Space Details

Key:	ORIOAIsearch
Name:	ORI-OAI-search
Description:	
Creator (Creation Date):	ycolmant@univ-valenciennes.fr (Jul 04, 2008)
Last Modifier (Mod. Date):	ycolmant@univ-valenciennes.fr (Nov 25, 2008)

Available Pages

- Version 1.1
  - Spécifications
  - Installation
    - Test
  - Utilisation

Version 1.1

This page last changed on Nov 25, 2008 by ycolmant@univ-valenciennes.fr.

ORI-OAI-search : Recherche de documents locaux et distants

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

Voir l'architecture du système

Composants obligatoires

- **ORI-OAI-indexing** pour faire des recherches sur l'index
- **ORI-OAI-vocabulary** pour la construction des interfaces de recherche et la traduction de certains termes dans les résultats

Description

Ce module offre une interface graphique pour la recherche de documents dans le système. Dialoguant avec le module ORI-OAI-indexing, il génère des requêtes dans la syntaxe Lucene et affiche les documents retrouvés.

Ce composant est entièrement configurable en ce qui concerne les formats de documents que l'on souhaite rechercher, et les types de recherches que l'on veut proposer à l'utilisateur. Il existe trois types de recherche :

- Par date : on propose à l'utilisateur de rechercher des documents suivant leur date de création, modification, etc. Ce type est utilisé pour afficher les nouveautés.
- Avancée : il est possible de configurer différents formulaires de recherche avec des critères plus ou moins avancés. On pourra proposer par exemple un formulaire de recherche composé d'un seul champ permettant une recherche sur le document complet et les métadonnées associées, ou encore un formulaire de recherche avancée proposant des champs de recherche pour chacune des métadonnées d'un format de description.
- Thématique : dans ce type de recherche, on ne demande aucune saisie à l'utilisateur. Elle est mise en place pour faire des recherches suivant des classifications de documents, des auteurs, des mots-clefs, etc. Elle peut être sollicitée par les utilisateurs souhaitant découvrir les documents référencés dans le système n'ayant aucun critère de recherche particulier. Par ce type de recherche, on guide également l'utilisateur dans ses recherches en ne proposant que les valeurs réellement indexées comme par exemple dans le cas de la recherche par mots-clefs.

Ce module permet donc une recherche multicritères et l'export des résultats de recherche sous forme de flux RSS, dans un catalogue au format RTF ou encore l'export des fiches de métadonnées en XML.

Notons également que le module peut être décliné en deux versions : servlet pour une installation standard sur un serveur Web et portlet pour une intégration dans un Environnement Numérique de Travail.

Voir la documentation technique

Spécifications

This page last changed on Nov 25, 2008 by ycolmant@univ-valenciennes.fr.

Documentation à venir ...

Installation

This page last changed on Nov 25, 2008 by ycolmant@univ-valenciennes.fr.

Modifications depuis la version 1.0.1

Fonctionnalités

Les nouvelles fonctionnalités qui ont été ajoutées dans cette version sont:

- gestion des flux RSS
- affiche la requête de la recherche avancée dans la page de résultats
- affiche le champ de recherche simple déjà lancé
- interprétation du code HTML contenu dans des champs de métadonnées dans les pages de résultats
- modification des CSS pour prendre en compte le highlighting quand il est utilisé dans le module ori-oai-indexing
- ajout d'un nouveau mode de recherche thématique sous forme d'arbre
- ajout en mode servlet de la sortie RTF de toutes les ressources dans une thématique donnée
- ajout d'un sitemap dynamique pour les moteurs de recherche type google en mode servlet
- recherche thématique simple sans passer par un vocabulaire
- possibilité de mapper en mode portlet (pour la recherche thématique qui ne passe pas par un vocabulaire) un attribut LDAP par sa valeur en fonction de l'utilisateur connecté
- négation sur la recherche dans un champ caché
- ajout de la possibilité de mapper en mode portlet une valeur de recherche d'un champ caché par sa valeur en fonction de l'utilisateur
- possibilité de filtrer l'accès aux menus de recherche sur un groupe du portail
- ajout de la possibilité de visionner la fiche XML d'origine
- possibilité de rebondir sur une notice depuis une autre notice comme par exemple dans le cas des relation du LOM
- ajout d'un filtrage d'accès en fonction des attributs de la personne connectée pour l'accès à des champs ou des groupes de champs d'une recherche avancée
- passage des interfaces en utf-8

Fichier de configuration

Les nouvelles fonctionnalités ainsi que différentes améliorations de l'outil nécessitent une adaptation du fichier de configuration **config.xml**:

- la configuration des connexions vers les autres modules se fait maintenant dans le fichier **services.properties** (cf.)
- gestion des flux RSS nécessite la configuration de la nouvelle balise **rss_fields** (cf.)
- ajout du paramètre **keep_navigation_session** dans la balise **thematic_search** pour dire si on garde le noeud de la dernière navigation en session
- ajout d'un nouveau mode de recherche thématique sous forme d'arbre: ajout du paramètre **thematicFullTree** dans les fichiers **properties/advanced/*** et **full_tree** dans **thematic_search** du fichier **properties/config.xml**
- ajout de l'attribut **formatsVocabulary** dans la balise **href_formats** pour gérer différents formats de métadonnées pour un meme identifiant
- suppression du champ **vocabulary_cache** qui se configure maintenant dans **ehcache.xml** (cf.)
- ajout de **value** dans la recherche thématique pour éviter de passer par un vocabulaire (cf.)
- possibilité de mapper en mode portlet (pour la recherche thématique qui ne passe pas par un vocabulaire) un attribut LDAP par sa valeur en fonction de l'utilisateur connecté (cf.)
- ajout du paramètre **not** dans les **hidden_field** pour faire de la négation sur la recherche dans un champ caché (cf.)
- ajout de la possibilité de mapper en mode portlet une valeur de recherche d'un champ caché par sa valeur en fonction de l'utilisateur (cf.)
- ajoute la possibilité de filtrer l'accès aux menus de recherche sur un groupe du portail par la valeur **portal_group** sur les balises **allowed** (cf.)
- ajoute la possibilité de visionner la fiche XML d'origine. L'attribut **showXmlLink** dans les balises **format** permet de dire si on veut exporter la fiche XML ou non (cf.)
- possibilité de rebondir sur une notice depuis une autre notice comme par exemple dans le cas des relation du LOM. Pour cela, ajout de la balise **notice_link** en fichier de config (cf.)
- ajout des balises **authorization** dans les fichiers de configuration des recherches avancées dans les balises **field** ou **group** (cf.)

Autres

Les autres modifications principales sont:

- utilisation de EHcache pour la gestion des vocabulaires (cf.)
- ajoute du cache sur la notice récupérée et transformée pour gagner en performance lors de l'affichage des notices
- utilisation de **ori-oai-commons-0.2.0.jar** pour les nouvelles interfaces des Web Services

Configuration et déploiement du module

Il existe plusieurs modes d'installation du module ori-oai-search. Tout d'abord en utilisant ori-oai-commons-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.

Installation rapide avec ori-oai-commons-quick-install

Ce mode d'installation permet une installation rapide de ORI-OAI en saisissant tous les paramètres généraux aux modules dans un seul fichier de configuration; Vous pourrez compléter votre installation et vos paramétrages par la suite sans aucun soucis.

Reportez-vous à la documentation en ligne de ori-oai-commons-quick-install.

Installation manuelle

Ce mode d'installation vous permet dès le départ une installation et des configurations poussées du module ori-oai-search.

La configuration générale du projet se fait dans le dossier **properties**. Ce dossier est découpé comme suit *(les autres fichiers ne sont pas à modifier lors du déploiement)* :

- Le fichier **init-build.properties** est celui utilisé pour la configuration du déploiement de l'application
- Le fichier **config.xml** est celui utilisé pour la configuration générale de l'application
- Le fichier **services.properties** permet de définir les paramètres de connexion aux autres modules ORI-OAI
- Le fichier **log4j.properties** permet de configurer l'emplacement et le niveau de logs
- Le fichier **ehcache.xml** définit les propriétés de gestion des différents caches dans le module
- Le dossier **advanced** contient tous les formulaires de recherche simple et recherche avancée
- Le dossier **messages** contient les bundles de messages dans toutes les langues gérées par l'application

Configuration minimale

Le module ori-oai-search est pré-configuré pour répondre au mieux aux attentes des différents établissements. Lors du premier déploiement, vous n'avez donc qu'à

- éditer le fichier de déploiement **init-build.properties**
- éditer le fichier **services.properties** pour vous connecter aux autres modules du projet

Modification de init-build.properties

Dans le cas où vous ne voulez pas bénéficier de la configuration par ori-oai-commons-quick-install, 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
#commons.parameters.central.file.url=[COMMONS_PARAMETERS_CENTRAL_FILE_URL]
```

Les autres paramètres de ce fichier sont:
deploy.home

Dossier webapps dans lequel vous voulez déployer l'application. En mode portlet, ce doit être le même que celui où est déployé votre contexte uPortal.

app.name.deploy

Nom du contexte que vous voulez donner à l'application.

deploy.type

Ce paramètre permet de dire si on déploie le module en mode *servlet* ou en mode *portlet*.
Le mode *servlet* permet de déployer le module dans un environnement standalone en dehors de tout ENT ou autre moteur de portlets.
Le module ori-oai-search est aussi disponible en version *portlet*. Il peut donc être intégré dans tout moteur de portlet comme un ENT ou un CMS.

configuration.file

Ce paramètre permet de définir la configuration que vous utiliserez pour les interfaces. Il correspond à un fichier de configuration **config.example.???.xml** , il faut aller dans le dossier **properties**. Il existe plusieurs configurations de base possibles (que vous pouvez adapter par la suite):

	config.example.complet.xml Configuration qui offre 4 menus de recherche: les ressources au format Dublon Core, LOM, CDM, et enfin un

	menu permettant la recherche croisée sur tous les formats. **config.example.unt.xml** Une configuration proposée pour les UNT. Elle permet des recherches thématiques, de nouveautés et avancées sur toutes les ressources des établissements partenaires. **config.example.documentaire.xml** Permet de faire des recherches uniquement sur les ressources documentaires au format Dublin Core et LOM (locales et distantes). **config.example.formations.xml** Permet la recherche parmis des formations au format CDM. Pour utiliser une de ces configurations, saisissez le nom de la configuration que vous voulez utiliser dans le paramètre configuration.file

Configurer la connexion aux autres modules

Dans le cadre de la communication entre ORI-OAI-search et les autres modules, il suffit de configurer les 2 paramètres suivants dans le fichier **properties/services.properties** :
indexing.ws.wsdlDocumentUrl

Correspond à l'adresse du Web service en frontal du module ORI-OAI-indexing. Sa valeur est du type **http:// [HOST_INDEXING] :[PORT_INDEXING]/[CONTEXT_INDEXING]/ xfire/IndexingService?WSDL**

- Exemple: **http://localhost:8080/ori-oai-indexing/xfire/IndexingService?WSDL**

vocabulary.ws.wsdlDocumentUrl

Cette valeur est l'adresse du Web service en frontal du module ORI-OAI-vocabulary.
La valeur est de type **http:// [HOST_VOCABULARY] :[PORT_VOCABULARY]/ [CONTEXT_VOCABULARY]/xfire/OriVocabularyService?WSDL**

- Exemple: **http://localhost:8080/ori-oai-vocabulary/xfire/OriVocabularyService? WSDL**

Configuration complémentaire

Configurer les logs

La configuration de l'emplacement et du niveau des logs se fait dans le fichier **properties/ log4j.properties**. Les paramètres à adapter sont dans la rubrique **logs ori-oai-search**. Vous pouvez trouver plus de documentation en ligne sur la syntaxe log4j à cette adresse: http://logging.apache.org/log4j/docs/manual.html.

Niveau de logs

Dans la rubrique **level**, vous pouvez changer le niveau d'affichage des logs: **INFO**, **DEBUG**, **ERROR**, etc. Il est préconisé d'utiliser le mode **INFO** en production et **DEBUG** lors des phases de test et de débogage.

Console ou fichier

Dans la rubrique **console or file** vous pouvez choisir que les messages de logs s'affichent dans la console de Tomcat ou dans un fichier de logs.

Pour afficher dans la console, vous devez commenter (en faisant commencer la ligne par un #) chaque ligne sous la rubrique **file**.

Pour afficher dans un fichier, vous commentez la ligne sous la rubrique **console**. Ensuite, vous spécifiez le nom et chemin du fichier dans le paramètre **log4j.appender.orioaisearch.File**, la taille maximale du fichier dans **log4j.appender.orioaisearch.MaxFileSize** et le nombre maximal de backups de fichiers de logs dans **log4j.appender.orioaisearch.MaxBackupIndex**.

Pattern

Vous pouvez si vous le désirez changer le pattern d'affichage des logs par le paramètre **log4j.appender.orioaisearch.layout.ConversionPattern** dans la rubrique **pattern** en vous reportant à la syntaxe log4j.

Une configuration pour tester le module seul

Dans le cas d'un premier déploiement, il est prévu de pouvoir tester le module de recherche indépendamment des autres modules. En effet, ceci permet de vérifier que l'application est bien configurée et bien déployée sans dépendre des autres entités du système. Cette manipulation fait que tous les vocabulaires et resultats de recherche sont statiques et avec des libellés sous forme de tests.

Pour faire cette manipulation, il faut changer une configuration des beans Spring. Cette configuration concerne les classes utilisées pour la connexion au module ORI-OAI-vocabulary et celle pour la recherche à partir de ORI-OAI-indexing. Vous devez donc à la place utiliser des classes "Mock" c'est-à-dire des classes qui simuleront les connexions aux autres modules.

Editez le fichier **webapp/WEB-INF/search-servlet.xml** ou **webapp/WEB-INF/search-portlet.xml** que vous soyez en déploiement *servlet* ou *portlet*.

Recherchez l'occurence du terme **id="vocabulary-service"**. Vous trouvez alors la configuration du bean Spring désiré. Il suffit de commenter la configuration du bean correspondant à l'implémentation et de dé-commenter celui appelant la classe de test comme suit:

```
<!-- ****************************************************** -->
<!-- implémentation de la couche service pour le vocabulaire -->

<!-- Implementation -->
<!--bean id="vocabulary-service"
 class="org.orioai.search.vocabulary.service.VocabularyServiceImpl">
        <property name="configurationService">
                <ref local="configuration-service"/>
        </property>
        <property name="oriOaiVocabularyService">
                <ref local="ori-oai-vocabulary-service"/>
        </property>
</bean-->

<!-- Classe de test -->
<bean id="vocabulary-service"
 class="org.orioai.search.vocabulary.service.MockVocabularyServiceImpl">
</bean>
<!-- ****************************************************** -->
```

Ensuite, vous devez faire la même chose pour le bean servant à la recherche en cherchant l'occurance du terme **id="search-service"** comme suit:

```
<!-- ****************************************************** -->
<!-- implémentation de la couche service pour la recherche dans le moteur de recherche -->

<!-- Implementation -->
<bean id="search-service" class="org.orioai.search.indexer.search.service.LuceneSearchServiceImpl">
        <property name="configurationService">
                <ref local="configuration-service"/>
        </property>
</bean>

<!-- Classe de test -->
<!--bean id="search-service"
 class="org.orioai.search.indexer.search.service.MockSearchServiceImpl">
        <property name="configurationService">
                <ref local="configuration-service"/>
        </property>
</bean-->
<!-- ****************************************************** -->
```

Déploiement de l'application

L'application a été développée et testée à partir d'une JVM 1.5 et d'un Tomcat 5.5. Notez que l'installation de ANT (http://ant.apache.org/) est un pré-requis au déploiement.

Important

Pour plusieurs déploiements du module ori-oai-search dans un même Tomcat, il est nécessaire d'affecter une clef différente à chaque instance.

Pour cela, dans le fichier **webapp/WEB-INF/web_servlet.xml** si on est en mode de déploiement servlet, ou **webapp/WEB-INF/web_portlet.xml** si on est en portlet, il faut affecter une valeur unique au paramètre **webAppRootKey** dans la partie suivante:

```
<context-param>
        <param-name>webAppRootKey</param-name>
        <param-value>ori-oai-search.servlet</param-value>
</context-param>
```

Important

Il est nécessaire de spécifier au Tomcat que vous utilisez l'encodage UTF-8 pour tous les modules. Pour cela, vous éditez le fichier **[TOMCAT_HOME]/bin/startup.sh** (startup.bat sous Windows) ou **[TOMCAT_HOME]/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)

Une fois cette configuration faite, executez les commandes suivantes:

```
ant init-build
```

initialisera le fichier build.properties

```
ant all
```

déploiera l'application dans le serveur Tomcat. Il n'y a plus qu'à démarrer ce serveur.

Précisions sur le mode portlet

Le module ori-oai-search est aussi disponible en version portlet. Il peut donc être intégré dans tout moteur de portlet comme un ENT ou un CMS.

Intégration dans ESUP Portail

Vous devez mentionner ce contexte dans le fichier [TOMCAT_HOME]/conf/server.xml . Par exemple:

```
<Context path="/ori-oai-search" docBase="C:/esupdev/esupdev-2.5/webapps/ori-oai-search"
 crossContext="true" reloadable="false" />
```

Enfin, vous devez publier cette portlet dans votre ENT. Voici les paramètres à utiliser:

- **Channel type** : Portlet
- **Channel Functional Name** : ori-oai-search
- **Portlet definition ID** : ori-oai-search.spring

Intégration dans infoglue

Le CMS infoglue supportant la norme des portlets JSR-168, il est possible d'intégrer le module ori-oai-search en mode portlet sous réserve d'avoir accepté l'option JSR-168 lors de l'installation. Ceci est intéressant pour associer dans un unique site des pages à la fois statiques mais aussi dynamiques comme les interfaces de recherche.

Pour cela, vous devez tout d'abord supprimer ou commenter tout le bloc comprenant les balises **<user-attribute>** dans le fichier **webapp/WEB-INF/portlet.xml**.

Il faut ensuite créer une archive **ori-oai-search.war** de votre module après l'avoir correctement configuré. Il est conseillé de commencer par un déploiement en mode servlet au préalable afin de bien s'assurer que les configurations apportées sont correctes et définitves. La création de l'archive se fait en utilisant la commande :

```
ant buildwar-portlet
```

Le fichier généré **ori-oai-search.war** se trouve alors à la racine de votre module.

> **Important**
>
> L'encodage des formulaires de votre portlet doit être **utf-8** dans infoglue. Vous devez donc fixer la valeur **utf-8** à la propriété **Input character encoding (deliver)**. Celle-ci est accessible depuis le menu **Administration**, **Application settings** et **Editer propriétés**.

Vous pouvez maintenant déployer cette portlet dans infoglue:

- Allez dans l'onglet **Administration**

- Puis le menu **Portlets**
- Choisissez Nouveau portlet et donnez les paramètres suivants
  - **Nom du contexte** : ori-oai-search
  - **Le fichier War contenant les portlets** : le fichier **ori-oai-search.war** généré
    Votre portlet a alors été ajoutée à infoglue.
- Pour prévenir tout problème de déploiement, il est conseillé de redémarrer votre serveur à cette étape

Vous pouvez maintenant tester cette intégration.
Créer un template dans l'onglet **Contenu** :

- Choisissez **Nouveau contenu** puis utilisez les paramètres suivants:
  - **Nom du contenu** : ori-oai-search-portlet
  - **Type de contenu** : HTMLTemplate
- Dans les propriétés qui suivent, renseignez:
  - **Name** : ori-oai-search-portlet
  - **Template HTML** :

```
#set($portletName = "ori-oai-search.spring")
#set($myPortlet = $portalLogic.getPortletWindow($portletName, "ori-oai-search.spring"))
$myPortlet.setAttribute("componentId", "ori-oai-search")
$myPortlet.setAttribute("languageCode", $templateLogic.locale.language)
$myPortlet.setAttribute("siteNodeId", $templateLogic.siteNodeId)

<html>
        <head>
                <link href="http://[HOST_SEARCH]:[PORT_SEARCH]/[CONTEXT_SEARCH]/css/ori-
oai-search.css" rel="stylesheet" type="text/css"/>
        </head>
        <body>
                $myPortlet.render()
        </body>
</html>
```

Notez qu'il est impératif d'ajouter les tags html, head et body car la portlet ne génère pas ce code. Vous devez aussi faire référence à la bonne CSS en adaptant l'URL proposée dans l'exemple.

- - **Group Name** : Basic Pages
- Enregistrez et publiez votre template

Vous devez maintenant créer une structure pour accéder à la portlet dans l'onglet **Structure** :

- Créer un Nouveau noeud avec les paramètres suivants:
  - **Nom du noeud** : ori-oai-search
  - **SiteNode type** : ComponentPage
- Enregistrez
- Choisissez **OUI** dans **Désactive le cache des pages** afin de signifier que vous avez une application dynamique
- Enregistrez
- Cliquez sur **affichage des compsants**
- Puis Click **here** to assign one or choose a page template below.
- Choisissez le composant **ori-oai-search-portlet**
- Revenez sur **couverture** pour **Enregistrer**
- Vous pouvez maintenant **Prévisualiser le noeud** et tester l'application

Ce paragraphe n'a montré qu'une configuration d'exemple d'intégration du module pour du test uniquement. Il est nécessaire d'adapter ces configurations à votre contexte d'utilisation dans infoglue.

Premiers tests de l'application

Une fois que vous aurez déployé l'application avec succès à partir de cette configuration, vous devrez vous brancher progressivement aux autres modules en éditant le fichier **webapp/WEB-INF/search-servlet.xml** ou **webapp/WEB-INF/search-portlet.xml** que vous soyez en déploiement *servlet* ou *portlet*.

La première étape consiste donc à éditer le bean **vocabulary-service** en commentant la classe de test et en dé-commentant la classe d'implémentation. Vous pouvez alors déployer à nouveau l'application et tester la connexion au module ORI-OAI-vocabulary. A cette étape, vous pouvez naviguer dans les recherches thématiques depuis l'interface de recherche et avoir des valeurs réélles dans les menus déroulants des recherches avancées.

La seconde étape permet de vous connecter au module ORI-OAI-indexing en éditant le bean **search-service** et en faisant une inversion de commentaires. Vous pouvez alors déployer l'application et tester des recherches.

Lorsque toutes ces étapes ont été validées, vous travaillez avec une version basique pré-configurée du module ORI-OAI-search.
Vous pouvez tester cette version en vous reportant à la documentation suivante.

Vous pouvez alors personnaliser les configurations et interfaces de recherche comme il est décrit dans la suite du document.

Personnalisation de l'application

Dans ce chapitre, nous verrons comment paramétrer l'application afin en fonction des besoins. Toutes les configurations se trouvent dans le fichier **properties/config.xml**.

Quelques paramètres généraux

Voici quelques paramètres globaux à l'application:
default_docs_per_page

> Ce paramètre est un entier qui indique le nombre de documents par défaut affichés par page lorsqu'un utilisateur lance une recherche.

max_docs_per_page

> Lorsqu'une page de résultats est affichée à l'utilisateur, il a la possibilités de changer le nombre de documents sur une page. Ce paramètre est donc un entier qui donne la limite que l'utilisateur peut sélectionner. Si le paramètre vaut par exemple 30, l'utilisateur aura la possibilité de choisir un nombre de documents entre 5 et 30 avec un intervalle de 5 entre chaque valeur: 5, 10, 15, 20, 25 et 30.

index_available

> Ce paramètre vaut **true** ou **false** et indique si une page d'accueil est disponible dans le menu de l'application. Si c'est le cas, cette page doit être éditée dans le fichier **webapp/WEB-INF/stylesheets/index/index_XX.jsp** où XX correspond au code de la langue courante dans l'application. Autrement dit, vous devez éditer une page par langue disponible dans l'application.

defaut_page

> Il est possible de configurer la page affichée par défaut dans l'application. Cela peut être la page d'accueil ou un menu de recherche:
>
> - **<defaut_page index="true"/>** si vous voulez que la page par défaut soir l'index (à condition que **index_available** soit placé à true)
> - **<defaut_page menu_key="mon_menu" search_key="mon_sous_menu"/>** où mon_menu correspond à la clef d'un menu de recherche **search_menu** et mon_sous_menu est la clef d'un formulaire de recherche **date_search**,

thematic_search ou **advanced_search** comme il sera vu dans la Section "Configurer les interfaces de recherche".

advanced_min_fields_two_columns

Dans une recherche avancée, les champs de recherche sont placés les uns à la suite des autres. Ce paramètre est donc un entier qui indique le nombre de champs de saisie à partir duquel on affiche les champs sur 2 colonnes. Par exemple, si le paramètre vaut 10 mais qu'il n'y a que 6 champs à afficher, ils seront sur une colonne. Si on passe alors ce paramètre à 4, comme 10 > 4, on affiche sur 2 colonnes: 5 et 5.

thematic_min_two_columns

Ce paramètre a le même principe que le précédent, mais correspond à l'affichage des catégories dans la recherche thématique.

locales

Il est possible de configurer plusieurs langues dans l'application. Des liens avec un drapeau pour chaque langue est affiché dans l'interface et on peut basculer l'application d'une langue à une autre en prenant garde que les bundles de messages aient été renseignés dans toutes les langues comme vu à la Section "Personnalisation des messages et libellés". Dans l'exemple suivant, on rend l'application disponible en anglais, français et espagnol où le français est la langue par défaut:

```
<locales default="fr">
        <locale>en</locale>
        <locale>fr</locale>
        <locale>es</locale>
</locales>
```

Si vous ne voulez par exemple ne gérer que la langue française, utilisez cette configuration:

```
<locales default="fr">
        <locale>fr</locale>
</locales>
```

Important

La possibilité de passage de l'interface d'une langue à une autre n'est à ce jour disponible qu'en version *servlet*. En mode *portlet*, utilisez donc uniquement la gestion d'une seule langue comme dans l'exemple précédent.

Notez qu'il ne faut surtout pas modifier les valeurs de métadonnées dans **static_metadatas**. Ces valeurs servent à identifier des métadonnées indexées par le module ori-oai-indexing en plus de celles contenues dans les fiches.

Vous pouvez cependant utilisez ces valeurs dans la configuration des interfaces de recherche. Elles sont à manipuler comme un nom de métadonnée classique pour la recherche et l'affichage des résultats.

doc_id

La valeur de ce champ est le nom de la métadonnée contenant l'identifiant du document.

repository

La valeur de ce champ est le nom de la métadonnée contenant l'entrepôt OAI du document. Si c'est un document local non moissonné en OAI, la valeur de la métadonnée est la valeur du champ workflow_name.

format

La valeur de ce champ est le namespace du format de métadonnées.

datestamp

La valeur de ce champ est le champ datestamp OAI du document si il vient de moissonnage OAI.

score

La valeur de ce champ est le nom de la métadonnée contenant le score lors des résultats de recherche.

workflow_name

Ceci est la valeur de la métadonnée repository lorsque l'on traite un document local.

Configurer les interfaces de recherche

Toutes les interfaces de recherche sont configurées dans la balise **menu**. Cette balise est composée de champs **search_menu** comme suit:

```
<menu>

        <search_menu key="menu_1">
                ...
        </search_menu>

        <search_menu key="menu_2">
                ...
        </search_menu>

        <search_menu key="menu_3">
                ...
        </search_menu>

</menu>
```

Chaque **search_menu** est donc un menu de recherche accessible depuis le menu de l'application. Chaque menu est alors découpé en différentes interfaces de recherche que l'on peut considérer comme des sous-menus. Ces sous-menus sont des recherches par date, thématiques et avancées identifiés par les balises **date_search** (Section "Recherche par date"), **thematic_search** (Section "Recherche thématique") et **advanced_search** (Section "Recherche avancée").

Un menu est donc composé par plusieurs sous-menus et par des balises **authorization** (Section "Autorisation d'accès") qui permettent de spécfier quel type d'utilisateur peut accéder à ce menu en mode *portlet*.

Notons que chaque menu doit avoir une clef unique pour le champ key.

Voici une illustration des possibilités de configuration:

```
<menu>

        <search_menu key="menu_1">
                <date_search/>
                <thematic_search/>
                <advanced_search/>
                <authorization/>
```

```
        </search_menu>

        <search_menu key="menu_2">
                <date_search/>
                <thematic_search/>
                <advanced_search/>
                <authorization/>
        </search_menu>

</menu>
```

Recherche par date

Ce type de recherche permet de faire de la recherche de documents uniquement en fonction de la durée entre la date courante et un champ date spécifié dans la fiche de métadonnées ou le datestamp OAI. Cette recherche se configure comme ceci:

```
<date_search key="date_test" hide="false" start_days_period="30" max_days_period="365"
 days_interval="30">

        <metadatas>
                <metadata dateFormat="yyyyMMdd">md-ori-oai-datestamp</metadata>
        </metadatas>

        <hidden_fields>
                ...
        </hidden_fields>
        <result_fields>
                ...
        </result_fields>
        <sort_fields>
                ...
        </sort_fields>
        <rss_fields>
                ...
        </rss_fields>
        <authorization>
                ...
        </authorization>

</date_search>
```

key

Contient la clef unique de ce sous-menu de recherche.

hide

Vaut **true** ou **false** suivant que l'on veuille cacher ou non ce sous-menu de recherche. Cacher un sous-menu peut être utile lorsque l'on y accède en tant que client et que l'on ne veut pas le voir dans le menu de l'interface.

start_days_period

La recherche se faisant en sélectionnant une durée de jours d'ancienneté dans une liste déroulante, cette valeur est un entier qui contient la première durée affichée par défaut. Si on configure 30, on aura par défaut en se présentant sur cette recherche tous les documents dont la date est inférieure à 30 jours.

max_days_period

Ceci est un entier qui indique la dernière valeur possible dans la liste déroulante des périodes;

days_interval

Cet entier est l'intervalle, en jours, entre chaque période de la liste.

metadatas

Cette balise contient un ensemble de sous-balises **metadata**. Chacune de ces balises contient une métadonnée sur laquelle on veut faire une recherche.

hidden_fields

Permet d'ajouter des champs de recherche cachés lors de la recherche. Voir la Section "Champs de recherche cachés" pour configurer cette partie.

result_fields

Permet de configurer les champs que l'on veut afficher dans la page de résultats. Voir la Section "Champs de résultats à afficher" pour configurer cette partie.

sort_fields

Permet de configurer l'ordre de tri des résultats. Voir la Section "Tri des résultats" pour configurer cette partie.

rss_fields

Permet de configurer les champs à utiliser lors de la génération du flux RSS en mode *servlet*. Voir la Section "Paramétrage des flux RSS" pour configurer cette partie.

authorization

Permet de configurer les autorisations d'accès en mode portlet. Voir la Section "Autorisation d'accès" pour configurer cette partie.

Recherche thématique

Ce type de recherche permet de naviguer dans différentes catégories provenant de vocabulaires du module ORI-OAI-vocabulary (thématiques de documents, auteurs, mots-clefs, etc.). Cette recherche se configure comme ceci:

```
<thematic_search key="thematic_test" hide="false" vocabulary_id="search_unit_taxonomie_regexp"
 keep_navigation_session="true" full_tree="false" >

        <metadatas>
                <metadata>//lom:classification/lom:taxonPath[lom:source/lom:string='dewey']/
lom:taxon/lom:id</metadata>
                <metadata vcard_att="FN">//lom:lifeCycle/lom:contribute[lom:role/
lom:value='author']/lom:entity(name)</metadata>
        </metadatas>

        <hidden_fields>
                ...
        </hidden_fields>
        <result_fields>
                ...
        </result_fields>
        <sort_fields>
                ...
        </sort_fields>
        <rss_fields>
                ...
        </rss_fields>
        <authorization>
                ...
        </authorization>
```

```
</thematic_search>
```

key

Contient la clef unique de ce sous-menu de recherche.

hide

Vaut **true** ou **false** suivant que l'on veuille cacher ou non ce sous-menu de recherche. Cacher un sous-menu peut être utile lorsque l'on y accède en tant que client et que l'on ne veut pas le voir dans le menu de l'interface.

vocabulary_id

Contient l'identifiant du vocabulaire (provenant du module ORI-OAI-vocabulary) dans lequel on veut naviguer.

value

Lorsque l'on ne veut pas passer par un vocabulaire avec l'attribut **vocabulary_id**, on peut utiliser une valeur simple. Par exemple, on peut saisir **value="V1"** et dans ce cas, à chaque clic on verra directement la recherche sur la valeur unique **V1**. Ceci permet de proposer une liste de résultats sur n'importe quel paramètre.
En mode *portlet*, il est aussi possible de faire correspondre cette valeur à une valeur d'un attribut LDAP de l'utilisateur connecté. En effet, en utilisant la syntaxe **value={mon_attribut_ldap}**, la valeur sera remplacée par la valeur de l'attribut **mon_attribut_ldap** de l'utilisateur. Exemple d'un menu de recherche qui présente à l'utilisateur connecté toutes les ressources pédagogiques dont il est auteur:

```
<thematic_search key="my_author" value="{displayName}">
        <metadatas>
                <metadata>//lom:lifeCycle/lom:contribute[lom:role/lom:value='author']/
lom:entity(name)</metadata>
        </metadatas>
        ....
</thematic_search>
```

keep_navigation_session

Ce paramètre vaut **true** ou **false**. Il permet de dire si on souhaite garder en session la navigation de l'utilisateur dans une recherche thématique. Si true, à chaque fois qu'un utilisateur reviendra dans la recherche thématique durant sa session, il se trouvera dans le dernier niveau de l'arbre visité. Dans le cas false, il reviendra toujours à la racine de l'arbre.

full_tree

Ce paramètre vaut **true** ou **false**. Dans le cas false, nous sommes dans une recherche thématique classique: la recherche se fait par navigation successives dans l'arborescence de la thématique. Si on configure true, toute l'arborescence nous est présentée sous forme d'un arbre dépliable. On peut alors sélectionner tous les niveaux souhaités en une seule requête.

metadatas

Cette balise contient un ensemble de sous-balises **metadata**. Chacune de ces balises contient une métadonnée sur laquelle on veut faire une recherche. Imaginons que l'on ait 2 métadonnées MD1 et MD2. On navigue dans la catégorie qui a pour valeurs V1 et V2. La requête générée sera alors du type: MD1=V1 ou MD1=V2 ou MD2=V1 ou MD2=V2.

	vcard_att

	Lorsque l'on veut naviguer dans un vocabulaire dont les métadonnées font référence à des vcards. Il est possible de spécifier sur quel champ de la vcard on veut faire la recherche. Si on veut chercher sur le champ FN de la vcard, il faut donc spécifier **vcard_att="FN"**.

hidden_fields

Permet d'ajouter des champs de recherche cachés lors de la recherche. Voir la Section "Champs de recherche cachés" pour configurer cette partie.

result_fields

Permet de configurer les champs que l'on veut afficher dans la page de résultats. Voir la Section "Champs de résultats à afficher" pour configurer cette partie.

sort_fields

Permet de configurer l'ordre de tri des résultats. Voir la Section "Tri des résultats" pour configurer cette partie.

rss_fields

Permet de configurer les champs à utiliser lors de la génération du flux RSS en mode *servlet*. Voir la Section "Paramétrage du flux RSS" pour configurer cette partie.

authorization

Permet de configurer les autorisations d'accès en mode portlet. Voir la Section "Autorisation d'accès" pour configurer cette partie.

Recherche avancée

Ce type de recherche permet de rechercher des documents depuis un formulaire de recherche simple ou avancée. Cette recherche se configure comme ceci:

```
<advanced_search key="advanced_test" hide="false" file="test.xml">

        <hidden_fields>
                ...
        </hidden_fields>
        <result_fields>
                ...
        </result_fields>
        <sort_fields>
                ...
        </sort_fields>
        <rss_fields>
                ...
        </rss_fields>
        <authorization>
                ...
        </authorization>

</advanced_search>
```

key

Contient la clef unique de ce sous-menu de recherche.

hide

Vaut **true** ou **false** suivant que l'on veuille cacher ou non ce sous-menu de recherche. Cacher un sous-menu peut être utile lorsque l'on y accède en tant que client et que l'on ne veut pas le voir dans le menu de l'interface.

file

Contient le nom du fichier XML contenant la définition du formulaire de recherche avancée. Ce fichier doit se trouver dans le dossier **properties/advanced**. Il se configure comme décrit à la Section "Configurer un formulaire de recherche avancée".

hidden_fields

Permet d'ajouter des champs de recherche cachés lors de la recherche. Voir la Section "Champs de recherche cachés" pour configurer cette partie.

result_fields

Permet de configurer les champs que l'on veut afficher dans la page de résultats. Voir la Section "Champs de résultats à afficher" pour configurer cette partie.

sort_fields

Permet de configurer l'ordre de tri des résultats. Voir la Section "Tri des résultats" pour configurer cette partie.

rss_fields

Permet de configurer les champs à utiliser lors de la génération du flux RSS en mode *servlet*. Voir la Section "Paramétrage du flux RSS" pour configurer cette partie.

authorization

Permet de configurer les autorisations d'accès en mode portlet. Voir la Section "Autorisation d'accès" pour configurer cette partie.

Configurer un formulaire de recherche avancée

Un formulaire de recherche avancée se configure comme dans l'exemple ci-dessous:

```
<form>

        <group id="group_id1">
                <field id="field_id1" format="text" defaultValue="value_1" readOnly="true"
 hidden="false">
                        <metadata>...</metadata>
                        <metadata>...</metadata>
                </field>
                <field id="field_id2" format="text" vocabularyId="vocabulary_1">
                        <metadata>...</metadata>
                </field>
                <field id="field_id3" format="text" thematicMenuKey="menu_test"
 thematicSearchKey="thematic_test" thematicFullTree="true">
                        <metadata>...</metadata>
                </field>
                <field id="field_id4" format="date:dd-MM-yyyy">
                        <metadata dateFormat="yyyyMMdd">...</metadata>
                </field>
        </group>

        <group id="group_id2">
                ...
        </group>
```

```
</form>
```

La configuration détaillée de ce fichier est la suivante:

group

Les formulaires de recherche avancée sont découpées en différents groupes de champs de recherche. Ces groupes sont identifiés par la balise **group** et les champs par les balises **field**. Dans l'interface, les groupes seront séparés les uns des autres, cela permet de ranger les champs de recherche dans différentes catégories.

id

Chaque groupe doit avoir ici un identifiant unique. N'utilisez pas de caractères spéciaux.

Aussi, un groupe est composé de différents champs de recherche (**field**).

field

Un **field** correspond à un champ de recherche du formulaire. Chaque champ a plusieurs possibilités de configurations suivants les attributs suivants:

id

Chaque champ doit avoir un identifiant unique dans tout le formulaire. N'utilisez pas de caractères spéciaux.

format

Ce champ permet de spécifier de quel format est la valeur que l'on veut saisir. Pour tous les formats autres que du texte, une validation est faite au moment de la saisie afin de valider ou non le formulaire.
Les valeurs possibles sont **text** pour du texte simple, **int** pour un entier, **float** pour un flottant, **boolean** pour un booléen et **date** pour une date. Dans le cas d'une date, il est nécessaire de spécifier le format de saisie de la date dans le formulaire à l'aide de dd pour le jour, MM pour le mois et yyyy pour l'année. Par exemple, pour dire que le 22 juin 2005 doit être saisi comme 22-06-2005, il faudra paramétrer le formulaire comme ceci: **format="date:dd-MM-yyyy"**.

vocabularyId

Ce champ permet d'utiliser des listes déroulantes dans les champs de saisie. Il suffit de spécifier un identifiant de vocabulaire (provenant du module ORI-OAI-vocabulary). L'interface proposée sera alors une liste déroulante composée à partir du premier niveau de valeurs dans l'arbre des vocabulaires.

defaultValue

Il est possible de spécififer ici une valeur par défaut au formulaire.
Dans le cas d'une liste déroulante basée sur un vocabulaire, il faut spécifier l'identifiant de la catégorie souhaitée dans le vocabulaire.

hidden

Cette valeur vaut **true** ou **false** suivant que l'on veut que ce champ soit visible ou non. Ceci est utile dans le cas où on veut cacher une valeur dans la saisie. On pourra utiliser le champ **defaultValue** pour mettre une valeur par défaut, et le champ **hidden** pour cacher cette valeur. On peut comme ceci ajouter des paramètres à la requête sans que l'utilisateur ne le voit.

readOnly

Dans le cas où on veut mettre une valeur par défaut dans un formulaire sans la cacher à l'utilisateur, mais sans que celui-ci ne puisse la modifier, on peut mettre se paramètre à la valeur **true**.

thematicMenuKey

Cette variable doit OBLIGATOIREMENT être couplée à thematicSearchKey. Les 2 variables associées permettent d'aller sélectionner des valeurs pour ce champ depuis une navigation dans une recherche thématique en mode "esclave". Cet attribut doit donc contenir la clef du menu de recherche (défini dans **config.xml**) contenant la recherche thématique souhaitée.

thematicSearchKey

Cette variable doit donc contenir la clef du sous-menu de recherche thématique souhaité.

thematicFullTree

Lors de l'utilisation des paramètres **thematicMenuKey** et **thematicSearchKey**, on souhaite saisir la valeur du champ par recherche thématique. Le paramètre **thematicFullTree** permet donc de dire si cette recherche thématique se fait sous forme d'arbre dépliable ou non.

metadata

Chaque field peut doit contenir une ou plusieures balise **metadata**. Chaque metadata permet de définir une métadonnée sur laquelle on va faire la recherche. On peut spécifier plusieurs métadonnées, dans ce cas, on lancera la recherche sur chacune de ces métadonnées.

dateFormat

Avec ce paramètre, on permet de dissocier le format de saisie dans le formulaire du format de date dans la requête. Ce paramètre n'est à utiliser que si l'on a spécifier qu'on est sur un champ de type date. On pourra alors spécifier que l'on saisit une date comme 22-06-2005 mais qu'elle est transformée en 20050622 pour la requête en mettant pour valeur **dateFormat="yyyyMMdd"**.

authorization

Dans les balises **group** et **field**, il est possible d'utiliser la balise authorization (cf. Section "Autorisation d'accès") afin de filtrer l'accès au groupe ou au champ de recherche. Exemple:

```
<form>

        <group id="group_id1">
                <field id="field_id1" format="text" defaultValue="value_1" readOnly="true"
 hidden="false">
                        <metadata>...</metadata>
                        <metadata>...</metadata>
                </field>
                <field id="field_id2" format="text" vocabularyId="vocabulary_1">
                        <metadata>...</metadata>
                        <authorization>
                                ...
                        </authorization>
                </field>
        </group>

        <group id="group_id2">
                ...
                <authorization>
                        ...
                </authorization>
        </group>

</form>
```

Champs de recherche cachés

Il est possible d'ajouter des valeurs cachées aux formulaires de recherche par date, thématique ou avancée. Ceci ajoutera des paramètres à la requête qui est envoyée au module ORI-OAI-indexing. La syntaxe est la suivante:

```
<hidden_fields>
<!-- On veut que le namespace soit DC ou dublin_core ou oai_dc -->
        <hidden_field>
                <metadata>md-ori-oai-namespace</metadata>
                <value>DC</value>
                <value>"dublin core"</value>
                <value>oai_dc</value>
        </hidden_field>

<!-- On ne veut pas que le namespace soit du CDM -->
        <hidden_field vocabularyId="search_metadata_ns_formats:cdm_id" not="true">
                <metadata>md-ori-oai-namespace</metadata>
        </hidden_field>

<!-- On veut que l'auteur soit abaddon -->
        <hidden_field vocabularyId="personnes:abaddon">
                <metadata vcard_att="FN">//lom:lifeCycle/lom:contribute[lom:role/
lom:value='author']/lom:entity(name)</metadata>
        </hidden_field>
</hidden_fields>
```

Il ne doit y avoir qu'un seul **hidden_fields** qui lui est composé de plusieurs **hidden_field**. La requête générée par l'utilisateur sera alors modifiée par l'ajout des critères de chaque **hidden_field**.

Un hidden_field correspond à un champ caché. On y spécifie toutes les métadonnées concernées et les valeurs associées. Il peut y avoir plusieurs métadonnées dans les champs **metadata**. Il existe 2 méthodes pour définir les valeurs qui doivent être associées aux métadonnées:
value

> On insère une série de balises **value** dans un **hidden_field**. On construira alors la requête où chaque métadonnée doit avoir au moins une valeur dans cette liste.

vocabularyId

> On spécifie cet attribut dans **hidden_field** (ne pas mettre dans ce cas de balises **value**) pour dire que les valeurs que doivent avoir les métadonnées spécifiées se trouve dans un vocabulaire du module ORI-OAI-vocabulary. La syntaxe est alors **vocabularyId="identifiant_vocabulaire:identifiant_categorie"** où *identifiant_vocabulaire* est l'identifiant du vocabulaire distant et *identifiant_categorie* est l'identifiant de la catégorie dans le vocabulaire. Les valeurs requises seront donc les valeurs associées à la catégorie dans le vocabulaire.

not

> Permet de dire si on veut faire une négation sur ce champ caché. Si la valeur est **not="true"**, il y aura alors négation sur la requête. Par exemple on ne veut pas que les résultats qui apparaissent soient de format CDM.

Dans le cas où la valeur données est une vcard, il faut spécifier l'attribut de la vcard sur lequel on souhaite faire la requête dans l'attribut **vcard_att** de **metadata**.

En mode *portlet*, il est aussi possible de faire correspondre une valeur de **value** à une valeur d'un attribut LDAP de l'utilisateur connecté. En effet, en utilisant la syntaxe **value={mon_attribut_ldap}**, la valeur sera remplacée par la valeur de l'attribut **mon_attribut_ldap** de l'utilisateur. Exemple d'un champ caché qui force l'interface de recherche à ne prendre que les ressources pédagogiques dont il est auteur:

```
<hidden_field>
        <metadata>//lom:lifeCycle/lom:contribute[lom:role/lom:value='author']/lom:entity(name)</
metadata>
        <value>{displayName}</value>
</hidden_field>
```

Champs de résultats à afficher

Dans cette partie, on indique les champs de résultats que l'on veut afficher. On va donc passer par une balise **result_fields** composée de plusieurs champs **result_field**, chaque result_field étant un champ de résultat. Voici un exemple:

```
<result_fields jsp_file="my_results.jsp">
        <result_field key="title" title_field="true" show_notice_link="true" href_document="true">
                <metadata>//lom:general/lom:title/lom:string[starts-with(@language,'fr')]</
metadata>
        </result_field>

        <result_field key="date" format="date:dd-MM-yyyy">
                <metadata dateFormat="yyyyMMdd">//lom:lifeCycle/lom:contribute[lom:role/
lom:value='author']/lom:date/lom:dateTime</metadata>
        </result_field>

        <result_field key="language" format="vocabulary:search_languages">
                <metadata>//lom:general/lom:language</metadata>
        </result_field>
</result_fields>
```

Il existe un attribut qui permet de spécifier la page JSP qui sert à la mise en page des résultats pour l'interface de recherche ici présente:
jsp_file

> Nom du fichier JSP à utiliser. Il se trouve dans le dossier **webapp/WEB-INF/stylesheets/results**. On prendra par défaut si aucune valeur n'est saisie le fichier **generic-results.jsp** pour l'affichage en tableaux.
>
> Il existe cependant plusieurs configurations d'affichage proposées au format jsp. Ces jsp ne sont compatibles qu'avec certaines configurations de formulaires de recherche En effet, les jsp proposées se reposent sur des champs de résultat (**result_field**) configurés dans les fichiers **config.example.???.xml**. Si vous voulez utiliser une de ces jsp dans votre contexte, il faut impérativement que tous les champs affichés par celle-ci soient référencés comme **result_field** dans votre configuration.
>
> - **documents-results.jsp** qui affiche les résultats sous forme allégée comme dans beaucoup de moteurs de recherche classiques. Cette configuration est compatible avec les champs de résultats proposés dans les recherches liées aux ressources en Dublin Core et en LOM. La description et les mots-clefs sont affichés par survol avec la souris.
> - **documents-results-unt.jsp** ressemble beaucoup à la précédente mais une partie de la description est affichée directement sous le titre.
> - **documents-results-cdm.jsp** peut être utilisée pour afficher des formations dans le cas d'une recherche sur des fiches CDM.

Les balises **result_field** sont composées comme ceci:
key

> Une clef unique pour ce champ de résultat dans cette interface de recherche.

show_notice_link

Si vaut **true**, un lien vers l'affichage de la fiche de métadonnée est affiché dans ce champ. La définition des différents formats de métadonnées est décrite dans la Section "Les formats de métadonnées supportés".

href_document

Si vaut **true**, un lien vers le document est disponible. La définition des métadonnées relatives à ces liens est décrite dans la Section "Lien vers les documents dans la liste des résultats de recherche".

format

Cet attribut permet de spécifier une transformation de la valeur de métadonnées retrouvée. En effet, il est possible d'effectuer une transformation de la valeur lors de l'affichage.
Les valeurs possibles pour ce champ peuvent être:

- **time** pour l'affichage d'une durée de type P1Y2M3DT4H5M6S
- *size pour une taille de fichier en octets
- **date pour une date. Dans ce cas, la date sera affichée dans le format désiré. La saisie de ce paramètre ce fait sous cette forme: *format="date:dd-MM-yyyy"**
- **vocabulary pour un vocabulaire. Dans ce cas, la valeur retrouvée est cherchée en correspondance dans les valeurs du vocabulaire choisi. Lorsque cette valeur est trouvée, on affiche le libellé correspondant dans la langue sélectionnée dans l'interface. Par exemple, on spécifiera comme ceci le vocabulaire des langues: *format="vocabulary:search_languages"**. Ceci peut être utilisée dans le cas de l'affichage de la langue de saisie d'une ressource pédagogique. Si la valeur *fr-FR* est retrouvée, elle sera automatiquement traduite en *Français* si on affiche en français, ou *French* si on affiche en anglais, etc.

title_field

Si vaut **true**, ce champ est considéré comme de type titre. Dans ce cas, la classe CSS utilisée pour l'affichage n'est pas la même que pour les autres champs.

Chaque **result_field** est également composé de sous-balises **metadata** décrites comme ceci:
dateFormat

Avec ce paramètre, on permet de dissocier le format d'affichage du résultat du format de date dans la métadonnée. Ce paramètre n'est à utiliser que si l'on a spécifier qu'on est sur un champ de type date. On pourra alors spécifier que l'on veut afficher une date comme 22-06-2005 mais qu'elle est stockée en 20050622 dans la métadonnée en mettant pour valeur **dateFormat="yyyyMMdd"**.

vcard_att

Lorsque l'on utilise un format vocabulaire qui fournit des vcards, il est nécessaire d'indiquer quel attribut on veut récupérer dans la vcard à afficher. C'est dans ce paramètre qu'on l'indique. Exemple, on mettra **vcard_att="FN"** si on veut récupérer le champ **FN** de la vcard.

Tri des résultats

Dans la configuration, vous pouvez spécifier dans quel ordre seront triés les resultats de la recherche. Si vous ne le faîtes pas, les documents seront triés par rapport au score géré par le module ORI-OAI-indexing.

```
<sort_fields ascending="true">
        <sort_field_key>md-ori-oai-score</sort_field_key>
        <sort_field_key>title</sort_field_key>
        <sort_field_key>date</sort_field_key>
        <sort_field_key>repository</sort_field_key>
</sort_fields>
```

Dans l'exemple de configuration précédent, on voit le tri de documents suivant plusieurs **sort_field_key**. Chacune de ces valeurs correspond à la clef donnée pour les champs de résultat dans les balises **result_fields/result_field**, sauf pour *md-ori-oai-score* où là on spécifie que l'on ne trie pas par rapport à un champ, mais par rapport au score (i.e. à la pertinence de résultat; cette pertinence est à configurer au niveau du module ORI-OAI-indexing dans la définition des setBoost Lius). Les champs affichés seront donc triés en fonction des champs de résultats définis plus haut et au score.

Chaque champ de tri est interprété dans l'ordre où il est configuré. Dans l'exemple précédent, en imaginant plusieurs documents ayant le même score, le tri entre-eux se fera ensuite sur le champ *title*, puis par rapport au champ *date* et enfin *repository*.

ascending

> Cet attribut peut valoir **true** ou **false** que l'on veuille trier dans l'ordre croissant ou décroissant. Pour le titre, on choisira true, tandis que pour trier des documents suivant la date la plus récente à la plus ancienne, on choisira false.

Notons que tous les champs mentionnés dans la balise **sort_fields** pourront être re-triés différemment par l'utilisateur. En effet, celui-ci aura la possibilité de cliquer sur le nom du champ pour trier suivant un autre champ ou dans l'ordre inversé.

Paramétrage du flux RSS

En mode *servlet*, il existe la possibilité de générer des flux RSS dynamiques sur toutes les requêtes formulées par l'utilisateur. Pour cela, une icône est disponible sur chaque page de résultat ainsi que dans la barre d'adresse de certains navigateurs. La configuration se fait dans les blocs de chaque menu de recherche:

```
<rss_fields link="notice">
        <title>title_md</title>
        <description>description_md</description>
        <pubDate>date_md</pubDate>
</rss_fields>
```

Les différents champs de ce bloc se configurent de la manière suivante:
link

> Ce champ permet de dire si on souhaite pointer vers la fiche de métadonnée ou vers le document lui-même dans le flux RSS. Pour pointer vers la fiche de métadonnée, la valeur de ce champ doit être **notice**. Lorsque l'on souhaite pointer directement sur le document, on n'utilise pas le paramètre **link**.

title

> Ce champ permet de dire quelle métadonnée va servir à remplir le titre dans les tags RSS. La valeur ici correspond à la valeur de l'attribut **key** du **result_field** désiré.

description

> Ce champ permet de dire quelle métadonnée va servir à remplir la description dans les tags RSS. La valeur ici correspond à la valeur de l'attribut **key** du **result_field** désiré.

pubDate

> Ce champ permet de dire quelle métadonnée va servir à remplir la date de publication du flux dans les tags RSS. La valeur ici correspond à la valeur de l'attribut **key** du **result_field** désiré.

Autorisation d'accès

En mode *portlet*, il est possible de spécifier quels utilisateurs pourront visualiser une partie de l'interface de recherche. Ce filtrage est basé sur les attributs de l'utilisateur. Ce filtrage se base sur la syntaxe

suivante. Dans ce cas, seuls les utilisateurs ayant pour uid ycolmant ou bourges auront accès et verront l'interface visée.

```
<authorization operator="or">
        <allowed attribute="uid" value="ycolmant"/>
        <allowed attribute="uid" value="bourges"/>
</authorization>
```

Il est possible de configurer plusieurs balises authorization à la suite. Dans ce cas, l'utilisateur aura accès si il répond à au moins un des 2 filtres.

operator

Cet attribut de la balise authorization a la valeur **or** ou **and** suivant que l'on utilise l'opération "ou" ou "et". Cette opération se fait sur l'ensemble des balises **allowed** contenues dans **authorization**.

allowed

Cette balise permet de spécifier une partie du filtre sur un attribut. L'utilisateur répond bien à ce filtre si son attribut contenu dans **attribute** a bien la valeur contenue dans **value**. Si l'opérateur **or** est utilisé, il faut que l'utilisateur réponde au minimum à un **allowed**, si c'est un **and**, il faut qu'il réponse à tous les **allowed**.
Il est possible de faire ce filtrage sur un nom de groupe du portail. En effet, en utilisant pour attribut la valeur **attribute="portal_group"**, le filtrage sera fait sur le groupe de l'utilisateur. Par exemple, **<allowed attribute="portal_group" value="pags.mon_groupe"/>** fera un fitrage pour tous les utilisateurs qui font partie du groupe **pags.mon_groupe**.

Lien vers les documents dans la liste des résultats de recherche

Les différents formats de métadonnées sont identifiés par le namespace XML du document. Pour chaque format, donc pour chaque namespace, il est nécessaire d'indiquer quelle est la métadonnée qui contient les URL des documents référencés par la fiche de métadonnées. C'est le but de la balise **href_formats**. Celle-ci est composée de plusieurs sous-balises **format** pour chaque format de métadonnée. Voici un exemple où on définit les métadonnées pour le LOM et le Dublin Core:

```
<href_formats formatsVocabulary="search_metadata_ns_formats">
        <format hrefFormatValue="http://ltsc.ieee.org/xsd/LOM">
                <metadata>//lom:technical/lom:location</metadata>
        </format>

        <format hrefFormatValue="vocabulary:search_metadata_ns_formats:dc_id">
                <metadata>//dc:identifier</metadata>
        </format>
</href_formats>
```

Voyons en détail les différents attributs de configuration:
formatsVocabulary

L'affichage des notices necessite la résolution d'un identifiant de format de métadonnées. Le paramètre **formatsVocabulary** est donc l'identifiant du vocabulaire qui contient tous les formats de métadonnées.

hrefFormatValue

Cet attribut permet d'identifier le namespace du format de métadonnées XML. La valeur peut donc être simplement ce namespace, ou il peut être déduit d'un vocabulaire. La syntaxe est alors **hrefFormatValue="vocabulary:identifiant_vocabulaire:identifiant_categorie"** où *identifiant_vocabulaire* est l'identifiant du vocabulaire distant et *identifiant_categorie* est l'identifiant de la catégorie dans le vocabulaire où se trouve le(s) namespace(s) souhaité(s).

Chaque balise **metadata** contient elle la métadonnée où se trouve le lien du document.

Les modes de recherche simple

En plus des interfaces de recherche décrites ci-dessus, il existe différentes possibilités d'effectuer des "recherches simples".

Champ de recherche simple accessible sur toutes les pages

Il est prévu dans l'interface de pouvoir lancer facilement une recherche depuis un champ accessible sur toutes les pages de l'interface. On appele ce mode *recherche simple*. Son fonctionnement est simple: ce champ n'est en réalité qu'un pointeur vers un champ de recherche d'un formulaire avancé configuré au préalable. En lançant une recherche depuis ce formulaire, vous lancez concrêtement une recherche depuis le champ de saisie pointé.

Pour configurer ce mode, vous devez changer les paramètres suivants:

```
<simple_search menu_key="menu_test" search_key="advanced_test" field_id="field_test"/>
```

menu_key

Cet attribut doit donc contenir la clef du menu de recherche (défini dans **config.xml**) contenant la recherche avancée souhaitée.

search_key

Cette variable doit donc contenir la clef du sous-menu de recherche avancée souhaité.

field_id

Ceci est l'identifiant du champ de recherche. Vous l'avez défini dans la configuration du formulaire avancé dans le dossier **properties/advanced**. Il correspond à l'attribut **id** d'une balise **field**. Dans le cas présent, il est souvent préférable que ce champ soit l'aggrégation de plusieurs métadonnées pour pouvoir lancer une recherche large (titre, description, auteur, etc.).

Notons que cette configuration génère un formulaire HTML dans toutes les interfaces. Vous pouvez visualiser ce formulaire en regardant le code source d'une des pages de l'application. Vous pouvez donc si vous le souhaitez intégrer ce formulaire dans une autre page web pour rebondir vers cette recherche (uniquement dans le cas d'un déploiement en mode *servlet*).

Voici un exemple de formulaire généré dans lequel il faut compléter les paramètres en caractère gars:

```
<form name="simple_search_form" method="get" action="http://[HOST_SEARCH]:[PORT_SEARCH]/
[CONTEXT_SEARCH]/simple-search.html">
        Recherche simple:<br/>
        <input type="hidden" name="menuKey" value="menu_test"/>
        <input type="hidden" name="submenuKey" value="advanced_test"/>
        <input type="hidden" name="fieldId" value="field_test"/>
        <input class="input-text" name="light-request" size="20" type="text"/>
        <input class="searchButton" value="OK" type="submit"/>
</form>
```

Recherche des nouveautés accessible sur toutes les pages

Tout comme le champ de recherche simple, il est possible d'avoir un lien sur toutes les pages vers la *recherche de nouveautés*. Ce lien n'est en fait qu'un pointeur vers le formulaire de recherche par date que vous aurez désigné.

Pour cela, vous devez renseigner les paramètres suivants:

```
<simple_date_search menu_key="menu_test" search_key="date_test"/>
```

menu_key

Cet attribut doit donc contenir la clef du menu de recherche (défini dans **config.xml**) contenant la recherche par date souhaitée.

search_key

Cette variable doit donc contenir la clef du sous-menu de recherche par date souhaité.

Plugin OpenSearch pour Firefox (*à partir de la version 2*) et Internet Explorer (*à partir de la version 7*)

OpenSearch est une collection de technologies permettant à des sites webs et des moteurs de recherche de publier des résultats de recherche dans un format standardisé (http://www.opensearch.org). Il permet l'intégration de plugins de recherche dans différents navigateurs comme Firefox 2 et Internet Explorer 7 à la manière des plugins Google.

Son fonctionnement est simple: on stocke sur le serveur des définitions de formulaires de recherche en XML au format OpenSearch. Ces fichiers sont référencés dans l'en-tête HEAD des pages HTML:

```
<link rel="search" type="application/opensearchdescription+xml" title="Recherche simple de mon
 établissement" href="...../opensearch/ma_recherche.xml"/>
```

Ceci permet alors au navigateur de détecter automatiquement qu'un plugin de recherche est disponible sur ce site et il est proposé à l'utilisateur. Il n'a plus qu'à l'intégrer à son navigateur. A partir de ce moment, l'utilisateur pourra lancer une recherche depuis son navigateur en rebondissant sur le moteur de recherche ORI-OAI.

La configuration est la suivante dans le fichier **properties/config.xml** :

```
<open_search>
        <link title="Recherche de ressources pédagogiques" file="lom.xml"/>
        <link title="Recherche de thèses" file="tef.xml"/>
</open_search>
```

Il est possible de mettre plusieurs balises **link** dans la configuration pour rendre disponibles plusieurs types de recherche.

title

Le titre affiché pour ce plugin dans le navigateur au moment de la sélection.

file

Le nom du fichier de configuration OpenSearch contenu dans le dossier **webapp/opensearch**.

Voyons maintenant la configuration d'un plugin OpenSearch compatible Firefox 2 et Internet Explorer 7 (Vous pouvez plus de documentation pour la création de ces plugins pour Firefox 2 notamment à cette adresse: http://www.gatellier.be/blog/plugin-recherche-opensearch-firefox2.):

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
        <OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/" xmlns:moz="http://
www.mozilla.org/2006/browser/search/">
        <ShortName>Recherche de ressources pédagogiques</ShortName>
        <Description>Recherche sur des ressources pédagogiques de l'Université de Test</
Description>
        <Contact>[SMTP_ADMINISTRATOR_MAIL]</Contact>
        <InputEncoding>iso-8859-1</InputEncoding>

        <Image width="16" height="16">......</
Image>
        <Url template="http://[HOST_SEARCH]:[PORT_SEARCH]/[CONTEXT_SEARCH]/simple-
search.html?menuKey=menu_test&amp;submenuKey=advanced_test&amp;fieldId=field_test&amp;light-
request={searchTerms}" type="text/html"/>

        <moz:SearchForm>http://[HOST_SEARCH]:[PORT_SEARCH]</moz:SearchForm>
</OpenSearchDescription>
```

ShortName

Le nom court du plugin qui est affiché dans le navigateur.

Description

La description du plugin.

Contact

L'adresse d'un contact.

Image

On renseigne ici le logo (.ico) qui est affiché dans le champ de recherche, il peut être de 2 types:

- On donne un lien vers une icône sur un serveur web:

```
<Image height="16" width="16" type="image/x-icon">http://example.com/favicon.ico</
Image>
```

- On converti l'icône en base64 et on l'ajoute dans le fichier de configuration:

```
<Image width="16" height="16">......</Image>
```

Un exemple d'outil en ligne pour la conversion d'images: http://www.motobit.com/util/base64-decoder-encoder.asp

Url

On définit ici l'URL de recherche. Cette URL est de ce type:

```
http://[HOST_SEARCH]:[PORT_SEARCH]/[CONTEXT_SEARCH]/simple-search.html?
menuKey=menu_test&amp;submenuKey=advanced_test&amp;fieldId=field_test&amp;light-
request={searchTerms}
```

	Les paramètres à remplacer signifient: **[HOST_SEARCH]:[PORT_SEARCH]/ [CONTEXT_SEARCH]** L'adresse, le port et le contexte de déploiement de la servlet de recherche ORI-OAI. **menu_test** Cet attribut est contenir la clef du menu de recherche (défini dans **config.xml**) contenant la recherche avancée souhaitée. **advanced_test** Cette variable est la clef du sous-menu de recherche avancée souhaité. **field_test** Ceci est l'identifiant du champ de recherche. Vous l'avez défini dans la configuration du formulaire avancé dans le dossier **properties/advanced**. Il correspond à l'attribut **id** d'une balise **field**. Dans le cas présent, il est souvent préférable que ce champ soit l'aggrégation de plusieurs métadonnées pour pouvoir lancer une recherche large (titre, description, auteur, etc.).

moz:SearchForm

Un lien vers le site institutionnel de votre établissement ou tout pointeur que vous voulez voir apparaître depuis Firefox 2.

Les formats de métadonnées supportés

Dans cette section, nous allons voir comment définir tous les formats de métadonnées supportés dans l'application en vue d'afficher la fiche de métadonnées.

Il ne faut définir ici que tous les formats dont on veut afficher la fiche de métadonnées. Pour un format où on n'affiche que l'ensemble des résultats de requête sans donner un lien vers l'affichage complet de la fiche, il n'est pas nécessaire de configurer cette partie.

Le bloc de configuration de cette partie se présente sous cette forme:

```
<notice_formats>
        <format formatMetadataValue="vocabulary:search_metadata_ns_formats:lom_id" prefix="lom"
 xsl="lom1.xsl" headXsl="lom.xsl">
                <metadata format="...">...</metadata>
        </format>

        <format ...>
                ...
        </format>
<notice_formats>
```

Le principe est de définir une balise **format** par format à supporter. Pour chacune de ces balises, on créé des sous-balises **metadata** pour chaque métadonnée que l'on veut transformer avant son affichage. Typiquement si l'on veut traduire un terme de la fiche ou modifier un format de date. C'est ici un principe semblable au formats que l'on définit dans les champs de résultats **result_field**.

Configuration des formats

Les attributs de la balise **format** sont les suivants:

formatMetadataValue

On renseigne ici le namespace du format de métadonnées que l'on veut supporter. On peut avoir 2 types de configuration:

- **formatMetadataValue="http://ltsc.ieee.org/xsd/LOM"** : on renseigne directement le namespace. Dans le cas d'un format où plusieurs namespaces sont disponibles (dans le cas de changement de version par exemple), on peut avoir la syntaxe_formatMetadataValue="namespace_1_**|***namespace_2***|***namespace_3"*.
- **formatMetadataValue="vocabulary:identifiant_vocabulaire:identifiant_categorie"** où *identifiant_vocabulaire* est l'identifiant du vocabulaire distant et *identifiant_categorie* est l'identifiant de la catégorie dans le vocabulaire où se trouve le(s) namespace(s) souhaité(s)

prefix

On renseigne le préfixe XML correspondant au namespace.

xsl

L'affichage de la fiche de métadonnées se fait par transformation XSLT. Dans ce paramètre, on renseigne donc le fichier de transformation XSL qui se trouve lui-même dans le dossier **webapp/WEB-INF/xsl**. Voir la Section "Transformation XSL des fiches de métadonnées" pour la configuration de ce fichier XSL. Dans la distribution, il existe 2 affichages possibles pour la notice LOM et Dublin Core: **lom1.xsl**, **lom2.xsl**, **dc1.xsl** et **dc2.xsl**

headXsl

Ce champ permet de spécifier la XSL à utiliser pour afficher le bloc HEAD de la page HTML rendue lors de l'affichage d'une notice. Ce champ est OBLIGATOIRE. La XSL renseignée ici permet de spécifier le champ **title** et les **metakeywords** et **description**. Le fichier donné en configuration doit obligatoirement se trouver dans le dossier **webapp/WEB-INF/xsl/head**. Dans cette distribution, les XSL du LOM et Dublin Core sont disponibles: respectivement **lom.xsl** et **dc.xsl**.

Il est possible de définir des espaces de noms supplémentaires dans cette configuration en utilisant la balise **additional_ns**. Ceci peut être utilisé dans le cas d'un format utilisant plusieurs espaces de noms ou si vous avez ajouté un espace de nom dans un format existant. Ceux-ci doivent obligatoirement être définis pour qu'ils soient traités lors de la transformation XSLT qui permet le rendu de la fiche de métadonnées complète. Sans la définition des ces espaces supplémentaires, vous ne pourrez pas faire apparaître les champs liés à ces espaces dans le moteur de recherche. La balise **additional_ns** est répétable. Elle contient les attributs suivants:
prefix

On renseigne le préfixe XML correspondant au namespace.

namespaceUri

Correspond au namespace.

showXmlLink

Permet de dire si on souhaite proposer la visualisation de la fiche XML dans son état d'origine au niveau de l'interface de recherche. **true** pour une visualisation, **false** sinon. La valeur par défaut est **true**.

Chaque balise **metadata** permet de configurer une transformation avant affichage des métadonnées. Les attributs et configurations possibles sont les suivants:

Attribut format

Ceci renseigne le format du champ que l'on veut transformer. Il peut y avoir plusieurs types différents:

- **format="date:dd-MM-yyyy"** pour transformer une date vers le format désiré. En prenant la configuration citée ici, le 22 juin 2005 sera affiché comme *22-06-2005*.
- **format="vocabulary:identifiant_vocabulaire"**. Dans ce cas, la valeur retrouvée est cherchée en correspondance dans les valeurs du vocabulaire choisi. Lorsque cette valeur est trouvée, on affiche le libellé correspondant dans la langue sélectionnée dans l'interface. Par exemple, on spécifiera comme ceci le vocabulaire des langues: **format="vocabulary:search_languages"**. Ceci peut être utilisée dans le cas de l'affichage de la langue de saisie d'une ressource pédagogique. Si la valeur *fr-FR* est retrouvée, elle sera automatiquement traduite en *Français* si on affiche en français, ou *French* si on affiche en anglais, etc.
- **format="size:search_traduction_size"** permet d'afficher une taille de fichiers en octets. *search_traduction_size* étant le vocabulaire permettant de traduire les termes *bytes*, *Kb*, *Mb*, etc.
- **format="time:search_traduction_time"** permet d'afficher une durée de type P1Y2M3DT4H5M6S. *search_traduction_time* étant le vocabulaire permettant de traduire les termes *year*, *month*, *hour*, etc.
- **format="vcard"**. On indique ici que la métadonnée contient une vcard. Cette vcard est alors transformée en XML dans la fiche de métadonnée avant d'être fournie à la XSL. La XSL peut alors afficher tout ou partie de cette vcard en XML.

Attribut metadataDateFormat

Cet attribut permet de dire dans quel format est stockée la date. Par exemple **metadataDateFormat="yyyy-MM-dd"**. Pour dire que la métadonnée que l'on récupère est de la forme *2005-06-22* .

Contenu de la balise

Le contenu est lui un chemin xpath correspondant au chemin de la métadonnée dans la fiche XML ou un nom de métadonnée dans le cas des métadonnées repository, datestamp, etc.

Ajout de liens vers la recherche thématique

Il est possible pour certaines métadonnées d'ajouter un lien dans l'interface pour rebondir vers une recherche thématique sur celui-ci. Par exemple, lors de l'affichage d'un auteur, on pourra mettre un lien sur son nom pour lancer une recherche thématique uniquement sur ce nom. On peut imaginer la même chose pour les mots-clefs, etc.
Ceci se spécifie dans une sous-balise de **metadata:thematic_link**. Les attributs de **thematic_link** sont les suivants:

	key Cette clef doit être unique pour tout le format et est utilisée dans la XSL pour identifier les liens HTML à afficher. Elle sera utilisée par l'attribut **ori-link-key** dans la XSL. **thematicMenuKey** Cette variable doit OBLIGATOIREMENT être couplée à thematicSearchKey. Les 2 variables associées permettent d'aller sélectionner des valeurs pour ce champ depuis une navigation dans une recherche thématique en mode "esclave". Cet attribut doit donc contenir la clef du menu de recherche (défini dans **config.xml**) contenant la recherche thématique souhaitée.

	thematicSearchKey Cette variable doit donc contenir la clef du sous-menu de recherche thématique souhaité. **showAbsoluteLink** Vaut **true** ou **false** que l'on veuille mettre le lien complet vers la recherche thématique ou non. Si **true**, on affichera par exemple: *J > Ja > James*. En revanche, si **false**, on affichera uniquement le dernier niveau: *James* . **vcardAttribute** Dans le cas où la métadonnées est une vcard, on spécifie ici l'attribut de cette vcard sur lequel on veut construire le lien.

Ajout de liens versune autre notice

Il est possible pour certaines métadonnées d'ajouter un lien dans l'interface pour rebondir vers une autre notice. Par exemple, lors de l'utilisation des relations dans le format LOM.
Le concept est le suivant:

- Une fiche A contient une métadonnée /mon/xpath/M1 qui a pour valeur 123
- Cette même fiche A contient une métadonnée /mon/xpath/ML qui a pour valeur le libellé correspondant à la métadonnée /mon/xpath/M1
- Une fiche B contient une métadonnée /monautrexpath/M2 qui a pour valeur 123
- Nous pouvons donc proposer dans l'affichage de A un lien vers la fiche unique B par la configuration suivante:

```
<metadata>
        /mon/xpath
        <notice_link key="mon_lien" labelXpath="ML" targetXpath="M1"
 xpathInTarget="/monautrexpath/M2"/>
</metadata>
```

Ceci se spécifie dans une sous-balise de **metadata:notice_link**. Les attributs de **notice_link** sont les suivants:

	key Cette clef doit être unique pour tout le format et est utilisée dans la XSL pour identifier les liens HTML à afficher. Elle sera utilisée par l'attribut **ori-notice-link-key** dans la XSL. **labelXpath** Cette variable permet de dire le xpath où se trouve le libellé à afficher sur le lien dans la fiche d'origine. Attention, le xpath est relatif au xpath défini dans la balise **metadata**. **targetXpath**

Cette variable doit contenir le xpath dans lequel se trouve la valeur qui permettra de rebondir vers l'autre fiche. Attention, le xpath est relatif au xpath défini dans la balise **metadata**.

xpathInTarget

Cette variable contient le xpath ABSOLU où se trouve la valeur désirée dans la fiche visée.

Exemple de la relation *haspart* du LOM:

```
<metadata>
        //lom:relation[lom:kind/
lom:value='haspart']
        <notice_link key="lom_haspart"
 labelXpath="lom:resource/lom:description/
lom:string[starts-with(@language,'fr')]"
 targetXpath="lom:resource/lom:identifier/
lom:entry" xpathInTarget="//lom:general/
lom:identifier/lom:entry"/>
</metadata>
```

Transformation XSL des fiches de métadonnées

Afin de personnaliser vos fichiers XSL de transformation ou ajouter de nouveaux formats, il est recommandé de regarder les formats fournits avec l'application.

Toutefois, nous pouvons voir que les messages liés à l'internationalisation doivent êtres passés en paramètre des XSL. Ces messages doivent être déclarés en en-tête de la XSL sous forme de balises comme l'exemple **<xsl:param name="xsl.lom.url"/>** où xsl.lom.url est la clef du message (c.f. la Section "Personnalisation des messages et libellés") saisie dans les différents bundles de langues. Le passage des paramètres est effectué au niveau de la JSP **webapps/WEB-INF/stylesheets/notice/notice.jsp** par des balises de la forme:

```
<x:param name="xsl.lom.url">
        <fmt:message key="xsl.lom.url"/>
</x:param>
```

où *<fmt:message key="xsl.lom.url"/>* représente la récupération du message dans les différents bundles en fonction de la langue et *<x:param name="xsl.lom.url">...</x:param>* effectue ce passage à la XSL.

La construction des liens vers des recherches thématiques nécessite aussi une petite explication. En effet, lors de la configuration des différents formats (Section "Configuration des formats"), la fiche XML est modifiée pour voir apparaître tous les paramètres nécessaires à la construction du lien. Une métadonnée dont le xpath serait *general/keyword* deviendrait alors:

```
<general>
        <keyword ori-link-key="..." ori-menuKey="..." ori-submenuKey="..." ori-id="..." ori-
label="...">Valeur d'origine</keyword>
</general>
```

où:

ori-link-key

la clef définie dans la configuration de la métadonnée dans le format

ori-menuKey

valeur de thematicMenuKey de la configuration

ori-submenuKey

valeur de thematicSearchKey

ori-id

l'identifiant de la catégorie du vocabulaire qui correspond à la valeur retrouvée dans la fiche de métadonnée

ori-label

le libellé traduit correspondant à la valeur initiale et trouvée dans le vocabulaire en fonction de la langue de l'utilisateur

Un exemple de lien généré peut être le suivant:

```
<a href="http://[HOST_SEARCH]:[PORT_SEARCH]/[CONTEXT_SEARCH]/thematic-search.html?menuKey={@ori-menuKey}&amp;submenuKey={@ori-submenuKey}&amp;id={@ori-id}"><xsl:value-of select="@ori-label"/></a>
```

Personnalisation des messages et libellés

Tous les messages et libellés de l'application sont stockés dans des bundles i18n dans le dossier **properties/messages**. Il y a 6 différents fichiers: **messages_XX.properties**, **menus_XX.properties**, **forms_XX.properties**, **errors_XX.properties**, **xsl_XX.properties**, **results_XX.properties** où XX représentent les codes de langues. Les différents fichiers doivent donc être disponibles pour chaque langue que l'on souhaite gérer dans l'application.

Tous les fichiers sont décrits en français et en anglais dans cette distribution. Voyons en détail comment sont construits ces fichiers.

messages_XX.properties

Ce fichier contient tous les messages généraux à l'application non liés aux configurations personnalisées des menus de recherche. Vous pouvez les adapter pour une interface graphique personnalisée.

menus_XX.properties

Ce fichier contient les libellés liés aux différents menus de recherche. Vous pouvez les adapter ou l'agrémenter si vous ajouter de nouvelles interfaces de recherche.

On notera dans la suite:

- *menuKey* la clef d'un menu de recherche saisie dans **properties/config.xml à modifier** en fonction du menu que l'on configure
- *submenuKey* la clef d'un formulaire de recherche **à modifier**
- *resultFieldKey* la clef d'un champ de résultat de recherche **à modifier**

Le libellé d'un menu de recherche est construit comme suit:

```
menu.label.menuKey=Libellé du menu
```

Les interfaces de recherche sont définies quant à elles comme ceci:

```
menu.label.menuKey.submenuKey=Libellé du formulaire de recherche
menu.description.menuKey.submenuKey=Description du formulaire de recherche
```

Enfin, on configure de la manière suivante les différents champs de résultat:

```
menu.result.menuKey.submenuKey.resultFieldKey=Libellé du champ
```

forms_XX.properties

Ce fichier permet de configurer tous les messages liés aux formulaires de recherche avancée.

On notera dans la suite:

- *menuKey* la clef d'un menu de recherche avancée saisie dans **properties/config.xml à modifier** en fonction du menu que l'on configure
- *submenuKey* la clef d'un formulaire de recherche avancée **à modifier**
- *groupId* l'identifiant d'une balise **group à modifier**
- *fieldId* l'identifiant d'une balise **field à modifier**

Tous les paramètres suivants sont obligatoires. Si vous ne voulez pas de messages dans certains cas, définissez la valeur avec une chaîne vide.

Le libellé d'un groupe de champs se configure comme ceci:

```
menu.form.label.menuKey.submenuKey.groupId=Libellé du groupe de champs de recherche
```

Les champs de recherche sont eux configurés comme ceci:

```
menu.form.label.menuKey.submenuKey.fieldId=Libellé du champ de recherche
menu.form.comment.menuKey.submenuKey.fieldId=Commentaire, aide sur le champ de recherche
```

errors_XX.properties

Tous les messages d'erreurs sont configurés ici. Vous n'avez à priori à y toucher qu'en cas de personnalisation des messages.

Pour information, un message provenant d'une levée d'exception est configuré comme ceci:

```
error.search.MonOriOaiException=Erreur à désigner ici en fonction de l'exception
```

et un message d'erreur provenant d'une erreur dans le format de saisie des champs de recherche avancée est constitué comme ceci pour l'exemple d'un entier:

```
error.search.type.int=Valeur entière réclamée pour le champ "{0}"
```

xsl_XX.properties

Tous les messages utilisés dans les XSL sont configurés ici. On préféreras la syntaxe suivante pour les clefs des messages:

```
xsl.prefixe_du_format.nom_de_metadonnee=Libellé de la métadonnée
```

results_XX.properties

Contient tous les messages liés à des affichages de résultats personnalisés. Cela correspond aux JSP contenues dans le dossier **webapp/WEB-INF/stylesheets/results**.

Gestion du cache

Dans cette version du module ori-oai-search, il existe du cache à deux niveaux:

- sur les vocabulaires: les vocabulaires utilisés dans ce modules sont récupérés auprès du module ori-oai-vocabulary. Afin de ne pas solliciter ce dernier trop souvent, les vocabulaires sont alors mis en cache dans le module ori-oai-search
- sur les fiches de métadonnées: lorsqu'une fiche de métadonnée est affichée dans le module ori-oai-search, elle subit plusieurs transformations comme la traduction de certains champs. Afin d'alléger les requêtes, ces fiches sont alors mises en cache dans le module ori-oai-search.

Le cache est enregistré sur disque dans le dossier [TOMCAT_HOME]/temp. Les différents fichiers de cache sont:

- **ori-oai-search_vocabulary.data** et **ori-oai-search_vocabulary.index** pour le cache sur le vocabulaire
- **ori-oai-search_notice.data** et **ori-oai-search_notice.index** pour le cache sur les fiches de métadonnées

Si vous souhaitez vider le cache du module ori-oai-search, il vous suffit donc de supprimer les fichiers correspondant et de redémarrer le serveur Tomcat.

Toute la configuration des caches se fait dans le fichier **properties/ehcache.xml**. Ce fichier contient la définition des paramètres de cache **ori-oai-search_vocabulary** et **ori-oai-search_notice**. Vous trouverez la documentation nécessaire à la configuration de ce fichier ici: http://ehcache.sourceforge.net/EhcacheUserGuide.html#id.s6.3

> Le cache sur les résultats de recherche n'est pas géré au niveau du module ori-oai-search, mais au niveau de **ori-oai-indexing**. Se reporter à la documentation de ce module pour plus de précision.

Personnalisation des interfaces graphiques

Toutes les interfaces ont été conçues pour pouvoir être au maximum modifiées depuis la CSS. La CSS utilisée dans ce module est **ori-oai-search.css** qui se trouve dans le dossier **webapp/css**.

Il existe plusieurs CSS proposées dans cette version. Pour changer le style de votre application, il suffit de renommer une des CSS choisies en ori-oai-search.css.

- **ori-oai-search.css** qui est la dernière CSS proposée pour ce module
- **ori-oai-search_old-style.css** qui est la première CSS qui a été créée
- **ori-oai-search_unr-npdc.css** qui est une CSS en cours d'élaboration pour l'UNR Nord-pas-de-Calais mais qui peut servir comme point de départ à d'éventuels établissements
- **ori-oai-search_portlet_uvhc.css** qui est la CSS qui a été développée pour le module ori-oai-search à l'Université de Valenciennes en mode *portlet*

Dans le cas où les configurations des CSS ne sont pas suffisantes pour personnaliser les pages, il faudra modifier les JSP de génération des interfaces. Toutes celles-ci se trouvent dans le dossier **webapp/WEB-INF/stylesheets**.

Lors d'un déploiement en mode portlet dans un ENT par exemple, aucun en-tête HTML n'est généré par l'application. En effet, le HTML généré par la portlet ori-oai-search est intégré dans un environnement autre. Dans ce cas, aucune CSS n'est associée par défaut à votre interface. Il est donc impératif d'ajouter le lien vers votre CSS **ori-oai-search.css** depuis l'en-tête HEAD de votre moteur de portlet.

Dans le cas de l'ENT ESUP Portail, vous pouvez par exemple importer la CSS de ori-oai-search depuis la CSS **???_portlet..css** du dossier **VOTRE_CHEMIN/uPortal/media/org/jasig/portal/layout/AL_TabColumn/integratedModes/VOTRE_DOSSIER_DE_SKINS/skin** comme ceci:

```
@import url("../../../../../../../../../../[CONTEXT_SEARCH]/css/ori-oai-search.css");
```

Où le nombre de sauts dans l'arborescence est à personnaliser selon votre cas.

Divers

Fonctionnalités diverses

En plus des différentes fonctionnalités qui ont été décrites dans ce document, il existe d'autres fonctions très utiles:

sitemap

Il existe la possibilité en mode *servlet* de générer un sitemap dynamique pour une meilleure indexation de votre site par des moteurs de recherche de type google par exemple. Ce sitemap est accessible par

```
http:// [HOST_SEARCH]:[PORT_SEARCH]/[CONTEXT_SEARCH]/sitemap.xml
```

listing de toutes les ressources

En mode de recherche thématique classique, il est possible de lister toutes les ressources de l'arborescence dans un fichier au format RTF. Pour cela, il suffit de faire une recherche thématique et de remplacer dans l'adresse de la requête le **.html** en **.rtf**. Un exemple est donc

```
http:// [HOST_SEARCH]:[PORT_SEARCH]/[CONTEXT_SEARCH]/thematic-search.rtf?menuKey=unt&submenuKey=thematic_unit&id=1
```

Test

This page last changed on Nov 25, 2008 by ycolmant@univ-valenciennes.fr.

Vous pouvez tester la bonne configuration du module et l'accès aux interfaces de recherche en allant sur :

```
http://[HOST_INSTALL]:8184/ori-oai-search
```

Notez que l'index est vide à cette étape, vous ne trouverez donc aucune ressource lors de la recherche.

Utilisation

This page last changed on Nov 25, 2008 by ycolmant@univ-valenciennes.fr.

Documentation à venir ...