Moissonnage de données OAI

Décembre 2007

Haouaou Moussa
Service NTICE

UNIVERSITÉ DE GENÈVE
DIVISION INFORMATIQUE

Table des matières

1 Contexte du projet

Dans le cadre de mon travail de diplôme à l'École Supérieure d'Informatique de Gestion, j'ai effectué un stage de 5 mois à l'université de Genève au service des nouvelles Technologies de l'Information et de la Communication .A l'échéance de ces 5 mois de stage, monsieur Pierre-Yves Burgi m'a proposé indépendamment de ce travail de diplôme de continuer le projet pour une autre durée de 4 mois.

Ce rapport représente donc, le résultat de 9 mois de travail global .Une première partie a été consacré au harvesting (Collecte des métadonnées) et l'autre partie au développement et à la transformation d'une base de données MYSQL en un entrepôt OAI. Avant d'aller dans le bref du sujet, voici la présentation du service, de l'outil et de l'environnement de ce travail.

1.1 Présentation du service

Le service des Nouvelles Technologies de l'Information, de la Communication, et de l'Enseignement (NTICE) dirigé par Monsieur Pierre-Yves Burgi, a pour mission de mettre en place les technologies nouvelles au service de la communauté universitaire. Ses principaux champs d'activités sont:

- Le Web
- L'e-learning
- Le multimédia
- La bibliothèque virtuelle

Avant de présenter le projet bibliothèque virtuelle, c'est-à-dire la partie principale du travail, voici l'explication des différents concepts cités ci dessus:

Gestion du Web

Le service NTICE assure la gestion du serveur institutionnel qui héberge le site Web www.unige.ch ainsi que les applications liées au Web.

Technologies de l'e-learning

Les technologies du domaine de l'e-learning mises à disposition par le service NTICE englobent les plates-formes d'enseignement, la numérisation des cours et leur diffusion sur Internet.

Technologies Multimédia

Les technologies multimédia sur lesquelles le service NTICE œuvre touchent à trois domaines principaux:

- Les outils de communication synchrones (en temps réel) ;
- La numérisation des documents ;
- La transmission des informations.

Bibliothèque virtuelle

La bibliothèque virtuelle de l'Université de Genève, bien que pour l'instant qu'à l'état de projet, a pour objectif :

- de permettre l'accès à l'ensemble des ressources et services documentaires en tout lieu et en tout temps ;
- de façon personnalisée et sécurisée, au travers du portail institutionnel ;
- d'introduire une approche intégrée de recherche et d'accès aux connaissances, provenant de sources multiples dans des formats variés, tout en assurant une qualification de cette information par des professionnels de la documentation ;
- de passer d'une approche traditionnelle centrée sur un catalogue unique à une approche intégrant des ressources distribuées basées sur des formats variés de métadonnées ;
- d'intégrer directement l'accès aux ressources documentaires dans les environnements d'enseignement et d'apprentissage, typiquement les plateformes d'eLearning institutionnelles ;
- de valoriser la production scientifique de l'Université de Genève par la mise en place d'un dépôt institutionnel, qui favorise un libre accès à ces ressources ;
- d'assurer une protection des documents en ligne qui respecte leurs droits d'accès.

1.2 Cahier des charges

Dans ce stage. Il a s'agit de mettre en place un système de HARVEST compatible OAI-PMH. Cela va Impliquer une étude préalable du concept OAI, de sa norme en vigueur et de son utilisation dans le contexte de Fedora.

Un modèle de HARVEST sera ensuite testé sur Fedora, ainsi que sur médiaserver, un server utilisé pour diffuser les documents audio-visuels. La finalité de ce travail est donc, une solution au problème de synchronisation de Base de données bibliographiques, reparties sur une multitude de serveurs, selon un standard bien défini. Voici au final les points principaux :

- Installation du logiciel PKP Harvester et familiarisation avec les différents concepts
- Installation de Fedora (Client et serveur) ;
- Collecte des métadonnées de RERO vers PKP Harvester ;
- Installation des plugins SRU ;
- Mise en œuvre des requêtes SRU ;
- Transformation du fichier XML obtenu à partir de la requête SRU, en fichiers XSL et FOXML ;
- Familiarisation avec les web-services, en particulier SOAP[1] ;
- Ingest de métadonnées dans Fedora ;
- Installation et configuration de PHPOAI2 pour rendre médiaserver compatible OAI ;
- Mise en correspondance des métadonnées médiaserver et Dublin Core ;
- Mise en œuvre des requêtes OAI-PMH à titre de vérification ;
- Collecte et ingest des métadonnées de Mediaserver dans Fedora

[1] SOAP - Simple Object Application est un protocole RPC basé sur XML, autorisant un objet à invoquer des méthodes d'objets situées sur une machine distante.

2 Outils et environnements de travail

2.1 Fedora

La version de Fedora utilisée ici est le 2.2.1. Fedora est construit sur des technologies open source récentes et se conforme a un grand nombre de standards, dont certains définis par le W3C, il se veut le plus évolutif et le plus modulable possible. C'est un logiciel open source qui fonctionne avec le serveur Apache et Tomcat et qui permet la gestion des ressources numériques. Le cœur de programme est écrit en Java. Toutes les fonctionnalités de Fedora sont exposées via un jeu d'interfaces web-services. L'accès à Fedora peut être généré par le les services de l'interface API-A, et l'ajout d'un objet dans le dépôt peut se faire par l'appel à l'interface API-M. Il faut noter qu'à l'intérieure de Fedora, Chaque objet est stocké avec un identifiant unique

Fedora est disponible à l'adresse http://www.fedora.info/download/2.1.1/download.cgi. Deux archives sont listées, un serveur et un client et il est bon de les télécharger toutes les deux.

- L'archive « serveur » contient tout le logiciel, intégré dans son propre conteneur Tomcat et prêt à être exécuté. Tous les scripts de configuration et d'interaction avec Fedora sont disponibles ;
- L'archive « client » contient la console d'administration ainsi que tous les objets de démonstration. Certains scripts, spécifiques à ces objets de démonstration, sont également inclus dans cette archive.

2.2 PHP5

C'est un langage procédural disposant en version 5 de fonctionnalités de modèle objet complètes. En raison de la richesse de sa bibliothèque, on désigne parfois PHP comme une plate-forme plus qu'un simple langage.

2.3 Mysql

C'est un système de gestion de bases de données relationnelles libre. Il est communément utilisé pour des applications dynamiques sur le Web. C'est dans MYSQL que nous avons stocké :

- La table oai_records avec ses métadonnées ;
- La table Harvester avec ses métadonnées ;
- La table Fedora avec ses métadonnées.

2.4 PKP Harvester

La version utilisée est le «harvester-2.0.1 » PKP harvester est un outil qui permet de créer un index consultable des métadonnées à partir d'archives conformes à OAI, élaboré dans le cadre du «Public Knowledge Project » (PKP) de l'Université de la Colombie-Britannique.

PKP Harvester (PKPH) est donc, un logiciel libre qui permet de faire la recherche et la collecte des métadonnées à travers les entrepôts OAI moissonnés.

La version 2.x de PKP Harvester comprend les caractéristiques suivantes:

- Ability to harvest OAI metadata in a variety of schemas (including unqualified DC, the PKP (Open Journal Systems/Open Conference Systems) Dublin Core extension, MODS,

and MARCXM : Aptitude à récolter des métadonnées OAI dans une variété de schémas (y compris les non qualifiés, DC) ;

- Flexible search interface that allows simple searching and advanced searching using crosswalked fields from all harvested archives: Flexibilité d'interface de recherche qui permet une recherche simple et avancées en utilisant tous les champs récoltés des archives. Advanced searching of archives that share the same schema will be possible using fields as defined in the schema : Recherches avancées des archives qui partagent le même schéma sera possible à l'aide des domaines tels que définis dans le schéma. When creating crosswalks for searching, admins can define elements are text, date, or HTML multiple select interface widgets : Lors de la création de passages pour la recherche, les administrateurs peuvent définir les éléments du texte, la date, ou sélectionnez l'interface HTML de multiples gadgets ;
- Ability to perform granular harvesting using setSpec and timest : Possibilité d'effectuer les récoltes sélectives ;
- Ability to perform post-harvest and pre-indexing filtering/normalization on metadata : Capacité à effectuer après la récolte et avant l'indexation le filtrage / normalisation des métadonnées.

Pour Installer PKP Harvester, nous avons besoins de la base de données MYSQL, du logiciel PHP et du serveur Apache.

2.5 Plugin SRU

Il définit une interface pour l'interaction avec des bases de données distantes, sans qu'il n'y ait besoin de connaître la logique interne de la base de données consultée pour trouver des résultats. Les requêtes sont en CQL (Contextual Query Language) et sont envoyées en URL .Les résultats sont transmis en XML.

Paramètres pour les requêtes SRU :

Nom	Obligatoire ou facultatif?	Description
Opération	Obligatoire	La chaîne: 'searchRetrieve'.
Version	Obligatoire	La version de la requête Si le serveur ne peut pas fournir une réponse dans cette version ou moins, alors il doit retourner un diagnostic
Requête	Obligatoire	Contient une requête exprimée en CQL à être traitées par le serveur. Une requête CQL peut se contenir une seule clause de recherche, ou plusieurs clauses de recherche reliés par des opérateurs booléens
StartRecord	Facultatif	La position du premier enregistrement (record).
MaximumRecords	Facultatif	Le nombre maximum de records à être retourné. La valeur doit être de 0 ou plus. La valeur par défaut est fournie si elle n'est pas déterminée par le serveur
RecordPacking	Facultatif	Déterminer la façon dont Les records doivent être empaquetés. Ça peut être En chaîne de caractère simple ou en XML La valeur par défaut est du "XML
RecordSchema	Facultatif	Le schéma à travers lequel les records peuvent être

		retournés
RecordXPath (La version 1.1)	Facultatif	Une expression XPath qui doit être appliquée aux documents avant leur restitution. Elle doit être appliquée par rapport au schéma fourni par le paramètre RecordSchema, Xpath est donc un langage utilisé pour traiter les documents XML

Paramètres pour les réponses SRU :

Les réponses SRU sont obtenues en XML avec les paramètres suivants.

Nom	Type	Obligatoire ou facultatif?	Description
version	*xsd:string*	Obligatoire	La Version de la réponse. Cela doit être égal à la version demandée par le client.
numberOfRecords	*xsd:integer*	Obligatoire	Le nombre de records correspondant à la requête. Si la requête échoue cela doit être égal à 0.
resultSetId	*xsd:string*	Facultatif	L'identificateur des ensembles des résultats crée à travers l'exécution d'une requête
resultSetIdleTime	*xsd:integer*	Facultatif	Le nombre de seconde entre la création et la suppression d'un résultat
records	*sequence of <record>*	Facultatif	Series de records
nextRecordPosition	*xsd:integer*	Facultatif	La position qui suit le résultat final
diagnostics	*sequence of <diagnostic>*	Facultatif	Parfois, les choses tournent mal. Dans ces cas, le serveur est obligé de signaler que quelque chose n'allait pas, par l'envoi d'un dossier de diagnostic expliquant ce qui s'est passé. Diagnostic sont retournés dans un schéma très simple, qui ne dispose que de trois éléments, 'URI', 'précisions' et' message ".
extraResponseData	*<xmlFragment>*	Facultatif	Information retournées par le serveur
echoedSearch RetrieveRequest	*<echoedSearch RetrieveRequest>*	Facultatif	Message «echo » au client dans un simple formulaire XML

Exemple d'une requête SRU qui nous renvoie les métadonnées dont le créateur est David :

http://localhost/harvester2.0.1/index.php/sru?version=1.1&operation=searchRetrieve&query=creator = David

La réponse est :

```
<searchRetrieveResponse>
<version>1.1</version>
<numberOfRecords>1</numberOfRecords>
   <records>
         <record>
               <recordSchema>info:srw/schema/1/dcv1.1</recordSchema>
               <recordPacking>xml</recordPacking>
               <recordData>
                     <srw_dc:dc xmlns:dc="http://purl.org/dc/elements/1.1/"
                           xmlns:srw_dc="http://www.loc.gov/zing/srw/">
                           <dc:title>Introduction to the study of literature</dc:title>
                           <dc:creator>Spurr David</dc:creator>
                           <dc:publisher>Faculté des lettres</dc:publisher>
                           <dc:date>2006-03-14</dc:date>
                           <dc:type>cours</dc:type>
                           <dc:format>audio</dc:format>
                           <dc:identifier>AN3-SPUR-2005-2006-14.3</dc:identifier>
                             <dc:source>http://129.194.9.169/audio/AN3-SPUR-2005-2006-
                             14.3.mp3</dc:source>
                           <dc:relation>AN3-SPUR-2005-2006</dc:relation>
                           <dc:coverage>2007-10-31</dc:coverage>
                           <dc:rights>Spurr David</dc:rights>
                     </srw_dc:dc>
                     </recordData>
               <recordPosition>1</recordPosition>
         </record>
   </records>
</searchRetrieveResponse>
```

2.6 Protocole OAI-MPH

(Open Archives Initiative's Protocol for Metadata Harvesting) C'est un protocole qui permet de faciliter l'échange de données entre des fournisseurs de données et fournisseurs de service. Il est basé sur le principe de collecte des métadonnées, opération nommée moisson (harvesting) dans les spécifications du protocole et s'appuie sur :

- Un jeu de requêtes (questions ou verbes) très simple ;
- Un accès via HTTP ;
- Les données sortant du serveur OAI-PMH sont en XML ;
- Une utilisation du schéma XML Dublin Core.

L'interrogation en OAI-PMH

Un moissonneur pour interroger un entrepôt OAI utilise une requête sous la forme d'une URL classique construite en deux parties : L'url de base de l'entrepôt et la requête d'interrogation à proprement parler. Cette seconde partie se construit à partir de différents verbes prévus par le protocole et listés dans le tableau suivant :

Verbe	Rôle	Arguments
GetRecord	Récupération d'un enregistrement donné.	• identifier • metadataPrefix
Identify	Informations sur l'entrepôt de données.	Aucun
ListIdentifiers	Récupère la liste des identifiants disponibles.	• from : date de début • until : date de fin • metadataPrefix • set • resumptionToken
ListMetadataFormats	Demande la liste des formats de métadonnées disponibles. Sans paramètres tous les formats disponibles pour au moins un item sont retournés. Avec le paramètre identifier, n'est retourné que les formats disponibles pour l'item concerné	• identifier
ListRecords	Retourne une liste d'enregistrements correspondant aux différents paramètres (dates, ensemble) demandés.	• from : date de début • until : date de fin • metadataPrefix • set • resumptionToken
ListSets	Demande la liste des ensembles disponibles sur un entrepôt. La réponse peut être sur plusieurs pages.	♦ resumptionToken

Verbes et attributs

	From *(depuis)*	**Until** *(jusqu'à)*	**Metadata prefix** *(format)*	**Identifier** *(identificateur)*	**Set** *(Nomenclature, thématique)*
Identify					
ListMetadataFormat				(f)	
ListSets					
ListIdentifiers	f	f	o		f
ListRecords	f	f	o		f
GetRecords			o	o	
exemple	*2007-08-20*	*2007-08-22*	*oai_dc*	*:csd-00008082 (version 1)*	*Mathematics/ Science*

o obligatoire
f facultatif

Glossaire :

Item
L'item est l'objet documentaire qui décrit une ressource. L'item est générateur d'enregistrements de métadonnées qui pourront être échangés grâce au protocole OAI, à la seule condition qu'à chaque item puisse être associé un identifiant unique au sein de son entrepôt.

Identifiant (identifier)
L'identifiant est une clé décrivant un document de manière univoque. Chaque entrepôt doit pouvoir associer un identifiant unique à chaque item qu'il contient.

Enregistrement (record)
Le concept d'enregistrement désigne une représentation XML concrète de métadonnées issues d'un item (objet documentaire), dans un format compatible avec le protocole OAI. Plusieurs enregistrements (formats DC non qualifié, DC qualifié, MarcXML…) peuvent être produits pour un même item.

Entrepôt OAI (repository)
Base de métadonnées constituée par un fournisseur de données. Les métadonnées y sont disponibles dans différents formats afin de répondre à différents types de demandes.

Exemple de requête OAI qui liste tous les métadonnées qui ont le format OAI_DC, se trouvant dans l'entrepôt RERO :

http://doc.rero.ch/oai2d.py?verb=ListMetadataFormats

Voici la réponse OAI obtenue :

```
<?xml version="1.0" encoding="UTF-8" ?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
    http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
    <responseDate>2007-10-16T13:19:26Z</responseDate>
  <request verb="ListMetadataFormats">http://doc.rero.ch/oai2d.py/</request>
   <ListMetadataFormats>
      <metadataFormat>
        <metadataPrefix>oai_dc</metadataPrefix>
        <schema>http://www.openarchives.org/OAI/1.1/dc.xsd</schema>
       <metadataNamespace>http://purl.org/dc/elements/1.1/</metadataNamespace>
    </metadataFormat>
   <metadataFormat>
      <metadataPrefix>marcxml</metadataPrefix>
```

```
    <schema>http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd</schema>
    <metadataNamespace>http://www.loc.gov/MARC21/slim</metadataNamespace>
  </metadataFormat>
 </ListMetadataFormats>
</OAI-PMH>
```

Elle affiche :

- La date et l'heure pendant laquelle la requête été effectuée qui est « responseDate » ;
- Les différentes sortes de métadonnées (Dublin Core et Marc21) ;
- Le mot clé (verb) de la requête et l'URL de l'entrepôt OAI(http://doc.rero.ch/oai2d.py);
- Le préfixe des métadonnées (oai_dc).

2.7 Formats Dublin Core

Le Dublin **Core** est un schéma de métadonnées générique qui permet de décrire des ressources numériques ou physiques et d'établir des relations avec d'autres ressources. Il comprend officiellement 15 éléments de description formels :

Elément	Elément (anglais)	Commentaire
1. Titre	Title	Titre principal du document
2. Créateur	Creator	Nom de la personne, de l'organisation ou du service à l'origine de la rédaction du document
3. Sujet	Subject	Mots-clefs, phrases de résumé, ou codes de classement
4. Description	Description	Résumé, table des matières, ou texte libre. Raffinements : table des matières, résumé
5. Éditeur	Publisher	Nom de la personne, de l'organisation ou du service à l'origine de la publication du document
6. Contributeur	Contributor	Nom d'une personne, d'une organisation ou d'un service qui contribue ou a contribué à l'élaboration du document
7. Date	Date	Date d'un évènement dans le cycle de vie du document
8. Type de ressource	Type	Genre du contenu

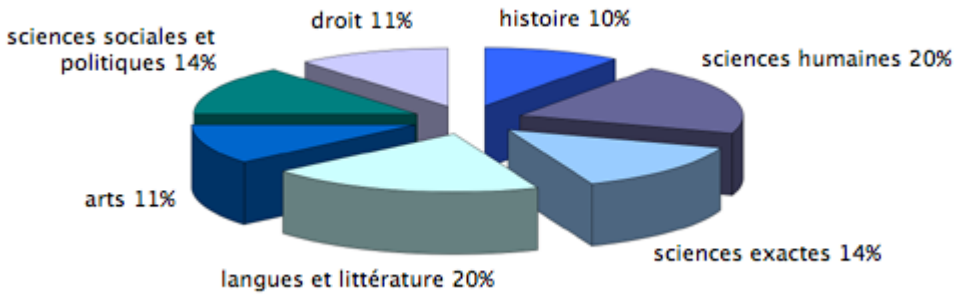
9. Format	Format	Type MIME, ou format physique du document
10. Identifiant de la ressource	Identifier	Identificateur non ambigu : il est recommandé d'utiliser un système de référencement précis, afin que l'identifiant soit unique au sein du site.
11. Source	Source	Ressource dont dérive le document : le document peut découler en totalité ou en partie de la ressource en question. Il est recommandé d'utiliser une dénomination formelle des ressources, par exemple leur URI
12. Langue	Language	
13. Relation	Relation	Lien avec d'autres ressources. De nombreux raffinements permettent d'établir des liens précis, par exemple de version, de chapitres, de standard, etc.
14. Couverture	Coverage	Couverture spatiale (point géographique, pays, régions, noms de lieux) ou temporelle
15. Droits	Rights	Droits de propriété intellectuelle, Copyright, droits de propriété divers

2.8 Qu'est-ce que RERO

RERO, acronyme de REseau ROmand, désigne le réseau des bibliothèques de Suisse occidentale. Né, voilà plus de vingt ans, de la volonté de coopération de plusieurs grandes bibliothèques romandes, RERO regroupe aujourd'hui la plupart des bibliothèques universitaires, publiques et patrimoniales des cantons de Genève, Fribourg, Jura, Neuchâtel, Valais et Vaud. RERO gère un catalogue collectif d'environ 215 bibliothèques au service des 35'000 étudiants des 4 universités romandes (Genève, Fribourg, Lausanne et Neuchâtel) et de quelque 150'000 lecteurs.

Plus important réseau de Suisse par le volume des collections signalées et le nombre de bibliothèques membres, RERO repose sur une volonté commune de ses partenaires de développement d'une politique documentaire basée sur la collaboration, la mise en commun de ressources et services, le partage de compétences, de moyens financiers et techniques. Le catalogue collectif, basé sur le partage du travail de catalogage, la bibliothèque numérique RERO DOC et l'utilisation d'un même logiciel de gestion informatisée de bibliothèques, sont des résultats de ce projet collectif de coopération.

Un catalogue collectif d'une grande richesse documentaire
Accessible en ligne depuis ce site ou à l'url http://opac.rero.ch, le catalogue collectif RERO donne accès à quelque 3.7 millions de références bibliographiques permettant de localiser plus de 6 millions d'ouvrages dans un rayon géographique de 100 km. Ce catalogue permet également de localiser 222'400 périodiques dont 3'169 périodiques électroniques.



Véritable catalogue encyclopédique, il donne accès non seulement à des références bibliographiques mais aussi à des documents numériques tels que thèses et revues électroniques, images, collections et textes patrimoniaux numérisés. Plusieurs centaines de langues différentes sont présentes dans le catalogue. Avec 37 % du fonds, les documents en français sont les plus nombreux suivis par ceux en anglais (23 %), en allemand (21 %), en italien (6 %) et en espagnol (2 %). Les documents référencés dans la base RERO sont pour la plupart librement disponibles.

3 Collecte des métadonnées

3.1 Installation et configuration de PKP Harvester

Pkp harvester est un logiciel qui nécessite PHP, mysql et apache pour son fonctionnement. Il ne nécessite pas beaucoup de configuration. La seule chose qu'il faut faire, dans le cas où nous avons ce genre de message :

- config.inc.php is writable (optional): NO
- cache/ is writable: NO
- cache/t_cache/ is writable: NO
- cache/t_compile/ is writable: NO
- cache/_db is writable: NO

Nous avons donc traité le répertoire **harvester -2.0.1** de façon récursive en utilisant la commande **« chmod -R 777 harvester -2.0.1 ».** Si tout se passe bien, on aura :

- config.inc.php is writable (optional): Yes
- cache/ is writable: Yes
- cache/t_cache/ is writable: Yes
- cache/t_compile/ is writable: Yes
- cache/_db is writable: Yes

Ensuite, il faut créer un compte d'utilisateur cet utilisateur deviendra l'administrateur du logiciel et aura l'approche complète au système. Les comptes d'utilisateur supplémentaires **peuvent** être créés après l'installation. Il faut aussi créer un compte d'utilisateur pour la base de données et achever l'installation.

À partir d'ici, on pourra faire la collecte des métadonnées-,à la seule condition d'avoir une adresse URL de base de l'entrepôt.

3.2 Collecte des métadonnées

Pour la collecte des métadonnées, il faut d'abord : (1) Créer des archives pour y mettre les métadonnées collectées, et (2) se connecter à l'URL de base de l'entrepôt. Par exemple, celui de RERO est : http://doc.rero.ch/oai2d.py/ . Pour pouvoir faire la collecte. À partir du moment que l'adresse de URL est vérifiée, on a la possibilité de faire aussi des moissonnages sélectives.

3.3 Installation de Plugins SRU et requêtes

L'installation de plugins SRU ne nécessite pas de configuration spécifique. La seule chose qu'il faut faire, c'est d'ouvrir le fichier **SRWSearch** et d'augmenter le nombre de record qui est à : maximumRecords = 200, par le nombre d'enregistrement que nous voulons à l'affichage. Il faut faire la même chose pour le fichier **RequestType**

Avant de lancer la requête, il fait vérifier que le module XSL est installé dans Apache. Sinon, nous aurons le problème de parsing.

Différence entre SRU et SRW

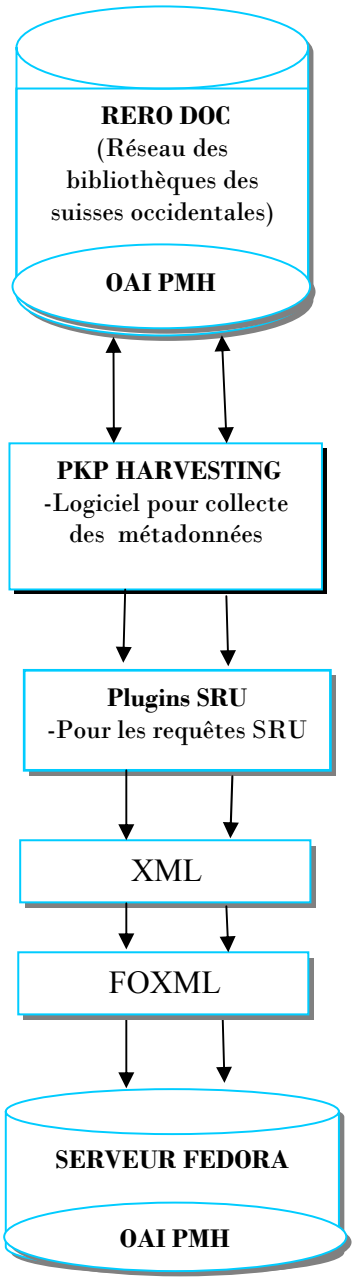
SRU et SRW utilisent le protocole HTTP pour l'échange de requêtes. Plus spécifiquement, SRU permet de faire circuler des requêtes à l'intérieur des URLs en utilisant l'architecture REST (Representational State Transfer) là où SRW, variante de SRU, fait transiter les requêtes sous SOAP (Simple Object Access Protocol). Pour ce qui concerne REST et SOAP, le premier permet de spécifier l'encodage d'une requête au sein même d'une URL, alors que le second encapsule la requête sous XML. Partant, REST ne fonctionne qu'avec le protocole HTTP, là où SOAP autorise en plus l'usage de nombre d'autres protocoles (email, telnet).

SRU/SRW utilisent tous les deux les mêmes instructions, qui permettent l'expression de la requête et de la réponse à cette requête. Les trois opérations principales sont « explain », « scan », et « searchRetrieve »

La réponse aux requêtes SRU et SRW est un fichier XML.

Une fois ce Fichier XML obtenu, nous n'avons qu'à faire les transformations XML et à faire l'ingest dans Fedora. Le schéma suivant explique le cheminement à suivre.

Voici le schéma illustrant le moissonnage et l'ingest des métadonnées de RERO avec PKP harvester et le Plugin SRU



Explication des différentes étapes du Schéma ci-dessus

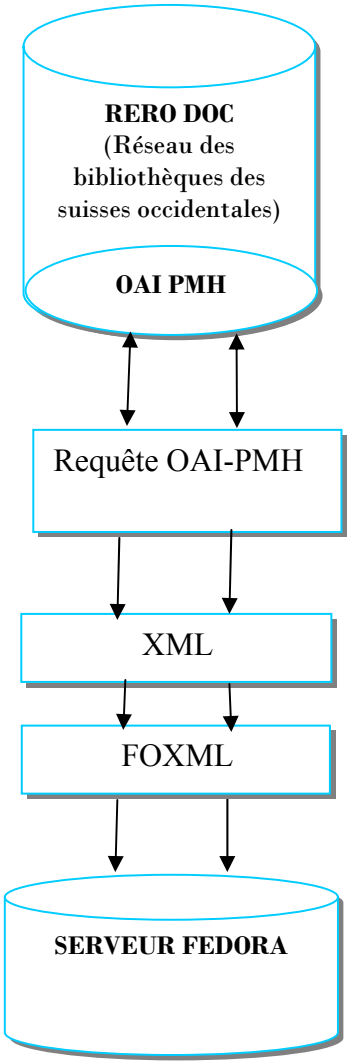
La première partie consiste à faire la collecte des métadonnées qui se trouve dans RERO pour les stocker dans la base de données MYSQL de PKP HARVESTER. Une fois ces métadonnées collectées, nous feront des requêtes SRU pour obtenir une page XML. Après nous transformeront ce XML en FOXML pour les exporter dans Fedora. Nous feront la même chose pour la base de données médiaserver, à la seule différence qu'il faut d'abord la rendre compatible OAI avant de faire la collecte et l'ingest.

4 Collecte avec OAI-PMH

(Sans PKP Harvester et sans Plugins SRU)

Vu que la requête OAI –PMH nous renvoie une page XML, nous avons constaté qu'il n'est pas forcement nécessaire s de passer par le logiciel PKP Harvester et par le Plugins SRU. Nous pouvons donc faire les transformations XSLT et FOXML directement avec cette page XML de OAI-PMH. Le résultat sera le même. Ainsi, nous nous épargnons des multiples installations et configurations.

Schémas



5 Traitement du document XML

5.1 Transformation OaiXsl.java

Le XSL pour e**X**tensible **S**tylesheet **L**anguage ou "langage extensible de feuilles de style" est une recommandation du W3C datant de novembre 1999. C'est donc un standard dans le domaine de la publication sur le Web. Le XSL est en quelque sorte le langage de feuille de style du XML. Un fichier de feuilles de style reprend des données XML et produit la présentation ou l'affichage de ce contenu XML selon les souhaits du créateur de la page.

Le XSL comporte en fait 3 langages :

- Le XSL**T** qui est un langage qui **T**ransforme un document XML en un format, généralement en Html, reconnu par un navigateur.
- Le Xpath qui permet de définir et d'adresser des parties de document XML.
- Le XML Formatter pour "formater" du XML (transformé) de façon qu'il puisse être rendu sur des PCpockets ou des unités de reconnaissance vocale.

Le XSL est dérivé du XML
Le langage XML est un langage de balises dérivé du langage XML. Le XSL reprend donc toutes les règles de syntaxe du XML :

- Les balises sensibles à la casse, s'écrivent en minuscules ;
- Toutes les balises ouvertes doivent être impérativement fermées ;
- Les balises vides auront aussi un signe de fermeture soit <balise/> ;
- Les balises doivent être correctement imbriquées ;
- Les valeurs des attributs doivent toujours être mises entre des guillemets.

En bref, il faut noter que XML est un langage de structuration des données, et non de représentation des données. Ainsi **XSL** est un langage pour effectuer la représentation des données de documents XML. XSL est lui-même défini avec le formalisme XML, cela signifie q'une feuille de style XSL est un document XML bien formé.

Exemple du document XSL

Un document XSL étant un document XML, il commence obligatoirement par la balise suivante :

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

D'autre part, toute feuille de style XSL est comprise entre les balises <xsl:stylesheet ...> et </xsl:stylesheet>.

La balise *xsl:stylesheet* encapsule des balises *xsl:template* définissant les transformations à faire subir à certains éléments du document XML.

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

<xsl:stylesheet
xmlns:xsl="http://www.w3.org/TR/WD-xsl"
xmlns="http://www.w3.org/1999/XSL/Transform"
```

```
result-ns="">

        <xsl:template ... >

        <!-- traitements à effectuer -->

        </xsl:template >

</xsl:stylesheet>
```

5.2 FOXML

Fedora Objet XML:

Un objet dans Fedora est encodé en FOXML format dérivé du XML.

Les composantes de base d'un objet digital dans Fedora sont les suivantes :

- **PID** : l'identifiant unique de l'objet dans Fedora ;
- Propriétés de l'objet : un groupe de métadonnées système décrivant l'objet et qui sont nécessaires à la gestion interne de l'objet par Fedora ;
- Flux de données : chaque flux de données (Datastream) de l'objet contient une donnée précise dans un certain type MIME. Chaque flux individuel représente une caractéristique de l'objet et, lorsqu'on les met tous ensembles, les flux de données définissent complètement l'objet ;
  Par défaut, chaque objet a un flux de métadonnées, encodées au format Dublin Core ;
- Disséminateurs : les disséminateurs correspondent aux différentes vues de l'objet que l'on souhaite mettre à disposition. Là encore, un objet peut avoir un ou plusieurs disséminateurs ;

Pour résumer, on peut dire que FOXML est un format simple de XML compréhensible par Fedora qui décrit directement le modèle numérique des objets à être ingérer.

Exemple de la structure FOXML :

```
<digitalObject PID="uniqueID">
    <!-- there are a set of core object properties -->
    <objectProperties>
            <property/>
            <property/>
            ...
    </objectProperties>
    <!-- there can be zero or more datastreams -->
    <datastream>
            <datastreamVersion/>
            <datastreamVersion/>
            ...
    </datastream>
    <!-- there can be zero or more disseminators -->
```

```
    <disseminator>
            <disseminatorVersion/>
            <disseminatorVersion/>
            ...
    </disseminator>
</digitalObject>
```

6 Développements

Explication de la structure des classes et algorithme

Cette section a pour but de présenter de manière détaillée les différents développements réalisés ainsi que les différents choix effectués.les classes nécessaires pour faire l'ingest des objets dans Fedora sont:

- Une classe qui traite le fichier XML et le découpe selon le nombre de records ;
- Une classe qui traite les PIDs ;
- Une classe qui fait l'ingest des métadonnées dans Fedora ;
- Pour faire l'ingest dans Fedora, nous sommes obligés d'établir une communication SOAP avec ce dernier.

Voici les différentes Classes :

TransformationOaiXsl. Java:

Sert à faire la transformation FOXML, à découper le document XML en plusieurs fichiers selon les nombres de records, facilement ingérables par Fedora.

C'est le programme principal, On doit lui passer comme paramètre le fichier XML de départ qui est la réponse que nous renvoie la requête OAI-PMH. Dans notre cas c'est (oaipmhFinal.xml), le fichier XSL qui est la transformation XSL et FOXML (myxslVersion2) pour qu'il fasse le découpage.

Nous allons donc découper notre fichier FOXML en plusieurs sous fichiers selon le nombre de records, stockés dans un seul répertoire, pour pouvoir ensuite les ingérer facilement dans Fedora.

Voici l'algorithme :

Manipulation de java avec Jdom: pour pouvoir utiliser Jdom, il faut au préalable importer ses librairies. L'algorithme est la suivante :

- Création d'un document Jdom avec comme argument le fichier XML ;
- Récupération du nombre de records ;
- Création d'une liste contenant tous les nœuds de records ;
- Appel de la fonction getPid pour l'attribution de pid à chaque records, avec comme argument le nombre de records et le nom de pid qui est « demo » ;
- Création d'un tableau pour stocker les pids ;
- Création d'un iterator ;

- Appel de la fonction outputXSL () qui sert à afficher les fichiers FOXML ;
- Appel de la méthode IngestFiles pour découper le fichier XML et les stocker dans un seul répertoire ;
- Affichage du document FOXML ;
- On définit un transformer avec la source XSL ;
- On transforme le document JDOMEntree grâce à notre transformer. La méthode transform() prend en argument le document d'entrée associé au transformer et in document de résultat JDOMResult, résultat de la transformation ;
- On crée le fichier FOXML correspondant aux résultats ;
- On stocke les fichiers FOXML crées dans le répertoire « dossierFoxml » ;
- Pour chaque erreur, on affiche la trace.

getPid.java:

Effectue une communication SOAP avec Fedora et attribue les PIDs (identifiants) à chaque record. Comme chaque objet dans Fedora possède son propre PID, nous avons fait un programme qui renvoie les PIDs disponibles dans Fedora et les attribue à chaque record automatiquement. L'utilisation de API-M et API_A de Fedora est nécessaire ici. Voici l'algorithme:

6.1 SOAP

SOAP - Simple Object Application est un protocole RPC basé sur XML, autorisant un objet à invoquer des méthodes d'objets situées sur une machine distante. C'est un protocole de transmission de messages. Il définit un ensemble de règles pour structurer des messages qui peuvent être utilisés dans de simples transmissions unidirectionnelles, mais il est particulièrement utile pour exécuter des dialogues requête-réponse RPC (Remote Procedure Call). Il n'est pas lié à un protocole particulier mais HTTP est populaire. Il n'est pas non plus lié à un système d'exploitation ni à un langage de programmation, donc, théoriquement, les clients et serveurs de ces dialogues peuvent tourner sur n'importe quelle plate-forme et être écrits dans n'importe quel langage du moment qu'ils puissent formuler et comprendre des messages SOAP.

6.2 APIs

Plusieurs API différentes regroupent les méthodes d'accès à Fedora :

L'API de consultation du dépôt (API-A) :

Elle met à disposition neuf méthodes qui permettent d'interagir avec Fedora de manière à rechercher des objets, afficher leur profil, leur historique, la liste de leurs flux de données ou de leurs disséminateurs. On peut donc, en combinant les appels à ces méthodes, créer un site de consultation du dépôt Fedora.

L'API de gestion du dépôt (API-M) :
Les méthodes mises à disposition par l'API-M permettent l'administration de tout le dépôt. On peut créer, modifier ou supprimer des objets ou composants d'objets. Le client d'administration fourni avec Fedora utilise d'ailleurs cette API pour interagir avec ce dernier

***Voici* l'algorithme :**

- Etablissement d'une communisation Soap avec Fedora ;
- Pour chaque erreur, afficher la trace ;
- Appel de la méthode getNextPID qui retourne le prochain PId disponible dans Fedora ;
- Appel de la méthode NonNegtiveInteger qui permet de donner les entiers positifs pour ces PIDs ;
- Affichage de trace en cas d'erreurs.

fedoraIngest.java :

Fait l'ingest des objets dans Fedora.

Nous allons encore établir une communication SOAP avec Fedora. Pour l'ingest, nous aurons besoin du document FOXML, de son format et de son encodage. Il est impératif d'utiliser les API-M de Fedora pour appeler certaines méthodes.

Voici* l'algorithme *:

- Communication soap avec Fedora ;
- Création d'une fonction de l'ingest avec comme argument le document, le format et le message log ;
- Pour que le fichier soit dans un format correct, il faut appeler la méthode getPrettyFormat, ainsi que mettre la bonne locale, ici ISO-8859-1 ;
- Création de la fonction filesIngest avec comme argument le document FOXML, son encodage et son format.

Encodage des caractères :

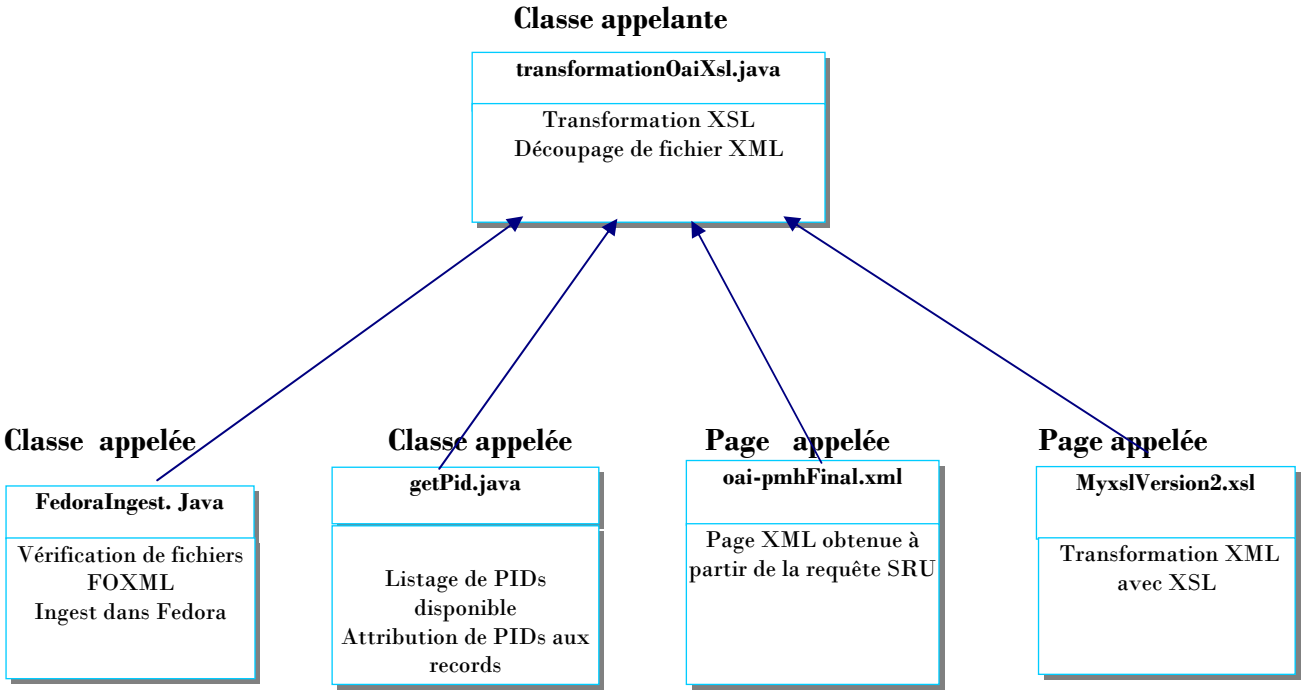
Différence entre encoding="iso-8859-1" et encoding="UTF-8"

En ISO-8859-1, le caractère « e » avec accent aigue est représenté sur un seul octet.

En UTF-8, il est représenté sur 2 octets. UTF-8 accepte de nombreux symboles et lettres de tellement de langues du monde.

En bref, ISO-8859-1 permet d'encoder la plupart des langages d'Europe de l'ouest. UTF-8 permet d'encoder tout Unicode (c'est à dire grosso modo la totalité des langages couramment écrits dans le monde).

Diagramme définissant la liaison entre les différentes classes



Note :

Les fichiers XML, XSLT, FOXML et les codes sont transmis en annexe.

7 Comment rendre harvestable Mediaserver

Mediaserver :

La base de données mediaserver de l'université de Genève est une base de données qui comprend **différents** types de documents audiovisuels produits ou diffusés par l'Université de Genève tels que cours, cycles de conférences, documentaires, films etc. Pour ce projet nous avons traité 8277 métadonnées.

Pour rendre Harvestable, cette base de données, nous avons utilisé PHPOAI2 qui est un outil de PHP permettant de transformer une base de données MYSQL en un entrepôt OAI et qui nécessite MYSQL et Apache pour son fonctionnement.

7.1 Installation et configuration phpoai2

Pour configurer PHPOAI2, il faut ouvrir le fichier de configuration qui est **« oai2/oaidpconfig.php »**. L'installation comme telle est très simple et n'a demandé que de configurer la variable qui pointe sur les outils d'accès aux bases de données depuis PHP (PEAR).

Ensuite, il faut configurer l'accès à notre base de données MYSQL. Il faut pour cela modifier des valeurs de variables telles que ***$DB_HOST***, ***$DB_NAME***, etc. En tout cas, Le fichier de configuration est très explicite à ce sujet.

Enfin, il faut modifier des variables concernant notre entrepôt OAI spécifique. Ce sont des variables telles que **$repositoryName** et surtout ***$METADATAFORMATS***.

Pour vérifier que tout est bien installer, il faut lancer la requête OAI et voir le résultat.

Exemple d'une requête OAI qui liste tous les records ayant un préfixe « oai_dc » :

http://localhost/phpoai2-1.8.0/oai2.php?verb=ListRecords&metadataPrefix=oai_dc

Note: Il ne faut pas oublier d'installer la dernière version de PEAR si cela est nécessaire.

Qu'est que PEAR?

PEAR (pour *PHP Extension and Application Repository*) est une bibliothèque de scripts PHP. Tous les scripts déposés dans PEAR respectent un certain nombre de règles qui les rend portables (indépendants de la plate-forme) et réutilisables.

L'avantage est qu'en l'installant soi-même, on arrive à comprendre déjà le fonctionnement, et on peut choisir la version voulue.

7.2 Correspondance Mediaserver - Dublin Core

Comme les données de médiaserver ne sont pas en Dublin Core, il faut créer une table intermédiaire, avec comme champs, les différentes éléments de Dublin Core et ensuite faire une requête pour copier les métadonnées de la table médiaserver (newcorrespondances) dans notre table intermédiaire « oai_records » .car, on ne doit en aucun cas modifier la base médiaserver.

Correspondance

MEDIASERVER	DUBLIN CORE
title	dc_title
author	dc_creator
keywords	dc_subject
faculté	dc_publisher
département	dc_publisher
year / date	dc_date
type	dc_type
media	dc:format
name	dc_identifier
url	dc_source
collection	dc_relation
expiryDate	dc_coverage
author	dc_rights

Données d'exploitation (Pas visible au moment de l'affichage)

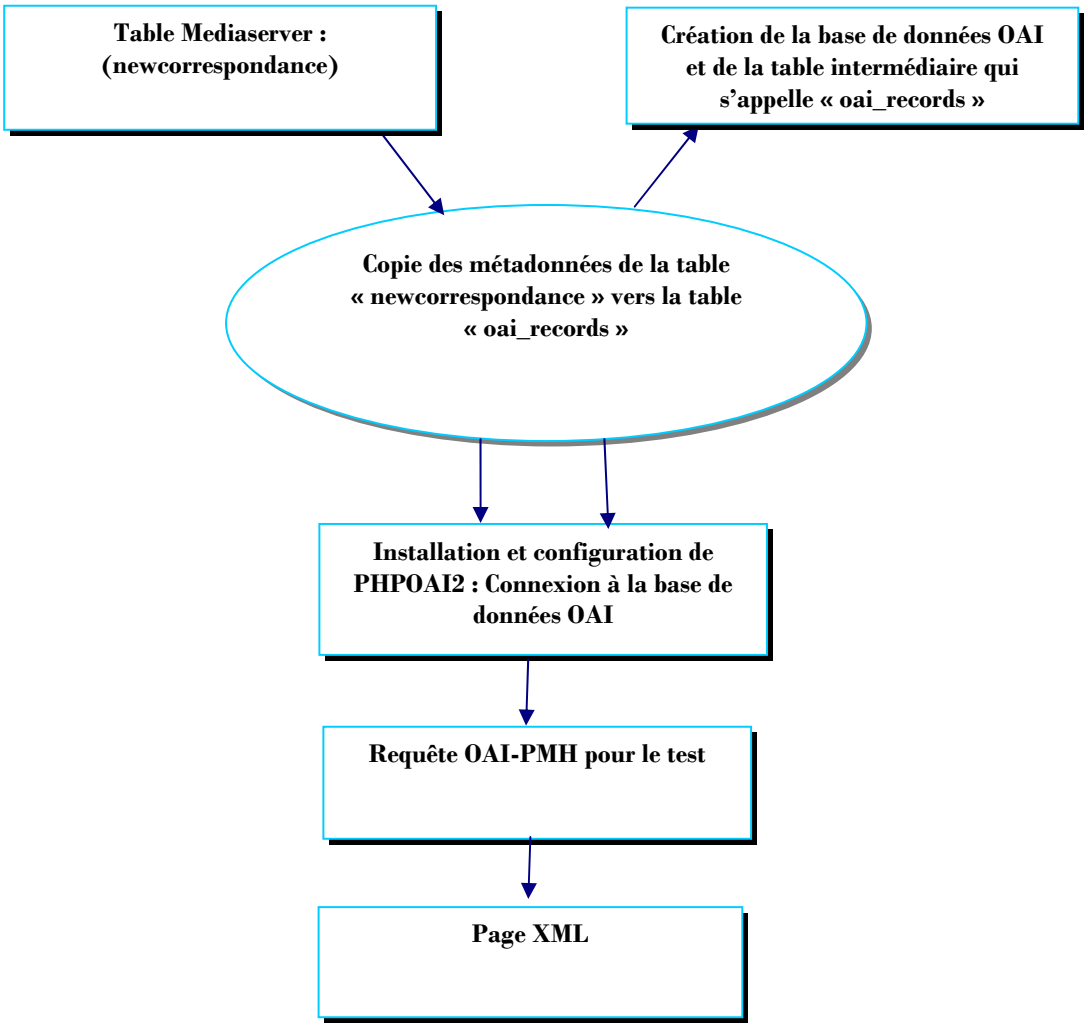
Champs mediaserver	explications
provider	Champ décrivant l'origine de la donnée
enterdate	Champ décrivant la date actuelle
oai_identifier	Pour afficher l'identifier au moment de la requête OAI-PMH
oai_set	Pour afficher les différents regroupements de métadonnées. Ex : thèses, mathématiques
datestamp	Pour afficher la date et l'heure de l'exécution de la requête
deleted	Pour afficher l'état de métadonnées (actif ou inactif)
academicYear	Année académique
control	champ obsolète initialement utilisé pour mentionner le contrôle par une personne de la ressource dans le catalogue.
fastfoward	champ obsolète mentionnant la présence d'un fastforward
storyboard	champ obsolète mentionnant la présence d'un storyboard
duration	champ facultatif mentionnant la durée du fichier
isVisible	booléen pour empêcher l'affichage de certains documents
prive	champ pour distinguer un document privé d'un document public
subTitle	Sous titre
ms_modif	Date de modification

Mise à jour

En ce qui concerne la mise à jour des métadonnées nous avons le champ «ms-modif »qui, avec une requête SQL, nous renverra les dernières modifications qui ont eu lieu dans la table Mediaserver (newcorrespondance). Donc, si nous lançons la requête OAI, nous n'aurons que les informations modifiées. Ceci nous évitera d'avoir la redondance des métadonnées

Première étape suivie

Avec création d'une table intermédiaire



7.3 Création de la table intermédiaire

Pour préserver les métadonnées mediaserver de toute modification, nous avons crée une table intermédiaire pour copier toutes les métadonnées afin de les traiter.

Exemple de scripts **pour la création de la table oai_reorcds avec des champs Dublin Core**

```
CREATE TABLE `oai_records` (
`serial` INT( 11 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`provider` VARCHAR( 255 ) NULL ,
`url` VARCHAR( 255 ) NULL ,
`enterdate` DATETIME NULL ,
`oai_identifier` VARCHAR( 255 ) NULL ,
`oai_set` VARCHAR( 255 ) NULL ,
`datestamp` DATETIME NULL ,
`deleted` ENUM( 'false', 'true' ) NOT NULL ,
`dc_title` VARCHAR( 255 ) NULL ,
`dc_creator` VARCHAR( 255 ) NULL ,
`dc_subject` VARCHAR( 255 ) NULL ,
`dc_description` VARCHAR( 255 ) NULL ,
`dc_contributor` VARCHAR( 255 ) NULL ,
`dc_publisher` VARCHAR( 255 ) NULL ,
`dc_date` DATE NULL ,
`dc_type` VARCHAR( 255 ) NULL ,
`dc_format` VARCHAR( 255 ) NULL ,
`dc_identifier` VARCHAR( 255 ) NULL ,
`dc_source` VARCHAR( 255 ) NULL ,
`dc_language` VARCHAR( 255 ) NULL ,
`dc_relation` VARCHAR( 255 ) NULL ,
`dc_coverage` VARCHAR( 255 ) NULL ,
`dc_rights` VARCHAR( 255 ) NULL
) ENGINE = MYISAM ;
```

7.4 Insertion des métadonnées :

Le but c'est de copier les métadonnées de mediaserver dans notre nouvelle table « oai_records ».

Exemple de scripts **:**

```
INSERT INTO `oai_records`
(`dc_title`,`dc_creator`,`dc_subject`,`dc_description`,`dc_publisher`,`dc_date`,`dc_type`,
`dc_identifier`,`dc_relation`,`dc_coverage`)
SELECT `title`, `author`,`keywords`,`keywords`,`faculty`,`date`,`type`,
`name` ,`collection`, `expiryDate` FROM `newcorrespondances` WHERE 1
```

Avec cette requête, nous avons notre table "oai_records" remplie avec comme des champs, des éléments de Dublin Core et comme données les métadonnées de médiaserver.

8 Rendre Harvestable Mediaserver sans table intermédiaire

Vue le problème des mises à jour en ce qui concerne la copie des métadonnées de la table source vers la table intermédiaire, et vue la place occupée dans la mémoire par cette manipulation , Nous avons trouvé une solution qui consistera à rendre mediaserver Harvestable sans toutefois passer par cette table intermédiaire: Il suffit de faire la correspondance mediaserver Dublin Core dans le fichier de configuration qui est « records_dc.php » et lui donner directement le nom de la base et de la table source

Si on se confère à la correspondance Mediaserver Dublin Core, voici la partie de l'affectation des champs mediaserver dans ce fichier records_dc.php de PHPOAI2

```
$output .= xmlrecord($record['title'], 'dc:title', '', $indent);
$output .= xmlrecord($record['author'],'dc:creator', '', $indent);
$output .= xmlrecord($record['keywords'], 'dc:subject', '', $indent);
$output .= xmlrecord($record['faculty'], 'dc:publisher', '', $indent);
$output .= xmlrecord($record['date'], 'dc:date', '', $indent);
$output .= xmlrecord($record['type'], 'dc:type', '', $indent);
$output .= xmlrecord($record['media'], 'dc:format', '', $indent);
$output .= xmlrecord($record['name'], 'dc:identifier', '', $indent);
$output .= xmlrecord($record['url'], 'dc:source', '', $indent);
$output .= xmlrecord($record['collection'], 'dc:relation', '', $indent);
$output .= xmlrecord($record['expiryDtae'], 'dc:coverage', '', $indent);
$output .= xmlrecord($record['author'], 'dc:rights', '', $indent);
```

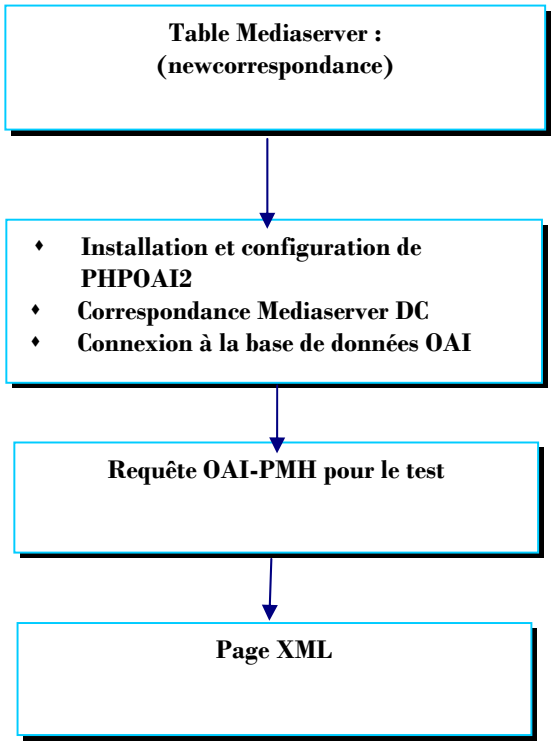
Une fois correspondance faite, il ne faut pas oublier de modifier la Clé primaire dans le fichier oaidp-congif.php qui est $SQL ['id_column'] = 'serial '; par la clé primaire de Mediaserver qui est $SQL ['id_column'] = 'myuid';

C'est dans ce même fichier qu'on modifiera :

- Les informations sur la base où sont stockées les métadonnées de Mediaserver qui sont:
  $DB_HOST = 'localhost';
  $DB_USER = 'root';
  $DB_PASSWD = 'pass';
  $DB_NAME = 'oaidb';
- le nom de la table de mediaserver qui est :$SQL ['table'] = 'newcorrespondances';
- L'identifier de mediaserver qui est : $SQL ['identifier'] = 'url';
- L'encodage voulue qui est : $charset = "utf-8".

Deuxième étape suivie

Sans table intermédiaire



9 Ingest avec Batch Tool de VITAL

Vital: VTLS *Information Technology for Advanced Learning*

C'est un logiciel pour la gestion des archives numériques telles que des documents, des textes entiers, des vidéos ou des images, etc.

Vital batch Tool:

C'est un outil de Vital qui, grâce aux fichiers de configuration, nous permet, à partir de certaines lignes de commande, de faire l'ingest des objets en masse dans Fedora

Les étapes à suivre **:**

Pour utiliser le batch Tool de Vital, nous avons six principales étapes à suivre:

- Création du fichier de configuration ;
- Création du model d'objet appelé par le fichier de configuration ;
- Création du feuille de style XSLT pour transformer XML en XSLT ;
- Création des datastreams. Il faut noter qu'avec Batch Tool de vital, on a aussi la possibilité de créer des datastreams qui contiennent le PDF ;
- Spécification de l'entrepôt vers lequel on veut ingérer les métadonnées ;
- Lancement du script de l'ingest.

Exemple du fichier XML pour faire l'ingest. Ce fichier est tiré à partir de la requête OAI-PMH et affiche un enregistrement de Mediaserver :

```
<?xml version="1.0" encoding="UTF-8"?>
<OAI-PMH xmlns="http://www.openarchives.org/OAI/2.0/"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
       http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
 <responseDate>2007-11-28T10:01:50Z</responseDate>
 <request verb="ListRecords" metadataPrefix="oai_dc">http://localhost/phpoai2-
1.8.0/oai2.php</request>
 <ListRecords>
  <record>
  <header>
   <identifier>oai:aName.org:http://129.194.9.169/audio/AN3-CONR-2005-2006-
15.3.mp3</identifier>
   <datestamp>2002-01-01T00:00:00Z</datestamp>
  </header>
  <metadata>
   <oai_dc:dc
    xmlns:oai_dc="http://www.openarchives.org/OAI/2.0/oai_dc/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
    http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
    <dc:title>Cours général III et IV : Histoire contemporaine</dc:title>
    <dc:creator>CONRAD Christoph</dc:creator>
    <dc:publisher>Faculté des lettres</dc:publisher>
    <dc:date>2006-03-15</dc:date>
    <dc:type>cours</dc:type>
    <dc:format>audio</dc:format>
    <dc:identifier>AN3-CONR-2005-2006-15.3</dc:identifier>
    <dc:source>http://129.194.9.169/audio/AN3-CONR-2005-2006-15.3.mp3</dc:source>
    <dc:relation>AN3-CONR-2005-2006</dc:relation>
    <dc:rights>CONRAD Christoph</dc:rights>
   </oai_dc:dc>
  </metadata>
 </record>
 <record>
</ListRecords>
</OAI-PMH>
```

Configuration :

- Création du fichier de configuration : « Config-Media_dc.xml » et chemin d'accès opt/vtls/vital/batch/examples/unigetests/testMediaserver/Config-Media_dc.xml
- Création de model d'objet : « model_mediaserver-dc.xml » et chemin d'accès opt/vtls/vital/batch/examples/unigetests/testMediaserver/model_mediaserver-dc.xml

Exemple de fichier de configuration :

```
<BatchConfig>
        <!-- Model Definition File -->
         <ModelFile>examples/unigetests/testMediaserver-dc/model_mediaserver-
        dc.xml</ModelFile>

        <!-- Location of file(s) to import (Directory/Filename/URL) -->
        <ImportTarget>examples/unigetests/testMediaserver-dc/2objetsMediaserver-
        dc.xml</ImportTarget>
         <XmlRecordsXPath>//*/record</XmlRecordsXPath>

         <!-- Fedora Connection Information -->
        <Host>vital-test.unige.ch</Host>
        <Port>8080</Port>
        <Username>fedoraAdmin</Username>
         <Password>fedoraAdmin</Password>

         <!-- Miscellaneous Options -->
         <TestOnly>false</TestOnly>
        <FoxmlOnly>false</FoxmlOnly>
         <OutputDirectory>.</OutputDirectory>

        <!-- VITAL Datastream File -->
        <VitalFile>xml/vital.xml</VitalFile>
</BatchConfig>
```

Exemple de fichier d'objet Model avec Datastream :

```
<? xml version="1.0" encoding="UTF-8"?>
        <ObjectModel>
            <Datastream id="DC" type="main">
                <State>A</State>
                <MIMEType>text/xml</MIMEType>
                <Label>Exemple record DC depuis mediaserver</Label>
                 <ControlGroup>X</ControlGroup>
            </Datastream>
        </ObjectModel>
```

Le script à lancer pour faire l'ingest :

```
./import.pl --config=examples/unigetests/testMediaserver-dc/Config-Media-dc.xml
```

La réponse obtenue si l'ingest est fait :

```
Log file initiated at logs/import-1141414372.log.
Start.
Successfully Connected to VTLS Training Repository (Version 3.0)
  - Adding datastream (DC).

Created FOXML Object at (./foxml-1.fox).
-----------------------------------------
Processing file (examples/unigetests/testMediaserver-dc/Config-Media-dc.xml)
```

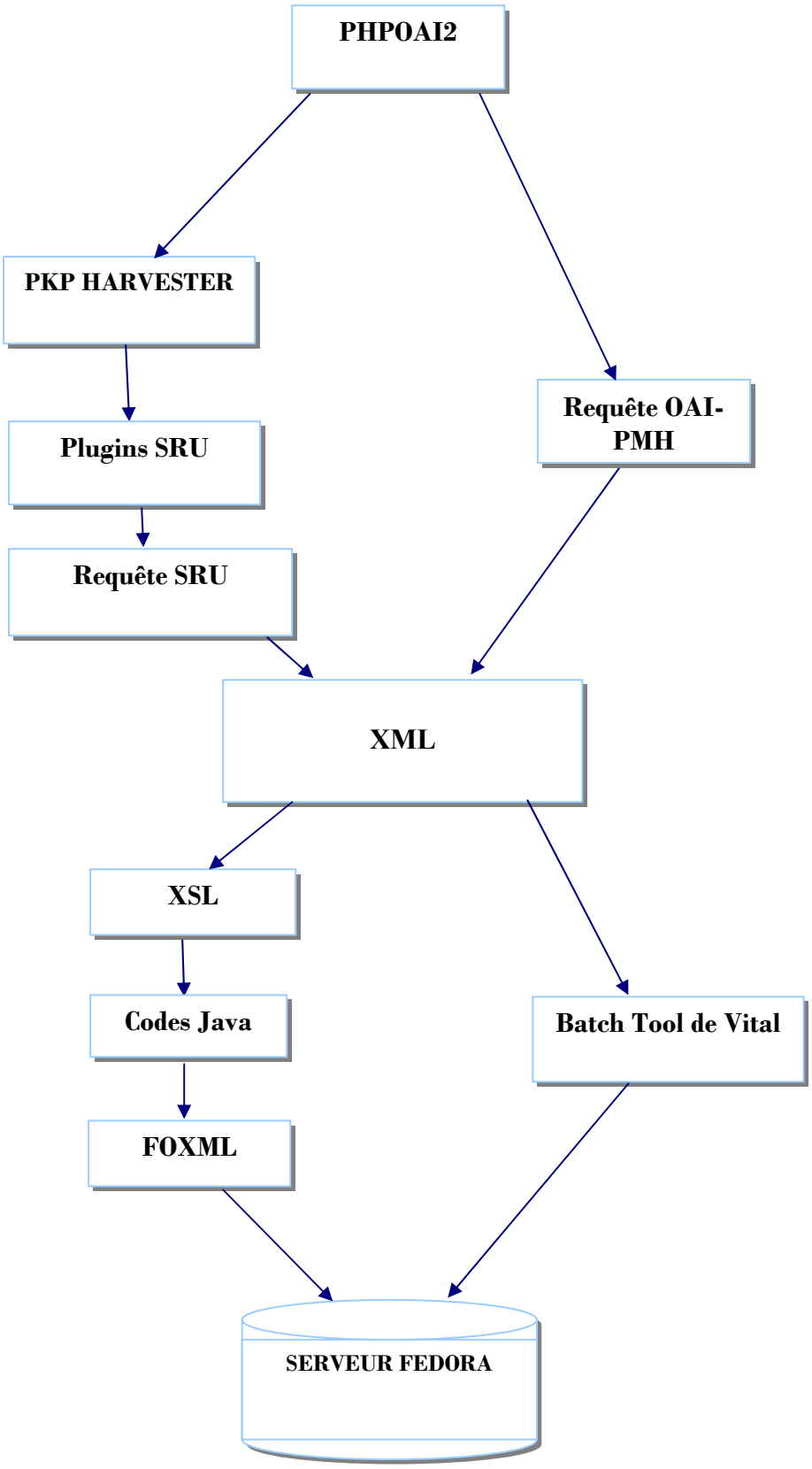
- Adding datastream (DC).

Created FOXML Object at (./foxml-2.fox).

Created 1 FOXML record(s) at (.).
Done.

Si l'ingest n'est pas fait, nous avons des messages d'erreurs qui sont très explicites. Ce qui facilite le débogage.

10 Synthèses des solutions



10.1 Constat sur la synthèse

Au début, nous avons constaté que nous n'avons pas besoin de PKP Harvester, ni de Plugins SRU pour obtenir une page XML, Car la requête OAI-PMH nous renvoie aussi une Page XML que nous pouvons transformer et Exporter dans Fedora.

Ensuite, nous avons Vu qu'avec le Batch Tool de Vital, nous n'avons pas besoin de faire nous même le traitement du fichier XML, ni de développer en Java pour faire L'ingest.
Donc avec le cheminement de la droite, nous gagnons en efficacité et en gain de temps

11 Difficultés

- Difficultés à trouver les documentations nécessaires ;
- Il était difficile de trouver les documents sur le logiciel PKP HARVESTER Il existe très peu d'informations La seule solution était de s'inscire au forum ;
- Difficultés à faire le harvesting avec les métadonnées de Mediaserver. Lors de la copie des métadonnées dans ma table Intermédiaire, je n'ai pas tenu compte du champ « URL » qui ne faisait pas partie des éléments de Dublin Core.Et je n'avais qu'un seul record après le Harvest . Donc, il fallait mettre ce champ « URL » ;
- Difficultés à trouver la syntaxe pour la requête SRU qui me permettrait de sélectionner tous mes records à la fois. J'ai contourné ce problème en utilisant le champ « date ». Ce problème n'a pas été complètement résolu ;
- Difficultés à trouver les nameSpaces spécifiques lors de la transformation XSL ;
- Difficulté au niveau de la syntaxe JAVA XML ;
- Difficulté à trouver la syntaxe de la requête OAI-PMH qui, pourrait me limiter le nombre de records lors de l'affichage. Ce problème n'a pas été résolu.

12 Conclusions

Les concepts sur lesquels repose le logiciel PKP Harvester ainsi que les technologies sur lesquelles il est construit permettent à chacun d'utiliser ce logiciel selon ses besoins, pour la consultation, la recherche, la collecte, l'ingest des métadonnées et bien d'autres fonctions. Malgré les multiples fonctions que ce logiciel pourrait nous offrir, il paraît contournable en ce qui concerne la collecte des métadonnées. Si la finalité du travail consiste à obtenir une page XML et de faire les transformations FOXML pour pouvoir faire l'Ingest dans Fedora, il serait préférable d'utiliser les réponses de la requête OAI-PMH qui renvoient aussi une page XML. Ainsi, nous nous épargnons des multiples installations et configurations

OAI-PMH est donc un protocole incontournable en ce qui concerne les archives ouvertes car il permet la centralisation, l'agrégation de l'ensemble des informations, il permet également de fournir un accès unique pour les recherches. Il est indispensable en ce qui concerne le moissonnage des données.

Il faut néanmoins noter que c'est un protocole pour lire des données. Il n'y a pas d'instruction (de verbe) dans le protocole pour insérer des documents ou des notices dans une archive ouverte. En conséquence, l'initiative de mise à jour ne peut venir que du fournisseur de services. Le moissonneur vient « quand il veut » pour faire la mise à jour. En conséquence, l'utilisateur n'a pas de garantie quant à la fraîcheur de l'information collectée

En ce qui concerne l'ingest des métadonnées dans Fedora. Ne serait- il pas préférable d'utiliser le Batch Tools de Vital qui non seulement nous fourni des paramètres pour les transformations XML, nous fourni aussi les lignes de commande pour l'ingest, que de passer par le code en Java qui nécessite au préalable que nous fassions nous même nos transformations XML, développions nous même nos méthodes avant de faire l'ingest ?

13 Remerciements

Merci tout d'abord à Pierre-Yves BURGI de m'avoir permis d'effectuer ce stage. Merci également à toute l'équipe du NTICE pour leur accueil et leurs coups de main, en particulier à Messieurs Jan MELICHAR, Olivier JEANIN, et autres membres du service NTICE.

14 Sources

Documentation générale Fedora

- **Site** officiel **Fedora**
  http://www.fedora.info/download/2.1.1/userdocs/

- **Wiki Fedora**
  http://www.fedora.info/wiki
  Documentations écrite par différents utilisateurs
  http://www.fedora.info/wiki/index.php/Fedoradocumentation_from_users

- **API de Fedora pour l'ingest**
  http://www.fedora.info/definitions/1/0/fedora-relsext-ontology.rdfs

- **Installation de Fedora**
  http://www.fedora.info/download/2.1.1/userdocs/distribution/installation.
  Html

- **PKP Harvester**
  http://pkp.sfu.ca/?q=harvester

- **Forum PKP harvester**
  http://pkp.sfu.ca/support/forum/

- **API Java 2 SE 5.0**

  http://java.sun.com/j2se/1.5.0/docs/api/

- **API Java EE SD 5.0**

http://java.sun.com/javaee/5/docs/api/

- **Dublin Core**
  http://dublincore.org/

- **Description des éléments**
  http://dublincore.org/documents/dcmi-terms/

- **OAI**
  http://www.openarchives.org/

- **OAI-PMH**
  Open Archives Forum
  http://www.oaforum.org/tutorial/english/page3.htm

- **PLUGINS SRU**
  http://harvester.oarinz.ac.nz/index.php/about

- **PHPOAI2**
  http://physnet.uni-oldenburg.de/oai/

15 Annexes

Liste des documents fournis dans le CD en annexes :

- oai-pmhFinal.xml contient le fichier XML obtenu à partie de la requête SRU avec 8277 records tirés de l'entrepôt Mediaserver ;
- myslVersion2.xsl : Contient le fichier .XSL de la transformation XSL et FOXML ;
- TransformationOaiXsl.java : contient le code pour la transformation et le découpage de fichier XML ;
- getPid.java : contient le code pour la vérification de PIDs disponible dans Fedora ;
- fedoraIngest.java : contient le code pour l'ingest des métadonnées dans Fedora ;
- oai2/oaidpconfig : fichier .PHP pour la configuration de module PHPOAI2 pour rendre harvestable mediaserver.