, lc);

String text = Indexer.getContent();
```

Exemple 6 : Récupération d'un objet de type lucene Document à partir de Lius

```
String indexDir = "c:\\index" ;
String log4j = "c:\\lius\\config\\log4j.properties";
String toIndex = "c:\\files\\test.pdf";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

Indexer indexer = IndexerFactory.getIndexer(toIndex, lc);
Collection liusFields = indexer.getPopulatedLiusFields();

Document luceneDoc =
    LuceneActions.getSingletonInstance().populateLuceneDoc(liusFields);
```

2- Mixed indexing

L'indexation mixte est une indexation par répertoire et consiste à stocker dans le même document Lucene (occurrence), tous les documents qui se trouvent dans le répertoire. Ceci est très utile lorsque le document ou unité documentaire est constitué de plusieurs fichiers (exemple : texte intégral en PDF et les métadonnées dans un fichier XML).

Exemple 7 : Exemple de l'indexation mixte

```
String indexDir = "c:\\index" ;
String log4j = "c:\\lius\\config\\log4j.properties";
String toIndex = "c:\\files\\testDir";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

Indexer indexer = new MixedIndexer();
indexer.setUp(lc);
indexer.setMixedContentsObj(toIndex);
indexer.index(indexDir);
```

Exemple 8 : Exemple d'un fichier de configuration mixte supportant le Pdf et le XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<uceneIndex>
<properties>
  <analyzer class="org.apache.lucene.analysis.lius.unicode.UTF8AccentRemoverAnalyzer" />
  <createIndex value="auto" />
  <indexWriterProperty mergeFactor="10" maxMergeDocs="100" optimize="true" />
</properties>
<index>
<!-- Mixed indexing tag -->
<mixedIndexing>
  <fields>
    <uceneField fileMimeType="text/xml"/>
    <uceneField fileMimeType="application/pdf"/>
  </fields>
</mixedIndexing>

<!-- Xml indexing -->
<xml>
  <xmlFile ns="http://purl.org/dc/elements/1.1/" setBoost="2.0">
    <indexer class="lius.index.xml.XmlFileIndexer">
      <mime>text/xml</mime>
    </indexer>
```

```
<fields>
  <.luceneField name="title" xpathSelect="//dc:title" type="Text" >
  <.luceneField name="subject" xpathSelect="//dc:subject" type="Keyword"/>
  <.luceneField name="creator" xpathSelect="//dc:creator" type="Text"/>
</fields>
</xmlFile>
</xml>
<!-- Pdf indexing -->
<pdf setBoost="1.6">
  <indexer class="lius.index.PDF.PdfIndexer">
    <mime>application/pdf</mime>
  </indexer>
  <fields>
    <.luceneField name="fullText" get="content" type="Text" />
  </fields>
</pdf>
</index>
<search>
  <multiFieldQueryParser>
    <searchFields sep=",">title ,creator ,fullText</searchFields>
  </multiFieldQueryParser>
</search>
<searchResult>
  <fieldsToDisplay setHighlighter="true">
    <.luceneField name="title" label="title" />
    <.luceneField name="creator" label="creator" />
    <.luceneField name="fullText" label="full text" setFragmenter="50" />
    <.luceneField name="filePath" label="File path" />
  </fieldsToDisplay>
</searchResult>
</luceneIndex>
```

3- XML Node Indexing

Une indexation par noeud XML permet de découper le document Xml en plusieurs nœuds et de les stocker dans des occurrences distinctes au niveau de l'index.

Exemple 9 : exemple d'indexation par noeud

```
String indexDir = "c:\\index" ;
String log4j = "c:\\lius\\config\\log4j.properties";
String toIndex = "c:\\files\\testNodes.xml";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

Indexer indexer = new XmlNodeIndexer();
indexer.setUp(lc);
indexer.setStreamToIndex(new FileInputStream(toIndex));
indexer.index(indexDir);
```

Exemple 10 : exemple de fichier de configuration par nœud

```
<?xml version="1.0" encoding="UTF-8" ?>
<uceneIndex>
<properties>
  <analyzer class="org.apache.lucene.analysis.lius.unicode.UTF8AccentRemoverAnalyzer" />
  <createIndex value="auto" />
  <indexWriterProperty mergeFactor="10" maxMergeDocs="100" optimize="true" />
</properties>
<index>
  <xml>
    <xmlNodes>
      <node select="//book">
        <uceneField name="title" xpathSelect="title" type="Text" />
        <uceneField name="creator" xpathSelect="creator" type="Text" />
      </node>
    </xmlNodes>
  </xml>
</index>
<search>
  <multiFieldQueryParser>
    <searchFields sep=",">title, creator </searchFields>
  </multiFieldQueryParser>
</search>
<searchResult>
  <fieldsToDisplay setHighlighter="true">
    <uceneField name="title" label="title" />
    <uceneField name="creator" label="creator" />
  </fieldsToDisplay>
</searchResult>
```

```
</fieldsToDisplay>
</searchResult>
</luceneIndex>
```

4- JavaBeans indexing

Le java bean indexer permet d'indexer des objets Java

Exemple 11 : exemple d'indexation de Java Beans

```
String indexDir = "c:\\index" ;
String log4j = "c:\\lius\\config\\log4j.properties";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

// Bean personne
Personne personne = new Personne();
personne.setNom("Benjelloun");
personne.setPrenom("Rida");
personne.setAdresse("Quebec Canada");

Indexer indexer = new BeanIndexer();
indexer.setUp(lc);
indexer.setObjectToIndex(personne);
indexer.index(indexDir);
```

Exemple 12 : exemple de fichier de configuration pour l'indexation de Java Beans

```
<?xml version="1.0" encoding="UTF-8" ?>
<luceneIndex>
<properties>
  <analyzer class="org.apache.lucene.analysis.lius.unicode.UTF8AccentRemoverAnalyzer" />
  <createIndex value="auto" />
  <indexWriterProperty mergeFactor="10" maxMergeDocs="100" optimize="true" />
</properties>
<index>
<JavaBeans>
  <Bean class="lius.test.beans.Personne">
    <luceneField name="nom" getMethod="getNom" type="Text"/>
    <luceneField name="prenom" getMethod="getPrenom" type="Text"/>
    <luceneField name="adresse" getMethod="getAdresse" type="Text"/>
  </Bean>
</JavaBeans>
</index>
</luceneIndex>
```

```

</JavaBeans>
</index>
<search>
  <multiFieldQueryParser>
    <searchFields sep=",">nom, prenom, adresse </searchFields>
  </multiFieldQueryParser>
</search>
<searchResult>
  <fieldsToDisplay setHighlighter="true">
    <luceneField name="nom" label="Nom" />
    <luceneField name="prenom" label="Prenom" />
    <luceneField name="adresse" label="Adresse" />
  </fieldsToDisplay>
</searchResult>
</luceneIndex>

```

5- Recherche

Exemple 13 : exemple de recherche sur un seul index

```

String indexDir = "c:\\index" ;
String log4j = "c:\\lius\\config\\log4j.properties";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

LiusHitList ls = SearchIndex.search("canada", indexDir, liusConfig);

System.out.println("Nb doc = " + ls.size());
for (int i = 0; i < ls.size(); i++) {
  LiusHit lh = (LiusHit) ls.get(i);
  System.out.println("=====*****=====");
  System.out.println("Score = " + lh.getScore());
  System.out.println("Doc id = " + lh.getDocId());
  List liusHitFields = lh.getLiusFields();
  for (int j = 0; j < liusHitFields.size(); j++) {
    LiusField lf = (LiusField) liusHitFields.get(j);
    String name = lf.getLabel();
    String[] values = lf.getValues();
    System.out.print(name + " : ");
    for (int k = 0; k < values.length; k++) {
      System.out.println("\t" + values[k]);
    }
  }
  System.out.println("=====");
}
}

```

Exemple 14 : exemple de recherche sur plusieurs index

```
String indexDir = "c:\\index" ;
String indexDir2 = "c:\\index2" ;
String log4j = "c:\\lius\\config\\log4j.properties";

LiusLogger.setLoggerConfigFile(log4j);
LiusConfig lc = LiusConfigBuilder.getSingletonInstance().getLiusConfig(liusConfig);

String[] indexes = { indexDir, indexDir2 };
LiusHitList res = SearchIndex.multiIndexSearch("maroc", indexes, liusConfig);

System.out.println("Nb doc = " + ls.size());
for (int i = 0; i < ls.size(); i++) {
    LiusHit lh = (LiusHit) ls.get(i);
    System.out.println("=====*****=====");
    System.out.println("Score = " + lh.getScore());
    System.out.println("Doc id = " + lh.getDocId());
    List liusHitFields = lh.getLiusFields();
    for (int j = 0; j < liusHitFields.size(); j++) {
        LiusField lf = (LiusField) liusHitFields.get(j);
        String name = lf.getLabel();
        String[] values = lf.getValues();
        System.out.print(name + " : ");
        for (int k = 0; k < values.length; k++) {
            System.out.println("\t" + values[k]);
        }
    }
    System.out.println("=====");
}
```