Version 1.5

ORI-OAI-md-editor : Interface de saisie des métadonnées

> ⚠ **Module optionnel**

Voir l'architecture du système

Dans quels cas l'utiliser

Pour la saisie des fiches de métadonnées

Composants fortement recommandés dans ORI-OAI

- **ORI-OAI-vocabulary** pour fournir les vocabulaires lors de la saisie des métadonnées
  --> note: ce module peut aussi être utilisé seul sans la connexion au module ORI-OAI-vocabulary. Ce cas se présentera plutôt dans le cadre de tests, de démonstration, ou pour utiliser un éditeur de métadonnées en dehors de ORI-OAI

Description

Ce composant est utilisé pour la saisie des métadonnées dans ORI-OAI. Les formulaires de saisie des métadonnées sont entièrement configurables. La technologie XForms associée au moteur de transformation Orbeon offre des formulaires de saisie dynamiques en fonction de fichiers XML XForms.

Le support des différents formats de métadonnées repose sur la possibilité d'enrichir le module ORI-OAI-md-editor de nouvelles configurations XForms.

Ce module permet une saisie des métadonnées très riche grâce notamment à de l'aide à la saisie, de l'auto complétion, de la recherche de personnes dans l'annuaire, etc.

Couplé au module ORI-OAI-workflow, il permet de proposer un formulaire de saisie des métadonnées à n'importe quelle étape du workflow. Utilisé seul, il permet l'édition de fiches de métadonnées en dehors du système. Il répond alors au besoin d'un éditeur simple et puissant de métadonnées au format XML.

Voir la documentation technique

Spécifications

Choix techniques

Choix techniques

Formulaire d'édition des MD : XForms + Orbeon Forms

Les formulaires d'édition proposés à l'utilisateur dans ORI-OAI-Workflow correspondent à une partie très importante de ORI-OAI-Workflow dans le sens où c'est la partie la plus visible de ORI-OAI-Workflow pour l'utilisateur final. Ces formulaires peuvent se décliner selon les différents schémas (DC, LOM, LOM-FR, ...), selon les différents rôles des différents utilisateurs. Très souples, ils acceptent un certain nombre de paramètres qui diffèrent selon les usages. On pense notamment aux taxonomies/vocabulaires utilisés lors de la saisie via des widgets à choix multiple. Enfin ils sont ergonomiques et permettent une saisie efficace des fiches XML de métadonnées : auto-complétion, ajout/suppresion instantané de champs, bref les forumlaires sont des formulaires dynamiques (Ajax).

Le choix a été d'opter pour un standard W3C XForms qui permet de créer des formulaires ergonomiques, dynamiques, ... cela via un langage XML. Vu que XForms n'est pas supporté nativement par les navigateurs et que les implémentations de plugins sur les navigateurs ne sont pas encore tout à fait au point, et enfin pour des questions de souplesse, la solution proposée par Orbeon, Orbeon Forms a été

retenue. Celle-ci permet entre autre de générer à partir de formulaires XForms une interface Html/Ajax interprétable par les principaux navigateurs clients du marché.

Orbeon est à l'image de Cocoon un framework complet de développement qui tire parti, de manière extrêmement ingénieuse, de la technique de transformation XML pour concevoir des applications complexes de manière efficace. On a choisi de retenir Orbeon Forms pour la saisie des métadonnées et donc pour sa qualité première et spécifique de technologie de formulaires.

Conclusion

Le module ORI-OAI-MD-Editor correspond à un XForms interprété par Orbeon Forms pour générer une interface Ajax à l'utilisateur. Il ne contient pas de logique applicative. Il recoit un XML et en renvoie un. Pour ce faire un protocole de communication spécifique (s'appuyant sur des Web Services SOAP) est cependant mis au point pour permettre à Orbeon et ori-oai-workflow (spring) d'interagir ensemble.

Implémentation

Implémentation

ori-oai-md-editor

On distingue 2 parties dans l'implémentation de ori-oai-md-editor :

- les XForms, c'est à dire les Editeurs de Métadonnées effectifs,
- l'interaction en SOAP avec les autres modules : ori-oai-workflow-spring et ori-oai-vocabulary.

XForms

Un certain nombre d'XForms sont proposés en standard dans ORI-OAI-Workflow.

Pour le LOM / LOMFR / SupLOMFR, plusieurs XForms sont donnés, ils permettent à des niveaux différents d'informer un fichier de métadonnées en LOM. Ces niveaux sont fonctions du degré de précision de la saisie.

Exemple : les personnes indexeurs auront accès à toutes les métadonnées du LOM (dont la dewey), tandis que le dépositaire de la ressource n'en informera que certaines.

Interaction en SOAP avec ori-oai-workflow-spring / ori-oai-vocabulary

ori-oai-md-editor communique avec ori-oai-workflow-spring en SOAP pour récupérer et sauver le XML présentant l'instance à éditer.
ori-oai-md-editor communique avec ori-oai-vocabulary pour récupérer les XMLs correspondant aux différents thésaurus/vocabulaires utilisés dans le formulaire XForms.

Mise à jour de la distribution Orbeon Forms [s'adresse uniquement aux développeurs/packagers]

Pour mettre à jour la distribution de orbeon forms :

- récupérez directement une distribution du WAR du type ops.war
- écrasez dans la distribution de ori-oai-md-editor :
    - le fichier WEB-INF/web.xml (modifiez le toutefois en supprimant les services/servlets inusités dans ori-oai-md-editor.
    - les librairies de WEB-INF/lib/
    - le répertoire WEB-INF/resources/apps/context en modificant toute fois le fichier servlet-initialized.xpl (en prenant exemple sur le diff entre l'ancien servlet-initialized.xpl_init et servlet-initialized.xpl : l'idée est d'appeler init-vocab-scheduler.xpl lorsque la servlet xforms-server est chargée)
    - le répertoire WEB-INF/resources/config en modifiant toute fois epilogue-servlet.xpl (en prenant exemple sur le diff entre l'ancien epilogue-servlet.xpl_init et epilogue-servlet.xpl) ansi que xforms-widgets.xsl, properties.xml. Recopiez également le fichier theme-ori.xsl (utilisé dans epilogue-servlet.xpl).

Suivant bien sûr les modifications apportées par la nouvelle version Orbeon Forms, ori-oai-md-editor devrait marcher directement tel quel.

Spécifications du module

Spécifications

ORI-OAI-MD-Editor (XForms+Orbeon)

Formulaires par défaut.

Un certain nombre de formulaires sont donnés par défaut : édition de fiche DC, LOM (complet et simple pour l'auteur). Libre aux utilisateurs d'en concevoir et d'en ajouter de nouveaux (et des les partager avec la communauté).
L'application étant prévu pour cela *des formulaires pour d'autres formats de métadonnées sont à l'étude*.
L'idée est effectivement de partager avec la communauté ces possibles contributions et les intégrer dans les formulaires proposés par ORI par défaut.

Ces formulaires sont en XForms (et utilisent pour faciliter/simplifier les parties de code XFORMS récurrentes un système de widgets réalisé pour l'instant via du XSL simplement : cf WEB-INF/resources/config/xforms-widgets.xsl) et interprétés par Orbeon. Certains peuvent appeler des fichiers XML distants qui sont des taxonomies/vocabulaires qui peuvent être partagés par une communauté donnée (via le module Ori-Oai-Vocabulary). Les URL de ces vocabulaires distants sont à modifier selon le besoin.

Interactions ORI-OAI-MD-Editor / ORI-OAI-Workflow-Spring

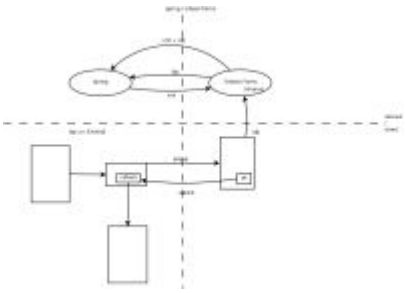
Lors d'un besoin d'édition de XML, ORI-OAI-MD-Editor est appelé depuis l'interface de ORI-OAI-Workflow-Spring avec comme paramètre un identifiant généré par Spring (idp), fonction de l'identifiant de la fiche à éditer.

Une popup (ou quelque chose de similaire) s'ouvre dans laquelle est affichée le formulaire de saisie produit par Orbeon Forms.

Pour afficher ce formulaire, Orbeon a demandé via WebService, le XML correspondant à l'identifiant

Le bouton "sauver" de la popup orbeon provoque : * l'envoi du XML et de l'identifiant idp à ORI-OAI-Workflow-Spring via WebService pour sauvegarde du XML,

- l'envoi d'une commande javascript (~ rafraichissement, submit d'un bouton, ...) sur la fenêtre principale Spring,
- la fermeture de la popup



Changements de version

Changements version 1.4.*

- Mise à jour de la version d'Orbeon Forms
- Traduction des interfaces utilisateurs en anglais, aide comprise
- Validation et mise en place des formulaires SupLOMFR (et précédemment LOMFR)
- Ajout de widgets génériques (cf xforms-widgets.xsl) et utilisation de celles-ci pour simplifier l'ensemble du code :

- widget:legend
- widget:lom-string
- widget:labels
- ...

- Mise en place d'une XSL pour transformer les fiches LOM que l'utilisateur souhaite éditer *actuellement, on transforme les fr en fre simplement*

Changements version 1.1.*

Les valeurs des propriétés principales par défaut on été modifiées : elles apparaissent entre crochet et en majuscule.
Elles peuvent soit être éditées manuellement, soit être modifées via la *configuration centrale du quick-install* voir plus bas.

ori-oai-md-editor

- L'installation de ce module peut se faire comme dans la version précédente en le disposant directement en tant qu'application

Tomcat.
Un build.xml est également maintenant disponible, il permet de déployer ce module de la même façon que tous les autres modules, via notamment le module quick-install.

- ori-oai-md-editor peut s'utiliser interconnecté avec un module ori-oai-vocabulary (ou non) et/ou avec un module de ori-oai-workflow (ou non).
  => il peut donc être utilisé sans aucune interconnection avec d'autres modules ori-oai (la version précédente imposait uen connection avec un module ori-oai-vocabulary distant)..
- Les vocabulaires utilisés sont des vocabulaires VDEX.
- Un formulaire LOMFR est à disposition pour tests, tout commentaire est le bienvenu.
- La version d'Orbeon Forms utilisée a été mis à jour : est utilisé Orbeon Forms 3.6 (sorti en Décembre 2007).
  ori-oai-md-editor bénéficie ainsi de facto de toutes les >améliorations ammenées avec cette version notamment au niveau des performances, consommation RAM, ...
- ... enfin un certain nombre d'améliorations ont été apportées sur des points de détails au niveau des formulaires ...

Installation

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

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

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

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

Installation manuelle

Installation

Vous noterez que le répertoire "source" est déposé tel quel en tant qu'application J2EE (dans webapps par exemple sur un Tomcat).

Pour faciliter les choses, on vous conseille (cf. la page Exploitation d'applications avec subversion) d'utiliser subversion pour télécharger le tag ou version stable de l'éditeur.

Une fois lancée, pour vérifier que l'application tourne comme il faut, pointez votre navigateur sur l'url de votre application tomcat
ORIOAIworkflow:HOST_MD_EDITOR:ORIOAIworkflow:PORT_MD_EDITOR/ORIOAIworkflow:CONTEXT_MD_EDITOR
c'est à dire par exemple
http://localhost:8080/ori-oai-md-editor

Vous devriez voir apparaître différents formulaires disponibles dans ori-oai-md-editor :

- Formulaire LOM complet standalone : Ce formulaire permet le référencement LOM d'une ressource pédagogique, indépendamment de tout processus - il permet l'édition de la totalité d'une fiche LOM.
- Formulaire SupLOMFR Auteur : Ce formulaire est à destination d'un auteur initiant un référencement SupLOMFR (déclinaison du standard LOM et de la norme LOMFR pour les besoins de l'enseignement supérieur français), qui sera par la suite complétée par d'autres acteurs dans le processus de référencement et d'indexation d'une ressource pédagogique - il ne permet donc l'édition que d'une partie d'une fiche SupLOMFR.
- Formulaire Dublin Core : Ce formulaire permet un référencement Dublin Core d'une ressource documentaire. Remarque : pour les ressources pédagogiques, utiliser préférentiellement les éditeurs basés sur le schéma LOM.
- Formulaire LOMFR Auteur : Ce formulaire est à destination d'un auteur initiant un référencement LOMFR (norme déclinant le standard LOM pour les besoins de l'enseignement en France), qui sera par la suite complétée par d'autres acteurs dans le processus de référencement et d'indexation d'une ressource pédagogique - il ne permet donc l'édition que d'une partie d'une fiche LOMFR.
- Formulaire LOM complet : Ce formulaire est à destination d'un acteur de type validateur (personnel TICE, bibliothécaire/documentaliste, ...) qui intervient pour vérifier et compléter le référencement LOM initié par un auteur dans le processus de référencement et d'indexation d'une ressource pédagogique - il permet l'édition de la totalité d'une fiche LOM.
- Formulaire LOM Auteur : Ce formulaire est à destination d'un auteur initiant un référencement LOM, qui sera par la suite complété par d'autres acteurs dans le processus de référencement et d'indexation d'une ressource pédagogique - il ne permet donc l'édition que d'une partie d'une fiche LOM.
- Formulaire LOMFR complet : Ce formulaire est à destination d'un acteur de type validateur (personnel TICE, bibliothécaire/documentaliste, ...) qui intervient pour vérifier et compléter le référencement LOMFR initié par un auteur dans le processus de référencement et d'indexation d'une ressource pédagogique - il permet l'édition de la totalité d'une fiche LOMFR.
- Formulaire SupLOMFR complet : Ce formulaire est à destination d'un acteur de type validateur (personnel TICE, bibliothécaire/documentaliste, ...) qui intervient pour vérifier et compléter le référencement SupLOMFR initié par un auteur dans le processus de référencement et d'indexation d'une ressource pédagogique - il permet l'édition de la totalité d'une fiche SupLOMFR.

Les fichiers propres à ORI-OAI (par rapport à une distribution Orbeon Forms) se trouvent ici :
ori-oai-md-editor/WEB-INF/resources/apps/ori-md-editor/

La configuration de base est de donner les urls des modules de workflow, de vocabulary ainsi que l'url du module md-editor lui-même. Cela se fait ici : ori-oai-md-editor/WEB-INF/resources/apps/ori-md-editor/config.xml

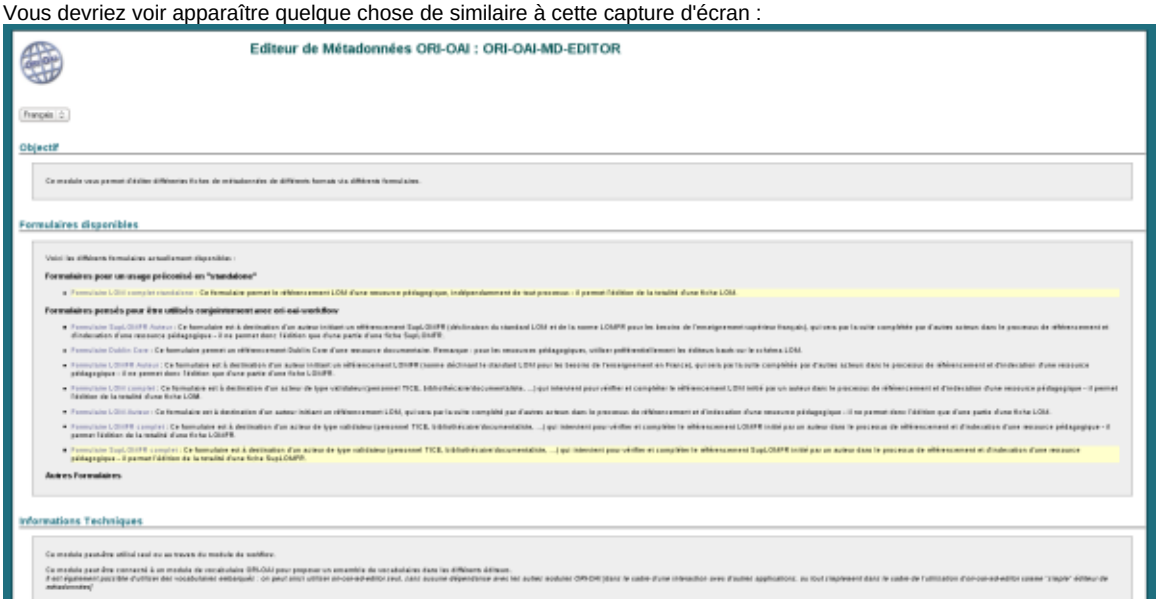
Si vous souhaitez utiliser l'éditeur seul, c'est à dire sans qu'il interagisse avec un module ori-oai-vocabulary et ori-oai-workflow, vous pouvez simplement supprimer l'url référencant le module ori-oai-vocabularry : les vocabulaires embarqués (localement) avec l'éditeur seront alors utilisés.
Le module ori-oai-workflow n'est utilisé que si on accède à l'éditeur depuis ce dernier.

Test

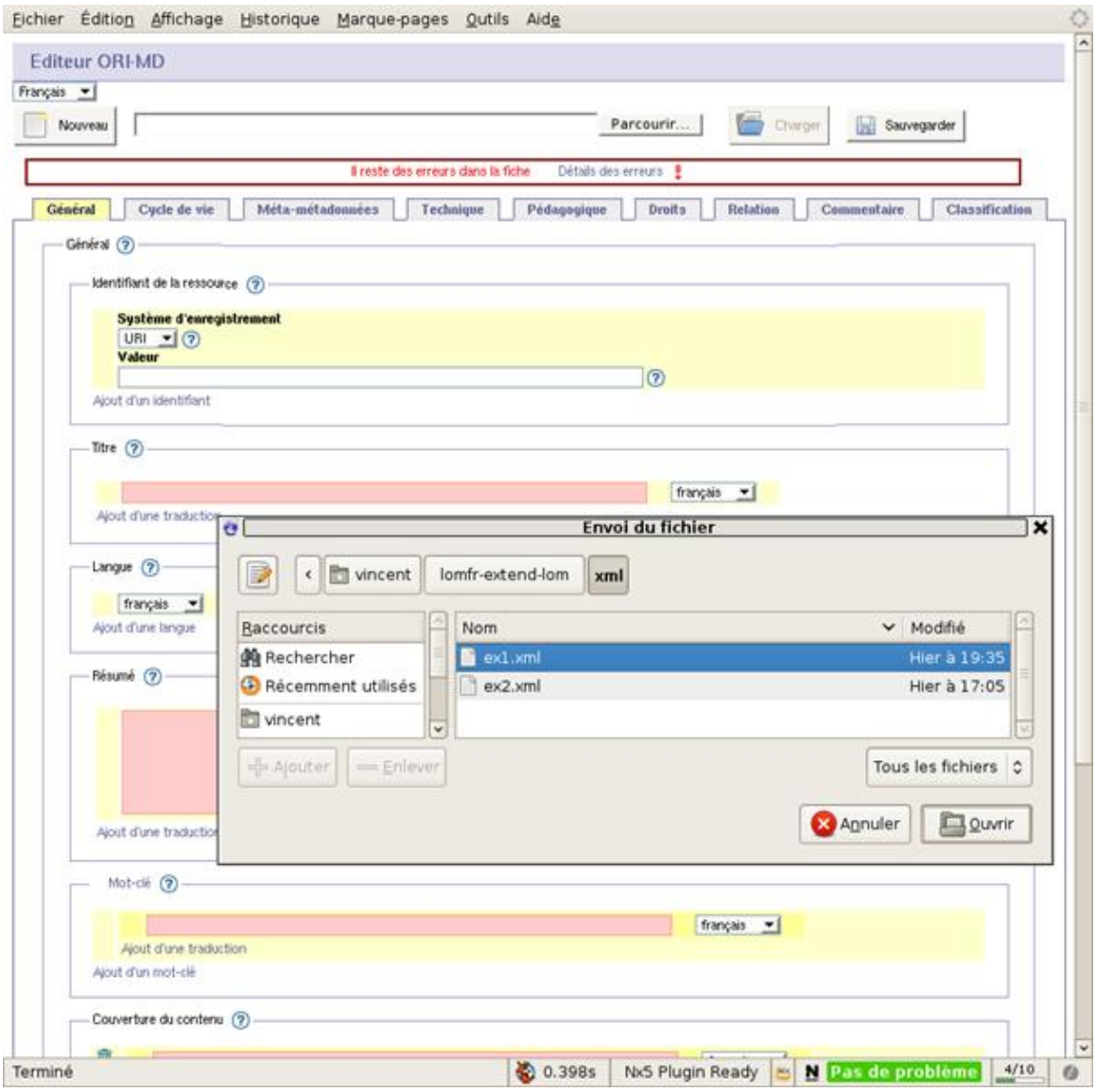
Accédez à la première page de l'éditeur depuis un navigateur web :

```
http://[HOST_INSTALL]:8186/ori-oai-md-editor
```

Vous devriez voir apparaître quelque chose de similaire à cette capture d'écran :



Note : Si le module de vocabulaire est bien configuré, lancé, que les applications dont il dépend sont également bien configurées et lancées. Vous pouvez utiliser les formulaires de l'éditeur de métadonnées directement par cette interface : vous utilisez alors le module comme un éditeur WEB de fiches de métadonnées :



Aspects pratiques

Comment ajouter une classification personnalisée dans un formulaire LOM ?

Objet

Intégrer une classification personnalisée dans un formulaire auteur, pour la 'recherche de discipline/taxonomie'.

Autrement dit, remplacer la "Classification 100 Dewey" par une "Classification personnalisée" ; et choisir un id et une entrée dans l'arborescence associée à la classification personnalisée.

Pour illustrer, nous partons de l'exemple d'un formulaire auteur LOMFR, fourni par l'Université de Rennes 2.
Nous allons remplacer dans ce formulaire, la "**Classification 100 Dewey**" par la classification nommée "**Classification UHB**".

Point de départ

- Notations utilisées pour les chemins

$EDITOR-HOME = **ori-oai-md-editor-svn/WEB-INF/resources/apps/ori-md-editor/**

$VOCAB_HOME = **ori-oai-vocabulary-svn/conf/properties/**

- Formulaire auteur LOMFR

  Contenu configuré dans $EDITOR-HOME/**xforms/lomfr-author-rennes-2/**.

  Patron de fiche configuré dans $EDITOR-HOME/**xml-bank/lomfr-author-rennes-2-blank.xml**.

Marche à suivre

Créer le vocabulaire associé à la classification

Nous nommerons le fichier de ce vocabulaire **uhb_taxonomie.xml**, par exemple.

Déposer le fichier, selon le cas (voir Implémentation standalone) :

- dans $VOCAB_HOME/**ori_vocabularies/override/** (si éditeur connecté à ori-oai-vocabulary) ;
- dans $EDITOR-HOME/**xml-vocab-local/** (si utilisation de l'éditeur seul).

Modifier la liste des classifications

Pour cela, mettre à jour les fichiers **common_lom_clas_source.xml** et **common_lom_clas_taxons.xml**, selon le cas (voir Implémentation standalone) :

- dans $VOCAB_HOME/**ori_vocabularies/override/**
  (pour l'éditeur connecté à ori-oai-vocabulary)
  Surcharger $VOCAB_HOME/ori_vocabularies/official/common_lom_clas_source.xml (voir Publication de vocabulaires statiques).
- dans $EDITOR-HOME/**xml-vocab-local/**
  (pour une utilisation de l'éditeur seul)
  Faire un copier/coller de $VOCAB_HOME/ori_vocabularies/official/common_lom_clas_source.xml et modifier.

Y ajouter les lignes suivantes :

```
<vdex:term>
<vdex:termIdentifier>Classification UHB</vdex:termIdentifier>
<vdex:caption>
<vdex:langstring language="fr">Classification UHB</vdex:langstring>
<vdex:langstring language="en">UHB  Classification</vdex:langstring>
</vdex:caption>
</vdex:term>
```

Créer le nouveau dialogue

Il s'agit de personnaliser la boîte de dialogue proposée quand on clique sur 'Recherche de taxonomie', avec la classification choisie.

- Dans **$EDITOR-HOME/xforms/lom-full/**, créer le fichier **dialog-uhbTaxonomie-search.xml**, en copiant/collant et en renommant **dialog-dewey-search.xml**. L'identifiant du dialogue est 'dialog-uhbTaxonomie-search'.
- Modifier **dialog-uhbTaxonomie-search.xml** en remplaçant toute chaîne 'dewey_100' par '**uhb_taxonomie**'. Cette instance est associée au vocabulaire "**uhb_taxonomie**".
- Modifier aussi cette ligne

```
<xxforms:hide dialog="search-taxonomy-dialog"/>
```

par

```
<xxforms:hide dialog="dialog-uhbTaxonomie-search"/>
```

Modification du formulaire

Dans $EDITOR-HOME/**xforms/lomfr-author-rennes-2/main-form.xhtml**

Remplacer toutes les chaînes 'Classification Dewey' par '**Classification UHB**'.

Remplacer les lignes suivantes en commentant (ou supprimant) celles qui existaient :

```
<xi:include href="oxf:/apps/ori-md-editor/xforms/lomfr-author-light/main-model.xml"
xxi:omit-xml-base="true"/>
par:
<xi:include href="oxf:/apps/ori-md-editor/xforms/lomfr-author-rennes-2/main-model.xml"
xxi:omit-xml-base="true"/>
```

puis

```
<xi:include href="content-xforms.xml"/>
par:
<xi:include href="oxf:/apps/ori-md-editor/xforms/lomfr-author-rennes-2/content-xforms.xml"/>
```

enfin

```
<xi:include href="../lom-full/dialog-dewey-search.xml"/>
par:
<xi:include href="../lom-full/dialog-uhbTaxonomie-search.xml"/>
```

Dans $EDITOR-HOME/**xforms/lomfr-author-rennes-2/content-xforms.xml**

Modifier les lignes suivantes :

```
<xxforms:show dialog="search-taxonomy-dialog"/>
par:
<xxforms:show dialog="dialog-uhbTaxonomie-search"/>
```

et

```
<!-- 9 Classifications -->
<xforms:group ref=
"lom:classification[lom:purpose/lom:value='discipline'][lom:taxonPath/lom:source/lom:string='Classifi
100 Dewey']/lom:taxonPath">
par:
<xforms:group ref=
"lom:classification[lom:purpose/lom:value='discipline'][lom:taxonPath/lom:source/lom:string='Classifi
UHB']/lom:taxonPath">
```

Dans $EDITOR-HOME/**xforms/lomfr-author-rennes-2/main-model.xml**

Remplacer la ligne suivante :

```
<xforms:instance id="dewey_100" src="/ori-md-editor/vocab/mdeditor_100_dewey_taxonomie"
xxforms:readonly="true" xxforms:shared="application"/>
par:
<xforms:instance id="uhb_taxonomie" src="/ori-md-editor/vocab/uhb_taxonomie" xxforms:readonly=
"true" xxforms:shared="application"/>
```

Dans $EDITOR-HOME/**i18n/ mdeditor_categories_author_lomfr.xml**

Ici, on personnalise le libellé à afficher à la place de 'Classification Dewey'.

Pour cela, modifier de la façon suivante :

```
<vdex:term validIndex="false">
<vdex:termIdentifier>4</vdex:termIdentifier>
<vdex:caption>
<vdex:langstring language="fr">Classification UHB</vdex:langstring>
</vdex:caption>
<vdex:description>
<vdex:langstring language="fr">Cliquez sur <i>Recherche de taxonomie</i> pour sélectionner une
discipline et une sous-discipline. Elle apparaît automatiquement dans le formulaire en cliquant
sur le bouton <i>Sélectionner<i> de la fenêtre.<br>
     Si la ressource appartient à plusieurs domaines, cliquez sur <i>Ajout d'un
taxon</i> pour sélectionner une ou plusieurs autres disciplines.</vdex:langstring>
</vdex:description>
<vdex:metadata>
<orioai:alert>
;<vdex:langstring language="fr">Problème</vdex:langstring>
<vdex:langstring language="en">Problem</vdex:langstring>
</orioai:alert>
</vdex:metadata>
</vdex:term>
```

Modifier le patron de la fiche auteur

Dans $EDITOR-HOME/**xml-bank/lomfr-author-rennes-2-blank.xml**

Remplacer le contenu de <lom:classification> par :

```
<lom:classification>
<lom:purpose>
<lom:source>LOMv1.0</lom:source>
<lom:value>discipline</lom:value>
</lom:purpose>
<lom:taxonPath>
<lom:source>
<lom:string>Classification UHB</lom:string>
</lom:source>
<lom:taxon>
<lom:id/>
<lom:entry>
<lom:string language="fre"/>
</lom:entry>
</lom:taxon>
</lom:taxonPath>
</lom:classification>
```

Récapitulatif des fichiers modifiés ou ajoutés:

Module md-editor:

```
$EDITOR-HOME/xml-vocab-local/uhb_taxonomie.xml
$EDITOR-HOME/xml-vocab-local/common_lom_clas_source.xml
$EDITOR-HOME/xml-vocab-local/common_lom_clas_taxons.xml
$EDITOR-HOME/xforms/lom-full/dialog-uhbTaxonomie-search.xml
$EDITOR-HOME/xforms/lomfr-author-rennes-2/main-form.xhtml
$EDITOR-HOME/xforms/lomfr-author-rennes-2/content-xforms.xml
$EDITOR-HOME/xforms/lomfr-author-rennes-2/main-model.xml
$EDITOR-HOME/i18n/mdeditor_categories_author_lomfr.xml
$EDITOR-HOME/xml-blank/lomfr-author-rennes-2-blank.xml
```

Module vocabulary:

```
$VOCAB_HOME /ori_vocabularies/override/uhb_taxonomie.xml
$VOCAB_HOME /ori_vocabularies/override/common_lom_clas_source.xml
$VOCAB_HOME /ori_vocabularies/override/common_lom_clas_taxons.xml
```

Quel fichier XML est utilisé par le formulaire ?

La réponse : plusieurs cas possibles :

* quand on ouvre l'éditeur 'directement" via http://localhost/ori-oai-md-editor/ori-md-editor/oaidc-full/blank cela correspond en fait à ouvrir le fichier oaidc-full-blank.xml
* quand on ouvre l'éditeur en passant par le workflow, 2 cas :
  - si on ouvre une fiche déjà existante, on l'ouvre tel qu'elle est présentée dans la base de données.
  - si on crée une nouvelle fiche, 2 cas à nouveau
    # si l'utilisateur courant a un "patron de page" (template, cf l'onglet profil) c'est ce patron de page qui va servir de nouvelle fiche
    # si l'utilisateur courant n'a pas de patron de page, est utilisé le fichier properties/xml/dc-prototype.xml (ou plutôt pour être plus précis le fichier indiqué en tant que defaultXmlFile dans le fichier de configuration spring-metadata-types.xml, pour le type de métadonnées utilisé - et pour oai_dc par défgaut c'est bien properties/xml/dc-prototype.xml)

Modification du formulaire Dublin Core OAI_DC

Pour des besoins de gestion interne, spécifiques à un établissement ou encore dans le projet d'initier un nouveau format de métadonnées pour échanger avec un certain nombre de partenaires, on souhaite parfois ajouter ou modifier profondément les possibilités données par le formulaire OAI_DC.
Ce cas peut se présenter pour la gestion des documents "administratifs" par exemple, pour contrôler plus finement les choix possibles d'un champ donné (dc:type par exemple), etc.

dc:type - liste de sélections

But: ne proposer qu'un vocabulaire fermé pour dc:type.

modifications dans l'éditeur

- xforms/oaidc-full/main-form.xhtml

Déclaration d'un nouveau vocabulaire adminTypes :

```
<xforms:instance id="adminTypes" src="/ori-md-editor/vocab/admin_types" xxforms:readonly="true"
xxforms:shared="application"/>
```

Le control XForms de dc:type n'est plus un xforms:input mais un xforms:select1

```
<widget:ori-block element="dc:type" minOccurs="0" maxOccurs="unbounded"
        preceding-elements="dc:date | dc:contributor | dc:publisher | dc:description | dc:subject
| dc:creator | dc:title">

     <xforms:select1 ref=".">
        <xforms:label class="hidden">Type</xforms:label>
        <xforms:alert>Problème</xforms:alert>

       <xforms:item>
  <xforms:label>[Select]</xforms:label>
  <xforms:value/>
       </xforms:item>
       <xforms:itemset nodeset="xxforms:instance('adminTypes')/vdex:term">
  <xforms:label ref="vdex:caption/vdex:langstring"/>
  <xforms:value ref="vdex:termIdentifier"/>
       </xforms:itemset>
     </xforms:select1>

    </widget:ori-block>
```

* xml-vocab-local/admin_types.xml
Pour un usage en local/standalone, on ajoute dans xml-vocab-local un vocabulaire admin_types.xml

vocabulary

Pour l'usage en utilisant le module ori-oai-vocabulary, on ajoute ce admin_types.xml dans conf/properties/ori_vocabularies/override

search

Pour prise en compte de ce nouveau vocabulary dans le moteur de recherche / recherche avancée :

Si on utilise un config.xml type documentaire, on modifiera properties/advanced/documentaire/advanced_distants.xml pour y ajouter :

```
<field format="text" id="doc_format" vocabularyId="admin_types" type="select:5">
 <metadata>//dc:type</metadata>
</field>
```

extension DC / OAI_DC

On peut avoir envie d'étendre le formulaire OAI_DC, c'est à dire d'ajouter des possibilités non présents par défaut : nouveaux champs etc. Comme pour l'extension d'autres formulaires (et donc de schémas de métadonnées) cela n'est pas annodin. En effet en étendant ces schémas, on ne va plus correspondre strictement à un schéma XML standardisé donné, mais on aura quelque chose de spécifique à notre établissement qui sera donc potentiellement difficile à échanger/partager avec d'autres.

Procéder à ce type de manipulations peut toute fois s'avérer nécessaire lorsqu'on souhaite mettre en place des métadonnées/champs effectivement très spécifiques à son établissement (métadonnées/champs internes non dédiés à être partagés), cela peut également être un travail à effectuer lors de l'élaboration d'un nouveau type de XML qui sera échangé par plusieurs partenaires (d'une même UNR, UNT ou plus encore).

Même si l'ajout est spécifique ou découle d'un besoin exprimé par plusieurs entités, il est préférable de "se raccrocher" au mieux à des usages et standards existants. Il est en effet très rare que l'on ressente un besoin jamais exprimé par d'autres établissements (à un niveau international).
Dans le cadre de OAI_DC, l'usage des extensions DC/DCMI est relativement usuelle.
On regardera donc attentivement les éléments que peuvent proposer ces différents schémas http://dublincore.org/schemas/xmls/ pour inclure certaines métadonnées/champs dans un formulaire spécifique OAI_DC.

Voir à ce propos également le formulaire dc_plus_fr en construction dans ori-oai-md-editor

Ajout d'une métadonnée dans le formulaire

> ⚠ Cette documentation n'est actuellement pas à jour (ne correspond pas à ce que l'on trouve en version 1.4) : les fichiers et codes correspondent à ce que l'on peut trouver dans les versions 1.1 de l'éditeur.

XML permet d'étendre voir de concevoir entièrement des langages (applications) XML. En respectant les espaces de noms, on peut ainsi arriver à rester complètement compatible avec un langage donné tout en ajoutant des éléments et attributs propres.

> ⚠ Attention, ajouter des éléments et attributs propres à un schéma XML n'est pas anodin : cela revient finalement à définir un nouveau schéma XML ou plutôt une nouvelle application XML.
> Dans le contexte de partages, d'échanges via des réseaux de portails communicant, cela doit se faire au maximum conjointement avec le plus de partenaires possibles et en utilisant au mieux des éléments/attributs déjà standardisés. En fait dans la majorité des cas, il faudra tenter de se contenter des éléments/attributs disponibles dans le schéma standard utilisé.
> Ici on décrit techniquement les manipulations qui peuvent permettre d'ajouter des éléments et attributs dans un formulaire donné ... mais cela présuppose qu'une étude et une réflexion poussée ont été réalisées en amont. On peut alors procéder à ces manipulations en tout connaissance de cause ...
> L'exemple donné ci-dessous se justifie comme ceci : on précise que l'on souhaite ajouter une métdonnée de gestion interne permettant de cibler l'UNT cible - l'idée étant finalement de créer des Sets OAI-PMH fonction des UNT, on répartit les fiches dans ces Sets en fonction du public cible (de cette nouvelle métadonnée).
> *On notera cependant pour être complet que, avec une réflexion plus poussée, l'utilisation d'une classification disciplinaire (type Dewey) devrait nous permettre de cibler l'UNT (ou les UNT) cible(s) d'une ressource donnée, et cela de manière plus élégante (car sans ajouter de spécificités internes aux applications XML que sont les LOM/LOMFR/SupLOMFR).*
>
> Enfin, à terme (**à terme** car c'est en cours de construction !) et pour un certain nombre de cas d'utilisation, l'idée ne sera pas d'étendre un schéma donné standard mais plutôt de proposer une description d'une même ressource via plusieurs applications XML, c'est la notion que l'on a appelée "format englobant" dans les échanges fonctionnels ORI-OAI et que l'on est en train de mettre en place actuellement. L'intérêt est alors de pouvoir présenter par exemple une fonctionnalité de screenshots via une application XML donné cela aussi bien pour les ressources dont les fiches sont locales ou moissonnées et qui sont déjà décrites en OAI_DC ou LOM ... on arrive alors finalement à plusieurs applications XML décrivant, par des points de vue fonctionnels différents, une même ressource.

Pour plus d'informations la dessus, revoyez également XmL et les Espaces de noms sur ce même site.

Ici, on montre comment on peut ajouter une métadonnée à un XML LOM dans un espace de noms propre et comment on peut ensuite proposer à l'utilisateur d'éditer ce XML dans le formulaire LOM de l'éditeur de métadonnées ORI-OAI : ORI-OAI-MD-EDITOR.

En bref, ce chapitre explique comment on peut étendre/modifier (et en fait créer) un formulaire dans ORI-OAI-md-editor basé, on le rappelle, sur Orbeon Forms.

Espace de noms Rennes1 et exemple d'XML LOM Rennes1

Sur un ori-oai-md-editor d'installé et qui fonctionne, on va donc ici montrer comment créer un nouveau formulaire sur la base du formulaire LOM pour lui ajouter une nouvelle métadonnée.

Ici on se place dans la position de l'Université de Rennes1 par exemple qui veut rajouter une métadonnée interne pour "taguer" ses ressources : l'Université faisant partie de chacune des UNT, on souhaiterait en effet lors de l'indexation d'une ressource pédagogique préciser à quel UNT la ressource peut correspondre. Cela permet de cibler un certain publique ... pour rester souple, on utilisera plutôt un nom de balise comme **targetPublic** par exemple.

Voici un exemple de fichier XML LOM étendu via un espace de noms et des balises propres à Rennes1 que l'on souhaiterait éditer dans notre nouveau formulaire donc :

```
<?xml version="1.0" encoding="utf-8"?>
<lom xmlns="http://ltsc.ieee.org/xsd/LOM"
     xmlns:lomur1="http://www.univ-rennes1.fr/xsd/LOMUR1">
  <general>
    <identifier>
      <catalog>URI</catalog>
      <entry>http://www.univ-rennes1.fr/12345</entry>
    </identifier>
    <title>
      <string language="fr">Médecine Informatique et Écologie</string>
    </title>
    <!-- etc. -->
  </general>
  <!-- etc. -->
  <lomur1:lomur1>
    <lomur1:targetPublic>UMVF</lomur1:targetPublic>
    <lomur1:targetPublic>UNIT</lomur1:targetPublic>
    <lomur1:targetPublic>UVED</lomur1:targetPublic>
  </lomur1:lomur1>
</lom>
```

Ce fichier est valide *lomLoose*, et on n'a pas forcément besoin de créer un schéma spécifique "LOM-Rennes1" pour notre besoin : celui-ci reste interne, notre espace de noms n'a pas vocation à être utilisé en dehors de nos applications. Par contre, on distingue bien grâce aux espaces de noms le LOM de nos balises personnelles.

Grâce aux espaces de noms, on peut noter que les différents modules peuvent travailler avec cet XML d'exemple ci-dessus : l'éditeur ori-oai-md-editor peut éditer cet XML directement (aussi bien la version complète, que la version auteur). Cependant, il ne nous permet pas d'éditer la partie spécifique Rennes1 qu'il ne connait pas.

C'est pourquoi on va réaliser un nouveau formulaire basé sur le formulaire LOM complet d'ori-oai-md-editor.

Formulaire LOM Rennes1

copier/coller

Confère la documentation du ori-oai-md-editor à http://sourcesup.cru.fr/ori-workflow/ (qui partage le projet sourcesup avec ori-oai-workflow), le répertoire spécifique à ori-oai-md-editor dans l'application Orbeon Forms est : **WEB-INF/resources/apps/ori-md-editor**.
On va dupliquer le formulaire lom-full, je vous liste les commandes :

[ori-md-editor]$ ls
close.xhtml config.xml i18n media page-flow.xml prototypes reload-vocab schemas welcome xforms xml-blank xpl
[ori-md-editor]$ ls xforms/
common lom-author-light lom-full oaidc-full
[ori-md-editor]$ cp -rf xforms/lom-full xforms/lom-full-extend
[ori-md-editor]$ cp xml-blank/lom-full-blank.xml xml-blank/lom-full-extend-blank.xml

Les formulaires en tant que tels correspondent aux sous-répertoires du répertoire xforms (excepté le répertoire common).

Les fichiers d'initialisation des formulaires en mode "standalone" (non connecté au workflow) sont les fichiers du répertoire **xml-blank** dont les noms doivent correspondre avec les noms des répertoires des formulaires.

Dès maintenant et sans redémarrer votre serveur ori-oai-md-editor, vous devez pouvoir utiliser ce nouveau formulaire directement :

```
http://NOM_SERVEUR:PORT/ori-oai-md-editor/ori-md-editor/lom-full-extend/blank
```

svn update

Pour ceux qui sont connectés via subversion (cf. **l'Annexe B** : « Exploitation d'applications avec subversion - installation et mises à jour », on vous encourage vivement à procéder de la sorte) vous pouvez garder le répertoire **.svn** copié au passage dans **xforms/lom-full-extend**, cela vous permet de continuer à vous synchroniser au répertoire **lom-full** du repository svn officiel de ori-oai-md-editor. Par contre pour ce faire vous devrez aller dans le répertoire **lom-full-extend** et faire un svn update indépendamment du svn update de l'ensemble :

```
cd xforms/lom-full-extend/
svn info
```

Path: .URL: http://subversion.cru.fr/ori-workflow/ori-oai-md-editor/trunk/WEB-INF/resources/apps/ori-md-editor/xforms/lom-full
Repository Root: http://subversion.cru.fr/ori-workflow
Revision: 471Node Kind: directorySchedule: normalLast Changed Author: vbonamyLast Changed Rev: 471Last Changed Date: 2007-12-16 14:05:14 +0100 (Sun, 16 Dec 2007)

a

```
svn update
```

U case-educational.xmlU main-form.xhtmlUpdated to revision 471.

Prototype

Dans XForms et Orbeon Forms, l'ajout de noeud via un bouton "**Ajouter un targetPublic**" correspond en fait à un copier coller d'un nœud existant "ailleurs". Le "ailleurs" peut correspondre à une autre instance XML que celle que l'on édite. On appelle cela dans ori-oai-md-editor un prototype.

Ici, on va ajouter un prototype pour le LOM Rennes1 :

```
cd prototypes
cp lom-prototype.xml lom-rennes1-prototype.xml
```

On y ajoute les tags vides qui vont bien, **en n'oubliant pas de déclarer l'espace de noms :**

```
....
xmlns:lomur1="http://www.univ-rennes1.fr/xsd/LOMUR1"
...
    <lomur1:lomur1>
        <lomur1:targetPublic/>
    </lomur1:lomur1>
...
```

Modification du formulaire

On se place dans le formulaire nouvellement créé pour maintenant le modifier :

```
cd xforms/lom-full-extend/
ls
```

case-annotation.xml case-educational.xml case-lifecycle.xml case-relation.xml case-technical.xml
dialog-taxonomy-search.xml entity.xmlcase-classification.xml case-general.xml case-metametadata.xml case-rights.xml
dialog-dewey-search.xml dialog-vcard-search.xml main-form.xhtml

Ces différents fichiers sont des XForms. Les modifications que l'on va apporter correspondent à reprendre du code existant dans ces formulaires pour l'adapter à nos nouvelles balises, modifier les vocabulaires etc.

Le fichier d'entrée du formulaire est en fait **main-form.xhtml**. Dans ce fichier main-form.xhtml :

- on change le prototype utilisé (on recherche simplement **lom-prototype.xml** que l'on va changer par **lom-rennes1-prototype.xml**)

```
<!-- LOM prototype -->
<xforms:instance id="ori-prototype"
                 src="oxf:/apps/ori-md-editor/prototypes/lom-rennes1-prototype.xml"
                 xxforms:readonly="true"
                 xxforms:shared="application"/>
```

- on ajoute un onglet rennes1 en dessous de celui de classification (en copiant collant celui de classification):

```
<widget:tab id="rennes1">
      <widget:label>Rennes1</widget:label>
     <xi:include href="case-rennes1.xml"/>
 </widget:tab>
```

On va ensuite éditer un **case-rennes1.xml** en prenant exemple sur les autres **case-**.xml* en n'oubliant pas de déclarer l'espace de noms rennes1 et de l'utiliser !

Voici ce que peut donner ici un tel **case-rennes1.xml** :

```
<xforms:group ref="." xmlns="http://www.w3.org/1999/xhtml"
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xforms="http://www.w3.org/2002/xforms"
          xmlns:ev="http://www.w3.org/2001/xml-events"
          xmlns:xxforms="http://orbeon.org/oxf/xml/xforms"
          xmlns:xi="http://www.w3.org/2001/XInclude"
          xmlns:f="http://orbeon.org/oxf/xml/formatting"
          xmlns:lom="http://ltsc.ieee.org/xsd/LOM"
          xmlns:xhtml="http://www.w3.org/1999/xhtml"
          xmlns:widget="http://orbeon.org/oxf/xml/widget"
          xmlns:vocab="http://www.ori-oai.org/xsd/orioaivocab"
              xmlns:lomur1="http://www.univ-rennes1.fr/xsd/LOMUR1">


  <!-- Rennes1 -->
  <fieldset>
    <legend>
          Rennes1 - Metadonnees specifiques
    </legend>

    <widget:ori-block element="lomur1:lomur1" minOccurs="0" maxOccurs="1" preceding-elements=
"lom:general | lom:lifeCycle | lom:metaMetadata | lom:technical | lom:educational | lom:rights |
lom:relation | lom:annotation | lom:classification">

      <!-- targetPublic -->
      <fieldset>

        <legend>
   targetPublic ...
</legend>

<widget:ori-block element="lomur1:targetPublic" minOccurs="0" maxOccurs="unbounded"
parent-element="lomur1:lomur1">
          <xforms:select1 ref=".">
            <xforms:item>
              <xforms:label>[Select]</xforms:label>
              <xforms:value></xforms:value>
            </xforms:item>
            <xforms:item>
              <xforms:label>UMVF</xforms:label>
              <xforms:value>UMVF</xforms:value>
            </xforms:item>
            <xforms:item>
              <xforms:label>UNIT</xforms:label>
              <xforms:value>UNIT</xforms:value>
            </xforms:item>
            <xforms:item>
              <xforms:label>UVED</xforms:label>
              <xforms:value>UVED</xforms:value>
            </xforms:item>
            <xforms:item>
              <xforms:label>UNJF</xforms:label>
              <xforms:value>UNJF</xforms:value>
            </xforms:item>
            <xforms:item>
              <xforms:label>AUNEGE</xforms:label>
              <xforms:value>AUNEGE</xforms:value>
            </xforms:item>
            <xforms:item>
              <xforms:label>UOH</xforms:label>
              <xforms:value>UOH</xforms:value>
```

```
            </xforms:item>
            <xforms:item>
              <xforms:label>UNISCIEL</xforms:label>
              <xforms:value>UNISCIEL</xforms:value>
            </xforms:item>
          </xforms:select1>
        </widget:ori-block>

      </fieldset>

    </widget:ori-block>

  </fieldset>
```

```
</xforms:group>
```

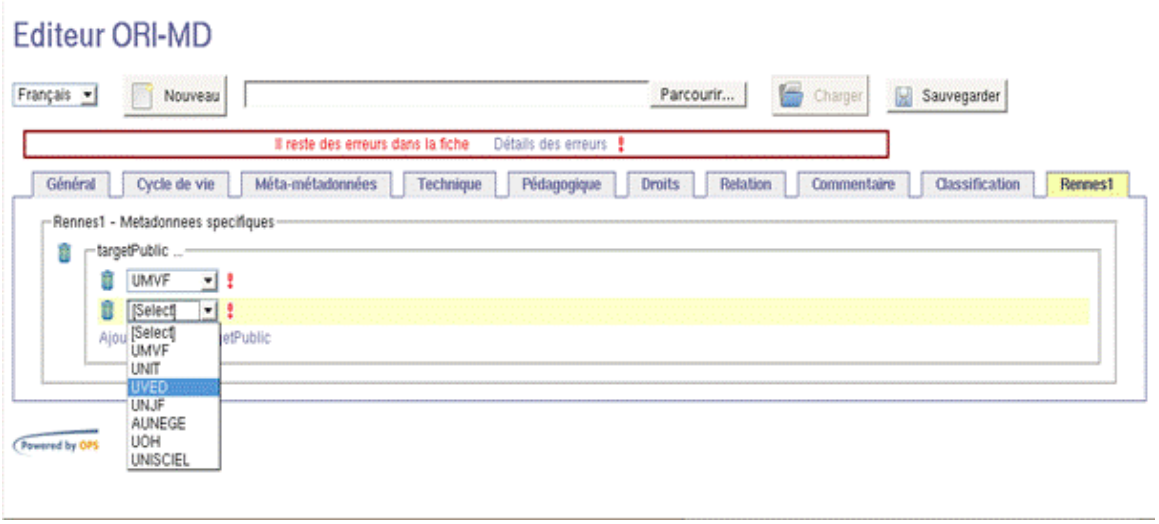
Termes i18n ...

Même si on a mis des termes en dur dans le XForms ci-dessus, la facilité des widgets qu'on donne dans ori-oai-md-editor nécessite le positionnement de paramètres i18n pour ajouter/supprimer les tags que l'on à rajouter. On ajoutera donc dans **i18n/fr_FR.xml** la partie suivante (pensez à faire de même pour **i18n/en_EN.xml** :

```
<!-- LOM Rennes1 - add ... -->
<add-lomur1>Ajouter un bloc Lomur1</add-lomur1>
<del-lomur1>Supprimer le bloc Lomur1</del-lomur1>
<add-targetPublic>Ajouter un bloc TargetPublic</add-targetPublic>
<del-targetPublic>Supprimer le bloc TargetPublic</del-targetPublic>
```

Redémarrez votre serveur tomcat.

Vous devriez maintenant avoir un nouvel onglet dans votre éditeur de métadonnées !



Personnalisation de formulaires

Création / personnalisation de formulaires

ori-oai-md-editor a été pensé pour permettre l'ajout de nouveaux formulaires.

Pour réaliser un **nouveau formulaire** (pour un nouveau schéma ou un schéma déjà supporté), le plus simple est de **copier/coller (en renommant)** depuis le répertoire ori-oai-md-editor/WEB-INF/resources/apps/ori-md-editor/xforms un dossier représentant un formulaire existant : oaidc-full par exemple.
Si vous renommez ce nouveau répertoire oaidc-full-custom par exemple, vous devriez le voir ensuite aussitôt apparaître dans la liste des formulaires disponibles sur http://localhost:8280/ori-oai-md-editor.

Il faut aussi créer le **patron de fiche**, qui sert à initialiser le formulaire et permet son affichage. Pour cela, **copier/coller** ori-oai-md-editor/WEB-INF/resources/apps/ori-md-editor/xml-blank/oaidc-full-blank.xml dans le même dossier, dans notre exemple **renommer** le en oaidc-full-custom-blank.xml.

Remarque : le patron de fiche mentionné ici sert à initialiser le formulaire quand l'éditeur est utilisé sans connexion au module ori-oai-workflow.

Les modifications effectives d'un formulaire se réalisent en éditant et modifiant les fichiers XForms du répertoire du formulaire : ori-oai-md-editor/WEB-INF/resources/apps/ori-md-editor/xforms/oaidc-full-custom par exemple.
Le **point d'entrée** d'un formulaire est obligatoirement le fichier **main-form.xhtml** (qui peut suivant les cas, être le seul fichier du répertoire).

Ne pas oublier, ***le cas échéant***, de modifier le fichier utilisé pour initialiser le formulaire, le patron de fiche : ici oaidc-full-custom-blank.xml.

Améliorer les performances

Performances

Cf Orbeon Performance and Tuning, il est possible d'améliorer les performance de Orbeon Forms de différentes manières. La version utilisée d'Orbeon Forms embarque désormais une base eXist utilisé pour stocker les instances XML en cours d'usage. Aussi ORI-OAI-MD-Editor a normalement moins besoin de RAM qu'avant. Cependant il reste conseillé d'allouer un large volume de RAM au Tomcat faisant tourner Orbeon Forms. ainsi par exemple vous pouvez positionner votre variable d'environnement comme cela :

export JAVA_OPTS='-Xmx512m -Xms512m'

(voir plus encore que 512MO si vous avez la possibilité d'allouer plus de RAM)
L'utilisation du JDK6 est conseillé également afin d'améliorer les performances.

Il est fortement conseillé d'utiliser les possibilités de compression des pages css, javascript et html (notamment la page html représentant le formulaire qui peut être relativement importante et qui ne doit d'ailleurs pas être mise en cache au niveau du navigateur comme du serveur web). Cela peut se faire via Tomcat directement ou plus aisément encore via Apache par exemple si vous l'utilisez comme serveur web frontal à Tomcat.

Voici par exemple via Apache comment procéder : il faudra s'assurer que les modes deflate et header sont chargés :

```
<Location /ori-oai-md-editor>

<IfModule mod_deflate.c>

SetOutputFilter DEFLATE

# Don't compress images
SetEnvIfNoCase Request_URI \
\.(?:gif|jpe?g|png)$ no-gzip dont-vary

# Make sure proxies don't deliver the wrong content
Header append Vary User-Agent env=!dont-vary

</IfModule>

</Location>
```

Enfin, il est aussi intéressant de cacher les images, css et javascript au niveau du serveur Apache (seul la page html représentant le formulaire ne doit pas être mis en cache au niveau du navigateur comme du serveur web, c'est important). Via le mode cache (disk_cache) d'Apache, voici comme cela peut se mettre en place simplement :

```
<IfModule mod_disk_cache.c>
CacheEnable disk /
</IfModule>
```

ori-oai-md-editor et le module ori-oai-vocabulary

ori-oai-md-editor et le module ori-oai-vocabulary

ori-oai-md-editor peut être client de ori-oai-vocabulary : il récupère alors par Web Services les vocabulaires dont il a besoin.
Ainsi par exemple, vous pouvez récupérer le vocabulaire des langues du module ori-oai-vocabulary d'UNIT via le module ori-oai-md-editor d'UNIT : http://www.unit.eu/ori-oai-md-editor/ori-md-editor/vocab/languages. Vous pouvez donc faire de même entre votre module ori-oai-md-editor et votre module ori-oai-vocabulary.
Cela vous permet de tester la bonne configuration de ori-oai-md-editor->ori-oai-vocabulary.
Notez que depuis la 1.4.0 le test et affichage des vocabulaires d'ori-oai-vocabulary peut se faire directement via l'interface Web d'ori-oai-vocabulary.

Les formulaires Orbeon Forms de Ori-Oai-Md-Editor partagent les mêmes vocabulaires qui sont mis en cache au niveau d'Orbeon Forms même.
Par défaut, ces caches sont initialisés lors du lancement de l'application (donc du Tomcat) puis ils sont rechargés automatiquement régulièrement : le paramètre vocab-cache-time dans le fichier config.xml stipule l'intervalle de rechargement (en millisecondes).
L'appel à des urls particulières permettent également de forcer manuellement à charger/décharger/recharger les caches : http://localhost:8280/ori-oai-md-editor/ori-md-editor/reload-vocab permet de recharger les vocabulaires.

Encodage

Encodage

De par leurs fonctionnalités et leurs objectifs d'interropérabilité, les modules ORI-OAI sont voués à fonctionner dans un encodage UTF-8. Il est préférable (pour éviter au mieux les problèmes d'encodage dans les données, le rendu etc.) de positionner l'encodage à UTF-8 pour tous les composants acteurs dans ORI-OAI (les bases de données, les serveurs d'applciation, etc). Cela peut se faire de différentes manières (depuis les variables d'environnement du système par exemple [LANG]).

Le positionner en tant qu'option Tomcat est également préférable:

```
CATALINA_OPTS=-Dfile.encoding=UTF-8
```

Encodage et MySQL

Côté MySQL

Pour MySQL, le plus simple est de configurer l'**UTF-8** à la fois au niveau des **"Connection Character Sets"** et des **"Collations"**.
Pour ce faire, on ajoute dans la configuration de mysql (sous debian /etc/mysql/my.cnf) la confiuration suivante (après [mysqld] au même niveau que **default-storage_engine = innodb** :

```
character-set-server=utf8
collation-server=utf8_general_ci
```

On notera qu'après une telle modification il faut redémarrer le serveur mysql.

On peut ensuite vérifier la bonne prise en compte de ces variables :

```
mysql> SHOW VARIABLES LIKE 'character_set%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | latin1                     |
| character_set_connection | latin1                     |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | latin1                     |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)

mysql> SHOW VARIABLES LIKE 'collation%';
+----------------------+-------------------+
| Variable_name        | Value             |
+----------------------+-------------------+
| collation_connection | latin1_swedish_ci |
| collation_database   | utf8_general_ci   |
| collation_server     | utf8_general_ci   |
+----------------------+-------------------+
3 rows in set (0.00 sec)
```

Côté client Hibernate

Si comme indiqué au dessus, on a positionné à la fois les **collation** et les **character set** à **utf-8**, cette configuration côté Hibernate n'est **a priori** pas nécessaire.

Ici on force Hibernate à demander un dialogue UTF-8 avec la base de données, cela en ajoutant dans les paramètres de connection

- useUnicode=true
- characterEncoding=utf8
- useOldUTF8Behavior=true

Pour ori-oai-workflow, dans **conf/properties/spring/common/dao/dao.xml**
On mettra à la place de

```
<property name="url">
  <value>${hibernate.connection.url}?emulateLocators=true</value>
</property>
```

cela :

```
<property name="url">
 <value>${hibernate.connection.url}?emulateLocators=true&amp;useUnicode=true
&amp;characterEncoding=utf8&amp;useOldUTF8Behavior=true</value>
</property>
```

Implémentation standalone

Implémentation standalone - Mise en place rapide d'un éditeur de métadonnées

Le module ori-oai-md-editor est destiné à être utilisé en connexion avec d'autres modules d'ORI-OAI (ori-oai-vocabulary, ori-oai-workflow). Il peut aussi être utilisé seul.

Pouvoir utiliser le module ori-oai-md-editor indépendamment d'un workflow est intéressant dans certains cas:
- pour tester un formulaire que l'on est en train de créer/modifier ;
- pour créer/éditer des fiches de métadonnées en dehors des autres modules d'ORI-OAI (ori-oai-md-editor jouant alors de rôle de simple éditeur de fiches XML dans des formats pré-définis dans le module : LOM et ses dérivés français, Dublin Core,...).

Par défaut, le module ori-oai-md-editor est configuré de manière à récupérer certains vocabulaires dont il a besoin auprès du module ori-oai-vocabulary. Dans ce cas de figure, il peut certes être utilisé indépendamment du module ori-oai-workflow, mais un module ori-oai-vocabulary reste obligatoire.

On présente ici la possibilité d'utiliser ori-oai-md-editor SEUL, comme éditeur de fiches XML LOM, LOMFR, SupLOMFR ou Dublin Core.
L'installation et la configuration de ce seul module sont alors rapide !

Installation

Se reporter à la documentation du module (une installation avec subversion est conseillée, notamment pour faciliter les mises à jour).

Configuration

(Notation : $EDITOR-HOME = **ori-oai-md-editor-svn/WEB-INF/resources/apps/ori-md-editor/**)

Pour utiliser l'**éditeur seul** (sans aucune interaction avec ori-oai-workflow, ni avec ori-oai-vocabulary), éditer le fichier $EDITOR-HOME/**config.xml**.

Dans la balise <vocabulary>, mettre

```
<location></location>
```

au lieu de

```
<location>http://[HOST_VOCABULARY]:[PORT_VOCABULARY]/[CONTEXT_VOCABULARY]</location>
```

Utilisation

A faire ...