Formation développeur à Chainedit

Romuald Lorthioir
CIRM / Université de Rennes1

Nadia Henry
CIRM / Université de Rennes1

Ce document est la version 2.0 du support de la formation intitulée « Formation Chainedit administrateur » organisée par le centre d'ingénierie et de ressources multimédias de l'Université de Rennes1.

La dernière version est accessible en ligne à l'URL suivante :
http://www.chainedit.fr/

Pour toute remarque, merci de vous adresser à : romuald.lorthioir@univ-rennes1.fr
Vous pouvez également utiliser la liste de diffusion disponible à l'adresse (service offert par le CRU): http://listes.cru.fr/sympa/info/chainedit.

Copyright

Table des matières

I. Généralités

ChainEdit est une application Internet qui permet de générer des contenus documentaires dans des formats variés (Html, Pdf, OpenOffice, Word, ..). Pour cela, l'application permet de saisir des contenus textes auxquels peuvent être liés des médias (vidéo, images, animations, ..). Elle nécessite une installation sur un serveur pour en mutualiser l'utilisation. L'utilisateur a besoin d'un navigateur standard pour l'utiliser (Internet Explorer ou FireFox).

Cet outil, à destination des services TICE et des enseignants, a pour objectif premier de permettre la création de ressources pédagogiques intégrant divers médias (vidéos, animations, photos, QCM, ..) et d'en assurer la cohérence. Le procédé retenu (Séparation fond/forme) permet de générer des ressources pédagogiques pour différents contextes (établissement, UNR, UNT) en respectant les contraintes graphiques et techniques de chacun d'eux.

1. Installation

L'installation de l'application ChainEdit se fait sur un serveur Linux ou Windows. Il est nécessaire de disposer d'une Jdk java 1.6, d'un serveur Tomcat et d'un OpenOffice installé si l'on souhaite générer des documents pdf directement à partir du serveur. Il faut également disposer de bibliothèques latex ou d'un outil Mimetex pour pouvoir générer des formules Latex.

Il existe également une version dite « Quick-Start » qui permet de tester l'application localement. Cette version n'a pas vocation à être installée sur un serveur.

L'ensemble de la procédure d'installation est décrit dans la documentation d'installation de ChainEdit. Ce document est disponible dans les distributions de l'application.

En plus de l'installation de l'application, des paramétrages complémentaires sont nécessaires pour une bonne intégration au système d'information. Cette partie est également décrite dans le document d'installation.

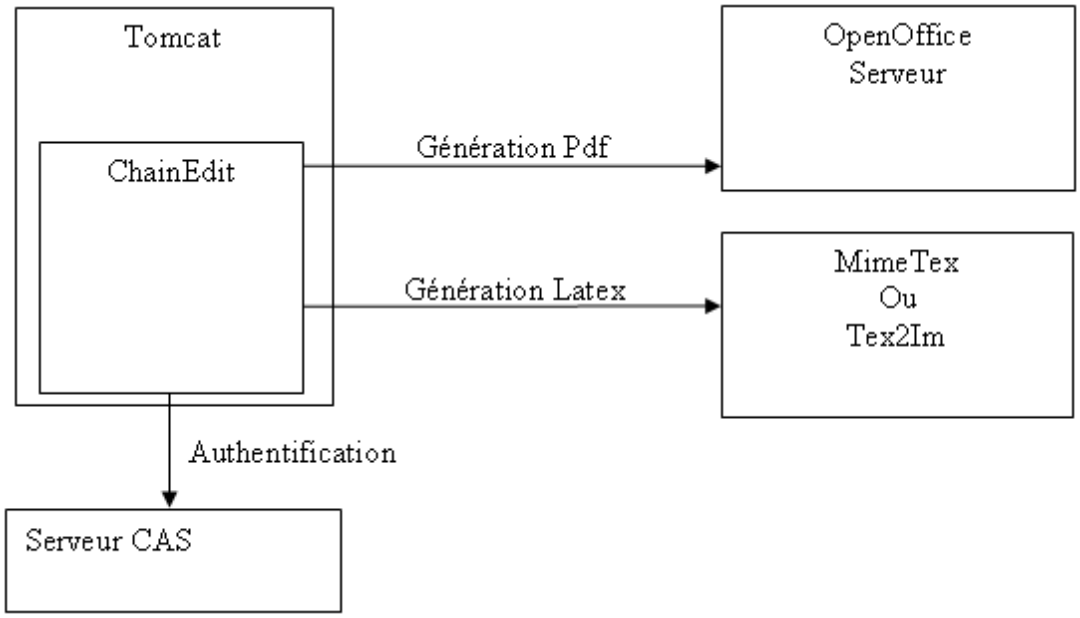
2. Communauté

L'application dont la base est développée par l'université de Rennes 1 est une application OpenSource. Elle est donc modifiable par tout établissement qui en a le besoin. De plus, un site (www.chainedit.fr) est disponible et permet de s'inscrire à des listes de diffusion pour être tenu informé des évolutions de l'application, signaler un problème, ou suggérer des améliorations. Tous les utilisateurs sont invités à s'inscrire à ces listes.

3. Architecture

Cette partie décrit le schéma de fonctionnement général de ChainEdit et une description de l'arborescence type de déploiement. Elle sera suivie d'une démonstration d'utilisation.

A. Schéma de fonctionnement



Pour fonctionner, ChainEdit a besoin :

- d'un environnement Java
- d'un serveur d'application tel que Tomcat.
- d'un serveur CAS si l'on souhaite une authentification SSO
- d'un OpenOffice en mode serveur si ChainEdit doit éditer directement des pdf
- d'un utilitaire tex2im ou MimeTex si ChainEdit doit générer du Latex

B. Arborescence applicative

Les éléments en gras sont ceux qui seront utilisés lors de cette formation.

Ecran Connexion

- Menu
  - Projets
    - Editer
  - Préférences
    - Editer
  - **Administration**
    - Gestion des utilisateurs
      - Ajouter
      - Modifier
      - Supprimer
    - **Gestion des chartes**
      - **Ajouter**
        - **Configuration**
        - **Gestion des traitements**
      - **Modifier**
        - **Configuration**
        - **Gestion des traitements**
      - Supprimer
      - Gestion des fichiers de la charte
      - Archiver et télécharger les fichiers de la charte
    - **Gestion des configurations**
      - **Ajouter**
        - **Gestion des répertoires**
      - **Modifier**
        - **Gestion des répertoires**
      - Supprimer
      - Archiver et télécharger les fichiers de la configuration
    - **Gestion des imports**
      - **Ajouter**
        - **Création, dépôt des éléments**
      - **Modifier**
        - **Configuration de l'import**
        - **Gestion des répertoires**

Supprimer

Gestion des référencements

Ajouter

Création, dépôt des éléments

Modifier

Configuration du référencement

Gestion des répertoires

Supprimer

Gestion des projets

Ajouter

Gestion des répertoires

Modifier

Gestion des répertoires

Supprimer

Gestion des catégories

Ajouter

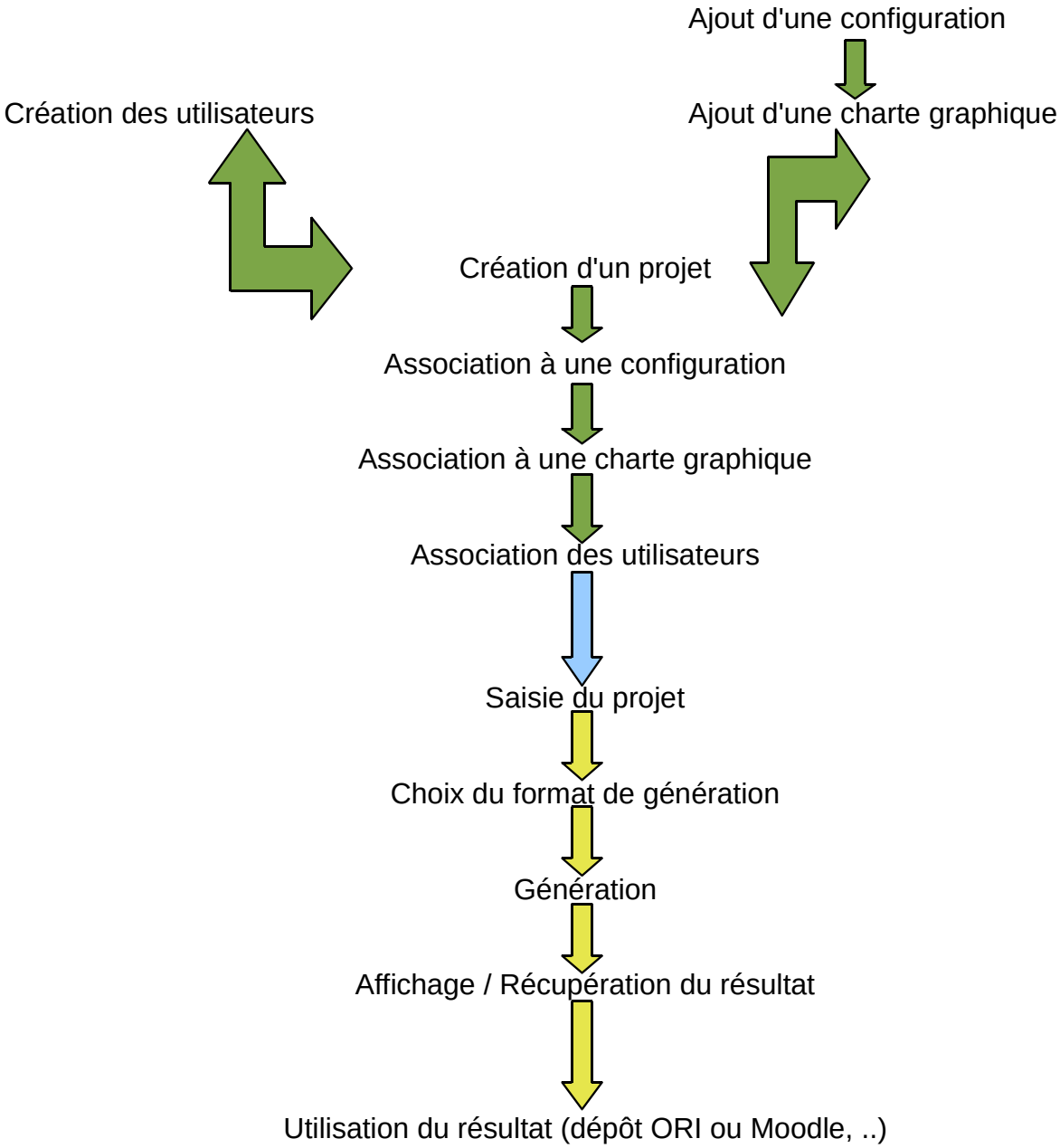
Création, configuration d'une catégorie

Modifier

Configuration d'une catégorie

Supprimer

C. Processus d'utilisation



a. Exercice : Utiliser ChainEdit

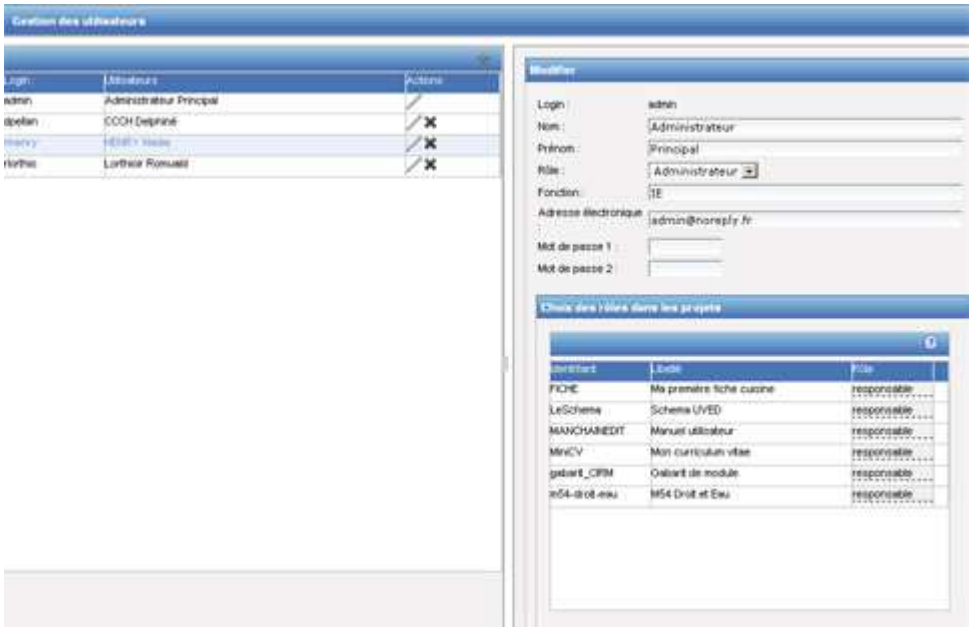
Enoncé :

Après la démonstration d'utilisation qui vous sera faite, utiliser ChainEdit pour éditer un contenu et générer un module.

Démarche :

- Lancer l'application ChainEdit
- Se connecter en admin / admin
- Aller sur l'écran des projets
- Ouvrir le manuel utilisateur (MANCHAINEDIT)
- Modifier le contenu
- Générer le contenu en Html
- Voir le résultat
- Générer le contenu en OpenOffice
- Voir le résultat

II. gestion des utilisateurs



1. Liste des utilisateurs

Le premier écran de gestion des utilisateurs affiche la liste des utilisateurs déjà inscrits dans ChainEdit. A partir de cet écran vous pouvez *ajouter*, *modifier* ou *supprimer* un utilisateur. Pour pouvoir modifier ou supprimer un utilisateur vous devez au préalable en sélectionner un dans la liste puis cliquer sur le bouton associé à l'opération que vous souhaitez effectuer.

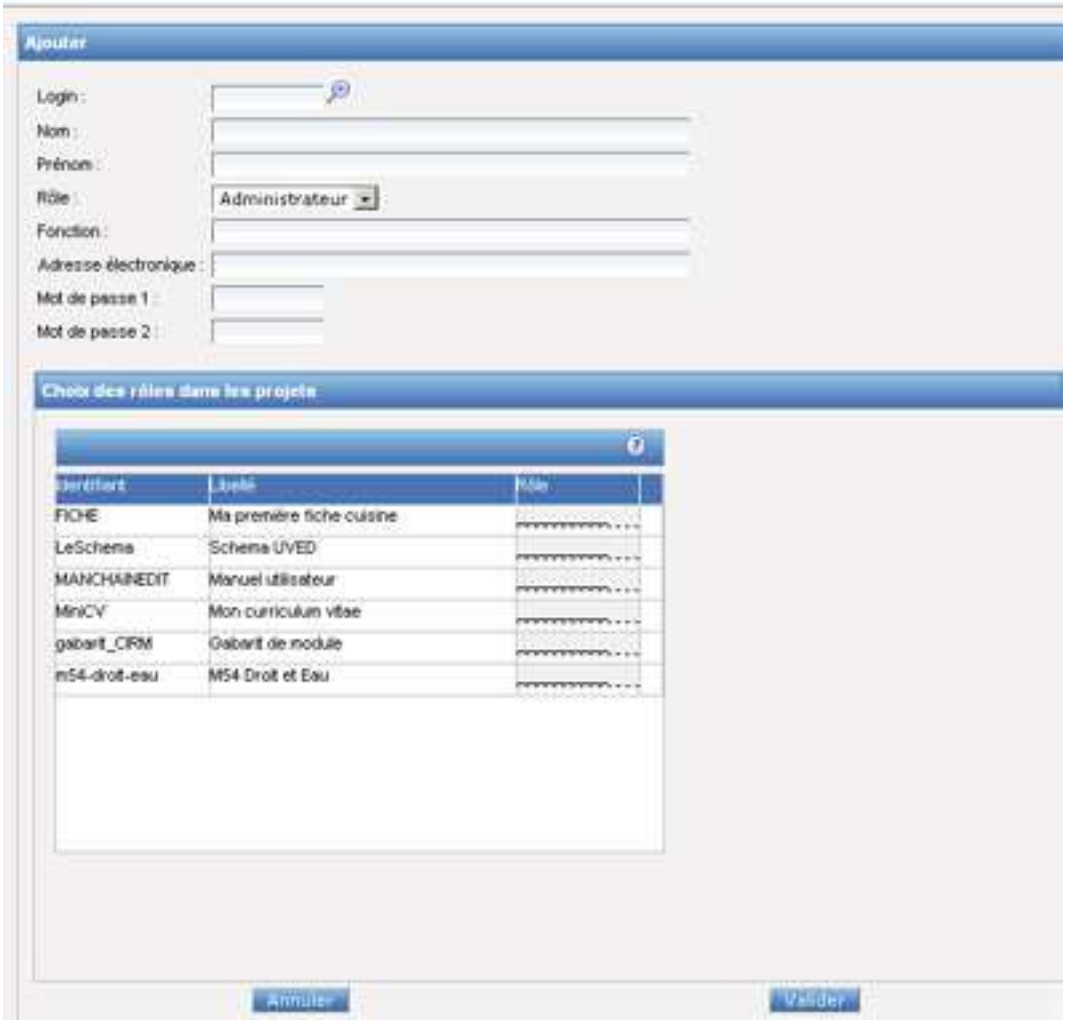
Liste des utilisateurs

2. Ajouter un utilisateur

Pour ajouter un utilisateur vous devez saisir l'ensemble des informations obligatoires : le login, le nom et le prénom.
Si vous disposez d'un annuaire LDAP et que l'application est correctement paramétrée, vous pouvez également aller rechercher les données directement dans cet annuaire. L'ensemble des informations obligatoires sera renseigné (sauf la mot de passe par sécurité).

Vous pouvez dès son ajout lui associer un rôle pour chacun des projets existants.

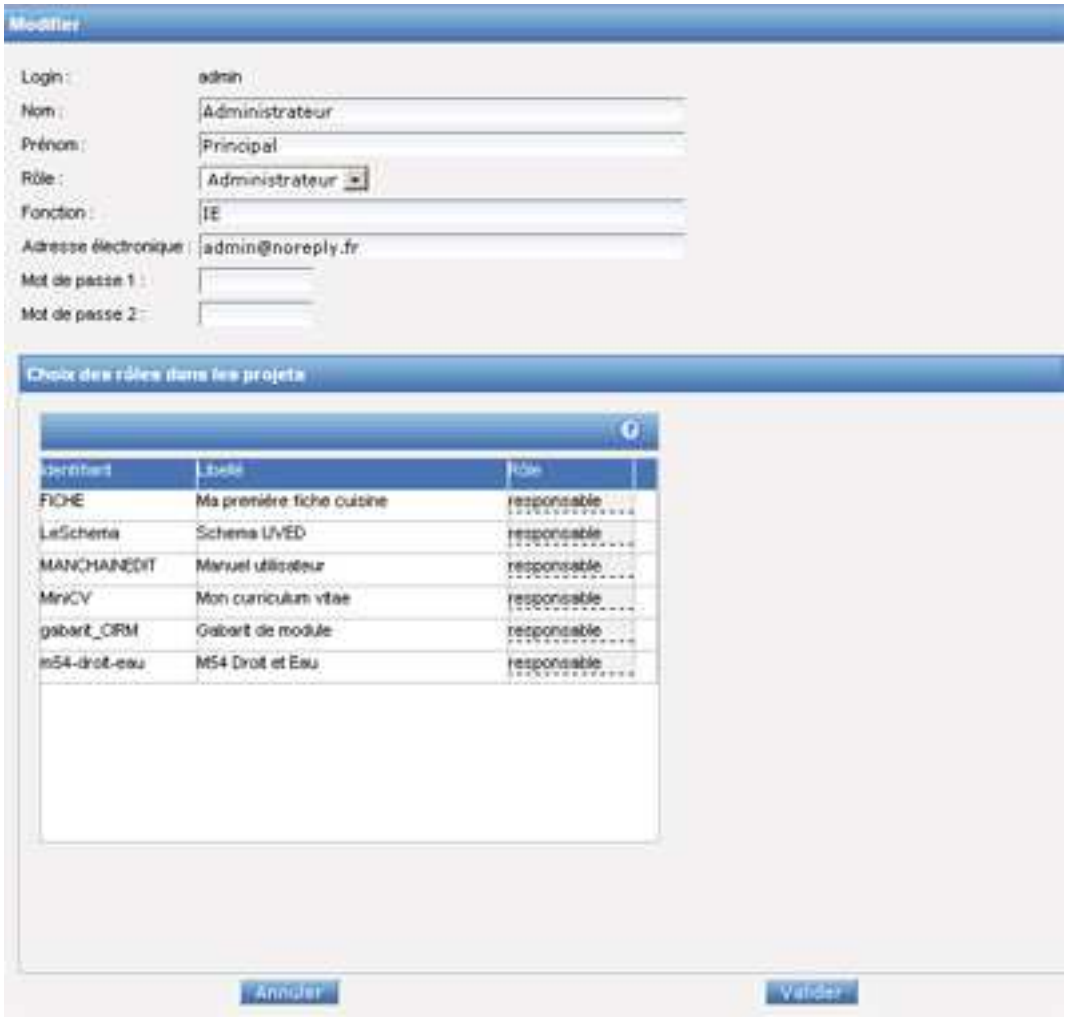
Ajouter un utilisateur



3. Modifier un utilisateur

Pour modifier un utilisateur, vous devez au préalable l'avoir sélectionner dans la liste. Vous ne pouvez pas changer le login.

Modifier un utilisateur



A. Supprimer un utilisateur

Pour supprimer un utilisateur, vous devez au préalable l'avoir sélectionner dans la liste. Vous devrez alors confirmer la suppression. En cas d'utilisation d'un annuaire LDAP, l'utilisateur ne sera pas supprimé de cet annuaire.

Supprimer un utilisateur

III. Modification d'un modèle

1. Travail préparatoire

Il est nécessaire de modifier un modèle dans plusieurs cas :

- Il manque un élément sémantique (balise XML ou attribut);
- Il y a une volonté de changer l'affichage graphique (Infographie);
- Il est nécessaire de changer la génération liée à un élément sémantique.

Pour effectuer ce travail il est nécessaire de bien connaître le besoin. Il faut ensuite l'exprimer techniquement et déterminer les éléments à modifier dans le modèle ChainEdit :

- Modification de la configuration (XSD et XML)
- Modification des feuilles de transformation (XSL)
- Modification de la charte graphique (CSS, scripts et images)

Pour savoir quels éléments vont être impactés par la modification, il faut définir :

- Les balises XML qui seront impactées
- Les attributs ou valeurs d'attributs à modifier
- Les conséquences sur les traitements
- Les éléments graphiques associés à la modification (couleurs, icônes, ..)

Pour les besoins de la formation, nous allons considérer qu'il y avait un besoin pour ajouter un élément sémantique « théorème ». Cet élément serait normalement une nouvelle forme d'un paragraphe, mais nous allons le considérer comme un élément spécifique pour répondre à notre objectif pédagogique. Le niveau de modification n'est donc pas très élevé, mais il va néanmoins être nécessaire d'intervenir dans :

- le fichier XSD lié au schéma CIRM
- le fichier de configuration XML lié au schéma CIRM
- la feuille de traitement XSL
- la CSS pour créer un nouveau style

Nous allons effectuer ces modifications pas à pas pour acquérir une bonne vue des différents éléments constitutifs d'un modèle graphique ChainEdit.

2. Modèle graphique

Dans ChainEdit, un modèle est constitué de 2 éléments :

- une configuration
- une charte graphique

A. Configuration

Une même configuration peut être partagée par plusieurs chartes graphiques. Lorsqu'une modification est effectuée dans une configuration, il faut penser à modifier l'ensemble des chartes graphiques qui y sont liées.

Une configuration est constituée du fichier XSD qui décrit la structure des fichiers XML qui seront saisis dans ChainEdit, d'un fichier de configuration XML qui décrit comment le fichier XSD doit être interprété par ChainEdit et permet également d'ajouter des aides à la saisie. De 2 feuilles de transformation (beforeFck.xsl et afterFck.xsl) qui permettent de transformer les informations saisie dans l'éditeur de ChainEdit pour les mettre dans un format XML et inversement. Ces 2 derniers fichiers seront étudiés plus tard.

B. Charte graphique

Une charte graphique au sens ChainEdit, est un ensemble de traitements qui peuvent être exécutés dans l'objectif de produire un résultat (Html, OpenOffice, Flash, ..). Elle est composée d'un ensemble de fichiers XSL permettant l'exécution de ces traitements et de répertoires qui contiennent les éléments scripts, styles et images de l'interface graphique que l'on souhaite générer.

La génération (c'est à dire les feuilles de transformations XSL) est fortement liée à la configuration à laquelle elle est associée. Par contre, plusieurs chartes graphiques peuvent utiliser des feuilles XSL identiques. Néanmoins, dans ChainEdit, ces feuilles XSL ne sont pas mutualisées. Chaque charte graphique « embarque » donc les feuilles XSL nécessaires à sa génération. Ceci à cause des éventuels écarts qui peuvent se produire lors de la modification d'un modèle. Ainsi, si une configuration est modifiée, et que le traitement XSL a été fait pour une charte graphique, il pourrait être dangereux d'appliquer automatiquement ces modifications aux autres chartes utilisant les mêmes XSL, car les fichiers CSS n'ont peut être pas encore été modifiés pour tenir compte des modifications.

Nous allons donc commencer par modifier la configuration pour répondre à notre nouveau besoin.

3. Modifier une configuration

Une configuration est constituée :

- d'un **fichier XSL** (beforeFck.xsl) permettant de transformer les éléments avant de les donner à TinyMce qui est l'éditeur de texte utilisé dans ChainEdit (Par défaut utilisez ceux donnés en exemple).
- d'un **fichier XSL** (afterFck.xsl) permettant de transformer les éléments après la validation des données de TinyMce
- du **schéma XSD** (nomDuModele.xsd) permettant de valider la conformité du contenu XML.
- d'un **fichier de configuration pour ChainEdit** (nomDuModeleConfig.xml) qui permet de sélectionner les champs du schéma qui devront être saisis, le bon outil pour le faire (Editeur ou champs simple), l'aide contextuelle à afficher, et le libellé « *en français* » qui sera affiché.

A. Schéma XSD

a. Principes du schéma

Nos contenus XML sont normés grâce à un schéma XSD (**XML Schema Definition**).
Le schéma sert à décrire la structure et limiter les contenus.
Le schéma permet d'indiquer le type des éléments du document :

xs:string	Texte
xs:data	Date
xs:integer	Nombre entier (1,2,3,...)

A ces nombreux types décrits sur le site, http://www.w3.org/XML/Schema**,** on peut ajouter des types personnalisés dont le balisage commence pas <xs:simpleType.

Le schéma permet également de préciser le nombre d'occurrences (nombre de fois ou sont répétés les éléments).

minOccurs=	"0" ou "1"
maxOccurs=	"1" ou "unbounded" (illimité)

Un schéma XSD doit commencer par une déclaration, qui indique l'utilisation du langage XML (comme les documents XML eux-mêmes), suivis d'une balise comportant l'indication d'un schéma global.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xml:lang="fr" xmlns:xs="http://www.w3.org/2001/XMLSchema">
...
</xs:schema>
```

C'est entre les deux balises indiquant l'utilisation d'un schéma que l'on déclare tous les éléments du document XML.
Le schéma est composé de balises, qui débutent toutes par **<xs** et se terminent toutes par **</xs:**
Ces balises suivent les mêmes règles que toutes autres balises :

- toute balise ouverte doit être fermée ;
- pas d'enchevêtrement des balises ;
- pas de caractères accentués ou de signes dans l'écriture des balises.

Lien supplémentaire :
Introduction aux schémas W3C XML Schema :
(http://xmlfr.org/documentations/tutoriels/001218-0001)

Partons du schéma OPUS.xsd :

En parcourant ce schéma, on trouve tous les éléments d'un module dont l'élément ZONE.

```
      <xs:element name="ZONE">
            <xs:annotation>
                  <xs:documentation>
                  Insertion d'une zone générique regroupant un ou plusieurs
blocs (paragraphe, liste, tableau, etc...). Cette zone peut être typée afin
d'être mise en exergue.
                  </xs:documentation>
            </xs:annotation>
            <xs:complexType>
                  <xs:choice minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="REDIRECTION" />
                        <xs:group ref="blocks-group" />
                        <xs:group ref="inclusions-group" />
                        <xs:group ref="inclusions-by-theme-group" />
                  </xs:choice>
                  <xs:attribute name="type" type="zones-type" use="optional" />
                  <xs:attribute name="id" type="identifier-type" use="optional"
/>
                  <xs:attribute name="title" type="xs:string" use="optional" />
            </xs:complexType>
      </xs:element>
```

L'élément ZONE peut contenir tous les éléments compris dans les groupes «*blocks-group*» et « *inclusions-group »* ainsi qu'un élément « *REDIRECTION* », un attribut « type » qui peut prendre une des valeurs énumérées dans le type « *zones-type »*, un attribut « *id* » qui respectera le type « *identifier-type* » et enfin, un attribut « *title* ».

Définition du type : « zones-type »

```
      <xs:simpleType name="zones-type">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="attention" />
                  <xs:enumeration value="chemistry" />
                  <xs:enumeration value="citation" />
                  <xs:enumeration value="complement" />
                  <xs:enumeration value="conclusion" />
                  <xs:enumeration value="cf" />
                  <xs:enumeration value="advise" />
                  <xs:enumeration value="definition" />
                  …
            </xs:restriction>
      </xs:simpleType>
```

Définition du type : « identifier-type »

```
      <xs:simpleType name="identifier-type">
            <xs:restriction base="xs:string">
                  <xs:pattern value="[-_a-zA-Z0-9]*" />
            </xs:restriction>
```

```
    </xs:simpleType>
```

Définition du groupe « blocks-group »

```
<xs:group name="blocks-group">
	<xs:choice>
		<xs:element ref="PARAGRAPHE" />
		<xs:element ref="LISTE" />
		<xs:element ref="TABLEAU" />
		<xs:element ref="MEDIA" />
		<xs:element ref="SYNTAXE" />
	</xs:choice>
</xs:group>
```

Voyons dans le détail l'élément « PARAGRAPHE »

```
<xs:element name="PARAGRAPHE">
	<xs:annotation>
		<xs:documentation>
			Insertion d'un paragraphe.
		</xs:documentation>
	</xs:annotation>
	<xs:complexType mixed="true">
		<xs:choice minOccurs="0" maxOccurs="unbounded">
			<xs:group ref="equations-group" />
			<xs:group ref="links-group" />
			<xs:group ref="texts-group" />
		</xs:choice>
		<xs:attribute name="id" type="identifier-type" use="optional"
/>
		<xs:attribute name="title" type="xs:string" use="optional" />
	</xs:complexType>
</xs:element>
```

b. Exercice : Modification d'un schéma

Enoncé : Créer dans le schéma OPUS.xsd, un nouvel élément THEOREME qui possèdera un attribut auteur obligatoire, un attribut titre optionnel et pourrait contenir l'ensemble des éléments des groupes « texts-group » et « links-group »

Démarche :

1. Trouver un élément proche de l'élément à créer et le copier
2. Le placer dans le fichier en respectant l'organisation des appels d'éléments
3. Appeler l'élément « THEOREME » dans les éléments dans lesquels on veut pouvoir l'intégrer.

B. Fichier de configuration XML

a. Principes du fichier de configuration

On définit dans notre fichier de configuration xml la façon dont seront saisis les éléments du schéma.

Le fichier de configuration de ChainEdit spécifie le schéma XML (fichier.xsd) à utiliser pour la validation et la façon dont seront saisi les éléments.

i. Eléments des fichiers de configuration

CONFIG

Elément racine des fichiers de configuration

FICHIERSCHEMA

Nom du fichier schéma à utiliser, spécifié avec l'attribut `nom`.

RACINE

Indication d'un élément racine du schéma. `RACINE` doit contenir un élément `BALISE`, avec la description de l'élément racine.

BALISE

Informations relatives à un élément du langage XML décrit. Les attributs sont:

- `nom` : nom de l'élément
- `titre` : titre du menu pour l'insertion de cet élément
- `type` : type d'élément. Les valeurs possibles pour `type` sont : `typeselect, typesimple, typevide, typedate, typefck, typechemin`.
- `groupe` : Pour un regroupement par groupe, des balises dans le menu.

ATTRIBUT

Attribut d'un élément (doit se trouver sous `BALISE`). Les attributs possibles sont `nom` (nom de l'attribut), titre (titre de l'attribut) et `type` (`valeurs possibles : typeselect, typesimple, typechemin`) et display (indique si la valeur de cet attribut est affiché près du nom de l'élément dans le menu).

SELECTVALUE

Valeurs d'un élément ATTRIBUT (doit se trouver sous `ATTRIBUT`). Lorsque l'attribut `type` de l'ATTRIBUT est égal à `typeselect`, l'ATTRIBUT peut prendre une des valeurs définies parmi les enfants SELECTVALUE. Si SELECTVALUE est absent l'attribut pourra prendre une valeur définie dans le schéma associé.

AIDE

Aide contextuelle apportée à la saisie d'un élément ou d'un attribut. (doit se trouver sous `BALISE` ou `ATTRIBUT`). Ne possède aucun attribut.

ICONES

Parent des éléments icône. Le seul attribut possible est chemin (chemin des icônes à partir de la racine du répertoire de la configuration).

ICONEOPEN

Nom de l'image à afficher dans le menu devant un élément ouvert dans l'arbre et n'étant pas en erreur dans la validation du XML. Le seul attribut possible et obligatoire est valeur.

ICONECLOSED

Nom de l'image à afficher dans le menu devant un élément fermé dans l'arbre et n'étant pas en erreur dans la validation du XML. Le seul attribut possible et obligatoire est valeur.

ICONEERROROPEN

Nom de l'image à afficher dans le menu devant un élément ouvert dans l'arbre et étant en erreur dans la validation du XML. Le seul attribut possible et obligatoire est valeur.

ICONEERRORCLOSED

Nom de l'image à afficher dans le menu devant un élément fermé dans l'arbre et étant en erreur dans la validation du XML. Le seul attribut possible et obligatoire est valeur.

PARAMETRE

Paramètre d'un élément (doit se trouver sous BALISE notamment lorsque l'attribut type a la valeur typeditor). Permet à l'application de trouver la classe permettant de lancer les transformations xsl (beforeFCK et afterFCK) ainsi que la valeur de la barre d'outil visible dans l'éditeur TinyMCE. Les attributs possibles sont nom (nom de l'attribut), valeur (valeur de l'attribut).

Les différents types que peuvent prendre les éléments **ATTRIBUT** et **BALISE** indiquent quels éléments de formulaire seront affichés dans l'interface pour la saisie de ces éléments et attributs. Ils peuvent prendre les valeurs suivantes.

- typeselect : liste à choix
- typesimple : champ de saisie
- typevide : Aucun champ de formulaire
- typedate : champ de saisie dont le contenu devra être au format date (jj/mm/aaaa)
- typeditor : zone de saisie TinyMce
- typechemin : champ de saisie dont le contenu devra être au format chemin

Si on se reporte au fichier de configuration OPUSConfig.xml, la façon dont sera saisi un élément « PARAGRAPHE » est la suivante :

```
      <BALISE nom="PARAGRAPHE" titre="Paragraphe " type="typeEditor">
            <ICONES chemin="OPUS/">
                  <ICONEOPEN valeur="tree-icons/default/paragraphe.png" />
                  <ICONECLOSED valeur="tree-icons/default/paragraphe.png" />
                  <ICONEERROROPEN valeur="tree-icons/error/paragraphe.png" />
                  <ICONEERRORCLOSED valeur="tree-icons/error/paragraphe.png" />
            </ICONES>
            <PARAMETRE nom="configEditorPlugins"
valeur="safari,spellchecker,pagebreak,style,layer,table,save,advhr,advimage,adv
link,emotions,iespell,inlinepopups,insertdatetime,preview,media,searchreplace,p
rint,paste,directionality,fullscreen,noneditable,visualchars,nonbreaking,xhtmlx
tras,template,refglossaire,refbiblio,refwebo,doclink,externallink,tooltiplink,i
nterncourslink,internpracticlink,interncaselink,internexercicelink,internmedial
ink,internannexlink,equajaxe,equatex,mathml,thumblink" />
            <PARAMETRE nom="configEditorToolbar1"
valeur="fullscreen,code,print,|,undo,redo,|,cut,copy,paste,pastetext,pasteword,
|,search,replace" />
            <PARAMETRE nom="configEditorToolbar2"
valeur="bold,italic,sub,sup,|,charmap,insertdate,inserttime" />
            <PARAMETRE nom="configEditorToolbar3"
valeur="refglossaire,refbiblio,refwebo,|,doclink,externallink,tooltiplink,|,int
erncourslink,internpracticlink,interncaselink,internexercicelink,|,internmedial
ink,internannexlink,|,equajaxe,equatex,mathml,|,thumblink" />
            <PARAMETRE nom="extended_valid_elements" valeur="a[*],img[*]" />
            <PARAMETRE nom="classEditor"
valeur="org.esupportail.chainedit.external.interfaces.defaultImpl.EditorTransfo
rmImpl" />
            <ATTRIBUT nom="id" titre="Identifiant unique " type="typesimple">
                  <AIDE>Identifiant unique du paragraphe</AIDE>
            </ATTRIBUT>
                  <ATTRIBUT nom="title" titre="Titre " type="typesimple">
                  <AIDE>Titre du paragraphe</AIDE>
            </ATTRIBUT>
            </BALISE>
```

On voit que notre élément et tous ces enfants seront affichés et saisis dans l'éditeur TinyMCE. Il faut renseigner le paramètre « *configEditorPlugins* » et « *configEditorToolbar »*, qui précisent les plugins et les barres d'outils à utiliser. Ce paramètre doit être obligatoirement présent. Un fichier de configuration de l'éditeur TinyMCE précise les éléments qui devront être présents pour une barre d'outils donnée.

Ces barres d'outils ont été adaptées au modèle afin de limiter les possibilités de formatage du texte. Ceci a été pensé de façon à respecter le principe de séparation du fond et de la forme. Les couleurs, alinéas, etc... sont gérées par la charte sélectionnée à la génération.

b. Exercice : Modification du fichier de configuration

Enoncé : Créez dans le fichier de configuration OPUSConfig.xml, la prise en charge de notre nouvel élément THEOREME.

Démarche :

1. Trouvez un élément proche de l'élément à créer et le copier.
2. Placez le dans le fichier en respectant l'organisation des appels d'éléments.
3. Appelez l'élément « THEOREME » dans les éléments dans lesquels on veut pouvoir l'intégrer.

C. Utilitaires beforeFck et afterFck

a. Principes de ces utilitaires

Notre élément paragraphe dans le contenu XML aura la forme suivante :

```
<PARAGRAPHE>Ici, le contenu texte d'un élément paragraphe standard pouvant contenir des éléments <TEXTE-GRAS>STRONG</TEXTE-GRAS>, <TEXTE-ITALIQUE>I</TEXTE-ITALIQUE>, <LIEN-INTERNE-REFERENCE-GLOSSAIRE reference-id="Mot">mot</LIEN-INTERNE-REFERENCE-GLOSSAIRE> et sa définition
</PARAGRAPHE>
```

Le format du texte affiché dans TinyMCE doit être en HTML. La transformation xsl de notre fichier beforeFck permet de transformer les éléments en un format html compris de TinyMCE.

Notre élément transformé par le fichier XSL devient :

```
Ici, le contenu texte d'un &eacute;l&eacute;ment paragraphe standard pouvant contenir des &eacute;l&eacute;ments <strong>STRONG</strong>, <em>I</em>, <a id="mot" rel="lienrefglossaire" rev="false" href="lien fictif">mot</a>
```

Après saisie dans l'éditeur TinyMCE, le fichier afterFck permet, la transformation inverse. Le html produit par l'éditeur redevient un xml conforme au modèle.

b. Exercice : Dépôt des configurations dans ChainEdit

Enoncé : L'ensemble des fichiers constituant une configuration étant maintenant modifiés, mettez à jour l'application avec cette nouvelle configuration via l'interface web.

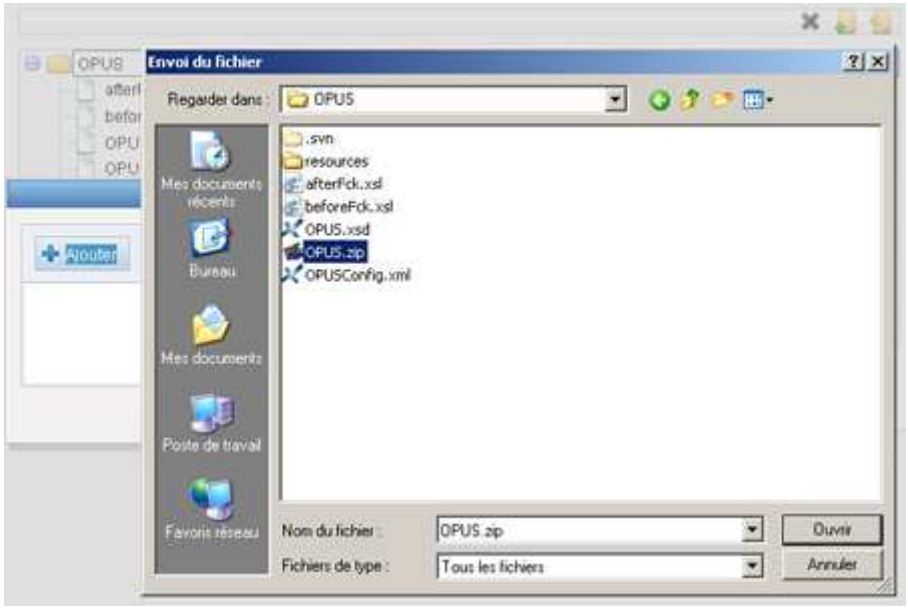
Démarche :

1. Sous l'onglet administration, sélectionnez « Gestion des configurations »

2. Dans l'arborescence sélectionnez la configuration que l'on souhaite modifier.

3. Pour déposer un fichier sélectionnez un répertoire et cliquez sur l'icône « télécharger sur le serveur » en haut à droite du cadre.

4.



5. Pour renommer ou supprimer un fichier de configuration, sélectionnez le fichier dans la liste. Faites un clic droit sur ce fichier ou utiliser les icônes en haut à droite du cadre. Si votre sélection porte sur un zip, l'option « dézipper le fichier » vous seras proposée.

6. Une fois que la sélection des fichiers est terminée, validez les changements.

c. Exercice : Ajouter un élément théorème

Enoncé : Nous pouvons maintenant ajouter des éléments théorème dans un projet.

Démarche :

1. Ouvrez un projet respectant le schéma OPUS.
2. Ajoutez un élément théorème.

Ce qui donne dans notre contenu XML :

```
<THEOREME auteur="Pythagore" titre="Théorème de Pythagore">Dans un triangle rectangle, le carré de la longueur de l'<LIEN-INTERNE-REFERENCE-GLOSSAIRE iconifiable="false" reference-id="hypoténuse"><TERME>hypoténuse </TERME></LIEN-INTERNE-REFERENCE-GLOSSAIRE> est égal à la somme des carrés des longueurs des côtés de l'angle droit.</THEOREME>
```

3. Générez le projet.
4. Visualisez l'élément théorème.

4. Modification de la charte graphique

Une charte graphique dans ChainEdit est un ensemble de feuilles de transformation XSL, associées à des traitements qui vont permettre de générer le résultat attendu (Html, OO, ..), et un ensemble de fichiers ou répertoires qui vont contenir les éléments graphiques (CSS, images, styles.xml, ..).

Ces deux ensembles d'éléments sont dépendant des fichiers de configuration qui ont été créés (XSD). Le travail de modification peut se mener en parallèle. L'infographiste modifie la CSS pour tenir compte du ou des nouveaux éléments, pendant ce temps, une autre personne peut modifier les feuilles de transformations pour générer le code nécessaire à la prise en compte du nouvel élément. Bien entendu, il est nécessaire de connaître les instructions Html nécessaires pour la bonne prise en compte par la CSS.

Si plusieurs chartes utilisent ces éléments, il faut toutes les mettre à jour.

Maintenant que nous avons ajouté un élément « THEOREME », il est nécessaire de modifier la charte graphique pour en tenir compte.

A. Transformation XSL

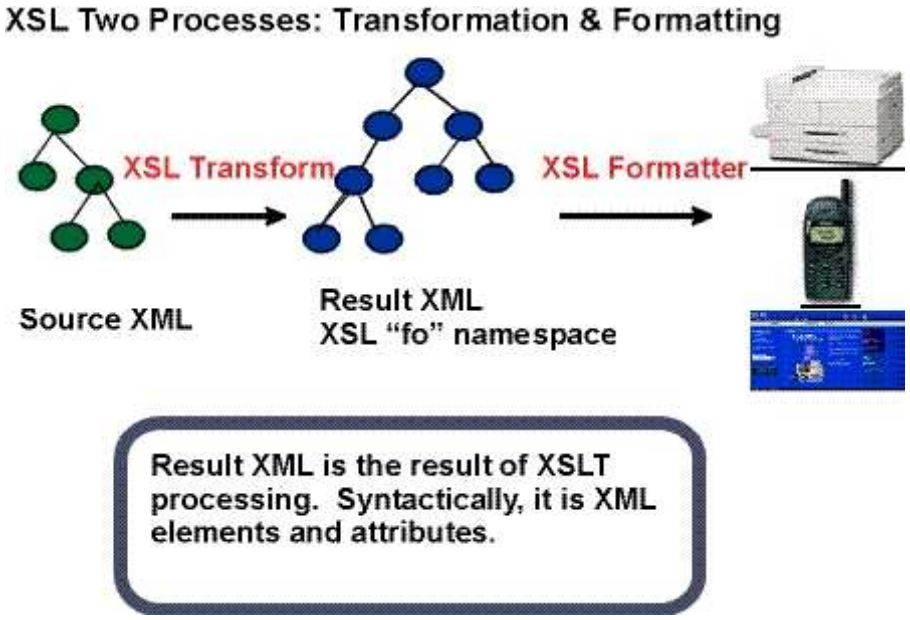
a. Principes de la transformation

Une feuille de styles XSL est définie comme un document XML contenant des spécifications de transformation et de formatage d'objets. Elle permet de transformer un document XML d'entrée en un autre document XML, dont les éléments de structure sont tous liés à des éléments typographiques représentant des intentions de formatage : des pages, des fenêtres, des paragraphes, des listes, etc.

Le document XML résultant de la transformation doit ensuite être pris en charge par un outil de formatage, qui créera une version web, HTML, FLASH, ... ou une version papier, PDF, RTF, ... ou tout autre format. La spécification différencie donc deux processus : la transformation d'arbre (*tree transformation*) et le formatage (*formating*).

Le document présenté pouvant être structurellement très différent du document XML d'origine, tout le pouvoir de transformation d'XSLT doit pouvoir être utilisé dans la première partie du processus, pour ajouter, par exemple, des tables de matières ou encore filtrer et réordonner des informations.

Principe de fonctionnement des processeurs XSL (issu de [recommandation Extensible Stylesheet Language]

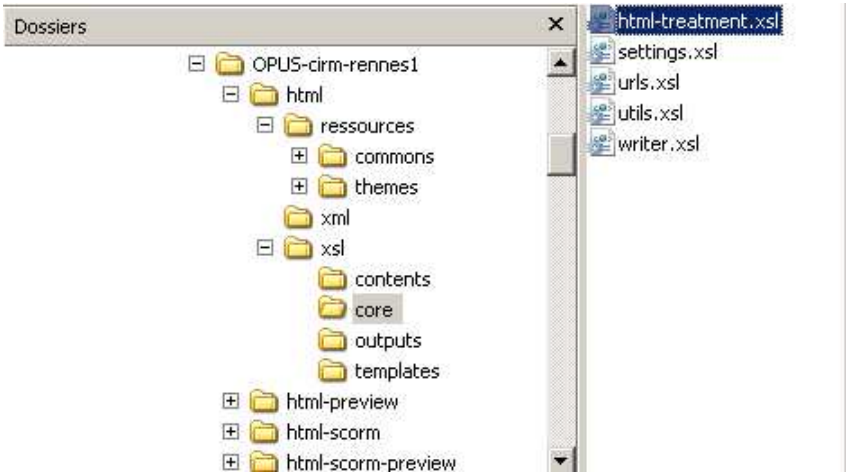


Lien supplémentaire :

Fiche technique XSL – auteur : http://www.3ie.fr/

i. Organisation dans ChainEdit

Les fichiers sont organisés par charte, puis par traitement. Les feuilles de traitements XSL qui seront enchaînés pour obtenir le résultat doivent se trouver dans le répertoire « xsl » du répertoire d'un traitement. Il est conseillé de créer un fichier XSL qui appelle tous les autres plutôt que d'enchaîner les appels dans ChainEdit.



Ces feuilles XSL vont prendre en paramètre le chemin où elles devront créer les fichiers à générer (par exemple : « .../projets/monprojet »). Elles vont ensuite générer les pages dans des sous répertoires (par exemple « /site » pour du Html). Ce dernier chemin est écrit « en dur » dans la XSL. Lors de la déclaration de la charte graphique dans ChainEdit, il faut donc que les informations indiquées correspondent. Ainsi, si les feuilles XSL vont créer des pages html dans un répertoire « /site/html », il faut, lors de la déclaration de la charte, indiquer la création d'un répertoire « /site » et d'un répertoire « /site/html ».

Il faut ensuite indiquer où le répertoire (généralement « /ressources ») qui contient les CSS, scripts, images, doit être copié (par exemple dans « /site »). Il faudra indiquer dans quel répertoire les éléments ajoutés par les auteurs (images, animations, ..) devront être copiés (par exemple « /site/html »).
Enfin, il faut indiquer quel répertoire contient la page index.html dans le cas d'une

génération Html.

Le fichier « charte.xml » à la racine du répertoire de chaque charte, décrit l'ensemble de ces informations. L'application viendra lire ce dernier et le mettre à jour via l'interface.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CHARTE id="OPUS-cirm-rennes1" label="[OPUS] - cirm-rennes1 - 1.0.0">
    <CHARTECONFIG id="OPUS\OPUSConfig.xml"/>
    <TRAITEMENTS nature="html" type="autre">
        <FICHIERXSL ordre="0" chemin="\html\xsl\html-master.xsl"/>
        <FICHIERXSL ordre="1" chemin="\html\xsl\xml-master.xsl"/>
        <FICHIERXSL ordre="2" chemin="\html\xsl\txt-master.xsl"/>
        <REPERTOIRECHARTE chemin="\html\ressources"/>
        <REPERTOIRECHARTE chemin="\html\.htaccess"/>
        <REPERTOIRECIBLE chemin="\site" defaut="non" charte="oui"
ressource="non"/>
        <REPERTOIRECIBLE chemin="\site\html" defaut="oui" charte="non"
ressource="oui"/>
    </TRAITEMENTS>
    <TRAITEMENTS nature="html-preview" type="preview">
        <FICHIERXSL ordre="0" chemin="\html-preview\xsl\html-master.xsl"/>
        <FICHIERXSL ordre="1" chemin="\html-preview\xsl\xml-master.xsl"/>
        <FICHIERXSL ordre="2" chemin="\html-preview\xsl\txt-master.xsl"/>
        <REPERTOIRECHARTE chemin="\html-preview\ressources"/>
        <REPERTOIRECHARTE chemin="\html-preview\.htaccess"/>
        <REPERTOIRECIBLE chemin="\site" defaut="non" charte="oui"
ressource="non"/>
        <REPERTOIRECIBLE chemin="\site\html" defaut="oui" charte="non"
ressource="oui"/>
    </TRAITEMENTS>
    <TRAITEMENTS nature="html-scorm" type="autre">
        <FICHIERXSL ordre="0" chemin="\html-scorm\xsl\html-master.xsl"/>
        <FICHIERXSL ordre="1" chemin="\html-scorm\xsl\xml-master.xsl"/>
        <FICHIERXSL ordre="2" chemin="\html-scorm\xsl\txt-master.xsl"/>
        <REPERTOIRECHARTE chemin="\html-scorm\ressources"/>
        <REPERTOIRECHARTE chemin="\html-scorm\.htaccess"/>
        <REPERTOIRECIBLE chemin="\site" defaut="non" charte="oui"
ressource="non"/>
        <REPERTOIRECIBLE chemin="\site\html" defaut="oui" charte="non"
ressource="oui"/>
    </TRAITEMENTS>
    <TRAITEMENTS nature="html-scorm-preview" type="autre">
        <FICHIERXSL ordre="0" chemin="\html-scorm-preview\xsl\html-master.xsl"/>
        <FICHIERXSL ordre="1" chemin="\html-scorm-preview\xsl\xml-master.xsl"/>
        <FICHIERXSL ordre="2" chemin="\html-scorm-preview\xsl\txt-master.xsl"/>
        <REPERTOIRECHARTE chemin="\html-scorm-preview\ressources"/>
        <REPERTOIRECHARTE chemin="\html-scorm-preview\.htaccess"/>
        <REPERTOIRECIBLE chemin="\site" defaut="non" charte="oui"
ressource="non"/>
        <REPERTOIRECIBLE chemin="\site\html" defaut="oui" charte="non"
ressource="oui"/>
    </TRAITEMENTS>
    <TRAITEMENTS nature="open-office" type="openOffice">
        <FICHIERXSL ordre="0" chemin="\open-office\xsl\xml-master.xsl"/>
        <FICHIERXSL ordre="1" chemin="\open-office\xsl\txt-master.xsl"/>
        <REPERTOIRECHARTE chemin="\open-office\Pictures"/>
        <REPERTOIRECHARTE chemin="\open-office\styles.xml"/>
        <REPERTOIRECIBLE chemin="\oo" defaut="oui" charte="oui"
ressource="non"/>
```

```
    </TRAITEMENTS>
</CHARTE>
```

La génération Html étant la plus simple c'est celle qui sera montée lors de la formation.

Des générations existent également pour obtenir des contenus en SCORM ou en OpenOffice. Elles pourront être montrées à titre d'exemple. Elles répondent au même principe, mais doivent générer dans d'autres normes que le Html ce qui les rend plus complexe à maîtriser pour une formation courte.

Le détail des feuilles de transformation de la charte Rennes1 à partir du schéma OPUS font l'objet d'une autre documentation plus détaillée.

ii. Modifier le XSL

Ce nouveau type « THEOREME » existe désormais dans notre contenu XML. Occupons-nous de son aspect à la génération.

Le plus simple est de trouver un autre élément dont nous pourrions copier le fonctionnement puis le modifier en fonction de nos besoins. Dans l'exemple suivant, nous avons justement un élément PARAGRAPHE dont le mode de fonctionnement est très proche de celui que nous souhaitons.

Nous allons étudier le fonctionnement de ce PARAGRAPHE et voir comment nous pourrions le réutiliser.

Extrait XSL du traitement d'un paragraphe:

```
<!-- ********************************************** -->
<!-- Traitement de l'élément PARAGRAPHE  -->
<!-- ********************************************** -->
   <xsl:template match="PARAGRAPHE">
      <xsl:choose>
            <xsl:when test="@type='STANDARD'">
            <div class="bloc-standard">
               <xsl:if test="@titre">
                  <h6><xsl:value-of select="@titre"/></h6>
               </xsl:if>
               <p>
                  <xsl:apply-templates />
               </p>
            </div>
            </xsl:when>
            <xsl:when test="@type='ACTIVITE'">
            <div class="bloc-activite">
               <h6> Activité </h6>
               <div class="content">
                     <xsl:if test="@titre">
                        <b><xsl:value-of select="@titre"/></b>
                        <br/>
                     </xsl:if>
                  <xsl:apply-templates />
               </div>
            </div>
            </xsl:when>
            .
            .
            .
            .
            <xsl:when test="@type='SYNTAXE'">
            <div class="bloc-syntaxe">
                 <h6><xsl:value-of select="./@titre"/></h6>
               <div class="content">
                 <pre class="commande"><xsl:apply-templates />
                 </pre>
               </div>
            </div>
            </xsl:when>
            <xsl:otherwise>
            <div class="bloc-standard">
               <xsl:if test="@titre">
                  <h6><xsl:value-of select="@titre"/></h6>
               </xsl:if>
               <p>
                  <xsl:apply-templates />
               </p>
            </div>
            </xsl:otherwise>
      </xsl:choose>
   </xsl:template>
```

b. Exercice : Modification des fichiers XSL

Enoncé : Ajoutez dans le fichier traitant du détail d'un paragraphe les instructions nécessaires à la prise en charge de l'élément « THEOREME ».

Démarche :

1. Trouvez le fichier xsl dans lequel sont traités en détail les éléments.
2. Ajoutez les lignes nécessaires au traitement d'un élément paragraphe et de ses attributs.

B. Transformation de la CSS

a. Principes des feuilles de styles graphiques

Rappel : Une charte graphique est composée des feuilles de transformation xsl et de l'ensemble des fichiers qui composent une charte graphique, CSS, Javascripts, images,... qui sont appelés dans les xsl.
Dans le cas d'une charte qui génèrera une version html, tous ces éléments sont présents sous le répertoire /chartes/OPUS/html/ressources (*cf:arborescence d'une charte*) mais un administrateur peut choisir d'organiser son arborescence comme bon lui semble. Les éléments de la charte graphique sont ensuite subdivisés. On y trouve entre autres :

- Un répertoire **js** : comprenant l'ensemble des fichiers java script inclus dans les pages html générées.
- Un répertoire **css** : comprenant l'ensemble des fichiers css définissant les classes de style.

Nous avons besoin pour distinguer l'élément THEOREME dans une page html de créer une classe qui lui sera dédiée.

Le détail des feuilles de style de la charte Rennes1 à partir du schéma OPUS font également l'objet d'une autre documentation plus détaillée.

b. Exercice : Modification d'une CSS

Enoncé : Créez les styles prenant en charge l'élément THEOREME et ses attributs.

Démarche :

1. Trouvez la feuille de styles traitant des types de paragraphe.
2. Prenez un exemple se rapprochant de l'élément THEOREME et copiez le
3. Créez les styles pour l'élément THEOREME.

c. Exercice : Mise à jour de la charte graphique dans ChainEdit

Enoncé : Mettez à jour le fichier CSS modifié dans la charte graphique CIRM.

Démarche :

1. Allez dans l'écran des chartes
2. Allez dans la gestion des répertoires
3. Allez dans le répertoire de la CSS
4. Ajoutez le fichier modifié

5. Création d'un modèle

Nous venons de modifier la saisie de contenus dans un contexte visant à créer des contenus pédagogiques existant. Mais ça peut être des contenus de formats voués à un autre contexte, exemple un CV. Nous allons construire et prendre en charge un nouveau modèle : un mini CV.

A. Travail préparatoire

Nos contenus XML sont normés grâce à un schéma XSD (**XML Schema Definition**). Le schéma sert à décrire la structure et limiter le contenu de notre mini CV.

Nous allons maintenant définir les éléments de notre mini CV et en faire l'implémentation complète.

a. Exercice : Réflexion sur les éléments du mini CV

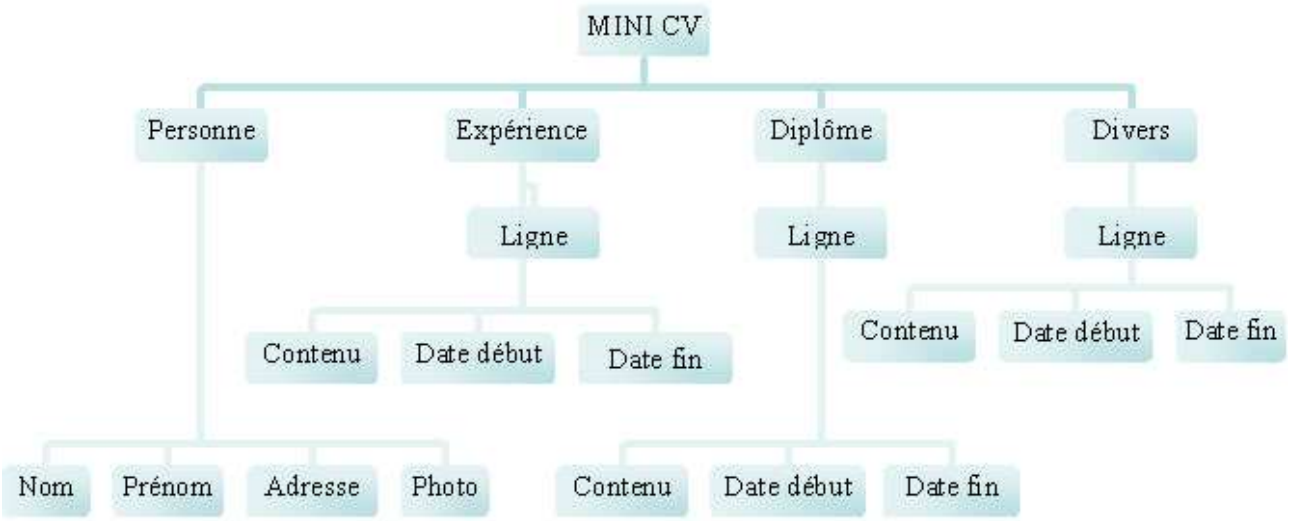
Enoncé : Définir les éléments qui seront présents dans le mini CV.

Démarche :

1. Jus de cervelle !!!!

B. Création du schéma XSD

Le schéma ci-dessous représente les éléments que nous allons retenir pour faire notre modèle de CV. Vous pouvez l'enrichir de vos propres éléments.



a. Exercice : Création de la configuration dans ChainEdit

Enoncé : Déposez les fichiers de configuration mini CV fournis dans ChainEdit.

Démarche :

1. Connectez-vous à ChainEdit
2. Allez dans le menu d'administration
3. Allez dans la gestion des configurations
4. Ajoutez une configuration en créant un répertoire MINICV et en y déposant le XSD et le MINICVConfig.xml et le beforeFCK et afterFCK associés.

b. Exercice : Création d'un projet de mini CV

Enoncé : Créez un nouveau projet de mini CV.

Démarche :

1. Connectez-vous à ChainEdit
2. Allez dans le menu d'administration
3. Allez dans la gestion des projets
4. Ajoutez un nouveau projet
5. Donnez lui un code, un nom, affectez le fichier de configuration et sélectionnez les utilisateurs
6. Après avoir valider, allez dans l'écran projets et éditez votre mini-CV (si vous le pouvez !!!).
7. Saisissez quelques informations.

C. Transformation xsl

Maintenant que l'on a un fichier XML et qu'il est valide, nous allons le transformer en une page HTML. Prenons un exemple de fichier XML correspondant au modèle:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<MINICV>
	<PERSONNE nom="Daniel" prenom="Tessier">
		<ADRESSE>206 Avenue du général Leclerc<BR/>35000 Rennes</ADRESSE>
		<PHOTO chemin="ressources/photo.jpg" format="jpg"/>
	</PERSONNE>
	<EXPERIENCE>
		<LIGNE>
			<DATEDEBUT>12/01/2001</DATEDEBUT>
			<DATEFIN>20/12/2001</DATEFIN>
			<CONTENU>
				<B>Mairie de Chassagnet</B><BR/>. mise en page du
				journal de la ville (diffusion 25 000 exemplaires)<BR/>.
				Mise en page de  documents divers (information des
				associations, prospectus,&#8230;<BR/>. contacts avec
				l&#8217;imprimeur
			</CONTENU>
		</LIGNE>
		<LIGNE>
			<DATEDEBUT>01/01/1999</DATEDEBUT>
			<DATEFIN>03/01/2000</DATEFIN>
			<CONTENU>
				<B>ECALYPSO</B>, agence de graphisme<BR/>Stage de
				maquettiste<BR/>. mise en page des documents des clients
				(revues, livres)
			</CONTENU>
		</LIGNE>
	</EXPERIENCE>
	<DIPLOMES>
		<LIGNE>
			<DATEDEBUT>01/02/1998</DATEDEBUT>
			<DATEFIN>30/11/1998</DATEFIN>
			<CONTENU>Formation PAO au Centre de formation du multimédia
				(Lannion)</CONTENU>
		</LIGNE>
	</DIPLOMES>
	<DIVERS>
		<LIGNE>
			<CONTENU>Peinture (huile, aquarelle)<BR/></CONTENU>
		</LIGNE>
		<LIGNE>
			<CONTENU>Natation</CONTENU>
		</LIGNE>
		<LIGNE>
			<DATEDEBUT>01/09/1997</DATEDEBUT>
			<CONTENU>Président du club de football de Saint Maugan
			</CONTENU>
		</LIGNE>
	</DIVERS>
</MINICV>
```

Nous aimerions transformer ce fichier XML en un fichier HTML qui aurait le code source suivant:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Curriculum vitae</title>
  </head>
      <link rel="stylesheet" type="text/css" href="medias/styles/style.css"
media="screen" />
  <body>
      <div id="wrapper">
            <h1>Daniel Tessier - CV</h1>
            <div class="adresse">206 Avenue du général Leclerc<br/>35000 Rennes
            </div>
            <img src="ressources/images/photo.jpg" />
            <div class="experience">
                  <table>
                  <tr>
                  <td class="date"> 12/01/2001</td>
                  <td rowspan="2"><b>Mairie de Chassagnet.</b><br/>Mise en page
du journal de la ville (diffusion 25 000 exemplaires).<br/>Mise en page de
documents divers (information des associations,
prospectus,&#8230;.<br/>Contacts avec l&#8217;imprimeur</td>
                  </tr>
                  <tr>
                  <td class="date"> 20/12/2001</td>
                  </tr>
                  <tr>
                  <td class="date">01/01/1999</td>
                  <td rowspan="2"><b>ECALYPSO</b>, agence de graphisme<br/>Stage
de maquettiste.<br/>Mise en page des documents des clients (revues,
livres)</td>
                  </tr>
                  <tr>
                  <td class="date">03/01/2000</td>
                  </tr>
                  </table>
            </div>
            <div class="diplomes">
                  <table>
                  <tr>
                  <td class="date">01/02/1998</td>
                  <td rowspan="2">Formation PAO au Centre de formation du
multimédia (Lannion)</td>
                  </tr>
                  <tr>
                  <td class="date">30/11/1998</td>
                  </tr>
                  </table>
            </div>
            <div class="divers">
                  <table>
                  <tr>
                  <td class="date"> </td>
```

```
                    <td>Peinture (huile, aquarelle)</td>
                    </tr>
                    <tr>
                    <td class="date"> </td>
                    <td>Natation</td>
                    </tr>
                    <tr>
                    <td class="date">01/09/1997</td>
                    <td>Président du club de football de Saint Maugan</td>
                    </tr>
                    </table>
                </div>
            </div>
        </body>
    </html>
```

a. 1ère étape : entête du document XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/">
        <html>
            <head><title>Curriculum vitae</title></head>
            <body>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

En appliquant cette feuille de style sur n'importe quel fichier XML on aura:

```
<html>
  <head>
    <title>Curriculum vitae</title>
  </head>
  <body>
  </body>
</html>
```

Remarques:

1. A la première ligne, on précise la version d'XML que l'on utilise, (n'oublions pas qu'un fichier XSLT est un fichier XML) ainsi que l'encodage (ici l'encodage europe de l'ouest)
2. Ensuite on trouve ce qui différencie un fichier XSLT d'un fichier XML. La racine du fichier est imposée, elle doit être *<xsl:stylesheet>* (ou son alias *<xsl:transform>*) et doit préciser la version d'XSLT utilisée.
3. On définit le type de fichier de sortie dont 3 sont reconnus: text, xml ou html. Ici on a bien entendu choisit HTML. On a également défini l'encodage.
4. *<xsl:template match="/">* est évoqué dés que la racine du fichier XML à transformer est rencontrée. Dans le cas présent dès que la racine est affichée le parseur écrit le contenu de la balise <xsl:template/>

Sur cet exemple, on ne le voit pas très bien mais un fichier XSLT n'est pas comme un programme ordinaire, il ne se lit pas nécessairement de haut en bas. Tout dépend du contenu de votre fichier XML.

b. 2nde étape : corps du document

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/">
        <html>
            <head><title>Curriculum vitae</title></head>
            <body>
                  <xsl:apply-templates select="MINICV"/>
            </body>
        </html>
    </xsl:template>

    <xsl:template match="MINICV">
        <div id="wrapper">
            <xsl:apply-templates select="PERSONNE" />
            ...
        </div>
    </xsl:template>

    <xsl:template match="PERSONNE">
        ...
    </xsl:template>
</xsl:stylesheet>
```

Remarques:

1. On a rajouté la balise <xsl:apply-templates> au milieu des balises *body* pour indiquer au parser qu'à cet endroit on souhaite appliquer les transformations de la balise *MINICV*.
2. La balise *MINICV* est très similaire à la précédente, on décrit le début du tableau, on demande la transformation des balises *PERSONNE* puis on décrit la fin du tableau.
3. La balise *PERSONNE* est très simple pour le moment.

c. Exercice : Création des feuilles XSL

Énoncé : Ajouter dans un fichier xsl les instructions nécessaires à la prise en charge de l'ensemble des balises.

Démarche :

1. Le fichier LANCEUR et OUTILS vous est donné.
2. Pour chaque balise décrite dans le XSD, écrivez le comportement souhaité.

Remarques:

1. Pour chacune des balises PERSONNE on veut que le contenu soit entre les balises HTML . On utilise la balise XSL <*xsl:value-of select="@..."*> pour afficher la valeur de ses attributs.

2. Pour chacune des balises ADRESSE on veut que le contenu soit entre les balises HTML . On utilise la balise XSL <*xsl:value-of*> pour afficher la valeur de ADRESSE

D. Mise en place d'une charte graphique

a. Procédure de mise en place

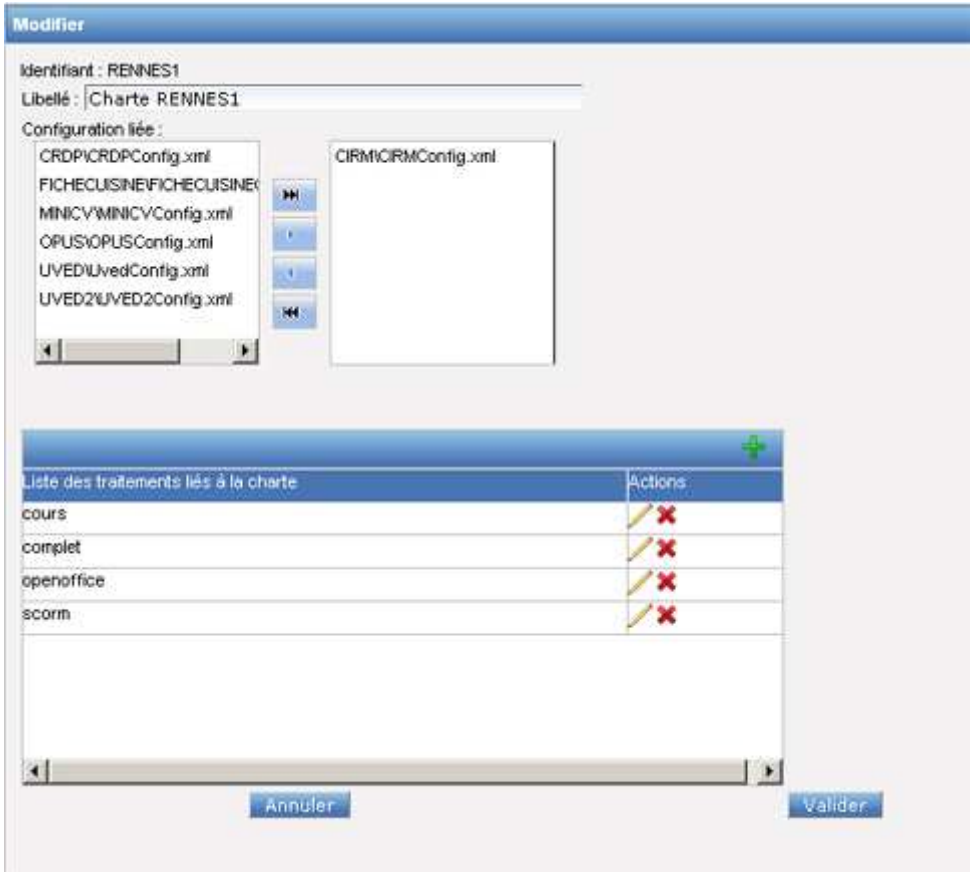
Pour mettre en place une nouvelle charte graphique, il est nécessaire de créer une archive zippée comprenant tous les éléments de la charte graphique, xsl, css, javascript, images, ... ; ces éléments étant souvent très nombreux le désarchivage après dépôt est le plus adapté.

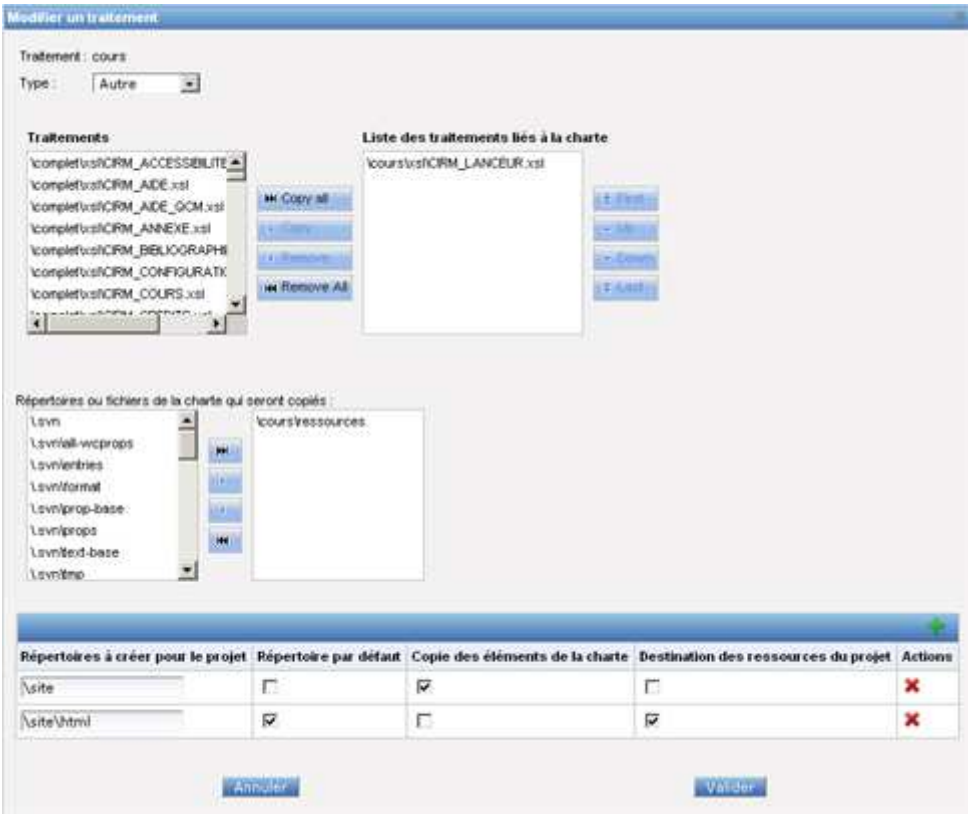
Il est possible après le dépôt de venir ajouter, modifier ou retirer des éléments. Dans notre cas, il suffit de zipper le répertoire contenant les répertoires xsl et «ressources» et de les organisés par traitement, puisque tous les éléments de charte y ont été centralisés. Il faut ensuite se connecter à ChainEdit pour mettre en place une nouvelle charte dans l'interface d'administration.

Il faut aller dans l'écran d' « Administration », sélectionner la « Gestion des chartes » graphiques et cliquer sur « Ajouter ».

Puis donner un identifiant à la charte et un label. Puis sélectionner le fichier Zip contenant les éléments nécessaires (XSL, CSS, Javascripts, ..).

Une fois cette opération effectuée, il faut aller modifier la charte nouvellement créée pour indiquer les traitements qui seront faits (appel XSL et copies de répertoires). Si votre archive contenait un fichier « charte.xml », la configuration décrite dans ce fichier sera prise en compte.


Modifier
Identifiant : RENNES1
Libellé : Charte RENNES1
Configuration liée :
CRDP\CRDPConfig.xml
FICHECUISINE\FICHECUISINE
MINICV\MINICVConfig.xml
OPUS\OPUSConfig.xml
UVED\UvedConfig.xml
UVED2\UVED2Config.xml
CIRM\CIRMConfig.xml
Liste des traitements liés à la charte
Actions
cours
complet
openoffice
scorm
Annuler
Valider



Pour chaque traitement il faut indiquer :

- les XSL à appeler
- les fichiers à copier
- les répertoires à créer et s'ils doivent contenir la charte et/ou les ressources de l'utilisateur (images, vidéos, ..)

b. Exercice : Dépôt de la charte graphique dans ChainEdit

Enoncé : Déposez la charte créée dans ChainEdit.

Démarche :

5. Allez dans l'écran des chartes
6. Ajoutez une charte
7. Paramétrez la charte
8. Ajoutez le fichier Zip
9. Complétez les informations

IV. Gestion des imports

Le premier écran de gestion des imports affiche la liste des imports possibles dans ChainEdit.

A partir de cet écran vous pouvez *ajouter*, *modifier* ou *supprimer* un import. Pour pouvoir modifier ou supprimer un import vous devez au préalable en sélectionner un dans la liste puis cliquer sur le bouton associé à l'opération que vous souhaitez effectuer.

Un import dans ChainEdit est un ensemble de traitements qui sont exécutés pour récupérer dans une configuration souhaitée, définie dans ChainEdit, (schéma cible) un projet provenant d'une configuration différente (schéma d'origine).

Liste des imports

1. Ajouter un import

Ajouter un import consiste à lui attribuer un identifiant unique et un libellé d'affichage.

Il faut ensuite indiquer un fichier zip qui contient généralement :
- un répertoire avec les fichiers de traitement nécessaires.
- un répertoire avec les fichiers XSL de transformation

Après la validation de l'import, un répertoire sera créé et le fichier zip sera décompressé dans ce répertoire (éviter de mettre un répertoire racine dans ce fichier).

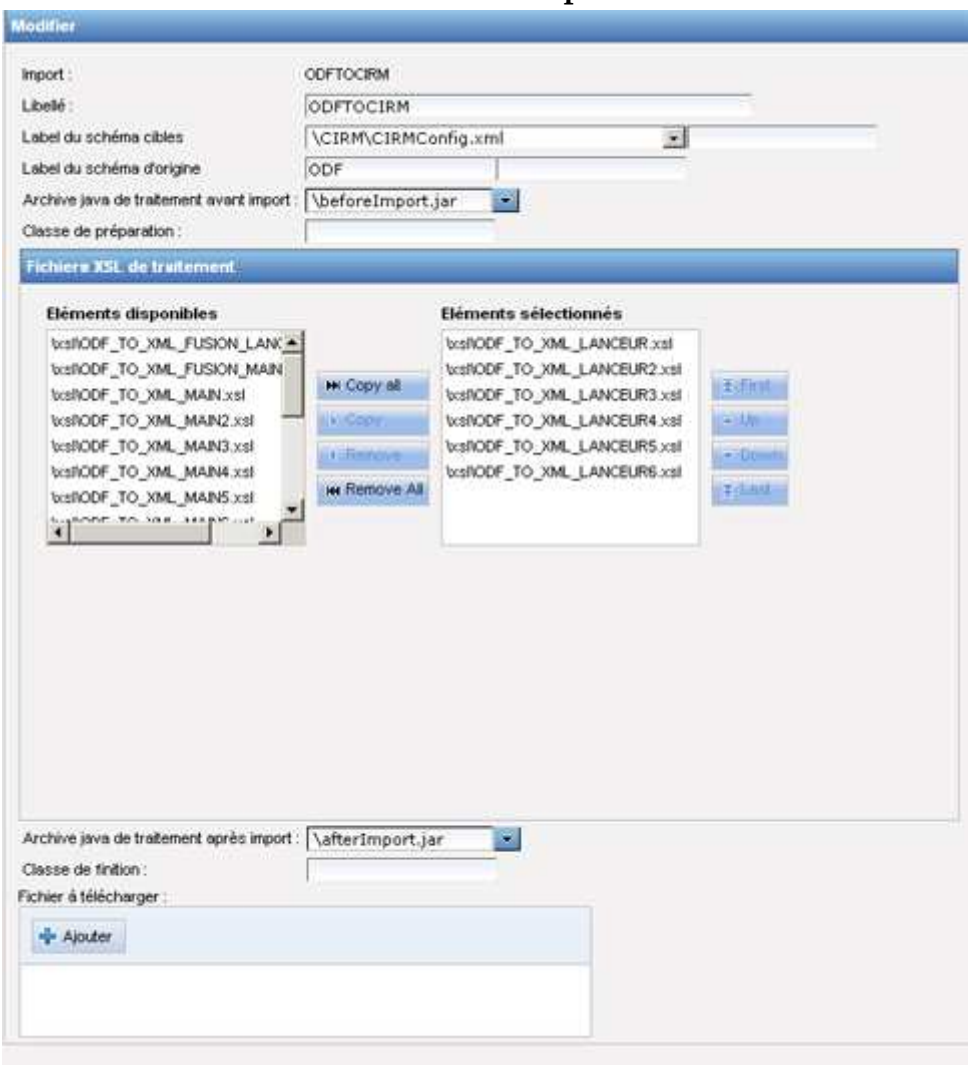
Il faut ensuite aller modifier l'import pour pouvoir définir les traitements qui seront exécutés.

Ajouter un import

2. Modifier un import

Après avoir ajouté un import, il faut le modifier pour préciser les traitements qui seront associés à cet import.
Modifier un import consiste donc à *ajouter*, *modifier* ou *supprimer* des traitements XSL ou des traitements JAVA qui serviront à préparer le bon déroulement de la transformation d'une part, et à ranger et "nettoyer" les répertoires après la transformation d'autre part.
Il faut aussi y définir :
- l'identifiant et le label du schéma d'origine du projet (quelle configuration suit le fichier que l'on souhaite importer),
- l'identifiant et le label du schéma cible du projet (quelle configuration suivra le résultat de l'import),
Il est également possible de *gérer les répertoires* associés à l'import pour le mettre à jour ou le modifier.

Modifier un import



3. Supprimer un import

Pour supprimer un import, il faut au préalable l'avoir sélectionner dans la liste. Vous devrez alors confirmer la suppression.

Supprimer un import

V. Gestion des référencements

Le premier écran de gestion des référencements affiche la liste des référencements disponibles dans ChainEdit.

A partir de cet écran vous pouvez *ajouter*, *modifier* ou *supprimer* un référencement. Pour pouvoir modifier ou supprimer un référencement vous devez au préalable en sélectionner un dans la liste puis cliquer sur le bouton associé à l'opération que vous souhaitez effectuer.

Référencer un contenu, c'est renseigner une fiche de métadonnées. Par exemple, pour un document pédagogique numérique, les métadonnées relatives à l'établissement d'origine, aux auteur(s), à la discipline, à l'usage pédagogique, (...), doivent figurer sur une fiche qui respecte un format xml donné. Cette fiche doit également renseignée l'url à laquelle le contenu est accessible.

Un référencement dans chainedit se compose de plusieurs étapes :
Etape 1 : création de la fiche xml contenant les métadonnées propres au module
Etape 2 : copie de la génération (zip, pdf, site) du module dans un répertoire de destination et construction de l'url d'accès à cette copie.
Etape 3 : Connexion à l'application de référencement. Cette application doit exposer un web service.
Etape 4 : Création d'une nouvelle entrée dans cette application de référencement. Récupération de l'url d'accès de notre fiche de métadonnées dans la partie éditeur de l'application de référencement.

Liste des référencements

1. Ajouter un référencement

Ajouter un référencement consiste à lui attribuer un identifiant unique et un libellé d'affichage.

Il faut ensuite indiquer un fichier zip qui contient généralement :
- un répertoire avec les fichiers XSL de transformation. Ces xsl créeront la fiche xmlde métadonnées.

Après la validation d'un référencement , un répertoire sera créé et le fichier zip sera décompressé dans ce répertoire (éviter de mettre un répertoire racine dans ce fichier).

Il faut ensuite aller modifier le référencement pour pouvoir définir tous les éléments nécessaires à chaque étape.

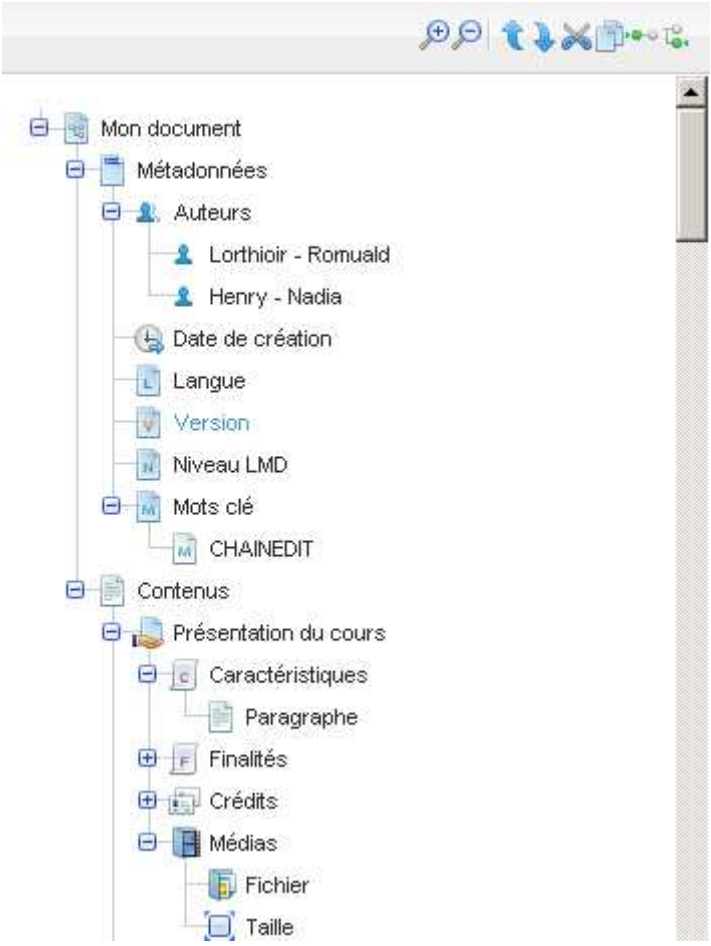
Ajouter un référencement

VI. Eléments avancés

1. Paramétrer l'interface de ChainEdit

A. Paramétrer les icônes dans l'arbre d'édition

Chaque icône présent devant les différents éléments présentés lors de la saisie peut être paramétrés.



Arbre de saisie

Pour effectuer ce paramétrage, il est nécessaire de faire 2 actions:

- mettre les icones nécessaires dans le répertoire configs/nom_de_la_config/ressources
- indiquer dans le fichier de configuration correspondant les icones à utiliser

a. Icônes dans configs/nom de la config/ressources

Il faut mettre les icônes dans le répertoire /ressources

b. Fichier de configuration

Il faut ensuite modifier le fichier de configuration associé au modèle. Par exemple dans OPUSConfig.xml, pour indiquer les icônes associé au noeud OPUS :

```
<BALISE nom="OPUS" titre="Module OPUS " type="typevide">
      <ICONES chemin="OPUS/">
            <ICONEOPEN valeur="tree-icons/default/module.png" />
            <ICONECLOSED valeur="tree-icons/default/module.png" />
            <ICONEERROROPEN valeur="tree-icons/error/module.png" />
            <ICONEERRORCLOSED valeur="tree-icons/error/module.png" />
      </ICONES>
      <CHILDS>
            <CHILD nom="PARAMETRAGES" />
            <CHILD nom="METADONNEES" />
            <CHILD nom="PRESENTATIONS" />
            <CHILD nom="COURS" />
            <CHILD nom="TRAVAUX-PRATIQUES" />
            <CHILD nom="ETUDES-CAS" />
            <CHILD nom="EXERCICES" />
            <CHILD nom="PLANNINGS" />
            <CHILD nom="GLOSSAIRE" />
            <CHILD nom="BIBLIOGRAPHIE" />
            <CHILD nom="WEBOGRAPHIE" />
            <CHILD nom="MEDIATHEQUE" />
            <CHILD nom="ANNEXES" />
            <CHILD nom="UTILITAIRES" />
      </CHILDS>
</BALISE>
```

B. Langues

L'internationalisation est native dans *esup-commons et par extension dans ChainEdit*. L'intérêt n'est pas seulement de fournir une application en plusieurs langages ; l'externalisation de toutes les chaînes de caractères, et la possibilité d'utiliser simultanément plusieurs fichiers de chaînes (les bundles) permet de simplifier la personnalisation des applications par les administrateurs.

Cette partie de documentation est issue de la documentation de la formation Esup-commons qui est disponible sur ce site http://perso.univ-rennes1.fr/pascal.aubry/formation/esup-commons/. Elle a néanmoins été simplifiée.

a. Configuration

L'internationalisation est définie dans le fichier de configuration **`/properties/i18n/i18n.xml`**. On y trouvera par exemple :

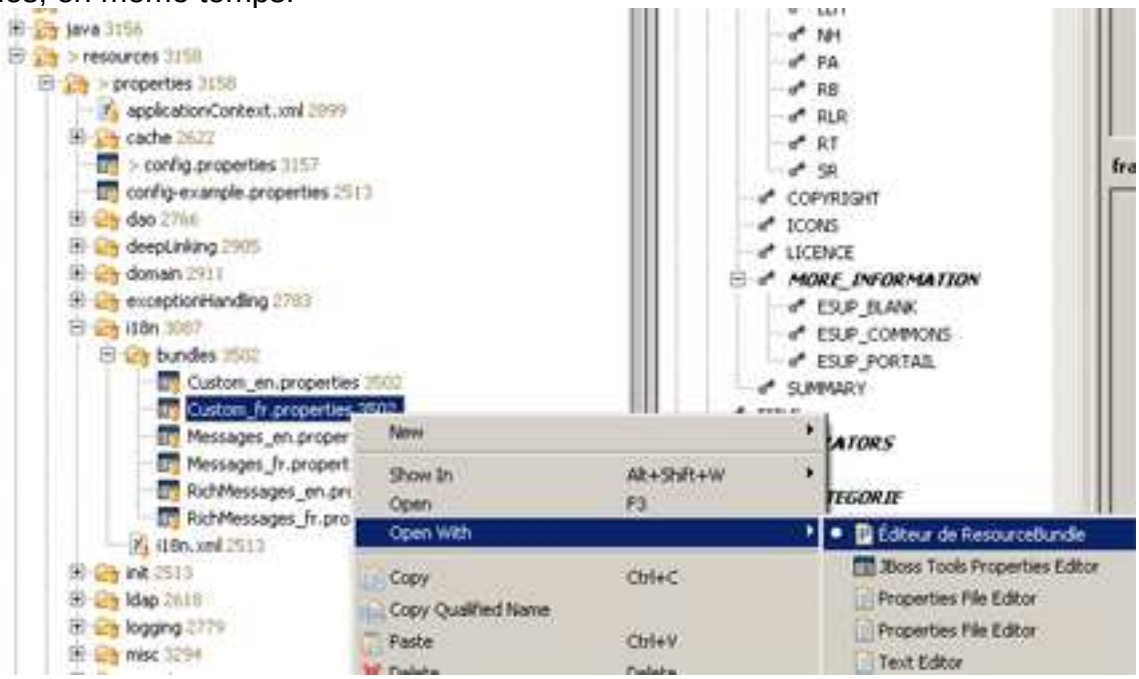
```
<bean
id="i18nService"
class="[...].commons.services.i18n.BundlesCachingI18nServiceImpl"
>
<property name="bundleBasenames">
<list>
<value>properties/i18n/bundles/Commons</value>
<value>properties/i18n/bundles/Messages</value>
<value>properties/i18n/bundles/Custom</value>
</list>
</property>
</bean>
```

La propriété **`bundleBasenames`** donne la liste des fichiers de messages à utiliser. Chaque valeur de la liste correspond à la racine du chemin du fichier de message dans le *classpath*. Par exemple, on cherchera pour la valeur **`Custom`** les fichiers **`Custom_<locale>.properties`** ou à défaut **`Custom.properties`**. **`<locale>`** représente la langue sélectionnée par exemple fr pour français.

Par convention, tous les fichiers de messages sont dans le répertoire **`/properties/i18n/bundles`**.

b. Modification des fichiers de messages

Pour modifier les fichiers de messages il est recommandé d'utiliser *Ressource Bundle Editor* (*RBE*). Il permet notamment d'éditer plusieurs fichiers, correspondants à plusieurs langues, en même temps.



Les messages enregistrés dans ces fichiers peuvent contenir des paramètres qui seront dynamiquement renseignés lors de l'exécution. Ces paramètres apparaissent dans les messages sous cette forme : **`{n}`** avec **`n`** commençant à **`0`**.

Exemple : **`L''utilisateur {0} est maintenant gestionnaire du service {1}.`**

Notes :

- On note ci-dessus l'échappement des apostrophes lorsque la chaîne contient un ou plusieurs paramètres.
- Il est important que *RBE* soit configuré de la même manière par tous les développeurs d'un même projet.

c. Ajout d'un langage

L'ajout d'un langage se fait dans le fichier de configuration **`/properties/jsf/application.xml`**. Il suffit ensuite d'écrire le *bundle* correspondant.

d. Fichiers disponibles

Dans ChainEdit, 3 fichiers sont disponibles par langue. Par défaut, ChainEdit dispose du français et de l'anglais. La langue du navigateur est détectée automatiquement. Par exemple pour le français :

- Commons_fr contient les messages de Esup-commons.
- Custom_fr contient les messages spécifiques de l'application.
- Messages_fr contient les messages principaux et les libellées commun des boutons de l'interface.

Tous les messages et tous les libellés affichés dans ChainEdit sont paramétrés dans ces fichiers. Si vous détectez une erreur, un message imprécis, une faute d'orthographe vous pourrez la corriger ici. Il est également souhaitable de le signaler à l'équipe de développement pour que la correction soit disponible pour tous.