3. Migration d'une charte

3.1. Encodage

Dans chaque feuille XSL, il faut remplacer

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

par

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

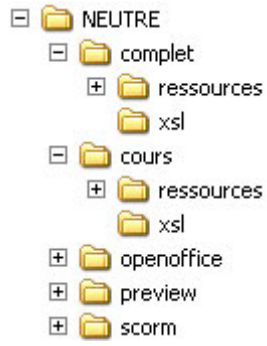
Une 2ème charte (génération en html) avait un fichier charte.xml ressemblant à ce qui suit :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<CHARTE label="Charte NEUTRE" id="NEUTRE">
    <TRAITEMENTS type="autre" nature="htm">
        <FICHIERXSL ordre="1" chemin="\xsl\CIRM_LANCEUR.xsl"/>
        <REPERTOIRECHARTE chemin="\ressources"/>
        <REPERTOIRECIBLE ressource="non" defaut="non" chemin="/site"
charte="oui"/>
        <REPERTOIRECIBLE ressource="oui" defaut="oui" chemin="/site/html"
charte="non"/>
    </TRAITEMENTS>
</CHARTE>
```

Dorénavant, une charte peut correspondre à plusieurs traitements (pdf, html…). Et le fichier charte.xml contient tous les traitements, comme vous pouvez le voir dans l'exemple ci-après :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CHARTE id="NEUTRE" label="Charte NEUTRE">
    <CHARTECONFIG id="CIRM\CIRMConfig.xml"/>
    <TRAITEMENTS nature="complet" type="autre">
        <FICHIERXSL ordre="0" chemin="\complet\xsl\CIRM_LANCEUR.xsl"/>
        <REPERTOIRECHARTE chemin="\complet\ressources"/>
        <REPERTOIRECIBLE chemin="/site" defaut="non" charte="oui"
ressource="non"/>
        <REPERTOIRECIBLE chemin="/site/html" defaut="oui" charte="non"
ressource="oui"/>
    </TRAITEMENTS>
    <TRAITEMENTS nature="cours" type="autre">
        <FICHIERXSL ordre="0" chemin="\cours\xsl\CIRM_LANCEUR.xsl"/>
        <REPERTOIRECHARTE chemin="\cours\ressources"/>
        <REPERTOIRECIBLE chemin="/site" defaut="non" charte="oui"
ressource="non"/>
        <REPERTOIRECIBLE chemin="/site/html" defaut="oui" charte="non"
ressource="oui"/>
    </TRAITEMENTS>
    <TRAITEMENTS nature="scorm" type="autre">
        <FICHIERXSL ordre="0" chemin="\scorm\xsl\CIRM_LANCEUR.xsl"/>
        <REPERTOIRECHARTE chemin="\scorm\ressources"/>
        <REPERTOIRECIBLE chemin="/site" defaut="non" charte="oui"
ressource="non"/>
        <REPERTOIRECIBLE chemin="/site/html" defaut="oui" charte="non"
ressource="oui"/>
    </TRAITEMENTS>
    <TRAITEMENTS nature="openoffice" type="openOffice">
        <FICHIERXSL ordre="0" chemin="\openoffice\xsl\CIRM_LANCEUR.xsl"/>
        <REPERTOIRECHARTE chemin="\openoffice\META-INF"/>
        <REPERTOIRECHARTE chemin="\openoffice\Pictures"/>
        <REPERTOIRECHARTE chemin="\openoffice\mimetype"/>
        <REPERTOIRECHARTE chemin="\openoffice\settings.xml"/>
        <REPERTOIRECHARTE chemin="\openoffice\styles.xml"/>
        <REPERTOIRECIBLE chemin="\oo" defaut="oui" charte="oui"
ressource="non"/>
    </TRAITEMENTS>
    <TRAITEMENTS nature="preview" type="preview">
        <FICHIERXSL ordre="0" chemin="\preview\xsl\CIRM_LANCEUR.xsl"/>
        <REPERTOIRECHARTE chemin="\preview\ressources"/>
        <REPERTOIRECIBLE chemin="\site" defaut="non" charte="oui"
ressource="non"/>
        <REPERTOIRECIBLE chemin="\site\html" defaut="oui" charte="non"
ressource="oui"/>
    </TRAITEMENTS>
</CHARTE>
```

De ce fait, l'arborescence est modifiée. On a toujours un répertoire portant le nom de la charte, mais ce qui est nécessaire pour générer n'est plus directement sous ce répertoire mais dans un sous-répertoire portant le nom du traitement. L'arborescence d'une charte se présente comme ceci :



Attention : Le nom du répertoire doit être identique à la valeur de l'attribut “nature” d'un TRAITEMENTS dans charte.xml

Il faut corriger la déclaration de la variable $cheminSite dans les fichiers XSL concernés. Ainsi, dans les fichiers où la variable $cheminSite est définie, il faut remplacer :

```
<xsl:variable name="cheminSite">
      <xsl:value-of select="concat($chainEditPath, '../site/html/')" />
</xsl:variable>
```

par :

```
<xsl:variable name="cheminSite">
      <xsl:value-of select="concat($chainEditPath, 'site/html/')" />
</xsl:variable>
```

3.3. Traitement Preview

Il y a plusieurs ajouts à faire pour qu'une charte puisse permettre la prévisualisation d'un cours développé dans ChainEdit.

Dans un fichier javascript tout d'abord, il faut ajouter le code suivant :

```
CIRM.module.edition = {

pageURL: './projetEdit.faces?args=node=',

message: 'Impossible d\'\u00E9diter ce contenu dans le context actuel: ChainEdit
est introuvable !',

initialize: function( event )
{
  $$( 'a[class=edition]' ).each( function( link )
  {
      var nodeID = link.get( 'id' );
      link.onclick = function()
      {
         if( $defined( window.opener )
             && $defined( window.opener.ChainEdit )
             && $defined( nodeID ) )
         {
            var win = window.opener;
            win.location.href = CIRM.module.edition.pageURL + nodeID;
            win.focus();
         }
         else
         {
            alert( CIRM.module.edition.message );
         }
         return( false );
      };
  } );
}


window.addEvent( 'domready', CIRM.module.edition.initialize );
```

Ensuite, dans le fichier *XXX_TEMPLATE.XSL*, il faut créer le *“template”* suivante :

```
<xsl:template name="edition">
    <xsl:param name="id" select="''" /> <!-- required -->
    <xsl:param name="display" select="'inline'" /> <!-- optional -->
    <xsl:if test="string-length( $id ) > 0">
        <xsl:choose>
            <xsl:when test="$display != 'inline'">
                <xsl:element name="p">
                    <xsl:element name="a">
                        <xsl:attribute name="id">
                            <xsl:value-of select="$id" />
                        </xsl:attribute>
                        <xsl:attribute name="class">
                            <xsl:value-of select="'edition'" />
                        </xsl:attribute>
                        <xsl:attribute name="href">
                            <xsl:value-of select="'#'" />
                        </xsl:attribute>
                        <xsl:attribute name="title">
                            <xsl:value-of select="'Editer cet élément'" />
                        </xsl:attribute>
                        <xsl:element name="span">
                            <xsl:text>[éditer]</xsl:text>
                        </xsl:element>
                    </xsl:element>
                </xsl:element>
            </xsl:when>
            <xsl:otherwise>
                <xsl:element name="a">
                    <xsl:attribute name="id">
                            <xsl:value-of select="$id" />
                        </xsl:attribute>
                        <xsl:attribute name="class">
                            <xsl:value-of select="'edition'" />
                        </xsl:attribute>
                        <xsl:attribute name="href">
                            <xsl:value-of select="'#'" />
                        </xsl:attribute>
                        <xsl:attribute name="title">
                            <xsl:value-of select="'Editer cet élément'" />
                        </xsl:attribute>
                    <xsl:element name="span">
                        <xsl:text>[éditer]</xsl:text>
                    </xsl:element>
                </xsl:element>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:if>
</xsl:template>
```

Ce *“template”* sera ensuite appelé pour chaque élément que l'on souhaite pouvoir modifier depuis l'aperçu :

```
<!-- edition -->
<xsl:call-template name="edition">
      <xsl:with-param name="id">
            <xsl:value-of select="name( current() )" />
            <xsl:text>_</xsl:text>
            <xsl:number level="any" />
      </xsl:with-param>
      <xsl:with-param name="display" select="'inline'" />
</xsl:call-template>
<!-- /edition -->
```

Pour le paramètre *“display”*, on peut choisir de passer les valeurs '*inline*' ou '*block*', selon le type d'élément.

Puis, dans le fichier module.css, seront définis les styles utilisés :

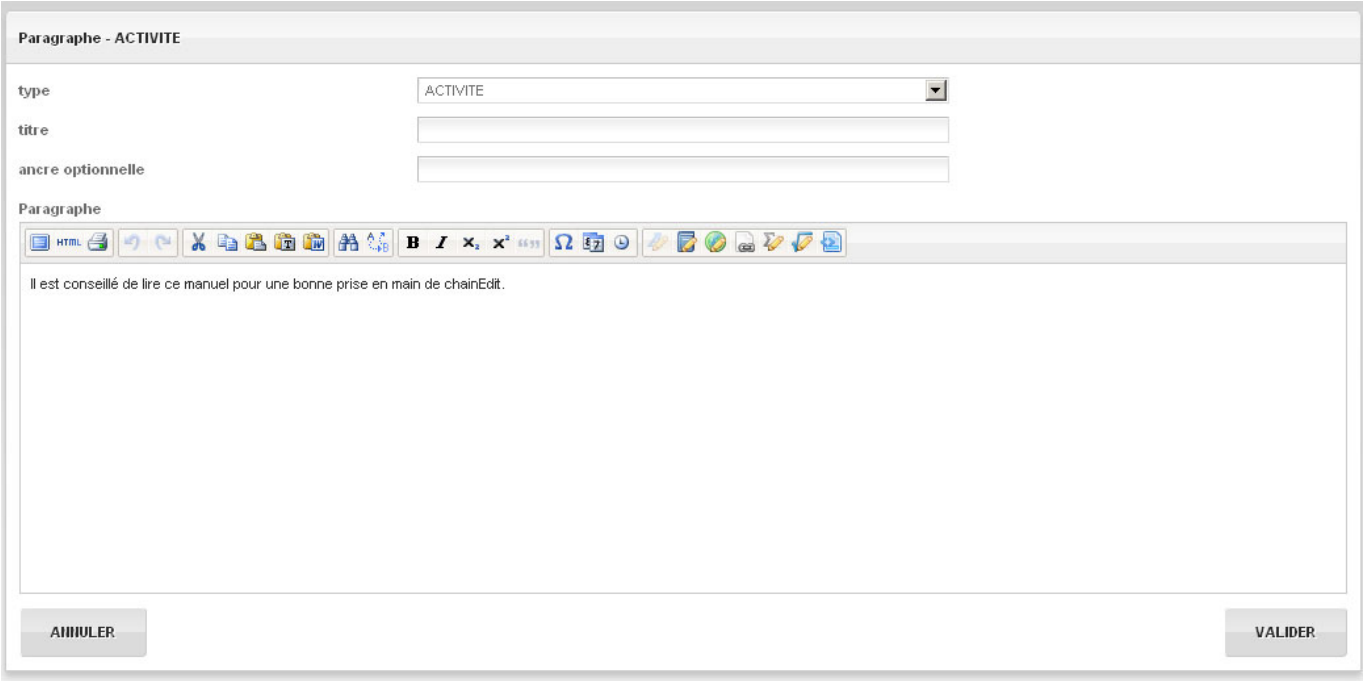
```
a.edition
{
      display:inline-block;
      width:16px;
      height:16px;
      vertical-align:middle;
      text-indent:-99999px;
      border:0;
      background:transparent url(../assets/navigations/edit.png) no-repeat 0 0;
      overflow:hidden;
      text-decoration:none;
}

a.edition:focus,
a.edition:hover
{
      text-decoration:none;
}

a.edition span
{
      position:absolute;
      top:-2000px;
      left:-2000px;
}
```

Il faut également ajouter l'icône cité ci-dessus `../assets/navigations/edit.png` au bon endroit dans les ressources de la charte Preview.

Ainsi, après avoir généré l'aperçu du module, il est possible de revenir dans la saisie, sur un élément précis, en cliquant sur l'icône figurant à la fin de cet élément dans l'aperçu :



3.4. Intégration

Pour intégrer la charte dans le nouveau ChainEdit, il faut aller dans la partie Administration / Gestion des configurations et cliquer sur l'icône pour gérer les fichiers. En se positionnant sur le dossier à la racine, portant le nom de la charte, il est possible de télécharger le fichier compressé (attention à l'outil utilisé pour la compression) contenant tous les fichiers de la charte, puis de décompresser ce dernier.