Search - Définir sa configuration de recherche

1ère étape : choisir ou s'inspirer d'une configuration existante

Dans ori-oai-search, plusieurs configurations sont proposées par défaut :

- **default_demo** : configuration d'exemple uniquement qui montre toutes les possibilités de paramétrage des interfaces du module. Il propose la recherche sur les formats Dublin Core, LOM, TEF, AO et CDMv1 indépendamment. Il propose aussi à titre d'exemple une recherche multi-format pour une recherche croisée sur les différents types de métadonnées.

ATTENTION : cette configuration n'est pas destinée à la production.

- **default_full** : configuration complète utilisable en production. Il propose la recherche sur les formats Dublin Core, LOM, TEF, AO et CDMv1 indépendamment. Il propose aussi une recherche multi-format pour une recherche croisée sur les différents types de métadonnées.
- **default_dc** : configuration pour la recherche de ressources en utilisant le format de métadonnées Dublin core.
- **default_lom** : configuration pour la recherche de ressources pédagogiques en utilisant le format de métadonnées LOM.
- **default_tef** : configuration pour la recherche de thèses en utilisant le format de métadonnées TEF.
- **default_ao** : configuration pour la recherche de publications scientifiques.
- **default_cdmv1** : configuration pour la recherche de formations en utilisant le format CDM (version 1).
- **default_mixed** : configuration qui propose un unique mode de recherche multi-format pour une recherche croisée sur les différents types de métadonnées.
- *d'autres à venir ...*

Vous avez le choix entre :

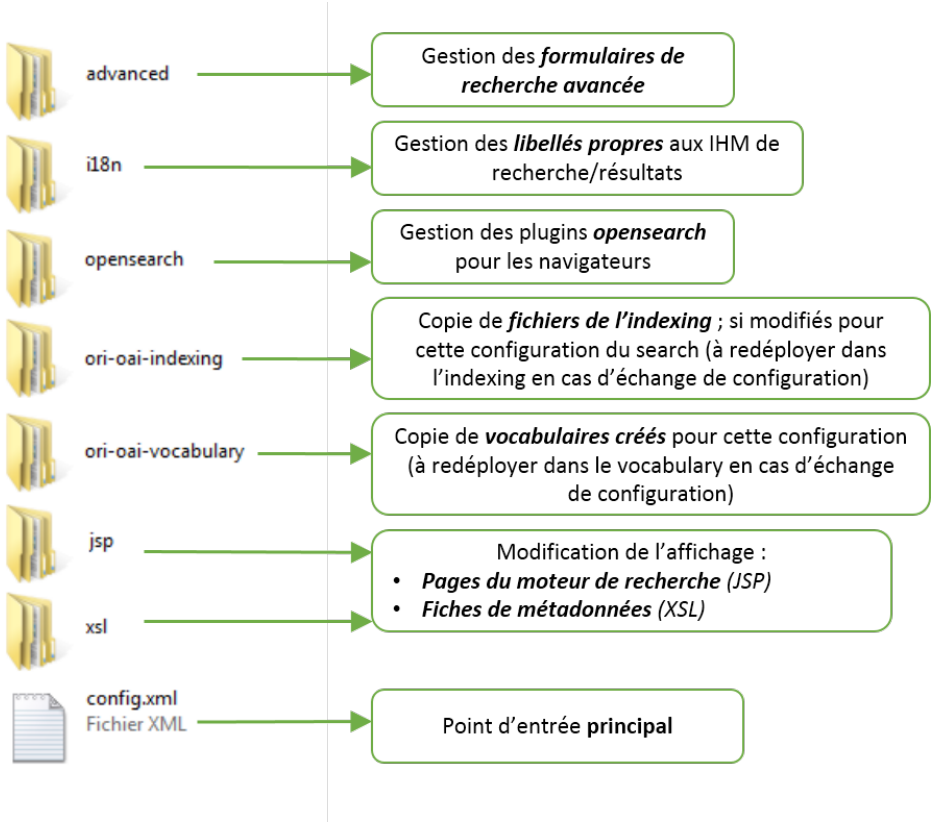
- utiliser une des configurations proposées par défaut,
- modifier une des configurations proposées par défaut,
- créer une configuration en partant de zéro, mais le travail sera plus fastidieux !

Une configuration personnalisée est représentée par un dossier dans [PATH_CUSTOM_CONFIG]/ori-oai-search/**contribs-search**.
Pour définir votre propre configuration, il est nécessaire de s'inspirer d'une configuration existante ! Pour cela, il faut recopier une configuration existante depuis les sources du modules ([ORI_HOME]/src/ori-oai-search-svn/src/main/resources/**contribs-search**) vers le dossier de customisation ([PATH_CUSTOM_CONFIG]/ori-oai-search/**contribs-search**).
Il en existe plusieurs par défaut (le nom du dossier commence par *default*). Certains proviennent aussi des membres de la communauté ORI-OAI (le nom commence par *contrib*).
Nous vous encourageons à contribuer et à nous fournir vos configurations. Nous les proposerons alors aux utilisateurs afin de faciliter de travail de configuration. Reportez-vous pour cela à la page suivante de notre site.

2nde étape : adapter la configuration

Dans ce chapitre, nous verrons comment paramétrer la recherche en fonction des besoins.

Structure d'une configuration personnalisée



Un dossier de configuration est composé comme suit :

- Fichier **config.xml** : Ce fichier correspond à la configuration principale des interfaces de recherche et de résultat.
- Dossier **advanced** : Dans le cas de l'utilisation de recherche avancée, il existe un fichier de configuration par recherche avancée. Ces fichiers doivent se trouvent dans le dossier **advanced**.
- Dossier **i18n** : Tous les libellés propres à la configuration personnalisée sont définis dans ce dossier. Il est possible de surcharger

également des bundles de message définis par défaut dans ORI-OAI. Pour cela, il faut les définir dans les fichiers **custom_XX.properties**. Ces messages seront donc prioritaires sur ceux définis par défaut.

- Dossier **opensearch** : En mode servlet, il est possible de définir des configurations pour le plugin opensearch. Tout se fait dans ce dossier.
- Dossier **jsp** : Dans certains cas, il est nécessaire de modifier les JSP utilisées lors de la génération des pages HTML du moteur de recherche. Les JSP par défaut peuvent donc être surchargées en les copiant depuis le dossier source.
- Dossier **xsl** : Tout comme il est possible d'ajouter ou de surcharger certaines JSP, il est possible de le faire avec les XSL utilisées lors de l'affichage d'une fiche de métadonnées.
- Dossier **ori-oai-indexing** : Votre configuration de recherche peut nécessiter une configuration spécifique du module ori-oai-indexing (configIndexing.xml, fieldsConfig.xml, etc.). Ainsi, les établissements utilisant votre configuration pourront déployer MANUELLEMENT ces fichiers de configuration dans leur instance locale. Aucun fichier contenu dans le dossier ori-oai-indexing n'est déployé ou utilisé en l'état. Ce dossier sert uniquement au bon échange des contributions dans la communauté.
- Dossier **ori-oai-vocabulary** : Votre configuration de recherche peut nécessiter l'utilisation de vocabulaires spécifiques. Dans ce cas, nous conseillons de stocker dans ce dossier les vocabulaires statiques et les configurations de vocabulaires dynamiques nécessaires au bon fonctionnement du moteur de recherche. Ainsi, les établissements utilisant votre configuration pourront déployer MANUELLEMENT ces vocabulaires dans leur instance locale. Aucun fichier contenu dans le dossier ori-oai-vocabulary n'est déployé ou utilisé en l'état. Ce dossier sert uniquement au bon échange des contributions dans la communauté.

Passons en détail tous les dossiers et fichiers composant une configuration de recherche :

Fichier "config.xml"

```
<config>

 <!-- ================================================================================ -->
 <!-- ========================== Paramètres généraux ========================== -->
 <...../>


 <!-- ================================================================================ -->
 <!-- ========== Définition des paramètres de recherche globaux  ============= -->
 <default_results>

  <!-- ======== Champs affichés dans les résultats ======== -->
  <result_fields/>

  <!-- ======== Champs de tri ======== -->
  <sort_fields/>

  <!-- ======== Champs pour les flux RSS ======== -->
  <rss_fields/>

  <!-- ======== Champs pour l'affinage - facettes ======== -->
  <refine_search_fields/>

  <!-- ======== Champs pour les rebonds depuis les résultats ======== -->
  <rebound/>

 </default_results>


<!-- ================================================================================ -->
<!-- ======================== Menus de recherche ============================== -->
<menu>

 <!-- Menu de niveau 1 -->
 <search_menu>

  <!-- Zone de recherche simple dans le menu -->
  <simple_date_search/>
  <simple_search/>
```

```
   <!-- Zone de focus dans le menu -->
   <tag_cloud/>
   <random/>

   <!-- Sous-menu de recherche par nouveautés -->
   <date_search>
    <metadatas>...</metadatas>  <!-- Champs sur lequels rechercher -->
    <hidden_fields>...</hidden_fields> <!-- Champs de recherche cachés pour filtrer
des résultats -->

    <result_fields/>  <!-- Champs affichés dans les résultats -->
    <sort_fields/>  <!-- Champs de tri -->
    <rss_fields/>  <!-- Champs pour les flux RSS -->
    <refine_search_fields/> <!-- Champs pour l'affinage - facettes -->
    <rebound/>   <!-- Champs pour les rebonds depuis les résultats -->
   </date_search>

   <!-- Sous-menu de recherche avancée -->
   <advanced_search>
    ...
   </advanced_search>

   <!-- Sous-menu de recherche thématique -->
   <thematic_search>
    ...
   </thematic_search>
  </search_menu>

 </menu>


 <!-- ================================================================================ -->
 <!-- =================== Champs de résultat du panier ======================== -->
 <cart_result_fields/>


 <!-- ================================================================================ -->
 <!-- ================ Zone de recherche simple et focus ===================== -->
 <simple_search/>
 <simple_date_search/>

 <tag_cloud/>
 <random/>


 <!-- ================================================================================ -->
 <!-- ======================== Plugin Open search ============================== -->
 <open_search/>


 <!-- ================================================================================ -->
 <!-- =========== Affichage de la fiche de métadonnées (notice)  ============= -->
 <notice_formats>
  <format/>
 </notice_formats>
```

```
</config>
```

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.*

<docs_per_page_interval>

> *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 l'intervalle entre chaque valeur.*

<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, le contenu de cette page peut être édité dans les fichiers [PATH_CUSTOM_CONFIG]/ori-oai-search/contribs-search/*votre_contrib*/jsp/index/content/**index-begin_XX.jsp** (haut de la page) et **index-end_XX.jsp** (bas de la page) 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.*
>
> *D'autres paramètres sont modifiables :*
>
> - ***simple_search** (vaut **true** ou **false**) : affiche ou non le champ de recherche simple sur la page d'accueil*
> - ***count_search** (vaut **true** ou **false**) : affiche ou non en bas de page le nombre de ressources par format*
> - ***refine_search** (vaut **true** ou **false**) : affiche ou non le bandeau de facettes sur la gauche de la page d'accueil*
> - ***refine_menu_key** : si les facettes sont affichées, renseigne la clef du menu dans lequel prendre la config de facette*
> - ***refine_search_key** : si les facettes sont affichées, renseigne la clef du sous-menu dans lequel prendre la config de facette*
> - ***refine_default_open** (vaut **true** ou **false**) : ouvre ou non par défaut toutes les facettes de la page d'accueil dans le cas où elles sont affichées.*

<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_availabl e>** soit placé à true)*
> - ***<defaut_page index="false" 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**.*

<static_pages>

> *Cette balise sert à définir une liste de pages statiques dans l'interface. La création d'une page à ce niveau nécessite qu'une page de contenu soit créée pour chaque page et chaque langue dans le dossier [PATH_CUSTOM_CONFIG]/ori-oai-search/contribs-search/*votre_contrib*/jsp/page/content. Par exemple, si l'identifiant de la page est "**ma-page**", alors il faut créer les fichiers "**ma-page_fr.jsp**" et "**ma-page_en.jsp**".*
>
> *La définition d'une page se fait comme ceci :*
>
> ```
> <page id="ma-page" position="footer|simple-search"/>
> ```
>
> - ***id** : l'identifiant de la page doit être unique.*
> - ***position** : définir la(les) position(s) où afficher un lien vers cette page dans l'interface. Il est possible de mettre plusieurs liens. Dans ce cas, il faut séparer les positions par le caractère "|" (par exemple "footer|simple-search"). Les valeurs possibles sont :*
>   - ***header** : affiche un lien dans l'en-tête de la page web,*
>   - ***simple-search**{:left|:right} : affiche un lien sous le champ de recherche simple de la page d'accueil ou du bandeau. Il est possible de spécifier si le lien doit être placé à gauche ("simple-search:left") ou à droite ("simple-search:right"),*
>   - ***menu**{:left|:right} : affiche un lien dans le menu principal. Il est possible de spécifier si le lien doit être placé*

à gauche ("menu:left") ou à droite ("menu:right"),

- ***submenu**{:left|:right} : affiche un lien dans le sous-menu. Il est possible de spécifier si le lien doit être placé à gauche ("submenu:left") ou à droite ("submenu:right"),*
- ***footer** : affiche un lien dans le pied de page de la page web.*

<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.

<one_level_submenu_top>

*Vaut **true** ou **false**.*
Dans le cas où il n'y a qu'un seul niveau de menu, il est possible de faire remonter les sous-menu au niveau du menu dans l'interface.

<ajax>

Il est possible d'utiliser AJAX dans certaines interfaces du module de recherche:

- ***search** qui vaut **true** ou **false** si vous souhaitez utiliser ou non la possibilité d'afficher et de naviguer les résultats en utilisant AJAX.*
- ***advanced** qui vaut **true** ou **false** si vous souhaitez utiliser ou non la possibilité d'ouvrir une fenêtre superposée lors de la modification d'une recherche avancée.*
- ***notice** qui vaut **true** ou **false** si vous souhaitez utiliser ou non la possibilité d'ouvrir une fenêtre superposée lors de l'affichage d'une notice depuis la liste des résultats.*
- ***autocomplete** qui vaut **true** ou **false** si vous souhaitez utiliser ou non l'autocompletion/aide à la saisie dans les champs de recherche simple ou recherche avancée.*

<cart>

- ***available** qui vaut **true** ou **false** si on souhaite ou non activer la fonctionnalité de panier.*

<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 "Dossier "i18n" : 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>
```

<default_results> : Définition des paramètres de recherche globaux

Cette zone permet de définir des paramètres de résultats utilisables dans tous les menus de recherche. Ceux-ci touchent les balises: **<result _fields>**, **<sort_fields>**, **<rss_fields>**, **<refine_search_fields>** et **<rebound>** :

```
<default_results>

 <!-- ======== Champs affichés dans les résultats ======== -->
 <result_fields id="lom" ...>
  ...
 </result_fields>

 <!-- ======== Champs de tri ======== -->
 <sort_fields id="score_title_date_author" ...>
  ...
 </sort_fields>

 <!-- ======== Champs pour les flux RSS ======== -->
 <rss_fields id="notice" ...>
  ...
 </rss_fields>

 <!-- ======== Champs pour l'affinage - facettes ======== -->
 <refine_search_fields id="lom">
  ...
 </refine_search_fields>

 <!-- ======== Champs pour les rebonds depuis les résultats ======== -->
 <rebound id="lom">
  ...
 </rebound>

</default_results>
```

Les définitions **<result_fields>**, **<sort_fields>**, **<rss_fields>**, **<refine_search_fields>** et **<rebound>** seront donc simplement appelées via leur identifiant **id** dans les différentes définitions de recherche.

Voyons les différents blocs :

<result_fields> : Champs affichés dans les résultats

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 id="lom" jsp_file="my_results.jsp" show_notice_link="true"
show_more_link="true" show_share_link="true" show_cart_link="true"
show_thumbnail="true" href_document="true">

 <result_field key="score" display_type="score" highlight="false"
show_label="true">
  <metadata>score</metadata>
 </result_field>

 <result_field key="title" title_field="true" show_label="false"
href_document="true">
  <metadata language="true">lom.general.title</metadata>
 </result_field>

 <result_field key="keyword" values_separator=",&#32;">
  <metadata language="true">lom.general.keyword</metadata>
 </result_field>

 <result_field key="date" format="date:dd-MM-yyyy,MM-yyyy,yyyy" highlight="false">
  <metadata
dateFormat="yyyy-MM-dd,yyyy-MM,yyyy">lom.lifeCycle.contribute.author.dateTime</meta
data>
 </result_field>

 <result_field key="learningResourceType"
format="vocabulary:search_lom_educ_learningResourceType" more_area="true">
  <metadata>lom.educational.learningResourceType</metadata>
 </result_field>

</result_fields>
```

Configuration de la balise <result_fields>

id

> *Il est obligatoire de définir un identifiant par bloc* ***<result_fields>****.*
> *Au niveau de la balise* ***<default_results>****, le bloc* ***<result_fields>*** *sera défini en entier.*
>
> *Au niveau du menu de recherche, il suffira simplement d'appeler le bloc* ***<result_fields>*** *par son identifiant comme suit:*
>
> ```
> <result_fields id="lom"/>
> ```

jsp_file

> *Attribut qui permet de spécifier la page JSP qui sert à la mise en page des résultats pour l'interface de recherche ici présente. Il se trouve dans le dossier* ***[ORI_HOME]/src/ori-oai-search-svn/src/main/webapp/WEB-INF/jsp/results*** *dans les sources ou dans* ***[PATH_CUSTOM_CONFIG]/ori-oai-search/contribs-search/jsp/results*** *dans le cas d'un fichier personnalisé. Si aucune valeur n'est saisie, on prendra par défaut le fichier* ***generic-results.jsp****.*

show_notice_link

> *Vaut* ***true*** *ou* ***false****.*
> *Indique si on affiche le lien vers la notice ou non.*

show_more_link

> *Vaut* ***true*** *ou* ***false****.*
> *Indique si on affiche ou non la zone "En savoir plus".*

show_share_link

> *Vaut* ***true*** *ou* ***false****.*
> *Permet d'activer ou non la zone de partage.*

show_cart_link

> *Vaut **true** ou **false**.*
> *Indique si on veut ou non afficher le lien pour ajouter dans le panier.*

show_thumbnail

> *Vaut **true** ou **false**.*
> *Indique si on veut ou non afficher la vignette d'aperçu du document dans la liste des résultats.*

href_document

> *Vaut **true** ou **false**.*
> *Permet d'afficher ou non le lien vers le(s) document(s).*

Configuration de la balise <result_field>

Les balises **<result_field>** sont composées comme ceci:

key

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

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.*

href_document

> *Si vaut **true**, un lien vers le document est disponible à partir de ce champ.*

values_separator

> *Dans le cas où la métadonnée est multi-valuée, il est possible de spécifier la chaîne de caractères à utiliser comme séparateur.*

show_label

> *Vaut **true** ou **false**.*
> *Permet de dire si on affiche ou non le libellé du champ.*

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"** (ou **format="date:dd-MM-yyyy,MM-yyyy,yyyy"** ; dans ce cas, les 3 formats seront tentés successivement en fonction du format proposé dans la métadonnée)*
> - ***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.*

highlight

> *Vaut **true** ou **false** si l'on souhaite oui ou non surligner les valeurs retrouvées dans l'interface des résultats.*

highlight_only

> *Vaut **true** ou **false**.*
> *Permet de dire si on ne veut afficher le champ que quand il y a du highlight.*

display_type

> *Pour des besoins particulier, il est possible de spécifier un type d'affichage. Ce type sera pris en compte dans la JSP en charge de l'affichage et adaptera en fonction du type choisi. Par exemple, le type **score** permet de ne pas afficher le score au format numérique, mais sous forme d'une barre plus ou moins remplie, etc.*
> *Vous pouvez créer vos propres types et les prendre en compte dans une JSP personnalisée pour l'affichage des résultats.*

more_area

> *Vaut **true** ou **false**.*
> *Pour synthétiser l'affichage des résultats, mais permettre à l'usager d'avoir un maximum d'informations, il est possible d'afficher certaines valeurs de champs dans une zone "En savoir plus". Pour afficher le champ de résultat dans cette zone, il faut donc mettre le champ **more_area** à **true**.*

Configuration de la balise <metadata>

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

<metadata>

> *La valeur de cette balise correspond au nom de la métadonnée définie dans le module ORI-OAI-indexing.*

La balise <metadata> comporte aussi différents attributs :
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.*

language

> *Vaut **true** ou **false**.*
> *Le module ORI-OAI-indexing permet de gérer différentes langues pour certaines métadonnées. Dans ce cas, les métadonnées fonctionnent en trio. Pour le titre au format LOM, nous avons ces 3 champs :*
>
> - ***lom.general.title_fr** : titre saisi en français dans la fiche LOM*
> - ***lom.general.title_en** : titre saisi en anglais dans la fiche LOM*
> - ***lom.general.title** : titre dont nous n'avons pas su déterminer la langue ou que cette langue n'est pas supportée*
>
> *Dans le cas du champ **titre** du LOM, il est donc intéressant d'afficher la valeur en fonction de la langue de l'utilisateur. Dans ce cas, il faut utiliser l'attribut **language** comme ceci : **<metadata language="true">**.*
> *Si l'utilisateur a choisi l'interface en anglais, le moteur de recherche essayera successivement d'afficher **lom.general.title_en**, puis s'il n'existe pas **lom.general.title_fr** et enfin **lom.general.title** si on n'a ni la valeur en anglais, ni en français.*

Configuration de la balise <format_filter>

```
<result_field>
 <metadata>...</metadata>
 <format_filter>cdm</format_filter>
</result_field>
```

Dans le cas où le <result_fields> est utilisé par plusieurs formats de métadonnées, il est possible de filtrer dans quel cas afficher ou non tel ou tel <result_field>.
La balise **<format_filter>** permet donc de filtrer pour n'afficher le champ que pour tel ou tel format. Dans l'exemple ci-dessus, le champ ne sera affiché que si le résultat est au format **cdm**.

<sort_fields> : Champs de tri

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 id="score_title_date_author">
 <sort_field key="score">
  <metadata>score</metadata>
 </sort_field>
 <sort_field key="title" ascending="true">
  <metadata language="true">title_sort</metadata>
 </sort_field>
 <sort_field key="date" ascending="false">
  <metadata>date_creation_sort</metadata>
 </sort_field>
 <sort_field key="author" ascending="true">
  <metadata>author_sort.name</metadata>
 </sort_field>
</sort_fields>
```

id

> *Il est obligatoire de définir un identifiant par bloc* ***<sort_fields>****.*
> *Au niveau de la balise* ***<default_results>****, le bloc* ***<sort_fields>*** *sera défini en entier.*
>
> *Au niveau du menu de recherche, il suffira simplement d'appeler le bloc* ***<sort_fields>*** *par son identifiant comme suit:*
>
> ```
> <sort_fields id="score_title_date_author"/>
> ```

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 *author*.

Dans l'exemple de configuration précédent, on voit le tri de documents suivant plusieurs **<sort_field>**.

Chaque balise <sort_field> doit contenir une clef unique (**key**), l'attribut optionnel **ascending** et une balise **<metadata>** contenant le nom de la métadonnée à utiliser pour le tri :

ascending

> *Cet attribut peut valoir* ***true*** *ou* ***false*** *(****true*** *par défaut) 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.*

<metadata>

> *La valeur de cette balise correspond au nom de la métadonnée définie dans le module ORI-OAI-indexing.*
> *Attention, cette métadonnée doit* ***OBLIGATOIREMENT*** *être monovaluée dans le module ORI-OAI-indexing !*

La balise <metadata> comporte aussi différents attributs :
language

> *Vaut* ***true*** *ou* ***false****.*
> *Le module ORI-OAI-indexing permet de gérer différentes langues pour certaines métadonnées. Dans ce cas, les métadonnées fonctionnent en trio. Pour le titre commun à tous les formats et utilisé pour le tri, nous avons ces 3 champs :*
>
> - ***title_sort_fr*** *: titre saisi en français dans la fiche*
> - ***title_sort_en*** *: titre saisi en anglais dans la fiche*
> - ***title_sort*** *: titre dont nous n'avons pas su déterminer la langue ou que cette langue n'est pas supportée*
>
> *Dans le cas du champ* ***titre****, il est donc intéressant de trier par la valeur en fonction de la langue de l'utilisateur. Dans ce cas, il faut utiliser l'attribut* ***language*** *comme ceci :* ***<metadata language="true">****.*
> *Si l'utilisateur a choisi l'interface en anglais, le moteur de recherche essayera successivement de trier par* ***title_sort_en****, puis s'il n'existe pas* ***title_sort_fr*** *et enfin* ***title_sort*** *si on n'a ni la valeur en anglais, ni en français.*

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.

<rss_fields> : Champs pour les flux RSS

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 id="notice" link="notice">
 <title>title_md</title>
 <description>description</description>
 <pubDate>date</pubDate>
</rss_fields>
```

id

> *Il est obligatoire de définir un identifiant par bloc* ***<rss_fields>****.*
> *Au niveau de la balise* ***<default_results>****, le bloc* ***<rss_fields>*** *sera défini en entier.*
>
> *Au niveau du menu de recherche, il suffira simplement d'appeler le bloc* ***<rss_fields>*** *par son identifiant comme suit:*

```
<rss_fields id="notice"/>
```

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**.*

Les balises de ce bloc se configurent de la manière suivante:
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.*

<refine_search_fields> : Champs pour l'affinage - facettes

Il existe la possibilité d'activer une zone d'affinage des résultats sur la gauche de l'écran :

```
<refine_search_fields id="lom">
 <text_field>
  <metadata>...</metadata>
  ...
 </text_field>

 <choice_field>
  <metadata>...</metadata>
  ...
 </choice_field>

 <choice_field>
  <metadata>...</metadata>
  ...
 </choice_field>
</refine_search_fields>
```

id

> *Il est obligatoire de définir un identifiant par bloc **<refine_search_fields>**.*
> *Au niveau de la balise **<default_results>**, le bloc **<refine_search_fields>** sera défini en entier.*
>
> *Au niveau du menu de recherche, il suffira simplement d'appeler le bloc **<refine_search_fields>** par son identifiant comme suit:*
>
> ```
> <refine_search_fields id="lom"/>
> ```

Les balises de ce bloc se configurent de la manière suivante:

Configuration de la balise <text_field>

La balise **<text_field>** permet de configurer la zone d'affinage par recherche simple sur la gauche de l'écran dans l'affichage des résultats.

```
<text_field highlight="true">
 <metadata language="true" tagName="dc.title_tag"
autocomplete="true">dc.title</metadata>
 <metadata language="true" tagName="dc.description_tag"
autocomplete="true">dc.description</metadata>
 <metadata language="true" tagName="dc.subject_tag"
autocomplete="true">dc.subject</metadata>

 <metadata autocomplete="true" tagName="dc.creator_tag">dc.creator</metadata>
 <metadata autocomplete="true"
tagName="dc.contributor_tag">dc.contributor</metadata>
 <metadata autocomplete="true" tagName="dc.publisher_tag">dc.publisher</metadata>

 <metadata>md-ori-oai-plaintext</metadata>
</text_field>
```

La balise se configure comme ceci :

highlight

> *Permet de dire si oui ou non on surligne le terme recherché dans la liste des résultats.*

Chaque balise <text_field> est composée de balises **<metadata>**. Le contenu de la balise correspond au nom de la métadonnée sur laquelle rechercher.
La balise peut aussi comporter les attributs suivants :

language

> *Vaut **true** ou **false**.*
> *Le module ORI-OAI-indexing permet de gérer différentes langues pour certaines métadonnées. Dans ce cas, les métadonnées fonctionnent en trio. Pour le titre au format LOM, nous avons ces 3 champs :*
>
> - ***lom.general.title_fr** : titre saisi en français dans la fiche LOM*
> - ***lom.general.title_en** : titre saisi en anglais dans la fiche LOM*
> - ***lom.general.title** : titre dont nous n'avons pas su déterminer la langue ou que cette langue n'est pas supportée*
>
> *Dans le cas du champ **titre** du LOM, il est donc intéressant d'afficher la valeur en fonction de la langue de l'utilisateur. Dans ce cas, il faut utiliser l'attribut **language** comme ceci : **<metadata language="true">**.*
> *Si l'utilisateur a choisi l'interface en anglais, le moteur de recherche essayera successivement d'afficher **lom.general.title_en**, puis s'il n'existe pas **lom.general.title_fr** et enfin **lom.general.title** si on n'a ni la valeur en anglais, ni en français.*

autocomplete

> *Vaut **true** ou **false**.*
> *Permet d'activer ou non l'autocomplétion sur la saisie du champ. Dans ce cas dès que l'utilisateur commence à saisir un mot, l'interface propose la liste de tous les mots commençant par ce terme.*

tagName

> *Le module ORI-OAI-indexing s'appuie sur le moteur Solr pour l'indexation et la recherche de documents. Solr permet de définir des règles utilisées lors de l'indexation.*
> *Pour être pertinent dans la recherche, Solr effectue une transformation des valeurs au moment de l'indexation et de la recherche. Par exemple, il supprime les mots vides (le, la, les, dans, mon, ma, etc.), il tronque les verbes conjugués, les pluriels, etc.*
> *C'est uniquement la valeur **indexée** qui est transformée de la sorte. Solr peut aussi stocker la valeur initiale de la métadonnée pour permettre l'affichage de celle-ci.*
>
> *Un incovénient à ce système est que lorsque l'on veut récupérer des tags de l'index pour afficher un nuage de tags ou pour par exemple la fonctionnalité d'autocomplétion, nous sommes obligés de nous appuyer sur la valeur **indexée** et non pas la valeur **stockée**.*
> *Mais la valeur indexée ne convient pas car les mots ont été tronqués !*
>
> *Il est donc nécessaire de dupliquer le champ au niveau de la configuration de Solr (schema.xml) pour avoir une version du champ donc la valeur indexée n'a pas été tronquée. Ceci se fait facilement par des **copyField** dans Solr, mais la configuration doit se faire manuellement.*
>
> *Par convention, ces champs dupliqués sont nommés de la même manière que le champ d'origine avec le suffixe **_tag**.*
>
> *Par exemple, la version non tronquée de la métadonnée **dc.title** est **dc.title_tag**.*
>
> *L'attribut **tagName** sert donc à indiquer le nom de la métadonnée non tronquée sur laquelle il faut s'appuyer pour récupérer la liste de tags.*

Configuration de la balise <choice_field>

La balise **<choice_field>** permet de configurer une facette dans la zone d'affinage sur la gauche de l'écran dans l'affichage des résultats.

```
<choice_field key="keyword" format="cloud:24" tag_cloud_random_display="true"
tag_cloud_show_occur="false">
 <metadata language="true">dc.subject_tag</metadata>
</choice_field>

<choice_field key="language" format="vocabulary:search_languages"
order="vocabulary">
 <metadata>dc.language</metadata>
</choice_field>

<choice_field key="doc_format" format="vocabulary:search_formats" operator="OR">
 <metadata>dc.format</metadata>
</choice_field>
```

La balise se configure comme ceci :

key

> *Clef unique pour cette facette.*

format

> *Format à utiliser pour la création de la facette.*
>
> *Plusieurs possibilités :*
>
> - pas de balise *: dans ce cas, on récupère simplement les valeurs indexées des métadonnées définie dans la balise <metadata>*
> - ***cloud:xx** : permet de récupérer simplement les valeurs indexées mais en les présentant sous forme de nuage de tags avec au maximum **xx** valeurs*
> - ***vocabulary:*mon-vocabulaire** *: permet de faire un mapping entre la valeur réellement indexée et un vocabulaire. Par exemple, si la valeur indexée est la langue au format "fr-FR", alors en utilisant le vocabulaire adéquat, la valeur affichée sera modifiée en "Français"*

operator

> *Vaut **OR** ou **AND**.*
> *Si la valeur vaut **OR**, alors la facette permettra de saisir plusieurs valeurs sous forme de cases à cocher.*
> *Si elle vaut **AND**, alors on cliquera successivement sur les valeurs en affinant au fur et à mesure.*

tag_cloud_random_display

> *Vaut **true** ou **false**.*
> *En cas de format de type **cloud**, permet d'afficher ou non les valeurs dans un ordre aléatoire (les plus fréquentes en police plus grosse que les moins fréquentes).*

tag_cloud_show_occur

> *Vaut **true** ou **false**.*
> *En cas de format de type **cloud**, définit si on affiche le nombre d'occurence de la valeur entre parenthèses.*

max_per_page

> *Pour réduire la taille d'affichage d'une facette, il y a une pagination entre les valeurs. Cet attribut permet de définir le nombre de valeurs par page au sein de la facette.*

default_open

> *Vaut **true** ou **false**.*
> *Permet d'ouvrir par défaut cette facette lorsque l'on affiche les résultats.*

order

> *Vaut **count**, **vocabulary**, **ascending**, **descending.***
> *Permet de choisir l'ordre d'affichage des valeurs au sein d'une facette :*
>
> - ***count** : affiche par nombre d'occurrences décroissant*
> - ***vocabulary** : dans le cas de l'utilisation d'un vocabulaire, on garde l'ordre du VDEX*
> - ***ascending** : tri alphabétique suivant la valeur de manière croissante*
> - ***descending** : tri alphabétique inversé suivant la valeur*

vocabularyRequest

> *Ce paramètre vaut **true** ou **false**.*
> *Permet, dans le cas d'une utilisation de vocabulaire, de ne pas générer la requête mais de la lire en tant que valeur dans le VDEX. Ceci permet de faire des requêtes complexes sur des champs croisés par exemple.*

Chaque balise <choice_field> est composée de balises **<metadata>**. Le contenu de la balise correspond au nom de la métadonnée sur laquelle rechercher.
La balise peut aussi comporter les attributs suivants :

language

> *Vaut **true** ou **false**.*
> *Le module ORI-OAI-indexing permet de gérer différentes langues pour certaines métadonnées. Dans ce cas, les métadonnées fonctionnent en trio. Pour le titre au format LOM, nous avons ces 3 champs :*
>
> - ***lom.general.title_fr** : titre saisi en français dans la fiche LOM*
> - ***lom.general.title_en** : titre saisi en anglais dans la fiche LOM*
> - ***lom.general.title** : titre dont nous n'avons pas su déterminer la langue ou que cette langue n'est pas supportée*
>
> *Dans le cas du champ **titre** du LOM, il est donc intéressant d'afficher la valeur en fonction de la langue de l'utilisateur. Dans ce cas, il faut utiliser l'attribut **language** comme ceci : **<metadata language="true">**.*
> *Si l'utilisateur a choisi l'interface en anglais, le moteur de recherche essayera successivement d'afficher **lom.general.title_en**, puis s'il n'existe pas **lom.general.title_fr** et enfin **lom.general.title** si on n'a ni la valeur en anglais, ni en français.*

<rebound> : Champs pour les rebonds depuis les résultats

Il existe la possibilité d'activer une zone de rebonds sur des termes sur la droite de l'écran :

```
<rebound id="lom">
 <tag_cloud/>
 <tag_cloud/>
</rebound>
```

id

> *Il est obligatoire de définir un identifiant par bloc **<rebound>**.*
> *Au niveau de la balise **<default_results>**, le bloc **<rebound>** sera défini en entier.*
>
> *Au niveau du menu de recherche, il suffira simplement d'appeler le bloc **<rebound>** par son identifiant comme suit:*
>
> ```
> <rebound id="lom"/>
> ```

Les balises de ce bloc se configurent de la manière suivante:

Configuration de la balise <tag_cloud>

La balise **<tag_cloud>** permet de configurer un nuage de tags dans la zone de rebond sur la droite des résultats.
Il est alors possible de cliquer sur une des valeurs. L'utilisateur est redirigé vers la recherche avancée sur un des champs pour cette valeur précise.

```
<tag_cloud menu_key="lom" search_key="advanced" field_id="doc_keyword"
max_results="24" random_display="true" show_occur="false" dynamic="true"
default_open="true"/>
<tag_cloud menu_key="lom" search_key="advanced" field_id="doc_author"
max_results="24" random_display="false" show_occur="false"/>
```

La balise se configure comme ceci :

menu_key

> *Étant donné que le nuage de tags permet de rebondir sur une recherche avancée, **menu_key** définir le menu dans lequel se trouve la recherche avancée.*

search_key

> *Ce paramètre définit la clef du sous-menu de recherche avancée à utiliser.*

field_id

> ***field_id*** *indique le champ du formulaire de recherche avancée à utiliser pour les rebonds. Les métadonnées utilisées seront celles définies pour ce champ dans le formulaire, à condition que le paramètre* ***rebound*** *soit à* ***true****.*

max_results

> *Nombre maximal de valeurs à afficher dans le nuage de tags.*

random_display

> *Permet d'afficher ou non les valeurs dans un ordre aléatoire (les plus fréquentes en police plus grosse que les moins fréquentes).*

show_occur

> *Vaut* ***true*** *ou* ***false****.*
> *Définit si on affiche le nombre d'occurence de la valeur entre parenthèses.*

dynamic

> *Vaut* ***true*** *ou* ***false****.*
> *Permet d'activer ou non le mode dynamique sur le nuage de tags. La version dynamique met les valeurs en mouvement circulaire.*

default_open

> *Vaut* ***true*** *ou* ***false****.*
> *Permet d'ouvrir par défaut cette facette lorsque l'on affiche les résultats.*

<menu> : Menus 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 authentifié.

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>
```

Pour permettre d'afficher le nombre de ressources par format, il existe les paramètres suivants dans **<search_menu>** :
default_search_key

> *Ce paramètre offre la possibilité de définir un sous-menu affiché par défaut quand on clique sur un menu dans l'interface. Si ce paramètre n'est pas utilisé, le sous-menu affiché sera le premier de la liste des sous-menus.*

count_search_key

> *Le comptage du nombre de ressources doit s'appuyer sur certains critères définis dans un menu de recherche existant (filtres, champs cachés, etc.).*
> *Pour cela, on s'appuie sur un menu existant pour y récupérer ces paramètres.*
> ***count_search_key** permet donc de spécifier ce menu de recherche.*

count_search_in_menu

> *Vaut **true** ou **false**.*
> *Permet d'afficher ou non le nombre de ressources à côté du libellé dans le menu.*

Les différents modes de recherche

<date_search> : 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.dc</metadata>
 </metadatas>

 <hidden_fields>...</hidden_fields>

 <result_fields>...</result_fields>
 <sort_fields>...</sort_fields>
 <rss_fields>...</rss_fields>
 <refine_search_fields>...</refine_search_fields>
 <rebound>...</rebound>

 <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. Voir la Section "Paramétrage des flux RSS" pour configurer cette partie.*

<refine_search_fields>

> *Permet de configurer la zone d'affinage des résultats.*

<rebound>

> *Permet de configurer la zone de rebonds.*

<authorization>

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

<thematic_search> : 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" vocabulary_language="false"
keep_navigation_session="true" full_tree="false" show_children="false"
show_children_depth="false" show_root_children_depth="false" category_to_force="0"
show_root_sublevels_results="false" show_sublevels_results="false"
show_all_results_link="true" alphabet="false" alphabet_show_all="false"
hide_nb_results="true" show_empty_categories="true"
show_rss_link_in_navigation="true" vocabularyRequest="*:*">

 <metadatas operator="VCARD_ORDER">
  <metadata vcard_att="N">lom.lifeCycle.contribute.author.name</metadata>
  <metadata vcard_att="ORG">lom.lifeCycle.contribute.author.organization</metadata>
 </metadatas>

 <hidden_fields>...</hidden_fields>

 <result_fields>...</result_fields>
 <sort_fields>...</sort_fields>
 <rss_fields>...</rss_fields>
 <refine_search_fields>...</refine_search_fields>
 <rebound>...</rebound>

 <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.*

vocabulary_language

> *Vaut **true** ou **false**.*
> *Lorsque l'on utilise un vocabulaire pour la recherche thématique, il est possible de le spécifier en fonction de la langue de l'usager.*
> *Dans ce cas, si on a **vocabulary_id="keywords"**, et que **vocabulary_language="true"**, le vocabulaire **keywords_fr** sera réellement utilisé si la langue de l'interface est en français, et **keywords_en** si c'est en anglais.*

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 authentifié, 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
vcard_att="N">lom.lifeCycle.contribute.author.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.*

show_children

> *Ce paramètre vaut **true** ou **false**.*
> *Il permet de dire si on souhaite afficher tous les sous-niveaux de l'arbre des catégories par rapport au nœud courant. Ce paramètre n'est utilisé que si "full_tree" est différent de "true".*

show_children_depth

> *Pour n'afficher qu'à une certaine profondeur les enfants.*

show_root_children_depth

> *Pour n'afficher qu'à une certaine profondeur les enfants lorsque l'on est sur la catégorie racine.*

category_to_force

> *Pour forcer l'affichage des catégories sur un niveau particulier. Par exemple **category_to_force="0"** affichera toujours la racine du vocabulaire.*

show_root_sublevels_results

> *Pour dire si on affiche les résultats des sous-niveaux lorsque l'on est sur la catégorie racine.*

show_sublevels_results

> *Pour afficher à un niveau N tous les résultats des valeurs du niveau courant et de tous les fils.*

show_all_results_link

> *Pour afficher le lien "Toutes les ressources" dans la recherche thématique.*

alphabet

> *Ce paramètre vaut **true** ou **false**.*
> *Permet d'afficher les catégories sous forme de recherche alphabétique.*

alphabet_show_all

> *Pour permettre d'afficher toutes les valeurs d'un vocabulaire alphabétique si aucune lettre n'est choisie.*

hide_nb_results

> *Ce paramètre vaut **true** ou **false**. Il permet de dire si on souhaite afficher ou non le nombre de résultats à côté de chaque catégorie.*

show_empty_categories

> *Ce paramètre vaut **true** ou **false**. Il permet de dire si on souhaite montrer à l'utilisateur les catégories ne contenant aucune ressource.*
> *Dans le cas où le paramètre vaut false, les catégories vides ne seront pas visibles dans l'IHM.*

show_rss_link_in_navigation

> *Ce paramètre vaut **true** ou **false**. Il permet de dire si on souhaite afficher un lien vers le flus RSS à côté de chaque catégorie dans une recherche thématique classique.*

vocabularyRequest

> *Ce paramètre vaut **true** ou **false**.*
> *Permet, dans le cas d'une utilisation de vocabulaire, de ne pas générer la requête mais de la lire en tant que valeur dans le VDEX. Ceci permet de faire des requêtes complexes sur des champs croisés par exemple.*

Configuration de la balise <metadatas>

operator

> *Permet de définir des règles sur l'ordre à appliquer dans le cas où on a plusieurs balises <metadata>.*
> *Si **operator="VCARD_ORDER"** alors on prendre successivement les différentes valeurs associées à chacune des **<metadata>** jusqu'à trouver une valeur correcte.*

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. Voir la Section "Paramétrage du flux RSS" pour configurer cette partie.*

<refine_search_fields>

> *Permet de configurer la zone d'affinage des résultats.*

<rebound>

> *Permet de configurer la zone de rebonds.*

<authorization>

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

<advanced_search> : Recherche avancée

Ce type de recherche permet de rechercher des documents depuis un formulaire de recherche simple ou avancée:

```
<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>
 <refine_search_fields>...</refine_search_fields>
 <rebound>...</rebound>

 <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 **advanced**. Il se configure comme décrit à la Section "Configurer un formulaire de recherche avancée".*

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 avancée. Si true, à*

> *chaque fois qu'un utilisateur reviendra dans la recherche avancée durant sa session, il retrouvera les champs du formulaire remplis tels qu'au dernier accès. Dans le cas false, il reviendra toujours à un formulaire vierge.*

<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. Voir la Section "Paramétrage du flux RSS" pour configurer cette partie.*

<refine_search_fields>

> *Permet de configurer la zone d'affinage des résultats.*

<rebound>

> *Permet de configurer la zone de rebonds.*

<authorization>

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

Paramétrage commun à tous les modes de recherche

<hidden_fields> : Champs de recherche cachés pour filtrer des résultats

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_namespaces:cdm" 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="N">lom.lifeCycle.contribute.author.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>**.
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.*

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.*

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 authentifié, 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.contribute.author.name</metadata>
 <value>{displayName}</value>
</hidden_field>
```

<result_fields> : Appel des champs affichés dans les résultats

Contrairement aux version 1.x de ORI-OAI-search, il n'est plus possible de définir complètement ce bloc dans cette partie.
Il est impératif ici d'appeler un bloc **<result_fields>** défini dans la partie **<default_results>** par son identifiant. Par exemple :

```
<result_fields id="lom"/>
```

Aussi, pour un besoin local, il est possible d'appeler un bloc **<result_fields>** défini dans la partie **<default_results>** en excluant localement certains champs. Pour cela, il faut utiliser l'attribut **show_value** comme suit :

```
<result_fields id="lom">
 <result_field key="score" show_value="false"/>
 <result_field key="fullText" display_type="fullText" show_value="false"/>
</result_fields>
```

<sort_fields> : Appel des champs de tri

Contrairement aux version 1.x de ORI-OAI-search, il n'est plus possible de définir complètement ce bloc dans cette partie.
Il est impératif ici d'appeler un bloc **<sort_fields>** défini dans la partie **<default_results>** par son identifiant. Par exemple :

```
<sort_fields id="score_title_date_author"/>
```

<rss_fields> : Appel des champs pour les flux RSS

Contrairement aux version 1.x de ORI-OAI-search, il n'est plus possible de définir complètement ce bloc dans cette partie.
Il est impératif ici d'appeler un bloc **<rss_fields>** défini dans la partie **<default_results>** par son identifiant. Par exemple :

```
<rss_fields id="notice"/>
```

<refine_search_fields> : Appel des champs pour l'affinage - facettes

Il est impératif ici d'appeler un bloc **<refine_search_fields>** défini dans la partie **<default_results>** par son identifiant. Par exemple :

```
<refine_search_fields id="lom"/>
```

<rebound> : Appel des champs pour les rebonds depuis les résultats

Il est impératif ici d'appeler un bloc **<rebound>** défini dans la partie **<default_results>** par son identifiant. Par exemple :

```
<rebound id="lom"/>
```

<authorization> : Autorisation d'accès

En mode authentifié, 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**.*

Recherche simple et focus dans les menus

Les balises concernées se trouvent dans une balise **<search_menu>** et sont les suivantes :

```
<search_menu>
 <simple_date_search/>
 <simple_search/>
 <tag_cloud/>
 <random/>

 ....
</search_menu>
```

Les modes de recherche simple au sein du menu

En plus de la recherche simple globale qui se trouve en page d'accueil ou dans le bandeau dans les autres pages, il est possible de définir une recherche simple / recherche rapide liée à un menu particulier.
Cette zone s'affichera alors sous le menu et sera liée au menu dans lequel on se trouve.

Les 2 zones de recherche simple (globale et liée à un menu) peuvent cohabiter en même temps.

<simple_search> : Champ de recherche simple accessible sur toutes les pages au sein du menu

Pour activer le champ de recherche simple lié à un menu particulier, il faut configurer comme ceci dans la balise **<search_menu>** :

```
<simple_search menu_key="lom" search_key="advanced" field_id="simple_all"
show_advanced_link="true"/>
```

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 **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*).

show_advanced_link

> *Vaut **true** ou **false**.*
> *Permet d'afficher ou non le lien vers le formulaire de recherche avancée sous le champ de recherche simple.*

<simple_date_search> : Recherche des nouveautés accessible sur toutes les pages au sein du menu

Pour activer le lien des nouveautés ou la zone de nouveautés dans le focus lié à un menu particulier, il faut configurer comme ceci dans la balise **<search_menu>** :

```
<simple_date_search menu_key="lom" search_key="news"        show_only_link="false"
max_days_period="30" max_results="3" title_field="title"/>
```

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.*

show_only_link

> *Vaut **true** ou **false**.*
> *Si **false**, alors en plus d'être un lien, ceci permettra d'afficher les nouveautés dans la zone de focus.*

max_days_period

> *Seulement si **show_only_link="false"**.*
> *Nombre de jours maximum pour lesquels il faut rechercher les nouveautés dans la zone de focus.*

max_results

> *Seulement si **show_only_link="false"**.*
> *Nombre de résultats maximum qu'il faut afficher dans la zone de focus.*

title_field

> *Seulement si **show_only_link="false"**.*
> *Clef du champ contenant le titre dans les <result_field> utilisé pour cette recherche par date pour pouvoir afficher le titre dans la zone nouveautés du focus.*

⚠ Si une zone de nouveautés dans la zone focus a été établie au niveau global dans config.xml, il sera alors par défaut présent aussi sur certaines pages des menus de recherche.
Pour le désactiver, il suffit de créer une balise vide dans **<search_menu>** :

```
<search_menu>
 <simple_date_search/>

 ....
</search_menu>
```

La zone de focus au sein du menu

<tag_cloud> : Nuage de tags au sein du menu

Via cette balise, il est possible de créer un nuage de tags dans la zone de focus :

```
<tag_cloud menu_key="lom" search_key="advanced" field_id="doc_keyword"
max_results="40" random_display="true" dynamic="true" show_occur="false"/>
```

Cette balise se configure comme ceci :

menu_key

> *Étant donné que le nuage de tags permet de rebondir sur une recherche avancée, **menu_key** définir le menu dans lequel se trouve la recherche avancée.*

search_key

> *Ce paramètre définit la clef du sous-menu de recherche avancée à utiliser.*

field_id

> ***field_id** indique le champ du formulaire de recherche avancée à utiliser pour les rebonds. Les métadonnées utilisées seront celles définies pour ce champ dans le formulaire, à condition que le paramètre **rebound** soit à **true**.*

max_results

> *Nombre maximal de valeurs à afficher dans le nuage de tags.*

random_display

> *Permet d'afficher ou non les valeurs dans un ordre aléatoire (les plus fréquentes en police plus grosse que les moins fréquentes).*

show_occur

> *Vaut **true** ou **false**.*
> *Définit si on affiche le nombre d'occurence de la valeur entre parenthèses.*

dynamic

> *Vaut **true** ou **false**.*
> *Permet d'activer ou non le mode dynamique sur le nuage de tags. La version dynamique met les valeurs en mouvement circulaire.*

⚠ Si un nuage de tags dans la zone focus a été établi au niveau global dans config.xml, il sera alors par défaut présent aussi sur certaines pages des menus de recherche.
Pour le désactiver, il suffit de créer une balise vide dans **<search_menu>** :

```
<search_menu>
 <tag_cloud/>

 ....
</search_menu>
```

<random> : Ressource aléatoire au sein du menu

Via cette balise, il est possible de mettre en avant une ressource prise aléatoirement dans la zone de focus :

```
<random menu_key="lom" search_key="advanced"/>
```

Cette balise se configure comme ceci :

menu_key

> *La recherche d'une ressource aléatoire doit se baser sur les contraintes de recherche d'un menu de recherche déjà défini. Il reprend les contraintes des **<hidden_fields>** et les champs à afficher des **<result_fields>**.*
> ***menu_key** défini le menu de recherche où prendre ces paramètres.*

search_key

> ***search_key*** *défini le sous-menu de recherche où prendre ces paramètres.*

⚠ Si un bloc de ressource aléatoire dans la zone focus a été établi au niveau global dans config.xml, il sera alors par défaut présent aussi sur certaines pages des menus de recherche.
Pour le désactiver, il suffit de créer une balise vide dans **<search_menu>** :

```
<search_menu>
 <random/>

 ....
</search_menu>
```

<cart_result_fields> : Champs de résultat du panier

Si le panier est activé, il est possible de définir des particularités sur des champs affichés ou d'ajouter des champs pour des exports dans différents formats.
Cette zone se définit par ces balises :

```
<cart_result_fields>
 <result_fields>
  ...
 </result_fields>
 <result_fields>
  ...
 </result_fields>
 <result_fields>
  ...
 </result_fields>
</cart_result_fields>
```

La balise **<cart_result_fields>** se compose de différentes balises **<result_fields>** définies comme suit :

```
<cart_result_fields>
 <result_fields namespace="vocabulary:search_metadata_namespaces:lom"
result_fields_id="lom">
  <result_field key="score" show_value="false"/>
  <result_field key="fullText" display_type="fullText" show_value="false"/>

  <export_result_field exportFormat="csv" key="repository"
format="vocabulary:search_repositories"/>

  <export_result_field key="ris" exportFormat="ris">
   <metadata language="true">lom.ris</metadata>
  </export_result_field>
 </result_fields>
</cart_result_fields>
```

namespace

> *Permet de définir le namespace du format de métadonnées concerné par ce bloc.*

result_fields_id

> *Correspond à l'identifiant unique du* ***<result_fields>****.*

Un bloc **<result_fields>** défini ici fait référence aux blocs **<result_fields>** définis dans la partie **<default_results>**.
En effet, c'est le même bloc de définition qui sera utilisé pour l'affichage des résultats, mais étant donné que l'on n'est pas dans une liste de résultats mais dans l'affichage d'un résultat en dehors d'une requête particulière, il est nécessaire de *désactiver* certains champs. Pour cela, on utilise une balise **<result_field>** avec la même clef que celle définie dans le bloc **<default_results>** en ajoutant l'attribut **show_value="false"**. Ainsi, l'attribut en question ne sera pas montré dans la liste des ressources du panier. Par exemple, afficher le score n'aurait aucun sens au sein du panier :

```
<result_field key="score" show_value="false"/>
```

Enfin, la fonctionnalité de panier permet aussi l'export des résultats présents dans le panier dans différents formats (pour le moment RIS et CSV).
Pour cela, on définit des balises **<export_result_field>**.
Pour définir la métadonnée qui contient le format d'export RIS pour chaque résultat, on définit comme ceci :

```
<export_result_field key="ris" exportFormat="ris">
 <metadata language="true">lom.ris</metadata>
</export_result_field>
```

La métadonnée **lom.ris** contient dans notre exemple une version RIS des métadonnées. Ce champ sera alors utilisé pour exporter tous les résultats du panier au format RIS.

Pour finir, il est aussi possible d'exporter les résultats au format CSV. Dans ce cas, tous les champs définis dans les **<result_field>** seront utilisés. Mais pour certains besoins, il est nécessaire de modifier la configuration d'un ou plusieurs champs pour l'export.
Par exemple, le champ **repository** utilise par défaut un vocabulaire qui transforme la valeur en une image pour afficher le logo à l'écran. Mais dans le cas d'un export CSV, on souhaite avoir un libellé et non une image pour ce champ.
Il est donc possible de *surcharger* la définition du champ par défaut en utilisant ici un autre vocabulaire qui ne contient plus des images, mais des libellés pour chaque entrepôt :

```
<export_result_field exportFormat="csv" key="repository"
format="vocabulary:search_repositories"/>
```

Zone de recherche simple et focus

Les modes de recherche simple

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

<simple_search> : 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" show_advanced_link="true"/>
```

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 **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*).

show_advanced_link

> *Vaut **true** ou **false**.*
> *Permet d'afficher ou non le lien vers le formulaire de recherche avancée sous le champ de recherche simple.*

Voici un exemple de formulaire généré :

```
<form id="simple-search-form" name="simple_search_form" method="get"
action="http://[HOST_SEARCH]:[PORT_SEARCH]/[CONTEXT_SEARCH]/simple-search.html">
 <input type="hidden" class="input-hidden" name="menuKey" value="menu_test"/>
 <input type="hidden" class="input-hidden" name="submenuKey"
value="advanced_test"/>
 <input type="hidden" class="input-hidden" name="fieldId" value="field_test"/>
 <input class="input-text input-text-search simple-search-form-query"
name="light-request" size="25" type="text" title="Recherche simple"/>
 <input value="Rechercher" class="simple-search-form-submit input-button"
type="submit">
</form>
```

<simple_date_search> : 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 est en fait un pointeur vers le formulaire de recherche par date que vous aurez désigné, mais sert aussi à afficher les nouveautés dans la zone de focus.

Pour cela, vous devez renseigner les paramètres suivants:

```
<simple_date_search menu_key="menu_test" search_key="date_test"
show_link="true" show_only_link="false" max_days_period="30" max_results="3"
title_field="title"/>
```

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.*

show_link

> *Vaut **true** ou **false**.*
> *Si **true**, alors le lien vers les nouveautés sera affiché dessous le champ de recherche simple.*

show_only_link

> *Vaut **true** ou **false**.*
> *Si **false**, alors en plus d'être un lien, ceci permettra d'afficher les nouveautés dans la zone de focus.*

max_days_period

> *Seulement si **show_only_link="false"**.*
> *Nombre de jours maximum pour lesquels il faut rechercher les nouveautés dans la zone de focus.*

max_results

> *Seulement si **show_only_link="false"**.*
> *Nombre de résultats maximum qu'il faut afficher dans la zone de focus.*

title_field

> *Seulement si **show_only_link="false"**.*
> *Clef du champ contenant le titre dans les <result_field> utilisé pour cette recherche par date pour pouvoir afficher le titre dans la zone nouveautés du focus.*

La zone de focus

<tag_cloud> : Nuage de tags

Via cette balise, il est possible de créer un nuage de tags dans la zone de focus :

```
<tag_cloud menu_key="lom" search_key="advanced" field_id="doc_keyword"
max_results="40" random_display="true" dynamic="false" show_occur="false"/>
```

Cette balise se configure comme ceci :

menu_key

> *Étant donné que le nuage de tags permet de rebondir sur une recherche avancée, **menu_key** définir le menu dans lequel se trouve la recherche avancée.*

search_key

> *Ce paramètre définit la clef du sous-menu de recherche avancée à utiliser.*

field_id

> ***field_id** indique le champ du formulaire de recherche avancée à utiliser pour les rebonds. Les métadonnées utilisées seront celles définies pour ce champ dans le formulaire, à condition que le paramètre **rebound** soit à **true**.*

max_results

> *Nombre maximal de valeurs à afficher dans le nuage de tags.*

random_display

> *Permet d'afficher ou non les valeurs dans un ordre aléatoire (les plus fréquentes en police plus grosse que les moins fréquentes).*

show_occur

> *Vaut **true** ou **false**.*
> *Définit si on affiche le nombre d'occurence de la valeur entre parenthèses.*

dynamic

> *Vaut **true** ou **false**.*
> *Permet d'activer ou non le mode dynamique sur le nuage de tags. La version dynamique met les valeurs en mouvement circulaire.*

<random> : Ressource aléatoire

Via cette balise, il est possible de mettre en avant une ressource prise aléatoirement dans la zone de focus :

```
<random menu_key="all" search_key="advanced" rotate_vocabulary_ids="true" >
 <hidden_field vocabularyId="indexed_repositories">
  <metadata>md-ori-oai-repository.dc</metadata>
  <metadata>md-ori-oai-repository.lom</metadata>
  <metadata>md-ori-oai-repository.tef</metadata>
  <metadata>md-ori-oai-repository.dcfr</metadata>
  <metadata>md-ori-oai-repository.cdm</metadata>
 </hidden_field>
</random>
```

Cette balise se configure comme ceci :

menu_key

> *La recherche d'une ressource aléatoire doit se baser sur les contraintes de recherche d'un menu de recherche déjà défini. Il reprend les contraintes des **<hidden_fields>** et les champs à afficher des **<result_fields>**.*
> ***menu_key** défini le menu de recherche où prendre ces paramètres.*

search_key

> ***search_key** défini le sous-menu de recherche où prendre ces paramètres.*

rotate_vocabulary_ids

> *La balise **<random>** permet de sélectionner aléatoirement une ressource dans l'index. Cette ressource est mise en cache (cf. paramétrage du temps de cache) et sera affichée à tous les utilisateurs jusqu'à ce que le cache expire.*
> *Mais dans certains cas, il est intéressant de changer de ressource après expiration du cache en fonction de certains critères.*
> *Par exemple : le lundi afficher une ressource aléatoire au format LOM, le mardi au format TEF, le mercredi au format CDM, etc.*

L'exemple ci-dessous décrit comment changer l'entrepôt d'où provient la ressource (au format LOM ou TEF) à chaque fois que le cache expire :

```
<random menu_key="all" search_key="advanced"
rotate_vocabulary_ids="true" >
 <hidden_field vocabularyId="indexed_repositories">
  <metadata>md-ori-oai-repository.lom</metadata>
  <metadata>md-ori-oai-repository.tef</metadata>
 </hidden_field>
</random>
```

Dans ce cas, on définit une balise ***<hidden_field>*** *avec un attribut* ***vocabularyId****. Cet attribut contient l'identifiant du vocabulaire contenant la liste des valeurs d'entrepôts sur lesquels il faut tourner.*
Ensuite, la balise ***<hidden_field>*** *contient un ensemble de balises* ***<metadata>*** *contenant les métadonnées sur lesquelles rechercher.*

Ainsi, en imaginant que le vocabulaire contienne les valeurs ***entrepôt_1****,* ***entrepôt_2*** *et* ***entrepôt_3****, nous aurons les affichages suivants :*

- ***Jour 1*** *: ressource prise aléatoirement dans tout l'index avec la contrainte* ***md-ori-oai-repository.lom=entrepôt_1 OU md-ori-oai-repository.tef=entrepôt_1***
- ***Jour 2*** *: ressource prise aléatoirement dans tout l'index avec la contrainte* ***md-ori-oai-repository.lom=entrepôt_2 OU md-ori-oai-repository.tef=entrepôt_2***
- ***Jour 3*** *: ressource prise aléatoirement dans tout l'index avec la contrainte* ***md-ori-oai-repository.lom=entrepôt_3 OU md-ori-oai-repository.tef=entrepôt_3***
- ***Jour 4*** *: ressource prise aléatoirement dans tout l'index avec la contrainte* ***md-ori-oai-repository.lom=entrepôt_1 OU md-ori-oai-repository.tef=entrepôt_1***
- etc.

<open_search> : 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 **config.xml** :

```
<open_search>
 <link title="Recherche de ressources pédagogiques" file="simple_lom.xml"/>
 <link title="Recherche de thèses" file="simple_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 ***opensearch****.*

La configuration d'un plugin (dans le dossier opensearch) est vue plus bas dans la partie **Dossier "opensearch" : définition des plugins Open search pour les navigateurs**

<notice_formats> : Affichage de la fiche de métadonnées (notice)

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 formatsVocabulary="search_metadata_namespaces">
 <format formatMetadataValue="vocabulary:search_metadata_namespaces:lom"
prefix="lom" xsl="lom.xsl" headXsl="lom.xsl" showXmlLink="true"
showThumbnail="true">
  <metadata format="...">...</metadata>
 </format>

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

formatsVocabulary

> *Spécifie le vocabulaire qui contient la liste de tous les formats de métadonnées supportés.*

Le principe est de définir une balise **<format>** par format à supporter. Pour chacune de ces balises, on créé des sous-balises **<metadata>** p our chaque **métadonnée de gestion** que l'on veut transformer avant son affichage. C'est ici un principe semblable au formats que l'on définit dans les champs de résultats **<result_field>**.

Avant la v2, on transformait ici les champs de gestion (namespace, datestamp, repository, etc.) et les métadonnées à l'intérieur des fiches. Depuis la v2, les métadonnées dans la fiche sont traitées en amont dans le module ORI-OAI-indexing. Seules les métadonnées de gestion sont encore gérées dans le module ORI-OAI-search.

<format> : 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 *e st 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 les sources dans le dossier **[ORI_HOME]/src/ori-oai-search-svn/src/mai n/webapp/WEB-INF/xsl** ou dans une configuration personnalisée dans **[PATH_CUSTOM_CONFIG]/ori-oai-search/contri bs-search/votre_contrib/xsl**. Voir la Section "Transformation XSL des fiches de métadonnées" pour la configuration de ce fichier 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 **de scription**. Le fichier donné en configuration doit obligatoirement se trouver dans le dossier **[ORI_HOME]/src/ori-oai-searc h-svn/src/main/webapp/WEB-INF/xsl/head** ou dans une configuration personnalisée dans **[PATH_CUSTOM_CONFIG]/o ri-oai-search/contribs-search/votre_contrib/xsl/head**.*

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**.*

showThumbnail

> *Permet de dire si on souhaite ou non afficher la vignette d'aperçu du document dans la notice. **true** pour une visualisation, **false** sinon. La valeur par défaut est **true**.*

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.*

Chaque balise **<metadata>** permet de configurer une transformation d'une métadonnée de gestion avant affichage. Les attributs et configurations possibles sont les suivants:
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.*

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 <metadata>

> *En dehors d'un ajout de lien vers la recherche thématique, le contenu est le nom de la métadonnée (md-ori-oai-repository, md-ori-oai-namespace, 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>**.*
>
> 
> *Dans ce cas, le contenu de la balise **<metadata>** est le Xpath correspondant à la donnée permettant le rebond vers une recherche thématique.*
> *La condition pour que ce rebond fonctionne est que la même métadonnée, avec le même Xpath ait subi une transformation dans le module ORI-OAI-indexing utilisant le même vocabulaire que celui utilisé par la recherche thématique vers laquelle on veut rebondir. Si la transformation n'est pas faite dans l'indexing ou si le vocabulaire n'est pas le même, ça ne fonctionnera pas.*
>
> *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 **orioai-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.*

Dossier "advanced" : configurer un formulaire de recherche avancée

Un formulaire de recherche avancée se configure comme dans l'exemple ci-dessous (eette recherche se configure dans le dossier **advanced**) :

```
<form>

 <group id="group_id1">

  <field id="field_id1" format="text" defaultValue="value_1" readOnly="true"
hidden="false">
   <metadata language="true" tagName="..._tag" autocomplete="true"
rebound="true">...</metadata>
   <metadata>...</metadata>
  </field>
  <field id="field_id2" format="text" vocabularyId="vocabulary_1" type="radio">
   <metadata>...</metadata>
   <hidden_voc_id>voc_id_1</hidden_voc_id>
  </field>
  <field id="field_id3" format="text" thematicMenuKey="menu_test"
thematicSearchKey="thematic_test">
   <metadata>...</metadata>

   <condition>...</condition>
  </field>
  <field id="field_id4" format="date:dd-MM-yyyy">
   <metadata dateFormat="yyyyMMdd">...</metadata>
  </field>

  <fields id="fields_id" maxChoiceSize="5">
   <field format="text" id="field_id5">
    <metadata>...</metadata>
   </field>
   <field format="text" id="field_id6">
    <metadata>...</metadata>
   </field>
  </fields>

  <condition>...</condition>
 </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 **<fields>** ou **<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 (**<fields>** ou **<field>**).

<fields>

Un **fields** correspond à un groupement de champs de recherche du formulaire. Ceci représente une interface de recherche experte dans laquelle on pourra croiser des recherches à partir des booléens **ET**, **OU** et **SAUF**. Une balise **field** contient plusieurs sous-balises **field**. Les attributs suivants sont à configurer :

id

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

maxChoiceSize

> *Ce champ permet de spécifier le nombre de croisements que l'on peut faire dans les interfaces. Ceci affichera un certain nombre de fois (valeur fixe) la répétition des champs.*

<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 (ou d'autres formats) 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.*

type

> *Cet attribut permet de dire quel type de champ on veut utiliser lors de saisie de valeur(s) parmi une liste fermée provenant d'un vocabulaire. Les valeurs possibles sont:*
>
> - ***select** permet de saisir plusieurs valeurs dans une liste déroulante. Il faut spécifier le nombre de valeurs que l'on souhaite afficher sans avoir à dérouler la liste. Par exemple"select:4".*
> - ***checkbox** permet la saisie de plusieurs valeurs dans des cases à cocher*
> - ***radio** pour la saisie d'une seule valeur dans une liste de type "radio"*

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.*

selector

> *Ce paramètre prend la valeur **true** ou **false** et permet de dire si on souhaite afficher le sélecteur lorsque le champ est de type date. La valeur est **true** par défaut.*

fullDepth

> *Vaut **true** ou **false**.*
> *Permet d'afficher un arbre complet de vocabulaire dans le cas où le champ utilise un vocabulaire.*

boost

> *Permet de spécifier un boost sur le champ pour permettre de booster un champ par rapport à un autre en recherche avancée.*

<metadata>

Chaque **<field>** doit contenir une ou plusieurs balises **<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.

language

> *Vaut **true** ou **false**.*
> *Le module ORI-OAI-indexing permet de gérer différentes langues pour certaines métadonnées. Dans ce cas, les métadonnées fonctionnent en trio. Pour le titre au format LOM, nous avons ces 3 champs :*
>
> - ***lom.general.title_fr** : titre saisi en français dans la fiche LOM*
> - ***lom.general.title_en** : titre saisi en anglais dans la fiche LOM*
> - ***lom.general.title** : titre dont nous n'avons pas su déterminer la langue ou que cette langue n'est pas supportée*
>
> *Dans le cas du champ **titre** du LOM, il est donc intéressant d'afficher la valeur en fonction de la langue de l'utilisateur. Dans ce cas, il faut utiliser l'attribut **language** comme ceci : **<metadata language="true">**.*
> *Si l'utilisateur a choisi l'interface en anglais, le moteur de recherche essayera successivement d'afficher **lom.general.title_en**, puis s'il n'existe pas **lom.general.title_fr** et enfin **lom.general.title** si on n'a ni la valeur en anglais, ni en français.*

autocomplete

> *Vaut **true** ou **false**.*
> *Permet d'activer ou non l'autocomplétion sur la saisie du champ. Dans ce cas dès que l'utilisateur commence à saisir un mot, l'interface propose la liste de tous les mots commençant par ce terme.*

tagName

> *Le module ORI-OAI-indexing s'appuie sur le moteur Solr pour l'indexation et la recherche de documents. Solr permet de définir des règles utilisées lors de l'indexation.*
> *Pour être pertinent dans la recherche, Solr effectue une transformation des valeurs au moment de l'indexation et de la recherche. Par exemple, il supprime les mots vides (le, la, les, dans, mon, ma, etc.), il tronque les verbes conjugués, les pluriels, etc.*
> *C'est uniquement la valeur **indexée** qui est transformée de la sorte. Solr peut aussi stocker la valeur initiale de la métadonnée pour permettre l'affichage de celle-ci.*
>
> *Un incovénient à ce système est que lorsque l'on veut récupérer des tags de l'index pour afficher un nuage de tags ou pour par exemple la fonctionnalité d'autocomplétion, nous sommes obligés de nous appuyer sur la valeur **indexée** et non pas la valeur **stockée**.*
> *Mais la valeur indexée ne convient pas car les mots ont été tronqués !*
>
> *Il est donc nécessaire de dupliquer le champ au niveau de la configuration de Solr (schema.xml) pour avoir une version du champ donc la valeur indexée n'a pas été tronquée. Ceci se fait facilement par des **copyField** dans Solr, mais la configuration doit se faire manuellement.*
>
> *Par convention, ces champs dupliqués sont nommés de la même manière que le champ d'origine avec le suffixe **_tag**.*
>
> *Par exemple, la version non tronquée de la métadonnée **dc.title** est **dc.title_tag**.*
>
> *L'attribut **tagName** sert donc à indiquer le nom de la métadonnée non tronquée sur laquelle il faut s'appuyer pour récupérer la liste de tags.*

rebound

> *Vaut **true** ou **false**.*
> *Lorsque le **<field>** est utilisé pour un rebond depuis la liste des résultats, il est possible d'utiliser ou non certaines métadonnées pour la recherche de tags. Ce paramètre sert à dire si oui on non on utilise la métadonnée.*

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"****.*

boost

> *Permet de spécifier un boost sur la métadonnée pour permettre de booster une métadonnée par rapport à une autre dans un champ en recherche avancée.*

<hidden_voc_id>

Chaque **field** peut contenir une ou plusieurs balise **hidden_voc_id** lorsque l'on est sur un champ de saisie basé sur un vocabulaire donné. Chaque hidden_voc_id permet de définir l'identifiant d'une entrée du vocabulaire sur laquelle on ne souhaite pas faire de recherche. Ainsi, l'entrée du vocabulaire ne sera pas proposée lors de la saisie dans l'interface de recherche.

<condition>

Chaque **<group>** ou **<field>** peut contenir une ou plusieurs balise **<condition>**. Cette balise permet de mettre une condition sur l'affichage de certains champs. On souhaitera par exemple n'afficher que les champs spécifiques au format LOM/LOMFR/SupLOMFR qu'à la condition d'avoir coché la case "Ressource pédagogique".
Cette fonctionnalité est extensible à tous les champs et à toutes les valeurs. Dès que l'utilisateur respecte la condition, les champs s'affichent ou disparaissent.

fieldId

> *Correspond à l'identifiant sur lequel on met la condition. Le* ***<group>*** *ou le* ***<field>*** *ne s'affichera ou ne disparaitra qu'à la condition que ce champ prenne une valeur donnée.*

value

> *Ceci est la valeur que doit prendre le champ défini par* ***fieldId*** *pour qu'une action se fasse.*

action

> *Ce champ peut prendre 2 valeurs:* ***show*** *ou* ***hide*** *suivant que l'on veuille montrer ou masquer le* ***field*** *ou le* ***group*** *quand la condition est respectée.*

<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>
```

Dossier "i18n" : personnalisation des messages et libellés

Tous les messages et libellés de la configuration de recherche personnalisée sont stockés dans des bundles i18n dans le dossier **i18n**. Il y a 3 différents fichiers: **menus_XX.properties**, **forms_XX.properties** et **custom_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.

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 **config.xml**, **à modifier** en fonction du menu que l'on configure
- *submenuKey* la clef d'un formulaire de recherche **à modifier**
- *resultFieldsId* l'identifiant du **<result_fields>** défini dans **config.xml 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
menu.label.menuKey.label.count=libellé du format utilisé lorsque l'on affiche le
nombre de ressources par format sur la page d'accueil
menu.label.menuKey.label.random=Titre de la zone de focus d'affichage de ressource
aléatoire
```

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.resultFieldsId.resultFieldKey=Libellé du champ
menu.result.resultFieldsId.resultFieldKey.description=Description 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 **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
```

Pour les champs que l'on utilise pour faire du rebond, les libellés utilisés dans la zone de rebond sont les suivants :

```
menu.form.rebound.label.menuKey.submenuKey.fieldId=Sur le champ
```

custom_XX.properties

Tous les libellés propres à la configuration personnalisée sont définis dans ces fichiers. Il est possible de définir des nouveaux bundles, ou de surcharger également des bundles de message définis par défaut dans ORI-OAI. Pour cela, il faut les définir dans les fichiers **custom_XX.properties**. Ces messages seront donc prioritaires sur ceux définis par défaut.

Mais pour les surcharger, il est nécessaire de savoir où ils sont stockés par défaut dans l'application :

- [ORI_HOME]/src/ori-oai-search-svn/src/main/resources/properties/i18n/**messages_fr.properties** et **messages_en.properties** : libellés génériques, ... ;
- [ORI_HOME]/src/ori-oai-search-svn/src/main/resources/properties/i18n/**errors_fr.properties** et **errors_en.properties** : messages d'erreur ;
- [ORI_HOME]/src/ori-oai-search-svn/src/main/resources/properties/i18n/**results_fr.properties** et **results_en.properties** : messages servant pour les résultats ;
- [ORI_HOME]/src/ori-oai-search-svn/src/main/resources/properties/i18n/**xsl_fr.properties** et **xsl_en.properties** : libellés utilisés dans les transformations XSL.

Si vous avez besoin de modifier un message, il faut donc copier/coller sa clef depuis le fichier d'origine vers les fichiers **custom_XX.properties** et modifier le libellé associé au message.

Dossier "opensearch" : définition des plugins Open search pour les navigateurs

Voyons maintenant la configuration d'un plugin OpenSearch compatible Firefox 2 et Internet Explorer 7 (Vous pouvez avoir 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.).
Ces fichiers sont stockés dans le dossier **opensearch** de votre configuration personnalisée.

```
<?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?me
nuKey=menu_test&amp;submenuKey=advanced_test&amp;fieldId=field_test&amp;light-reque
st={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?m
enuKey=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 **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.

Dossier "jsp" : surcharge des pages HTML générées

Dans certains cas, il est nécessaire de modifier les JSP utilisées lors de la génération des pages HTML du moteur de recherche pour un besoin propre à la configuration établie. Les JSP par défaut peuvent donc être surchargées en les copiant depuis le dossier **[ORI_HOME]/src/ori-oai-search-svn/src/main/webapp/WEB-INF/jsp** vers le dossier **jsp** de votre configuration. Attention l'arborescence des fichiers surchargés doit rester la même entre [ORI_HOME]/src/ori-oai-search-svn/src/main/webapp/WEB-INF/jsp et jsp.

Dans le cas où les configurations nécessitent une personnalisation des pages, il faudra surcharger les JSP de génération des interfaces qui se trouvent dans le dossier **jsp** de chaque contribution de recherche.

Dans le cas où vous souhaitez pousser la modification d'autres JSP il vous suffit de surcharger les JSP d'origine en les écrasant depuis le dossier **jsp** de la contribution de recherche sélectionnée. Dans ces cas là, faites un copier/coller de la JSP en respectant bien la même structure de dossier dans le dossier search.
Par exemple, pour surcharger le comportement de la JSP **generic-results.jsp**:

- Copiez le fichier [ORI_HOME]/src/ori-oai-search-svn/src/main/webapp/WEB-INF/jsp/**generic-results.jsp** vers [PATH_CUSTOM_CONFIG]/ori-oai-search/contribs-search/*votre_contrib*/jsp/results/**generic-results.jsp**
- Modifiez ce nouveau fichier qui surchargera le fichier d'origine lors du déploiement

Dossier "xsl": transformation XSL des fiches de métadonnées

Afin de personnaliser vos fichiers XSL servant à l'affichage des notices, ou ajouter de nouveaux formats de métadonnées, il est recommandé de regarder les formats fournis avec l'application.

Il est possible de modifier les XSL fournies en les surchargeant dans ce dossier. Les XSL par défaut sont stockées dans le dossier [ORI_HOME]/src/ori-oai-search-svn/src/main/webapp/WEB-INF/xsl.
Ce dossier se découpe comme suit :

- **dc.xsl** : XSL servant à afficher une notice au format Dublin Core
- **lom.xsl** : XSL servant à afficher une notice au format LOM
- **tef.xsl** : XSL servant à afficher une notice au format TEF
- **dcfr.xsl** : XSL servant à afficher les publications
- **head** : dossier contenant les XSL (une par format de métadonnées) servant à générer les en-têtes HTML lors de l'affichage de la notice
- **title** : dossier contenant les XSL (une par format de métadonnées) servant à générer les titres lors de l'affichage de la notice
- **templates** : templates XSL génériques appelés dans les différentes XSL

Transformation des fiches dans ORI-OAI-indexing

La "traduction" des fiches en fonction des langues au niveau du module ORI-OAI-indexing nécessite une petite explication. En effet, lors de la transformation des fiches dans les différentes langues au niveau du module ORI-OAI-indexing, la fiche XML est modifiée pour voir apparaître toutes les informations de "traduction" des champs en fonction d'un vocabulaire.

Prenons par exemple la métadonnées lom:general/lom:keyword du format LOM. Nativement, cette métadonnée ressemble à ceci :

```
<keyword>
    <string language="fre">climat</string>
</keyword>
```

Une fois "transformée", et les liens faits avec le vocabulaire "**keywords_fr**", cette métadonnée ressemble à ceci :

```
<keyword>
    <string language="fre"
                orioai-vocabularyId="keywords_fr"
                orioai-termId="climat"
                orioai-label="Climat"

orioai-termFullLabel="C&amp;#160;&amp;#160;&amp;#187;&amp;#160;&amp;#160;Climat"
                orioai-termFullLabelClean="C; Climat"
    >climat</string>
</keyword>
```

où:
orioai-vocabularyId

> *l'identifiant du vocabulaire utilisé pour la traduction*

orioai-termId

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

orioai-label

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

orioai-termFullLabel

> *le libellé traduit correspondant au chemin complet vers la valeur trouvée dans le vocabulaire en fonction de la langue de l'utilisateur (encodage HTML)*

orioai-termFullLabelClean

> *le libellé traduit correspondant au chemin complet vers la valeur trouvée dans le vocabulaire en fonction de la langue de l'utilisateur*

Rebond vers une recherche thématique

La construction des liens vers des recherches thématiques nécessite aussi une petite explication. En effet, une fois la transformation des fiches faites dans les différentes langues au niveau du module ORI-OAI-indexing, il est possible de définir des rebonds vers une recherche thématique à partir de la valeur retrouvée dans un vocabulaire. Une fois le rebond calculé, la fiche XML est modifiée pour voir apparaître

tous les paramètres nécessaires à la construction du lien.

Pour définir ces rebonds, il faut par exemple créer le bloc suivant dans le fichier **config.xml** :

```
<metadata>
    //lom:general/lom:keyword/lom:string
    <thematic_link key="lom_keywords" thematicMenuKey="lom"
thematicSearchKey="keywords" showAbsoluteLink="false"/>
</metadata>
```

Dans ce cas, un lien sera fait vers la recherche thématique du menu **lom** et sous-menu **keywords** (uniquement dans le cas où la transformation de la fiche et le menu **lom/*keywords** s'appuient sur le même vocabulaire !

Dans notre exemple précédent, le XML avant transformation XSLT deviendrait alors:

```
<keyword>
    <string language="fre"
                orioai-vocabularyId="keywords_fr"
                orioai-termId="climat"
                orioai-label="Climat"

orioai-termFullLabel="C&amp;#160;&amp;#160;&amp;#187;&amp;#160;&amp;#160;Climat"
                orioai-termFullLabelClean="C; Climat"

                orioai-link-key="lom_keywords"
                orioai-menuKey="lom"
                orioai-submenuKey="keywords"
    >climat</string>
</keyword>
```

où:
orioai-link-key

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

orioai-menuKey

> *valeur de thematicMenuKey de la configuration*

orioai-submenuKey

> *valeur de thematicSearchKey*

Enfin, l'utilisation du template

```
<xsl:call-template name="print_template">
```

dans la XSL automatise la génération du lien.

Un exemple de lien généré est alors le suivant:

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

Définitions de nouveaux libellés à appeler dans la XSL

Dans le cas où il est nécessaire de définir de nouveaux messages, la première étape est de créer les messages dans le dossier **i18n**. Au sein des fichiers **custom_fr.properties** et **custom_en.properties**, ajouter par exemple la clef suivante :

```
xsl.ma_clef_de_message=Mon nouveau message
```

Ensuite, il est nécessaire de faire transiter ce message jusqu'à la XSL. Pour cela, il faudra surcharger le fichier **notice-content-i18n-custom .jsp** depuis **[ORI_HOME]/src/ori-oai-search-svn/src/main/webapp/WEB-INF/jsp/notice** vers le dossier **jsp** de la contribution.
Dans ce fichier, il faut alors définir le passage de paramètres au transformateur XSLT :

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

Enfin, dans la XSL concernée, il faut définir ce message comme un paramètre d'entrée :

```
<xsl:param name="xsl.ma_clef_de_message"/>
```

Dossiers "ori-oai-indexing" et "ori-oai-vocabulary" : dans le cas de partage des configurations

Dossier ori-oai-indexing
Votre configuration de recherche peut nécessiter une configuration spécifique du module ori-oai-indexing (configIndexing.xml, fieldsConfig.xml, etc.). Ainsi, les établissements utilisant votre configuration pourront déployer MANUELLEMENT ces fichiers de configuration dans leur instance locale. Aucun fichier contenu dans le dossier ori-oai-indexing n'est déployé ou utilisé en l'état. Ce dossier sert uniquement au bon échange des contributions dans la communauté.

Dossier ori-oai-vocabulary
Votre configuration de recherche peut nécessiter l'utilisation de vocabulaires spécifiques. Dans ce cas, nous conseillons de stocker dans ce dossier les vocabulaires statiques et les configurations de vocabulaires dynamiques nécessaires au bon fonctionnement du moteur de recherche. Ainsi, les établissements utilisant votre configuration pourront déployer MANUELLEMENT ces vocabulaires dans leur instance locale. Aucun fichier contenu dans le dossier ori-oai-vocabulary n'est déployé ou utilisé en l'état. Ce dossier sert uniquement au bon échange des contributions dans la communauté.