Version 1.5

ORI-OAI-indexing : Indexation des ressources

> ⚠ **Module obligatoire quelque soit la configuration choisie**

Voir l'architecture du système

Description

Une fois le dépôt de ressources et la saisie de métadonnées validés, ces dernières sont indexées par le module ORI-OAI-indexing. Ce module a pour rôle l'indexation des fiches de métadonnées ainsi que des documents associés.

Pour cela, il utilise le moteur d'indexation Lucene. Celui-ci permet l'indexation de différentes sources offrant une recherche puissante et rapide en se reposant sur différents analyseurs. L'analyseur de la langue française permettra notamment la gestion des verbes conjugués, des pluriels ou encore des accents et caractères spéciaux. Un système de pondération permet aussi de rendre une métadonnée plus pertinente qu'une autre. Par exemple, on préfèrera retrouver en premier les documents dont l'élément recherché se trouve dans le titre plutôt que dans la description.

Lius est un framework d'indexation basé sur le projet Lucene. Il permet une indexation de différents formats de fichiers comme XML, PDF, OpenOffice, ZIP, MP3, etc. Il est utilisé dans notre projet pour offrir une configuration avancée des champs à indexer dans les différents formats de fiches de métadonnées XML et, par la suite, pour indexer les documents associés en plein texte.

En plus de l'aspect indexation, ORI-OAI-indexing offre un service de recherche de documents via Web service en se reposant sur la syntaxe des requêtes Lucene. Il est utilisé par différents composants dans le système.

Voir la documentation technique

Spécifications

- Indexing dans Ori-Oai
- Lucene et applications attachées
- Ehchache
- Crawler web
- Quartz

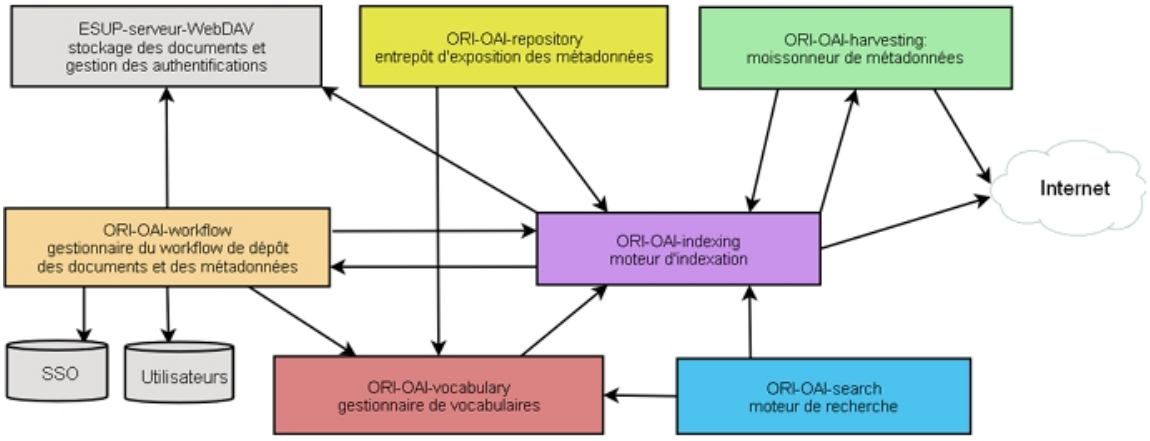
Indexing dans Ori-Oai

Indexing dans ORI-OAI

ORI-OAI-indexing permet de gérer l'indexation et la recherche de documents locaux et distants. Il est appelé par le workflow via le frontal d'indexation pour indexer des fiches locales. En les indexant il reçoit un identifiant ainsi que d'autres éléments pour chacune d'elles. Le moissonneur fait également appel à lui, toujours par l'intermédiaire du frontal, en ce qui concerne l'indexation. ORI-OAI-indexing indexera une chaîne de caractères contenant la fiche. Le moissonneur envoie d'ailleurs un identifiant qui sera indexé avec la fiche pour faire le lien entre les deux modules.

Les recherches dans l'index sont effectuées par le module ORI-OAI-Search. Ce composant envoie une requête de type Lucene au module d'indexation. Il récupère ensuite de la part de ce dernier les attributs qui correspondent à sa recherche tels que le titre ou l'auteur par

exemple.



Lucene et applications attachées

Lucene et applications attachées

Lucene est un moteur de recherche appartenant à la fondation apache permettant l'indexation et la recherche de texte. Il est entièrement écrit en langage Java. La version utilisée dans cette version du module d'indexation est la 2.3.2. Le lien vers le projet est le suivant : http://lucene.apache.org/java/docs/.

LIUS

LIUS, qui signifie Lucene Index Update and Search, est un framework d'indexation basé sur le projet Jakarta Lucene. Il a été développé à partir d'un ensemble de technologies JAVA et d'applications entièrement "open source". La documentation vers LIUS est donnée par le lien suivant : http://sourceforge.net/projects/lius/

LIUS ajoute à Lucene plusieurs fonctionnalités d'indexation de type de documents tel que : Ms Word, Ms Excel, Ms PowerPoint, RTF, PDF, XML, HTML, TXT, la suite Open Office et les JavaBeans. Cet outil permet également d'effectuer une indexation mixte, qui a pour but d'intégrer tout le contenu d'un répertoire sous la même occurrence. Ceci est très utile lorsque l'utilisateur veut indexer des métadonnées en XML et le texte intégral en PDF ou dans un autre format. Ceci permet par la suite d'effectuer par exemple des recherches sur le titre, auteur et le texte intégral en même temps.

Toute la configuration de l'indexation, telle que le type de fichiers à indexer ou encore les champs par exemple, ainsi que la recherche sont définies dans un fichier XML, il ne reste plus qu'à écrire le code pour exécuter l'indexation ou la recherche.

Luke

Luke (http://www.getopt.org/luke/) est une interface graphique permettant de visualiser un index. Il peut être utile en tant qu'outil de diagnostic de ce dernier.

Ehchache

Ehcache

Il s'agit d'un gestionnaire de cache en Java. Il est capable de stocker des données en mémoire vive ou sur le disque. La page d'accueil de cette application est la suivante : http://ehcache.sourceforge.net/. Ehcache est utlisé dans le cadre du projet ORI-OAI-Indexing en ce qui concerne la gestion des différents caches utiles à l'optimisation de la recherche dans l'index. La version de la librairie utilisée dans le module d'indexation est la 1.3.

Crawler web

Crawler web

Le crawler utilisé par Ori-Oai-Indexing pour visiter les pages web contenant des liens vers des documents plein texte est consultable grâce au lien suivant : https://crawler.dev.java.net/. Il s'agit d'un crawler développé par Java qui présente divers avantages : il est facile d'utilisation, son intégration est simple. Il est également possible de limiter le nombre de liens visités ou la profondeur du crawling. Enfin ce crawler gère les redirections.

Quartz

Quartz

Cette application créée par OpenSymphony est utilisée dans le cadre d'Ori-Oai-Indexing pour la tâche planifiée de crawling qui se déroule généralement la nuit. Quartz permet de créer des tâches planifiées très simples ou plus complexes. Cet outil est consultable à l'adresse suivante : http://www.opensymphony.com/quartz/

Changements de version

Modifications de la version 1.5

Le module d'indexation a subi de nombreuses modifications et ajouts.

> L'index créé à partir de la version 1.4 n'est pas compatible avec la version 1.5. Il faut donc le supprimer en utilisant le bouton de réinitialisation de l'index dans la partie Visualisation puis la fonction de réindexation complète dans les modules Ori-Oai-Workflow et Ori-Oai-Harvester.

Nouvelles fonctionnalités

Les performances de la recherche ont été améliorées grâce à un nouveau système de tri géré par Lucene.

Le module d'indexation utilise maintenant le champ N d'une vCard.

Un champ a été ajouté dans l'index pour obtenir un identifiant unique.

Il est maintenant possible d'optimiser manuellement l'index grâce au bouton "Optimiser l'index" situé dans la page "Gestion de l'index" de la partie Visualisation.

Une visualisation de la progression du crawling de l'index est maintenant possible.

Un nouvel onglet "Métadonnées et UTF-8" a été ajouté dans la partie Visualisation. Il permet d'encoder en UTF-8 une valeur pour faciliter le remplissage du fichier liusConfig.xml.

Un bug a été corrigé dans l'onglet "Recherche" de la partie Visualisation.

Modifications de la version 1.4

Le module d'indexation comprend de nouvelles fonctionnalités.

> L'index créé à partir de la version 1.1 n'est pas compatible avec la version 1.4. Il faut donc le supprimer en utilisant le bouton de réinitialisation de l'index dans la partie Visualisation puis la fonction de réindexation complète dans les modules Ori-Oai-Workflow et Ori-Oai-Harvester

Nouvelles fonctionnalités

Le format englobant a été inséré dans le module d'indexation. Le principe de cette fonctionnalité est d'insérer tous les formats d'un identifiant dans une seule entrée de l'index.

Ori-Oai-Indexing est maintenant capable d'indexer des documents en texte intégral. Cette fonctionnalité est utile dans le cadre du crawler web.

Un crawler Web est disponible dans cette nouvelle version. Il permet de rechercher le document plein texte à partir de la fiche indexée. Une tâche planifiée est ajoutée pour lancer automatiquement le crawling. Par ailleurs un pool de thread permet d'augmenter les performances du crawling.

Des champs ont été ajoutés pour la prise en charge du LOM-fr ainsi que du SupLOM-fr.

Le système de cache est maintenant assuré grâce à Ehcache. Certains caches sont maintenant persistants.

La partie visualisation a été améliorée et mise en conformité avec les design des autres modules.

Le système de highlighting a été amélioré.

Fichier de configuration

Une partie concernant les paramètres du crawler web a été ajoutée dans les options modifiables par l'utilisateur du fichier configIndexing.xml

Installation

Il existe plusieurs modes d'installation de ce module. Le mode recommandé est l'utilisation ori-oai-quick-install. Ceci vous permettra de déployer la suite ori-oai avec un minimum de personnalisation tout ceci en utilisant un seul fichier de configuration.

L'installation manuelle vous fera éditer manuellement différents fichiers afin de configurer au mieux votre application.

Il est préférable d'utiliser la première solution. En effet, celle-ci vous apportera un déploiement rapide de ORI-OAI sur un serveur de production avec une configuration de base. Vous pourrez toutefois après cette installation apporter toutes les configurations avancées que vous souhaitez à vos modules.

Reportez-vous à la documentation en ligne d'installation de ORI-OAI.

Installation manuelle

Pré-Requis

- ORI-OAI-Indexing est une application fonctionnant avec le langage Java. Le JDK (version 5 ou ultérieure) doit donc être installé sur la machine de déploiement.
- Les tâches de compilation, de déploiement et certaines actions utilisent ANT.
- Une version de Tomcat doit être disponible sur la machine de déploiement. Le module a été testé avec la version 6.0.20 de Tomcat.

> Il est nécessaire de spécifier au Tomcat que vous utilisez l'encodage UTF-8 pour tous les modules. Pour cela, éditez le fichier PATH_TOMCAT_INDEXING/bin/startup.sh (startup.bat sous Windows) ou PATH_TOMCAT_INDEXING/bin/catalina.sh (catalina.bat sous Windows) et y ajoutez la commande suivante:
> - export CATALINA_OPTS="-Dfile.encoding=UTF-8 $CATALINA_OPTS" (sous Unix)
> - set CATALINA_OPTS="-Dfile.encoding=UTF-8 %CATALINA_OPTS%" (sous Windows)

> **Important**
> Il est fortement conseillé de modifier les paramètres du Tomcat en ajoutant -Xms256m et -Xmx512m ce qui permet d'ajouter de la mémoire vive disponible au Tomcat dédié au module d'indexation. Ces paramètres sont d'autant plus nécessaires si les recherches renvoient beaucoup de résultats ou si vous utilisez le crawler web.

Configuration pour un déploiement manuel

init-build.properties

Ce fichier se trouve à la racine du repertoire ori-oai-indexing. Il contient les données nécessaires au déploiement correct de l'application dans Tomcat. Copiez ce fichier en build.properties. Ouvrez le nouveau fichier créé et modifiez-le comme indiqué ci-dessous.

> Ne supprimez pas le fichier init-build.properties. Il est utilisé dans le cadre d'une installation rapide avec ori-oai-commons-quick-install.

Le fichier se présente de la manière suivante :

```
############################
##                        ##
##   PARTIE INSTALLATION  ##
## 2 modes d'installation ##
##        1) ou 2)        ##
##                        ##
############################


#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
# 1) Utilisation de ori-oai-commons-quick-install
#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#

#URL du fichier contenant toutes les proprietes pour ce module en installation rapide
#Commentez le parametre si vous ne voulez pas utiliser les fonctionnalites d'installation de
ori-oai-commons-quick-install
commons.parameters.central.file.url=[COMMONS_PARAMETERS_CENTRAL_FILE_URL]


#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#
## 2) Installation manuelle du module
# Dans ce cas, il est necessaire de commenter
# le parametre commons.parameters.central.file.url
#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#-#

# Cette partie du fichier doit etre mise a jour avant
# la premiere utilisation de votre application dans votre environnement

#Repertoire d'installation de Tomcat
tomcat.home  = [PATH_TOMCAT_INDEXING]

#Repertoire de deploiement
deploy.home  = [PATH_TOMCAT_INDEXING]/webapps/

#Nom de ditribution de l'application
app.name.deploy=[CONTEXT_INDEXING]

#URL vers Tomcat
tomcat.URL.deploy = http://[HOST_INDEXING]:[PORT_INDEXING]

#Dossier dans lequel se trouve l'index
index.directory = [INDEXES_DATA_DIR]/index-indexing/index
```

Etant donné que vous faites une installation manuelle, il est nécessaire de commenter la variable commons.parameters.central.file.url dans ce fichier comme ceci:

```
#URL du fichier contenant toutes les propriétés pour ce module en installation rapide
#Commentez le parametre si vous ne voulez pas utiliser les fonctionnalites d'installation de
ori-oai-commons-quick-install
#commons.parameters.central.file.url=[COMMONS_PARAMETERS_CENTRAL_FILE_URL]
```

tomcat.home

> *Répertoire racine de Tomcat. Remplacez [PATH_TOMCAT_INDEXING] par le chemin où se trouve le serveur Tomcat où sera installé le module ori-oai-indexing*

deploy.home

> *Répertoire dans lequel sera déployée l'application (il doit s'agir du répertoire* webapps *de Tomcat). Remplacez [PATH_TOMCAT_INDEXING] par la même valeur que celle donnée dans tomcat.home*

app.name.deploy

> *Nom de l'application dans le contexte Tomcat. Remplacez [CONTEXT_INDEXING] par le nom que vous souhaitez donner au répertoire de déploiement du module (exemple : ori-oai-indexing ou indexing). Ce nom servira également à retrouver la page d'accueil du module qui sera :*
> ***http:// [HOST_INDEXING] :[PORT_INDEXING]/[CONTEXT_INDEXING]/***
> *(voir section 4 de la page utilisation).*

tomcat.URL.deploy

> *URL vers la page d'accueil de Tomcat. Ce champ est utile dans le cadre des Premiers Tests. Remplacez [HOST_INDEXING] et [PORT_INDEXING] par le nom de la machine et le numéro de port du Tomcat où est installé le*

> *module ori-oai-indexing.*

index.directory

> *Répertoire dans lequel se trouve l'index.*

log4j.properties

Le fichier log4j.properties se présente de la manière suivante :

```
# ----- PARTIE A MODIFIER PAR L'UTILISATEUR
log4j.appender.fichier.file=[PATH_TOMCAT_INDEXING]/logs/ori-oai-indexing.log

# ----- Definition des logs fichier
log4j.logger.org.orioai.indexing=INFO,fichier
#log4j.appender.fichier=org.apache.log4j.ConsoleAppender
log4j.appender.fichier=org.apache.log4j.FileAppender

log4j.appender.fichier.layout=org.apache.log4j.PatternLayout
log4j.appender.fichier.layout.ConversionPattern=%5p %d{MMM/dd HH:mm:ss} %c :: %m%n
```

Les paramètres à modifier éventuellement sont les suivants :

log4j.logger.org.orioai.indexing

> *Niveau de logs Trois niveaux sont utilisés dans ORI-OAI-Indexing.*
>
> - *ERROR : C'est le niveau à utiliser en production. Seules les erreurs seront notifiées dans le fichier pour optimiser au maximum les performances.*
> - *INFO : Il contient les erreurs mais aussi les informations sur l'identifiant des fiches indexées ainsi que les requetes lancées et le nombre de résultats correspondants.*
> - *DEBUG : Version de deboggage en cas de problème avec le module. A ce niveau les performances de l'indexeur sont plus basses.*

log4j.appender.fichier.file

> *Emplacement et nom du fichier de logs. Remplacez [PATH_TOMCAT_INDEXING] par le chemin où se trouve le serveur Tomcat où sera installé le module ori-oai-indexing.*

configIndexing.xml

Il reste une dernière étape avant de déployer l'application. Il s'agit ici de remplir le fichier configIndexing.xml situé dans le dossier "properties". Pour cela il suffit simplement de remplacer [INDEXES_DATA_DIR] dans la balise "indexDir". Il faut également modifier [PROXY_HOST] et [PROXY_PORT] par la valeur adéquate. Si vous n'utilisez pas de proxy il suffit alors de supprimer ces valeurs. Enfin si vous souhaitez que le crawler se lance automatiquement, remplacez [INDEXING_SCHEDULE_CRAWLER] par la valeur de votre choix. Si vous supprimez la valeur, le crawler ne sera lancera pas. La section "Personnalisation de la configuration" offre plus de renseignements sur ce fichier ce qui permet une configuration plus poussée du module.

Déploiement de l'application

- Si c'est votre premier déploiement ou si vous souhaitez supprimer un index existant lancez : **ant init**

Un message vous préviendra que vous tentez de supprimer l'index. Appuyez sur "y" puis la touche "Entrée" pour continuer l'initialisation des répertoires.

- Lancez la commande **ant all** pour compiler les fichiers sources et créer le contexte du module d'indexation dans le serveur Tomcat.

Lancement du module

Il ne reste plus qu'à demarrer Tomcat pour lancer ORI-OAI-Indexing. Vous pouvez vérifier si le module fonctionne en testant sur un navigateur l'URL suivante : **http:// [HOST_INDEXING] :[PORT_INDEXING]/ori-oai-indexing/** en modifiant "[HOST_INDEXING]:[PORT_INDEXING]" par la valeur adéquate. Vous devriez obtenir un affichage similaire à celui-ci :

Le module est complètement chargé au bout de quelques secondes. S'il est sollicité trop tôt, une erreur 404 survient alors.

Configurat avancées

Voici les différentes étapes par lesquelles passe la configuration avancée du module:

Personnalisation de la configuration

Personnalisation de la configuration

Configuration générale

Le fichier configIndexing.xml contient la configuration générale de l'application. Il se trouve dans le dossier *properties*. Il se décompose en trois sections :

- Partie à modifier par l'utilisateur. C'est dans cette section que l'utilisateur indiquera notamment le répertoire où sera créé l'index ainsi que les paramètres du proxy.
- Options modifiables par l'utilisateur. Cette section propose divers éléments que l'utilisateur peut modifier s'il le souhaite comme le nombre de documents à indexer avant d'optimiser l'index.
- Partie à ne pas modifier par l'utilisateur. Cette section ne doit en aucun cas être modifiée car cela engendrerait des dysfonctionnements graves du module.
  Le fichier se présente de la manière suivante :

```
<config>

    <!-- PARTIE A MODIFIER PAR L'UTILISATEUR -->

        <!-- Repertoire de l'index -->
        <indexDir>[INDEXES_DATA_DIR]/index-indexing/index</indexDir>

        <!-- Repertoire des fichiers temporaires -->
        <tmpDir>[INDEXES_DATA_DIR]/index-indexing/tmp</tmpDir>

        <!-- Proxy -->
        <proxy>
            <proxyHost>[PROXY_HOST]</proxyHost>
            <proxyPort>[PROXY_PORT]</proxyPort>
        </proxy>

 <!-- Planification du crawler -->
 <scheduleCrawler>[INDEXING_SCHEDULE_CRAWLER]</scheduleCrawler>
 <!--scheduleCrawler>0 15 23 * * ?</scheduleCrawler-->


    <!-- OPTIONS MODIFIABLES PAR L'UTILISATEUR -->

    <!-- Fichier de configuration LIUS -->
        <liusConfigFile>liusConfig.xml</liusConfigFile>

        <!-- Frequence d'optimisation de l'index -->
```

```
        <frequencyOfOptimization>200</frequencyOfOptimization>

        <!-- Classe de transformation de requete -->

<queryTransformerClass>org.orioai.indexing.search.querytransformer.AccentRemoverTransformation</query
<!-- Formats de métadonnées -->
        <format id="dublin_core">
            <namespace prefix="dc" uri="http://purl.org/dc/elements/1.1/"/>
<namespace prefix="oaidc" uri="http://www.openarchives.org/OAI/2.0/oai_dc/"/>
<xpathToPlainText>//dc:relation</xpathToPlainText>
</format>

        <format id="formation">
            <namespace prefix="cdm" uri="http://www.w3.org/2001/XMLSchema-instance"/>
<xpathToPlainText />
        </format>

        <format id="pedagogique">
            <namespace prefix="lom" uri="http://ltsc.ieee.org/xsd/LOM"/>
<xpathToPlainText>//lom:technical/lom:location</xpathToPlainText>
</format>

        <!-- Chaines de remplacement -->
        <replacement stringToReplace=":" stringReplacement="@">
            <metadata>md-ori-oai-id</metadata>
            <metadata>md-ori-oai-namespace</metadata>
        </replacement>

        <replacement stringToReplace=" " stringReplacement="_">
            <formatId>pedagogique</formatId>
            <xpath>
//lom:classification/lom:taxonPath[lom:source/lom:string='dewey']/lom:taxon/lom:id</xpath>
</replacement>

        <replacement stringToReplace="vCardToUpperCaseORI" stringReplacement="vCardToUpperCaseORI"
>
            <formatId>pedagogique</formatId>
            <xpath>//lom:contribute/lom:entity</xpath>
</replacement>

 <!-- Crawler web -->

  <!-- Frequence d'optimisation du plein texte -->
  <frequencyOfOptimizationFullText>5</frequencyOfOptimizationFullText>

  <!-- Nombre d'essais de crawling d'une fiche -->
  <attemptsCrawling>3</attemptsCrawling>

  <!-- Nombre maximum de liens a indexer -->
  <nbMaxLinksToIndex>50</nbMaxLinksToIndex>

  <!-- Entrepots -->
  <repository name="INP Toulouse Theses">
   <xpathToUrl format_id="dublin_core" value="//dc:relation" />
<depth>1</depth>

<allowedMimeTypes>application/pdf,application/vnd.ms-powerpoint,application/msword</allowedMimeTypes>
</repository>

  <repository name="DSpace at MIT">
   <xpathToUrl format_id="dublin_core" value="//dc:identifier" />
<depth>1</depth>
   <allowedMimeTypes>application/pdf,application/vnd.ms-powerpoint</allowedMimeTypes>
  </repository>

  <repository name="ORI UNIT">
   <xpathToUrl format_id="pedagogique" value="//lom:technical/lom:location" />
<xpathToUrl format_id="dublin_core" value="//dc:identifier" />
<urlsToNotCrawl></urlsToNotCrawl>
   <depth>1</depth>

<allowedMimeTypes>application/pdf,application/vnd.ms-powerpoint,application/vnd.ms-excel,text/html</a
</repository>

  <repository name="ori-oai-workflow">
```

```
   <xpathToUrl format_id="dublin_core" value="//dc:identifier" />
<xpathToUrl format_id="pedagogique" value="//lom:technical/lom:location" />
<depth>1</depth>

<allowedMimeTypes>application/pdf,application/vnd.ms-powerpoint,application/msword</allowedMimeTypes>
</repository>

  <repository name="default">
   <xpathToUrl format_id="dublin_core" value="//dc:identifier" />
<xpathToUrl format_id="pedagogique" value="//lom:technical/lom:location" />
<depth>1</depth>
   <allowedMimeTypes>all</allowedMimeTypes>
  </repository>

  <!-- Entrepots a ne pas crawler -->
  <doNotCrawl></doNotCrawl>


    <!-- PARTIE A NE PAS MODIFIER PAR L'UTILISATEUR -->

        <!-- Nom des metadonnees communes aux modules ORI-OAI -->
        <static_metadatas>
            <doc_id>md-ori-oai-id</doc_id>
            <repository>md-ori-oai-repository</repository>
            <format>md-ori-oai-namespace</format>
            <datestamp>md-ori-oai-datestamp</datestamp>
            <score>md-ori-oai-score</score>
            <workflow_name>ori-oai-workflow</workflow_name>
            <notice_content>md-ori-oai-notice-content</notice_content>
            <crawled>md-ori-oai-crawled</crawled>
        </static_metadatas>

        <!-- Valeurs possibles du champ crawled -->
 <staticCrawledValues>
  <notCrawled>no</notCrawled>
  <notSuccessful>failed</notSuccessful>
  <unreachable>unreachable</unreachable>
  <unreachables>some_unrechables</unreachables>
  <forbidden>forbidden</forbidden>
  <successful>yes</successful>
 </staticCrawledValues>
```

```
</config>
```

Les parties static_metadatas ainsi que staticCrawledValues ne doivent pas être modifiées.

Le fichier se configure comme ceci:

indexDir

> *Chemin vers le répertoire dans lequel se trouve l'index, ici il s'agit de "[ORI_HOME]/data/indexes/index". Le répertoire "[ORI_HOME]/data/indexes" doit impérativement être créé mais le répertoire "Index" pourra normelement être créé par l'application au moment de l'indexation mais Il est possible en cas de problèmes que ce répertoire soit créé à la main.*

proxyHost

> *Paramètre du proxy. Si l'application ne passe pas par un proxy, il suffit d'effacer le contenu de la balise.*

proxyPort

> *Port du proxy. Si l'application ne passe pas par un proxy, il suffit d'effacer le contenu de la balise.*

scheduleCrawler

> *Donne la date et l'heure du lancement du crawling automatique. En laissant cette valeur à vide le crawling ne se lancera pas automatiquement.*

liusConfigFile

> *Fichier de configuration LIUS qui sera utilisé par ORI-OAI-Indexing. Par défaut il s'agit de liusConfig.xml qui fonctionne à la base avec les formats DC, LOM et CDM. Ici on ne renseigne pas le chemin mais simplement le nom du fichier. C'est pourquoi le fichier doit obligatoirement se trouver dans le répertoire properties. Pour plus d'informations sur ce fichier veuillez consulter la section suivante nommée "Configuration de Lius".*

frequencyOfOptimization

> *Permet d'optimiser l'index dès que l'on a atteint le nombre de fiches indiqué. Il est préférable de ne pas dépasser 500 sous peine de rendre l'index inutilisable.*

queryTransformerClass

> *Chemin vers la classe permettant de modifier la requete pour retrouver plus de résultats pertinents dans l'index. La classe donnée par défaut sert à supprimer les accents dans la requête pour gérer au mieux les caractères spéciaux de la langue française. Il est possible de créer une nouvelle classe qui héritera de la classe abstraite "QueryTransform" dans le but de modifier autrement la requête.*

format

> *Contient le namespace et le prefix d'une format de métadonnée.*

replacement

> *Contient les caractères à modifier dans une chaine provenant d'une métadonnée supplémentaire (md-ori-oai-...) ou provenant d'un xpath d'un format précis.*

Le couple "format - replacement" permet de gérer au mieux l'indexation et la recherche de documents. Pour comprendre au mieux son utilité prenons l'exemple du code Dewey. Il s'agit en réalité d'une donnée composée de chiffres et d'espaces (ex : 125 17.1). Or cette donnée n'est pas indexée correctement à cause du caractère d'espacement. Donc pour résoudre ce problème, ce caractère est remplacé par un autre caractère, ici "_" au moment de l'indexation. Pour remplacer un caractère par un autre, deux éléments sont nécessaires; le premier est le ou les formats concernés par le remplacement. Le second est le xpath permettant de retrouver la donnée à modifier. Dans notre exemple le format est le LOM dont l'identifant de format est "pédagogique". La donnée indexée sera alors "125_17.1". Il faut noter que ces remplacements de caractères ne modifient en rien la forme de la requête lors d'une recherche, c'est-à-dire dans notre cas que l'on recherche le code Dewey "125 17.1" et non pas "125_17.1".

frequencyOfOptimizationFullText

> *Permet d'optimiser l'index lorsqu'on indexe du texte intégral dès que l'on a atteint le nombre de fiches indiqué. L'indexation*

> *en texte intégral pouvant générer de gros fichiers, il est fortement conseillé d'optimiser régulièrement.*

attemptsCrawling

> *Donne le nombre de tentatives de crawling d'une même fiche. A chaque nouveau lancement de crawling une nouvelle tentative sera faite sur les fiches non crawlées. Cette fonctionnalité est utile lorsque un serveur distant n'est pas joignable lors du crawling.*

nbMaxLinksToIndex

> *Donne le nombre maximum de liens à indexer. Cette fonctionnalité est utile lors de l'indexation de texte intégral au format HTML.*

repository

> *Contient les informations nécessaires au bon crawling de fiches pour un entrepôt donné.*

doNotCrawl

> *Indique le nom des entrepôts à ne pas tenter de crawler.*

Pour plus d'informations concernant le crawling, veuillez consulter la page intitulée "Système de crawling".

> ⚠ L'identifiant et le namespace sont modifiés lors de l'indexation. En effet le caractère ":" est remplacé par "@". Ceci est utile dans le cadre d'une recherche. Un requête de type Lucene utilise le caractère ":" pour séparer l'attribut de la valeur (ex: "nom : toto"). Donc la valeur ne peut pas comporter la caractère ":" ce qui explique la modification. Mais du point de vue de l'utilisateur, il n'y a aucune modification de caractères à réaliser. Ceci ne sert qu'en interne au module d'indexation et est donc transparent pour tous les autres modules.

Configuration de Lius

Le fichier de configuration de LIUS, appelé liusConfig.xml, se trouve dans le dossier "Properties".

Selection de l'analyseur

Il permet d'indexer le contenu dans une langue donnée. Il est défini dans la balise "analyzer", elle-même incluse dans la balise "properties". Par défaut, c'est l'analyseur de la langue française qui est choisi. Cet analyseur a été modifié dans le cadre du projet ORI-OAI pour prendre en compte plus de spécificités manquantes de la langue française comme les mots composés par exemple.

> ⚠ Les autres éléments de la balise "properties" n'ont pas à être modifés.

Ajout d'un nouveau format xml

Ceci peut-être utile si vous souhaitez indexer un format différent du DC, du LOM ou du CDM, tous trois déjà présents par défaut. Pour se faire, ajoutez dans la balise "xml" le code suivant :

```
<xmlFile ns="[namespace]" setBoost="[valeur]">
    <indexer class="org.orioai.indexing.index.indexer.OriOaiXmlFileIndexer">
        <mime>text/xml</mime>
    </indexer>
    <fields>
    </fields>
</xmlFile>
```

namespace

> *Il s'agit du namespace correspondant au format*

valeur

> *Valeur décimale de boost. Le maximum est 2.0. L'attribut setBoost permet de prioriser un format par rapport à un autre. Cet attribut est optionnel; si il n'est pas ajouté, la valeur de boost sera 1.0. Le boost a une influence sur le calcul de la pertinence d'un résultat de la recherche*

Ajouter un nouveau format ne suffit pas à indexer le contenu d'une fiche. Il faut également ajouter les xpaths à indexer pour le format (voir section 5.2.3).

Ajout d'un nouveau xpath à indexer dans le format

Ceci peut-être utile si vous souhaitez indexer une métadonnée différente de celles présentes par défaut dans les formats DC, LOM, CDM ou d'un nouveau format que vous avez créé.

Pour se faire, trois étapes sont nécessaires (notez que le fichier liusConfig.xml est divisé en trois parties, correspondant aux trois balises "index", "search" et "searchResult") :

- Dans la **balise "fields"** (elle-même incluse dans la **balise "index"**), ajoutez ceci pour le format concerné :

```
<luceneField name="[xpath_encodé_UTF8]" xpathSelect="[xpath]" type="[type]" setBoost=
"[valeur]"/>
```

  xpath

  > *Il s'agit du xpath vers la métadonnée que l'on souhaite indexer.*

  xpath_encodé_UTF8

  > *Le nom de ce champ est par convention dans ORI-OAI le xpath qui a été encodé grâce à la méthode java.net.UrlEncoder.encode.*

  type

  > *Cet attribut indique le type de métadonnée. Plusieurs types sont possibles : "Text", "concatDate" s'il s'agit d'une date et "vcard" si la métadonnée contient une vCard.*

  valeur

  > *Valeur décimale de boost. Le maximum est 2.0. L'attribut setBoost permet de prioriser une métadonnée par rapport à une autre. Cet attribut est optionnel; s'il n'est pas ajouté, la valeur de boost sera 1.0. Le boost a une influence sur le calcul de la pertinence d'un résultat de la recherche*
  >
  > 
  > *L'attribut "setBoost" peut être à la fois dans la balise "format" et dans la balise "luceneField" d'un format.*

Pour vous faciliter l'encodage d'un xpath en UTF-8, un nouvel onglet appelé "Métadonnées et UTF-8" est présent dans le partie Visualisation du module.

- Dans la **balise "searchFields"** (elle-même incluse dans la **balise "search"**), créez une nouvelle ligne et ajoutez-y xpath_encodé_UTF8 précédé d'une virgule.

La virgule est indispensable au bon fonctionnement du module.

- Dans la **balise "fieldsToDisplay"** (elle-même incluse dans la **balise "searchResult"**), ajoutez ceci :

```
<luceneField name="[xpath_encodé_UTF8]" label="[xpath]"/>
```

  xpath

  > *Il s'agit du xpath vers la métadonnée que l'on souhaite indexer.*

  xpath_encodé_UTF8

  > *Le nom de ce champ est par convention dans ORI-OAI le xpath qui a été encodé grâce à la méthode java.net.UrlEncoder.encode.*

Il vous est maintenant possible d'utiliser les xpaths de type substring tels que : substring(//dc:date,1,4) dans le module d'indexation.

Système de surlignage

Il permet de mettre en valeur les résultats de votre recherche.
Pour activer/désactiver ce système il suffit de remplir l'attribut setHighlighter de la balise fieldsToDisplay à true/false (vers la fin du fichier).

Gestion des caches

Gestion des caches

Différents caches sont utilisés dans le module d'indexation. Ils servent tous à améliorer les performances de la recherche. Ils sont définis dans le fichier ehcache.xml placé dans le répertoire properties. Le fichier se présente comme suit :

```
<ehcache>

    <!-- Sets the path to the directory where cache .data files are created.

         If the path is a Java System Property it is replaced by
         its value in the running VM.

         The following properties are translated:
         user.home - User's home directory
         user.dir - User's current working directory
         java.io.tmpdir - Default temp file path -->
    <diskStore path="java.io.tmpdir"/>


    <!--Default Cache configuration. These will applied to caches programmatically created through
        the CacheManager.

        The following attributes are required for defaultCache:

        maxInMemory       - Sets the maximum number of objects that will be created in memory
        eternal           - Sets whether elements are eternal. If eternal,  timeouts are ignored
and the element
                            is never expired.
        timeToIdleSeconds - Sets the time to idle for an element before it expires.
                            i.e. The maximum amount of time between accesses before an element
expires
                            Is only used if the element is not eternal.
                            Optional attribute. A value of 0 means that an Element can idle for
infinity
        timeToLiveSeconds - Sets the time to live for an element before it expires.
                            i.e. The maximum time between creation time and when an element
expires.
                            Is only used if the element is not eternal.
        overflowToDisk    - Sets whether elements can overflow to disk when the in-memory cache
                            has reached the maxInMemory limit.

        -->


    <cache name="ori-oai-indexing_results"
        maxElementsInMemory="2"
        eternal="false"
        timeToIdleSeconds="900"
        timeToLiveSeconds="900"
        diskPersistent="true"
        overflowToDisk="true"
        />

    <cache name="ori-oai-indexing_fragments"
```

```
        maxElementsInMemory="5"
        eternal="false"
        timeToIdleSeconds="900"
        timeToLiveSeconds="900"
        diskPersistent="true"
        overflowToDisk="true"
        />

    <cache name="ori-oai-indexing_notices"
        maxElementsInMemory="5"
        eternal="false"
        timeToIdleSeconds="900"
        timeToLiveSeconds="900"
        diskPersistent="true"
        overflowToDisk="true"
        />


<cache name="ori-oai-indexing_nbResults"
        maxElementsInMemory="300"
        eternal="false"
        timeToIdleSeconds="7200"
        timeToLiveSeconds="7200"
        diskPersistent="true"
        overflowToDisk="true"
        />


    <defaultCache
            maxElementsInMemory="300"
            eternal="false"
            timeToIdleSeconds="7200"
            timeToLiveSeconds="7200"
            overflowToDisk="true"
            diskPersistent="false"
            diskExpiryThreadIntervalSeconds="60"
            memoryStoreEvictionPolicy="FIFO"
```

```
            />
</ehcache>
```

Quatre caches sont présents dans Ori-Oai-Indexing :

- Le cache de la recherche par attributs et de la recherche searchXMLDocs appelé ori-oai-indexing_results
- Le cache des fragments de la recherche plein texte appelé ori-oai-indexing_fragments
- Le cache de la recherche des fiches XML appelé ori-oai-indexing_notices
- Le cache du nombre de résultats des recherches appelé ori-oai-indexing_nbResults

Il vous est possible de modifier la durée de vie de ces caches en modifiant les valeurs *timeToIdleSeconds* et *TimeToLiveSeconds*. Par défaut trois de ces caches sont réglés à 15 minutes et le cache du nombre de résultats est réglé à deux heures.

Par ailleurs le cache persistent est placé dans le répertoire temporaire du Tomcat du module d'indexation.

Dans la version 1.5 la plupart des caches ont été désactivés. Seul le cache "ori-oai-indexing_nbResults" reste actif. Ceci a pour objectif de laisser Lucene gérer lui-même les caches.

Système de crawling

Système de crawling

Système de crawling

L'ajout d'un crawler web au module d'indexation permet d'enrichir la fiche en lui associant le ou les documents en texte intégral dont elle fait référence. Une métadonnée supplémentaire appelée "fullText" sera alors indexée. Vous pouvez réaliser un crawling manuel en vous rendant dans la page "Crawler" de la partie administration du module. Vous pouvez également lancer le crawling de manière automatique grâce à la balise "scheduleCrawler" depuis le fichier commons-parameters.properties ou le fichier configIndexing.xml si vous n'utilisez pas ori-oai-commons-quick-install.

La valeur "0 15 23 * * ?" vous permet de lancer le crawling tous les jours à 23h15. Il est recommandé de lancer cette tâche la nuit car c'est le moment où le module d'indexation et les serveurs distants sont moins sollicités. Pour plus d'informations sur la planification du crawling, veuillez consulter le lien suivant :
http://quartz.sourceforge.net/javadoc/org/quartz/CronTrigger.html

Si vous ne souhaitez pas lancer le crawler web il vous suffit de laisser la valeur INDEXING_SCHEDULE_CRAWLER du fichier commons-parameters.properties ou la balise "scheduleCrawler" du fichier configIndexing.xml (dans le cas d'une installation manuelle) vide.

Pour que le crawling soit efficace il faut également créer une balise "repository" dans ce fichier. Prenons l'exemple suivant :

```
<repository name="INP Toulouse Theses">
 <xpathToUrl format_id="dublin_core" value="//dc:relation" />
<urlsToNotCrawl></urlsToNotCrawl>
 <depth>1</depth>
 <allowedMimeTypes>application/pdf,application/vnd.ms-powerpoint,application/msword</allowedMimeTypes
```

Le nom de l'entrepôt doit être le même que la valeur indexée dans la métadonnée "md-ori-oai-repository(...)" de la fiche.

La balise "xpathToUrl" donne le xpath contenant l'URL vers le document plein texte. Il est possible de placer plusieurs balises de ce type. Dans ce cas la première balise sera prioritaire. Si une fiche est indexée dans différents formats, on regardera chaque balise xpathToUrl et la première qui correspondra à un format indexé sera utilisée pour retrouver le plein texte.

La balise "urlsToNotCrawl" permet de ne pas tenter de crawler certains serveurs. **Les valeurs doivent être séparées par des virgules**.

La balise "depth" indique la profondeur de crawling. Par défaut 1 est suffisant.

La balise "allowedMimeTypes" indique les types mime de documents plein texte que l'on souhaite indexer pour cet entrepôt. Dans notre exemple on autorise l'indexation de PDF, Microsoft PowerPoint et Microsoft Word. NB : Le module d'indexation n'est pas capable d'indexer tous les types de fichiers.

La balise "doNotCrawl" indique les entrepôts à ne pas tenter de crawler. Les valeurs, **séparées par des virgules**, doivent correspondre à la valeur de la métadonnée "md-ori-oai-repository(...)" des fiches provenant de cet entrepôt.

Dans la page "Visualisation de toutes les fiches" de la partie Administration du module une colonne indique l'état de crawling de chaque fiche. Les différents états sont :

- **no** : La fiche n'a pas encore été crawlée
- **yes** : La fiche a été crawlée avec succès
- **failed_x**: La crawling a échoué x fois.
- **unreachable**: Le document en texte intégral distant est injoignable
- **some_unreachables**: Certains liens sont injoignables
- **forbidden**: Le crawling est impossible sur cette fiche. Il peut s'agir d'un type mime non indexable ou d'une fiche dont l'entrepôt est interdit

Après une phase de crawling, l'index n'est pas toujours optimisé ce qui peut entrainer de legères baisses de performances lors de la recherche. Pour pallier à ce problème, un bouton "Opitmiser l'index" est maintenant disponible dans l'onglet "Gérer l'index" de la partie Visualisation du module.

Premiers tests

Premiers tests

Si le module est correctement installé lancez le tomcat :

```
[ORI_HOME]/tomcat-indexing/bin/startup.sh
```

Ouvrez un navigateur et tapez l'url :

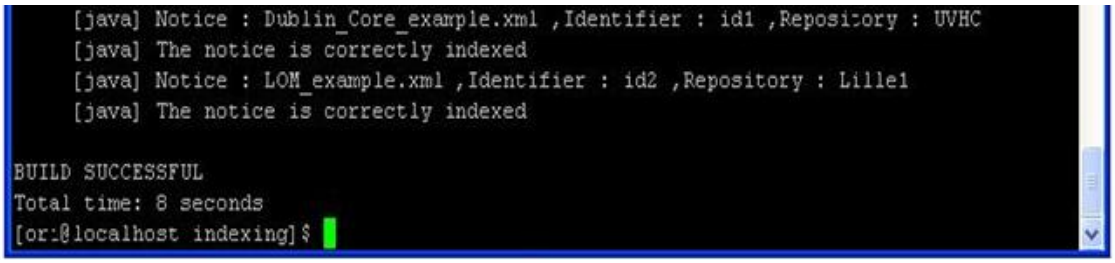
```
http://[HOST_INSTALL]:8182/ori-oai-indexing/
```

Vous devriez obtenir l'affichage suivant :

Pour vérifier si le module fonctionne correctement, placez-vous dans le répertoire **[ORI_HOME]/src/ori-oai-indexing-svn** puis tapez :

```
ant testIndex
```

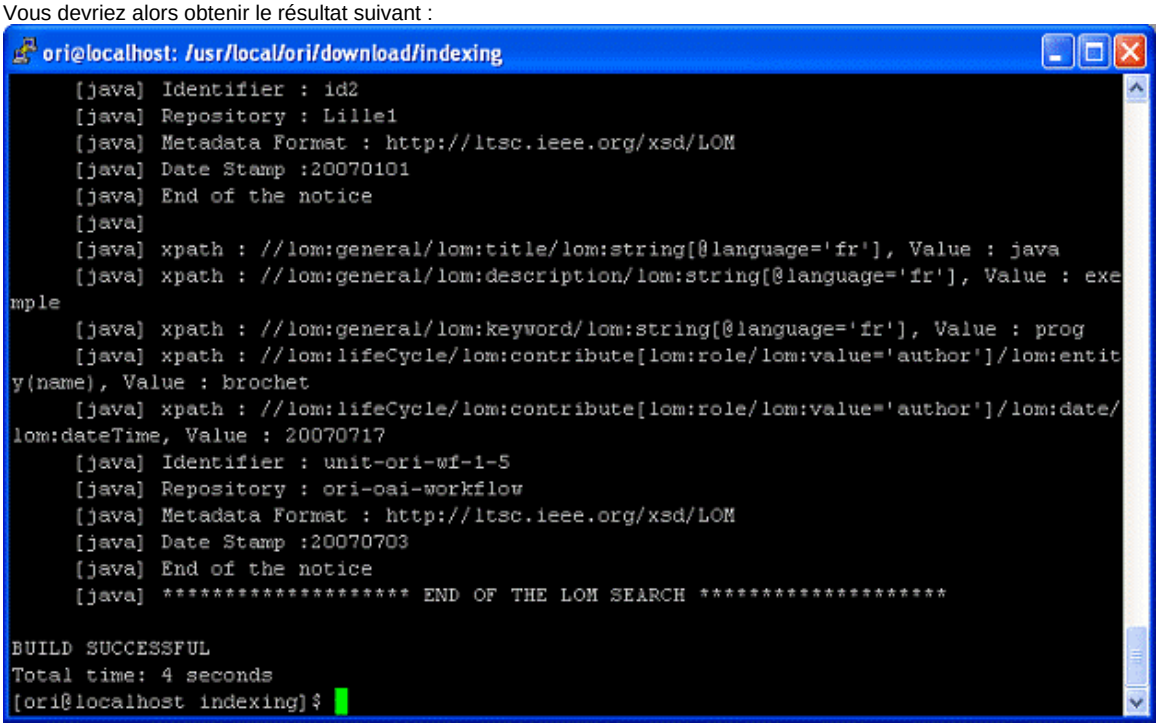
Deux documents vont alors être indexés. Si l'indexation s'est bien passée vous devriez avoir :



Il reste à vérifier si la recherche fonctionne correctement. Pour cela tapez :

```
ant testSearch
```

Vous devriez alors obtenir le résultat suivant :



Si tout s'est correctement déroulé, il vous faut réinitialiser votre index. Pour cela il vous suffit de vous rendre dans l'onglet "Gestion de l'index" de la partie Visualisation du module et de cliquer sur le bouton "Réinitialiser l'index". Après avoir confirmé votre choix l'index sera vidé.

NB : Vous pouvez visualiser votre index en vous rendant à la page **http:// [HOST_INSTALL] :8182/ori-oai-indexing/** et en consultant l'onglet "Visualisation de toutes les pages". Si vous cliquez sur l'identifiant d'une fiche, vous verrez alors apparaitre toutes les métadonnées indexées de celle-ci.

Important : Il est fortement recommandé de sauvegarder régulièrement votre index en copiant le dossier **index**. Si votre index devenait inutilisable, il vous suffirait alors de supprimer le dossier index et de le remplacer par votre copie la plus récente. Un redémarrage de votre serveur Tomcat hébergeant le module d'indexation sera nécessaire.

Il existe également des procédures de restauration de l'index depuis les modules ori-oai-workflow et ori-oai-harvester.

Aspects pratiques

Connexion au Web Service

Connexion au Web Service

Voici la méthode permettant de se connecter au module d'indexation :

```
private static OriOaiIndexingServiceInterface getService(String url) throws Exception {
    ObjectServiceFactory objectServiceFactory = new ObjectServiceFactory();
    Class classe = OriOaiIndexingServiceInterface.class;
    Service serviceModel = objectServiceFactory.create(classe);
    OriOaiIndexingServiceInterface port = (OriOaiIndexingServiceInterface) new XFireProxyFactory().create(serviceModel, url);

    return port;
}
```

Il ne reste plus qu'à ajouter dans le code l'appel à cette méthode en donnant comme URL celle du Web Service de l'indexeur :
OriOaiIndexingServiceInterface service = getService(url);

Ori-Oai-Indexing intègre la librairie ori-oai-commons.jar. Elle est indispensable pour se connecter au Web Service.

Méthodes publiques du web service

IndexOrUpdate

Indexation d'une fiche s'il s'agit de la première indexation de celle-ci ou mise à jour si elle est déjà présente. Elle est utilise pour ORI-OAI-Workflow car ce module ne gère pas la présence ou non d'une fiche dans l'index. L'index sera toujours optimisé lors de l'utilisation de cette méthode.

Les paramètres de cette méthode sont les suivants :

String metadataFile

> *Fiche de métadonnées à indexer*

String id

> *Identifiant associé à la fiche*

String namespace

> *Namespace de la fiche*

String datestamp

> *Date au format YYYY MM DD. Le caractère de séparation sera supprimé au moment de l'indexation et la date sera de la forme : YYYYMMDD. Il n'est pas possible d'indexer une objet de type date.*

String repository

> *Repository dans lequel se trouve la fiche. Si c'est une fiche locale, la valeur de ce paramètre sera "null".*

boolean doOptimize

> *Booléen à true lorsque l'index doit être optimisé. L'optimisation est une phase relativement longue (selon la taille de l'index).*

Cette méthode renvoie un entier indiquant le bon fonctionnement de l'opération.

Les différentes valeurs renvoyées sont :

- -1 en cas d'erreur. La fiche n'a pas pu être indexée ou mise à jour.
- 0 en cas de mise à jour d'un format. La réindexation de ce format avec les autres formats s'il y en a s'est bien déroulée.
- 1 en cas d'indexation réussie d'une nouvelle entrée de l'index ou d'un nouveau format.

IndexOrUpdate

Cette méthode est la même que la précédente mais ne contient pas de booléen d'optimisation de l'index. L'optimisation se fera alors à chaque appel de cette méthode.

Index

Depuis la version 1.5 **cette méthode est dépréciée**. En effet le module d'indexation est dorénavant capable de savoir si une fiche est indexée pour la première fois ou s'il s'agit d'une mise à jour. Il est préférable d'utiliser la méthode indexOrUpdate contenant le booléen d'optimisation notamment dans le cadre d'une moisson et de demander l'optimisation à la dernière fiche moissonnée.

Update

Depuis la version 1.5 **cette méthode est dépréciée**. En effet le module d'indexation est dorénavant capable de savoir si une fiche est indexée pour la première fois ou s'il s'agit d'une mise à jour. Il est préférable d'utiliser la méthode indexOrUpdate à la place de celle-ci.

DeleteNotice

Suppression d'une fiche dans l'index.

Les paramètres sont :

String id

> *Identifiant de la fiche à supprimer.*

Cette méthode permet de supprimer tous les formats indexés pour un identifiant donné. Elle renvoie un entier indiquant le bon fonctionnement de la suppression.

DeleteNotice

Suppression d'une fiche dans l'index.

Les paramètres sont :

String id

> *Identifiant de la fiche à supprimer.*

String namespace

> *Namespace du format a supprimer.*

Cette méthode renvoie un entier indiquant le bon fonctionnement de la suppression.

DeleteNotices

Suppression de plusieurs fiches de l'index. Pour chaque fiche à suprimer elle appelle la méthode deleteNotice Elle prend en paramètres :

String []ids

> *Tableau contenant les identifiants à supprimer*

Cette méthode renvoie un tableau d'entiers donnant le résultat de chaque suppression.

DeleteNotices

Suppression de plusieurs fiches de l'index. Pour chaque fiche à suprimer elle appelle la méthode deleteNotice Elle prend en paramètres :

String []ids

> *Tableau de d'identifiants à supprimer*

String []namespaces

> *Namespaces associés aux identifiants.*

Cette méthode renvoie un tableau d'entiers donnant le résultat de chaque suppression.

DeleteNotices

Suppression de plusieurs fiches de l'index. Pour chaque fiche à suprimer elle appelle la méthode deleteNotice Elle prend en paramètres :

String id

> *Identifiant dont on souhaite supprimer certains formats indexés*

String []namespaces

> *Namespaces des formats à supprimer*

Cette méthode renvoie un tableau d'entiers donnant le résultat de chaque suppression.

SearchForNumberOfResults

Nombre de résultats pour une requête donnée.

String request

> *Requete dont on cherche a connaitre le nombre de résultats*

Cette méthode renvoie un entier long correspondant au nombre de résultats d'une requête.

SearchForSomeNumberOfResults

Nombre de résultats pour un tablau composé de plusieurs requêtes.

String request[]

> *Tableau de requêtes dont on cherche a connaitre le nombre de résultats*

Cette méthode renvoie un tableau d'entiers long correspondant au nombre de résultats de chaque requête.

SearchXMLDoc

Récupère une fiche XML locale ou moissonnée.

Elle prend comme paramètres :

String id

> *Identifiant de la fiche.*

String namespace

> *Format de la fiche*

Cette méthode renvoie une chaîne de caractères correspondant à la fiche XML.

SearchXMLDoc

Récupère une fiche XML locale ou moissonnée.

Elle prend comme paramètres :

String id

> *Identifiant de la fiche.*

Cette méthode renvoie une table de hachage contenant pour chaque format indexé la fiche correspondante.

SearchXMLDoc

Récupère une fiche XML locale ou moissonnée.

Elle prend comme paramètres :

String id

> *Identifiant de la fiche.*

List<String> namespaces

> *Liste des formats de fiches à récupérer.*

Cette méthode renvoie une table de hachage contenant pour chaque format indexé demandé la fiche correspondante.

SearchXMLDocs

Récupère plusieurs fiches XML.

Les paramèrtes sont :

String request

> *Requête dont on souhaite obtenir les fiches correspondant aux résultats.*

int firstDocumentId

> *Numéro du premier document dans la liste des résultats à renvoyer.*

int lastDocumentId

> *Numéro du dernier document dans la liste des résultats à renvoyer.*

Cette méthode renvoie un objet de type SearchResults. Cette classe est présentée dans la section suivante.

SearchXMLDocs

Récupère plusieurs fiches XML.

Les paramèrtes sont :

String request

> *Requête dont on souhaite obtenir les fiches correspondant aux résultats.*

int firstDocumentId

> *Numéro du premier document dans la liste des résultats à renvoyer.*

int lastDocumentId

> *Numéro du dernier document dans la liste des résultats à renvoyer.*

Cette méthode renvoie un objet de type SearchResults. Cette classe est présentée dans la section suivante.

SearchXMLDocs

Récupère plusieurs fiches XML.

Les paramèrtes sont :

String request

> *Requête dont on souhaite obtenir les fiches correspondant aux résultats.*

int firstDocumentId

> *Numéro du premier document dans la liste des résultats à renvoyer.*

int lastDocumentId

> *Numéro du dernier document dans la liste des résultats à renvoyer.*

Cette méthode renvoie un objet de type SearchResults. Cette classe est présentée dans la section suivante.

SearchFromAttributes

Recherche par attributs. Ici on ne récupère pas de fiches mais certains attributs (ex : titre, auteur..) de cette dernière.

Les paramètres de cette méthode sont :

String request

> *Requête dont on souhaite connaitre les résultats.*

String sortAttributes[]

> *Attributs de tri de la liste de résultats. Si cette valleur est "null" alors les résultats seront triés par leur identifiant.*

int firstDocumentId

> *Numéro du premier document dont on souhaite connaitre les valeurs des attributs.*

int lastDocumentId

> *Numéro du dernier document dont on souhaite connaitre les valeurs des attributs.*

String []attributes

> *Attributs dont on souhaite connaitre la valeur.*

boolean ascending

> *Booléen à true si les résultats doivent être rangés dans l'ordre croissant.*

boolean highlightTerms

> *Booléen qui indique si on doit tenter de surligner la valeur des attributs.*

Cette méthode renvoie un objet de type SearchResults. Cette classe est présentée dans la section suivante.

En donnant "-1" comme valeur à firstDocumentId et lastDocumentId, tous les résultats sont alors retournés. Dans ce cas le temps de réponse peut être plus long si le nombre de résultats est important.

uniqueValues

Recherche de valeurs uniques de l'index. Elle est utilisée pour le recherche par auteur notamment.

String xpath

> *Xpath ou nom de métadonnée dont on souhaite connaitre toutes les valeurs uniques.*

Cette méthode renvoie un tableau de chaînes de caractères correspondant à toutes les valeurs uniques.

clearCache

Méthode qui vide les cache sans avoir à redémarrer Tomcat. Elle est utilisée en cas de tests de recherche.

String name

> *Nom du cache à vider. S'il vaut null alors tous les caches sont vidés.*

La classe SearchResults

Elle est utilisée lors de la recherche par attributs ou par la recherche de fiches XML. Elle contient un *entier long *correspondant au nombre de résultats d'une requête ainsi qu'une *liste d'objets de type SearchResult*. Pour les obtenir il faut respectivement utiliser les méthodes **getNumberOfResults()** et **getResults()**.

La classe SearchResult contient plusieurs chaînes de caractères : id, namespace, repository, datestamp que l'on récupère grâce à des méthodes get. Elle contient également une chaîne correspondant à la fiche et que l'on obtient grâce à la méthode **getNoticeContent()**. Elle est utilisée dans le cadre de la recherche de fiches XML. En ce qui concerne la recherche par attributs il faut utiliser **getAttributesValues()** qui renvoie une liste de liste de String. En effet chaque attribut peut avoir plusieurs valeurs (ex : plusieurs auteurs), et plusieurs attributs peuvent être demandés (ex : titre, auteur, description...). Donc l'élément renvoyé est bien une liste de liste de chaînes de caractères.

Contraintes d'utilisation

Contraintes d'utilisation

Pour un fonctionnement optimal du moteur d'indexation, certaines règles doivent être respectées.

- Lors d'une indexation, le paramètre datestamp doit être une chaîne de caractères de la forme YYYY/MM/DD. Le caractère de séparation n'a pas d'importance car il sera supprimé avant d'être indexé. Lors d'une recherche, la date aura alors la forme : YYYYMMDD.
- Pour indexer des fiches contenant des Vcards, il est indispensable que la chaîne de caractères correspondant à la fiche conserve l'indention de celle-ci. La chaîne de caractères ne doit donc pas être sur une seule ligne. Cette règle doit être également respectée

lors de la récupération d'une fiche.

- Une requête doit avoir une forme bien précise : "xpath:(valeur)". L'utilisation des parenthèses est indispensable pour la bonne utilisation des requêtes. Voici un exemple de requete correcte : "md-ori-oai-id:(mon_identifiant OR mon_autre_identifiant) AND md-ori-oai-namespace:(mon_namespace)". De plus il ne doit pas y avoir d'espaces entre le xpath et le caractère de séparation ":".
- Le xpath d'une requête doit être encodé (grâce à la méthode java.net.URLEncoder.encode). Par exemple : "//dc:title:(mon_titre)" doit s'écrire : "%2F%2Fdc%3Atitle:(mon_titre)".

> Il est impératif de sauvegarder très régulièrement l'index. Pour cela il faut conserver tout le répertoire contenant l'index. Il se trouve au niveau du chemin renseigné dans la balise *indexDir* du fichier configIndexing.xml.

Utilisation

Administration du module d'indexation

Quelques fonctionnalités permettent de consulter l'index depuis le navigateur grâce à l'adresse suivante : http://HOST_INDEXING:PORT_INDEXING/CONTEXT_INDEXING.

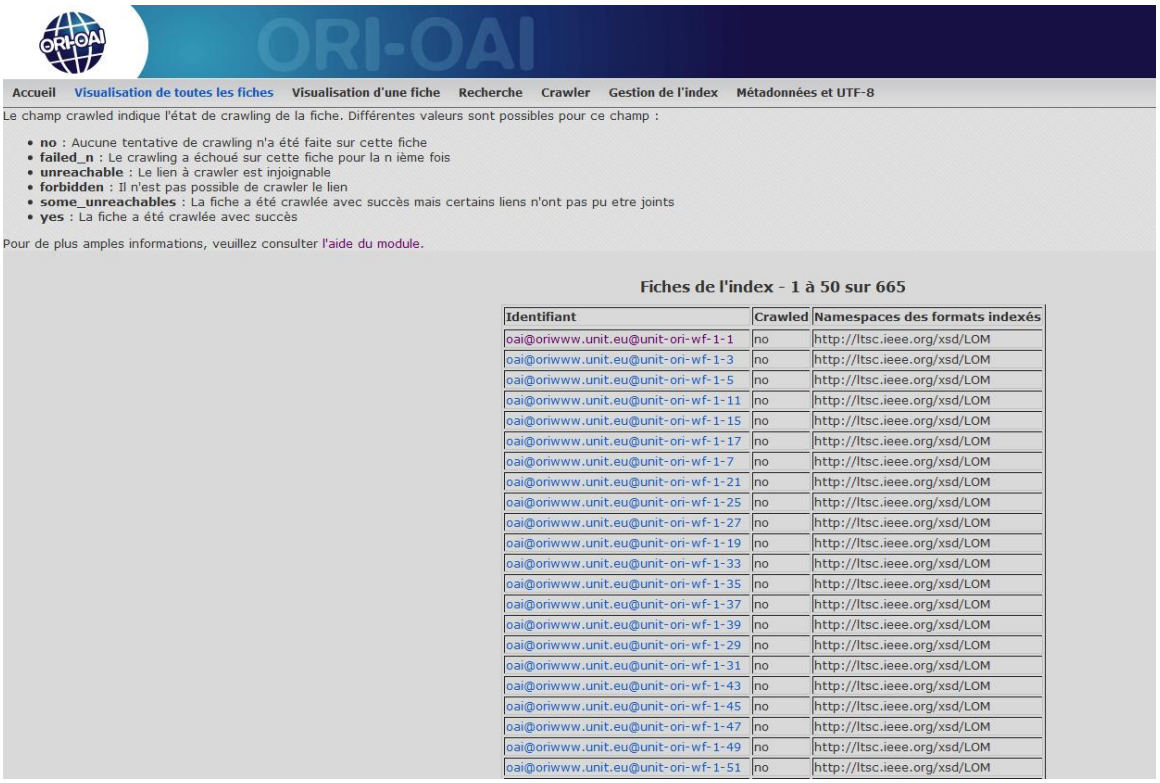
- **Accueil**

La page se présente de la manière suivante :

Cette page indique que le module a été correctement déployé et qu'il est prêt à être utilisé. Cette page donne également le lien vers le site du projet ORI-OAI ainsi que le lien vers la documentation du module.

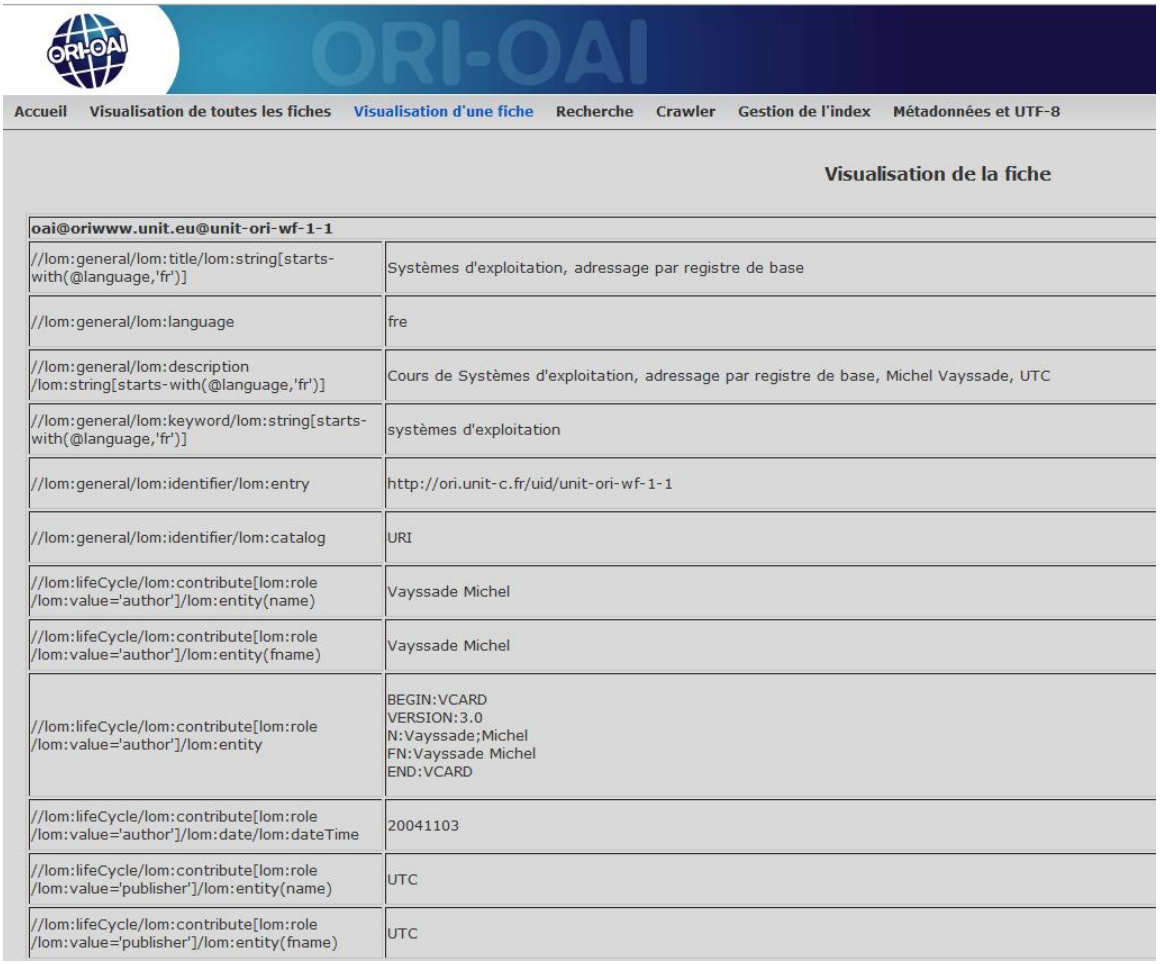
- **Visualisation des fiches**

Cet onglet montre toutes les fiches présentes dans l'index. Voici une copie d'écran de cette page :



Il s'agit d'un tableau renseignant sur l'identifiant de chaque fiche indexée, son état de crawling, ainsi que les formats dans lesquels l'identifiant a été indexé. On peut remarquer dans cet exemple que les fiches ont été indexées au format Dublin Core. Il faut noter que les identifiants ne sont pas triés dans l'ordre alphabétique mais dans l'ordre dans lequel ils ont été indexés, c'est-à-dire dans l'ordre croissant de leur position dans l'index Lucene.
Pour augmenter les performances de l'affichage de cette page, seules 50 fiches sont présentées par page. Vous trouverez en bas de la page des liens "page suivante" et/ou "page précédente" pour naviguer entre les différentes pages.

Par ailleurs les identifiants des fiches sont des liens cliquables. En cliquant sur l'un de ces identifiants, on peut en visualiser toutes ses données. On passe alors dans l'onglet "Visualisation d'une fiche". Si on clique sur l'onglet "Visualisation d'une fiche" un message indique qu'il faut retourner dans la page "Visualisation de toutes les fiches" et cliquer sur un identifiant pour voir la fiche complète. La page se présente comme suit :

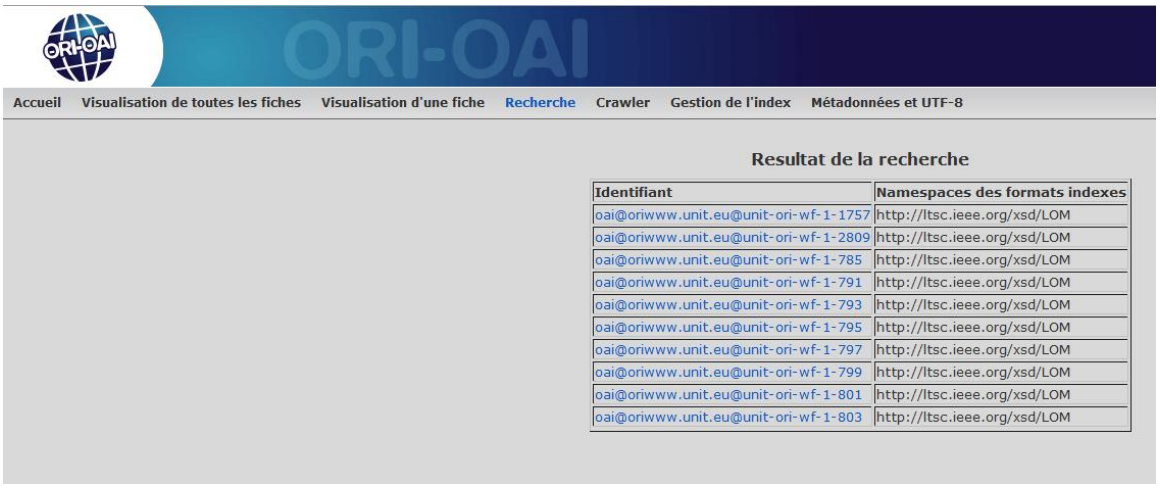


La partie gauche présente les métadonnées indexées et la partie droite montre leur contenu. Ceci permet de vérifier notamment que la fiche est bien indexée et que toutes les métadonnées sont dans l'index.

- **Recherche**

Cette page permet de lancer une requête Lucene. Pour se faire il suffit d'entrer votre requête dans le formulaire et de cliquer sur "Lancer la recherche". Voici une copie d'écran de cette page :

Il faut noter que le nom des métadonnées doit être encodé en UTF-8. Ainsi "//dc:title" devient "%2F%2Fdc%3Atitle". Ceci a pour objectif de ne pas utiliser le caractère ":" qui est déjà sollicité dans une requête Lucene pour faire la séparation entre la métadonnée et le contenu recherché.
La copie d'écran suivante montre les résultats de la recherche :



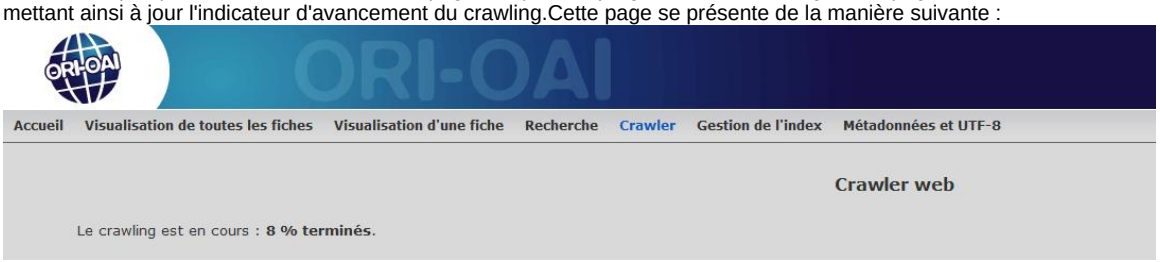
Les résultats se présentent sous la forme d'un tableau contenant l'identifiant de la fiche ainsi que les formats indexés. Là encore l'identifiant est un lien cliquable qui aboutit à la visualisation de la fiche.

- **Crawler**

Cet onglet permet de lancer le crawling. La copie d'écran suivante montre la page de crawling :

Pour lancer le crawling manuellement il suffit de cliquer sur le bouton "Lancer le crawling".

Au bout de quelques secondes une nouvelle page indiquera la progression du crawling. Cette page se rafraichit toutes les 10 secondes mettant ainsi à jour l'indicateur d'avancement du crawling.Cette page se présente de la manière suivante :



1 - Quitter cette page ne stoppe pas le crawling.

2 - Il n'est plus possible de voir l'état d'avancement du crawling après avoir quitté cette page.

* h3. **Gestion de l'index**

Cette page est utile lorsque vous souhaitez réinitialiser l'index ou lorsque vous faites des recherches. Les caches permettent de ne pas trop solliciter l'index. Si une requête est lancée plusieurs fois, la première servira à mettre en cache les résultats de la requête. Les suivantes ne feront que consulter le cache. Trois boutons sont présents :

- Réinitialiser l'index : cela vous permettra de supprimer l'index et de le recréer à vide. Le cache sera lui aussi vidé.

- Optimiser l'index : ce bouton lancera l'optimisation de l'index. Cette étape peut être utile après la phase de crawling.

- Vider le cache : cela supprimera toutes les entrées du cache. Cette fonctionnalité est utile notamment si l'index a été modifié après avoir effectué des recherches. Vider le cache permettra de prendre en compte ces modifications dans les résultats des recherches.

Un message de confirmation apparaitra alors et ce sur chacun des deux boutons.

> ⚠ Depuis la version 1.4, arrêter et relancer le Tomcat du module d'indexation ne permet plus de vider le cache.

La page de gestion de l'index se présente comme suit :

- **Métadonnées et UTF-8**

Cette page vous permet d'encoder ou de décoder facilement vos xpaths en UTF-8. Ceci est utile lorsque vous souhaitez ajouter un nouveau xpath dans le fichier liusConfig.xml. La page se présente comme suit :

Pour encoder un xpath en UTF-8, entrez sa valeur dans le formulaire (exemple : //dc:title) et cliquez sur le bouton "Lancer l'opération". La copie d'écran suivante présente le résultat :

Il est possible d'effectuer l'opération inverse en séletionnant "decode" et en donnant le xpath encodé.