Exploitation esup-canal-sof

1. Introduction

Cette page a pour objet de présenter toutes les informations utiles au paramétrage et à l'adaptation de SOF à votre environnement.

Elle sera mise à jour régulièrement en fonction des questions et des réponses posées sur la liste odf-utilisateurs.

> Attention
>
> La lecture de ce document doit se faire en lien avec les différents MPD qui sont fournis au format PDF et qui sont disponibles dans le répertoire docs /database/mpd

2. Connexion de SOF avec une base externe de scolarité (Apogée)

SOF permet de récupérer des données d'une base de données de scolarité (typiquement Apogée).

Un ensemble de requêtes sont livrées de base avec SOF pour faire ce lien mais il est possible pour chaque site de les adapter à ses besoins.

Il est aussi possible de ne pas connecter SOF à une base de données de scolarité.

2.1. Principe

La connexion à la base de scolarité externe est définie dans le fichier sql-scol.properties.

Les requêtes permettant d'importer des données ou des listes de valeurs depuis cette base dans SOF sont définies dans le fichier sql-import-scol.xml.

2.2. Ne pas connecter SOF à une base de scolarité

Il faut, avant de lancer SOF pour la première fois, modifier le fichier csof.xml afin de lui indiquer de ne pas chercher à récupérer l'établissement par défaut dans la base de scolarité : <scol initEnabled="N"/>Ensuite, il faudra supprimer l'accès aux menus "importer" et "synchroniser" (voir paragraphe consacré au paramétrage des menus).

2.3. Liens avec une base externe

2.3.1. Requêtes livrées de base

Un ensemble de requêtes sont livrées avec SOF et permettent de retrouver des données dans Apogée.

Elles se trouvent dans le fichier sql-import-scol.xml :

- getEtablissement : permet de retrouver les données d'un établissement
- getComposante : permet de retrouver les données d'une composante
- getVersionDiplome : permet de retrouver les données d'un diplome
- getElement : permet de retrouver les données d'un élément pédagogique
- getCodEtbDef : permet de retrouver le code de l'établissement par défaut (votre université). Ce code est ensuite utilisé en paramètre de la requête getEtablissement
- getCYC_DIP : permet de retrouver les éléments de la LOV des cycles
- getNAT_DIP : permet de retrouver les éléments de la LOV des natures de diplômes
- getTYP_DIP : permet de retrouver les éléments de la LOV des types de diplômes
- getNIV_DIP : permet de retrouver les éléments de la LOV des niveaux de formation

2.3.2. Requêtes personnalisées

2.3.2.1. Pour l'importation des données

Vous avez la possibilité de remplacer les requêtes d'importation d'objets livrées dans SOF par les votres (connexion à une autre base qu'Apogée ou recherche d'autres valeurs par exemple).

Vous devez ajouter ces requêtes dans le fichier sql-specif-scol.xml en prenant garde aux points suivants :

- Les paramètres des requêtes doivent être identiques à ceux des requêtes livrées
- Les noms de vos requêtes doivent être différents de celles livrées avec SOF
  Vous devez ensuite faire référence à ces requêtes dans le fichier csof.xml.

Ainsi si vous avez ajouté une requête nommée getEtbNancy2 dans sql-specif-scol.xml pour l'importation des établissements, vous devez ajouter dans csof.xml :

```
<options>

<query name="getEtablissement" value="getEtbNancy2"/>

</options>
```

2.3.2.2. Pour les listes de valeurs

Il est possible, lors de la première initialisation de SOF, d'initialiser des listes de valeurs de SOF avec le résultat de requêtes sur une base externe (Apogée).

L'initialisation ne concernera que la langue française.

Un certain nombre de requêtes permettant d'initialiser des listes à l'aide d'Apogée sont livrées de base (voir plus haut) mais si voulez importer de nouvelles listes ou ne pas utiliser Apogée, voici la marche à suivre :

- Ajoutez vos requêtes dans les fichier sql-import-scol.xml en vérifiant les points suivants :
- Donnez un id à vos requêtes différent de ceux existant déjà pour les requêtes SOF
- Votre requête doit retourner 2 colonnes nommées COD_DET_LOV et LIB_DET_LOV
- Faites reférence à votre requête personnalisée dans la table FUN_LOV en positionnant son nom dans la colonne REQ_SYNC_LOV pour le COD_LOV correspondant(null pour ne pas faire de lien du tout).

2.3.3. Lien des informations avec la base externe

Les informations à utiliser pour chaque type d'objet sont présentes dans la table FUN_INFO_TYP

Selon la valeur de la colonne TYP_SYNC_INF, l'information sera :

- Importée/Synchronisée avec une base externe directement via une requête (valeur A)
- Importée/Synchronisée via un calcul se faisant dans la classe implémentant l'interface ISync (voir exemple plus bas)
- Non synchronisée(valeur N)
  Pour les informations synchronisées automatiquement, la colonne COL_SYNC_INF vous permet d'indiquer à quelle colonne de votre requête sur la base externe vous souhaitez lier cette information.

  > Note
  > Dans le cas de la synchronisation de données d'un objet déjà existant, ne seront synchronisées que les informations pour lesquelles FUN_INFO.TEM_MOD_INF = 'I'

  Le mécanisme de synchronisation est implémenté dans le package sync. Deux classes sont proposées par défaut :
- L'une pour la synchronisation à partir de la base externe (voir ci-dessus)
- L'autre pour la synchronisation à partir d'un annuaire LDAP
  Les applications dont sont importés les objets sont définies dans la table FUN_TYP_OBJ (colonne APP_ORI_TYP_OBJ).

Pour modifier le comportement par défaut du mécanisme de synchronisation, il convient donc de définir sa propre classe étendant l'interface ISync et d'adapter en conséquence les lignes suivantes dans le fichier de configuration du canal :

```
<classfactory name="syncPers" class="fr.unire.portal.channels.fun.csof.sync.SyncDefLDAP"/>

<classfactory name="syncAll" class="fr.unire.portal.channels.fun.csof.sync.SyncDefImpl"/>
```

3. Adaptation du référentiel

Cette section présente la structure fondamentale du référentiel en précisant les tables impactées et ce que vous pouvez adapter.

3.1. Années

Chaque année possède les attributs suivants :

- Un témoin en service
- Un témoin indiquant si des modifications sont possibles sur les objets de l'année (non géré pour l'instant)
- Un témoin indiquant si la publication des diplômes est activée pour l'année
  Par défaut, le canal propose les années suivantes : 2004/2005, 2005/2006, 2006/2007 et 2007/2008. Il est naturellement possible d'ajouter de nouvelles années via la table FUN_ANNEE.

Deux années spécifiques ont été créées pour représenter les valeurs minimales et maximales, appelées "Première année" et "Dernière année". Ces années ne doivent pas être supprimées.

Si vous importez des objets faisant référence à des années qui n'existent pas dans SOF, celles-ci seront automatiquement créées.

3.2. Langues

Les informations d'un objet peuvent être définies en plusieurs langues(voir la table FUN_OBJ_LANG).

La liste des langues possibles est définit dans la table FUN_LANGUE, leurs libellés étant spécifiés dans la table FUN_LIB_DET_LOV via la table FUN_DET_LOV. Il est ainsi possible de spécifier leurs libellés en plusieurs langues.

De manière générale, toutes les listes de valeurs de l'application peuvent potentiellement être traduites dans les langues spécifiées.

La langue par défaut est spécifiée dans la table FUN_PARAM (LANG_DEF).

3.3. Groupes d'objets

Les groupes d'objets correspondent aux différents types d'objets que l'on retrouve dans CDM :

- Les orgUnit (ou structures) qui sont le groupe n°1
- Les program (ou diplômes) qui sont le groupe n°2
- Les courses (ou éléments) qui sont le groupe n°3
- Les persons (ou personnes) qui sont le groupe n°5

A ces 4 groupes a été ajouté le groupe header (ou titres, n°4) qui n'existe pas dans CDM mais nous servira à modéliser des titres pour certaines listes (listes de parcours, listes d'éléments...).
Ces groupes sont des invariants dans SOF. Il n'est donc pas possible d'en ajouter.

3.4. Types d'objets

Les types d'objets sont des instances des groupes d'objets. Par exemple, le type d'objet Diplôme appartiendra au groupe program, comme le type d'objet parcours ou spécialité

On retrouvera les types d'objets dans la table FUN_TYP_OBJ.

Il est possible d'ajouter ses propres types (voir exemple plus bas).

3.5. Groupes d'informations

Les groupes d'informations permettent de regrouper logiquement des informations liées à un groupe d'objet.

Pour un type d'objet, une information ne pourra appartenir qu'à un et un seul groupe d'information.

La répartition des informations entre les groupes est fondamentale car on donnera ensuite des droits de modification aux utilisateurs globalement sur un groupe.

Il faudra donc regrouper entre elles les informations pour lesquelles on souhaite donner des droits identiques.

Les groupes d'informations se trouvent dans la FUN_GRP_INFO.

Il est possible de ne pas afficher/saisir une groupe d'information pour un type d'objet précis en jouant sur la table FUN_GRP_INFO_TYP.

Pour qu'un utilisateur ait les droits de modifications sur un groupe, il faut qu'il y ait une occurence dans la table FUN_DRT_PROF_UTI avec son code profil, le code du groupe et le numéro du groupe de l'objet.

Exemple : Autorisation des utilisateurs de profil "Scolarité" à modifier le groupe "Informations générales" des objets de type "Diplôme"

```
insert into FUN_DRT_PROF_UTI values('SCOL','GEN',2,'');
```

3.6. Informations

Les informations sont liées à un groupe d'information unique. Elles permettent de définir les règles de saisie d'un élément CDM.

Une information est définie :

- Par son type(TYP_INF) qui peut prendre comme valeur :
- LIB pour les libellés simples
- XHMTL pour les zones de saisie riche (infoBlock)
- DATE pour les dates
- NB pour les nombres
- LOV pour les listes de valeurs
- Son caractère obligatoire(TEM_OBL_INF)
- Le code de la liste de valeurs rattachée(COD_LOV) si TYP_INF=LOV. Dans ce cas, COD_LOV doit exister dans FUN_LOV.
- Ses tailles minimum et maximum (pour les libellés et zones riches) : colonnes LON_MIN_INF et LON_MAX_INF
- Son caractère multilingue ou non (TEM_LANG_INF)

Les informations sont contenues dans la table FUN_INFO.

3.7. Listes de valeurs

Les liste de valeurs permettent de restreindre les valeurs prises par certaines informations à une liste fixe.

Pour cela, les informations doivent avoir un FUN_INFO.TYP_INF=LOV et faire référence à une liste de valeur dans FUN_INFO.COD_LOV.

Les listes de valeurs peuvent être multilingues.

Voici les tables contenant les données des listes de valeurs :

- FUN_LOV : contient les listes de valeurs
- FUN_DET_LOV : contient les différentes valeurs d'une LOV
- FUN_LIB_DET_LOV : contient le libellé d'une valeur dans (éventuellement) plusieurs langues

3.8. Exemple : ajout d'un nouveau type d'objets

Dans cet exemple nous allons voir les différentes étapes pour ajouter le type d'objet Année qui appartiendra au groupe Titres(n°4) et qui sera lié à une version d'étape dans Apogée.

- Ajout de l'item Année dans la liste de valeur TYP_OBJ insert into FUN_DET_LOV values(21,'TYP_OBJ','AN','O');
  insert into FUN_LIB_DET_LOV values(21,40,'Année');
- Ajout du type d'objet Année insert into FUN_TYP_OBJ values(21,4,'APOGEE','VET','O','Code étape','Version étape'l);
  insert into FUN_TYP_OBJ_SUP values(21,20,'O',0);
- Indication de la manière dont les informations du groupe 4 sont synchronisées pour le nouveau type n°21 insert into FUN_INFO_TYP values ('NOM',4,21,'A','LIB_ETP');
  insert into FUN_INFO_TYP select COD_INF, NUM_GRP_OBJ, 21, 'N', null from FUN_INFO I where NUM_GRP_OBJ = 4 and not exists (select 1 from FUN_INFO_TYP where NUM_TYP_OBJ = 21 and NUM_GRP_OBJ = I.NUM_GRP_OBJ and COD_INF = I.COD_INF);
- Réaffectation de tous les groupes d'informations aux types d'objets correspondant (procéder plus finement en production) delete from FUN_GRP_INFO_TYP;
  insert into FUN_GRP_INFO_TYP

```
select distinct GI.COD_GI, GI.NUM_GRP_OBJ, IT.NUM_TYP_OBJ

from FUN_GRP_INFO GI, FUN_INFO_TYP IT, FUN_INFO I

where GI.TES_GI = 'O'

and   I.NUM_GRP_OBJ = GI.NUM_GRP_OBJ

and   I.COD_GI = GI.COD_GI

and   IT.COD_INF = I.COD_INF

and   IT.NUM_GRP_OBJ = I.NUM_GRP_OBJ

order by IT.NUM_TYP_OBJ, GI.NUM_GRP_OBJ;
```

- Création d'une classe qui implémente l'interface ISyncDefImpl. Pour cet exemple, il suffit d'écrire une classe qui étend la classe SyncDefImpl et qui surcharge sa méthode setObjet pour prendre en compte le cas du type d'objet n°21
- Il faut ensuite référencer la nouvelle classe(appellons la SyncDefImplPerso) dans le fichier csof.xml.

Par exemple :

```
<classfactories>

...

<classfactory name="syncAll" class="fr.unire.portal.channels.fun.csof.sync.SyncDefImplPerso"/>

...

</classfactories>
```

4. La gestion des droits d'accès dans l'application

4.1. Les différents types de profils intervenant dans SOF

Il existe deux types de profils qui peuvent intervenir quand on travaille avec le canal SOF :

- Un profil dynamique qui dépend des droits de l'utilisateur sur l'objet courant (un objet étant un élément de l'offre de formation : diplôme, UE, personne (intervenant), etc.)
- Un profil statique, associé à certains logins par l'administrateur de la base de données de SOF (sauf pour le cas particulier du profil statique ANONYME)
  Les droits affectés à une personne sont ceux du profil dynamique s'il existe, ceux du profil statique sinon.

Il existe une exception à cette règle :

Si une personne possède un profil statique ADMIN, le profil dynamique n'intervient jamais. On dit que le profil statique ADMIN est bloquant (ce comportement peut se régler au niveau de la table FUN_PROF_UTI de la base de données de SOF).

Quand une personne n'a pas de profil dynamique pour l'objet courant et qu'elle n'a pas de profil statique, le profil statique ANONYME lui est attribué par défaut.

4.1.1. Les profils dynamiques

Il y a actuellement 3 types de profils dynamiques :

- RESP : responsable direct de l'objet courant
- RESPSUP : responsable d'un objet ancêtre de l'objet courant
- RESPDIP : responsable d'un diplôme de publication

Selon l'objet courant, la personne peut avoir l'un de ces trois profils (ou un profil statique si elle ne possède aucun droit correspondant à ces profils dynamiques). Pour un objet donné, on ne peut avoir qu'un seul profil dynamique (on ne peut être en même temps RESPDIP et RESP d'un objet de l'offre de formation).
Ces profils sont donc qualifiés de dynamiques : ils dépendent de l'endroit où se situe la personne dans l'offre de formation (une personne peut ainsi avoir un profil RESP pour un semestre, RESPSUP pour une UE de ce semestre, aucun profil temporaire pour les élements de formation d'une composante à laquelle elle n'est point rattachée, etc.).

Par défaut, l'ordre d'évaluation des profils dynamiques est le suivant : RESP, RESPDIP, RESPSUP. Cet ordre peut être modifié pour mettre en oeuvre une autre politique de gestion de droits sur les éléments de l'offre de formation. Il faut alors créer une classe implémentant l'interface Iprofil (particulièrement la méthode calcDynamicProfil) et référencer ensuite cette classe dans le fichier csof.xml : <classfactory name="profil" class="NomDeLaNouvelleClasse"/>Le profil dynamique RESP est attribué à une personne quand elle travaille sur un objet (diplôme, semestre, UE, etc.) dont elle est responsable (sous-entendu : responsable direct).

Le profil dynamique RESPSUP est attribué à une personne quand elle travaille sur un objet qui est un descendant d'un objet dont elle est responsable. Typiquement, si une personne est responsable (RESP) d'un semestre, lorsqu'elle travaillera sur une UE de ce semestre, elle sera RESPSUP (sauf si, par exemple, elle a aussi été désignée comme responsable de cette UE ; cas où elle sera alors RESP pour cette UE - cf. l'ordre d'évaluation des profils dynamiques dont nous avons parlé précédemment).

Le profil dynamique RESPDIP est affecté à une personne responsable (RESP) d'un diplôme quand elle travaille sur un objet qui est un descendant de ce diplôme. Là-aussi, l'ordre d'évaluation des profils dynamiques peut faire que la personne ait un profil RESP pour un semestre appartenant au diplôme dont il est question.

La responsabilité d'un objet (RESP) peut être attribuée à une personne par une personne possédant un profil statique "fort" (comme ADMIN), ou par un RESPSUP de l'objet (ceci dépendant alors des pouvoirs effectifs du profil dynamique RESPSUP).

Les différents droits des profils dynamiques RESP, RESPSUP et RESPDIP sont fixés par les entrées des tables FUN_DRT_PROF_UTI et FUN_MEN_PROF_UTI.

Les 3 profils dynamiques que nous venons de décrire sont stockés dans la table FUN_PROF_UTI.

4.1.2. Les profils statiques

Il existe actuellement dans SOF 3 profils statiques :

- ADMIN, qui correspond à l'administrateur SOF
- SCOL, qui correspond à une personne de la scolarité disposant de droits élevés
- ANONYME, qui est le profil statique par défaut, avec très peu de droits

Un profil statique est attribué au login d'une personne (hormis le cas particulier du profil statique ANONYME).
Les associations entre login et profil statique sont stockées dans la table FUN_PROF_UTI_PERS de la base de données de SOF.

Exemples d'associations possibles

Login	Profil statique
ingenp01	ADMIN
secrep01	SCOL
secrep02	SCOL

Pour l'objet courant, le profil statique n'entre en compte que s'il n'y a pas de profil dynamique. Toutefois, certains profils statiques ont la propriété d'être bloquants : ils empêchent l'évaluation du profil dynamique et sont ainsi attribués à la personne. C'est le comportement par défaut du profils statique ADMIN. Ce n'est pas le cas de SCOL et ANONYME.

L'aspect bloquant ou non d'un profil statique peut être modifié en mettant à jour le champ TEM_CAL_PRO_UTI de la table FUN_PROF_UTI pour l'enregistrement correspondant au profil statique auquel on s'intéresse.

Le profil statique ANONYME entre en jeu quand une personne examine un objet de l'offre de formation pour lequel elle n'a aucun profil dynamique (elle n'a aucune responsabilité vis à vis de cet objet) et quand le login de cette personne n'est pas lié à un profil statique. Dans cette situation, la personne se voit attribuer le profil statique ANONYME qui dispose de droits très faibles.

Ainsi, un membre du personnel de l'Université sans rapport avec la formation n'aura pas de profil dynamique (un administrateur ou un responsable d'un objet de la formation ne lui aura pas donné de profil dynamique en le rendant responsable d'un semestre ou d'une UE). C'est donc son profil statique qui entrera en compte. Comme ce membre du personnel n'aura pas d'entrée dans la table FUN_PROF_UTI_PERS, il se retrouvera, pour chaque élément de l'offre de formation, avec comme droits ceux du profil statique par défaut : ANONYME.

Les profils statiques ADMIN et SCOL disposent des droits maximaux. Ils peuvent modifier la hiérarchie de l'offre de formation (détruire un objet et tous ses enfants, en créer d'autres, modifier les données descriptives d'un objet, etc.).

A l'inverse, le profil statique ANONYME ne permet que de consulter l'offre de formation.

Les différents droits des profils statiques ADMIN, SCOL et ANONYME sont fixés par les entrées des tables FUN_DRT_PROF_UTI et FUN_MEN_PROF_UTI.

Les 3 profils statiques que nous venons de présenter sont stockés dans la table FUN_PROF_UTI.

4.2. Paramétrage des menus

4.2.1. Principes

Les menus sont définit dans la table FUN_MENU.

Chaque menu est composé d'un ensemble d'entrées, stockées dans la table FUN_DET_MENU.

- Elles peuvent être activées/désactivées, affichées ou pas
- Leur ordre d'affichage est paramétrable lorsque le menu est présenté sous forme de liste
- Il est possible de spécifier un commentaire associé, qui peut ou pas être utilisé lorsque le menu est présenté sous forme de liste (voir les feuilles de style default.xsl et menuObjet.xsl)
- Elles sont fonction du profil de la personne connectée (FUN_MEN_PROF_UTI) ainsi que du type de l'objet (FUN_NOT_DET_MENU)
    - Chaque entrée d'un menu est associée à une ou plusieurs actions, stockées dans la FUN_DET_MENU_ACT. Il est possible de spécifier des paramètres associés.

4.2.2. Exemple

Prenons l'exemple du menu d'un objet.

> Note
>
> Cette section se réfère au menu par défaut proposé dans le fichier referentiel.sql

Il est identifié par le code OBJ et a comme nom Objet .

```
insert into FUN_MENU values('OBJ','Objet');Il est composé des entrées suivantes :
```

- Propriétés : Cette entrée est en service, positionnée en première position, affichée à l'utilisateur et n'a pas de commentaire associéinsert into FUN_DET_MENU values(12,'OBJ','O','Propriétés',1,'O','');* Pour qu'elle soit utilisable par les utilisateurs d'un profil donné, il faut qu'il y ait dans la table FUN_MEN_PROF_UTI une occurence avec comme code Profil celui du profil considéré et comme numéro d'entrée de menu celui de l'entrée considérée. Exemple avec le profil statique SCOL :insert into FUN_MEN_PROF_UTI values('SCOL',12);
  Il y a donc, pour une entrée donnée, autant d'occurences dans FUN_MEN_PROF_UTI qu'il y a de profils autorisés à l'utiliser.
- Cette entrée n'a pas d'occurence dans la table FUN_NOT_DET_MENU, elle est donc proposée quelque soit le type de l'objet (Diplôme, Semestre, Composantes...)
- Personnes : Même principe...
- Responsables : Même principe...
- Cohabilitations : Même principe...
- Cette entrée n'est proposée que pour les diplômes. On trouve donc dans la table FUN_NOT_DET_MENU autant d'occurences correspondant à cette entrée qu'il y a de types d'objet autre que Diplôme. Exemple avec le type d'objet Semestreinsert into FUN_NOT_DET_MENU values(15,22);
- Groupes d'informations : Même principe...
- Visualisation : Même principe...
- Cohérence : Même principe...
- Validation : Même principe...
- Génération du CDM : Même principe...
- Synchronisation : Même principe...
- Web Services : Même principe...
  A chacune de ces entrées sont associées des actions. Prenons l'exemple de l'entrée Propriétés. Deux actions sont spécifiées :
- L'une affichant du formulaire de mise à jour des propriétés
- Le témoin TEM_DEF_ACT_MEN est à 'O' pour indiquer qu'il s'agit de l'action par défaut pour cette entrée. C'est donc cette action qui est ajoutée sur le lien du menuinsert into FUN_DET_MENU_ACT values(12,'updatePropObj','','O');
- L'autre effectuant la mise à jour
- Le témoin TEM_DEF_ACT_MEN est à 'N'. Cette action n'est pas utilisée pour le lien du menu.insert into FUN_DET_MENU_ACT values (12,'doUpdatePropObj','','N');

  > Note
  > Le canal vérifie à chaque appel d'une action que le profil de l'utilisateur lui permet de l'exécuter. A partir du moment où l'utilisateur a les droits sur une entrée d'un menu, il est autorisé à exécuter l'ensemble des actions associées.

5. Mécanisme de "Workflow"

Le canal fournit un mécanisme de worflow pour la validation/publication d'un objet. Ce mécanisme permet aux sites d'enrichir les phases de validation /publication de traitements qui leurs sont spécifiques.

L'implémentation de ce mécanisme est effectuée dans le package workflow. Ce package contient :

- Une interface IWorkflowValidation, définissant les appels générés lors des processus de validation/publication d'un objet
- Une classe WorkflowValidationResult, dont les objets sont en fait le résultat des méthodes de l'interface IWorkFlowValidation
- Un objet de ce type définit l'action vers laquelle se rediriger, le résultat de la validation/publication (true ou false) et un message éventuel
- Une classe d'implémentation par défaut WorkflowValidationDefImpl
- Les sites souhaitant un autre comportement que celui proposé par défaut doivent développer leur propre classe (implémentant IWorkflowValidation). La classe d'implémentation est paramétrable dans le fichier de configuration csof.xml

```
<classfactory name="workflowValidation" class="fr.unire.portal.channels.fun.csof.workflow.
WorkflowValidationDefImpl"/>
```

5.1. Validation d'un objet

Deux points de sortie sont prévus :

- L'un après la validation d'un groupe d'information
- L'autre après la validation de l'objet de lui-même
  Il est ainsi possible, par exemple, d'ajouter un mécanisme de notification par mail des gestionnaires d'un objet lors de la validation de l'un de ses groupes.
- Les méthodes de l'interface IWorkflowValidation utilisées sont : public WorkflowValidationResult apresValidationGroupeInfoObj(GroupeInfoObjet groupeInfoObjet, boolean result);
  public WorkflowValidationResult apresValidationObjet(Objet objet, boolean result);
  Ce sont donc ces méthodes qu'ils convient de redéfninir/surcharger pour personnaliser le comportement.
- L'implémentation par défaut proposée dans la classe WorkflowValidationDefImpl est la suivante :
- Dans le cas des groupes d'information, si la validation est réussie, une demande de validation de l'objet est lancée. Dans le cas contraire, une redirection vers l'édition des groupes d'information pour corriger les erreurs détectées est effectuée.
- Dans le cas de l'objet, si l'un de ses groupes d'information n'est pas valide quelque soit la langue (ie au moins l'un des TEM_VAL_LANG de l'objet est égal à 'N'), alors une redirection vers l'édition des groupes d'information est faite. Dans le cas contraire, l'affichage du rapport de cohérence est demandé.

5.2. Publication du CDM d'un objet

5.2.1. Publication

La méthode publicationObjet de l'interface IWorkflowValidation permet de publier un document CDM connaissant

- Le numéro de l'objet
- L'année universtaire
- Une chaîne représentant le CDM pour cet objet et cette année
  Le parti a été pris de publier le CDM d'un objet année par année ce qui permet d'avoir une description différente d'un même objet dans le temps.

WorkflowValidationDefImpl est l'implémentation SOF type de IWorlflowValidation. Elle effectue les opérations suivantes :

- Lancement d'une thread qui publie le CDM sur tous les web-services associés à l'objet

5.2.2. Suppression

Il a aussi été prévu de pouvoir retirer le CDM d'un objet du serveur d'affichage de l'offre de formation

Ceci est réalisé par la méthode suppressionObjet de l'interface IWorkflowValidation

L'implémentation type WorkflowValidationDefImpl effectue les opérations suivantes en cas de suppression :

- Lancement d'une thread qui supprime le CDM sur tous les web-services associés à l'objet et pour toutes les années

6. Publication des fichiers CDM

Cette section décrit le principe par défaut de publication des objets SOF au format CDM et comment l'adapter à vos besoins

6.1. Principe par défaut

Les profils qui sont autorisés ont accès à un menu Publication disponible uniquement sur les diplômes de publication.

Définition

On appelle diplôme de publication un diplôme(objet du groupe Program) situé directement sous un objet de type orgUnit

Ainsi, les mentions ou spécialités ne sont pas des diplômes de publication si elles sont elles-mêmes situées dans un diplôme
Le menu Publication déclenche les opérations suivantes :

- Génération du fichier CDM pour le diplôme
- Validation du fichier CDM généré par rapport au schéma XSD de CDM
- Publication du fichier CDM sur les web-services de publication de l'objet et pour l'année en cours

6.2. Génération du fichier CDM

C'est la méthode generate de la classe CDM qui est chargée de cette tâche.

6.2.1. Génération de la structure d'un objet au format CDM

Chaque groupe d'objet peut disposer d'une classe spécifique à la génération de sa structure au format CDM. Cette classe doit implémenter l'interface IObjCdm.

Par structure on entend la manière dont sont imbriqués les éléments CDM entre eux (par exemple pour générer la structure d'un diplôme ou d'un cours).

Par défaut dans SOF, la classe ObjCdmDefImpl génére le CDM de tout type d'objet. Ceci est précisé ainsi dans le fichier csof.xml : <classfactory name=" cdmAll" class="fr.unire.portal.channels.fun.csof.cdm.ObjCdmDefImpl"/>Si vous voulez par exemple mettre en oeuvre une classe différente pour générer le CDM d'une personne, créez votre propre classe et faites y référence dans csof.xml ainsi : <classfactory name="cdmPers" class="fr.unire.portal.channels. fun.csof.cdm.PersCdmDefImpl"/>

6.2.2. Récupération des informations CDM d'un objet

Il est possible aussi dans SOF de paramétrer où sont cherchées les données pour chaque type d'objet CDM

Les interfaces definissant les données à générer en CDM sont définies dans :

- ICdmOrgUnit pour les structures
- ICdmProgram pour les diplômes(mentions et spécialités aussi)
- ICdmCourse pour les éléments(UE, semestres, matières...)
- ICdmPerson pour les personnes
  Par défaut dans SOF ce sont les classes CdmOrgUnitDefImpl, CdmProgramDefImpl, CdmCourseDefImpl et CdmPersonDefImpl qui sont chargées de cette tâche.

Il est cependant possible de redéfinir vos propres classes et d'y faire référence dans le fichier csof.xml ainsi :

```
<classfactory name="cdmGenOrgUnit" class="fr.unire.portal.channels.fun.csof.cdm.CdmOrgUnitPerso"/>

<classfactory name="cdmGenProgram" class="fr.unire.portal.channels.fun.csof.cdm.CdmProgramPerso"/>

<classfactory name="cdmGenCourse" class="fr.unire.portal.channels.fun.csof.cdm.CdmCoursePerso"/>

<classfactory name="cdmGenPerson" class="fr.unire.portal.channels.fun.csof.cdm.CdmPersonPerso"/>
```

6.3. Validation par rapport au schéma CDM

Cette opération est faite via la classe JAXPValidator et par rapport au schéma CDMv2.01.xsd (indiqué dans la classe ApplicationContext)

6.4. Publication sur les web-services

La liste des web-services de publication est disponible dans la table FUN_WS_PUB.

Cette table est livrée vide, il vous faut l'alimenter pour pouvoir publier vos fichiers.

Par défaut un objet de publication est publié sur tous les web-services disponibles dans cette table.

Par contre dès lors qu'un web-service est présent dans la table FUN_WSP_NOT_OBJ pour un objet alors il n'est pas utilisé pour la publication.

7. Personnalisation de l'interface

7.1. Affichage AJAX

Asynchronous JavaScript And XML(AJAX) est une méthode de développement de sites web utilisée dans SOF afin de modifier seulement un partie de l'affichage du canal plutôt que de rafraichir tout l'ENT (ce qui peut être plus lent).

AJAX utilise intensivement javascript, ce qui peut poser des problèmes de compatibilité avec certains navigateurs (par exemple SOF n'a pas été testé avec Safari pour Mac)

Si vous rencontrez des soucis d'affichage dus à l'utilisation d'Ajax vous avez la possibilité de désactiver ce type d'affichage en modifiant le paramètre suivant dans le fichier csof.xml : <ajax enabled="O"/>
Il faudra alors passer l'attribut enabled à N.

7.2. Personnalisation du look

Il est possible d'adapter un certain nombre d'éléments du look en modifiant le fichier webpages/stylesheets/fr/unire/portal/channels/fun/csof/actions /templates/constantes.xsl

7.2.1. Icônes

Il est possible de spécifier pour chaque icône l'image utilisée. Le chemin indiqué est relatif au mediaPath du canal.

Exemple :

```
<xsl:param name="IMG_ACCUEIL">home.gif</xsl:param>
```

7.2.2. Fonds

Les fonds des zones spécifiques (comme le cartouche d'entête d'un objet) peuvent être adaptés.

Exemple :

```
<xsl:param name="FOND_OBJ">uportal-input-text</xsl:param>
```

Il est en de même pour les couleurs utilisées pour mettre en évidence les champs ayant le focus ou étant disabled (disponible uniquement sous FireFox 1.5)

Exemple :

```
<xsl:param name="FOND_FOCUS_INPUT">#FFFFCC</xsl:param>

<xsl:param name="TEXT_FOCUS_INPUT">#000000</xsl:param>
```

7.2.3. Taille des libellés dans le rappel du parcours dans l'arborescence

Pour obtenir des libellés de taille plus importante (notamment pour voir intégralement le nom de votre université) dans le récapitulatif de l'arborescence, modifiez la valeur du paramètre LENGTH_LIB_OBJ :<xsl:param name="LENGTH_LIB_OBJ">20</xsl:param>

8. Commentaires sur la base de données

8.1. Langues

Les langues sont des listes de valeurs.

Cependant, la table FUN_LANGUE est nécessaire pour pouvoir définir facilement les libellés des listes de valeurs en plusieurs langues.

9. Requêtes utiles

9.1. Liste des types d'objets

```
select DL.*, LDL.LIB_DET_LOV

from FUN_DET_LOV DL, FUN_LIB_DET_LOV LDL

where DL.COD_LOV='TYP_OBJ'

and LDL.NUM_DET_LOV = DL.NUM_DET_LOV
```

10. Divers

10.1. Génération des logs pour l'application

Voici les modifications à apporter au fichier Logger.properties afin d'avoir des logs détaillés sur l'application SOF

```
############################################################

log iBATIS

############################################################

log4j.category.com.ibatis= DEBUG, IBATIS

log4j.logger.com.ibatis.common.jdbc.ScriptRunner=DEBUG, IBATIS

log4j.logger.com.ibatis.common.jdbc.SimpleDataSource=DEBUG, IBATIS

log4j.logger.java.sql.Connection=DEBUG, IBATIS

log4j.logger.java.sql.Statement=DEBUG, IBATIS

log4j.logger.java.sql.PreparedStatement=DEBUG, IBATIS

log4j.additivity.com.ibatis = false


log4j.appender.IBATIS = org.apache.log4j.RollingFileAppender

log4j.appender.IBATIS.File = C:/esupdev/ibatis.log

log4j.appender.IBATIS.Encoding=UTF-8

log4j.appender.IBATIS.MaxFileSize=3000KB

log4j.appender.IBATIS.MaxBackupIndex=10

log4j.appender.IBATIS.layout=org.apache.log4j.PatternLayout

log4j.appender.IBATIS.layout.ConversionPattern=%5p %t %c

{2}.[%x] %d{MMM/dd HH:mm:ss}    - %m%n

############################################################
#         log CSOF
############################################################
log4j.category.fr.unire.portal.channels.fun.csof= DEBUG, CSOF
log4j.additivity.fr.unire.portal.channels.fun.csof = false

log4j.appender.CSOF = org.apache.log4j.RollingFileAppender
log4j.appender.CSOF.File = C:/esupdev/cSof.log
log4j.appender.CSOF.Encoding=UTF-8
log4j.appender.CSOF.MaxFileSize=3000KB
log4j.appender.CSOF.MaxBackupIndex=10
log4j.appender.CSOF.layout=org.apache.log4j.PatternLayout
log4j.appender.CSOF.layout.ConversionPattern=%5p [%t] %c{2}.%x %d
```

```
{MMM/dd HH:mm:ss}- %m%n
```