Space Details

Key:	ORIOAIvocabulary
Name:	ORI-OAI-vocabulary
Description:	
Creator (Creation Date):	ycolmant@univ-valenciennes.fr (Jul 04, 2008)
Last Modifier (Mod. Date):	ycolmant@univ-valenciennes.fr (Aug 29, 2008)

Available Pages

- Version 1.1
  - Spécifications
    - Choix techniques
    - Spécifications du module
    - Modélisation
    - Implémentation
  - Installation
    - Changements version 1.1.0
    - Installation via Quick Install
    - Configurations
      - Configurations minimales
      - Configuration avancée
      - Passage à VDEX (version 1.0 à 1.1)
    - Déploiement
    - Plugin
  - Utilisation
  - Aspects pratiques
    - Test
    - Encodage

Version 1.1

This page last changed on Jan 05, 2009 by ycolmant@univ-valenciennes.fr.

ORI-OAI-vocabulary : Gestion des classifications et vocabulaires

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

Voir l'architecture du système

Dans quels cas l'utiliser

Pour la gestion des vocabulaires et des classifications dans tout le système

Composants optionnels

- **ORI-OAI-indexing** pour générer des vocabulaires à partir de champs indexés
- Un **annuaire LDAP** pour générer des vocabulaires à partir de membres de l'établissements ou de groupes LDAP

Description

Le composant ORI-OAI-vocabulary est celui qui gère les vocabulaires utilisés dans différents modules. On entend par vocabulaire un ensemble fermé de valeurs disponibles pour un critère donné.

Les vocabulaires se reposent sur le format VDEX. Ils peuvent être statiques et configurés via des fichiers XML comme par exemple des classifications de documents ou des valeurs strictes de champs de métadonnées LOM. Ils peuvent aussi être dynamiques comme dans le cas de toutes les valeurs disponibles dans l'index d'une métadonnée précise. Par exemple, on pourra constituer dynamiquement, en interrogeant le module ORI-OAI-indexing, la liste des mots-clefs libres ou des auteurs qui ont déjà été saisis dans les documents pédagogiques.

Ces vocabulaires sont utilisés par :

- Le module ORI-OAI-md-editor pour proposer des listes de valeurs lors de la saisie des métadonnées.
- Le moteur de recherche ORI-OAI-search pour les recherches thématiques ou les valeurs disponibles pour certains champs de la recherche avancée.
- L'entrepôt ORI-OAI-repository pour générer dynamiquement des sets OAI en fonction par exemple d'une thématique donnée.

Voir la documentation technique

Spécifications

This page last changed on Jan 05, 2009 by ycolmant@univ-valenciennes.fr.

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Choix techniques

2 choix importants sont à noter : * d'une part on choisit Esup-Commons comme Framework principal pour Ori-Oai-Vocabulary (ce qui induit différents choix technologiques comme Spring bien sûr, mais aussi XFire, EhCache, etc.),

- d'autre part, et ce dans la mesure du possible, on utilise VDEX pour l'échange des vocabulaires entre modules : http://www.imsglobal.org/vdex

> ⚠ Dans les premières versions (dont la version actuelle) de Ori-Oai-Vocabulary, le format des vocabulaires n'est cependant pas un format VDEX mais un langage XML spécifique à ORI (~ orioaivocab). Dans la suite de ce document, même si l'on fait référence à VDEX, notez que celui-ci n'est pour l'instant pas utilisé (ou très peu) dans la version actuelle du module.

This page last changed on Jan 05, 2009 by ycolmant@univ-valenciennes.fr.

Spécifications

Les sources de vocabulaires que gère Ori-Oai-Vocabulary : * Autres modules Ori-Oai-Vocabulary : l'idée est qu'une UNT par exemple puisse proposer des vocabulaires partagés aux différents partenaires.

- Ldap : le but est de récupérer les VCards des personnes physiques et "morales" (départements, services, laboratoires, ...)
- Schémas XML ("Normalisés") : l'idée est de mettre en forme en VDEX les vocabulaires spécifiés dans les XML Schémas du Lom, Lom-Fr, etc.
- Vocabulaires "dynamiques" : ici l'idée est de tenter d'unifier certains vocabulaires issus de champs libres (comme les mots-clés par exemple), ils sont mis à jour via le module d'indexation (Ori-Oai-Indexing) qui fournit les valeurs uniques de champs donnés.
- Vocabulaires XML "statiques" : cela correspond à un vocabulaire stocké directement en XML dans un fichier statique.
- Vocabulaires XML "éditables/modifiables" : le principe est d'utiliser le module ORI-OAI-Workflow pour permettre à des utilisateurs de proposer la modification de vocabulaires.
- Vocabulaires SQL provenant de bases de données diverses (Apogée, ...)
- Autres ... 

Ori-Oai-Vocabulary permet de gérer et fusionner différents vocabulaires.

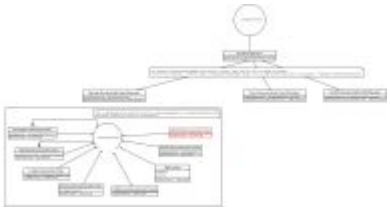
This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Modélisation

Cette section présente une sorte de première analyse de Ori-Oai-Vocabulary. Même si par rapport à l'implémentation qui en a été faite, cette modélisation reste conceptuelle, elle permet de mieux appréhender l'architecture de Ori-Oai-Vocabulary.

Diagramme de classes

Ce diagramme de classes se veut simple pour une bonne compréhension de l'ensemble. Il ne présente pas un certain nombres d'attributs et surtout de classes qui sont d'ordre architectural (provenant notamment de Esup-Commons).



Quelques Explications sur certaines classes

VocabularyService

> C'est cette classe qui est exposée via WebService aux autres modules afin de permettre la consultation les différents vocabulaires. L'exposition par WebService se fait par fichiers de configuration (XFire + Spring)
> Elle a en attributs une liste de VocabularyProviderManager qu'elle consulte un par un dans l'ordre lorsqu'on lui demande un vocabulaire. Si aucun vocabularyProviderManager ne peut finalement répondre favorablement à la requête, une exception est levée.
> C'est cet objet qu'on propose de brancher sur un système de cache via un fichier de configuration spring (cf implémentation).

VocabularyProviderManager

> Cette classe construit les différents vocabulaires. Ces vocabulaires ainsi construits sont exposés par VocabularyService.
> Cette classe correspond à la fois à la super classe des différents VocabularyProviderManager et à une implémentation simple d'un VocabularyProviderManager : elle fournit les vocabulaires proposés par les différents VocabularyProvider qu'elle a directement en attribut. Afin d'alléger les configurations, on préfèrera plutôt utiliser directement un DynamicVocabularyProviderManager.

DynamicVocabularyProviderManager

> Cette classe étend la classe de base VocabularyProviderManager : elle permet simplement de ne pas spécifier les VocabularyProvider en tant qu'attribut. On spécifie simplement une targetClass (VocabularyProvider) et elle récupère dynamiquement tous les VocabularyProvider déclarés dans la configuration Spring.

RemoteVocabularyProviderManager

> Cette classe étend la classe de base VocabularyProviderManager : elle permet de se connecter à un VocabularyService pour en récupérer ses vocabulaires.

FileBrowserVocabularyProviderManager

Cette classe étend la classe de base VocabularyProviderManager : elle permet d'exposer les vocabulaires présent dans un répertoire.

VocabularyProvider

Un VocabularyProvider est un fournisseur de vocabulaires, il propose une méthode getXmlStream qui renvoie le vocabulaire produit sous format XML. Au niveau du déploiement (configuration), il peut bien sûr y avoir plusieurs instances de chaque classe de VocabularyProvider.
Chaque VocabularyProvider doit implémenter une méthode getXmlStream qui doit retourner un InputStream.

> ⚠ On pourrait envisager de finalement modifier cela et demander à retourner un String plutôt qu'un InputStream afin que cela soit serializable et donc cacheable par une config spring/ehCache ... à voir ...

Les différents VocabularyProvider disponibles

SqlProvider

Chargé de récupérer un vocabulaire via une requête SQL sur une base de données.

LdapVocabularyProvider

Chargé de récupérer un vocabulaire via des recherches Ldap (le contenu des vocabulaires même sera sous forme de VCard).

OOVRemoteVocabularyProvider

Chargé de récupérer un vocabulaire en appelant un autre module ORI-OAI-Vocabulary distant.

XmlStaticVocabularyProvider

Chargé de récupérer un vocabulaire depuis un fichier statitique sur le système (dans le classpath).

XmlEditedVocabularyProvider non implémenté

Chargé de récupérer un vocabulaire qui est directement éditable/modifiable par des utilisateurs depuis un module Ori-Oai-Workflow configuré pour

> ⚠ (à étudier ...)

OriIndexerVocabularyProvider

Chargé de récupérer un vocabulaire dynamiquement depuis un module ORI-OAI-Indexing.

MergingVocabularyProvider

Utilise dom4j pour fusionner des vocabulaires en fonction d'un xpath spécifique au standard Vdex. VdexMergingService est donc capable de fusionner une liste de vocabulaires en un seul.

AlphabetVocabularyProvider

Tri et catégorise (optionnel) par ordre alphabétique un vocabulaire. En fonction d'un paramètre treeDeep, les catégories seront données avec une profondeur donnée :

```
....
                        <category id="A">
                            ....
                        </category>
                        <category id="B">
                            ....
                        </category>
```

```
                                    ....
```

Catégoriser des vocabulaires permet par exemple de lister dans le moteur de recherche les auteurs sous une forme de type Annuaire (liste des premières lettres des auteurs en premier niveau, puis listes des auteurs en deuxième niveau par exemple).

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Implémentation

On utilise au mieux EsupCommons et donc les différentes technologies proposées par celui-ci : Spring, XFire, LdapService, dom4j, ...

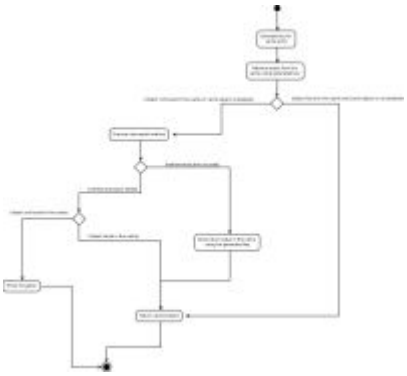
Cache

Comme dit plus haut, on utilise un système de cache EhCache.

L'implémentation utilisée est dérivée de celle donnée dans le projet Spring Modules. En effet, on a gardé du Spring Module Cache la possibilité de mettre en oeuvre un cache ehCache performant et astucieux uniquement via les fichiers de configurations spring tout en adaptant une politique de cache plus spécifique et astucieuse dans notre contexte d'utilisation.

Dans spring module, la politique de mise en cache via l'AOP et donc via les fichiers de configuration ressemble à cela [cf [le
tutoriel Declarative Caching Services for Spring page 2|http://
dev2dev.bea.com/pub/a/2006/05/declarative-caching.html?page=2]] :

Ce que l'on a implémenté pour Ori-Oai-Vocbulary est un peu différent : nous avons un service de cache (qui s'applique de la même façon via AOP) qui permet de ne recherger le cache que si la méthode arrive à fonctionner cela permet de préserver l'ancien cache du vocabulaire des VCards si le Ldap est indisponible par exemple .... Allié aux fonctionnalités de persistence de EhCache sur disque entre 2 lancements de JVM, on obtient une solution générale qui accroit la disponibilité de ce module initialement sensible aux interruptions de services de l'ensemble des éléments interne et externe au Système d'Information auxquels il est relié : Ldap, Base Sql, module de Vocabulaire distant ...



Installation

This page last changed on Jan 05, 2009 by ycolmant@univ-valenciennes.fr.

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Changements version 1.1.0

- Certaines propriétés apparaissant entre crochet et en majuscule peuvent soit être éditées manuellement, soit modifées par la *configuration centrale du quick-install*
- passage des vocabulaires au format VDEX
- modifications des paramètres des provider dynamiques

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

ORI-OAI-commons-quick-install : Installation rapide de ORI-OAI

Composant fortement recommandé pour l'installation de ORI-OAI

Description

Depuis la version 1.1 de ORI-OAI, ce composant permet une installation et une prise en main beaucoup plus rapide du projet avec une configuration centralisée des paramètres de configuration et de déploiement les plus importants. Le principe est de n'éditer qu'un seul fichier de configuration qui est partagé par tous les modules de ORI-OAI. La mise en place de l'outil se fait donc plus rapidement sans avoir à ouvrir et à connaître le fonctionnement de chaque fichier de configuration.

Voir la documentation technique

Ce mode d'installation permet une installation rapide de l'ensemble en saisissant tous les paramètres généraux aux modules dans un seul fichier de configuration.
Vous pouvez toutefois compléter votre installation et vos paramétrages par la suite sans aucun soucis.

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Configurations

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Configurations minimales

Dans la configuration "par module", un fichier build.properties doit être créé à partir du fichier init-build.properties et éditer pour changer les parmètres **en gras** de la section 2):

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


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

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



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

# Cette partie du fichier doit être mise à jour avant
# la première utilisation de votre application dans votre environnement

deploy.home=[PATH_TOMCAT_VOCABULARY]/webapps

#Nom de ditribution de l'application - distribution name
app.name.deploy=[CONTEXT_VOCABULARY]

custom.recover.files=
#conf/properties/domain/domain.xml \
#conf/properties/static_vocabularies/unitvocab/languages.xml \
#conf/properties/static_vocabularies/unitvocab/keywords.xml

# indicate here the directory containing custom old static vocabularies
# (used ONLY when you call ant upgrade.to.vdex is called)
vocab.upgrade.dir=[VDEX_UPGRADE_DIR]
```

- *PATH_TOMCAT_VOCABULARY* est le chemin du Tomcat où doit s'executer le module
- **CONTEXT_VOCABULARY** est le contexte auquel sera accéder le module, et correspond au nom du répertoire de déploiement dans Tomcat
- **VDEX_UPGRADE_DIR** facultatif, sert à indiquer un répertoire contenant d'anciens vocabulaires (d'avant la version VDEX), afin de l'utiliser avec la target ANT upgrade.to.vdex (voir section "Migration des vocabulaires statiques")

Le fichier principal de configuration du module et qui doit permettre de mettre en place rapidement un module Ori-Oai-Vocabulary fonctionnel est le fichier conf/properties/main-config.properties.

Les paramètres **entre crochet** sont à modifier à la main, ou seront modifiés par la configuration centrale du quick-install

```
# ldap [ldap.xml]
# WARNING: you should modify ldapVocabulary.xml TOO (config vcard)
ldap.url=ldap://[LDAP_ETABLISSEMENT]:[PORT_LDAP_ETABLISSEMENT]
ldap.username=
ldap.password=
ldap.base=[LDAP_BASE_DN]
ldap.people.searchBase=[VOCABULARY_LDAP_PEOPLE_SEARCH_BASE]
ldap.people.objectClassValue=[VOCABULARY_LDAP_PEOPLE_OBJECTCLASS_VALUE]
ldap.people.uid=[VOCABULARY_LDAP_PEOPLE_UID]
ldap.people.filter=[VOCABULARY_LDAP_PEOPLE_FILTER]
ldap.group.searchBase=[VOCABULARY_LDAP_GROUP_SEARCH_BASE]
ldap.group.objectClassValue=[VOCABULARY_LDAP_GROUP_OBJECTCLASS_VALUE]
ldap.group.uid=[VOCABULARY_LDAP_GROUP_UID]
ldap.group.filter=[VOCABULARY_LDAP_GROUP_FILTER]


# exceptions [exceptionHandling.xml]
exceptions.recipientEmail=[SMTP_ADMINISTRATOR_MAIL]


# smtp [smtp.xml]
smtp.smtpFromAddress.address=[SMTP_ADMINISTRATOR_MAIL]
smtp.smtpFromAddress.personal=[SMTP_ADMINISTRATOR_NAME]
smtp.smtpInterceptAddress.address=[SMTP_ADMINISTRATOR_MAIL]
smtp.smtpInterceptAddress.personal=[SMTP_ADMINISTRATOR_NAME]
smtp.smtpServer.host=[SMTP_ETABLISSEMENT]
smtp.smtpServer.port=25


# lifeTime for cache in seconds
# WARN : if you change this value, you must clean the last cache (vocabularyServiceCache.data and
 vocabularyServiceCache.index)
# from your tmp directory [so that the new value of cache.lifeTime is used]
cache.lifeTime=3600


# indexing [indexingVocabulary.xml]
indexing1.wsdlDocumentUrl=http://[HOST_INDEXING]:[PORT_INDEXING]/[CONTEXT_INDEXING]/xfire/
IndexingService?WSDL
indexing1.lookupServiceOnStartup=false


# OriOaiVocabulary [domain.xml]
remoteVocabulary1.wsdlDocumentUrl=http://vocabulary.ori-oai.org/xfire/OriVocabularyService?WSDL
remoteVocabulary1.lookupServiceOnStartup=false
remoteVocabulary2.wsdlDocumentUrl=http://www.unit.eu/ori-oai-vocabulary-vdex/xfire/
OriVocabularyService?WSDL
remoteVocabulary2.lookupServiceOnStartup=false

# Edited vocabulary [domain.xml]
editor.home.override=properties/ori_vocabularies/override
editor.home.edited=properties/ori_vocabularies/edited
```

Voici la description de ces paramètres :

PATH_TOMCAT_VOCABULARY

Racine du serveur Tomcat sur lequel est déployé ori-oai-vocabulary

HOST_VOCABULARY

Nom de domaine de la machine sur laquelle est déployée ori-oai-vocabulary

PORT_VOCABULARY

Port du serveur Tomcat par lequel est appelé ori-oai-vocabulary

CONTEXT_VOCABULARY

Nom du contexte choisi pour le déploiement de ori-oai-vocabulary

VOCABULARY_LDAP_PEOPLE_SEARCH_BASE

le subdn de la branche contenant les individus

VOCABULARY_LDAP_PEOPLE_OBJECTCLASS_VALUE

l'ObjectClass utilisé pour les individus

VOCABULARY_LDAP_PEOPLE_UID

l'attribut d'un individu désignant son uid

VOCABULARY_LDAP_PEOPLE_FILTER

un filtre permettant de filtrer les individus Ã exploiter/lister vcard

VOCABULARY_LDAP_GROUP_SEARCH_BASE

le subdn de la branche contenant les groupes

VOCABULARY_LDAP_GROUP_OBJECTCLASS_VALUE

l'ObjectClass utilisé pour les groupes

VOCABULARY_LDAP_GROUP_UID

l'attribut d'un groupe désignant son uid

VOCABULARY_LDAP_GROUP_FILTER

un filtre permettant de filtrer les groupes à exploiter/lister vcard

VOCABULARY_LDAP_PROVIDER_PEOPLE_ORG

Paramètre qui permet de pré-remplir le champ ORG d'une vcard d'une personne dans un vocabulaire LDAP

VOCABULARY_LDAP_PROVIDER_PEOPLE_URL

Paramètre qui permet de pré-remplir le champ URL d'une vcard d'une personne dans un vocabulaire LDAP

VOCABULARY_LDAP_PROVIDER_GROUP_ORG

Paramètre qui permet de pré-remplir le champ ORG d'une vcard d'un groupe dans un vocabulaire LDAP

VOCABULARY_LDAP_PROVIDER_GROUP_URL

Paramètre qui permet de pré-remplir le champ URL d'une vcard d'un groupe dans un vocabulaire LDAP

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Configuration avancée

Afin de personnaliser la configuration des vocabulaires, vous devrez également modifier les fichiers présents dans le répertoire conf/properties/domain/ , par exemple le fichier conf/properties/domain/ ldapVocabulary.xml pour configurer le vocabulaire des VCards issues du Ldap.

Concernant la réalisation du vocabulaire des VCards via le LDAP, notez qu'à ce jour le service permettant de récupérer ces vocabulaires réalise un certain nombre de requêtes consécutives sur votre LDAP. Ce nombre correspond au nombre de personnes que vous rapatriez. Suivant la configuration de votre LDAP et le nombre de personnes rapatriées, votre serveur LDAP peut donc subir une forte charge lors de la création de votre vocabulaire.

Depuis la version 1.1.0, vous pouvez adoucir ces requêtes LDAP en jouant sur plusieurs paramètres (voir section 1.2.1 "Modifications de configuration" pour plus de détails)

Vous devez également filtrer les résultats via un filtre ldap, ce paramètre correspond à la propriété ldapFilter du bean spring peopleLdapLocalProvider qui est déclarée dans conf/ properties/domain/ldapVocabulary.xml.

Ces vocabulaires vont être utilisés dans l'éditeur de métadonnées pour la saisie des vcards via un système d'autocomplétion. Si le vocabulaire est trop gros, il se peut que cela pose des problèmes à l'éditeur (il faudra notamment lui allouer énormément de RAM), l'autocomplétion risque de ne pas être très fonctionnelle. Rapatrier les 40.000 étudiants et personnels de votre établissement n'est donc pas recommandé d'où la présence de ce filtre.

Pour réaliser une configuration avancée, les fichiers de configuration "métier" de ce module sont les fichiers placés dans le répertoire conf/properties/domain/. Ces fichiers sont des fichiers de configuration Spring et correspondent à la paramétrisation des différents Services décrits dans les spécifications donc notamment et surtout les fournisseurs de vocabulaires ("provider"). Les configurations vont vous permettre d'ajouter de nouveaux vocabulaires, de modifier pour certains leur construction, d'en générer de nouveaux en catégorisant ou fusionnant ceux existants, etc. Ces fichiers de configuration sont très verbeux mais les configurations en elles-mêmes restent relativement simples et compréhensibles (et reposent entièrement sur Spring).

Pour ajouter de nouveaux vocabulaires, vous pouvez : * déclarer un nouvel import (tag import) d'un fichier xml dans domain.xml

```
<import resource="monVocabulaire.xml" />
```

- créer ce fichier monVocabulaire.xml en copiant/collant le fichier oriVocabulary.xml par exemple : vous supprimez tous les beans et en recréez d'autres en prenant exemple sur les différents bean de type "Provider" : XmlStaticVocabularyProvider, SqlProvider, LdapVocabularyProvider, etc cf la partie "spécifications"

Notez que le vocabularyService récupère les vocabulaires en appelant successivement (dans l'ordre) les ProviderManager définis dans domain.xml. Vous pouvez donc redéfinir un vocabulaire distant proposé par les remoteVocabularyService (1 et 2) en déclarant localement un nouveau vocabulaire (avec l'identifiant qui correspond au vocabulaire que vous voulez redéfinir).

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Passage à VDEX (version 1.0 à 1.1)

Lors du passage à VDEX, deux aspects doivent être pris en compte par rapport à une installation antérieure, les **modifications de configuration** et la **migration des vocabulaires statiques**.

Modifications de configuration

La configuration des providers dynamiques a changée, et certains paramètres ont été ajoutés.

La propriété **validIndex** permet de choisir si un terme du vocabulaire peut être utilisé pour effectuer une recherche thématique. Cette propriété est positionnée avec une valeur par défaut selon les types de vocabulaires. Au niveau générique, cette propriété est positionné par défaut à "true", mais ce comportement varie ensuite en fonction du type réél du provider comme indiqué par la suite.

Liste des providers impactés associés à leurs fichiers de configuration : * properties/domain/domain.xml : ajout d'un bean **PrefixManager** Ce bean gère les associations entre préfixes, espaces de noms et URL des schémas utilisés :

```
<bean id="prefixManager"
                class="org.orioai.vocabulary.domain.beans.PrefixManager" lazy-init="false">
                <description>
                                Mapper between prefixes, namespaces and schemaLocations
                                This singleton bean MUST be loaded BEFORE provider declarations.
                </description>
                <property name="prefixSchemas">
                        <map>
                                <entry key="vdex" value="http://www.imsglobal.org/xsd/imsvdex_v1p0
 http://www.imsglobal.org/xsd/imsvdex_v1p0.xsd" />
                         <entry key="orioai" value="http://www.ori-oai.org/static/xsd/orioaivocab
 http://www.ori-oai.org/static/xsd/orioaivocab.xsd" />
                         <entry key="xforms" value="http://www.w3.org/2002/xforms http://
www.w3.org/MarkUp/Forms/2002/XForms-Schema.xsd" />
                 </map>
                </property>
        </bean>
```

- properties/domain/alphabetVocabulary.xml :
  - - sortFieldXpath : cible ce qui va servir au classement alphabétique
    - treeDeep : profondeur voulu pour l'arbre alphabétique (1 = A,B,C... 2 = A(AA,AB,AC...) etc...)
    - validTreshold : seuil à partir duquel les termes seront "recherchables" dans le moteur de recherche (défaut =1, signifiant que les racines A...Z ne seront pas recherchables, mais n'auront qu'un finalité d'agencement).
      Exemple :

```
<bean id="people_vcard"
 class="org.orioai.vocabulary.domain.providers.AlphabetVocabularyProvider" init-
method="init">
                <property name="elementToSortXpath" value="/vdex:vdex/vdex:term"/>
                <property name="sortFieldXpath" value="vdex:caption/
vdex:langstring"/>
                <property name="treeDeep" value="1" />
                <property name="validTreshold" value="1" />
                <property name="vocabularyProvider" ref="mergingPeople"/>
```

```
        </bean>
```

- properties/domain/ldapVocabulary.xml :
  Certains paramètres ont été ajouté pour adoucir les requêtes LDAP :
  - - **useSoftRequests** : defaut true; si a false, aucun découpage en sous-requêtes
    - **delayBetweenRequests** : défaut 10; délai minimal en millisecondes entre les requêtes LDAP
    - **hashDeep** : défaut 2; profondeur du découpage en sous-requêtes alphabétiques (26 puissance n)
    - **hashField** : défaut cn ; champ LDAP utilisé comme critère de sous requête
- properties/domain/ldapVocabulary.xml : Un paramètre **mergeMode** a été ajouté pour gérer la façon dont doivent fusionner deux termes ayant le même identifiant. * fusionSubTerms : both terms subterms are gathered under the same (first) term. * replaceByLast : last term replaces preceding term. * replaceByFirst(default) : first term remains unchanged, all following terms are skipped.

Migration des vocabulaires statiques

Si vous avez créés des vocabulaires statiques dans la version 1.0.x, vous avez le choix entre deux processus de migration :

- 1. première méthode :
  Cas : Vos vocabulaires étaient situés dans l'arborescence*
  properties/ori_vocabularies*
  Déplacez simplement vos vocabulaires personnalisés qui étaient dans l'arborescence properties/ori_vocabularies dans l'aborescence properties/old_ori_vocabularies et lancez la target :

```
ant vdex
```

Cela va convertir vos anciens vocabulaires en VDEX et les replacera dans l'abrorescence properties/ori_vocabularies

- 2. deuxième méthode :
  Cas : Vos vocabulaires étaient situés dans un autre répertoire.
  Vous devez indiquer leur emplacement dans le paramètre **vocab.to.upgrade** du fichier build.properties, et lancer la target :

```
ant upgrade.to.vdex
```

Cela va convertir vos anciens vocabulaires en VDEX et les replacera dans l'arborescence properties/ori_vocabularies

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Déploiement

Une fois configuré le build.properties et les fichiers de configuration décrits ci dessus, la target deploy permet de déployer simplement l'application.

ant deploy

Vous pouvez alors démarrer l'applciation en lancant le Tomcat concerné.

Vous pouvez alors vérifier que votre Web Service répond bien * en pointant l'url sur le wsdl du WEB Service du vocabulaire (point d'entrée de ce module pour les autres modules ORI-OAI) : pour UNIT, cela donne http://www.unit.eu/ori-oai-vocabulary/xfire/OriVocabularyService?WSDL

- en consultant les vocabulaires via le module ori-oai-md-editor utilisé dans ori-oai-workflow (cf la documentation de ori-oai-workflow)

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

Déploiement d'un plugin au module de vocabulary

Afin d'installer un plugin dans le module ori-oai-vocabulary, on doit le déployer dans le même Tomcat que ce dernier. Pour cela, on doit avoir déployé au préalable le projet ori-oai-vocabulary. Pour déployer le plugin, on doit donner le chemin d'accès au projet ori-oai-vocabulary (aux sources) dans l'arborescence du projet (configuration de la variable ori-oai-vocabulary-src.dir du fichier build.properties, par exemple : ../ori-oai-vocabulary si les 2 projets sontt au même niveau). Ensuite, on déploie le plugin grâce au ant deploy, ce qui déploie le plugin dans le même deploy.home que le module vocabulary.

Utilisation

This page last changed on Jan 05, 2009 by ycolmant@univ-valenciennes.fr.

Documentation à venir ...

Aspects pratiques

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

- Test
- Encodage

Test

This page last changed on Jan 05, 2009 by ycolmant@univ-valenciennes.fr.

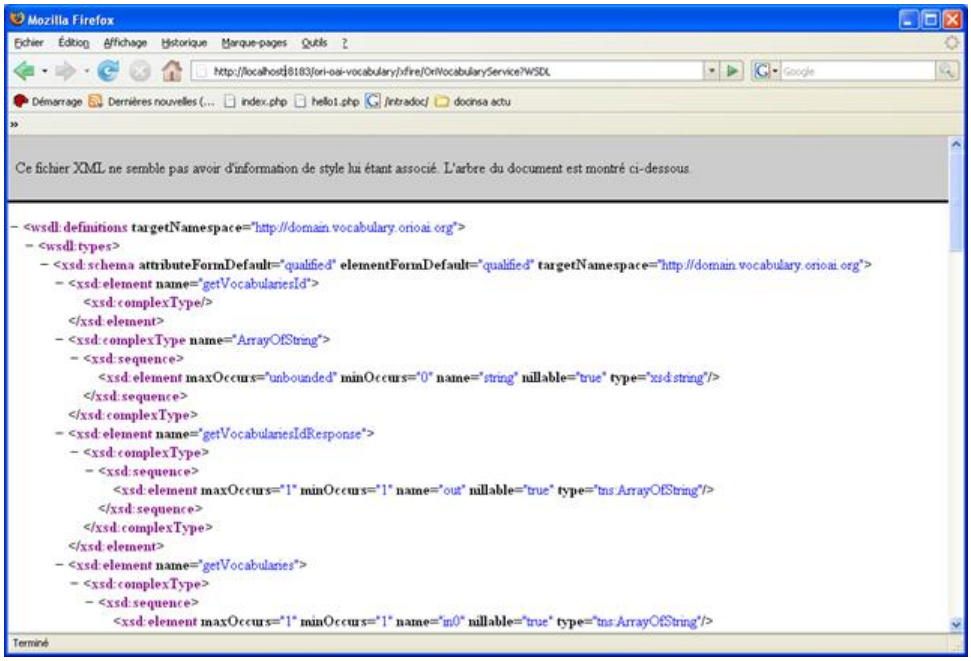
Pour tester le module ori-oai-vocabulary, lancer la commande :

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

Accédez à l'URL:

```
http://[HOST_INSTALL]:8183/ori-oai-vocabulary/xfire/OriVocabularyService?WSDL
```

Vous devriez obtenir l'affichage suivant :



Afin de visualiser les différents vocabulaires, Vous pouvez également accéder à l'URL **http:// [HOST_INSTALL] :8183/ori-oai-vocabulary/** pour accéder à l'interface de consultation de tous les vocabulaires disponibles.

A partir de cette URL, on a la liste des différents vocabulaires existants, et on peut cliquer sur "Show" pour visualiser le contenu de chacun des vocabulaires.

This page last changed on Oct 14, 2008 by vibonamy@univ-rennes1.fr.

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 une bonne option également :

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